The Dissertation Committee for Dohyung Park
certifies that this is the approved version of the following dissertation:

Efficient Non-convex Algorithms for Large-scale Learning Problems

Committee:

Sujay Sanghavi, Supervisor

Constantine Caramanis, Co-supervisor

Joydeep Ghosh

Alexandros G. Dimakis

Eric Price

Efficient Non-convex Algorithms for Large-scale Learning Problems

by

Dohyung Park, B.S.; M.S.

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

December 2016

Dedicated to my family.

Acknowledgments

I was extremely fortunate to have been a graduate student of Prof. Constantine Caramanis and Prof. Sujay Sanghavi. Since the time we first met on a hot summer day in Austin, they have always inspired me with deep insights and penetrating questions and comments. I am profoundly grateful to them for their continuous guidance and support, especially during my final year. This dissertation would not have been made without their advice and encouragement.

I had the great fortune to learn from Prof. Sae-Young Chung who was my master's degree advisor. Before I met him, I had not been able to think scientifically and independently. I am also grateful to Prof. Sanjay Shakkottai, Prof. Alex Dimakis, and Prof. Inderjit Dhillon for offering essential courses and having passionate discussions with me. Though I did not have many opportunities to collaborate, a glimpse of their enthusiasm has been such a great influence to me. I would also like to thank Prof. Joydeep Ghosh and Prof. Eric Price for being in my committee and providing valuable comments. I was also fortunate to collaborate with many brilliant peers, Joe Neeman, Anastasios Kyrillidis, Srinadh Bhojanapalli, Yudong Chen, and Xinyang Yi. It was my privilege to work with each of them.

I would like to thank my friends in WNCG - Sid, Praneeth, Srinadh, Avik, Namyoon, Junil, Muryong, Jeonghun, Chang-sik, Sungwoo, Yongseok, Yubin, Taewan, Jinseok, Tasos, Megas, Karthik, Yudong, Xinyang, Vatsal, Shanshan and many others who shared unforgettable moments with me. And

I am very grateful to the people in the Lord's Church of Austin who have prayed for my family and encouraged us to overcome our life struggles.

Last but not least, my gratitute goes to my family. I thank my parents and brother for their unconditional love and support from Korea. Sean has been my energy source since his birth. Sukyung has been and always will be my reason to live, and I am indebted to her for life. Finally I thank God who has been always with us in this journey.

Dohyung Park

Austin, August 2016

Efficient Non-convex Algorithms for Large-scale Learning Problems

Publication No. _____

Dohyung Park, Ph.D.
The University of Texas at Austin, 2016

Supervisors: Sujay Sanghavi
Constantine Caramanis

The emergence of modern large-scale datasets has led to a huge interest in the problem of learning hidden complex structures. Not only can models from such structures fit the datasets, they also have good generalization performance in the regime where the number of samples are limited compared to the dimensionality. However, one of the main issues is finding computationally efficient algorithms to learn the models. While convex relaxation provides polynomial-time algorithms with strong theoretical guarantees, there are demands for even faster algorithms with competitive performances, due to the large volume of the practical datasets.

In this dissertation, we consider three types of algorithms, *greedy methods*, *alternating minimization*, and *non-convex gradient descent*, that have been key non-convex approaches to tackle the large-scale learning problems. For each theme, we focus on a specific problem and design an algorithm based on the designing ideas. We begin with the problem of subspace clustering, where one needs to learn underlying unions of subspaces from a set of data points

around the subspaces. We develop two greedy algorithms that can perfectly cluster the points and recover the subspaces. The next problem of interest is collaborative ranking, where underlying low-rank preference matrices are to be learned from pairwise comparisons of the entries. We present an alternating minimization based algorithm. Finally, we develop a non-convex gradient descent algorithm for general low-rank matrix optimization problems. All of these algorithms exhibit low computational complexities as well as competitive statistical performances, which make them scalable and suitable for a variety of practical applications of the problems. Analysis of the algorithms provides theoretical guarantees of their performances.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Statistical learning considers the problem of finding an underlying model from a set of samples drawn from the model. It has been studied for a variety of purposes such as prediction of future samples and simpler representation of the given samples. For example, linear or logistic regression is used to predict output variables for input variables which have not been observed. Principal component analysis (PCA) finds low-dimensional linear representations of datasets. The solutions to these problems are analytically well-known, and classical methods such as convex optimization and singular value decomposition can provide these solutions.

The recent explosion in the size and the dimensionality of modern datasets has led to considering more complicated statistical models to learn. The models with sparse and low-rank structures have arisen in the field where the number of samples is limited relative to their high dimensionality. Heterogeneity in the datasets has led to the need of representation by multiple low-dimensional subspaces rather than a single subspace using PCA. For these models, the maximum likelihood estimations are non-convex optimization problems, which are generically NP-hard. While many classical problems were solved by their own analytic or algorithmic solutions in virtue of the simple structures of the models, it is diffcult to design algorithms solving such large-scale problems.

A tremendous number of different algorithms have been proposed to tackle these problems. Convex relaxation is one of the most attractive approaches in the sense that the exact learning of the model is guaranteed with strong statistical complexity by polynomial-time algorithms. However, the recent unprecedented size of data led to a continuing demand for even faster algorithms. Toward this direction, several related themes arise in the literature.

Greedy methods: In many problems with parsimonius representation, such as sparse regression [150, 182, 39], low-rank estimation [104], and learning sparse graphical models [83, 126, 133], algorithms that incrementally select elements of the models in a greedy fashion are shown to be promising alternatives with comparable statistical guarantees and lower computational complexities.

Alternating minimization: For several learning problems, *e.g.*, matrix factorization, phase retrieval, and mixed regression, the maximum likelihood estimations are non-convex due to the non-linear combination of two underlying parameters. It is shown in [82, 128, 127, 175] that alternately solving for one parameter while fixing the other provides desirable statistical complexity. As opposed to convex relaxation, this class of algorithms do not expand the variable space and hence save computational costs.

Non-convex gradient descent: Very recently, algorithms with simple gradient descent on non-convex formulations are proved to be efficient in low-rank matrix estimation problems [90, 91, 29, 38, 152, 183, 184, 185, 19]. These fomulations are based on the reparametrization of low-rank matrices using two factors, and the reduced size of the variable space results in a significant computational gain.

In this dissertation, we design algorithms in these themes, tailored to

specific problems. We present greedy algorithms for the problem of subspace clustering, two algorithms based on convex relaxation and alternating minimization for collabortaive ranking, and a non-convex gradient descent algorithm for general low-rank matrix optimization problems. These algorithms have low computational complexity and hence scalability in the large-scale setting. Analysis of the algorithms is another main part of this dissertation, where we provide the performance guarantees.

1.1 Main problems and contributions

In this section, we introduce the problems that we consider, the designed algorithms, and their contributions.

Greedy methods for subspace clustering. We consider the setting where sample points in a high-dimensional ambient space are lying on or near unions of low-dimensional linear subspaces. This model naturally arises when a heterogeneous dataset contains sample points with latent labels, and each subset of the points with the same label can be approximated by a low-dimensional subspace. This problem is hard when the number of points are limited. In this regime, general clustering algorithms based on Euclidean distances do not perform properly since for each point there can be many points on different subspaces closer to the most of the points on the same subspace. To learn this model, we propose new greedy algorithms to cluster the points with respect to the hidden label. Then one can learn the subspaces by applying principal component analysis (PCA) to each subset of points. We prove that our algorithms has significantly lower computational costs and also strong theoretical guarantees, at least comparable to the state-of-the-art convex optimization

based algorithms (See Table 2.1 for comparison).

Alternating minimization for collaborative ranking. Collaborative ranking is a problem of learning preference orderings of multiple items for multiple users. The samples are given as partial orderings or more simply pairwise comparisons. As observed in the collaborative filtering literature, a low-rank matrix model is a reasonable heuristic for estimating preferences of multiple users. This model not only captures the similarity between the preferences of the users, it can also be learned even when the samples from each user are insufficient for an independent ranking model to be learned. In order to solve this problem, we first establish a convex relaxation based algorithm, which minimizes the empirical risk function under a nuclear norm constraint. It is shown that this method is nearly optimal in the sense that the excess risk bound matches the lower bound up to a logarithmic factor. Then we propose a non-convex algorithm which is based on alternating minimization. We demonstrate that on real-world datasets with numerical relevance scores our algorithm performs even better than state-of-the-art collaborative ranking algorithms. We also show scalability of our algorithm on multi-core environments.

Non-convex gradient descent for low-rank matrix optimization. We consider a more general setting where one minimizes a general convex function f over low-rank matrices. This is a general formulation in many problems to learn low-rank matrix models. For example, in the matrix sensing problem, f is the mean squared error of an estimate from the linear measurements of a true low-rank matrix. For the logistic PCA problem, f is the log likelihood of an estimate with respect to the binary observation of an underlying matrix.

We study a general gradient descent method over the factor space, which is an extensively used heuristic. Since a low-rank matrix can be re-parametrized by two factors ($X = UV^\top$ where $X \in \mathbb{R}^{m\times n}, U \in \mathbb{R}^{m\times r}, V \in \mathbb{R}^{n\times r}, r \ll m, n$), the first-order method can be applied to the two factors. This method has advantages in computational cost over existing methods which operate over the original X space of much more parameters to optimize or require low-rank projection using singular value decomposition (SVD) at each iteration.

For this non-convex optimization problem, we propose an algorithm which operates gradient descent with a carefully selected step size. We also prove local convergence guarantees for two classes of functions, smooth and convex f, and strongly convex and smooth f. Our results are analogous to the standard convergence rates for convex optimization. In particular, for smooth and convex f, we provide a $O(1/t)$ convergence guarantee, while we prove that a linear convergence rate can be achieved for smooth and strongly convex f.

1.2 Organization

In each chapter of the remainder of this dissertation, our designed algorithm is presented. Chapter 2 presents greedy subspace clustering algorithms for learning unions of spaces. Chapter 3 shows another work in which two algorithms based on convex relaxation and alternating minimization are designed for the problem of collaborative ranking. Chapter 4 presents our non-convex gradient descent algorithm for low-rank matrix estimation problems. We conclude by describing the main theme of this thesis in Chapter 5.

1.3 Notation

Sets and subspaces are denoted by calligraphic letters. Matrices and key parameters are denoted by letters in upper case, and vectors and scalars are denoted by letters in lower case. We will use the following notations.

X_{ij}	the element of matrix X in the ith row and jth column
$\lvert \cdot \rvert$	absolute value
$\lVert \cdot \rVert_*$	nuclear/trace norm (The sum of all singular values)
$\lVert \cdot \rVert_2$	ℓ_2 norm for vectors, operator norm for matrices
$\lVert \cdot \rVert_F$	Frobenius norm
$\mathbb{E}$	expectation
$\mathbb{P}_X$	probability distribution parametrized by X
$\mathcal{N}(\mu, \Sigma)$	multivariate Gaussian distribution with mean μ and covariance matrix Σ
$[n] \triangleq \{1, \ldots, n\}$	set of n indices
$\mathrm{span}\{\cdot\}$	subspace spanned by a set of vectors
$\mathrm{Proj}_{\mathcal{U}}\, y$	projection of y onto set $\mathcal{U}$
$\mathbb{I}\{\cdot\}$	indicator function
Π_n	set of all permutations of $[n]$
$\mathbb{R}^p$	p-dimensional Euclidean space
$\mathbb{S}^{p-1}$	unit sphere in $\mathbb{R}^p$
$\bigoplus$	direct sum
$\overset{d}{=}$	equality in distribution $A \overset{d}{=} B$ if $\Pr\{A \geq t\} = \Pr\{B \geq t\}$ for any $t \in \mathbb{R}$
$\overset{d}{\geq}, \overset{d}{\leq}$	stochastic domination in the corresponding directions $A \overset{d}{\geq} B$ if $\Pr\{A \geq t\} \geq \Pr\{B \geq t\}$ for any $t \in \mathbb{R}$
dim	dimension of a linear subspace

$\mathcal{B}_k(\cdot)$	Stiefel manifold (of orthonormal k-frames) in a linear subspace $\mathcal{B}_k(\mathcal{D}) \triangleq \{D \in \mathbb{R}^{p\times k} : D^\top D = I, Dx \in \mathcal{D},\ \forall x \in \mathbb{R}^k\}$
$\mathcal{B}(\mathcal{D})$	$\mathcal{B}_k(\mathcal{D})$ with $k = \dim \mathcal{D}$.
$\langle \cdot, \cdot \rangle$	inner product
$\sigma_i(\cdot)$	the ith largest singular value
∇	gradient operator
$I_{d\times d}$	$d \times d$ identity matrix
$D(\cdot\|\|\cdot)$	Kullback-Leibler divergence between two distributions

Chapter 2

Greedy Algorithms for Subspace Clustering

One of the main problems in high-dimensional data analysis is to find an underlying structure that approximates a given large set of data points. Principal component analysis (PCA) is one of the most fundamental approaches where one approximates the points by a low-dimensional linear subspace. While PCA performs properly when all of the points lie on or near a single subspace, those in many practical datasets lie on *unions of multiple subspaces*, where each subspace fits a subset of the unlabeled points. In this setting, one needs to jointly find the subspaces and the points corresponding to each. This problem of finding unions of subspaces is known as *Subspace Clustering*.

Due to its generality, there is a broad range of applications of subspace clustering, which includes the following.

- Motion segmentation [158] : Given a video sequence of multiple rigid-body motions, the point trajectories associated with each motion lie on a 4-dimensional subspace. Since the subspaces are different depending on the motion, we can segment the trajectories in terms of their motion by subspace clustering algorithms.

[1]This work has been published as Dohyung Park, Constantine Caramanis, and Sujay Sanghavi, "Greedy Subspace Clustering," in Proceedings of Advances in Neural Information Processing Systems (NIPS), 2014. My contributions are design of algorithms, statement and proofs of main results, and design and implementation of experiments.

- Face clustering [72] : Under varying illumination conditions, different images of a single face from a fixed view span a low-dimensional subspace. When the images of multiple faces are mixed in a dataset, one can cluster the images using subspace clustering algorithms.

- Gene expression analysis [95, 44] : DNA microarray data is often given in the form of matrices representing expression levels of multiple genes under multiple conditions. A subgroup of the genes with a common function provides varying expression levels for the related conditions, their expression profiles (the vectors of the entire expression levels correponding to the genes) lie on an axis-aligned subspace.

- Blind source separation [59] : Blind source separation is the problem that finds multiple audio sources from their mixed observations. While this problem is more related to sparse coding, it is shown in [59] that subspace clustering can also perform as well as sparse coding. In this particular problem, subspace clustering can be regarded as group-sparse coding, where each data point is represented by one of the dictionary groups, which is the basis of a subspace.

- Hybrid system identification [157, 76] : When the discrete-time state of a linear hybrid system evolves under one of the multiple parameters at each time, one needs to find all of the parameters. This problem can be reduced to finding multiple subspaces from a given points lying on the subspaces.

There is now a sizable literature on empirical methods for this particular problem and some statistical analysis as well. Many recently proposed methods, which perform remarkably well and have theoretical guarantees on their performances, can be characterized as involving two steps: *(a)* finding

a "neighborhood" for each data point, and *(b)* finding the subspaces and/or clustering the points given these neighborhoods. Here, neighbors of a point are other points that the algorithm estimates to lie on the same subspace as the point (and not necessarily the closest in Euclidean distance).

2.1 Contribution

In this chapter, we devise new algorithms for each of the two steps above; *(a)* we develop a new method, Nearest Subspace Neighbor (NSN), to determine a neighborhood set for each point, and *(b)* a new method, Greedy Subspace Recovery (GSR), to recover subspaces from given neighborhoods. Each of these two methods can be used in conjunction with other methods for the corresponding other step, while in this chapter we focus on two algorithms, which we call NSN+GSR and NSN+Spectral. They use NSN followed by GSR and Spectral clustering, respectively. Our main contribution is the following.

Statistical guarantees for exact clustering with general subspace conditions. Our algorithms guarantee exactly correct clustering without any assumptions on the subspaces. Most existing (polynomial-time) algorithms are guaranteed correct neighborhoods[2] for some particular conditions on the subspaces. The only statistical results for arbitrary subspaces are in [142, 69, 163, 70], and combined with the algorithm in [164] exact clustering is guaranteed. Our algorithms also guarantee exact clustering without any assumptions on the subspaces, and the difference from the existing results will be described in the following.

[2]By correct neighborhoods, we mean that each point has only the points on the same subspace as neighbors.

Improving conditions with more points. A common remark of the existing statistical guarantees on arbitrary subspaces was that the conditions deteriorate as the number of data points grows (See Table 2.1). These results may not properly explain the clustering performances of their algorithms, which numerically improve with more data points. The reason for this inconsistency is that they are conditions for correct neighborhoods for *all points*, while in practice spectral clustering may result in exact clustering even without any points having correct neighborhoods. Our second step algorithm, GSR, can guarantee exact clustering with only a few points with correct neighborhoods. This enables us to provide *the exact clustering guarantee on arbitrary subspaces that improves as the number of data points increases.*

Statistical guarantees in the presence of noise. When there is noise, the data points are not exactly lying on the subspaces but near the subspaces. We claim that NSN can find correct neighbors also in this case. Our statistical guarantee in the noisy observation model is order-wise the same as the existing algorithms [143, 162, 69].

An efficient algorithm for practical applications. NSN+Spectral provides competitive clustering performance with lower computational cost on practical benchmark datasets. The existing neighborhood selection algorithms with the best clustering performance on practical datasets are either based on convex optimization with the number of variables quadratic in the number of data points [53, 109], or of computational complexity exponential in the subspace dimension [34]. NSN, which is greedy, provides improved neighborhood selection performance with significantly reduced computational time. The experimental results show that NSN+Spectral on public benchmark datasets for

motion segmentation and face clustering achieves comparable clustering error with reduced computation time by an order of magnitude.

2.2 Related work

2.2.1 Subspace clustering

The problem was first formulated in the data mining community [95]. Most of the related work in this field assumes that an underlying subspace is parallel to some canonical axes. Subspace clustering for unions of arbitrary subspaces is considered mostly in the machine learning and the computer vision communities [155]. Most of the results from those communities are based on empirical justification. They provided algorithms derived from theoretical intuition and showed that they perform empirically well with practical datasets. To name a few, GPCA [156], Spectral curvature clustering (SCC) [34], and some expectation-maximization(EM)-like iterative methods [22, 151, 181] show their good empirical performance for subspace clustering. However, they lack theoretical analysis that guarantees exact clustering.

In the theoretical side of the problem, [107] showed that the solution of an optimization problem based on ℓ_p distance with $p \in (0, 1]$ gives perfect recovery of the subspaces, while it is NP-hard to find the solution. [8] provided an algorithm which extends [34]. It is guaranteed exact clustering in the general subspace condition, but the algorithm requires either exponential sample complexity or exponential computational complexity.

As described above, several algorithms with a common structure are recently proposed with both theoretical guarantees and remarkable empirical performance. [53] proposed an algorithm called Sparse Subspace Clustering (SSC), which uses ℓ_1-minimization for neighborhood construction. They

proved that if the subspaces have *no intersection*[3], SSC always finds a correct neighborhood matrix. Later, [142] provided a statistical guarantee of the algorithm for subspaces with intersection. [52] proposed another algorithm called SSC-OMP, which uses Orthogonal Matching Pursuit (OMP) instead of ℓ_1-minimization in SSC. This algorithm is later statistically analyzed by [70, 177]. Another algorithm called Low-Rank Representation (LRR) which uses nuclear norm minimization is proposed by [109]. [163] proposed an hybrid algorithm, Low-Rank and Sparse Subspace Clustering (LRSSC), which involves both ℓ_1-norm and nuclear norm. [69] presented Thresholding based Subspace Clustering (TSC), which constructs neighborhoods based on the inner products between data points. All of these algorithms use spectral clustering for the clustering step.

Most of the above results guarantee correct neighborhoods, where the algorithms find neighbor points from the same subspace for every data point. This does not necessarily imply that the points are clustered perfectly, because the points from the same subspace can be segmented into multiple groups. To ensure exact clustering, one can apply the post-processing algorithm in [164] after the neighborhood selection.

While the above results consider the noiseless model where the points are exactly lying on the subspaces, there have been also analytic results on the noisy model. [143, 162, 69] provide statistical conditions for SSC and TSC to guarantee correct neighborhoods. As opposed to the noiseless model, it is not easy to find an algorithm for perfect clustering from correct neighborhoods. [164] shows that with an additional assumption, which is called Restricted

[3]By no intersection between subspaces, we mean that they share only the null point.

eigenvalue assumption, one can obtain perfect clustering from correct neighborhoods with size at least d for each.

2.2.2 Learning Gaussian mixture model (GMM)

The statistical model we consider in this problem (See Section 2.5.1) is a special case of GMM. Each cluster has the mean point at zero and the covariance matrix which is a low-dimensional projection matrix. Hence, our results can be compared with more general results on learning GMM.

Since the subspace clustering model is a significantly special class of GMM. The sufficient conditions for the subspace clustering algorithms are much weaker than those for the general GMM. While the sample complexity of learning GMM should be polynomial in the ambient dimension and exponential in the number of clusters [122, 66], the algorithms for his specific class can have linear sample complexity in the number of clusters and the subspace dimension which is much smaller than the ambient dimension.

2.3 Problem Setup

There is a set of N data points in $\mathbb{R}^p$, denoted by $\mathcal{Y} = \{y_1, \ldots, y_N\}$. The data points are lying on or near a union of L subspaces $\mathcal{D} = \cup_{i=1}^{L} \mathcal{D}_i$. Each subspace $\mathcal{D}_i$ is of dimension d_i which is smaller than p. For each point y_j, w_j denotes the index of the nearest subspace. Let N_i denote the number of points whose nearest subspace is $\mathcal{D}_i$, i.e., $N_i = \sum_{j=1}^{N} \mathbb{I}\{w_j = i\}$.

2.4 Algorithms

We propose two algorithms for subspace clustering as follows.

- NSN+GSR : Run Nearest Subspace Neighbor (NSN) to construct a neighborhood matrix $W \in \{0,1\}^{N\times N}$, and then run Greedy Subspace Recovery (GSR) for W.
- NSN+Spectral : Run Nearest Subspace Neighbor (NSN) to construct a neighborhood matrix $W \in \{0,1\}^{N\times N}$, and then run spectral clustering for $Z = W + W^\top$.

2.4.1 Nearest Subspace Neighbor (NSN)

NSN approaches the problem of *finding neighbor points most likely to be on the same subspace* in a greedy fashion. At first, given a point y without any other knowledge, the one single point that is most likely to be a neighbor of y is the nearest point of the line $\mathrm{span}\{y\}$. In the following steps, if we have found a few correct neighbor points (lying on the same true subspace) and have no other knowledge about the true subspace and the rest of the points, then the most potentially correct point is the one closest to the subspace spanned by the correct neighbors we have. This motivates us to propose NSN described in the following.

NSN collects K neighbors sequentially for each point. At each step k, a k-dimensional subspace $\mathcal{U}$ spanned by the point and its $k-1$ neighbors is constructed, and the point closest to the subspace is newly collected. After $k \geq k_{\max}$, the subspace $\mathcal{U}$ constructed at the $k_{\max}$th step is used for collecting neighbors. At last, if there are more points lying on $\mathcal{U}$, they are also counted as neighbors. The subspace $\mathcal{U}$ can be stored in the form of a matrix $U \in \mathbb{R}^{p\times \dim(\mathcal{U})}$ whose columns form an orthonormal basis of $\mathcal{U}$. Then $\|\operatorname{Proj}_{\mathcal{U}} y_j\|_2$ can be computed easily because it is equal to $\|U^\top y_j\|_2$. While a naive implementation requires $O(K^2pN^2)$ computational cost, this can be reduced to $O(KpN^2)$, and

Algorithm 1 Nearest Subspace Neighbor (NSN)

Input: A set of N samples $\mathcal{Y} = \{y_1, \ldots, y_N\}$, The number of required neighbors K, Maximum subspace dimension $k_{\max}$.
Output: A neighborhood matrix $W \in \{0,1\}^{N \times N}$

(Optional) $y_i \leftarrow y_i / \|y_i\|_2$, $\forall i \in [N]$ ▷ Normalize magnitudes
for $i = 1, \ldots, N$ **do** ▷ Run NSN for each data point
  $\mathcal{I}_i \leftarrow \{i\}$
  for $k = 1, \ldots, K$ **do** ▷ Iteratively add the closest point to the current subspace
    if $k \leq k_{\max}$ **then**
      $\mathcal{U} \leftarrow \text{span}\{y_j : j \in \mathcal{I}_i\}$
    end if
    $j^* \leftarrow \arg\max_{j \in [N] \setminus \mathcal{I}_i} \| \text{Proj}_{\mathcal{U}} y_j \|_2$
    $\mathcal{I}_i \leftarrow \mathcal{I}_i \cup \{j^*\}$
  end for
  $W_{ij} \leftarrow \mathbb{I}_{j \in \mathcal{I}_i \text{ or } y_j \in \mathcal{U}}, \ \forall j \in [N]$ ▷ Construct the neighborhood matrix
end for

the faster implementation is described in Section 2.6.1. We note that this computational cost is much lower than that of the convex optimization based methods (e.g., SSC [53] and LRR [109]) which solve a convex program with N^2 variables and pN constraints.

NSN for subspace clustering shares the same philosophy with Orthogonal Matching Pursuit (OMP) for sparse recovery in the sense that it incrementally picks the point (dictionary element) that is the most likely to be correct, assuming that the algorithms have found the correct ones. In subspace clustering, that point is the one closest to the subspace spanned by the currently selected points, while in sparse recovery it is the one closest to the residual of linear regression by the selected points. In the sparse recovery literature, the performance of OMP is shown to be comparable to that of Basis Pursuit (ℓ_1-minimization) both theoretically and empirically [150, 96]. One of the con-

tributions of this work is to show that this high-level intuition is indeed born out, provable, as we show that NSN also performs well in collecting neighbors lying on the same subspace.

2.4.2 Greedy Subspace Recovery (GSR)

Suppose that NSN has found correct neighbors for a data point. How can we check if they are indeed correct, that is, lying on the same true subspace? One natural way is to count the number of points close to the subspace spanned by the neighbors. If they span one of the true subspaces, then many other points will be lying on the span. If they do not span any true subspaces, few points will be close to it. This fact motivates us to use a greedy algorithm to recover the subspaces. Using the neighborhood constructed by NSN (or some other algorithm), we recover the L subspaces. If there is a neighborhood set containing only the points on the same subspace for each subspace, the algorithm successfully recovers the unions of the true subspaces exactly.

Recall that the matrix W contains the labelings of the points, so that $W_{ij} = 1$ if point i is assigned to subspace j. PCA$(\cdot)$ denotes a principal subspace of the input set of vectors. This can be obtained by taking the first d left singular vectors of the matrix whose columns are the vector in the set. If there are only d vectors in the set, Gram-Schmidt orthogonalization will give us the subspace. As in NSN, it is efficient to store a subspace $\mathcal{W}_i$ in the form of its orthogonal basis because we can easily compute the norm of a projection onto the subspace.

Testing a candidate subspace by counting the number of near points has already been considered in the subspace clustering literature. In [173], the authors proposed to run RANdom SAmple Consensus (RANSAC) iteratively.

Algorithm 2 Greedy Subspace Recovery (GSR)

Input: N points $\mathcal{Y} = \{y_1, \ldots, y_N\}$, A neighborhood matrix $W \in \{0,1\}^{N \times N}$, Error bound ϵ

Output: Estimated subspaces $\hat{\mathcal{D}} = \cup_{l=1}^{L} \hat{D}_l$. Estimated labels $\hat{w}_1, \ldots, \hat{w}_N$

$y_i \leftarrow y_i / \|y_i\|_2, \forall i \in [N]$ ▷ Normalize magnitudes

$\mathcal{W}_i \leftarrow \text{span}\{y_j : W_{ij} = 1\}, \forall i \in [N]$ ▷ Estimate a subspace using the neighbors for each point

$C_i \leftarrow \sum_{j=1}^{N} \mathbb{I}\{W_{ij} = 0, \| \text{Proj}_{\mathcal{W}_i} y_j\|_2 \geq 1 - \epsilon\}, \forall i \in [N]$

$\mathcal{I} \leftarrow [N]$, $L \leftarrow 0$

while $\max_{i \in \mathcal{I}} C_i \geq C$ **do**

$i^* \leftarrow \arg\max_{i \in \mathcal{I}} C_i$ ▷ Iteratively pick the best subspace estimates

$\hat{\mathcal{D}}_L \leftarrow \text{PCA}\{y_j : \| \text{Proj}_{\mathcal{W}_{i^*}} y_j\|_2 \geq 1 - \epsilon\}$

$\mathcal{I} \leftarrow \mathcal{I} \setminus \{j : \| \text{Proj}_{\mathcal{W}_{i^*}} y_j\|_2 \geq 1 - \epsilon\}$

$L \leftarrow L + 1$

end while

$\hat{w}_i \leftarrow \arg\max_{l \in [L]} \| \text{Proj}_{\hat{D}_l} y_i\|_2, \ \forall i \in [N]$ ▷ Label the points using the subspace estimates

RANSAC randomly selects a few points and checks if there are many other points near the subspace spanned by the collected points. Instead of randomly choosing sample points, GSR receives some candidate subspaces (in the form of sets of points) from NSN (or possibly some other algorithm) and selects subspaces in a greedy way as specified in the algorithm above.

2.5 Statistical results

We analyze our algorithms in two standard noiseless models. The main theorems present sufficient conditions under which the algorithms cluster the points exactly with high probability. For simplicity of analysis, we assume that every subspace is of the same dimension, and the number of data points on each subspace is the same, i.e., $d \triangleq d_1 = \cdots = d_L, \quad n \triangleq N_1 = \cdots = N_L$. We assume that d is known to the algorithm. Nonetheless, our analysis can

extend to the general case.

2.5.1 Models

We consider two models which have been used in the subspace clustering literature:

- Fully random model: The subspaces are drawn iid uniformly at random, and the points are also iid randomly generated.
- Semi-random model: The subspaces are arbitrarily determined, but the points are iid randomly generated.

Let $D_i \in \mathbb{R}^{p\times d}, i \in [L]$ be a matrix whose columns form an orthonormal basis of $\mathcal{D}_i$. An important measure that we use in the analysis is the *affinity* between two subspaces, defined as

$$\operatorname{aff}(i,j) \triangleq \frac{\|D_i^\top D_j\|_F}{\sqrt{d}} = \sqrt{\frac{\sum_{k=1}^d \cos^2\theta_k^{i,j}}{d}} \in [0,1],$$

where $\theta_k^{i,j}$ is the kth principal angle between $\mathcal{D}_i$ and $\mathcal{D}_j$. Two subspaces $\mathcal{D}_i$ and $\mathcal{D}_j$ are identical if and only if $\operatorname{aff}(i,j) = 1$. If $\operatorname{aff}(i,j) = 0$, every vector on $\mathcal{D}_i$ is orthogonal to any vectors on $\mathcal{D}_j$. We also define the maximum affinity as

$$\max \operatorname{aff} \triangleq \max_{i,j\in[L],i\neq j} \operatorname{aff}(i,j) \in [0,1].$$

There are $N = nL$ points, and there are n points exactly lying on each subspace. We assume that each observation y_i is the sum of a data point x_i drawn iid from the spherical Gaussian on $\mathcal{D}_{w_i}$ and a noise z_i drawn iid from the spherical Gaussian in the ambient space. Equivalently, we can write

$$y_i = D_{w_i}x_i + z_i, \quad x_i \sim \mathcal{N}\left(0, \frac{1}{d}I_{d\times d}\right),\ z_i \sim \mathcal{N}\left(0, \frac{\sigma^2}{p}I_{p\times p}\right), \quad \forall i \in [N].$$

We note that in the noiseless case ($\sigma^2 = 0$) the analysis for the Gaussian distribution of x_i is equivalent to that for the uniform distribution on $\mathcal{S}^{d-1}$, which is assumed in the existing results. From the uniform distribution, we can scale each vector by an independent Rayleigh random variable to obtain the Gaussian distribution. From the Gaussian distribution, we can normalize each vector to obtain uniform distribution. However, we consider the Gaussian case for simpler analysis.

2.5.2 Main results

We first consider the points without noise ($\sigma^2 = 0$). The first theorem gives a statistical guarantee for the fully random model.

Theorem 2.1. *Suppose L d-dimensional subspaces and n points on each subspace are generated in the fully random model. There are constants $c_1, c_2 > 0$ such that if*

$$\frac{n}{d} \geq c_1 \left(\log \frac{n}{d\delta}\right)^2, \quad \frac{d}{p} \leq \frac{c_2 \log n}{\log(ndL\delta^{-1})}, \tag{2.1}$$

then with probability at least $1 - \frac{3L\delta}{1-\delta}$, NSN+GSR[4] clusters the points exactly. Also, there are other constants $c_1', c_2' > 0$ such that if (2.1) with c_1 and c_2 replaced by c_1' and c_2' holds then NSN+Spectral[5] clusters the points exactly with probability at least $1 - \frac{3L\delta}{1-\delta}$.

Our sufficient conditions for exact clustering explain when subspace clustering becomes easy or difficult, and they are consistent with our intuition. For NSN to find correct neighbors, the points on the same subspace should

[4]NSN with $K = d - 1, k_{max} = 1 \vee \lfloor 2 \log d \rfloor$ followed by GSR with arbitrarily small ϵ.
[5]NSN with $K = d - 1, k_{max} = 1 \vee \lfloor 2 \log d \rfloor$.

be many enough so that they look like lying on a subspace. This condition is spelled out in the first inequality of (2.1). We note that the condition holds even when n/d is a constant, i.e., n is linear in d. The second inequality implies that the dimension of the subspaces should not be too high for subspaces to be distinguishable. If d is high, the random subspaces are more likely to be close to each other, and hence they become more difficult to be distinguished. However, as n increases, the points become dense on the subspaces, and hence it becomes easier to identify different subspaces. These scaling property holds when the number of clusters L is at most polynomial in the subspace dimension d. By replacing δ with δ/L, we obtain a sufficient condition which is order-wise the same as (2.1).

Let us compare our result with the conditions required for success in the fully random model in the existing literature.[6] In [142], it is guaranteed for SSC to have correct neighborhoods that n should be superlinear in d when d/p fixed. In [68, 163], the conditions on d/p becomes worse as we have more points. On the other hand, our algorithms are guaranteed exact clustering of the points, and the sufficient condition is order-wise at least as good as the conditions for correct neighborhoods by the existing algorithms (See Table 2.1). Moreover, exact clustering is guaranteed even when n is linear in d, and d/p fixed.

For the semi-random model, we have the following general theorem.

Theorem 2.2. *Suppose L d-dimensional subspaces are arbitrarily chosen, and n points on each subspace are generated in the semi-random model. There are*

[6] We consider L is at most polynomial in d so that the guarantees of the existing work are valid, i.e., the success probabilities approach one.

constants $c_1, c_2 > 0$ *such that if*

$$\frac{n}{d} \geq c_1 \left(\log \frac{n}{d\delta}\right)^2, \quad \max \text{aff} \leq \sqrt{\frac{c_2 \log n}{\log(dL\delta^{-1}) \cdot \log(ndL\delta^{-1})}}. \tag{2.2}$$

then with probability at least $1 - \frac{3L\delta}{1-\delta}$*, NSN+GSR*[7] *clusters the points exactly.*

In the semi-random model, the sufficient condition does not depend on the ambient dimension p. When the affinities between subspaces are fixed, and the points are exactly lying on the subspaces, the difficulty of the problem does not depend on the ambient dimension. It rather depends on $\max \text{aff}$, which measures how close the subspaces are. As they become closer to each other, it becomes more difficult to distinguish the subspaces. The second inequality of (2.2) explains this intuition. The inequality also shows that if we have more data points, the problem becomes easier to identify different subspaces.

Compared with other algorithms, NSN+GSR is guaranteed exact clustering, and more importantly, the condition on $\max \text{aff}$ improves as n grows. This remark is consistent with the practical performance of the algorithm which improves as the number of data points increases, while the existing guarantees of other algorithms are not. In [142], correct neighborhoods in SSC are guaranteed if $\max \text{aff} = O(\sqrt{\log(n/d)}/\log(nL))$. In [68], exact clustering of TSC is guaranteed if $\max \text{aff} = O(1/\log(nL))$. However, these algorithms perform empirically better as the number of data points increases.

2.5.3 Guarantees on the noisy model

Now let us consider the observation is noisy. Opposed to the noiseless model, even when NSN finds correct neighbors from the same true subspace,

[7] NSN with $K = d - 1$ and $k_{max} = \lfloor 2 \log d \rfloor$ followed by GSR with arbitrarily small ϵ.

the intermediate subspace $\mathcal{U}$ spanned by the neighbors is distinct from the true subspace. Our analysis claims that as long as the noise variance σ^2 is bounded and small, NSN can find all correct neighbors using the perturbed intermediate subspaces.

Theorem 2.3. *Suppose L d-dimensional subspaces are arbitrarily chosen, and n points on each subspace are generated in the noisy semi-random model. There are constants $c_1, c_2 > 0$ such that if*

$$\frac{n}{d} \geq c_1 \log \frac{n}{d\delta}, \quad \max \text{aff} < \frac{c_2}{\log(nL)} - 10(1+\sigma)^2\sqrt{\frac{d}{p}}, \tag{2.3}$$

then with high probability, NSN finds correct neighbors for every point.

When the model is noisy, similarly to the existing results [143, 162, 69], we can only guarantee correct neighborhoods for all points, and the condition on $\max \text{aff}$ deteriorates as n grows. Again, this is because one requires every point to have correct neighborhoods, and it is reasonable that this condition becomes worse as n and L increases.

Our result demonstrates that the noise variance affects the sufficient condition on $\max \text{aff}$. If the noise gets higher, the affinity between subspaces should be smaller. A surprising remark is that the noise power σ^2 can be as large as $O(\sqrt{\frac{p}{d(\log nL)^2}})$, which is in general much higher than the signal power which is $O(1)$. In this case, a intermediate subspace $\mathcal{U}$ is obtained from very noisy points, and it is far from a true subspace even when the algorithm have collected the points all from the subspace. However, since that is due to the noise, $\mathcal{U}$ is even farther than the other true subspaces, and it is less likely for their points to be collected. This remark, being able to pick correct neighbors in the highly noisy model, has been also observed with the existing algorithms [162, 69].

2.6 Implementation

2.6.1 A faster implementation for NSN

At each step of NSN, the algorithm computes the projections of all points onto a subspace and find one with the largest norm. A naive implementation of the algorithm requires $O(pK^2N^2)$ time complexity.

In fact, we can reduce the complexity to $O(pKN^2)$. Instead of finding the maximum norm of the projections, we can find the maximum squared norm of the projections. Let $\mathcal{U}_k$ be the subspace $\mathcal{U}$ at step k. For any data point y, we have

$$\| \operatorname{Proj}_{\mathcal{U}_k} y\|_2^2 = \| \operatorname{Proj}_{\mathcal{U}_{k-1}} y\|_2^2 + |u_k^\top y|^2$$

where u_k is the new orthogonal axis added from $\mathcal{U}_{k-1}$ to make $\mathcal{U}_k$. That is, $\mathcal{U}_{k-1} \perp u_k$ and $\mathcal{U}_k = \mathcal{U}_{k-1} \bigoplus u_k$. As $\| \operatorname{Proj}_{\mathcal{U}_{k-1}} y\|_2^2$ is already computed in the $(k-1)$'th step, we do not need to compute it again at step k. Based on this fact, we have a faster implementation as described in the following. Note that P_j at the kth step is equal to $\| \operatorname{Proj}_{\mathcal{U}_k} y_j\|_2^2$ in the original NSN algorithm.

2.6.2 Estimation of the number of clusters

When L is unknown, it can be estimated at the clustering step. For Spectral clustering, a well-known approach to estimate L is to find a knee point in the singular values of the neighborhood matrix. It is the point where the difference between two consecutive singular values are the largest. For GSR, we do not need to estimate the number of clusters a priori. Once the algorithms finishes, the number of the resulting groups will be our estimate of L.

Algorithm 3 Fast Nearest Subspace Neighbor (F-NSN)

Input: A set of N samples $\mathcal{Y} = \{y_1, \ldots, y_N\}$, The number of required neighbors K, Maximum subspace dimension k_{max}.

Output: A neighborhood matrix $W \in \{0,1\}^{N \times N}$

(Optional) $y_i \leftarrow y_i / \|y_i\|_2$, $\forall i \in [N]$

for $i = 1, \ldots, N$ **do**

  $\mathcal{I}_i \leftarrow \{i\}$, $u_1 \leftarrow y_i$

  $P_j \leftarrow 0, \forall j \in [N]$

  for $k = 1, \ldots, K$ **do**

    if $k \leq k_{max}$ **then**

      $P_j \leftarrow P_j + \|u_k^\top y_j\|^2, \ \forall j \in [N]$

    end if

    $j^* \leftarrow \arg\max_{j \in [N], j \notin \mathcal{I}_i} P_j$

    $\mathcal{I}_i \leftarrow \mathcal{I}_i \cup \{j^*\}$

    if $k < k_{max}$ **then**

      $u_{k+1} \leftarrow \frac{y_{j^*} - \sum_{l=1}^{k} (u_l^\top y_{j^*}) u_l}{\|y_{j^*} - \sum_{l=1}^{k} (u_l^\top y_{j^*}) u_l\|_2}$

    end if

  end for

  $W_{ij} \leftarrow \mathbb{I}_{j \in \mathcal{I}_i \text{ or } P_j = 1}, \ \forall j \in [N]$

end for

2.6.3 Parameter setting

The choices of K and $k_{\max}$ depend on the dimension of the subspaces d. If data points are lying exactly on the model subspaces, $K = k_{\max} = d$ is enough for GSR to recover a subspace. In practical situations where the points are near the subspaces, it is better to set K to be larger than d. $k_{\max}$ can also be larger than d because if we have collected correct neighbors then the $k_{\max} - d$ additional dimensions, which may be induced from the noise, do not intersect with the other subspaces in practice. In our motion segmentation and face clustering experiments on real datasets, we found that our algorithm performs well if $K = k_{\max}$ is set to be around $2d$.

2.7 Experimental results

In this section, we empirically compare our algorithms with the existing algorithms in terms of clustering performance and computational time (on a single desktop). For NSN, we used the fast implementation described in Section 2.6.1. The compared algorithms are K-means, K-flats[8], SSC, LRR, SCC, TSC[9], and SSC-OMP[10]. The numbers of replicates in K-means, K-flats, and the K-means used in the spectral clustering are all fixed to 10. The algorithms are compared in terms of *Clustering error (CE)* and *Neighborhood*

[8] K-flats is similar to K-means. At each iteration, it computes top-d principal subspaces of the points with the same label, and then labels every point based on its distances to those subspaces.

[9] The MATLAB codes for SSC, LRR, SCC, and TSC are obtained from `http://www.cis.jhu.edu/~ehsan/code.htm`, `https://sites.google.com/site/guangcanliu`, and `http://www.math.duke.edu/~glchen/scc.html`, `http://www.nari.ee.ethz.ch/commth/research/downloads/sc.html`, respectively.

[10] For each data point, OMP constructs a neighborhood for each point by regressing the point on the other points up to 10^{-4} accuracy.

selection error (NSE), defined as

$$(\text{CE}) = \min_{\pi \in \Pi_L} \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}(w_i \neq \pi(\hat{w}_i)), \quad (\text{NSE}) = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}(\exists j : W_{ij} \neq 0, w_i \neq w_j)$$

where Π_L is the permutation space of $[L]$. CE is the proportion of incorrectly labeled data points. Since clustering is invariant up to permutation of label indices, the error is equal to the minimum disagreement over the permutation of label indices. NSE measures the proportion of the points which do not have all correct neighbors. Having all correct neighbors for a point has been of interest for analysis in recent subspace clustering literature. Such a property is called subspace detection property [142], exact feature selection [52], or self-expressiveness property [53]. To be fair in comparison, we fixed the number of neighbors to be d over all algorithms. [11]

2.7.1 Synthetic data: Fully random model

We compare the performances on synthetic data generated from the fully random model. In $\mathbb{R}^p$, five d-dimensional subspaces are generated uniformly at random. Then for each subspace n unit-norm points are generated iid uniformly at random on the subspace. To see the agreement with the theoretical result, we ran the algorithms under fixed d/p and varied n and d. We set $d/p = 3/5$ so that each pair of subspaces has intersection.

Figures 2.1 and 2.2 show CE and NSE, respectively. Each error value is averaged over 100 trials. Figure 2.1 indicates that our algorithm clusters the data points better than the other algorithms. As predicted in the theorems, the

[11] For the neighborhood matrices from SSC, LRR, and SSC-OMP, the d points with the maximum weights are regarded as neighbors for each point. For TSC, the d nearest neighbors are collected for each point.

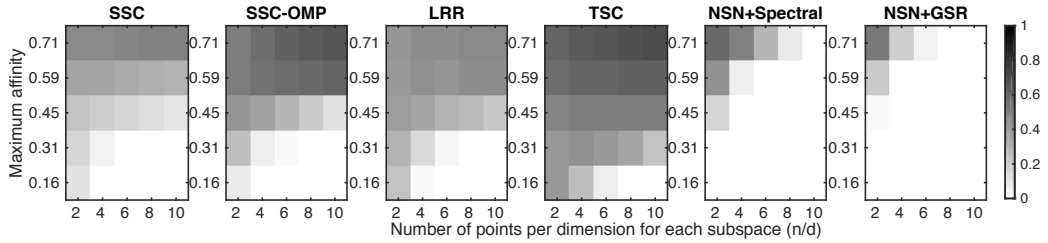


Figure 2.1: CE of algorithms in the fully random model. Five random d-dimensional subspaces are generated iid uniformly at random, and n iid uniformly random unit-norm points are drawn on each subspace. The figures shows CE for different numbers of n/d and ambient dimension p. d/p is fixed to be 3/5. Brighter cells represent that less data points are clustered incorrectly.

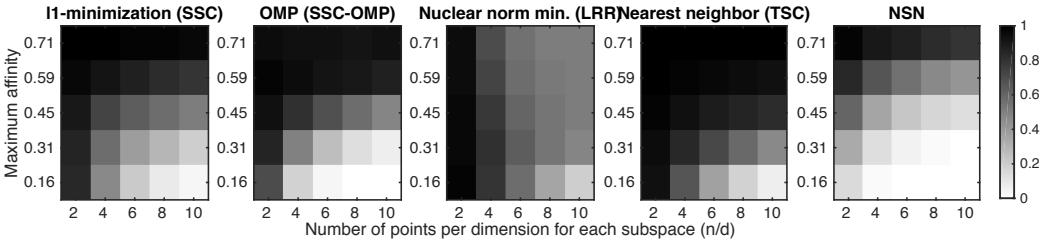


Figure 2.2: NSE in the fully random model with the same model parameters as those in Figure 2.1. Brighter cells represent that more data points have all correct neighbors.

clustering performance improves as the number of points increases. However, it also improves as the dimension of subspaces grows in contrast to the theoretical analysis. We believe that this is because our analysis on GSR is not tight. In Figure 2.2, we can see that more data points obtain correct neighbors as n increases or d decreases, which conforms the theoretical analysis.

We also compare the computational time of the neighborhood selection algorithms for different numbers of subspaces and data points. As shown

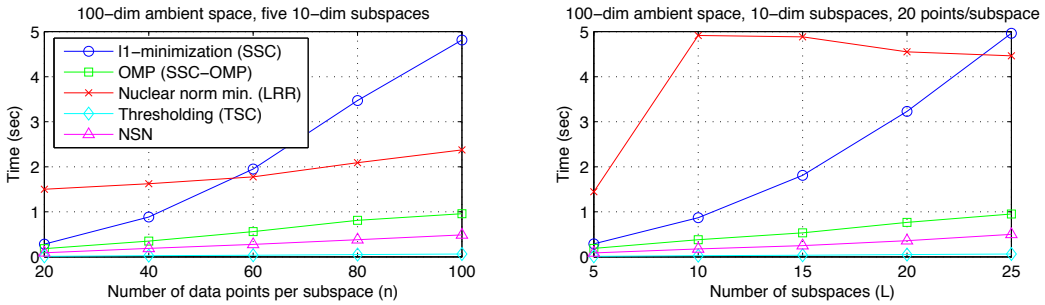


Figure 2.3: Average computational time of the neighborhood selection algorithms

in Figure 2.3, the greedy algorithms (OMP, Thresholding, and NSN) are significantly more scalable than the convex optimization based algorithms (ℓ_1-minimization and nuclear norm minimization).

2.7.2 Synthetic data: Semi-random model

Now we evaluate our algorithm for controlled affinity betweeen subspaces. In $\mathbb{R}^{50}$, four 5-dimensional subspaces are generated as follows. Let $D^{(1)}$ and $D^{(2)}$ be random orthonormal matrices such that $D^{(1)\top}D^{(1)} = D^{(2)\top}D^{(2)} = I_{5\times 5}$, and $D^{(1)\top}D^{(2)} = 0$. Then the orthonormal basis of subspace $\mathcal{D}_i$ is given by

$$D_i = \cos(\theta \cdot i) \cdot D^{(1)} + \sin(\theta \cdot i) \cdot D^{(2)}, \quad i \in \{1, 2, 3, 4\}.$$

Note that if $\theta \leq \pi/4$, the maximum affinity between the subspaces is given by $\max \text{aff} = \sin(\theta)$. Given these subspaces, n unit-norm points are generated uniformly at random for each subspace. Each error value is averaged over 100 trials.

Figures 2.4 and 2.5 show CE and NSE, respectively. Each error value is

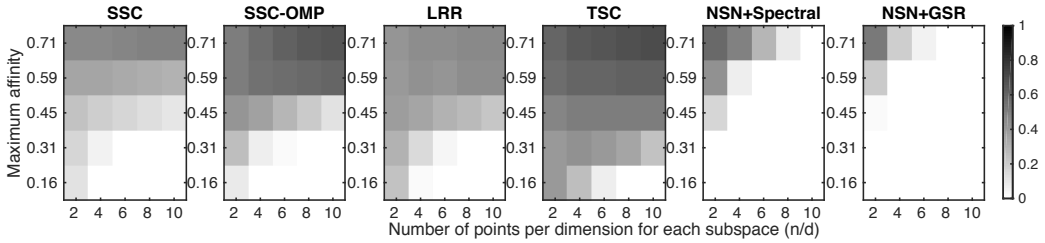


Figure 2.4: CE of algorithms in the semi-random model. Four 5-dimensional subspaces are generated with fixed max aff, and iid uniformly random unit-norm points are drawn on each subspace. The figures shows CE for different numbers of n/d and max aff. Brighter cells represent that less data points are clustered incorrectly.

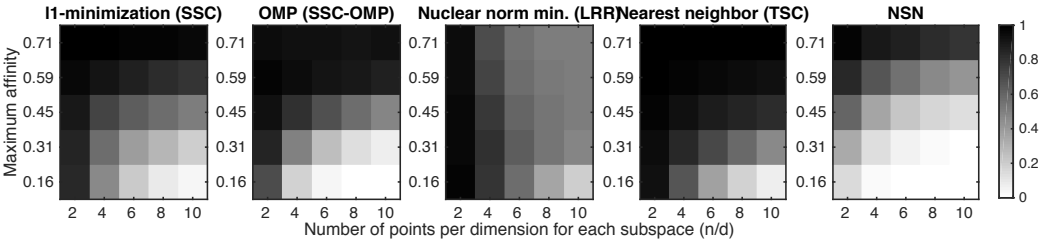


Figure 2.5: NSE in the semi-random model with the same model parameters as those in Figure 2.4. Brighter cells represent that more data points have all correct neighbors.

averaged over 100 trials. Figure 2.4 indicates that our algorithm clusters the data points better than the other algorithms. As predicted in the theorems, clustering becomes harder as the maximum affinity grows, but the performance improves as the number of points increases. Also in Figure 2.2, we can see a similar performance behavior.

2.7.3 Motion segmentation

For the motion segmentation, we used Hopkins155 dataset [149], which contains 155 video sequences of 2 or 3 motions. Table 2.2 shows CE and average computational time. We can see that NSN+Spectral performs competitively with the methods with the lowest errors, but much faster. Compared to the other greedy neighborhood construction based algorithms, SSC-OMP and TSC, our algorithm performs significantly better.

2.7.4 Face clustering

For the face clustering, we used Extended Yale B dataset with cropped images from [58, 105]. The dataset contains 64 images of size 192×168 pixels for each of 38 individuals in frontal view and different illumination conditions. To compare with the existing algorithms, we used the set of 48×42 resized raw images provided by the authors of [53]. The parameters of the existing algorithms were set as provided in their source codes.[12]

Table 2.3 show CE and average computational time for K-means, K-flats, SSC, SSC-OMP, TSC and NSN+Spectral.[13] We omitted the result from NSN+GSR since it did not perform well in this practical dataset. However, we can see that NSN+Spectral performs competitively with the methods with the lowest errors, but much faster. Compared to the other greedy neighborhood construction based algorithms, SSC-OMP and TSC, our algorithm performs significantly better.

[12] As SSC-OMP and TSC do not have proposed number of parameters for motion segmentation, we found the numbers minimizing the mean CE. The numbers are given in the table.

[13] The LRR code provided by the author did not perform properly with the face clustering dataset that we used.

2.8 Proofs

2.8.1 Proofs for the noiseless model (Theorems 2.1 and 2.2)

Exact clustering of our algorithms depends on whether NSN can find all correct neighbors for the data points so that the following algorithm (GSR or Spectral clustering) can cluster the points exactly. For NSN+GSR, exact clustering is guaranteed when there is a point on each subspace that have all correct neighbors which are at least $d-1$. For NSN+Spectral, exact clustering is guaranteed when each data point has only the $n-1$ other points on the same subspace as neighbors. In the beginning step, we explain why these are true.

Step 1a: Exact clustering condition for GSR

The two statistical models have a property that for any d-dimensional subspace in $\mathbb{R}^p$ other than the true subspaces $\mathcal{D}_1, \ldots, \mathcal{D}_L$ the probability of any points lying on the subspace is zero. Hence, we claim the following.

Fact 2.4 (Best d-dimensional fit)**.** *With probability one, the true subspaces $\mathcal{D}_1, \ldots, \mathcal{D}_L$ are the L subspaces containing the most points among the set of all possible d-dimensional subspaces.*

Then it suffices for each subspace to have one point whose neighbors are $d-1$ all correct points on the same subspace. This is because the subspace spanned by those d points is almost surely identical to the true subspace they are lying on, and that subspace will be picked by GSR.

Fact 2.5. *If NSN with $K \geq d-1$ finds all correct neighbors for at least one point on each subspace, GSR recovers all the true subspaces and clusters the*

data points exactly with probability one. That is,

$$\forall l \in [L], \exists i : w_j = l : W_{ij} = 1 \;\Rightarrow\; \hat{\mathcal{D}} \equiv \mathcal{D}, \exists \pi \in \Pi_L : \pi(\hat{w}_i) = w_i, \forall i \in [N].$$

where Π_L *is the permutation space for* $[L]$.

In the following steps, we consider one data point for each subspace. We show that NSN with $K = d - 1$ finds all correct neighbors for the point with probability at least $1 - \frac{3\delta}{1-\delta}$. Then the union bound and Fact 2.5 establish exact clustering with probability at least $1 - \frac{3L\delta}{1-\delta}$.

Step 1b: Exact clustering condition for spectral clustering

It is difficult to analyze spectral clustering for the resulting neighborhood matrix of NSN. A trivial case for a neighborhood matrix to result in exact clustering is when the points on the same subspace form a single fully connected component. If NSN with $K = k_{max} = d$ finds all correct neighbors for every data point, the subspace $\mathcal{U}$ at the last step ($k = K$) is almost surely identical to the true subspace that the points lie on. Hence, the resulting neighborhood matrix W form L fully connected components each of which contains all of the points on the same subspace.

In the rest of the proof, we show that if (2.1) holds, NSN finds all correct neighbors for a fixed point with probability $1 - \frac{3\delta}{1-\delta}$. Let us assume that this is true. If (2.1) with c_1 and c_2 replaced by $4c_1$ and $\frac{c_2}{2}$ holds, we have

$$\frac{n}{d} > 4c_1 \left(\log \frac{n}{d\delta}\right)^2 \geq c_1 \left(\log \frac{n}{d(\delta/n)}\right)^2,$$
$$\frac{d}{p} < \frac{c_2 \log n}{2 \log(ndL\delta^{-1})} \leq \frac{c_2 \log n}{\log(ndL(\delta/n)^{-1})}.$$

Then it follows from the union bound that NSN finds all correct neighbors for all of the n points on each subspace with probability at least $1-\frac{3L\delta}{1-\delta}$, and hence we obtain $W_{ij} = \mathbb{I}_{w_i=w_j}$ for every $(i,j) \in [N]^2$. Exact clustering is guaranteed.

Step 2: Analysis of NSN using an Oracle algorithm

Now the only proposition that we need to prove is that for each subspace $\mathcal{D}_i$ NSN finds all correct neighbors for a data point (which is a uniformly random unit vector on the subspace) with probability at least $1-\frac{3\delta}{1-\delta}$. As our analysis is independent of the subspaces, we only consider $\mathcal{D}_1$. Without loss of generality, we assume that y_1 lies on $\mathcal{D}_1$ ($w_1 = 1$) and focus on this data point.

Consider the Oracle algorithm in the following. Unlike NSN, this algorithm knows the true label of each data point. It picks the point closest to the current subspace among the points with the same label. Since we assume $w_1 = 1$, the Oracle algorithm for y_1 picks a point in $\{y_j : w_j = 1\}$ at every step.

Note that the Oracle algorithm returns failure if and only if the original algorithm picks an incorrect neighbor for y_1. The reason is as follows. Suppose that NSN for y_1 picks the first incorrect point at step k. By the step $k-1$, correct points have been chosen because they are the nearest points for the subspaces in the corresponding steps. The Oracle algorithm will also pick those points because they are the nearest points among the correct points. Hence $\mathcal{U} \equiv \mathcal{V}_k$. At step k, NSN picks an incorrect point as it is the closest to $\mathcal{U}$. The Oracle algorithm will declare failure because that incorrect point is closer than the closest point among the correct points. In the same manner, we see that NSN fails if the Oracle NSN fails. Therefore, we can instead analyze the

Algorithm 4 NSN Oracle algorithm for y_1 (assuming $w_1 = 1$)

Input: A set of N samples $\mathcal{Y} = \{y_1, \dots, y_N\}$, The number of required neighbors $K = d - 1$, Maximum subspace dimension $k_{max} = \lceil 2 \log d \rceil$

$\mathcal{I}_1^{(1)} \leftarrow \{1\}$

for $k = 1, \dots, K$ **do**

  if $k \leq k_{max}$ **then**

    $\mathcal{V}_k \leftarrow \text{span}\{y_j : j \in \mathcal{I}_1^{(k)}\}$

  else

    $\mathcal{V}_k \leftarrow \mathcal{V}_{k_{\max}}$

  end if

  $j_k^* \leftarrow \arg\max_{j \in [N] \setminus \mathcal{I}_k : w_j = 1} \| \text{Proj}_{\mathcal{V}_k} y_j \|_2$

  if $\max_{j \in [N] : w_j = 1, j \notin \mathcal{I}_i^{(k)}} \| \text{Proj}_{\mathcal{V}_k} y_j \|_2 \leq \max_{j \in [N] : w_j \neq 1} \| \text{Proj}_{\mathcal{V}_k} y \|_2$ **then**

    Return FAILURE

  end if

  $\mathcal{I}_1^{(k+1)} \leftarrow \mathcal{I}_1^{(k)} \cup \{j_k^*\}$

end for

Return SUCCESS

success of the Oracle algorithm. The success condition is written as

$$\| \text{Proj}_{\mathcal{V}_k} y_{j_k^*} \|_2 > \max_{j \in [N] : w_j \neq 1} \| \text{Proj}_{\mathcal{V}_k} y \|_2, \quad \forall k = 1, \dots, K. \tag{2.4}$$

Remark 2.6. *For all k, $\mathcal{V}_k$ is independent of the points $\{y_j : j \in [N] : w_j \neq 1\}$.*

The rest of the proof is the following technical lemmas.

Lemma 2.7. *If the conditions* (2.2) *holds for the semi-random model, then* (2.4) *holds with probability at least* $1 - \frac{3\delta}{1-\delta}$.

Lemma 2.8. *If the conditions* (2.1) *holds for the fully random model, then* (2.4) *holds with probability at least* $1 - \frac{3\delta}{1-\delta}$.

The proofs can be found in Appendices A.1.2 and A.1.3. This technique is similar to the proof technique for OMP [150, 36]. The key difference, which

is the main difficulty, is that the incremental subspace at each step is still dependent of the correct points, while the independence between the residual and the correct points is crucial in the proof for OMP. We provide a novel intuition to disentangle this dependence using stochastic ordering (Lemma A.1.4).

2.8.2 Proof outline for the noisy model (Theorem 2.3)

When there is noise ($\sigma^2 > 0$), we only consider correct neighborhood selection of NSN. Similarly to the proof for the noiseless model in Section 2.8.1, we only need to show (2.4) holds for every point. If the following lemma holds, then we can use the union bound to complete the proof of Theorem 2.3.

Lemma 2.9. *If the condition* (2.3) *holds for the semi-random model, then* (2.4) *holds with probability at least* $1 - \frac{3\delta}{n(1-\delta)} - \frac{2}{(ndL)^2} - 4e^{-d/16}$.

Since the probability of failure approaches zero even when multipled by $N = nL$, NSN finds an all-correct neighborhood with size d for every data point with high probability. The proof of Lemma 2.9 is provided in Appendix A.1.5.

Algorithm	Neighborhood Selection	Clustering	Statistical conditions for: Arbitrary subspaces	Random subspaces
SSC [53, 142]	ℓ_1-min.	Spectral	$\max \text{aff} = O(\frac{\sqrt{\log(n/d)}}{\log(nL)})$	$\frac{d}{p} = O(\frac{\log(n/d)}{\log(nL)})$
LRR [109]	Nuc. norm min.	Spectral	-	-
SSC-OMP [52, 70, 177]	OMP	Spectral	$\max \text{aff} = O(\frac{\sqrt{\log(n/d)}}{\log(nL)})$	$\frac{d}{p} = O(\frac{\log(n/d)}{\log(nL)})$
TSC [68, 69]	NN	Spectral	$\max \text{aff} = O(\frac{1}{\log(nL)})$	$\frac{d}{p} = O(\frac{1}{\log(nL)})$
LRSSC [163]	Hybrid	Spectral	-	$\frac{d}{p} = O(\frac{1}{\log(nL)})$
NSN+GSR	NSN	GSR	$\max \text{aff} = O(\sqrt{\frac{\log n}{(\log dL)\cdot\log(ndL)}})$	$\frac{d}{p} = O(\frac{\log n}{\log(ndL)})$
NSN+Spectral	NSN	Spectral	-	$\frac{d}{p} = O(\frac{\log n}{\log(ndL)})$

Table 2.1: (Polynomial-time) Subspace clustering algorithms with theoretical guarantees. LRR has only deterministic guarantees, not statistical ones. In the two standard statistical models, there are n data points on each of L d-dimensional subspaces in $\mathbb{R}^p$. For the definition of max aff, we refer the readers to Section 2.5.1.

L	Algorithms	K-flats	SSC	LRR	SCC	OMP(8)	TSC(10)	NSN+Spec(5)
2	Mean CE (%)	13.62	1.52	2.13	2.06	16.92	18.44	3.62
	Median CE (%)	10.65	0.00	0.00	0.00	12.77	16.92	0.00
	Avg. Time (sec)	0.80	3.03	3.42	1.28	0.50	0.50	0.25
3	Mean CE (%)	14.07	4.40	4.03	6.37	27.96	28.58	8.28
	Median CE (%)	14.18	0.56	1.43	0.21	30.98	29.67	2.76
	Avg. Time (sec)	1.89	5.39	4.05	2.16	0.82	1.15	0.51

Table 2.2: CE and computational time of algorithms on Hopkins155 dataset. L is the number of clusters (motions). The numbers in the parentheses represent the number of neighbors for each point collected in the corresponding algorithms.

L	Algorithms	K-means	K-flats	SSC	SSC-OMP	TSC	NSN+Spectral
2	Mean CE (%)	45.98	37.62	1.77	4.45	11.84	1.71
	Median CE (%)	47.66	39.06	0.00	1.17	1.56	0.78
	Avg. Time (sec)	-	15.78	37.72	0.45	0.33	0.78
3	Mean CE (%)	62.55	45.81	5.77	6.35	20.02	3.63
	Median CE (%)	63.54	47.92	1.56	2.86	15.62	3.12
	Avg. Time (sec)	-	27.91	49.45	0.76	0.60	3.37
5	Mean CE (%)	73.77	55.51	4.79	8.93	11.90	5.81
	Median CE (%)	74.06	56.25	2.97	5.00	33.91	4.69
	Avg. Time (sec)	-	52.90	74.91	1.41	1.17	5.62
8	Mean CE (%)	79.92	60.12	7.75	12.90	38.55	8.46
	Median CE (%)	80.18	60.64	5.86	12.30	40.14	7.62
	Avg. Time (sec)	-	101.3	119.5	2.84	2.24	11.51
10	Mean CE (%)	82.68	62.72	9.43	15.32	39.48	9.82
	Median CE (%)	82.97	62.89	8.75	17.11	39.45	9.06
	Avg. Time (sec)	-	134.0	157.5	5.26	3.17	14.73

Table 2.3: CE and computational time of algorithms on Extended Yale B dataset. For each number of clusters (faces) L, the algorithms ran over 100 random subsets drawn from the overall 38 clusters.

Chapter 3

Convex Relaxation and Alternating Minimization for Collaborative Ranking

Rank aggregation is a classic problem which learns the global ordering of a set of entities from their partial orderings. To this end, one usually assumes an underlying *score vector* which determines the global ordering and a model which gives samples of partial orderings according to this vector. This problem is interesting in several applications such as recommender systems and webpage ranking. However, in many cases, it is not desirable to learn one single ordering. Recommender systems have to recommend items to each user based on the user's *personalized* preference ordering. In webpage ranking, one should have different rankings of webpages depending on the users even if they have the same query terms. In this personalization perspective, it is more reasonable to learn a preference ordering for each user. This problem is referred to as *collaborative ranking*.

A naive method is to learn each user's preference ordering independently from the sample provided by the users. However, this has two main issues: *(a)* The number of samples provided by the users are very limited com-

[1]This work has been published as Dohyung Park, Joe Neeman, Jin Zhang, Sujay Sanghavi, and Inderjit S. Dhillon, "Preference Completion: Large-scale Collaborative Ranking from Pairwise Comparisons," in Proceedings of International Conference on Machine Learning (ICML), 2015. My contributions are design of the large-scale non-convex algorithm, and design and implementation of experiments.

pared to the number of all items. *(b)* Since it does not exploit the similarity of preferences between users, the method may be inefficient. As seen from the collaborative filtering and the matrix completion literature, a low-rank matrix model is a reasonable heuristic that we can assume. Then the formal description of the problem can be as follows: Given sample partial orderings from users, we fit a low-rank matrix where each row corresponds to a user's preference score vector.

To simplify the problem, we especially consider collaborative ranking from *p*airwise comparisons. The problem can be stated as follows: Given a set of items, a set of users, and non-numerical *pairwise comparison* data, find the underlying preference ordering of the users. The observed pairwise comparisons are of the form "user i prefers item j over item k", for different ordered user-item-item triples i, j, k. Pairwise preference data is wide-spread; indeed, almost any setting where a user is presented with a menu of options – and chooses one of them – can be considered to be providing a pairwise preference between the chosen item and every other item that is presented.

Essentially, we fit a low-rank users $\times$ items *score matrix* X to pairwise comparison data by trying to ensure that $X_{ij} - X_{ik}$ is positive when user i prefers item j to item k.

3.1 Contribution

We present two algorithms to infer the score matrix X from training data; once inferred, this can be used for predicting future preferences. While there has been some recent work on fitting low-rank score matrices to pairwise preference data (which we review and compare to below), in this chapter we present the following two contributions.

A statistical analysis for the convex relaxation. We bound the *generalization error* of the solution to our convex program. Essentially, we show that the minimizer of the empirical loss also almost minimizes the true expected loss. We also give a lower bound showing that our error rate is sharp up to logarithmic factors.

A non-convex algorithm based on alternating minimization. We provide a non-convex algorithm that we call Alternating Support Vector Machine (AltSVM). This non-convex algorithm is more practical than the convex program in a large-scale setting; it explicitly parameterizes the low-rank matrix in factored form and minimizes the hinge loss. Crucially, each step in this algorithm can be formulated as a standard SVM that updates one of the two factors; the algorithm proceeds by alternating updates to both factors. We apply a stochastic version of dual coordinate descent [73, 140] with lock-free parallelization. This exploits the problem structure and ensures it parallelizes well. We show that our algorithm outperforms several existing collaborative ranking algorithms in both speed and prediction accuracy., and it achieves significant speedups as the number of cores increases.

3.2 Related Work

The rank aggregation from pairwise comparisons to learn a single ordering has been considered since a few decades ago. Let us briefly introduce recent work on this problem. [84] and [4] consider an active query model with noiseless responses; [85] give an algorithm for exactly recovering the true ranking under a low-rank assumption similar to ours, while [4] approximately recovers the true ranking without such an assumption. [166] and [124] learn

a ranking from noisy pairwise comparisions; [124] consider a Bradley-Terry-Luce model similar to ours and attempt to learn an underlying score vector, while [166] get by without structure assumptions, but only attempt to learn the ranking itself. [63] considered a problem to learn a single ranking given a more generalized partial rankings from the Plackett-Luce model and provided a minimax-optimal algorithm.

There are a few results that considers learning multiple rankings from comparisons [117, 130], which are the most related work to ours. They also considered the problem of learning preference orderings for multiple users. [117] analyzed a convex program with nuclear norm regularization in the setting where the pairwise comparisons are drawn from the Bradley-Terry-Luce model. [130] considered partial orderings from the multinomial logit model and learned the rankings by solving a nuclear norm regularized convex program similarly to [117].

There is also a line of work on collaborative ranking that considers learning multiple rankings for multiple users but from (ordinal or binary) relevance scores. [167] attempted to directly optimize Normalized Discounted Cumulative Gain (NDCG), a widely used performance measure for ranking problems. [11], and [159] converted this problem into a learning-to-rank problem and solved it using the existing algorithms. While these works considered the low-rank matrix model, different models are proposed by [171] and [103]. [171] proposed a tensor model to rank items for different queries and users, and [103] proposed a weighted sum of low-rank matrix models. There are some algorithms which take pairwise comparisons from this relevance scores and learn the rankings [135, 110]. [174] took a purely optimization-based approach. Rather than assuming a probabilistic model, they minimized a convex

objective using the hinge loss on a low-rank matrix. In a slightly different model, [75] and [141] consider the problem of learning from latent feedback.

We finally note that there is another hugh literature from the larning-to-rank community [111], which considers the setting where one learns a ranking function which depend on each item's feature vector. Given a set of pairs of a feature vector and a relevance score, the ranking function is learned so that the feature vectors with larger score is ranked more highly. There have also been algorithms [71, 87] based on pairwise comparisons between the training samples. One of our algorithms is related to these algorithms, while we assume the feature vectors of the items to be latent and need to be learned.

3.3 Problem Setup

The task is to estimate rankings of multiple users on multiple items. We denote the numbers of users by d_1, and the number of items by d_2. We are given a set of triples $\Omega \subset [d_1] \times [d_2] \times [d_2]$, where the preference of user i between items j and k is observed if $(i, j, k) \in \Omega$. The observed comparison is then given by $\{Y_{ijk} \in \{1, -1\} : (i, j, k) \in \Omega\}$ where $Y_{ijk} = 1$ if user i prefers item j over item k, and $Y_{ijk} = -1$ otherwise. Let $\Omega_i = \{(j, k) : (i, j, k) \in \Omega\}$ denote the set of item pairs that user i has compared.

We predict rankings for multiple users by estimating a score matrix $X \in \mathbb{R}^{d_1 \times d_2}$ such that $X_{ij} > X_{ik}$ means that user i prefers item j over item k. Then the sorting order for each row provides the predicted ranking for the corresponding user.

We propose (as have others) that X is low-rank or close to low-rank, the intuition being that each user bases their preferences on a small set of features

that are common among all the items. Then the empirical risk minimization (ERM) framework can naturally be formulated as

$$\begin{aligned} \underset{X}{\text{minimize}} \quad & \sum_{(i,j,k)\in\Omega} \mathcal{L}(Y_{ijk}(X_{ij} - X_{ik})) \\ \text{subject to} \quad & \text{rank}(X) \leq r \end{aligned} \tag{3.1}$$

where $\mathcal{L}(\cdot)$ is a monotonically non-increasing loss function which induces $X_{ij} > X_{ik}$ if $Y_{ijk} = 1$, and $X_{ij} < X_{ik}$ otherwise. (e.g., hinge loss, logistic regression loss, etc.)

3.4 Convex Relaxation

Solving (3.1) is NP-hard because of the rank constraint. Our first method is the convex relaxation of (3.1), which involves a nuclear norm constraint.

$$\begin{aligned} \underset{X}{\text{minimize}} \quad & \sum_{(i,j,k)\in\Omega} \mathcal{L}(Y_{ijk}(X_{ij} - X_{ik})) \\ \text{subject to} \quad & \|X\|_* \leq \sqrt{\lambda d_1 d_2} \end{aligned} \tag{3.2}$$

Here, for any matrix X, the nuclear/trace norm $\|X\|_*$ denotes the sum of its singular values; it is a well-recognized convex surrogate for low-rank structure (most famously in matrix completion).

The only parameter of this algorithm is λ, which governs the trade-off between better optimizing the likelihood of the observed data, and the strictness in imposing approximate low-rank structure. Since we motivated our algorithm with the assumption that X has low rank, we should point out how our algorithm's parameter λ compares to the rank: note that if X is

a $d_1 \times d_2$ rank-r matrix whose largest absolute entry is bounded by C then $\|X\|_* \leq \sqrt{r}\|X\|_F \leq C\sqrt{rd_1d_2}$. In other words, λ is a parameter that takes into account both the rank of X and the size of its elements, and it is roughly proportional to the rank.

3.4.1 Excess risk bounds

We analyze (3.2) by assuming a standard model for pairwise comparisons. Then we provide a statistical guarantee of the method under the model. Assume that each user-item-item triple (i, j, k) independently belongs to Ω with probability $p_{i,j,k}$, and let $m = \sum_{i,j,k} p_{i,j,k}$ be the expected size of Ω. We will assume that the $p_{i,j,k}$ are approximately balanced in the sense that no user-item pair is observed too frequently:

Assumption 3.1. *There is a constant $\kappa > 0$ such that for every i, j,*

$$\sum_k p_{i,j,k} \leq \kappa \frac{m}{d_1 d_2}.$$

Note that if $\kappa = 1$ in Assumption 3.1 then the $p_{i,j,k}$ are all equal, meaning that each user-item-item triple has an equal chance to be observed.

In order to state our error bounds, we first introduce some notation: let $\mathbb{P}_X$ be the distribution of $\{Y_{i,j,k} : 1 \leq i \leq d_1, 1 \leq j < k \leq d_2\}$ (i.e. the complete distribution of all pairwise preferences, even those that are not observed).

Our main upper bound shows that if m is sufficiently large then our algorithm finds a solution with almost minimal risk. Given a loss function $\mathcal{L}$, define the expected risk of X by

$$R(X) = \frac{1}{d_1 d_2^2} \sum_{i=1}^{d_1} \sum_{j,k=1}^{d_2} \mathbb{E}_{X^*} \mathcal{L}(Y_{ijk}(X_{ij} - X_{ik})),$$

where the expectation is with respect to the distribution parametrized by the true parameters X^*.

Theorem 3.2. *Suppose that $\mathcal{L}$ is 1-Lipschitz, and let Y and Ω be distributed as $\mathbb{P}_{X^*}$ for some $d_1 \times d_2$ matrix X^*. Under Assumption 3.1,*

$$\mathbb{E}R(\hat{X}) \leq \inf_{\{X:\|X\|_* \leq \sqrt{\lambda d_1 d_2}\}} \mathbb{E}R(X) + C\kappa\sqrt{\frac{\lambda(d_1+d_2)}{m}}\log(d_1+d_2),$$

where C is a universal constant.

We recall that the parameter λ is related to rank in that if X is a $d_1 \times d_2$ rank-r matrix whose largest absolute entry is bounded by C then $\|X\|_* \leq \sqrt{r}\|X\|_F \leq C\sqrt{rd_1d_2}$. In other words, λ is a parameter that takes into account both the rank of X^* and the size of its elements, and it is roughly proportional to the rank. In particular, Theorem 3.2 shows that once we observe $m \sim r(d_1+d_2)\log^2(d_1+d_2)$ pairwise comparisons, then we can accurately estimate the probability of any user preferring any item over any other. In other words, we need to observe about $r(1+d_2/d_1)\log^2(d_1+d_2)$ comparisons per user, which is substantially less than the $rd_2\log(d_2)$ comparisons that we would have required if each user were modelled in isolation. Moreover, our lower bound (below) shows that at least $r(1+d_2/d_1)$ comparisons per user are required, which is only a logarithmic factor from the upper bound.

Theorem 3.3. *Suppose that $\mathcal{L}'(0) < 0$. Let $\mathcal{A}$ be any algorithm that receives $\{Y_{i,j,k} : (i,j,k) \in \Omega\}$ as input and produces $\hat{X}$ as output. For any $\lambda \geq 1$ and $m \geq d_1+d_2$, there exists X^* with $\|X^*\|_* \leq \sqrt{\lambda d_1 d_2}$ such that when Y and Ω are distributed according to $\mathbb{P}_{X^*}$ then with probability at least $\frac{1}{2}$,*

$$\mathbb{E}R(\hat{X}) \geq R(X^*) + c\min\left\{1, \sqrt{\frac{\lambda(d_1+d_2)}{m}}\right\},$$

where $c > 0$ is a constant depending only on $\mathcal{L}$.

Together, Theorems 3.2 and 3.3 show that (up to logarithmic factors) if X^* has rank r then about $r(1 + d_2/d_1)$ comparisons per user are necessary and sufficient for learning the users' preferences.

3.4.2 Maximum likelihood estimation for the BTL model

Recall the classical Bradley-Terry-Luce model [23, 118] for pairwise preferences of a single user, which assumes that the probability of item j being preferred over k is given by a logistic of the difference of the underlying preference scores of the two items. For multiple users, we assume that there is some true score matrix $X^* \in \mathbb{R}^{d_1 \times d_2}$ and

$$\Pr(Y_{ijk} = 1) \;=\; \frac{\exp(X^*_{ij} - X^*_{ik})}{1 + \exp(X^*_{ij} - X^*_{ik})}.$$

By specializing the loss function $\mathcal{L}$, Theorem 3.2 has a simple corollary for maximum-likelihood estimation of X^*. Recall that if μ and ν are two probability distributions on a finite set S the the Kullback-Leibler divergence between them is

$$D(\mu\|\nu) = \sum_{s \in S} \mu(s) \log \frac{\mu(s)}{\nu(s)},$$

under the convention that $0 \log 0 = 0$. We recall that although $D(\cdot\|\cdot)$ is not a metric it is always non-negative, and that $D(\mu\|\nu) = 0$ implies $\mu = \nu$.

Corollary 3.4. *Let Y and Ω be distributed as $\mathbb{P}_{X^*}$ for some $d_1 \times d_2$ matrix X^*. Define the loss function $\mathcal{L}$ by $\mathcal{L}(z) = \log(1+\exp(z)) - z$. Under Assumption 3.1,*

$$\frac{1}{d_1 d_2^2} \sup_{\{X : \|X\|_* \le \sqrt{\lambda d_1 d_2}\}} D(\mathbb{P}_{X^*}\|\mathbb{P}_{\hat{X}}) - D(\mathbb{P}_{X^*}\|\mathbb{P}_X) \le C\kappa\sqrt{\frac{\lambda(d_1+d_2)}{m}}\log(d_1+d_2),$$

where C is a universal constant.

Note that the loss function in Corollary 3.4 is exactly the negative logarithm of the logistic function, and so $\hat{X}$ in Corollary 3.4 is the maximum-likelihood estimate for X^*. Thus, Corollary 3.4 shows that the distribution induced by the maximum-likelihood estimator is close to the true distribution in Kullback-Leibler divergence.

3.5 Large-scale Non-convex Implementation

While the convex relaxation is statistically near optimal, it is not ideal for large-scale datasets because it requires the solution of a convex program with $d_1 \times d_2$ variables. In this section we develop a non-convex variant which both scales and parallelizes very well, and has better empirical performance as compared to several existing empirical baseline methods.

Our approach is based on the following steps:

- We represent the low-rank matrix in explicit factored form $X = UV^\top$ and replace the regularizer appropriately. This results in a non-convex optimization problem in $U \in \mathbb{R}^{d_1 \times r}$ and $V \in \mathbb{R}^{d_2 \times r}$, where r is the rank parameter.

- We solve the non-convex problem by alternating between updating U while keeping V fixed, and vice versa. With the hinge loss (which we found works best in experiments), each of these becomes an SVM problem - hence we call our algorithm AltSVM.

- The problem is of course not symmetric in U and V because users rank items but not vice versa. For the U update, each user vector naturally decouples and can be done in parallel (and in fact just reduces to the case of rankSVM [87]).

- For the V update, we show that this can *also* be made into an SVM problem; however it involves coupling of all item vectors, and all user ratings. We employ several tricks (detailed below) to speed up and effectively parallelize this step.

The non-convex problem can be written as

$$\min_{U,V} \sum_{(i,j,k)\in\Omega} \mathcal{L}(Y_{ijk} \cdot u_i^\top (v_j - v_k)) + \frac{\lambda}{2}(\|U\|_F^2 + \|V\|_F^2) \tag{3.3}$$

where we replace the nuclear norm regularizer using the property $\|X\|_* = \min_{X=UV^\top} \frac{1}{2}(\|U\|_F^2 + \|V\|_F^2)$ [145]. $u_i^\top$ and $v_i^\top$ denote the ith rows of U and V, respectively. While this is a non-convex algorithm for which it is hard to find the global optimum, it is computationally more efficient since only $(d_1 + d_2)r$ variables are involved. We propose to use L2 hinge loss, i.e., $\mathcal{L}(x) = \max(0, 1 - x)^2$.

In the alternating minimization of (3.3), the subproblem for U is to solve

$$U \leftarrow \arg\min_{U\in\mathbb{R}^{d_1\times r}} \sum_{(i,j,k)\in\Omega} \mathcal{L}(Y_{ijk} \cdot u_i^\top (v_j - v_j)) + \frac{\lambda}{2}\|U\|_F^2, \tag{3.4}$$

while V is fixed. This can be decomposed into n independent problems for u_i's where each solves for

$$u_i \leftarrow \arg\min_{u\in\mathbb{R}^r} \frac{\lambda}{2}\|u\|_2^2 + \sum_{(j,k)\in\Omega_i} \mathcal{L}(Y_{ijk} \cdot u^\top (v_j - v_k). \tag{3.5}$$

This part is in general a small-scale problem as the dimension is r, and the sample size is $|\Omega_i|$ for each user i.

On the other hand, solving for V with fixed U can be written as

$$V \leftarrow \arg\min_{V \in \mathbb{R}^{d_2 \times r}} \left\{ \frac{\lambda}{2} \|V\|_F^2 + \sum_{(i,j,k) \in \Omega} \mathcal{L}(\langle V, A^{(u,i,j)} \rangle) \right\} \tag{3.6}$$

where $A^{(i,j,k)} \in \mathbb{R}^{d_2 \times r}$ is such that the lth row of $A^{(i,j,k)}$ is $Y_{ijk} \cdot u_i^\top$ if $l = j$, $-Y_{ijk} \cdot u_i^\top$ if $l = k$, and 0 otherwise. It is a much larger SVM problem than (3.5) as the dimension is $d_2 r$ and the sample size is $|\Omega|$.

We note that the feature matrices $\{A^{(i,j,k)} : (i,j,k) \in \Omega\}$ are highly sparse since in each feature matrix only $2r$ out of the $d_2 r$ elements are nonzero. This motivates us to apply the stochastic dual coordinate descent algorithm [73, 140], which not only converges fast but also takes advantages of feature sparsity in linear SVMs. Each coordinate descent step takes $O(r)$ computation, and iterations over $|\Omega|$ coordinates provide linear convergence [140].

Now we describe the dual problems of our two subproblems explicitly. Let $\alpha \in \mathbb{R}^{|\Omega_i|}$ denote the dual vector for (3.5), in which each coordinate is denoted by α_{ijk} where $(j,k) \in \Omega_i$. Then the dual problem of (3.5) is to solve

$$\min_{\alpha \in \mathbb{R}^{|\Omega_i|}, \alpha \geq 0} \frac{1}{2} \left\| \sum_{(j,k) \in \Omega_i} \alpha_{ijk} Y_{ijk} (v_j - v_k) \right\|_2^2 + \frac{1}{\lambda} \sum_{(j,k) \in \Omega_i} \mathcal{L}^*(-\lambda \alpha_{ijk}) \tag{3.7}$$

where $\mathcal{L}^*(z)$ is the convex conjugate of $\mathcal{L}$. At each coordinate descent step for α_{ijk}, we find the value of α_{ijk} minimizing (3.7) while all the other variables are fixed. If we maintain $u_i = \sum_{(j,k) \in \Omega_i} \alpha_{ijk} Y_{ijk} (v_j - v_k)$, then the coordinate descent step is simply to find δ^* minimizing

$$\frac{1}{2} \|u_i + \delta^* Y_{ijk} (v_j - v_k)\|_2^2 + \frac{1}{\lambda} \mathcal{L}^*(-\lambda(\alpha_{ijk} + \delta^*)) \tag{3.8}$$

and update $\alpha_{ijk} \leftarrow \alpha_{ijk} + \delta^*$.

The dual problem of (3.6) is to solve

$$\min_{\beta \in \mathbb{R}^{|\Omega|}, \beta \geq 0} \frac{1}{2} \left\| \sum_{(i,j,k) \in \Omega} \beta_{ijk} A^{(i,j,k)} \right\|_F^2 + \frac{1}{\lambda} \sum_{(i,j,k) \in \Omega} \mathcal{L}^*(-\lambda \beta_{ijk}) \tag{3.9}$$

where β is the dual vector for the subproblem (3.6). Similarly to α_{ijk}, the coordinate descent step for β_{ijk} is to replace β_{ijk} by $\beta_{ijk}+\delta^*$ where δ^* minimizes

$$\frac{1}{2} \left(\|v_j + \delta^* Y_{ijk} u_i\|_2^2 + \|v_k - \delta^* Y_{ijk} u_i\|_2^2 \right) + \mathcal{L}^*(-\lambda(\beta_{ijk} + \delta^*)), \tag{3.10}$$

and maintain $V = \sum_{(i,j,k) \in \Omega} \beta_{ijk} Y_{ijk} A^{(i,j,k)}$.

The detailed description of AltSVM is presented in Algorithm 5. In each subproblem, we run the stochastic dual coordinate descent, in which a pairwise comparison $(i, j, k) \in \Omega$ is chosen uniformly at random, and the dual coordinate descent for α_{ijk} or β_{ijk} is computed. We note that each coordinate descent step takes the same $O(r)$ computational cost in both subproblems, while the subproblem sizes are much different.

3.5.1 Parallelization

For each subproblem, we parallelize the stochastic dual coordinate descent algorithm asynchronously without locking. Given T processors, each processor randomly sample a triple $(i, j, k) \in \Omega$ and update the corresponding dual variable and the user or item vectors. We note that this update is for a sparse subset of the parameters. In the user part, a coordinate descent step for one sample updates only r out of the rd_1 variables. In the item part, one coordinate descent step for a sample update only $2r$ out of the rd_2 variables. This motivates us not to lock the variables when updated, so that we ignore the conflicts. This lock-free parallelism is shown to be effective in [129] for stochastic gradient descent (SGD) on the sum of sparse functions. Moreover, in [74],

it is also shown that the stochastic dual coordinate descent scales well without locking. We implemented the algorithm using the OpenMP framework. In our implementations, we also parallelized steps 3 and 13 of Algorithm 5. We show in the next section that our proposed algorithm scales up favorably.

3.6 Experimental results

Now we demonstrate that our algorithm performs well as a collaborative ranking method on rating data. We used the datasets specified in Table 3.1. Given a training set of ratings for each user, our algorithm will only use non-tying pairwise comparisons from the set, while other competing algorithms use the ratings themselves. Hence, they have more information than ours. The competing algorithms are those with publicly available codes provided by the authors.

- CofiRank [167][2] This algorithm uses alternating minimization to directly optimize NDCG.
- Local Collaborative Ranking (LCR) [103][3] : The main idea is to predict preferences from the weighted sum of multiple low-rank matrices model.
- RobiRank [178][4] : This algorithm uses stochastic gradient descent to optimize the loss function motivated from robust binary classification.

[2] `http://www.cofirank.org`, The dimension and the regularization parameter are set as suggested in the paper. For the rest of the parameters, we left them as provided.

[3] `http://prea.gatech.edu`, We run the code with each of the 48 sets of loss function and parameters given in the main code, and the best result is reported. We could not run this algorithm on the Netflix dataset due to time constraint.

[4] `https://bitbucket.org/d_ijk_stra/robirank`, We used the part for collaborative ranking from binary relevence score. We left the parameter settings as provide with the implementation.

	MovieLens1m	MovieLens10m	Netflix
Users	6,040	71,567	480,000
Items	3,900	10,681	17,000
Ratings	1,000,209	10,000,054	100,000,000

Table 3.1: Datasets to be used for simulation

- Global Ranking : To see the effect of personalized ranking, we compare the results with a global ranking of the items. We fixed U to all ones and solved for V.

The algorithms are compared in terms of two standard performance measures of ranking, which are NDCG and Precision@K. NDCG@K is the ranking measure for numerical ratings. NDCG@K for user i is defined as

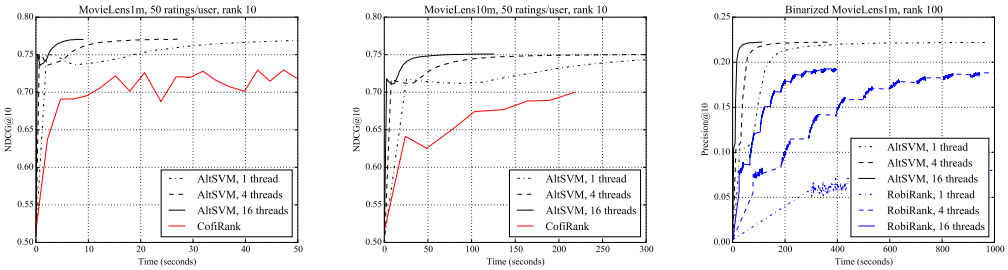
$$\text{NDCG@}K(i) = \frac{\text{DCG@}K(i, \pi_i)}{\text{DCG@}K(i, \pi_i^*)}$$

where

$$\text{DCG@}K(i, \pi_i) = \sum_{k=1}^{K} \frac{2^{M_{i\pi_i(k)}} - 1}{\log_2(k+1)},$$

and $\pi_u(k)$ is the index of the kth ranked item of $\mathcal{T}_i$ in our prediction. M_{ij} is the true rating of item j by user i in the given dataset, and π_u^* is the permutation that maximizes DCG@K. This measure counts only the top K items in our predicted ranking and put more weights on the prediction of highly ranked items. We measured NDCG@10 in our experiments. Precision@K is the ranking measure for binary ratings. Precision@K for user i is defined as

$$\text{Precision@}K(i) = \frac{1}{K} \sum_{j \in \mathcal{P}_K(i)} M_{ij}$$



(a) (b) (c)

Figure 3.1: NDCG@10 and Precision@10 over time for different algorithms.

where M_{ij} is the binary rating on item j by user i given in the dataset. This counts the number of relevant items in the predicted top K recommendation. These two measures are averaged over all of the users.

We first compare our algorithm with numerical rating based algorithms, CofiRank and LCR. We follow the standard setting that are used in the collaborative ranking literature [167, 11, 159, 103]. For each user, we subsampled N ratings, used them for training, and took the rest of the ratings for test. The users with less than $N+10$ ratings were dropped out. Table 3.2 compares AltSVM with numerical rating based algorithms. While $N = 20$ is too small so that a global ranking provides the best NDCG, our algorithm performs the best with larger N. We also ran our algorithm with subsampled pairwise comparions with the largest numerical gap (AltSVM-sub), which are as many as N for each user (the number of numerical ratings used in the other algorithms). Even with this, we could achieve better NDCG. We can also observe that the statistical performance is better with the hinge loss than with the logistic loss.

We have also experimented with collaborative ranking on binary rat-

Datasets	N	AltSVM	AltSVM-sub	AltSVM-logistic	Global	CofiRank	LCR
MovieLens1m	20	0.7308	0.6998	0.7125	**0.7500**	0.7333	0.7007
	50	**0.7712**	0.7392	0.7141	0.7501	0.7441	0.7081
	100	**0.7902**	0.7508	0.7446	0.7482	0.7332	0.7151
MovieLens10m	20	0.7059	0.7053	0.7031	**0.7264**	0.7076	0.6977
	50	**0.7508**	0.7212	0.7115	0.7176	0.6977	0.6940
	100	**0.7692**	0.7248	0.7292	0.7101	0.6754	0.6899
Netflix	20	0.7132	0.6822	-	**0.7605**	0.6615	-
	50	**0.7642**	0.7111	-	**0.7640**	0.6527	-
	100	**0.8007**	0.7393	-	0.7656	0.6385	-

Table 3.2: NDCG@10 on different datasets, for different numbers of observed ratings per user.

	AltSVM			RobiRank
Precision@	$C = 1000$	$C = 2000$	$C = 5000$	
1	0.2165	0.2973	**0.3635**	0.3009
2	0.1965	0.2657	**0.3297**	0.2695
5	0.1572	0.2097	**0.2697**	0.2300
10	0.1265	0.1709	**0.2223**	0.1922
100	0.0526	0.0678	**0.0819**	0.0781

Table 3.3: Precision@K on the binarized MovieLens1m dataset.

ings. We compare our algorithm against RobiRank [178], which is a recently proposed algorithm for collaborative ranking with binary ratings. We ran an experiment on a *binarized* version of the Movielens1m dataset. In this case, the movies rated by a user is assumed to be relevant to the user, and the other items are not. Since it is inefficient to take all possible comparisons which are in average a half million per user, we subsampled C comparisons for each user. Both algorithms are set to estimate rank-100 matrices. Table 3.3 shows that our algorithm provides better performance than RobiRank.

We now show the computational speed and scalability of our practical algorithm, AltSVM. The experiments were run on a single 16-core machine in the Stampede Cluster at University of Texas.

Figures 3.1a and 3.1b show NDCG@10 over time of our algorithms with

1, 4, and 16 threads, compared to CofiRank. Figure 3.1c shows Precision@10 over time of our algorithm with $C = 5000$. We note that our algorithm converges faster, while the sample size $|\Omega|$ for our algorithm is larger than the number of training ratings that are used in the competing algorithms.

Algorithm 5 Alternating Support Vector Machine (AltSVM)

Input: Ω, $\{Y_{ijk} : (i,j,k) \in \Omega\}$, and $\lambda \in \mathbb{R}^+$
Output: $U \in \mathbb{R}^{d_1 \times r}$, $V \in \mathbb{R}^{d_2 \times r}$

Initialize U, and set $\alpha, \beta \leftarrow 0 \in \mathbb{R}^{|\Omega|}$
while not converged **do**
    $v_j \leftarrow \sum_{(i,j,k)\in\Omega} \beta_{ijk} Y_{ijk} u_i$
        $- \sum_{(i,k,j)\in\Omega} \beta_{ikj} Y_{ikj} u_i$, $\forall j \in [d_2]$
    for all threads $t = 1, \ldots, T$ **in parallel do**
        for $s = 1, \ldots, S$ **do**
            Choose $(i,j,k) \in \Omega$ uniformly at random
            Find δ^* minimizing (3.10).
            $\beta_{ijk} \leftarrow \beta_{ijk} + \delta^*$
            $v_j \leftarrow v_j + \delta^* Y_{ijk} u_i$
            $v_k \leftarrow v_k - \delta^* Y_{ijk} u_i$
        end for
    end for
    $u_i \leftarrow \sum_{(i,j,k)\in\Omega} \alpha_{ijk} Y_{ijk} (v_j - v_k)$, $\forall i \in [d_1]$
    for all threads $t = 1, \ldots, T$ **in parallel do**
        for $s = 1, \ldots, S$ **do**
            Choose $(i,j,k) \in \Omega$ uniformly at random.
            Find δ^* minimizing (3.8).
            $\alpha_{ijk} \leftarrow \alpha_{ijk} + \delta^*$
            $u_i \leftarrow u_i + \delta^* Y_{ijk} (v_j - v_k)$
        end for
    end for
end while

Chapter 4

Non-convex Gradient Descent for Low-rank Matrix Optimization

We study matrix problems of the form:

$$\underset{X \in \mathbb{R}^{m \times n}}{\text{minimize}} \quad f(X), \tag{4.1}$$

where the minimizer $X^\star \in \mathbb{R}^{m \times n}$ is rank-$r^\star$ ($r^\star \leq \min\{m, n\}$), or *approximately* low rank, *i.e.*, $\|X^\star - X^\star_{r^\star}\|_F$ is sufficiently small, for $X^\star_{r^\star}$ being the best rank-$r^\star$ approximation of $X^\star$. In our discussions, f is a smooth convex function; further assumptions on f will be described later in the text. Note, in particular, that in the absence of further assumptions, $X^\star$ may not be unique.

Specific instances of (4.1), where the solution is assumed low-rank, appear in several applications in diverse research fields; a non-exhaustive list includes factorization-based recommender systems [144, 136, 47, 17, 93, 78], multi-label classification tasks [3, 18, 32, 119, 161, 170], dimensionality reduction techniques [138, 40, 88, 154, 61, 113], density matrix estimation of quantum systems [1, 60, 89], phase retrieval applications [27, 160], sensor localization [20, 168] and protein clustering [116] tasks, image processing problems

[1]This work is in preparation for journal publication, and it has been published in part as Dohyung Park, Anastasios Kyrillidis, Constantine Caramanis, and Sujay Sanghavi, "Finding low-rank solutions to convex smooth problems via the Burer-Monteiro approach," in Proceedings of 54th Annual Allerton Conference on Communication, Control, and Computing, 2016. My contributions are design of algorithms, statement and proofs of main results.

[6], as well as applications in system theory [56], just to name a few. Thus, it is critical to devise easy-to-implement, efficient and provable algorithms that solve (4.1), taking into consideration such near low-rank structure of $X^\star$.

While, in general, imposing a low-rank constraint may result in an NP-hard problem, (4.1) with a rank-constraint can be solved in polynomial-time in numerous applications, due to the special structure of the objective f. A prime –and by now well-known– example of this is the matrix sensing/matrix completion problem [31, 134, 78] (we discuss this further in the following section). There, f is a least-squares objective function and the measurements satisfy the appropriate restricted isometry/incoherence assumptions. In such a scenario, the optimal low-rank $X^\star$ can be recovered in polynomial time, by solving (4.1) with a rank-constraint [80, 16, 12, 104, 97, 147], or by solving its convex nuclear-norm relaxation, as in [108, 15, 26, 14, 35, 180].

In view of the above, although the resulting algorithms have attractive convergence rates, they directly manipulate the $n \times n$ variable matrix X, which in itself is computationally expensive. Specifically, each iteration typically requires computing the top-r singular value/vectors of the matrix. As the size n of the matrix scales, this computational demands at each iteration can be prohibitive.

Optimizing over factors. In this chapter, we follow a different path: a rank-r matrix $X \in \mathbb{R}^{m\times n}$ can be written as a product of two matrices $UV^\top$, where $U \in \mathbb{R}^{m\times r}$ and $V \in \mathbb{R}^{n\times r}$. Based on this premise, we consider optimizing f over the U and V space. Particularly, we are interested in solving (4.1) via the parametrization:

$$\underset{U\in\mathbb{R}^{m\times r},\ V\in\mathbb{R}^{n\times r}}{\text{minimize}} \quad f(UV^\top) \qquad \text{where } r \leq \text{rank}(X^\star) \leq \{m,\ n\}. \tag{4.2}$$

Note that (4.2) and (4.1) are equivalent in the case $\text{rank}(X^\star) = r$.[2] Observe that such parameterization leads to a very specific kind of non-convexity in f. Even more importantly, proving convergence for these settings becomes a harder task, due to the bi-linearity of the variable space.

Motivation. Our motivation for studying (4.2) originates from large-scale problem instances: when r is much smaller than $\min\{m, n\}$, factors $U \in \mathbb{R}^{m\times r}$ and $V \in \mathbb{R}^{n\times r}$ contain far fewer variables to maintain and optimize, than $X = UV^\top$. Thus, by construction, such parametrization also makes it easier to update and store the iterates U, V.

Even more importantly, observe that $UV^\top$ reformulation automatically encodes the rank constraint. Standard approaches, that operate in the original variable space, either enforce the $\text{rank}(X) \leq r$ constraint at every iteration or involve a nuclear-norm projection. Doing so requires computing a truncated SVD[3] per iteration, which can get cumbersome in large-scale settings. In stark contrast, working with $f(UV^\top)$ replaces singular value computation per iteration with matrix-matrix multiplication $UV^\top$. Thus, such non-conventional approach turns out to be a more practical and realistic option, when the dimension of the problem is large. We defer this discussion to Section 4.8.1 for some empirical evidence of the above.

[2]Here, by equivalent, we mean that the set of global minima in (4.2) contains that of (4.1). It remains an open question though whether the reformulation in (4.2) introduces spurious local minima in the factored space for the majority of f cases.

[3]This holds in the best scenario; in the convex case, where the rank constraint is "relaxed" by the nuclear norm, the projection onto the nuclear-norm ball often requires a full SVD calculation.

4.1 Contributions

While the computational gains are apparent, such bi-linear reformulations $X = UV^\top$often lack theoretical guarantees. Only recently, there have been attempts in providing answers to when and why such non-convex approaches perform well in practice, in the hope that they might provide a new algorithmic paradigm for designing faster and better algorithms; see [79, 5, 152, 184, 38, 19, 183, 146, 185].

As we detail below and in greater detail in Section 4.3, our work is more general, addressing important settings that could not (as far as we know) be treated by the previous literature. Our contributions can be summarized as follows:

(i) We study a gradient descent algorithm on the non-convex formulation given in (4.2) for *non-square* matrices. We call this *Bi-Factored Gradient Descent* (`BFGD`). Recent developments (cited above, and see Section 4.3 for further details) rely on properties of f for special cases [146, 152, 185, 183], and their convergence results seem to rely on this special structure. In this work, we take a more generic view of such factorization techniques, closer to results in convex optimization. We provide local convergence guarantees for general smooth (and strongly convex) f objectives.

(ii) In particular, when f is only smooth (not strictly convex), we show that a simple lifting technique leads to a local sublinear rate convergence guarantee, using results from that of the square and PSD case [19]. Moreover, we provide a simpler and improved proof than [19], which requires a weaker initial condition.

(*iii*) When f is both strongly convex and smooth, results from the PSD case do not readily extend as above. In such cases, of significant importance is the use of a regularizer in the objective, that restricts the geometry of the problem at hand. Here, we improve upon [152, 185, 176] –where such a regularizer was used only for the cases of matrix sensing/completion and robust PCA– and solve a different formulation to prove a local linear rate convergence guarantee. Our proof technique proves a significant generalization: using any smooth and strongly convex regularizers on the term $(U^\top U - V^\top V)$, with optimum at zero, one can guarantee linear convergence.

(*iv*) Our theory is backed up with extensive experiments, including affine rank minimization (Section 4.8.2), compressed noisy image reconstruction from a subset of image pixels (Section 4.8.3), and 1-bit matrix completion tasks (Section 4.8.4). Overall, our proposed scheme shows at least competitive recovery performance, as compared to state-of-the-art approaches, while being (*i*) simple to implement, (*ii*) scalable in practice and, (*iii*) versatile to various applications.

4.2 Applications

In this section, we describe some applications that can be modeled as in (4.2). The list includes criteria with (*i*) smooth and strongly convex objective f (*e.g.*, quantum state tomography from a limited set of observations and compressed image de-noising), and (*ii*) just smooth objective f (*e.g.*, 1-bit matrix completion and logistic PCA). For all cases, we succinctly describe the problem and provide useful references on state-of-the-art approaches; we restrict our discussion on first-order, gradient schemes. Some discussion regarding recent developments on factorized approaches is deferred to Section 4.3. Section 4.8

provides specific configuration of our algorithm, for representative tasks, and make a comparison with state of the art.

Matrix sensing applications. Matrix sensing (MS) problems have gained a lot of attention the past two decades, mostly as an extension of Compressed Sensing [48, 13] to matrices; see [55, 134]. The task involves the reconstruction of an *unknown and low-rank ground truth matrix $X^\star$, from a limited set of measurements*. The assumption on low-rankness depends on the application at hand and often is natural: *e.g.*, in background subtraction applications, $X^\star$ is a collection of video frames, stacked as columns where the "action" from frame to frame is assumed negligible [28, 165]; in (robust) principal component analysis [33, 28], we intentionally search for a low-rank representation of the data an hand; in linear system identification, the low rank $X^\star$ corresponds to a low-order linear, time-invariant system [115]; in sensor localization, $X^\star$ denotes the matrix of pairwise distances with rank-dependence on the, usually, low-dimensional space of the data [86]; in quantum state tomography, $X^\star$ denotes the density state matrix of the quantum system and $X^\star$ is designed to be rank-1 (pure state) or rank-r (almost pure state), for r relatively small [1, 57, 89].

In a non-factored form, MS is expressed via the following criterion:

$$\begin{aligned} \underset{X\in\mathbb{R}^{m\times n}}{\text{minimize}} \quad & f(X) := \tfrac{1}{2}\cdot\|y-\mathcal{A}(X)\|_2^2 \\ \text{subject to} \quad & \text{rank}(X)\leq r, \end{aligned} \tag{4.3}$$

where usually $m\neq n$ and $r\ll\min\{m,\ n\}$. Here, $y=\mathcal{A}\left(X^\star\right)+\varepsilon\in\mathbb{R}^p$ contains the (possibly noisy) samples, where $p\ll m\cdot n$. Key role in recovering $X^\star$ plays the sampling operator $\mathcal{A}$: it can be a Gaussian-based linear map [55, 134], a Pauli-based measurement operator [114] (used in quantum state tomography applications), a Fourier-based measurement operator [94, 134] (used due to

their structure which leads to computational gains in practice), or even a permuted and sub-sampled noiselet linear operator [165] (used in image and video compressive sensing applications).

Critical assumption for $\mathcal{A}$ that renders (4.3) a polynomially solvable problem, is the *restricted isometry property* (RIP) for low-rank matrices [30]:

Definition 4.1 (Restricted Isometry Property (RIP))**.** *A linear map $\mathcal{A}$ satisfies the r-RIP with constant δ_r, if*

$$(1-\delta_r)\|X\|_F^2 \leq \|\mathcal{A}(X)\|_2^2 \leq (1+\delta_r)\|X\|_F^2,$$

is satisfied for all matrices $X \in \mathbb{R}^{n\times n}$ such that rank$(X) \leq r$.

It turns out linear maps that satisfy Definition 4.1 also satisfy the (restricted) strong convexity [123]; see Theorem 2 in [37].

State-of-the-art approaches. The most popularized approach to solve this problem is through *convexification*: [54, 134, 31] show that the nuclear norm $\|\cdot\|_*$ is the tightest convex relaxation of the non-convex rank$(\cdot)$ constraint and algorithms involving nuclear norm have been shown to be effective in recovering low-rank matrices. This leads to:

$$\underset{X\in\mathbb{R}^{n\times p}}{\text{minimize}} \quad f(X) \quad \text{subject to} \quad \|X\|_* \leq t, \tag{4.4}$$

and its regularized variant:

$$\underset{X\in\mathbb{R}^{n\times p}}{\text{minimize}} \quad f(X) + \lambda \cdot \|X\|_*. \tag{4.5}$$

Efficient implementations include Augmented Lagrange Multiplier (`ALM`) methods [108], convex conic solvers like the `TFOCS` software package [15] and, convex proximal and projected first-order methods [26, 14]. However, due to the

nuclear norm, in most cases these methods are binded with full SVD computations per iteration, which constitutes them impractical in large-scale settings.

From a non-convex perspective, algorithms that solve (4.3) in a non-factored form include `SVP` and Randomized `SVP` algorithms [80, 16], Riemannian Trust Region Matrix Completion algorithm (`RTRMC`) [21], `ADMiRA` [104] and the `Matrix ALPS` framework [97, 147].[4]

In all cases, algorithms admit fast linear convergence rates towards $X^\star$. Moreover, the majority of approaches assumes a *first-order* oracle: information of f is provided through its gradient $\nabla f(X)$. For MS, $\nabla f(X) = -2\mathcal{A}^* \left(y - \mathcal{A}(X)\right)$, which requires $O(\mathrm{T}_{\mathrm{map}})$ complexity, where $\mathrm{T}_{\mathrm{map}}$ denotes the time required to apply linear map (or its adjoint $\mathcal{A}^*$) $\mathcal{A}$. Moreover, formulations (4.3)-(4.5) require at least one top-r SVD calculation per iteration; this translates into additional $O(mnr)$ complexity.

Motivation for factorizing (4.3). For this case, the initial problem can be factorized as follows:

$$\underset{U \in \mathbb{R}^{n\times r}, V \in \mathbb{R}^{p\times r}}{\text{minimize}} \quad f(UV^\top) := \tfrac{1}{2} \cdot \|y - \mathcal{A}(UV^\top)\|_2^2. \tag{4.6}$$

For this case and assuming a first-order oracle over the factors U, V, the gradient of f with respect to U and V can be computed respectively as $\nabla_U f(UV^\top) := \nabla f(X)V$ and $\nabla_V f(UV^\top) := \nabla f(X)^\top U$, respectively. This translates into $2 \cdot O(\mathrm{T}_{\mathrm{map}} + mnr)$ time complexity. However, one avoids performing any SVD calculations per iteration, which in practice is considered a great computational bottleneck, even for moderate r values. Thus, if there ex-

[4]Based on the results of [97], we use `Matrix ALPS II` in our experiments, as that scheme seems to be more effective and faster than the aforementioned algorithms.

ist linearly convergent algorithms for (4.6), intuition indicates that one could obtain computational gains.

Logistic PCA and low-rank estimation on binary data. Finding a low-rank approximation of binary matrices has gain a lot of interest recently, due to the wide appearance of categorical responses in real world applications [138, 40, 88, 154, 61, 113]. While regular linear principal component analysis (PCA) is still applicable for binary or categorical data, (i) the way data are pre-processed (*e.g.*, centering data before applying PCA), and/or (ii) the least-squares nature of the underlying objective criterion, constitute PCA a natural choice mostly for *real-valued* data, where observations are assumed to follow a Gaussian distribution. [148, 43] propose generalized versions of PCA for other type of datasets: In the case of binary data, this leads to Logistic Principal Component Analysis (Logistic PCA), where each binary data vector is assumed to follow the multivariate Bernoulli distribution, parametrized by the principal components that live in a r-dimensional subspace. Moreover, collaborative filtering on binary data and network sign prediction tasks have shown that standard least-squares loss functions perform poorly, while generic logistic loss optimization shows more interpretable and promising results.

To rigorously formulate the problem, let $Y \in \{0,1\}^{m\times n}$ be the observed binary matrix, where each of the m rows stores a n-dimensional binary feature vector. Further, assume that each entry Y_{ij} is drawn from a Bernoulli distribution with mean q_{ij}, according to: $\mathbb{P}\left[Y_{ij} \mid q_{ij}\right] = q_{ij}^{Y_{ij}} \cdot (1-q_{ij})^{1-Y_{ij}}$. Define the log-odds parameter $X_{ij} = \log\left(\frac{q_{ij}}{1-q_{ij}}\right)$ and the logistic function $\sigma(X_{ij}) = \left(1+e^{-X_{ij}}\right)^{-1}$. Then, we equivalently have $\mathbb{P}\left[Y_{ij} \mid X_{ij}\right] = \sigma(X_{ij})^{Y_{ij}} \cdot$

$\sigma(-X_{ij})^{1-Y_{ij}}$, or in matrix form,

$$\mathbb{P}\left[Y \mid X\right] = \prod_{ij} \sigma(X_{ij})^{Y_{ij}} \cdot \sigma(-X_{ij})^{1-Y_{ij}},$$

where we assume independence among entries of Y. The negative log-likelihood is given by:

$$f(X) := -\sum_{ij} \left(Y_{ij} \cdot \log \sigma(X_{ij}) + (1 - Y_{ij}) \cdot \log \sigma(-X_{ij})\right).$$

Assuming a compact, *i.e.*, low-rank, representation for the latent variable X, we end up with the following optimization problem:

$$\begin{array}{ll} \underset{X \in \mathbb{R}^{m \times n}}{\text{minimize}} & f(X) := -\sum_{ij} \left(Y_{ij} \cdot \log \sigma(X_{ij}) + (1 - Y_{ij}) \cdot \log \sigma(-X_{ij})\right) \\ \text{subject to} & \operatorname{rank}(X) \leq r; \end{array} \tag{4.7}$$

observe that the objective criterion is just a smooth convex loss function.

State-of-the-art approaches.[5] In [40], the authors consider the problem of *sign prediction* of edges in a signed network and cast it as a low-rank matrix completion problem: In order to model sign inconsistencies between the entries of binary matrices[6], the authors consider more appropriate loss functions to minimize, among which is the logistic loss. The proposed algorithmic solution follows (stochastic) gradient descent motions; however, no guarantees

[5]Here, we note that [99] proposes a slightly different way to generalize PCA than [43], based on a different interpretation of Pearson's PCA formulation [132]. The resulting formulation looks for a *weighted* projection matrix $UU^\top$ (instead of $UV^\top$), where the number of parameters does not increase with the number of samples and the application of principal components to new data requires only one matrix multiplication. For this case, the authors in [99] propose, among others, an alternating minimization technique where convergence to a local minimum is guaranteed. Even for this case though, our framework applies, based on ideas from [19].

[6]Here, we assume a matrix is binary if it has $\{\pm 1\}$ entries.

are provided. [88] utilizes logistic PCA for collaborative filtering on implicit feedback data (page clicks and views, purchases, etc.): in order to find a local minimum, an alternating gradient descent procedure is used—further, the authors use AdaGrad [51] to adaptively update the gradient descent step size, in order to reduce the number of iterations for convergence. A similar alternating gradient descent approach is followed in [138], with no known theoretical guarantees.

Motivation for factorizing (4.7). Following same arguments as before, in logistic PCA and logistic matrix factorization problems, we often assume that the observation binary matrix is generated as the `sign` operation on a linear factored model: $\texttt{sign}(UV^T)$. Assuming the probability of $\{\pm 1\}$ values is distributed according to a logistic function, parameterized by the latent factors U, V, we obtain the following optimization criterion:

$$\underset{U\in\mathbb{R}^{m\times r}, V\in\mathbb{R}^{n\times r}}{\text{minimize}} \quad f(UV^\top) := -\sum_{ij} \left(Y_{ij} \log \sigma(U_i V_j^\top) + (1 - Y_{ij}) \log \sigma(-U_i V_j^\top)\right), \tag{4.8}$$

where U_i, V_j represent the i-th and j-th row of U and V, respectively.

4.3 Related work

As it is apparent from the discussion above, this is not the first time such transformations have been considered in practice. Early works on principal component analysis [41, 137] and non-linear estimation procedures [172], use this technique as a heuristic; empirical evaluations show that such heuristics work well in practice [136, 62, 7]. [24, 25] further popularized these ideas for solving SDPs: their approach embeds the PSD and linear constraints into the objective and applies low-rank variable re-parameterization. While the

constraint considered here is of different nature—*i.e*, rank constraint vs. PSD constraint—the motivation is similar: in SDPs, by representing the solution as a product of two factor matrices, one can remove the positive semi-definite constraint and thus, avoid computationally expensive projections onto the PSD cone.

We provide an overview of algorithms that solve instances of (4.2); for discussions on methods that operate on X directly, we defer the reader to [2, 78, 97] for more details. We divide our discussion into two problem settings: (i) $X^\star$ is square and PSD and, (ii) $X^\star$ is non-square.

Square and PSD $X^\star$. A rank-r matrix $X \in \mathbb{R}^{n\times n}$ is PSD if and only if it can be factored as $X = UU^\top$ for some $U \in \mathbb{R}^{n\times r}$. This is a special case of the problem discussed above, where $m = n$ and (4.1) includes a PSD constraint. Thus, after the re-parameterization, (4.2) takes the form:

$$\underset{U\in\mathbb{R}^{n\times r}}{\text{minimize}} \quad f(UU^\top) \qquad \text{where } r = \text{rank}(X^\star) \leq n. \tag{4.9}$$

Several recent works have studied (4.9). For the special case where f is a least-squares objective for an underlying linear system[7], [152] and [184] propose gradient descent schemes that function on the factor U. Both studies employ careful initialization (performing few iterations of `SVP` [80] for the former and, using a spectral initialization procedure—as in [29]—for the latter) and step size selection, in order to prove convergence. However, their analysis is designed only for least-squares instances of f. Some results and discussion on their step size selection/initialization and how it compares with this work are provided in Section 4.8.

[7]This includes affine rank minimization problems, as well as matrix completion instances.

The work of [38] proposes a first-order algorithm for (4.9), where f is more generic. The algorithmic solution proposed can handle additional constraints on the factors U; the nature of these constraints depends on the problem at hand.[8] The authors present a broad set of exemplars for f—matrix completion and sensing, as well as sparse PCA, among others. For each problem, a set of assumptions need to be satisfied; *i.e.*, faithfulness, local descent, local Lipschitz and local smoothness conditions; see [38] for more details. Under such assumptions and with proper initialization, one can prove convergence with $O(\frac{1}{\varepsilon})$ or $O(\log(\frac{1}{\varepsilon}))$ rate, depending on the nature of f, and for problems that even fail to be locally convex.

[19] proposes *Factored Gradient Descent* (`FGD`) algorithm for (4.9). `FGD` is also a first-order scheme; key ingredient for convergence is a novel step size selection that can be used for any f, as long as it is gradient Lipschitz continuous; when f is further strongly convex, their analysis lead to faster convergence rates. Using proper initialization, this is the first paper that *provably* solves (4.9) for general convex functions f and under common convex assumptions. An extension of these ideas to some constrained problem cases can be found in [131].

To summarize, most of these results guarantee convergence –up to linear rate– on the factored space, starting from a "good" initialization point and employing a carefully selected step size.

Non-square $X^\star$. [81] propose `AltMinSense`, an alternating minimization

[8] Any additional constraints should satisfy the *faithfulness* property: a constraint set $\mathcal{C}$ is faithful if for each $U \in \mathcal{C}$, within some bounded radius from optimal point, we are guaranteed that the closest (in the Euclidean sense) rotation of optimal $U^\star$ lies within $\mathcal{U}$.

algorithm for matrix sensing and matrix completion problems. This is one of the first works to prove linear convergence in solving (4.2) for the MS model. Moreover, [65] improves upon [81] for the case of reasonably well-conditioned matrices. Their algorithm handles problem cases with bad condition number and gaps in their spectrum. Recently, [152] extended the `Procrustes Flow` algorithm to the non-square case, where gradient descent, instead of exact alternating minimization, is utilized. A few days before this paper, [185] also extended the first-order method of [38] for matrix completion to the rectangular case. All the studies above focus on the case of least-squares objective f.

[146] generalize the results in [81, 65]: the authors show that, under common incoherence conditions and sampling assumptions, most first-order variants (*e.g.*, gradient descent, alternating minimization schemes and stochastic gradient descent, among others) indeed converge to the low-rank ground truth $X^\star$. Specifically, for the alternating gradient descent variant, the authors propose several step size selection procedures[9]. Both the theory and the algorithm proposed are restricted to the matrix completion objective.

Recently, [183]—based on the inexact first-order oracle, previously used in [10]— proved that linear convergence is guaranteed if $f(UV^\top)$ is strongly convex over either U and V, when the other is fixed. While the technique applies for generic f and for non-square X, the authors provide algorithmic solutions only for matrix completion / matrix sensing settings.[10] Furthermore,

[9] However, the restricted Armijo rule, as well as the line search procedure, can be applied to any the aforementioned algorithms too.

[10] *E.g.*, in the gradient descent case, the step size proposed depends on RIP [134] constants and it is not clear what a good step size would be in other problem settings.

their algorithm requires QR-decompositions after each update of U and V; this is required in order to control the notion of inexact first order oracle.

4.4 Preliminaries

Notation. For matrices $X, Y \in \mathbb{R}^{m\times n}$, $\langle X, Y\rangle = \operatorname{tr}\left(X^\top Y\right)$ represents their inner product. We use $\|X\|_F$ and $\sigma_1(X)$ for the Frobenius and spectral norms of a matrix, respectively; we also use $\|X\|_2$ to denote spectral norm. Moreover, we denote as $\sigma_i(X)$ the i-th singular value of X. For a rank-r matrix $X = UV^\top$, the gradient of f w.r.t. U and V is $\nabla f(UV^\top)V$ and $\nabla f(UV^\top)^\top U$, respectively. With a slight abuse of notation, we will also use the terms $\nabla_U f(UV^\top) \triangleq \nabla f(UV^\top)V$ and $\nabla_V f(UV^\top) := \nabla f(UV^\top)^\top U$.

Given a matrix X, we denote its best rank-r approximation with X_r; X_r can be computed in polynomial time via the SVD. For our discussions from now on and in an attempt to simplify our notation, we denote the optimum point we search for as $X_r^\star$, both (i) in the case where we intentionally restrict our search to obtain a rank-r approximation of $X^\star$ –while $\operatorname{rank}(X^\star) > r$– and (i) in the case where $X^\star \equiv X_r^\star$, *i.e.*, by default, the optimum point is of rank r.

An important issue in optimizing f over the factored space is the existence of non-unique possible factorizations for a given X. Since we are interested in obtaining a low-rank solution in the original space, we need a notion of distance to the low-rank solution $X_r^\star$ while we are optimizing over the factors. Among infinitely many possible decompositions of $X_r^\star$, we focus on the set of

balanced factorizations [152]:

$$\mathcal{X}_r^\star = \Big\{ (U^\star, V^\star) : \; U^\star \in \mathbb{R}^{m\times r}, V^\star \in \mathbb{R}^{n\times r}, U^\star V^{\star\top} = X_r^\star, \\ \sigma_i(U^\star) = \sigma_i(V^\star) = \sigma_i(X_r^\star)^{1/2}, \forall i \in [r] \Big\}. \tag{4.10}$$

Note that $(U^\star, V^\star) \in \mathcal{X}_r^\star$ if and only if the pair can be written as $U^\star = A^\star \Sigma^{\star 1/2} R$, $V^\star = B^\star \Sigma^{\star 1/2} R$, where $A^\star \Sigma^\star B^\star$ is the singular value decomposition of $X_r^\star$, and $R \in \mathbb{R}^{r\times r}$ is an orthogonal matrix.

Given a pair (U, V), we define the distance to $X_r^\star$ as:

$$\operatorname{dist}(U, V; X_r^\star) = \min_{(U^\star, V^\star)\in\mathcal{X}_r^\star} \left\| \begin{bmatrix} U \\ V \end{bmatrix} - \begin{bmatrix} U^\star \\ V^\star \end{bmatrix} \right\|_F.$$

Assumptions. We consider applications that can be described (i) either by restricted *strongly* convex functions f with *gradient Lipschitz continuity*, or (ii) by convex functions f that have only Lipschitz continuous gradients.[11] We state these standard definitions below.

Definition 4.2. *Let $f : \mathbb{R}^{m\times n} \to \mathbb{R}$ be a convex differentiable function. Then, f is gradient Lipschitz continuous with parameter L (or L-smooth) if:*

$$\|\nabla f(X) - \nabla f(Y)\|_F \le L \cdot \|X - Y\|_F, \tag{4.11}$$

$\forall X, Y \in \mathbb{R}^{m\times n}$.

Definition 4.3. *Let $f : \mathbb{R}^{m\times n} \to \mathbb{R}$ be convex and differentiable. Then, f is μ-strongly convex if:*

$$f(Y) \ge f(X) + \langle \nabla f(X), Y - X \rangle + \tfrac{\mu}{2} \|Y - X\|_F^2, \tag{4.12}$$

$\forall X, Y \in \mathcal{X} \subseteq \mathbb{R}^{m\times n}$.

[11] Our ideas can be extended in a similar fashion to the case of restricted strong convexity [123, 2].

4.5 The Bi-Factored Gradient Descent (BFGD) algorithm

In this section, we provide an overview of the *Bi-Factored Gradient Descent* (BFGD) algorithm for two problem settings in (4.1): (i) f being a L-smooth convex function and, (ii) f being L-smooth and μ-strongly convex. For both cases, we assume a good initialization point $X_0 = U_0 V_0^\top$ is provided; given X_0 and under proper assumptions, we further describe the theoretical guarantees that accompany BFGD.

As introduced, BFGD is built upon non-convex gradient descent over each factor U and V, written as

$$U_{t+1} = U_t - \eta \cdot \nabla_U f(U_t V_t^\top), \quad V_{t+1} = V_t - \eta \cdot \nabla_V f(U_t V_t^\top). \tag{4.13}$$

When f is convex and smooth, BFGD follows exactly the motions in (4.13); in the case where f is also strongly convex, BFGD is based on a different set of recursions, which we discuss in more detail in the rest of the section.

4.5.1 Reduction to FGD: When f is convex and L-smooth

In [78], the authors describe a simple technique to transform problems similar to (4.1), where the variable space is that of low-rank, non-square X, into problems where we look for a square and PSD solution. The key idea is to *lift* the problem and introduce a stacked matrix of the two factors, as follows:

$$W = \begin{bmatrix} U \\ V \end{bmatrix} \in \mathbb{R}^{(m+n)\times r},$$

and optimize over a new function $\hat{f} : \mathbb{R}^{(m+n)\times(m+n)} \to \mathbb{R}$ defined as

$$\hat{f}\left(WW^\top\right) = \hat{f}\left(\begin{bmatrix} UU^\top & UV^\top \\ VU^\top & VV^\top \end{bmatrix}\right) = f(UV^\top).$$

Following this idea, one can utilize algorithmic solutions designed only to work on square and PSD-based instances of (4.1), where f is just L-smooth. Here, we use the *Factored Gradient Descent* (`FGD`) algorithm of [19] on the W-space, as follows:

$$W_{t+1} = W_t - \eta \cdot \nabla_W \hat{f}(W_t W_t^\top). \tag{4.14}$$

Then, it is easy to verify the following remark:

Remark 4.4. *Define*

$$\hat{f}\left(\begin{bmatrix} A & B \\ C & D \end{bmatrix}\right) = \frac{1}{2} f(B) + \frac{1}{2} f(C^\top)$$

for $A \in \mathbb{R}^{m \times m}$, $B \in \mathbb{R}^{m \times n}$, $C \in \mathbb{R}^{n \times m}$, $D \in \mathbb{R}^{n \times n}$. Then `FGD` for minimizing $\hat{f}(WW^\top)$ with the stacked matrix $W = [U^\top, V^\top]^\top \in \mathbb{R}^{(m+n) \times r}$ is equivalent to (4.13).

A natural question is whether this reduction gives a desirable convergence behavior. Since `FGD` solves for a different function $\hat{f}$ from the original f, the convergence analysis depends also on $\hat{f}$. When f is convex and smooth, we can rely on the result from [19].

Proposition 4.5. *If f is convex and L-smooth, then $\hat{f}$ is convex and $\frac{L}{2}$-smooth.*

Proof. For any $Z_1 = \begin{bmatrix} A_1 & B_1 \\ C_1 & D_1 \end{bmatrix}, Z_2 = \begin{bmatrix} A_2 & B_2 \\ C_2 & D_2 \end{bmatrix} \in \mathbb{R}^{(m+n)\times(m+n)}$, we have

$$\begin{aligned}
&\left\|\nabla \hat{f}(Z_1) - \nabla \hat{f}(Z_2)\right\|_F \\
&= \left\| \begin{bmatrix} 0 & \frac{1}{2}\cdot\nabla f(B_1) \\ \frac{1}{2}\cdot\nabla f(C_1^\top)^\top & 0 \end{bmatrix} - \begin{bmatrix} 0 & \frac{1}{2}\cdot\nabla f(B_2) \\ \frac{1}{2}\cdot\nabla f(C_2^\top)^\top & 0 \end{bmatrix} \right\|_F \\
&= \tfrac{1}{2}\cdot\sqrt{\|\nabla f(B_1) - \nabla f(B_2)\|_F^2 + \left\|\nabla f(C_1^\top) - \nabla f(C_2^\top)\right\|_F^2} \\
&\leq \tfrac{L}{2}\cdot\sqrt{\|B_1 - B_2\|_F^2 + \|C_1 - C_2\|_F^2} \\
&\leq \tfrac{L}{2}\cdot\|Z_1 - Z_2\|_F
\end{aligned}$$

where the first inequality follows from the L-smoothness of f. □

Based on the above proposition, we use `FGD` to solve (4.2) with $\hat{f}$: its procedure is exactly (4.13) (up to a factor of 2 for the step size). While one can rely on the sublinear convergence analysis from [19], we provide a new guarantee with a weaker initial condition. Our step size condition is the following:

$$\eta \leq \frac{1}{20L\left\|\begin{bmatrix} U_0 \\ V_0 \end{bmatrix}\right\|_2^2 + 3\|\nabla f(U_0 V_0^\top)\|_2} \tag{4.15}$$

In Section 4.6, we discuss a convergence guarantee under this step size condition.

4.5.2 Using `BFGD` when f is L-smooth and strongly convex

Now we assume f function satisfies both properties in Definitions 4.2 and 4.3. In this case, we cannot simply rely on the lifting technique as above since $\hat{f}$ is clearly not strongly convex. Instead, we consider a slight variation, based on [152], where we appropriately regularize the objective and force the

solution pair $(\widehat{U}, \widehat{V})$ to be "balanced", according to the definition in (4.10). In particular, we consider the following optimization problem:

$$\underset{U\in\mathbb{R}^{m\times r},\ V\in\mathbb{R}^{n\times r}}{\text{minimize}} \quad f(UV^\top) + \lambda \cdot g(U^\top U - V^\top V), \tag{4.16}$$

where $g : \mathbb{R}^{r\times r} \to \mathbb{R}$ is an additional convex *regularizer*. We require the selected g to be such that:

- (i) g is convex and minimized at zero point; *i.e.*, $\nabla g(0) = 0$.
- (ii) The gradient, $\nabla g(U^\top U - V^\top V) \in \mathbb{R}^{r\times r}$, is symmetric for any such pair.
- (iii) g is μ_g-strongly convex and L_g-smooth.

The necessity of the regularizer. As we show next, the theoretical guarantees of `BFGD` heavily depend on the condition number of the pair $(U^\star, V^\star)$ the algorithm converges to. In particular, one of the requirements of `BFGD` is that every estimate U_t (resp. V_t) be "relatively close" to the convergent point $U^\star$ (resp. $V^\star$), such that their distance $\|U_t - U^\star\|_F$ is bounded by a function of $\sigma_r(U^\star)$, for all t. Then, it is easy to observe that, for arbitrarily ill-conditioned $(U^\star, V^\star) \in \mathcal{X}_r^*$, such a condition might not be easily satisfied by `BFGD` per iteration[12], unless we "force" the sequence of estimates (U_t, V_t), $\forall t$, to converge to a better conditioned pair $(U^\star, V^\star)$. This is the key role of regularizer g: it guarantees U and V are not too ill-conditioned. Note that adding g does not change the optimum of f in the original X space.[13]

[12]Even if $UV^\top$ is close to $U^\star V^{\star\top}$, the condition numbers of U and V can be much larger than the condition number of $UV^\top$.

[13]In particular, for any rank-r solution $UV^\top$, there is a factorization $(\tilde{U}, \tilde{V})$ minimizing g with the same function value $f(\tilde{U}\tilde{V}^\top) = f(UV^\top)$, which are

$$\tilde{U} = A\Sigma^{\frac{1}{2}}, \quad \tilde{V} = B\Sigma^{\frac{1}{2}}$$

An example of g is the Frobenius norm (weighted by $\mu/2$), as proposed in [152]. Other examples are sums of element-wise (at least) μ_g-strongly convex and (at most) L_g-gradient Lipschitz functions (of the form $g(X) = \sum_{i,j} g_{ij}(X_{ij})$) with the optimum at zero. However, any other user-friendly g can be selected, as long as it satisfies the above conditions. We show in this chapter that any such regularizer results provably in convergence, with attractive convergence rate. However, as we observe in practice, one can remove g from the objective and observe slightly different performance in practice.

The `BFGD` algorithm. `BFGD` is a first-order, gradient descent algorithm, that operates on the factored space (U, V) in an alternating fashion. Principal components of `BFGD` is a proper step size selection and a "decent" initialization point. `BFGD` can be considered as the non-squared extension of `FGD` algorithm in [19], which is specifically designed to solve problems as in (4.2), for $U = V$ and $m = n$. The key differences with `FGD` though, other than the necessity of a regularizer g, are two-fold:

(i) The main recursion followed is different in the two schemes: in the non-squared case, we update the left and right factors $(U,\ V)$ with a different rule, according to which:

$$U_{t+1} = U_t - \eta \left(\nabla_U f(U_t V_t^\top) + \lambda \cdot \nabla_U g(U_t^\top U_t - V_t^\top V_t) \right),$$
$$V_{t+1} = V_t - \eta \left(\nabla_V f(U_t V_t^\top) + \lambda \cdot \nabla_V g(U_t^\top U_t - V_t^\top V_t) \right).$$

The parameter $\lambda > 0$ is arbitrarily chosen.

---

where $UV^\top = A\Sigma B^\top$ is the singular value decomposition.

Algorithm 6 `BFGD` for smooth and strongly convex f

Input: Function f, target rank r, # iterations T.

1: Set initial values for U_0, V_0
2: Set step size η as in (4.17).
3: **for** $t = 0$ to $T - 1$ **do**
4: $\quad U_{t+1} = U_t - \eta \left(\nabla_U f(U_t V_t^\top) - \lambda \cdot \nabla_U g(U_t^\top U_t - V_t^\top V_t) \right)$
5: $\quad V_{t+1} = V_t - \eta \left(\nabla_V f(U_t V_t^\top) - \lambda \cdot \nabla_V g(U_t^\top U_t - V_t^\top V_t) \right)$
6: **end for**

Output: $X = U_T V_T^\top$.

(*ii*) Due to this new update rule, a slightly different and proper step size selection should be devised for the case of `BFGD`. Our step size is selected as follows:

$$\eta \leq \frac{1}{12 \cdot \max\{L,\ L_g\} \cdot \left\| \begin{bmatrix} U_0 \\ V_0 \end{bmatrix} \right\|_2^2}. \tag{4.17}$$

The scheme is described in Algorithm 6. Observe that η has similar formula with the step size in [19]. Though, as we show next, our analysis simplifies further the selection of the step size.[14]. As shown in the next section, constant step size (4.17) is sufficient to lead to attractive convergence rates for `BFGD`, for f L-smooth and μ-strongly convex.

4.6 Local convergence for `BFGD`

This section includes the main theoretical guarantees of `BFGD`, both for the cases of just smooth f, and f being smooth and strongly convex. Our results follow and improve upon the results for the square and PSD case from

[14] There is no dependence on the spectral norm of the gradient.

[19]. To provide such local convergence results, we assume that there is a known "good" initialization which ensures the following.

Assumption A1. *Define* $\kappa = \frac{\max\{L, L_g\}}{\min\{\mu, \mu_g\}}$ *where* μ_g *and* L_g *are the strong convexity and smoothness parameters of* g*, respectively. Then, we assume we are provided with a "good" initialization point* $X_0 = U_0 V_0^\top$ *such that:*

$$\operatorname{dist}(U_0, V_0; X_r^\star) \leq \frac{\sqrt{2} \cdot \sigma_r(X_r^\star)^{1/2}}{10\sqrt{\kappa}} \qquad \text{(Strongly convex and smooth } f\text{)},$$

$$\operatorname{dist}(U_0, V_0; X_r^\star) \leq \frac{\sqrt{2} \cdot \sigma_r(X_r^\star)^{1/2}}{10} \qquad \text{(Smooth } f\text{)}.$$

Moreover, for our analysis, we will use the following step size assumptions:

$$\widehat{\eta} \leq \frac{1}{8 \max\{L, L_g\} \cdot \left\| \begin{bmatrix} U_t \\ V_t \end{bmatrix} \right\|_2^2} \qquad \text{(Strongly convex and smooth } f\text{)}, \tag{4.18}$$

$$\widehat{\eta} \leq \frac{1}{15L \left\| \begin{bmatrix} U_t \\ V_t \end{bmatrix} \right\|_2^2 + 3\|\nabla f(U_t V_t^\top)\|_2} \qquad \text{(Smooth } f\text{)} \tag{4.19}$$

The assumption and the step size depends on the strong convexity and smoothness parameters of g. When μ and L are known a priori, this dependency can be removed since one can choose g such that at least μ-restricted strongly convex and at most L-smooth. Then, κ becomes the condition number of f, and the step size depends only on L.

Observe that step sizes in (4.18) and (4.19) are computationally inefficient in practice: they require at most two spectral norm computations of U_t, V_t and $\nabla f(U_t V_t^\top)$ per iteration. However, as the following lemma states, even in the case where we cannot afford such calculations per iteration, there is a constant-fraction connection between $\widehat{\eta}$ and η.

Lemma 4.6. *Let* (U_0, V_0) *be such that Assumption A1 is satisfied. Then,* (4.18) *holds if* (4.17) *is satisfied, and* (4.19) *holds if* (4.15) *is satisfied.*

The proof is provided in the Appendix C.1. By this lemma, our analysis below is equivalent –up to constants– to that if we were using the original step size η of the algorithm. However, for clarity reasons and ease of exposition, we use $\widehat{\eta}$ below.

4.6.1 Linear local convergence rate when f is L-smooth and μ-strongly convex

The following theorem proves that, under proper initialization, `BFGD` admits linear convergence rate, when f is both L-smooth and μ-restricted strongly convex.

Theorem 4.7. *Suppose that* f *is* L*-smooth and* μ*-restricted strongly convex and regularizer* g *is* L_g*-smooth and* μ_g*-restricted strongly convex. Define* $\mu_{\min} := \min\{\mu, \mu_g\}$ *and* $L_{\max} := \max\{L, L_g\}$*. Denote the unique minimizer of* f *as* $X^\star \in \mathbb{R}^{m\times n}$ *and assume that* $X^\star$ *is of arbitrary rank. Let* $\widehat{\eta}$ *be defined as in* (4.18)*. If the initial point* (U_0, V_0) *satisfies Assumption A1, then* ***BFGD*** *algorithm in Algorithm 6 converges linearly to* $X_r^\star$*, within error* $O\left(\sqrt{\frac{\kappa}{\sigma_r(X_r^\star)}}\left\|X^\star - X_r^\star\right\|_F\right)$*, according to the following recursion:*

$$\operatorname{dist}(U_{t+1}, V_{t+1}; X_r^\star)^2 \leq \gamma_t \cdot \operatorname{dist}(U_t, V_t; X_r^\star)^2 + \widehat{\eta} L \left\|X^\star - X_r^\star\right\|_F^2, \tag{4.20}$$

for every $t \geq 0$*, where the contraction parameter* γ_t *satisfies:*

$$\gamma_t = 1 - \widehat{\eta} \cdot \tfrac{\mu_{\min}\cdot\sigma_r(X_r^\star)}{8} \geq 1 - \tfrac{\mu_{\min}}{17\cdot L_{\max}} \cdot \tfrac{\sigma_r(X_r^\star)}{\sigma_1(X_r^\star)} > 0.$$

The proof is provided in Section C.2. The theorem states that if $X^\star$ is (approximately) low-rank, the iterates converge to a close neighborhood of $X_r^\star$.

The above result can also be expressed w.r.t. the function value $f(UV^\top)$, as follows:

Corollary 4.8. *Under the same initial condition with Theorem 4.7, Algorithm* (6) *satisfies the following recursion w.r.t. to the distance of function values from the optimal f values:*

$$f(U_t V_t^\top) - f(X^\star) \leq \gamma^t \cdot \sigma_1(X^\star) \cdot \big(f(U_0 V_0^\top) - f(X^\star)\big) + \frac{\sqrt{\mu L}}{\sigma_r(X^\star)} \|X^\star - X_r^\star\|_F^2$$

4.6.2 Local sublinear convergence

In Section 4.5.1, we showed that a lifting technique can reduce our problem (4.2) to a rank-constrained semidefinite program, and applying `FGD` from [19] is exactly `BFGD` (4.13). While the sublinear convergence guarantee of `FGD` can also be applied to our problem, we provide an improved result.

Theorem 4.9. *Suppose that f is L-smooth with a minimizer $X^\star \in \mathbb{R}^{m\times n}$. Let $\hat{X}_r$ be any target rank-r matrix, and let $\widehat{\eta}$ be defined as in* (4.19)*. If the initial point $X_0 = U_0 V_0^\top$, $U_0 \in \mathbb{R}^{m\times r}$ and $V_0 \in \mathbb{R}^{n\times r}$, satisfies Assumption A1, then `FGD` converges with rate $O(1/T)$ to a tolerance value according to:*

$$f(U_T V_T^\top) - f(U^\star V^{\star\top}) = \hat{f}(W_T W_T^\top) - \hat{f}(W^\star W^{\star\top}) \leq \frac{10 \cdot \mathrm{dist}(U_0, V_0; X_r^\star)^2}{\eta T}$$

Theorem 4.9 guarantees a local sublinear convergence with a looser initial condition. While [19] requires $\min_{R\in O(r)} \|W - W^\star R\|_F \leq \frac{\sigma_r^2(W^\star)}{100\sigma_1^2(W^\star)} \cdot \sigma_r(W^\star)$, our result requires that the initial distance to the $W^\star$ is merely a constant factor of $\sigma_r(W^\star)$.

4.7 Initialization

Our main theorem guarantees linear convergence in the factored space given that the initial solution (U_0, V_0) is within a ball around the closest target factors $(U_0^\star, V_0^\star)$ with radius $O(\kappa^{-1/2}\sigma_r(X_r^\star)^{1/2})$. To find such a solution, we propose an extension of the initialization in [19].

Lemma 4.10. *Consider an initial solution $U_0V_0^\top$ which is the best rank-r approximation of*

$$X_0 = -\tfrac{1}{L}\nabla f(0) \tag{4.21}$$

Then we have

$$\left\|U_0V_0^\top - X_r^\star\right\|_F \le 2\sqrt{2\left(1-\tfrac{1}{\kappa}\right)}\left\|X^\star\right\|_F + 2\left\|X^\star - X_r^\star\right\|_F$$

Combined with Lemma 5.14 in [152], which transforms a good initial solution from the original space to the factored space, we can obtain an appropriate initial solution. The following corollary gives one sufficient condition for global convergence of `BFGD` with the SVD of (4.21) as initialization.

Corollary 4.11. *If*

$$\left\|X^\star - X_r^\star\right\|_F \le \frac{\sigma_r(X^\star)}{100\sqrt{\kappa}}, \quad \kappa \le 1 + \frac{\sigma_r(X_r^\star)^2}{4608\cdot\left\|X_r^\star\right\|_F^2}$$

then the initial solution

$$U_0 = A_0\Sigma_0^{1/2},\ V_0 = B_0\Sigma_0^{1/2}$$

where $A_0\Sigma_0B_0$ is the SVD of $-\frac{1}{L}\nabla f(0)$ satisfies the initial condition of Theorem 4.7.

While our theoretical results can only guarantee global convergence for a well-conditioned problem (κ close to one), we show in the experiments that the algorithm performs well in practice where the sufficient conditions are yet to be satisfied.

4.8 Experiments

In this section, we first provide comparison results regarding the actual computational complexity of SVD and matrix-matrix multiplication procedures; while such comparison is not thoroughly complete, it provides some evidence about the gains of optimizing over U, V factors, in lieu of SVD-based rank-r approximations. Next, we provide extensive results on the performance of `BFGD`, as compared with state of the art, for the following problem settings: (i) affine rank minimization, where the objective is smooth and (restricted) strongly convex, (ii) image denoising/recovery from a limited set of observed pixels, where the problem can be cast as a matrix completion problem, and (iii) 1-bit matrix completion, where the objective is just smooth convex. In all cases, the task is to recover a low rank matrix from a set of observations, where our machinery naturally applies.

4.8.1 Complexity of SVD and matrix-matrix multiplication procedures in practice

To provide an idea of how matrix-matrix multiplication scales, in comparison with truncated SVD,[15] we compare it with some state-of-the-art SVD

[15]Here, we consider algorithmic solutions where both SVD and matrix-matrix multiplication computations are performed with high-accuracy. One might consider *approximate* SVD—see the excellent monograph [64]—and matrix-matrix multiplication

subroutines: (i) the Matlab's `svds` subroutine, based on `ARPACK` software package [106], (ii) a collection of implicitly-restarted Lanczos methods for fast truncated SVD and symmetric eigenvalue decompositions (`irlba`, `irlbablk`, `irblsvds`) [9] [16], (iii) the limited memory block Krylov subspace optimization for computing dominant SVDs (`LMSVD`) [112], and (iv) the `PROPACK` software package [100]. We consider random realizations of matrices in $\mathbb{R}^{m\times n}$ (w.l.o.g., assume $m = n$), for varying values of m. For SVD computations, we look for the best rank-r approximation, for varying values of r. In the case of matrix-matrix multiplication, we record the time required for the computation of 2 matrix-matrix multiplications of matrices $\mathbb{R}^{m\times m}$ and $\mathbb{R}^{m\times r}$, which is equivalent with the computational complexity required in our scheme, in order to avoid SVD calculations. All experiments are performed in a Matlab environment.

Figure 4.1 (left panel) shows execution time results for the algorithms under comparison, as a function of the dimension m. Rank r is fixed to $r = 100$. While both SVD and matrix multiplication procedures are known to have $O(m^2 r)$ complexity, it is obvious that the latter on dense matrices is at least two-orders of magnitude faster than the former. In Table 4.1, we also report the approximation guarantees of some faster SVD subroutines, as compared to `svds`: while `irblablk` seems to be faster, it returns a very rough approximation of the singular values, when r is relatively large. Similar findings are depicted in Figures 4.1 (middle and right panel).

approximations—see [49, 50, 98, 42]; we believe that studying such alternatives is an interesting direction to follow for future work.

[16] IRLBA stands for Implicitly Restarted Lanczos Bidiagonalization Algorithms.

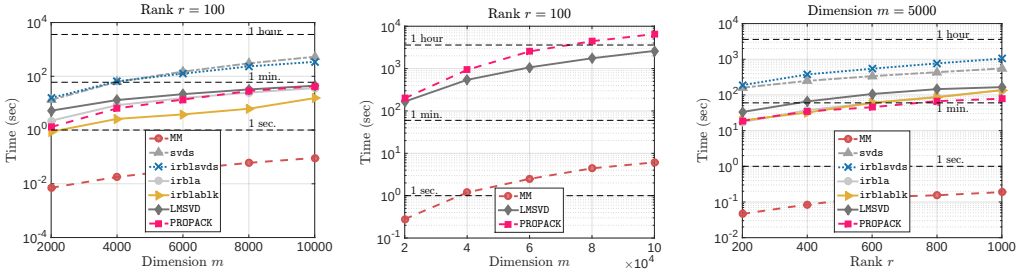


Figure 4.1: Comparison of SVD procedures versus Matrix Matrix (MM) multiplication. *Left panel:* Varying dimension m and constant rank $r = 100$. *Middle panel:* Similar to left panel where m scales larger and we focus on a subset of SVD algorithms that can scale up. *Right panel:* Varying rank values and constant dimension $m = 5 \cdot 10^3$.

Algorithms	Error $\frac{\|\widehat{\Sigma}-\Sigma^\star\|_F}{\|\Sigma^\star\|_F}$, where $\Sigma^\star$ is diagonal matrix with top r singular values from `svds`.				
$\widehat{\Sigma}$	$m = 2 \cdot 10^3$	$m = 4 \cdot 10^3$	$m = 6 \cdot 10^3$	$m = 8 \cdot 10^3$	$m = 10^4$
`irblsvds`	3.63e-15	4.33e-09	8.11e-11	4.79e-12	5.82e-10
`irbla`	6.00e-15	9.01e-07	1.05e-04	2.99e-04	7.29e-04
`irblablk`	1.48e+03	1.67e+03	1.24e+03	1.45e+03	7.91e+11
`LMSVD`	2.14e-14	4.49e-12	3.94e-11	1.33e-10	7.30e-10
`PROPACK`	4.10e-12	2.46e-10	1.63e-12	7.90e-12	3.55e-11

Table 4.1: Approximation errors of singular values, in the form $\frac{\|\widehat{\Sigma}-\Sigma^\star\|_F}{\|\Sigma^\star\|_F}$. Here, $\widehat{\Sigma}$ denote the diagonal matrix, returned by SVD subroutines, containing r top singular values; we use `svds` to compute the reference matrix $\Sigma^\star$, that contains top-r singular values of the input matrix. Observe that some algorithms deviate singificantly from the "ground-truth": this is due to either early stopping (only a subset of singular values could be computed) or due to accumulating approximation error.

4.8.2 Affine rank minimization using noiselet linear maps

In this task, we consider the problem of *affine rank minimization*. In particular, we observe unknown $X^\star$ through a limited set of observations $y \in$

$\mathbb{R}^p$, that satisfy:

$$y = \mathcal{A}\left(X^\star\right), \tag{4.22}$$

where $X^\star \in \mathbb{R}^{m \times n},\ p \ll m \cdot n$, and $\mathcal{A} : \mathbb{R}^{m \times n} \to \mathbb{R}^p$ is a given linear map. The task is to recover $X^\star$, using $\mathcal{A}$ and y. Here, we use permuted and sub-sampled noiselets for the linear operator $\mathcal{A}$, due to their efficient implementation [165]; similar results can be obtained for $\mathcal{A}$ being a subsampled Fourier linear operator or, even, a random Gaussian linear operator. For the purposes of this experiment, the ground truth $X^\star$ is synthetically generated as the multiplication of two tall matrices, $U^\star \in \mathbb{R}^{m \times r}$ and $V^\star \in \mathbb{R}^{n \times r}$, such that $X^\star = U^\star V^{\star\top}$ and $\|X^\star\|_F = 1$. Both $U^\star$ and $V^\star$ contain random, independent and identically distributed (i.i.d.) Gaussian entries, with zero mean and unit variance.

List of algorithms. We compare the following state-of-the-art algorithms: (*i*) the Singular Value Projection (`SVP`) algorithm [80], a non-convex, projected gradient descent algorithm for (4.3), with *constant* step size selection (we study the case where $\mu = 1/3$, as it is the one that showed the best performance in our experiments), (*ii*) the Matrix ALPS II variant in [97], an accelerated, first-order, non-convex algorithm, with adaptive step size and optimized subprocedures for the criterion in (4.3), (*iii*) the SparseApproxSDP extension to non-square cases for (4.5) in [78], based on [67], where a putative solution is refined via rank-1 updates from the gradient[17], (*iv*) the matrix completion algorithm in [146], which we call `GuaranteedMC`[18], where the objective is (4.6),

[17] SparseApproxSDP in [67] avoids computationally expensive operations per iteration, such as full SVDs. In theory, at the r-th iteration, these schemes guarantee to compute a $\frac{1}{r}$-approximate solutio, with rank at most r, *i.e.*, achieves a sublinear rate.

[18] We note that the original algorithm in [146] is designed for the matrix *completion* problem, not the matrix *sensing* problem here.

(v) the Procrustes Flow algorithm in [152] for (4.6), and (vi) the `BFGD` algorithm.[19]

Implementation details. To properly compare the algorithms in the above list, we preset a set of parameters that are common. In all experiments, we fix the number of observations in y to $p = C \cdot n \cdot r$, where $n \geq m$ in our cases, and for varying values of C. All algorithms in comparison are implemented in a MATLAB environment, where no `mex`-ified parts present, apart from those used in SVD calculations; see below.

In all algorithms, we fix the maximum number of iterations to $T = 4000$, unless otherwise stated. We use the same stopping criteria for the majority of algorithms as:

$$\frac{\|X_t - X_{t-1}\|_F}{\|X_t\|_F} \leq \text{tol}, \tag{4.23}$$

where $X_t,\ X_{t-1}$ denote the current and the previous estimates in the X space and $\text{tol} := 5 \cdot 10^{-6}$. For SVD calculations, we use the `lansvd` implementation in PROPACK package [100]. For fairness, we modified all the algorithms so that they *exploit the true rank* r; however, we observed that small deviations from the true rank result in relatively small degradation in terms of the reconstruction performance.[20]

In the implementation of `BFGD`, we set g to be $\frac{1}{16} \cdot \|U^\top U - V^\top V\|_F^2$, as suggested in [152], for ease of comparison. Moreover, for our implementation

[19] The algorithm in [183] assumes step size that depends on RIP constants, which are NP-hard to compute; since no heuristic is proposed, we do not include this algorithm in the comparison list.

[20] In case the rank of $X^\star$ is unknown, one has to predict the dimension of the principal singular space. The authors in [80], based on ideas in [92], propose to compute singular values incrementally until a significant gap between singular values is found.

of Procrustes Flow, we set the constant step size as $\mu := \frac{2}{187} \cdot \{\frac{1}{\|U_0\|_F^2}, \frac{1}{\|V_0\|_F^2\|}\}$, as suggested in [152]. We use the implementation of [146], with random initialization (unless otherwise stated) and regularization type `soft`, as suggested by their implementation. In [78], we require an upper bound on the nuclear norm of $X^\star$; in our experiments we assume we know $\|X^\star\|_*$, which requires a full SVD caclulation. Moreover, for our experiments, we set the curvature constant $C_f = 1$.

For initialization, we consider the following settings: (i) random initialization, where $X_0 = U_0 V_0^\top$ for some randomly selected U_0 and V_0 such that $\|X_0\|_F = 1$, and (ii) specific initialization, as suggested in each of the papers above. Our specific initialization is based on the discussion in Section 4.7, where $X_0 = \mathrm{Proj}\, r(-\frac{1}{L}\nabla f(0))$. Algorithms SVP, Matrix ALPS II, SparseApproxSDP and the solver in [146] work with random initialization. For the initialization phase of [152], we consider two cases: (i) the condition number κ is known, where according to Theorem 3.3 in [152], we require $T_{\text{init}} := \lceil 3\log(\sqrt{r}\cdot\kappa) + 5\rceil$ SVP iterations[21], and (ii) the condition number κ is unknown, where we use Lemma 3.4 in [152].

Results using random initialization. Figure 4.2 depicts the convergence performance of the above algorithms w.r.t. total execution time. Top row corresponds to the case $m = n = 1024$, bottom row to the case $m = 2048,\ n = 4096$. For all cases, we fix $r = 50$; from left to right, we decrease the number of available measurements, by decreasing the constant C. Matrix ALPS II shows the best performance in terms of execution time: while still using SVD

[21] Observe that setting $T_{\text{init}} = 1$ leads to spectral method initialization and the algorithm in [184] for non-square cases, given sufficient number of samples.

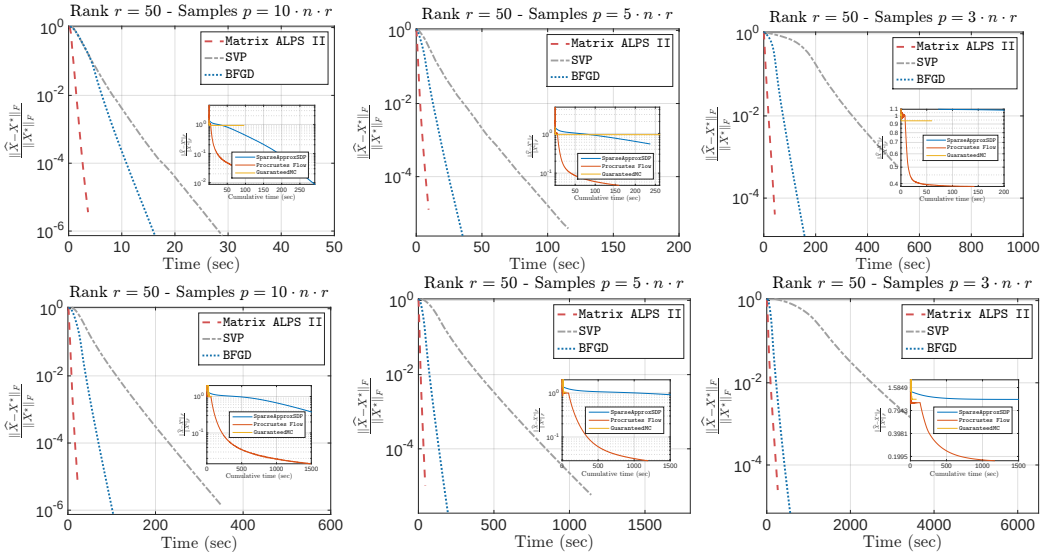


Figure 4.2: Convergence performance of algorithms under comparison w.r.t. $\frac{\|\widehat{X}-X^\star\|_F}{\|X^\star\|_F}$ vs. the total execution time. Top row corresponds to dimensions $m = n = 1024$; bottom row corresponds to dimensions $m = 2048$, $n = 4096$. Details on problem configuration are given on plots' title. For all cases, we used $\mathcal{A}$ as noiselets and $r = 50$.

routines per iteration, Matrix ALPS II is specialized to solve matrix sensing problem instances and performs several subroutines per iteration (subspace exploration, debias steps, adaptive step size selection, among others). Hoever, Matrix ALPS II applies only to this problem. Compared to Matrix ALPS II, `BFGD` shows the second best performance, compared to the rest of the algorithms. It is notable that `BFGD` performs better than SVP, by avoiding SVD calculations and employing a better step size selection.[22] For this setting, `GuaranteedMC` converges to a local minimum while SparseApproxSDP and Procrustes Flow show close to sublinear convergence rate.

[22]If our step size is used in `SVP`, we get slightly better performance, but not in a universal manner.

To further show how the performance of each algorithms scales as dimension increases, we provide aggregated results in Tables 4.2-4.3. Observe that `BFGD` is one order of magnitude faster than the rest non-convex factorization algorithms, while being competitive with MATRIX ALPS II algorithm. Table 4.4 shows the *median time* per iteration, spent by each algorithm, for both problem instances and $C = 3$. Observe that MATRIX ALPS II and SVP require one order of magnitude more time to complete one iteration, mostly due to the SVD step. In stark contrast, all factorization-based approaches spend less time per iteration, as was expected by the discussion in the Introduction section; however, less progress is achieved by performing only matrix-matrix computations.

Results using specific initialization. In this case, we study the effect of initialization in the convergence performance of each algorithm. To do so, we focus only on the factorization-based algorithms: Procrustes Flow, `GuaranteedMC`, and `BFGD`. We consider two problem cases: (i) all these schemes use *our initialization procedure*, and (ii) each algorithm uses its own suggested initialization procedure. The results are depicted in Tables 4.5-4.6, respectively.

Using our initialization procedure for all algorithms, we observe that both Procrustes Flow and `GuaranteedMC` schemes can compute an approximation $\widehat{X}$ that is no closer to $X^\star$ than 10^{-1} normalized distance, *i.e.*, $\frac{\|\widehat{X}-X^\star\|_F}{\|X^\star\|_F} > 10^{-1}$. In contrast, our approach achieves a solution $\widehat{X}$ that is close to the stopping criterion, *i.e.*, $\frac{\|\widehat{X}-X^\star\|_F}{\|X^\star\|_F} \approx 10^{-6}$.

Using different initialization schemes per algorithm, the results are depicted 4.6. We remind that `GuaranteedMC` is designed for matrix completion

tasks, where the linear operator is a selection mask of the entries. Observe that Procrustes Flow's performance improves significantly by using their proposed initialization: the idea is to perform SVP iterations to get to a good initial point; then switch to non-convex factored gradient descent for low per-iteration complexity. However, this initialization is computationally expensive: Procrustes Flow might end up performing several SVP iterations. This can be observed *e.g.*, in the case $m = n = 1024,\ r = 5$ and comparing the results in Tables 4.5-4.6: for this case, Procrustes Flow performs $T = 4000$ iterations when our initialization is used and spends ~ 200 seconds, while in Table 4.6 it performs $T \ll 4000$ iterations, at least 20% of them using SVP, and consumes ~ 2000 seconds.

As a concluding remark, we note that similar results have been observed in noisy settings and, thus, are omitted.

4.8.3 Image denoising as matrix completion problem

In this example, we consider the matrix completion setting for an image denoising task: In particular, we observe a limited number of pixels from the original image and perform a low rank approximation based only on the set of measurements—similar experiments can be found in [97, 169]. We use real data images: While the true underlying image might not be low-rank, we apply our solvers to obtain low-rank approximations.

Figures 4.3-4.5 depict the reconstruction results for three image cases. In all cases, we compute the best 100-rank approximation of each image (see *e.g.*, the top middle image in Figure 4.3, where the full set of pixels is observed) and we observe only the 35% of the total number of pixels, randomly selected—a realization is depicted in the top right plot in Figure 4.3. Given

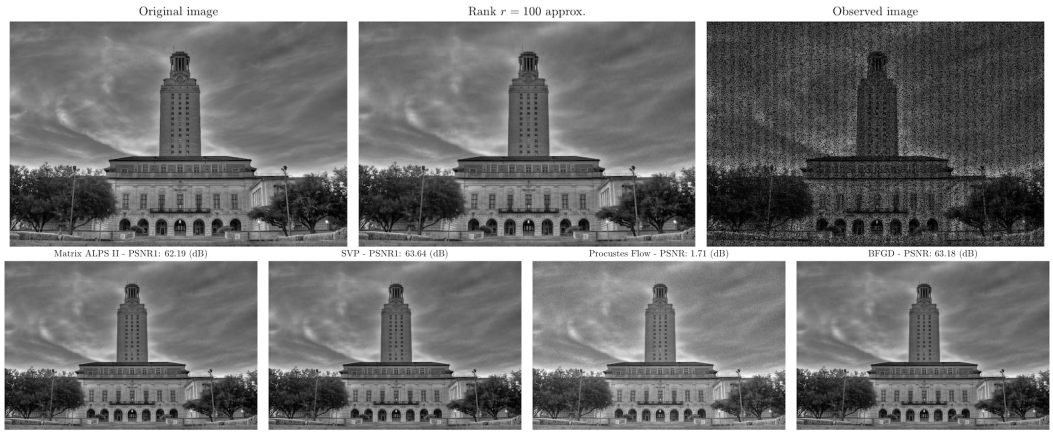


Figure 4.3: Reconstruction performance in image denoising settings. The image size is 2845×4266 ($12,136,770$ pixels) and the approximation rank is preset to $r = 100$. We observe 35% of the pixels of the true image. We depict the median reconstruction error with respect to the true image in dB over 10 Monte Carlo realizations.

a fixed common tolerance level and the same stopping criterion as before, the top rows of Figures 4.3-4.5 show the recovery performance achieved by a range of algorithms under consideration—the peak signal to noise ration (PSNR), depicted in dB, corresponds to median values after 10 Monte-Carlo realizations. In all cases, we note that Matrix ALPS II has overall slightly better performance as compared to the rest of the algorithms, as a more specialized algorithms for matrix completion problems. Our algorithm shows competitive performance compared to simple gradient descent schemes as `SVP` and Procrustes Flow, while being a fast and scalable solver. Table 4.7 contains timing results from 10 Monte Carlo random realizations for all image cases.

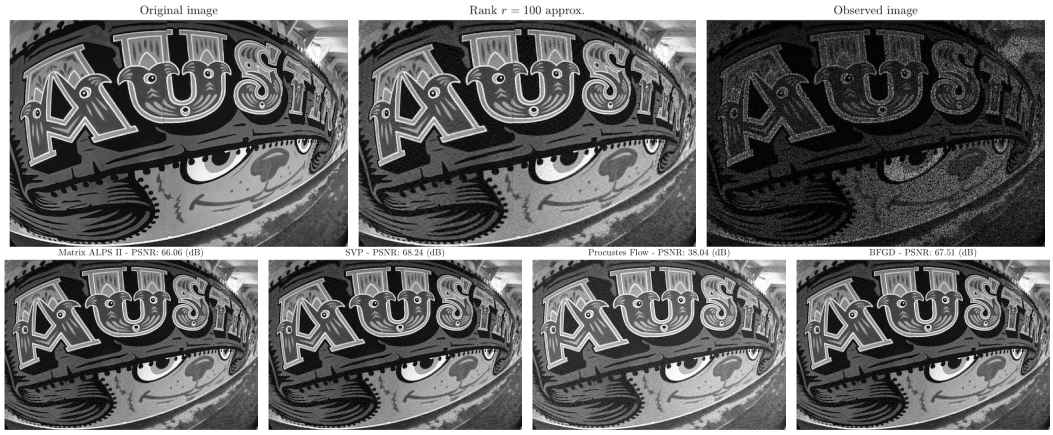


Figure 4.4: Reconstruction performance in image denoising settings. The image size is 3309×4963 $(16,422,567$ pixels) and the approximation rank is preset to $r = 100$. We observe 30% of the pixels of the true image. We depict the median reconstruction error with respect to the true image in dB over 10 Monte Carlo realizations.

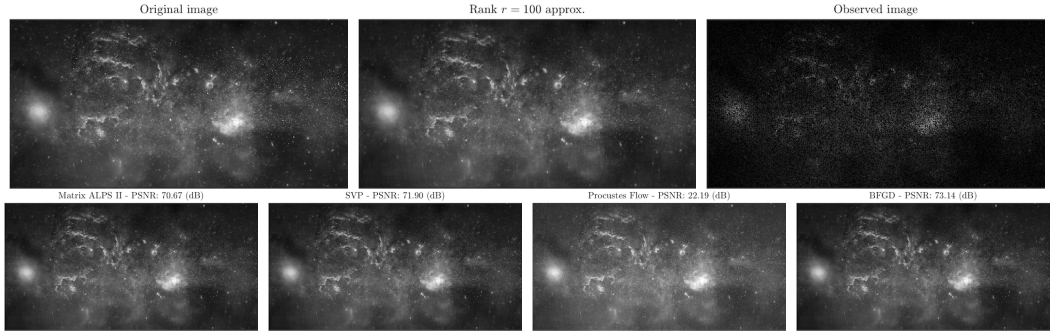


Figure 4.5: Reconstruction performance in image denoising settings. The image size is 4862×9725 $(47,282,950$ pixels) and the approximation rank is preset to $r = 100$. We observe 30% of the pixels of the true image. We depict the median reconstruction error with respect to the true image in dB over 10 Monte Carlo realizations.

4.8.4 1-bit matrix completion

For this task, we repeat the experiments in [45] and compare `BFGD` with their proposed schemes. The observational model we consider here satisfies the

following principles: We assume $X^\star \in \mathbb{R}^{m\times n}$ is an unknown low rank matrix, satisfying $\|X^\star\|_\infty \leq \alpha,\ \alpha > 0$, from which we observe only a subset of indices $\Omega \subset [m] \times [n]$, according to the following rule:

$$Y_{i,j} = \begin{cases} +1 & \text{with probability } \sigma(X^\star_{i,j}) \\ -1 & \text{with probability } 1 - \sigma(X^\star_{i,j}) \end{cases} \qquad \text{for } (i,j) \in \Omega. \tag{4.24}$$

Similar to classic matrix completion results, we assume Ω is chosen uniformly at random, *e.g.*, we assume Ω follows a binomial model, as in [45]. Two natural choices for h function are: (i) the logistic regression model, where $\sigma(x) = \frac{e^x}{1+e^x}$, and (ii) the probit regression model, where $\sigma(x) = 1 - \Phi(-x/\sigma)$ for Φ being the cumulative Gaussian distribution function. Both models correspond to different noise assumptions: in the first case, noise is modeled according to the standard logistic distribution, while in the second case, noise follows standard Gaussian assumptions. Under this model, [45] propose two convex relaxation algorithmic solutions to recover $X^\star$: (i) the convex maximum log-likelihood estimator under nuclear norm and infinity norm constraints:

$$\begin{array}{ll} \underset{X\in\mathbb{R}^{m\times n}}{\text{minimize}} & f(X), \\ \text{subject to} & \|X\|_* \leq \alpha\sqrt{rmn}, \quad \|X\|_\infty \leq \alpha, \end{array} \tag{4.25}$$

and (ii) the the convex maximum log-likelihood estimator under only nuclear norm constraints. In both cases, $f(X)$ satisfies the expression in (4.7). [45] proposes a *spectral projected-gradient descent* method for both these criteria; in the case where only nuclear norm constraints are present, SVD routines compute the convex projection onto norm balls, while in the case where both nuclear and infinity norm constraints are present, [45] propose a alternating-direction method of multipliers (ADMM) solution, in order to compute the joint projection onto these sets.

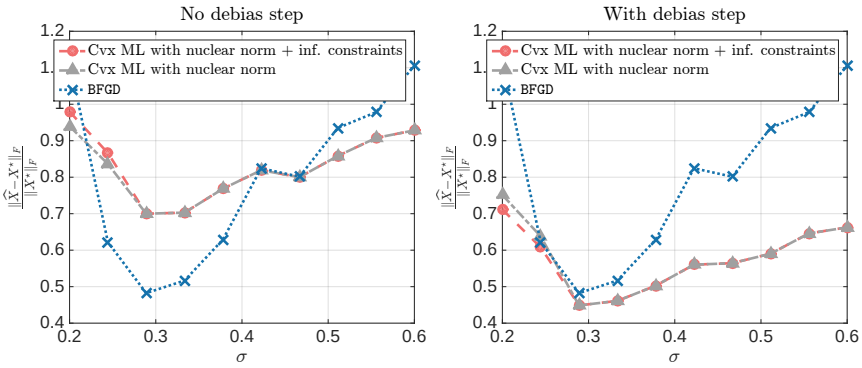


Figure 4.6: Comparison of 1-bit matrix procedures. *Left panel:* Output of (4.25) is not projected onto rank-r set. *Right panel:* Output of (4.25) is projected onto rank-r set.

Synthetic experiments. We synthetically construct $X^\star \in \mathbb{R}^{m\times n}$, where $m = n = 100$, such that $X^\star = U^\star V^{\star\top}$, where $U^\star \in \mathbb{R}^{m\times r}$, $V^\star \in \mathbb{R}^{n\times r}$ for $r = 1$. The entries of $U^\star$, $V^\star$ are drawn i.i.d. from Uni $\left[-\frac{1}{2}, \frac{1}{2}\right]$. Moreover, according to [45], we scale $X^\star$ such that $\|X^\star\|_\infty = 1$. Then, we observe $Y \in \mathbb{R}^{m\times n}$ according to (4.24), where $|\Omega| = \frac{1}{4}\cdot mn$. we consider the probit regression model with additive Gaussian noise, with variance σ^2.

Figure 4.6 depicts the recovery performance of `BFGD`, as compared to variants of (4.25) in [45]. We consider their performance over different noise levels w.r.t. the normalized Frobenius norm distance $\frac{\|\widehat{X}-X^\star\|_F}{\|X^\star\|_F}$. As noted in [45], the performance of all algorithms is poor when σ is too small or too large, while in between, for moderate noise levels, we observe better performance for all approaches.

By default, in all problem settings, we observe that the estimate of (4.25) *is not of low rank*: to compute the closest rank-r approximation to that, we further perform a *debias* step via truncated SVD. The effect of the debias step is better illustrated in Figure 4.6, focusing on the differences between left

and right plot: without such step, BFGD has a better performance in terms of $\frac{\|\widehat{X}-X^\star\|_F}{\|X^\star\|_F}$, within the "sweet" range of noise levels, compared to the convex analog in (4.25). Applying the debias step, both approaches have comparable performance, with that of (4.25) being slightly better.

Perhaps somewhat surprisingly, the performance of BFGD, in terms of *estimating the correct sign pattern* of the entries, is better than that of [45], even with the debias step. Figure 4.7 (left panel) illustrates the observed performances for various noise levels.

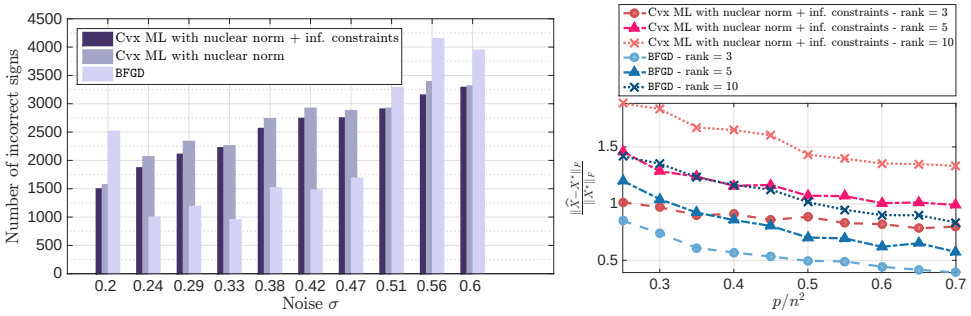


Figure 4.7: *Left panel:* Comparison of 1-bit matrix procedures w.r.t. sign pattern estimation. *Right panel:* Recovery of $X^\star$ from $p = C{\cdot}n^2$ measurements. $X^\star$ is designed to be low rank: $r = 3, 5$ and 10. x-axis represents C for various values.

Finally, we study the performance of the algorithms under consideration as a function of the number of measurements, for fixed settings of dimensions $m = n = 200$ and noise level $\sigma = 0.244$. By the discussion above, such noise level leads to good performance from all schemes. We considered matrices $X^\star$ with rank $r \in \{3,\ 5,\ 10\}$ and generate $p = C \cdot n^2$, over a wide range of $0 < C < 1$. Figure 4.7 (right panel) shows the performance of BFGD and the approach for (4.25) in [45], in terms of the relative Frobenius norm of the error. All approaches do poorly when there are only $p < 0.35 \cdot n^2$ measurements,

since this is near the noiseless information-theoretic limit. For higher numbers of measurements, the non-convex approach in `BFGD` returns more reasonable solutions and outperforms convex approaches, taking advantage of the prior knowledge on low-rankness of the solution.

Recommendation system using the MovieLens dataset. We compare 1-bit matrix completion solvers on the 100k MovieLens dataset. To do so, we repeat the experiment in Section 4.3 of [45]: we use the MovieLens 100k, which consists of 100k movie ratings, from 1000 users on 1700 movies. Each user entry denotes the movie rating, ranging from 1 to 5. To convert this dataset into 1-bit measurements, we convert these ratings to binary observations by comparing each rating to the average rating for the entire dataset (which is approximately 3.5), according to [45]. To evaluate the performance of the algorithms, we assume part of the observed ratings as unobserved (5k of them) and check if the estimate of $X^\star$, $\widehat{X}$, predicts the sign of these ratings. We perform ML estimation using logistic function $h(x) = \frac{e^x}{1+e^x}$ in f.

We compare the following algorithms: (i) the spectral projected gradient descent (`SPG`) implementation of (4.25) in [45] for 1-bit matrix completion, (ii) the standard matrix completion implementation `TFOCS` [15], where we observe the *unquantized* dataset (actual values)[23], (iii) `BFGD` for various values of rank parameter r. The results are shown Table 4.8 over 10 Monte Carlo realizations (*i.e.*, we randomly selected 5k ratings as test sets and solved the problem for different runs of the algorithms). The values in Table 4.8 denote the accuracy in predicting whether the unobserved ratings are above or below

[23]Using `TFOCS`, we set the regularizer $\mu = 10^{-3}$ as the parameter value that returned the best recovery results over a wide range of μ values.

the average rating of 3.5. `BFGD` shows competitive performance, compared to convex approaches. Moreover, setting the parameter r is an "easier" and more intuitive task: our algorithm administers precise control on the rankness of the solution, which might lead to further interpretation of the results. Convex approaches lack of this property: the mapping between the regularization parameters and the number of rank-1 components in the extracted solution is highly non-linear. At the same time, `BFGD` shows much faster convergence to a good solution, which constitutes it a preferable algorithmic solution for large scale applications.

Algorithm	$r = 50$, $C = 10$		$r = 50$, $C = 5$		$r = 50$, $C = 3$	
	$\frac{\|\widehat{X}-X^\star\|_F}{\|X^\star\|_F}$	Total time	$\frac{\|\widehat{X}-X^\star\|_F}{\|X^\star\|_F}$	Total time	$\frac{\|\widehat{X}-X^\star\|_F}{\|X^\star\|_F}$	Total time
`Matrix ALPS II`	1.6096e-06	3.9296	1.0070e-05	9.6347	3.7877e-05	42.9211
`SVP`	6.8623e-07	29.0563	3.7511e-06	115.9088	5.7362e-04	517.5673
`Procrustes Flow`	8.6546e-03	291.0442	5.4418e-01	236.4496	1.0831e+00	223.2486
`SparseApproxSDP`	1.5616e-02	223.1522	4.9298e-02	158.5459	3.8444e-01	141.5906
`GuaranteedMC`	9.2570e-01	95.5135	9.3168e-01	260.8471	9.3997e-01	59.4259
`BFGD`	7.0830e-07	16.2818	2.3199e-06	35.3988	1.1575e-05	157.6610

Table 4.2: Summary of comparison results for reconstruction and efficiency. Here, $m = n = 1024$, resulting into $1,048,576$ variables to optimize, and $\mathcal{A}$ is a noiselet-based subsampled linear map. The number of samples p satisfies $p = C \cdot n \cdot r$ for various values of constant C. Time reported is in seconds.

Algorithm	$r = 50$, $C = 10$		$r = 50$, $C = 5$		$r = 50$, $C = 3$	
	$\frac{\|\widehat{X}-X^\star\|_F}{\|X^\star\|_F}$	Total time	$\frac{\|\widehat{X}-X^\star\|_F}{\|X^\star\|_F}$	Total time	$\frac{\|\widehat{X}-X^\star\|_F}{\|X^\star\|_F}$	Total time
`Matrix ALPS II`	2.9961e-06	23.7907	7.3350e-06	46.7165	2.2546e-05	266.1574
`SVP`	1.4106e-06	349.7021	5.6928e-06	1144.7489	1.4110e-04	4703.2012
`Procrustes Flow`	2.1174e-01	1909.4748	8.7944e-01	1653.7020	1.1088e+00	1692.7454
`SparseApproxSDP`	1.4268e-02	1484.2220	2.8839e-02	1187.5287	1.7544e-01	1165.4210
`GuaranteedMC`	1.0114e+00	69.2279	1.0465e+00	53.1636	1.1147e+00	78.2304
`BFGD`	6.9736e-07	103.8387	1.7946e-06	195.5111	6.6787e-06	561.8248

Table 4.3: Summary of comparison results for reconstruction and efficiency. Here, $m = 2048$, $n = 4096$, resulting into $8,388,608$ variables to optimize, and $\mathcal{A}$ is a noiselet-based subsampled linear map. The number of samples p satisfies $p = C \cdot n \cdot r$ for various values of constant C. Time reported is in seconds.

Algorithm	$m = n = 1024$, $C = 3$ Median time per iter.	$m = 2048,\ n = 4096$, $C = 3$ Median time per iter.
`Matrix ALPS II`	2.360e-01	2.461e+00
`SVP`	1.604e-01	1.040e+00
`Procrustes Flow`	5.871e-02	4.525e-01
`SparseApproxSDP`	3.407e-02	3.001e-01
`GuaranteedMC`	7.142e-02	4.059e-01
`BFGD`	5.337e-02	3.981e-01

Table 4.4: Median time per iteration. Time reported is in seconds.

Algorithm	$m = n = 1024$, $C = 10$, $r = 50$ $\frac{\|\widehat{X}-X^\star\|_F}{\|X^\star\|_F}$	Total time	$m = n = 1024$, $C = 10$, $r = 5$ $\frac{\|\widehat{X}-X^\star\|_F}{\|X^\star\|_F}$	Total time
`Procrustes Flow`	2.2703e+01	281.2095	4.0432e+01	192.0993
`GuaranteedMC`	9.2570e-01	96.8512	4.7646e-01	2.4792
`BFGD`	3.7055e-06	52.5205	8.1246e-06	65.4926

Table 4.5: Summary of results of factorization algorithms using our proposed initialization.

Algorithm	$m = n = 1024$, $C = 10$, $r = 50$ $\frac{\|\widehat{X}-X^\star\|_F}{\|X^\star\|_F}$	Total time	$m = n = 1024$, $C = 10$, $r = 5$ $\frac{\|\widehat{X}-X^\star\|_F}{\|X^\star\|_F}$	Total time
`Procrustes Flow`	3.2997e-05	390.6830	8.5741e-04	2017.7942
`GuaranteedMC`	9.2570e-01	114.9332	1.0114e+00	68.1775
`BFGD`	3.6977e-06	64.2690	3.1471e-06	74.2345
Algorithm	$m = 2048, n = 4096$, $C = 10$, $r = 50$ $\frac{\|\widehat{X}-X^\star\|_F}{\|X^\star\|_F}$	Total time	$m = 2048, n = 4096$, $C = 10$, $r = 5$ $\frac{\|\widehat{X}-X^\star\|_F}{\|X^\star\|_F}$	Total time
`Procrustes Flow`	4.9896e-02	265.2787	4.2263e-02	1497.6867
`GuaranteedMC`	4.7646e-01	4.0752	1.0302e+00	35.0559
`BFGD`	8.1381e-06	83.3411	5.8428e-06	379.1430

Table 4.6: Summary of results of factorization algorithms using each algorithm's proposed initialization.

Algorithm	Time (sec.)		
	UT Campus	Graffiti	Milky way
`Matrix ALPS II`	2550.2	2495.9	7332.5
`SVP`	5224.1	4154.9	7921.4
`Procrustes Flow`	5383.4	6501.4	12806.3
`BFGD`	4062.4	3155.9	9119.6

Table 4.7: Summary of execution time results for the problem of image denoising. Timings correspond to median values on 10 Monte Carlo random instantiations.

Algorithm	Ratings (%)					Overall (%)	Time (sec)
	1	2	3	4	5		
`SPG` ($\alpha\sqrt{r} = 0.32$)	73.7	68.4	52.5	74.9	91.0	71.3	79.5
`SPG` ($\alpha\sqrt{r} = 4.64$)	77.2	71.0	58.5	72.5	86.9	71.8	213.4
`SPG` ($\alpha\sqrt{r} = 10.00$)	76.2	71.3	58.3	71.0	85.7	71.0	491.8
`TFOCS`	70.4	69.4	59.2	39.1	59.4	64.8	42.3
`BFGD` ($r = 3$)	79.4	74.5	56.9	72.5	88.2	72.2	25.4
`BFGD` ($r = 5$)	79.0	72.4	56.8	71.6	86.2	71.2	27.5
`BFGD` ($r = 10$)	77.6	75.0	57.5	70.5	84.1	70.9	30.3

Table 4.8: Summary ofresults for the problem of 1-bit matrix completio on MovieLens dataset. Individual and overall ratings correspond to percentages of signs correctly estimated (+1 corresponds to original rating above 3.5, -1 corresponds to original rating below 3.5). Timings correspond to median values on 10 Monte Carlo random instantiations.

Chapter 5

Conclusion

In this dissertation, we have presented efficient algorithms for non-convex learning problems. We first developed algorithms for two particular problems, subspace clustering for learning unions of subspaces and collaborative ranking for learning low-rank matrices from pairwise comparisons. Then we designed a non-convex gradient descent algorithm for a more general matrix optimization problem which arises in several applications, such as matrix sensing and matrix completion.

All of these problems, which stem from the unprecedently high volume and dimensionality of modern datasets, require to estimate statistical models for which the generic maximum likelihood estimations cannot be efficiently solved by classical methods such as convex optimization and singular value decomposition. While convex relaxation can provide polynomial-time algorithms for a wide range of non-convex learning problems, there are needs for developing even faster algorithms with competitive performances. Providing such algorithms for several particular problems, this dissertation makes efforts to deal with the continued growth of data sizes.

An interesting direction for this line of research is application of randomization and parallelism to these non-convex approaches. While most of our approaches are currently simple, deterministic, and serial, the techniques such as subsampling, randomized dimensionality reduction, and stochastic op-

timization will improve the scalability of the designed algorithms. Also, we believe that all of our non-convex algorithms can be parallelized very well. These ideas will help our algorithms more scalable and improve their practical performances for even larger datasets than those we have used in this work.

Appendices

Appendix A

Technical Proofs for Chapter 2

A.1 Proofs for Chapter 2

A.1.1 Preliminary lemmas

Before we step into the technical parts of the proof, we introduce the technical ingredients which will be used. The following lemma is about upper and lower bounds on the order statistics for the projections of iid Gaussian vectors.

Lemma A.1. *Let $x_1, \ldots, x_n$ be drawn iid from $\mathcal{N}(0, \frac{1}{d} I_{d\times d})$. Let $z_{(n-m+1)}$ denote the m'th largest value of $\{z_i \triangleq \|Ax_i\|_2, 1 \le i \le n\}$ where $A \in \mathbb{R}^{k\times d} (k \le d)$.*

a. Suppose the rows of A are orthonormal to each other. If

$$\frac{n}{m} \ge 6 \left(\log \frac{ne}{m\delta}\right)^2 \tag{A.1}$$

we have

$$z^2_{(n-m+1)} > \frac{k}{d} + \frac{2}{d} \cdot \log \left(\frac{n-m+1}{6m \left(\log \frac{ne}{m\delta}\right)^2} \right)$$

with probability at least $1 - \delta^m$.

b. Let A be arbitrary full-rank matrix. Suppose $m \le d$. If

$$\frac{n}{m} \ge 9 d^\alpha \log \frac{ne}{m\delta},$$

for some $\alpha > 0$, then we have

$$z^2_{(n-m+1)} > \frac{\|A\|_F^2}{d} \cdot \left(1 + \frac{1}{2k} \log\left(\frac{n-m+1}{m \log \frac{ne}{m\delta}}\right)\right)$$

for $1 \le k \le \max\{1, \lfloor \frac{\alpha}{2} \log d \rfloor\}$ with probability at least $1 - \delta^m$.

The following lemma provides a concentration bounds for uniformly random subspaces.

Lemma A.2. *Let the columns of $X \in \mathbb{R}^{d\times k}$ be the orthonormal basis of a k-dimensional random subspace drawn uniformly at random in d-dimensional space.*

a. For any matrix $A \in \mathbb{R}^{p\times d}$.

$$E[\|AX\|_F^2] = \frac{k}{d}\|A\|_F^2$$

b. [120, 102] If $\|A\|_2$ is bounded, then we have

$$\Pr\left\{\|AX\|_F > \sqrt{\frac{k}{d}}\|A\|_F + \|A\|_2 \cdot \left(\sqrt{\frac{8\pi}{d-1}} + t\right)\right\} \le e^{-\frac{(d-1)t^2}{8}}.$$

A.1.2 Proof of Lemma 2.7

Step 1: Lower bounds on the projection of correct points

Let $V_k \in \mathbb{R}^{p\times k}$ be such that $V_k \in \mathcal{BV}_k$. In this step, we want to lower bound the LHS of (2.4), which can be written as

$$\|V_k^\top y_{j_k^*}\|_2 = \|V_k^\top D_1 x_{j_k^*}\|_2$$

It is not trivial to analyze $\|V_k^\top D_1 x_{j_k^*}\|_2$ because V_k and $x_{j_k^*}$ are dependent. $D_1 x_{j_k^*}$ should not be too close to $V_1, \ldots, V_{k-1}$ since it has not been selected in

the preceding steps. In order to avoid this dependence, we instead analyze another random variable that is stochastically dominated by $\|V_k^\top D_1 x_{j_k^*}\|_2^2$. Then we use a high-probability lower bound on that variable which also lower bounds $\|V_k^\top D_1 x_{j_k^*}\|_2^2$ with high probability.

Lemma A.3. *For a fixed k, let $\hat{x}_1, \ldots, \hat{x}_{n-1}$ be drawn iid uniformly at random from $\mathcal{N}(0, \frac{1}{d} I_{d\times d})$, independently of V_k. Define $\rho_{(n-m)}$ as the m'th largest value of $\|V_k^\top D_1 \hat{x}_1\|_2, \ldots, \|V_k^\top D_1 \hat{x}_{n-1}\|_2$. Then*

$$\|V_k^\top D_1 x_{j_k^*}\|_2^2 \overset{d}{\geq} \rho_{(n-k)}^2.$$

Now we can apply Lemma A.1a to $\rho_{(n-k)}^2$ in the above lemma. Since we assume $n = \Omega(d)$, the condition (A.1) holds for any $m \in [d]$. With probability at least $1 - \delta^k$, we have

$$\begin{aligned}
\|V_k^\top D_1 x_{j_k^*}\|_2^2 &\geq \frac{k \wedge k_{\max}}{d} + \frac{2}{d} \log \left(\frac{n-k+1}{6k \left(\log \frac{ne}{k\delta} \right)^2} \right) \\
&\geq \tfrac{1}{d} \cdot \left(2 \log \frac{n-d}{6} + (k \wedge k_{\max}) - 2 \log k - 4 \log \log \frac{ne}{\delta} \right) \\
&\geq \frac{c_4 \log n}{d} \qquad \text{(A.2)}
\end{aligned}$$

for some constant $c_4 > 0$, because $(k \wedge k_{\max}) \geq 2 \log k$. The union bound gives that (A.2) holds for all $k \in [K]$ simultaneously with probability at least $1 - \frac{\delta}{1-\delta}$.

Step 2: Upper bounds on the projection of incorrect points

The RHS of (2.4) can be written as

$$\max_{j : j \in [N], w_j \neq 1} \|V_k^\top y_j\|_2 = \max_{j : j \in [N], w_j \neq 1} \|V_k^\top D_{w_j} x_j\|_2 \qquad \text{(A.3)}$$

In this step, we want to bound (A.3) for every $k \in [K]$ by using the concentration inequalities for V_k and x_j. Note that the points of $\{x_j : w_j \neq 1\}$ are all independent of each other and independent of $V_1, \ldots, V_K$. (Remark 2.6)

Consider a point $y_j = D_l x_j$ from $\mathcal{D}_l$. Let $V_k^\top D_l = U\Sigma V^\top$ be the SVD. Since we have

$$\|V_k^\top D_l x_j\|_2^2 = \|U\Sigma V^\top x_j\|_2^2 = \|\Sigma x_j\|_2^2 = \sum_{i=1}^{k} \sigma_i^2 x_{j,i}^2,$$

$\|V_k^\top D_l x_j\|_2^2$ is a weighted Chi-square random variable. It follows from [101] that

$$\Pr\left\{ d \cdot \|V_k^\top D_l x_j\|_2^2 \geq \|V_k^\top D_l\|_F^2 + 2\sqrt{\sum_{i=1}^{k} \sigma_i^4} \cdot \sqrt{t} + 2\|V_k^\top D_l\|_2^2 \cdot t \right\} \leq \exp(-t)$$

When $t \geq 1$, we obtain

$$\Pr\left\{ \|V_k^\top D_l x_j\|_2^2 \geq \frac{\|V_k^\top D_l\|_F^2}{d} \cdot 5t \right\} \leq \exp(-t)$$

Applying the union bound, we obtain that with probability $1 - \delta$,

$$\max_{j: j \in [N], w_j \neq 1} \|V_k^\top D_{w_j} x_j\|_2^2 \leq \frac{\max_{l \neq 1} \|V_k^\top D_l\|_F^2}{d} \cdot 5 \log \frac{n k_{\max} L}{\delta} \tag{A.4}$$

for all $k = 1, \ldots, k_{\max} - 1$.

Now let us consider $\max_{l \neq 1} \|V_k^\top D_l\|_F$. In our statistical model, the new axis added to $\mathcal{V}_k$ at the kth step (u_{k+1} in Algorithm 3) is chosen uniformly at random from the subspace in $\mathcal{D}_1$ orthogonal to $\mathcal{V}_k$. Therefore, V_k is a random matrix drawn uniformly from the $d \times k$ Stiefel manifold, and the probability measure is the normalized Haar (rotation-invariant) measure. From Lemma

A.1b and the union bound, we obtain that with probability at least $1-\delta/dL$,

$$\begin{aligned}\|V_k^\top D_l\|_F &\le \sqrt{\frac{k\wedge k_{\max}}{d}}\|D_1^\top D_l\|_F + \|D_1^\top D_l\|_2 \cdot \left(\sqrt{\frac{8\pi}{d-1}} + \sqrt{\frac{8}{d-1}\log\frac{dL}{\delta}}\right)\\ &\le \|D_1^\top D_l\|_F \cdot \left(\sqrt{\frac{k\wedge k_{\max}}{d}} + \sqrt{\frac{50}{d}\log\frac{dL}{\delta}}\right)\\ &\le \max \mathrm{aff} \cdot \left(\sqrt{2\log d} + \sqrt{50\log\frac{dL}{\delta}}\right). \end{aligned} \tag{A.5}$$

The union bound gives that with probability at least $1-\delta$, $\max_{l\neq 1}\|V_k^\top D_l\|_F$ is also bounded by (A.5) for every $k = 1,\dots,k_{\max}$.

Putting (A.4) and (A.5) together, we obtain

$$\begin{aligned}&\max_{j:j\in[N],w_j\neq 1}\|V_k^\top D_{w_j}x_j\|_2\\ &\le \max \mathrm{aff} \cdot \left(\sqrt{2\log d} + \sqrt{50\log\frac{dL}{\delta}}\right) . \cdot \sqrt{5\log\frac{ndL}{\delta}}\end{aligned} \tag{A.6}$$

for all $k = 1,\dots,d-1$ with probability at least $1-2\delta$.

Step 3: Proof of the statement

Putting (A.2) and (A.6) together, we obtain that if

$$\max \mathrm{aff} < \frac{\sqrt{c_4\log n}}{\left(\sqrt{2\log d} + \sqrt{50\log\frac{dL}{\delta}}\right) . \cdot \sqrt{5\log\frac{ndL}{\delta}}}, \tag{A.7}$$

then (2.4) holds, and hence NSN finds all correct neighbors for y_1 with probability at least $1-\frac{3\delta}{1-\delta}$. We can see that (A.7) holds because of the assumption of the lemma.

A.1.3 Proof of Lemma 2.8

Step 1: Lower bounds on the projection of correct points

Since there is no different probability relation for the subspace $\mathcal{D}_1$ and the points from $\mathcal{D}_1$, we again use Lemma A.3. Then we have

$$\|V_k^\top D_1 x_{j_k^*}\|_2^2 \geq \frac{c_4 \log n}{d} \tag{A.8}$$

for all $k = 1, \ldots, d-1$ simultaneously with probability at least $1 - \frac{\delta}{1-\delta}$.

Step 2: Upper bounds on the projection of incorrect points

Since $\|\operatorname{Proj}_{\mathcal{V}_k} y_j\|_2 = \|V_k^\top y_j\|_2 = \|V_k^\top D_1 x_j\|_2$, the RHS of (2.4) can be written as

$$\max_{j: j \in [N], w_j \neq 1} \|V_k^\top y_j\|_2 \tag{A.9}$$

Since the true subspaces are independent of each other, y_j with $w_j \neq 1$ is also independent of D_1 and V_k, and its marginal distribution is $\mathcal{N}(0, \frac{1}{p} I_{p\times p})$. Hence, $p \cdot \|V_k^\top y_j\|_2^2$ is a Chi-square random variable with $(k \wedge k_{\max})$ degrees of freedom, we have

$$\Pr\left\{p \cdot \|V_k^\top y_j\|_2^2 \geq 2(k \wedge k_{\max}) + 3t\right\} \leq \exp(-t)$$

It follows that with probability at least $1 - \delta/d$,

$$\begin{aligned} \max_{j: j \in [N], w_j \neq 1} \|V_k^\top y_j\|_2^2 &\leq \frac{2(k \wedge k_{\max}) + 3\log \frac{ndL}{\delta}}{p} \\ &\leq \frac{4\log d + 3\log \frac{ndL}{\delta}}{p}. \end{aligned} \tag{A.10}$$

The union bound provides that (A.10) holds for every $k = 1, \ldots, d-1$ with probability at least $1 - \delta$.

Step 3: Proof of the main theorem

Putting (A.8) and (A.10) together, we obtain that if

$$\frac{d}{p} < \frac{c_4 \log n}{7 \log \frac{ndL}{\delta}}, \tag{A.11}$$

then (2.4) holds, and hence NSN finds all correct neighbors for y_1 with probability at least $1 - \frac{3\delta}{1-\delta}$. We can see that (A.11) holds because of the assumption of the lemma.

A.1.4 Proof of Lemma A.3

The key idea is that the projection of the non-selected points onto the new axis at step k in $\mathcal{P}_k$[1] are independent of the direction of that axis.

We construct a generative model for two random variables that are equal in distribution to $\|V_k^\top D_1 x_{j_k^\dagger}\|_2^2$ and $\rho^2_{(n-k)}$. Then we show that the one corresponding to $\|V_k^\top D_1 x_{j_k^\dagger}\|_2^2$ is greater than the other corresponding to $\rho^2_{(n-k)}$. This generative model uses the fact that for any isotropic distributions the marginal distributions of the components along any orthogonal axes are invariant.

Let $D_1^\top V_{k_{\max}} = QR$ be the reduced QR decomposition where

$$Q \in \mathbb{R}^{d \times k_{\max}}, Q^\top Q = I_{k_{\max} \times k_{\max}}, R \in \mathbb{R}^{k_{\max} \times k_{\max}}.$$

The generative model is given as follows.

- For $k = 1, \ldots, k_{\max}$, repeat 2.
- Draw $n-1$ iid random variables $W_1^{(k)}, \ldots, W_{n-1}^{(k)}$ from $\mathcal{N}(0, \frac{1}{d})$, and draw

[1] This axis is lying on $\mathcal{P}_k$ but orthogonal to $\mathcal{P}_{k-1}$

other $n-1$ iid random variables $Z_1^{(k)}, \ldots, Z_{n-1}^{(k)}$ from $\mathcal{N}(0, \frac{\sigma^2}{p})$. Define

$$X_j^{(k)} = X_j^{(k-1)} + \left(\sum_{\ell=1}^{k} R_{\ell k} \cdot W_j^{(\ell)} \right)^2,$$
$$Y_j^{(k)} = Y_j^{(k-1)} + \left(\left(\sum_{i=\ell}^{k} R_{\ell k} \cdot W_j^{(\ell)} \right) + Z_j^{(\ell)} \right)^2,$$

and find

$$\pi_k^\dagger \triangleq \arg \max_{j: j \neq \pi_1^*, \ldots, \pi_{k-1}^*} X_j^{(k)},$$
$$\pi_k^* \triangleq \arg \max_{j: j \neq \pi_1^*, \ldots, \pi_{k-1}^*} Y_j^{(k)}.$$

- For $k = k_{\max} + 1, \ldots, K$, repeat finding

$$\pi_k^\dagger \triangleq \arg \max_{j: j \neq \pi_1^*, \ldots, \pi_{k-1}^*} X_j^{(k_{\max})},$$
$$\pi_k^* \triangleq \arg \max_{j: j \neq \pi_1^*, \ldots, \pi_{k-1}^*} Y_j^{(k_{\max})}.$$

Before we claim the stochastic equivalence, we note that

$$V_k^\top D_1 = R_k^\top Q_k^\top$$

for every $k = 1, \ldots, k_{\max}$, where $Q_k \in \mathbb{R}^{d \times k}$ is the left submatrix of Q, and $R_k \in \mathbb{R}^{k \times k}$ is the upper left submatrix of R. We will use Q_k and R_k in the following.

We first claim that the following two sets of random variables are equal in distribution.

$$A_k \triangleq \left\{ (W_j^{(k)}, Z_j^{(k)}) : j \in [n-1], j \neq \pi_1^*, \ldots, \pi_{k-1}^* \right\},$$
$$B_k \triangleq \left\{ (q_k^\top x_j, v_k^\top z_j) : w_j = 1, j \neq 1, j_1^*, \ldots, j_{k-1}^* \right\}.$$

Since j^*_{k-1} is a function of

$$\left\{(V_{k-1}^\top D_1 x_j, V_{k-1}^\top z_j) : w_j = 1, j \neq 1, j_1^*, \ldots, j_{k-1}^*\right\}$$

$$\left\{((I - V_{k-1}V_{k-1})^\top x_j, (I - V_{k-1}V_{k-1})^\top z_j) : w_j = 1, j \neq 1, j_1^*, \ldots, j_{k-1}^*\right\}$$

Since x_j's are isotropically

Then we have

$$\begin{aligned}
X^{(k)}_{\pi_k^\dagger} &= \max_{j: j \neq \pi_1^*, \ldots, \pi_{k-1}^*} X_j^{(k)} \\
&= \max_{j: j \neq \pi_1^*, \ldots, \pi_{k-1}^*} \sum_{i=1}^{k} \left(\sum_{\ell=1}^{i} R_{\ell i} \cdot W_j^{(\ell)} \right)^2 \\
&\overset{d}{=} \max_{j: w_j = 1, j \neq 1, j_1^*, \ldots, j_{k-1}^*} \sum_{i=1}^{k} \left(\sum_{\ell=1}^{i} R_{\ell i} \cdot (q_\ell^\top x_j) \right)^2 \\
&= \max_{j: w_j = 1, j \neq 1, j_1^*, \ldots, j_{k-1}^*} \|R_k^\top Q_k^\top x_j\|_2^2 \\
&= \max_{j: w_j = 1, j \neq 1, j_1^*, \ldots, j_{k-1}^*} \|V_k^\top D_1 x_j\|_2^2 \\
&= \|V_k^\top D_1 x_{j_k^*}\|_2^2.
\end{aligned}$$

Similarly, it is shown that the k'th maximum of $\{X_j^{(k)} : j \in [n-1]\}$ is equal in distribution to the k'th maximum of $\{\|V_k^\top D_1 \hat{x}_j\|_2^2 : j \in [n-1]\}$, because $\hat{x}_j$'s are independent of Q and R, and hence $(q_\ell^\top \hat{x}_j)$ is a single gaussian $\mathcal{N}(0, \frac{1}{d})$. Since $X^{(k)}_{\pi_k^\dagger}$ is the maximum in a subset with $n-k$ variables of $\left\{X_1^{(k)}, \ldots, X_{n-1}^{(k)}\right\}$, it is greater than or equal to the k'th maximum of the set. Therefore, $\|V_k^\top D_1 x_{j_k^\dagger}\|_2^2$ stochastically dominates $P^2_{k,(k)}$.

Now it suffices to show that A_k is stochastically equivalent to B_k. We prove by induction.

- Base case : The elements of $\{x_j : w_j = 1, j \neq 1\}$ are iid isotropic Gaussians with $\mathcal{N}(0, \frac{1}{d} I_{d\times d})$ independent of $Q_1 = q_1$. Therefore, the elements of $\{q_1^\top x_j : w_j = 1, j \neq 1\}$ are iid univariate Gaussians with $\mathcal{N}(0, \frac{1}{d})$, which are stochastically equivalent to $\{W_1^{(1)}, \ldots, W_{n-1}^{(1)}\}$.

  Similarly, the elements of $\{z_j : w_j = 1, j \neq 1\}$ are iid isotropic Gaussians with $\mathcal{N}(0, \frac{1}{p} I_{p\times p})$ independent of $V_1 = v_1$. Therefore, the elements of $\{v_1^\top z_j : w_j = 1, j \neq 1\}$ are iid univariate Gaussians with $\mathcal{N}(0, \frac{1}{p})$, which are stochastically equivalent to $\{Z_1^{(1)}, \ldots, Z_{n-1}^{(1)}\}$.

- Induction: Assume that the joint distribution of A_{k-1} is equal to the joint distribution of B_{k-1}. It is sufficient to show that given $A_{k-1} \stackrel{d}{=} B_{k-1}$ the conditional joint distribution of $A_k = \{(W_j^{(k)}, Z_j^{(k)}) : j \in [n-1], j \neq \pi_1^*, \ldots, \pi_k^*\}$ is equal to the conditional joint distribution of $B_k = \{(q_k^\top x_j, v_k^\top z_j) : w_j = 1, j \neq 1, j_1^*, \ldots, j_k^*\}$.

  The two terms are independent of each other because $V_k \perp v_k$, and x_j is isotropically distributed. Hence, we only need to show that $((v_k^\top x_j)^2 : w_j = 1, j \neq 1, j_1^*, \ldots, j_k^*)$ is equal in distribution to $(Y_j^{(k+1)} : j \in [n-1], j \neq \pi_1, \ldots, \pi_k)$.

  Since v_k is a normalized vector on the subspace $\mathcal{V}_k^\perp \cap \mathcal{D}_1$, and $x_{j_k^*}$ is drawn iid from an isotropic distribution, v_k is independent of $V_k^\top x_{j_k^*}$. Hence, the marginal distribution of v_k given $\mathcal{V}_k$ is uniform over $(\mathcal{V}_k^\perp \cap \mathcal{D}_1) \cap \mathbb{S}^{p-1}$. Also, v_k is also independent of the points $\{x_j : w_j = 1, j \neq 1, j_1^*, \ldots, j_k^*\}$. Therefore, the random variables $(v_k^\top x_j)^2$ for j with $w_j = 1, j \neq 1, j_1^*, \ldots, j_k^*$ are iid equal in distribution to $Y_j^{(k+1)}$ for any j.

A.1.5 Proof of Lemma 2.9

The proof is similar to the proof of Lemma 2.7. The difference is that $\mathcal{V}_k$ is no longer a subspace of $\mathcal{D}_1$, since the points are not exactly lying on the subspaces. In this case, we need to introduce the following notations.

Definition A.4. *For each $\mathcal{V}_k$, we define $\mathcal{P}_k$ and $\mathcal{Q}_k$ as the projection of $\mathcal{V}_k$ onto $\mathcal{D}_1$ and the null space of $\mathcal{D}_1$, namely,*

$$\mathcal{P}_k \triangleq \{\mathrm{Proj}_{\mathcal{D}_1} y : y \in \mathcal{V}_k\}, \quad \mathcal{Q}_k \triangleq \{\mathrm{Proj}_{\mathcal{D}_1^\perp} y : y \in \mathcal{V}_k\}.$$

Definition A.5. *Let $V_k \in \mathbb{R}^{p\times k}$ be such that $V_k \in \mathcal{B}\mathcal{V}_k$, and the left $k-1$ columns of V_k is identical to V_{k-1}. This can be obtained by stacking the new axis at each step as a column.*

Definition A.6. *Let $P_k \in \mathbb{R}^{p\times k}$ be such that $P_k \in \mathcal{B}\mathcal{P}_k$, and the left $k-1$ columns of P_k is identical to P_{k-1}. This can be obtained as follows.*

$$P_k = D_1 Q, \quad A \in \mathbb{R}^{d\times k}$$

where $D_1^\top V_k = QR$ is the reduced QR decomposition.

Remark A.7. *For every k, the marginal distributions of $\mathcal{P}_k$ and $\mathcal{Q}_k$ are uniform over $\mathcal{B}_k\mathcal{D}_1$ and $\mathcal{B}_k\mathcal{D}_1^\perp$, respectively.*

Step 1: Lower bounds on the projection of correct points

In this step, we want to lower bound the LHS of (2.4), which can be written as

$$\|V_k^\top y_{j_k^*}\|_2.$$

Define

$$j_k^\dagger = \arg \max_{j \in [N] \setminus \mathcal{J}_k : w_j = 1} \|V_k^\top D_1 x_j\|_2$$

for every $k = 1, \ldots, K$. Since the definition of j_k^* and the triangle inequality lead to

$$\begin{aligned} \|V_k^\top y_{j_k^*}\|_2 &\geq \|V_k^\top y_{j_k^\dagger}\|_2 \\ &\geq \|V_k^\top D_1 x_{j_k^\dagger}\|_2 - \|V_k^\top z_{j_k^\dagger}\|_2, \end{aligned}$$

we instead lower bound $\|V_k^\top D_1 x_{j_k^\dagger}\|_2$ and $\|V_k^\top z_{j_k^\dagger}\|_2$ separately. The same technique as Lemma A.3 can be used to obtain such bounds.

Corollary A.8. *For a fixed k, let $\hat{x}_1, \ldots, \hat{x}_{n-1}$ be drawn iid uniformly at random from $\mathcal{N}(0, \frac{1}{d} I_{d \times d})$, independently of V_k. And let $\hat{z}_1, \ldots, \hat{z}_{n-1}$ be drawn iid uniformly at random from $\mathcal{N}(0, \frac{\sigma^2}{p} I_{p \times p})$, independently of V_k. Define $\rho_{(n-m)}$ as the m'th largest value of $\|V_k^\top D_1 \hat{x}_1\|_2, \ldots, \|V_k^\top D_1 \hat{x}_{n-1}\|_2$. Then*

$$\|V_k^\top D_1 x_{j_k^\dagger}\|_2^2 \overset{d}{\geq} \rho_{(n-k)}^2.$$

and

$$\|V_k^\top z_{j_k^\dagger}\|_2^2 \overset{d}{\leq} \max_{i \in [n-1]} \|V_k^\top \hat{z}_i\|_2^2.$$

Lemma A.9. *If $p \geq 64\sigma^2 d$, with probability at least $1 - 4e^{-d/16}$,*

$$\|V_k^\top D_1\|_F \geq \frac{1}{8(1+\sigma)}$$

for all $k = 1, \ldots, k_{\max}$.

Now we can apply Lemma A.1b to obtain a lower bound on $\rho_{(n-k)}^2$. Combined with Corollary A.8, Lemma A.9, and Lemma A.12, we obtain that

with probability $1 - \frac{\delta}{1-\delta} - \frac{1}{n^2} - 4e^{-d/16}$,

$$\begin{aligned}
\|V_k^\top y_{j_k^\dagger}\|_2 &\geq \frac{1}{8(1+\sigma)\sqrt{d}} \cdot \sqrt{\frac{1}{2 \vee (\alpha \log d)} \log\left(\frac{n-d}{d \log \frac{ne}{d\delta}}\right)} - 4\sigma\sqrt{\frac{\log(nd)}{p}} \\
&\geq \frac{1}{8(1+\sigma)\sqrt{d}} \cdot \sqrt{\frac{1}{2 \vee (\alpha \log d)} \log\left(8d^\alpha\right)} - 4\sigma\sqrt{\frac{\log(nd)}{p}} \\
&\geq \frac{c_6}{8(1+\sigma)\sqrt{d}} - 4\sigma\sqrt{\frac{\log(nd)}{p}} \qquad \text{(A.12)}
\end{aligned}$$

for some constant $c_6 > 0$.

Step 2: Upper bounds on the projection of incorrect points

Using the triangle inequality, the RHS of (2.4) can be upper bounded as

$$\begin{aligned}
\max_{j \in [N]: w_j \neq 1} \|V_k^\top y_j\|_2 &= \max_{j \in [N]: w_j \neq 1} \|V_k^\top (D_{w_j} x_j + z_j)\|_2 \\
&\leq \max_{j \in [N]: w_j \neq 1} \|V_k^\top D_{w_j} x_j\|_2 + \max_{j \in [N]: w_j \neq 1} \|V_k^\top D_{w_j} x_j\|_2 \qquad \text{(A.13)}
\end{aligned}$$

In this step, we want to bound (A.13) for every $k = 1, \ldots, k_{\max}$ by using the concentration inequalities for V_k, x_j, and z_j. Note that the points of $\{x_j : w_j \neq 1\}$ are all independent of each other and independent of $V_1, \ldots, V_{k_{\max}}$.

Lemma A.10. *With probability at least* $1 - \frac{1}{(nk_{\max}L)^2}$,

$$\max_{j \in [N]: w_j \neq 1} \|V_k^\top D_{w_j} x_j\|_2^2 \leq \frac{\max_{l \neq 1} \|V_k^\top D_l\|_F^2}{d} \cdot (1 + 12 \log(nk_{\max}L))$$

for all $k = 1, \ldots, k_{\max}$ *simultaneously.*

Lemma A.11. *Suppose* $\delta \leq 10^{-2}$. *With probability at least* $1 - 2\delta$.

$$\max_{l \neq 1} \|V_k^\top D_l\|_F \leq \left(\max \text{aff} \cdot \|V_k^\top D_1\|_F + \sqrt{\frac{d}{p-d}}\right) \cdot \left(\sqrt{k} + \sqrt{50 \log \frac{k_{\max} L}{\delta}}\right)$$

for all $k = 1, \ldots, k_{\max}$ *simultaneously.*

Lemma A.12. *With probability at least* $1 - \frac{1}{N^2}$,

$$\max_{j\in[N]:w_j\neq 1} \|V_k^\top z_j\|_2 \leq 4\sigma\sqrt{\frac{\log(ndL)}{p}}$$

for all $k = 1, \ldots, k_{\max}$ *simultaneously.*

Applying the above lemmas, we obtain

$$\begin{aligned}
&\max_{j\in[N]:w_j\neq 1} \|V_k^\top y_j\|_2 \\
&\leq \left(\max \text{aff} \cdot \|V_k^\top D_1\|_F + \sqrt{\frac{d}{p-d}}\right) \cdot \left(\sqrt{k} + \sqrt{50\log\frac{k_{\max}L}{\delta}}\right) \\
&\quad \cdot \sqrt{\frac{13}{d}\log(nk_{\max}L)} + 4\sigma\sqrt{\frac{\log(ndL)}{p}} \qquad \text{(A.14)}
\end{aligned}$$

for all $k = 1, \ldots, k_{\max}$ with probability at least $1 - 2\delta - \frac{2}{(nk_{\max}L)^2}$.

Step 3: Proof of the lemma

Putting (A.12) and (A.14) together, we obtain that if

$$\begin{aligned}
&\frac{c_6}{8(1+\sigma)} - 4\sigma\sqrt{\frac{d\log(nd)}{p}} \\
&\geq \left(\max \text{aff} \cdot \|V_k^\top D_1\|_F + \sqrt{\frac{d}{p-d}}\right) \cdot \sqrt{50\log\frac{dL}{\delta}} \cdot \sqrt{13\log(nk_{\max}L)}
\end{aligned}$$

for all $k = 1, \ldots, K$, then (2.4) holds. Using Lemma A.9, the above condition is satisfied if

$$\max \text{aff} \leq \frac{c_7 - 3\sigma(1+\sigma)\sqrt{\frac{d\log n}{p}}}{\log(nL)} - 10(1+\sigma)\sqrt{\frac{d}{p}}$$

for some constant $c_7 > 0$. This completes the proof.

A.2 Proofs of Auxiliary Lemmas

A.2.1 Proof of Lemma A.1a

We use the following lemma.

Lemma A.13 (Chi-square upper-tail lower-bound)**.** *For any $k \in \mathbb{N}$ and any $\epsilon \geq 0$, we have*

$$\Pr\{\chi_k^2 \geq k(1+\epsilon)\} \geq \frac{1}{3k\epsilon + 6} \exp\left(-\frac{k\epsilon}{2}\right).$$

where χ_k^2 is the chi-square random variable with k degrees of freedom.

Then it follows that for $\epsilon \geq 0$,

$$\begin{aligned}
&\Pr\left\{z_{(n-m+1)}^2 < \frac{k}{d}(1+\epsilon)\right\} \\
&\overset{(a)}{\leq} \Pr\left\{\exists I \subset [n], |I| = n-m+1 : z_i^2 < \frac{k}{d}(1+\epsilon), \forall i \in I\right\} \\
&\overset{(b)}{\leq} \binom{n}{m-1} \cdot \Pr\left\{z_1^2 < \frac{k}{d}(1+\epsilon)\right\}^{n-m+1} \\
&\overset{(c)}{\leq} \left(\frac{ne}{m}\right)^m \cdot \left(1 - \frac{1}{3k\epsilon+6}\exp\left(-\frac{k\epsilon}{2}\right)\right)^{n-m+1} \\
&\overset{(d)}{\leq} \exp\left\{m\log\frac{ne}{m} - \frac{n-m+1}{3k\epsilon+6}\exp\left(-\frac{k\epsilon}{2}\right)\right\} \qquad \text{(A.15)}
\end{aligned}$$

where (a) follows from that the event $\left\{z_{(n-m+1)}^2 < \frac{k}{d}(1+\epsilon)\right\}$ implies that there is a size-$(n-m+1)$ subset of z_i's where every squared value is smaller than $\frac{k}{d}(1+\epsilon)$, (b) follows from the union bound and the independence between z_i's, (c) follows from the fact $\binom{n}{m} \leq \left(\frac{ne}{m}\right)^m$ and Lemma A.13, and (d) follows from the fact $1 + x \leq e^x, \forall x$.

Choose ϵ such that

$$\epsilon = \frac{2}{k}\log\left(\frac{n-m+1}{6m\left(\log\frac{ne}{m\delta}\right)^2}\right).$$

This ϵ is non-negative because of the assumption of the lemma. Then we obtain

$$
\begin{aligned}
(\text{A.15}) &\leq \exp\left\{ m\log\frac{ne}{m} - \frac{n-m+1}{6\log\left(\frac{n-m+1}{6m\left(\log\frac{ne}{m\delta}\right)^2}\right)+6} \cdot \frac{6m\left(\log\frac{ne}{m\delta}\right)^2}{n-m+1} \right\} \\
&= \exp\left\{ m\log\frac{ne}{m} - \frac{\log\frac{ne}{m\delta}}{\left(1+\log\left(\frac{1}{6}\cdot\frac{n-m+1}{m}\cdot(\log\frac{ne}{m\delta})^{-2}\right)\right)} \cdot m\log\frac{ne}{m\delta} \right\} \\
&\leq \exp\left\{ m\log\frac{ne}{m} - \frac{(1+\log\frac{n}{m})}{(1+\log\frac{n}{6m}-2\log\log\frac{ne}{m\delta})} m\log\frac{ne}{m\delta} \right\} \\
&\leq \exp\left\{ m\log\frac{ne}{m} - m\log\frac{ne}{m\delta} \right\} \\
&\leq \delta^m.
\end{aligned}
$$

This completes the proof.

A.2.2 Proof of Lemma A.13

For $k \geq 2$, it follows from [77, Proposition 3.1] that

$$
\begin{aligned}
&\Pr\{\chi_k^2 \geq k(1+\epsilon)\} \\
&\geq \frac{1-e^{-2}}{2}\frac{k(1+\epsilon)}{k\epsilon+2\sqrt{k}}\exp\left(-\frac{1}{2}(k\epsilon-(k-2)\log(1+\epsilon)+\log k)\right) \\
&\geq \frac{1}{3\sqrt{k}\epsilon+6}\exp\left(-\frac{k}{2}(\epsilon-\log(1+\epsilon))\right) \\
&\geq \frac{1}{3k\epsilon+6}\exp\left(-\frac{k\epsilon}{2}\right).
\end{aligned}
$$

For $k=1$, we can see numerically that the inequality holds.

A.2.3 Proof of Lemma A.1b

Let $A = U\Sigma V^\top$ be the singular value decomposition of A. For $x \sim \mathcal{N}(0, \frac{1}{d} I_{d\times d})$ and $\epsilon \geq 1$, we have

$$\begin{aligned}
\Pr\left\{\|Ax\|_2^2 > \frac{\|A\|_F^2}{d}(1+\epsilon)\right\} &= \Pr\left\{\|\Sigma x\|_2^2 > \frac{\|A\|_F^2}{d}(1+\epsilon)\right\} \\
&= \Pr\left\{\sum_{i=1}^{k} \sigma_i^2 x_i^2 > \sum_{i=1}^{k} \frac{\sigma_i^2}{d}(1+\epsilon)\right\} \\
&\overset{(a)}{\geq} \prod_{i=1}^{k} \Pr\left\{x_i^2 > \frac{1}{d}(1+\epsilon)\right\} \\
&\overset{(b)}{\geq} \exp(-2k\epsilon)
\end{aligned}$$

where (a) follows from the following event leads to the preceding event, and (b) follows by lower bounding the Gaussian tail for $\epsilon \geq 1$. Then we have

$$\begin{aligned}
&\Pr\left\{z_{(n-m+1)}^2 < \frac{\|A\|_F^2}{d}(1+\epsilon)\right\} \\
&\overset{(a)}{\leq} \Pr\left\{\exists I \subset [n], |I| = n-m+1 : z_i^2 < \frac{\|A\|_F^2}{d}(1+\epsilon), \forall i \in I\right\} \\
&\overset{(b)}{\leq} \binom{n}{m-1} \cdot \Pr\left\{z_1^2 < \frac{\|A\|_F^2}{d}(1+\epsilon)\right\}^{n-m+1} \\
&\overset{(c)}{\leq} \left(\frac{ne}{m}\right)^m \cdot \left(1 - \exp\left(-2k\epsilon\right)\right)^{n-m+1} \\
&\overset{(d)}{\leq} \exp\left\{m \log \frac{ne}{m} - (n-m+1)\exp\left(-2k\epsilon\right)\right\}
\end{aligned}$$

where (a) follows from the event $\left\{z_{(n-m+1)}^2 < \frac{\|A\|_F^2}{d}(1+\epsilon)\right\}$ implies that there is a size-$(n-m+1)$ subset where the squared values are all smaller than $\frac{\|A\|_F^2}{d}(1+\epsilon)$, (b) follows from the union bound and the independence between z_i's, (c) follows from previous inequality, and (d) follows from the fact $1+x \leq e^x, \forall x$.

Choosing

$$\epsilon = \frac{1}{2k} \log \left(\frac{n-m+1}{m \log \frac{ne}{m\delta}} \right),$$

we get

$$\Pr \left\{ z^2_{(n-m+1)} < \frac{\|A\|_F^2}{d}(1+\epsilon) \right\} \leq \delta^m$$

This ϵ is valid since we can have

$$\epsilon = \frac{1}{2k} \log \left(\frac{n-m+1}{m \log \frac{ne}{m\delta}} \right) \geq \frac{1}{2 \vee \lfloor \alpha \log d \rfloor} \log \left(\frac{n-m+1}{m \log \frac{ne}{m\delta}} \right) \geq 1$$

if $\frac{n}{m} \geq 9d^\alpha \log \frac{ne}{m\delta}$.

A.2.4 Proof of Lemma A.2a

Let $A = U\Sigma V^\top$ be the singular value decomposition of A. Then we have

$$\begin{aligned} E[\|AX\|_F^2] &= E[\|U\Sigma V^\top X\|_F^2] \\ &= E[\|\Sigma X\|_F^2] \\ &= \sum_{i=1}^{\min(p,d)} \sigma_i^2 \cdot \left(\sum_{j=1}^{k} E[X_{ij}^2] \right) \\ &= \sum_{i=1}^{\min(p,d)} \sigma_i^2 \cdot \frac{k}{d} = \frac{k}{d}\|A\|_F^2. \end{aligned}$$

where the second last equality follows from that X_{ij} is a coordinate of a uniformly random unit vector, and thus

$$E[X_{ij}^2] = \frac{1}{d}, \quad \forall i, j.$$

A.2.5 Proof of Lemma A.2b

Consider the Stiefel manifold $V_k(\mathbb{R}^d)$ equipped with the Euclidean metric. We see that X is drawn from $V_k(\mathbb{R}^d)$ with the normalized Harr probability measure. We have

$$\|AX\|_F - \|AY\|_F \leq \|AX - AY\|_F = \|A(X-Y)\|_F \leq \|A\|_2 \|X-Y\|_F$$

for any $X, Y \in \mathbb{R}^{d\times k}$. Since $\|A\|_2 \leq 1$, $\|AX\|_F$ is a 1-Lipschitz function of X. As stated in [102, p.27], we have

$$\Pr\{\|AX\|_F > m\|AX\|_F + t\} \leq e^{-\frac{(d-1)t^2}{8}},$$

where $m\|AX\|_F$ is the median of $\|AX\|_F$. Also, we have

$$\Pr\{|\|AX\|_F - m\|AX\|_F| > t\} \leq 2e^{-\frac{(d-1)t^2}{8}},$$

and then it follows that

$$\begin{aligned} |E\|AX\|_F - m\|AX\|_F| &\leq E\left[|\|AX\|_F - m\|AX\|_F|\right] \\ &\leq \int_0^\infty 2e^{-\frac{(d-1)t^2}{8}} dt \\ &= \sqrt{\frac{8\pi}{d-1}}. \end{aligned}$$

It follows from Jensen's inequality and Lemma A.1a that

$$E\|AX\|_F \leq \sqrt{E\|AX\|_F^2} = \sqrt{\frac{k}{d}}\|A\|_F$$

Putting the above inequalities together using the triangle inequality, we obtain the desired result.

A.2.6 Proof of Lemma A.10

Since $\|V_k^\top D_{w_j} x_j\|_2^2$ is a weighted Chi-square random variable, we use the following concentration bound.

Lemma A.14 ([101]). *Let $x_1, \ldots, x_k$ be drawn iid from $\mathcal{N}(0,1)$. Then*

$$\Pr\left\{\sum_{i=1}^k a_i x_i^2 \geq \sum_{i=1}^k a_i + 2\sqrt{\sum_{i=1}^k a_i^2 \cdot t} + 2a_{\max} t\right\} \leq \exp(-t).$$

Applying the above lemma to the union bound, we have

$$\Pr\Big\{\exists k \in [k_{\max}], j \in [N], w_j \neq 1 : \|V_k^\top D_{w_j} x_j\|_2^2 \geq \frac{\|V_k^\top D_{w_j}\|_F^2}{d} + \frac{2\|V_k^\top D_{w_j}\|_F^2 \cdot t}{d} + \frac{2\|V_k^\top D_{w_j}\|_2^2 \cdot t}{d}\Big\} \leq n k_{\max} L \cdot \exp(-t).$$

By replacing $t = 3\log(nk_{\max}L)$, we have that with probability at least $1 - \frac{1}{(nk_{\max}L)^2}$,

$$\max_{j \in [N]: w_j \neq 1} \|V_k^\top D_{w_j} x_j\|_2^2 \leq \frac{\max_{l \neq 1} \|V_k^\top D_l\|_F^2}{d} \cdot (1 + 12\log(nk_{\max}L))$$

for all $k = 1, \ldots, k_{\max}$.

A.2.7 Proof of Lemma A.11

Since $V_k(P_k P_k^\top + Q_k Q_k^\top) = V_k$, we have

$$\begin{aligned}
\|V_k^\top D_l\|_F &\leq \|V_k^\top P_k P_k^\top D_l\|_F + \|V_k^\top Q_k Q_k^\top D_l\|_F \\
&\leq \|V_k^\top P_k\|_F \cdot \|P_k^\top D_l\|_F + \|V_k^\top Q_k\|_2 \cdot \|Q_k^\top D_l\|_F \\
&\leq \|V_k^\top D_1\|_F \cdot \|P_k^\top D_l\|_F + \|Q_k^\top D_l\|_F,
\end{aligned}$$

where the first inequality follows from the triangle inequality, and the third inequality follows from that $\|V_k\|_2 = \|P_k\|_2 = \|Q_k\|_2 = 1$. Since P_k and Q_k is

drawn uniformly at random from $\mathcal{B}_k\mathcal{D}_1$ and $\mathcal{B}_k\mathcal{D}_1^\perp$, we use Lemma A.2b and the union bound to obtain that

$$\begin{aligned}\|P_k^\top D_l\|_F &\le \sqrt{\frac{k}{d}}\|D_1^\top D_l\|_F + \|D_1^\top D_l\|_2 \cdot \left(\sqrt{\frac{8\pi}{d-1}} + \sqrt{\frac{8}{d-1}\log\frac{k_{\max}L}{\delta}}\right) \\ &\le \|D_1^\top D_l\|_F \cdot \left(\sqrt{\frac{k}{d}} + \sqrt{\frac{50}{d}\log\frac{k_{\max}L}{\delta}}\right) \\ &\le \max \text{aff} \cdot \left(\sqrt{k} + \sqrt{50\log\frac{k_{\max}L}{\delta}}\right)\end{aligned}$$

for all $k = 1, \ldots, k_{\max}$ with probability at least $1 - \delta$. We also have

$$\begin{aligned}\|Q_k^\top D_l\|_F &\le \sqrt{\frac{k}{p-d}}\|D_l\|_F + \|D_l\|_2 \cdot \left(\sqrt{\frac{8\pi}{p-d-1}} + \sqrt{\frac{8}{p-d-1}\log\frac{k_{\max}L}{\delta}}\right) \\ &\le \frac{\sqrt{kd} + \sqrt{50\log\frac{k_{\max}L}{\delta}}}{\sqrt{p-d}}\end{aligned}$$

for all $k = 1, \ldots, k_{\max}$ with probability at least $1-\delta$. Putting the above bounds together, we obtain

$$\|V_k^\top D_l\|_F \le \left(\max \text{aff} \cdot \|V_k^\top D_1\|_F + \sqrt{\frac{d}{p-d}}\right) \cdot \left(\sqrt{k} + \sqrt{50\log\frac{k_{\max}L}{\delta}}\right)$$

A.2.8 Proof of Lemma A.12

Since each $\frac{p}{\sigma^2}\|V_k^\top z_j\|_2^2$ is a chi-square random variable with k degrees of freedom, it follows from Lemma A.14 and the union bound that

$$\begin{aligned}&\Pr\left\{\exists k \in [k_{\max}], \exists j \in [N], w_j \ne 1 : \frac{p}{\sigma^2}\|V_k^\top z_j\|_2^2 \ge k + 2\sqrt{kt} + 2t\right\} \\ &\le nk_{\max}L \cdot \exp(-t).\end{aligned}$$

By replacing $t = 3\log(nk_{\max}L)$, we have that with probability at least $1 - \frac{1}{(nk_{\max}L)^2}$,

$$\begin{aligned}\max_{j\in[N]:w_j\neq 1} \|V_k^\top z_j\|_2 &\leq \frac{\sigma}{\sqrt{p}} \cdot \sqrt{k + 2\sqrt{kt} + 2t} \\ &\leq \frac{\sigma}{\sqrt{p}} \cdot \sqrt{2k + 3t} \\ &\leq \frac{\sigma}{\sqrt{p}} \cdot \sqrt{2k_{\max} + 9\log(nk_{\max}L)} \\ &\leq \frac{4\sigma}{\sqrt{p}} \cdot \sqrt{\log(ndL)}\end{aligned}$$

for all $k = 1, \ldots, k_{\max}$.

A.2.9 Proof of Lemma A.9

We have

$$\|V_k^\top D_1\|_F \geq \|D_1^\top v_1\|_2 = \frac{\|D_1^\top y_1\|_2}{\|y_1\|_2} \geq \frac{\|x_1\|_2 - \|D_1^\top z_1\|_2}{\|x_1\|_2 + \|z_1\|_2}$$

where the first inequality is trivial, and the second inequality follows from the triangle inequality. Now we use concentration bounds of the norms of Gaussian vectors. It follows from [102, 101] that

$$\begin{aligned}&\Pr\left\{\|x_1\|_2 \leq 2^{-1}\right\} \leq e^{-d^2/16}, \quad \Pr\left\{\|x_1\|_2 \geq 2\right\} \leq e^{-d/2}, \\ &\Pr\left\{\|z_1\|_2 \geq 2\right\} \leq e^{-p/2}, \quad \Pr\left\{\sqrt{\frac{p}{\sigma^2 d}}\|D_1^\top z_1\|_2 \geq 2\right\} \leq e^{-d/2}.\end{aligned}$$

Hence, with probability $1 - 2e^{-d/2} - e^{-p/2} - e^{-d^2/16}$,

$$\|V_k^\top D_1\|_F \geq \frac{\|x_1\|_2 - \|D_1^\top z_1\|_2}{\|x_1\|_2 + \|z_1\|_2} \geq \frac{\frac{1}{2} - 2\sigma\sqrt{\frac{d}{p}}}{2 + 2\sigma} \geq \frac{1}{8(1+\sigma)}$$

where the last inequality follows from the assumption $p \geq 64\sigma^2 d$.

Appendix B

Technical Proofs for Chapter 3

B.1 Proof of Theorem 3.1

We write $L(X)$ for the function being optimized; i.e.,

$$L(X) = \sum_{(i,j,k)\in\Omega} \mathcal{L}(Y_{i,j,k}(X_{i,j} - X_{i,k})).$$

Note that for any fixed X, $\mathbb{P}_{X^*}L(X) = mR(X)$ (where $\mathbb{P}_{X^*}$ denotes the expectation taken with respect to future samples from $\mathbb{P}_{X^*}$, as distinct from $\mathbb{E}$ which denotes the expectation over the samples used to generate $\hat{X}$). Let K be the set of $d_1 \times d_2$ matrices with nuclear norm at most 1. The proof of Theorem 3.1 proceeds in three main steps.

1. By some algebraic of manipulations L, we reduce the problem to showing a uniform law of large numbers for the family of functions $\{L(X) : X \in \sqrt{\lambda d_1 d_2}K\}$.

2. Using symmetrization and duality properties of K, we reduce the problem to bounding the norm of a matrix M whose entries are sums of random signs.

3. We bound the norm of M using various concentration inequalities and a theorem of Seginer [139].

Since $\hat{X}$, by definition, minimizes $L(\hat{X})$, for any $\tilde{X} \in \sqrt{\lambda d_1 d_2} K$ we can bound

$$\begin{aligned}\mathbb{P}_{X^*}[L(\hat{X}) - L(\tilde{X})] &\leq \mathbb{P}_{X^*}[L(\hat{X})] - L(\hat{X}) - \left(\mathbb{P}_{X^*}[L(\tilde{X})] - L(\tilde{X})\right) \\ &\leq 2 \sup_{X \in \sqrt{\lambda d_1 d_2} k} |\mathbb{P}_{X^*} L(X) - L(X)|.\end{aligned}$$

In other words, it suffices to show a uniform law of large numbers for $\{L(X) : X \in \sqrt{\lambda d_1 d_2} K\}$.

Let $\epsilon_{i,j,k}$ be i.i.d. ± 1-valued variables and let $\xi_{i,j,k}$ be the indicator that $(i,j,k) \in \Omega$. By Giné-Zinn's symmetrization (as in [46]),

$$\begin{aligned}&\sup_{X \in \sqrt{\lambda d_1 d_2} K} |\mathbb{P}_{X^*} L(X) - L(X)| \\ &\leq 2\mathbb{E} \sup_{X \in \sqrt{\lambda d_1 d_2} K} \left| \sum_{i,j,k \in \Omega} \epsilon_{i,j,k} \mathcal{L}(Y_{i,j,k}(X_{i,j} - X_{i,k})) \right|.\end{aligned}$$

Since $\mathcal{L}$ is 1-Lipschitz, we obtain

$$\begin{aligned}\sup_{X \in \sqrt{\lambda d_1 d_2} K} |\mathbb{P}_{X^*}[L(X)] - L(X)| &\leq 2\mathbb{E} \sup_{X \in \sqrt{\lambda d_1 d_2} K} \left| \sum_{i,j,k \in \Omega} \epsilon_{i,j,k} Y_{i,j,k} (X_{i,j} - X_{i,k}) \right| \\ &= 2\mathbb{E} \sup_{X \in \sqrt{\lambda d_1 d_2} K} \left| \sum_{i,j,k} \xi_{i,j,k} \epsilon_{i,j,k} (X_{i,j} - X_{i,k}) \right|,\end{aligned}$$

where in the last line, we recognized that $\epsilon_{i,j,k} Y_{i,j,k}$ has the same distribution as $\epsilon_{i,j,k}$. Now, let M denote the matrix where $M_{ij} = \sum_k (\xi_{i,j,k} \epsilon_{i,j,k} - \xi_{i,k,j} \epsilon_{i,k,j})$. Then

$$\sum_{i,j,k} \xi_{i,j,k} \epsilon_{i,j,k} (X_{i,j} - X_{i,k}) = \operatorname{tr}(M^T X)$$

and so

$$\sup_{X \in \sqrt{\lambda d_1 d_2} K} \sum_{i,j,k} \xi_{i,j,k} \epsilon_{i,j,k} (X_{i,j} - X_{i,k}) = \sup_{X \in \sqrt{\lambda d_1 d_2} K} \operatorname{tr}(M^T X) = \sqrt{\lambda d_1 d_2} \|M\|.$$

Putting everything together, we have (for any $\tilde{X} \in \sqrt{\lambda d_1 d_2} K$)

$$\mathbb{E}\left[\mathbb{P}_{X^*}[L(\hat{X})] - \mathbb{P}_{X^*}[L(\tilde{X})]\right] \leq 4\sqrt{\lambda d_1 d_2}\mathbb{E}\|M\|.$$

Together with the following lemma (which we prove in Appendix B.2), this completes the proof of Theorem 3.1

Lemma B.1. *With* $p = \frac{m}{d_1 d_2}$,

$$\mathbb{E}\|M\| \leq C\kappa\sqrt{p(d_1+d_2)}\log(d_1 d_2).$$

B.2 Proof of Lemma B.1

We will decompose M into two parts, $M = M^{(1)} - M^{(2)}$, with

$$M^{(1)}_{ij} = \sum_{k \neq j} \xi_{i,j,k}\epsilon_{i,j,k}$$
$$M^{(2)}_{ij} = \sum_{k \neq j} \xi_{i,k,j}\epsilon_{i,k,j}.$$

Then $\|M\| \leq \|M^{(1)}\| + \|M^{(2)}\|$. Since $M^{(1)}$ and $M^{(2)}$ have the same distribution,

$$\mathbb{E}\|M\| \leq 2\mathbb{E}\|M^{(1)}\|,$$

and so we are reduced to studying $M^{(1)}$, which has i.i.d. entries. Now, we apply Seginer's theorem [139]:

$$\mathbb{E}\|M^{(1)}\| \leq C\left(\mathbb{E}\max_i \|M^{(1)}_{i*}\|_2 + \mathbb{E}\max_j \|M^{(1)}_{*j}\|_2\right), \tag{B.1}$$

where $M^{(1)}_{i*}$ denotes the ith row of $M^{(1)}$ and $M^{(1)}_{*j}$ denotes the jth column, and $\|\cdot\|_2$ denotes the Euclidean norm.

We will separate the task of bounding $\mathbb{E}\max_i \|M^{(1)}_{i*}\|_2$ into two parts: if $\|x\|_0$ denotes the number of non-zero coordinates in x and $\|x\|_\infty$ denotes

$\max_j |x_j|$ then $\|x\|_2 \leq \sqrt{\|x\|_0}\|x\|_\infty$; with the Cauchy-Schwarz inequality, this implies that

$$\left(\mathbb{E}\left[\max_i \|M_{i*}^{(1)}\|_2\right]\right)^2 \leq \mathbb{E}\left[\max_i \|M_{i*}^{(1)}\|_0\right] \mathbb{E}\left[\max_i \|M_{i*}^{(1)}\|_\infty^2\right] \tag{B.2}$$

First, we will show that every row of $M^{(1)}$ is sparse. Let $Z_{ij} = \sum_{k \neq j} \xi_{i,j,k}$ and let Y_{ij} be the indicator that $Z_{ij} > 0$. Recalling that $\mathbb{E}\xi_{i,j,k} = p_{i,j,k}$, we have (by Assumption 3.1) $\mathbb{E}Z_{ij} \leq \kappa p$. Since Z_{ij} takes non-negative integer values, we have $\Pr(Y_{ij} = 1) = \Pr(Z_{ij} > 0) \leq \kappa p$. By Bernstein's inequality, for any fixed i

$$\Pr(\|M_{i*}^{(1)}\|_0 \geq \kappa d_2 p + t) \leq \Pr(\sum_{j=1}^{d_2} Y_{ij} \geq \kappa d_2 p + t) \leq \exp\left(-\frac{t^2/2}{\kappa p d_2 + t/3}\right).$$

Integrating by parts, we have

$$\mathbb{E}\left[\|M_{i*}^{(1)}\|_0\right] \leq \kappa d_2 p + \int_{\kappa d_2 p}^{\infty} \Pr(\|M_{i*}^{(1)}\|_0 \geq t)\ dt \leq \kappa d_2 p + \frac{3}{8}.$$

Next, we will consider the size of the elements in $M^{(1)}$. First of all, $M_{ij}^{(1)} \leq Z_{ij}$ (this fairly crude bound will lose us a factor of $\sqrt{\log(d_1 d_2)}$). Now, Bernstein's inequality applied to Z_{ij} gives

$$\Pr(M_{ij}^{(1)} \geq \kappa p + t) \leq \Pr(Z_{ij} \geq \kappa p + t) \leq \exp\left(-\frac{t^2/2}{\kappa p + t/3}\right).$$

Taking a union bound over i and j, if $t \geq C\kappa \log(d_1 d_2)$ then

$$\Pr(\max_{ij} M_{ij}^{(1)} \geq t) \leq d_1 d_2 \exp(-ct) \leq \exp(-c't).$$

Integrating by parts,

$$\begin{aligned}\mathbb{E}\left[\max_{ij} M_{ij}^{(1)}\right] &\leq \kappa \log^2(d_1 d_2) + \int_{\kappa \log^2(d_1 d_2)}^{\infty} \Pr(\max_{ij} M_{ij}^{(1)} \geq \sqrt{t})\ dt \\ &\leq \kappa \log^2(d_1 d_2) + C.\end{aligned}$$

Going back to (B.2), we have shown that

$$\mathbb{E} \max_i \|M_{i*}^{(1)}\| \leq C\kappa\sqrt{pd_2}\log(d_1 d_2).$$

The same argument applies to $M_{*j}^{(1)}$ (but with $\sqrt{pd_1}$ instead of $\sqrt{pd_2}$), and so we conclude from (B.1) that

$$\mathbb{E}\|M^{(1)}\| \leq C\kappa\sqrt{p(d_1+d_2)}\log(d_1 d_2).$$

B.3 Proof of Theorem 3.2

B.3.1 A sketch of the proof

The proof of Theorem 3.2 uses Fano's inequality.

1. We construct matrices $X^1, \ldots, X^\ell$. These matrices all have small nuclear norm, and for every pair i, j the KL-divergence between the induced observation distributions is $\Theta(\log \ell)$. We construct these matrices randomly, using concentration inequalities and a union bound to show that we can take ℓ of the order $\sqrt{\lambda m(d_1+d_2)}$.

2. We apply Fano's inequality to show that if we generate data according to a randomly chosen X^i, then any algorithm has a reasonable chance to choose a different X^j (using the fact that the KL-divergence is $O(\log \ell)$). Since the KL-divergence is $\Omega(\log \ell)$, this implies that the algorithm incurs a substantial penalty whenever it makes a wrong choice.

In any application of Fano's inequality, the key is to construct a large number of admissible models that are close to one another in KL-divergence. Specifically, if we can construct distributions $\mathbb{P}_1, \ldots, \mathbb{P}_\ell$ with $D(\mathbb{P}_i \| \mathbb{P}_j) + 1 \leq \frac{1}{2}\log \ell$ for all i, j, then given a single sample from some $\mathbb{P}_i$, no algorithm can

accurately identify which $\mathbb{P}_i$ it came from. In order to apply this denote by $\mathbb{P}_{X,m}$ the distribution of the data when the true parameters are X. We will construct $X^1 \ldots, X^\ell \in \sqrt{\lambda d_1 d_2} K$ such that for all $i \neq j$,

$$D(\mathbb{P}_{X^i,m} \| \mathbb{P}_{X^j,m}) + 1 \leq \frac{1}{2} \log \ell, \tag{B.3}$$

$$R_j(X^i) \geq R_j(X^j) + c \frac{\log \ell}{m} \tag{B.4}$$

for some constant $c > 0$, where R_j denotes the expected risk when the true parameters are given by X^j. Given a single observation from some $\mathbb{P}_{X^j,m}$, (B.3) will imply (by Fano's inequality) that no algorithm can correctly identify which X^j was the true parameter. On the other hand, (B.4) will imply that if the algorithm makes a mistake – say it chooses X^i for $i \neq j$ – then its risk will be $c\frac{\log \ell}{m}$ larger than the best in the class. In particular, if we can prove (B.3) and (B.4) with $\log \ell \sim \sqrt{\lambda m (d_1 + d_2)}$ then it will imply Theorem 3.2.

We construct a set of matrices satisfying (B.3) and (B.4) using a probabilistic method. Supposing that $d_2 \geq d_1$, we choose a parameter $\gamma > 0$ and set B to be an integer that is approximately $\lambda \gamma^{-2}$. We define X^1 by filling its top $B \times d_2$ block with independent, uniform $\pm\gamma$ entries, and then copying that top block B/d_1 times to fill the matrix. Then let $X^2, \ldots, X^\ell$ be independent copies of X^1. First of all, each $X^i \in \sqrt{\lambda d_1 d_2} K$ because $\|X^i\|_* \leq \sqrt{\mathrm{rank}(X^i)} \|X^i\|_F \leq \sqrt{\lambda d_1 d_2}$.

Now, let us consider $D(\mathbb{P}_{X^1,m} \| \mathbb{P}_{X^2,m})$. For a single i, j, k triple, there is probability $1/4$ of having $X^1_{i,j} - X^1_{i,k}$ different from $X^2_{i,j} - X^2_{i,k}$, in which case they differ by 4γ. If γ is bounded above, each different entry contributes $\Theta(\alpha^2 \gamma^2)$ to the KL-divergence between $\mathbb{P}_{X^1,m}$ and $\mathbb{P}_{X^2,m}$. Since about m entries are observed in $\mathbb{P}_{X^1,m}$, we see that

$$D(\mathbb{P}_{X^1,m} \| \mathbb{P}_{X^2,m}) \asymp m\gamma^2. \tag{B.5}$$

On the other hand, $R_1(X^1)$ and $R_1(X^2)$ differ by $\Theta(\gamma^2)$, because for a constant fraction of triples i, j, k, the chance that $Y_{i,j,k}$ is 1 differs by $O(\gamma)$ in X^1 and X^2, and on the event that $Y_{i,j,k}$ differs in these two models the loss differs by another $O(\gamma)$ factor.

Applying standard concentration inequalities, we show that one can apply the union bound to $\ell = \exp(cBd_2)$ of these matrices. In view of (B.3) and (B.5), we need to take $Bd_2 = \frac{\lambda^2}{\gamma^2 d_1} \asymp m\gamma^2$. Eliminating γ, we end up with $\log \ell \asymp \sqrt{\lambda m/d_1}$ (which is within a constant factor of $\sqrt{\lambda m(d_1+d_2)}$ under our assumption that $d_2 \geq d_1$).

B.3.2 Some concentration lemmas

We begin by quoting some standard concentration results (see, e.g. [153]).

Definition B.2. *A random variable X is σ^2-subgaussian if $\mathbb{E}e^{\theta X} \leq e^{\theta^2\sigma^2/2}$ for all $\theta > 0$. A random variable X is L-subexponential if $\mathbb{E}e^{\theta X} \leq (1 - \theta^2 L^2)$ for $\theta < 1/L$.*

One can easily show that the product of two subgaussian variables is subexponential:

Lemma B.3. *If X is σ^2-subgaussian and Y is τ^2-subgaussian then XY is $C\sigma\tau$-subexponential for a universal constant C.*

Moreover, one has a Bernstein-type inequality for sums of independent subexponential variables.

Lemma B.4. *If $X_1, \ldots, X_k$ are i.i.d. L-subexponential then*

$$\Pr(\sum_i X_i \geq t) \leq \exp\left(-\frac{ct^2}{L^2 k + Lt}\right).$$

B.3.3 Construction of a packing set

Let $0 < \gamma < 1$ be some parameter to be determined such that $B := \lambda\gamma^{-2}$ is an integer.

Proposition B.5. *Suppose that $\mathcal{L}'(0) < 0$. For every sufficiently small γ (depending on $\mathcal{L}$), there exists a set $\mathcal{X} \subset \sqrt{\lambda d_1 d_2} K$ of $\exp(cBd_2)$ $d_1 \times d_2$ matrices such that for any two $X^1, X^2 \in \mathcal{X}$,*

$$\frac{1}{d_1 d_2^2} \sum_{i=1}^{d_1} \sum_{j,k=1}^{d_2} \mathbb{E}_{X^1}[\mathcal{L}(Y(X_{ij}^2 - X_{ik}^2)) - \mathcal{L}(Y(X_{ij}^1 - X_{ik}^1))] \geq c\gamma^2$$

and for any m,

$$\frac{1}{m} D(\mathbb{P}_{X^1,m} \| \mathbb{P}_{X^2,m}) \leq C\gamma^2,$$

where $0 < c < C$ are universal constants.

Following Davenport et al., we construct this set $\mathcal{X}$ randomly: let X be a random $B \times d_2$ matrix, where each element is chosen independently to be either γ or $-\gamma$.

Lemma B.6. *Let X^1 and X^2 be independent copies of X. Then with probability at least $1 - \exp(-cBd_2)$,*

$$\sum_{i=1}^{B} \sum_{j,k=1}^{d_2} (X_{ij}^1 - X_{ik}^1 - X_{ij}^2 + X_{ik}^2)^2 \geq 2\gamma^2 B d_2^2,$$

where $c > 0$ is a universal constant.

Before proving Lemma B.6, let us see how it implies Proposition B.5. First of all, for X a random $B \times d_2$ matrix as above, let $\tilde{X}$ be the $d_1 \times d_2$

matrix obtained by stacking $\lceil d_1/B \rceil$ copies of X, and filling out any remaining entries by zeros. Then, for random X and Y, with high probability

$$\sum_{i=1}^{d_1}\sum_{j,k=1}^{d_2}(\tilde{X}^1_{ij}-\tilde{X}^1_{ik}-\tilde{X}^2_{ij}+\tilde{X}^2_{ik})^2 = \lceil d_1/B\rceil \sum_{i=1}^{B}\sum_{j,k=1}^{d_2}(X^1_{ij}-X^1_{ik}-X^2_{ij}+X^2_{ik})^2 \\ \asymp \gamma^2 d_1 d_2^2, \tag{B.6}$$

where the lower bound for the last line came from Lemma B.6, and the upper bound just came from the observation that each term in the sum is bounded by $16\gamma^2$. Let $\mathfrak{X}$ be the set obtained by choosing $\exp(cBd_2/4)$ random copies of $\tilde{X}$ in this way. The high-probability estimate in Lemma B.6 implies that with high probability, *every* pair $\tilde{X}^1, \tilde{X}^2$ in $\mathfrak{X}$ satisfies (B.6). Now,

$$D(\mathbb{P}_{X^1,m}\|\mathbb{P}_{X^2,m}) = \mathbb{E}_\Omega\left[\sum_{(i,j,k)\in\Omega} D(f(X^1_{ij}-X^1_{ik})\|f(X^2_{ij}-X^2_{ik}))\right] \\ \asymp \frac{m}{d_1 d_2^2}\sum_{i,j,k}(X^1_{ij}-X^1_{ik}-X^2_{ij}+X^2_{ik})^2,$$

where $f(x) = e^x/(1+e^x)$ is the logistic function, and the last line follows from a Taylor expansion of $D(f(x)\|f(y))$ around $x=y$, because all the X^1_{ij} and X^2_{ij} are bounded by $\gamma < 1$. Together with (B.6), this proves the first inequality in Proposition B.5; the second inequality follows because each term of the form $D(f(X_{ij}-X_{ik})\|f(Y_{ij}-Y_{ik}))$ is bounded by a constant times γ^2. This proves the second inequality of Proposition B.5.

By Taylor expansion again, if γ is sufficiently small (depending on $\mathcal{L}$) then

$$\mathcal{L}(Y_{i,j,k}(X^2_{i,j}-X^2_{i,k})) - \mathcal{L}(Y_{i,j,k}(X^1_{i,j}-X^1_{i,k})) \asymp Y_{i,j,k}(X^1_{i,j}-X^1_{i,k}-X^2_{i,j}+X^2_{i,k}).$$

Now, if i,j,k is a triple for which $2\gamma = X^1_{i,j}-X^1_{i,k} > X^2_{i,j}-X^2_{i,k}$ (and under the event of Lemma B.6, there are at least cBd_2^2 such triples) then $\mathbb{E}_{X^1}[Y_{i,j,k}] \asymp \gamma$

and so

$$\mathbb{E}_{X^1}[\mathcal{L}(Y_{i,j,k}(X^2_{i,j} - X^2_{i,k})) - \mathcal{L}(Y_{i,j,k}(X^1_{i,j} - X^1_{i,k}))] \asymp \gamma^2.$$

The same holds when i, j, k is a triple for which $-2\gamma = X^1_{i,j} - X^1_{i,k} < X^2_{i,j} - X^2_{i,k}$. Finally, if i, j, k is a triple such that $X^1_{i,j} - X^1_{i,k} = X^2_{i,j} - X^2_{i,k}$ then the expectation is zero. Summing over all triples, we see that on the event that Lemma B.6 holds,

$$\frac{1}{Bd_2^2} \sum_{i,j,k} \mathbb{E}_{X^1}[\mathcal{L}(Y_{i,j,k}(X^2_{i,j} - X^2_{i,k})) - \mathcal{L}(Y_{i,j,k}(X^1_{i,j} - X^1_{i,k}))] \geq c\gamma^2.$$

After summing over all $\lceil d_1/B \rceil$ blocks, this proves the first inequality of Proposition B.5.

Proof of Lemma B.6. We expand the square:

$$\begin{aligned}
&\sum_{ijk} (X_{ij} - X_{ik} - Y_{ij} + Y_{ik})^2 \\
&= 2 \sum_{ijk} X_{ij}^2 + Y_{ij}^2 + 2X_{ij}Y_{ik} - X_{ij}X_{ik} - Y_{ij}Y_{ik} - 2X_{ij}Y_{ij} \\
&= 4\gamma^2 B d_2^2 + 2 \sum_{ijk} 2X_{ij}Y_{ik} - X_{ij}X_{ik} - Y_{ij}Y_{ik} - 2X_{ij}Y_{ij}. \qquad \text{(B.7)}
\end{aligned}$$

We may study each of the cross-terms separately: for the $X_{ij}Y_{ik}$ term, note that $\sum_j X_{ij}$ and $\sum_k Y_{ik}$ are both $\gamma^2 d_2$-subgaussian (by Hoeffding's inequality). Hence, $\sum_{jk} X_{ij}Y_{ik}$ is $C\gamma^2 d_2$-subexponential (by Lemma B.3) and so by Lemma B.4,

$$\Pr\left(\left| \sum_{ijk} X_{ij}Y_{ik} \right| \geq \frac{1}{8}\gamma^2 B d_2^2 \right) \leq 2\exp(-cBd_2).$$

The similar argument applies to the $X_{ij}X_{ik}$ term: $\sum_j X_{ij}$ is $\gamma^2 d_2$-subgaussian and so $\sum_{ijk} X_{ij}X_{ik} = \sum_i (\sum_j X_{ij})^2$ is $C\gamma^2 d_2$-subexponential; hence

$$\Pr\left(\left| \sum_{ijk} X_{ij}X_{ik} \right| \geq \frac{1}{8}\gamma^2 B d_2^2 \right) \leq 2\exp(-cBd_2).$$

Of course, the $Y_{ij}Y_{ik}$ term is identical. Finally, note that $\sum_{ijk} X_{ij}Y_{ij} = d_2 \sum_{ij} X_{ij}Y_{ij}$. Since the terms in this sum are i.i.d., we may apply Hoeffding's inequality to obtain

$$\Pr\left(\left|\sum_{ijk} X_{ij}Y_{ij}\right| \geq \frac{1}{8}\gamma^2 B d_2^2\right) = \Pr\left(\left|\sum_{ij} X_{ij}Y_{ij}\right| \geq \frac{1}{8}\gamma^2 B d_2\right) \leq 2\exp(-cB^2 d_2^2).$$

Putting everything together, we see that with high probability, the total of all the cross-terms in (B.7) is at most half of the first term. □

B.3.4 Completing the proof

Let C denote the constant from Proposition B.5. Assume that $d_1 \leq d_2$ and that m is large enough so

$$\sqrt{\frac{d_2}{m}} \leq 8C\sqrt{\lambda} \leq \sqrt{\frac{m}{d_2}}. \tag{B.8}$$

Note that under the assumptions $\lambda \geq 1$ and $m \geq d_1 + d_2$ from Theorem 3.2, the lower bound of (B.8) is satisfied. Moreover, if the upper bound of (B.8) is not satisfied then we may decrease λ until it is; the conclusion of Theorem 3.2 will not be affected because as long as (B.8) fails, the minimum in Theorem 3.2 will be 1.

By the lower bound in (B.8), there is an integer B such that

$$B \leq \sqrt{\frac{\lambda m}{d_2}} \leq 2B;$$

fix this B and define γ by

$$\gamma^2 = \lambda/B \asymp \sqrt{\frac{\lambda d_2}{m}}.$$

By the upper bound in (B.8), $\gamma \leq 1$.

Now, Fano's inequality states that if we first select a random $X \in \mathcal{X}$ and then draw a sample from $\mathbb{P}_{X,m}$, then any algorithm trying to identify X can succeed with probability at most

$$\frac{\min\{D(\mathbb{P}_{X,m} \| \mathbb{P}(Y,m)) : X, Y \in \mathcal{X}\} + 1}{\log |\mathcal{X}|} \leq \frac{2Cm\gamma^2}{Bd_2} \leq \frac{1}{2}.$$

Finally, note that by the first inequality in Proposition B.5, the error incurred by choosing the wrong $X \in \mathcal{X}$ is at least $c\gamma^2 \asymp \sqrt{\frac{\lambda d_2}{m}}$.

Now, we have so far only discussed the case $d_2 \geq d_1$. The case $d_1 \leq d_2$ is not exactly equivalent because our model is not symmetric in its treatment of users and items. However, the proof of Theorem 3.2 does not change very much. We take horizontally stacked blocks of size $d_1 \times B$ instead of $B \times d_2$. The main difference is in the calculation leading to (B.6): there are extra cross-terms appearing due to the fact that items in different blocks need to be compared with one another. However, all of these additional terms may be controlled with Lemmas B.3 and B.4 in much the same way as the existing terms are controlled.

B.4 Comparison to Stochastic Gradient Descent

Another practical algorithm to optimize (3) is Stochastic Gradient Descent (SGD). We have experimented SGD on the same datasets in Table 1. We ran the algorithm with the same regularization parameters and different step sizes. The statistical results for SGD were observed to be no better than AltSVM, and hence we did not present them in the main paper.

Let us first describe the SGD procedure. At each step, ones chooses a triple $(i, j, k) \in \Omega$ uniformly at random and run a SGD step, which can be

Datasets	N	NDCG@10
ML1m	20	0.6852
	50	0.7666
	100	0.7728
ML10m	20	0.6977
	50	0.7452
	100	0.7659

Table B.1: NDCG@10 of SGD on different datasets, for different numbers of observed ratings per user.

Precision@	SGD with $C = 5000$
1	0.1556
2	0.1498
5	0.1236
10	0.1031
100	0.0441

Table B.2: Precision@K for SGD of (3) on the binarized MovieLens1m dataset.

written as

$$
\begin{aligned}
u_i^+ &\leftarrow u_i - \eta \cdot \left\{ g \cdot (v_j - v_k) + \frac{\lambda}{|\Omega_i|} u_i \right\} \\
v_j^+ &\leftarrow v_j - \eta \cdot \left\{ g \cdot u_i + \frac{\lambda}{|\Omega^j|} v_j \right\} \\
v_j^+ &\leftarrow v_j - \eta \cdot \left\{ -g \cdot u_i + \frac{\lambda}{|\Omega^k|} v_k \right\}
\end{aligned}
$$

where $\Omega^{(j)}$ denotes the number of comparisons in Ω which involve item j. η is a step size and $g \in \partial\mathcal{L}(u_i^\top (v_j - v_k))$.

The following tables show the statistical result of SGD. The step size is chosen by $\eta = \frac{\alpha}{1+\beta t}$ as suggested in [179]. α and β were the powers of 10^{-1}, and the best result is reported. The results are comparable to AltSVM, but it did not achieve better results. We note that this is the best result from several different step sizes, while AltSVM does not have any other parameter to choose except for the regularization parameter.

Appendix C

Technical Proofs for Chapter 4

C.1 Proof of Lemma 4.6

Proof of (4.18)⇒(4.17) Let $R_t^\star$ be the $r \times r$ orthogonal matrix such that $\mathrm{dist}(U_t, V_t; X_r^\star) = \|W_t - W^\star R_t^\star\|_F$. By the triangle inequality, we have

$$\begin{aligned}\|W_t\|_2 = \|W_t - W^\star R_t^\star + W^\star R_t^\star\|_2 &\overset{(i)}{\leq} \|W^\star R_t^\star\|_2 + \|W_t - W^\star R_t^\star\|_2 \\ &\overset{(ii)}{\leq} \|W^\star\|_2 + \tfrac{\sqrt{2}\sigma_r(X_r^\star)^{1/2}}{10} \\ &\overset{(iii)}{\leq} \|W^\star\|_2 + \tfrac{\sigma_r(W^\star)}{10} \\ &\leq \tfrac{11}{10} \cdot \|W^\star\|_2 \end{aligned} \tag{C.1}$$

where (i) is due to triangle inequality, (ii) is due to Assumption A.1 (iii) is due to the fact that $\sqrt{2} \cdot \sigma_r(X_r^\star)^{1/2} = \sigma_r(W^\star)$ and $\kappa \geq 1$. The above bound holds for every $t = 0, 1, \ldots$.

On the other hand, we have:

$$\begin{aligned}\|W_0\|_2 = \|W_0 - W^\star R_t^\star + W^\star R^\star\|_2 &\overset{(i)}{\geq} \|W^\star R_t^\star\|_2 - \|W_0 - W^\star R_t^\star\|_2 \\ &\geq \|W^\star\|_2 - \tfrac{\sqrt{2}\sigma_r(X_r^\star)^{1/2}}{10} \\ &\geq \|W^\star\|_2 - \tfrac{\sigma_1(W^\star)}{10} \\ &\geq \tfrac{9}{10} \cdot \|W^\star\|_2 \end{aligned} \tag{C.2}$$

Combining (C.1) and (C.2), we obtain:

$$\|W_t\|_2 \leq \tfrac{11}{10} \cdot \|W^\star\|_2 \leq \tfrac{11}{9}\|W_0\|_2 \Longrightarrow \tfrac{81}{121} \cdot \|W_t\|_2^2 \leq \|W_0\|_2^2$$

and, finally,

$$\frac{1}{8\cdot\max\{L,\ L_g\}\cdot\|W_t\|_2^2} = \frac{1}{8\cdot\frac{121}{81}\cdot\max\{L,\ L_g\}\cdot\frac{81}{121}\cdot\|W_t\|_2^2} \geq \frac{1}{12\cdot\max\{L,\ L_g\}\cdot\|W_0\|_2^2}$$

Proof of (4.19)⇒(4.15) We have

$$\begin{aligned}
&\|\nabla f(U_tV_t^\top)\|_2 \\
&\leq \|\nabla f(U_0V_0^\top)\|_2 + \|\nabla f(U_tV_t^\top) - \nabla f(U_0V_0^\top)\|_2 \\
&\overset{(i)}{\leq} \|\nabla f(U_0V_0^\top)\|_2 + L\|U_tV_t^\top - U_0V_0^\top\|_F \\
&\overset{(ii)}{\leq} \|\nabla f(U_0V_0^\top)\|_2 + L\|U_tV_t^\top - U^\star V^{\star\top}\|_F + L\|U_0V_0^\top - U^\star V^{\star\top}\|_F \qquad \text{(C.3)}
\end{aligned}$$

where (i) is due to the fact that f is L-smooth and, (ii) holds by adding and subtracting $U^\star V^{\star\top}$ and then applying triangle inequality. To bound the last two terms on the right hand side, we observe:

$$\begin{aligned}
\|U_tV_t^\top - U^\star V^{\star\top}\|_F &= \|U_tV_t^\top - U^\star RV_t^\top + U^\star RV_t^\top - U^\star RR^\top V^{\star\top}\|_F \\
&\overset{(i)}{\leq} \|U^\star R\|_2 \cdot \|V_t - V^\star R\|_F + \|V_t\|_2 \cdot \|U_t - U^\star R\|_F \\
&\leq (\|U^\star\|_2 + \|V_t\|_2) \cdot \mathrm{dist}(U_t, V_t; X_r^\star) \\
&\overset{(ii)}{\leq} \frac{21}{10} \cdot \|W^\star\|_2 \cdot \frac{\sigma_r(W^\star)}{10} \\
&\leq \frac{7}{10} \cdot \|W_0\|_2^2
\end{aligned}$$

where (i) is due to the triangle and Cauchy-Schwartz inequalities, (ii) is by Assumption A1 and (C.1). Similarly, one can show that $\|U_0V_0^\top - U^\star V^{\star\top}\|_F \leq \frac{7}{10} \cdot \|W_0\|_2^2$. Thus, (C.3) becomes:

$$\|\nabla f(U_0V_0^\top)\|_2 \geq \|\nabla f(U_tV_t^\top)\|_2 - \frac{3L}{2}\|W_0\|_2^2 \qquad \text{(C.4)}$$

Applying (C.1), (C.2), and the above bound, we obtain the desired result.

C.2 Proof of Linear Convergence (Theorem 4.7)

For clarity, we omit the subscript t, and use $(U,\ V)$ to denote the current estimate and (U^+, V^+) the next estimate. Further, we abuse the notation by denoting $\nabla g \triangleq \nabla g(U^\top U - V^\top V)$, where the gradient is taken over both U and V. We denote the stacked matrices of (U, V) and their variants as follows:

$$W = \begin{bmatrix} U \\ V \end{bmatrix},\ W^+ = \begin{bmatrix} U^+ \\ V^+ \end{bmatrix},\ W^\star = \begin{bmatrix} U^\star \\ V^\star \end{bmatrix}.$$

Observe that $W, W^+, W^\star \in \mathbb{R}^{(m+n)\times r}$. Then, the main recursion of `BFGD` in Algorithm 6 can be succinctly written as

$$W^+ = W - \widehat{\eta}\nabla_W(f + \tfrac{1}{2}g),$$

where

$$\nabla_W(f + \tfrac{1}{2}g) = \begin{bmatrix} \nabla_U f(UV^\top) + \frac{1}{2}\nabla_U g \\ \nabla_V f(UV^\top) + \frac{1}{2}\nabla_V g \end{bmatrix} = \begin{bmatrix} \nabla f(UV^\top)V + \frac{1}{2}U\nabla g \\ \nabla f(UV^\top)^\top U - \frac{1}{2}V\nabla g \end{bmatrix}.$$

In the above formulations, we use as regularizer of g function $\lambda = \frac{1}{2}$.

Our discussion below is based on the Assumption A.1, where:

$$\mathrm{dist}(U, V; X_r^\star) \leq \frac{\sqrt{2}\cdot\sigma_r(X_r^\star)^{1/2}}{10\sqrt{\kappa}} = \frac{\sigma_r(W^\star)}{10\sqrt{\kappa}}, \tag{C.5}$$

holds for the current iterate. The last equality is due to the fact that $\sigma_r(W^\star) = \sqrt{2}\cdot\sigma_r(X_r^\star)^{1/2}$, for $(U^\star, V^\star)$ with "equal footing". For the initial point (U_0, V_0), (C.5) holds by the assumption of the theorem. Since the right hand side is fixed, (C.5) holds for every iterate, as long as $\mathrm{dist}(U, V; X_r^\star)$ decreases.

To show this, let $R \in O_r$ be the minimizing orthogonal matrix such that $\mathrm{dist}(U, V; X_r^\star) = \|W - W^\star R\|_F$; here, O_r denotes the set of $r \times r$ orthogonal

matrices such that $R^\top R = I$. Then, the decrease in distance can be lower bounded by

$$\begin{aligned}
&\operatorname{dist}(U, V; X_r^\star)^2 - \operatorname{dist}(U^+, V^+; X_r^\star)^2 \\
&= \|W - W^\star R\|_F^2 - \min_{Q \in O_r} \left\|W^+ - W^\star Q\right\|_F^2 \\
&\geq \|W - W^\star R\|_F^2 - \left\|W^+ - W^\star R\right\|_F^2 \\
&= 2\widehat{\eta} \cdot \left\langle \nabla_W (f + \tfrac{1}{2} g), W - W^\star R \right\rangle - \widehat{\eta}^2 \cdot \left\|\nabla_W (f + \tfrac{1}{2} g)\right\|_F^2 \qquad \text{(C.6)}
\end{aligned}$$

where the last equality is obtaining by substituting W^+, according to its definition above. To bound the first term on the right hand side, we use the following lemma; the proof is provided in Section C.2.1.

Lemma C.1 (Descent lemma)**.** *Suppose* (C.5) *holds for* W*. Let* $\mu_{\min} = \min\{\mu,\ \mu_g\}$ *and* $L_{\max} = \max\{L,\ L_g\}$ *for* (μ, L) *and* (μ_g, L_g) *the strong convexity and smoothness parameters pairs for* f *and* g*, respectively. Then, the following inequality holds:*

$$\begin{aligned}
&\left\langle \nabla_W (f + \tfrac{1}{2} g), W - W^\star R \right\rangle \\
&\geq \tfrac{\mu_{\min} \cdot \sigma_r(W^\star)^2}{20} \|W - W^\star R\|_F^2 + \tfrac{1}{4L_{\max}} \left\|\nabla f(UV^\top)\right\|_F^2 \\
&\qquad + \tfrac{1}{16 L_{\max}} \|\nabla g\|_F^2 - \tfrac{L}{2} \|X^\star - X_r^\star\|_F^2 \qquad \text{(C.7)}
\end{aligned}$$

For the second term on the right hand side of (C.6), we obtain the

following upper bound:

$$
\begin{aligned}
&\left\|\nabla_W(f+\tfrac{1}{2}g)\right\|_F^2 \\
&= \left\|\begin{bmatrix}\nabla f(UV^\top)V + \frac{1}{2}U\nabla g \\ \nabla f(UV^\top)^\top U - \frac{1}{2}V\nabla g\end{bmatrix}\right\|_F^2 \\
&= \left\|\nabla f(UV^\top)V + \tfrac{1}{2}U\nabla g\right\|_F^2 + \left\|\nabla f(UV^\top)^\top U - \tfrac{1}{2}V\nabla g\right\|_F^2 \\
&\overset{(a)}{\le} 2\left\|\nabla f(UV^\top)V\right\|_F^2 + \tfrac{1}{2}\left\|U\nabla g\right\|_F^2 + 2\left\|\nabla f(UV^\top)^\top U\right\|_F^2 + \tfrac{1}{2}\left\|V\nabla g\right\|_F^2 \\
&= 2\left\|\nabla f(UV^\top)V\right\|_F^2 + 2\left\|\nabla f(UV^\top)^\top U\right\|_F^2 + \tfrac{1}{2}\left\|W\nabla g\right\|_F^2 \\
&\overset{(b)}{\le} 2\left\|\nabla f(UV^\top)\right\|_F^2 \cdot \left(\|U\|_2^2 + \|V\|_2^2\right) + \tfrac{1}{2}\|W\|_2^2\left\|\nabla g\right\|_F^2 \\
&\overset{(c)}{\le} \left(4\left\|\nabla f(UV^\top)\right\|_F^2 + \tfrac{1}{2}\left\|\nabla g\right\|_F^2\right)\cdot\|W\|_2^2\,, \qquad \text{(C.8)}
\end{aligned}
$$

where (a) follows from the fact $\|A+B\|_F^2 \le 2\|A\|_F^2 + 2\|B\|_F^2$, (b) is due to the fact $\|AB\|_F \le \|A\|_F \cdot \|B\|_2$, and (c) follows from the observation that $\|U\|_2, \|V\|_2 \le \|W\|_2$.

Plugging (C.7) and (C.8) in (C.6), we get

$$
\begin{aligned}
&\operatorname{dist}(U,V;X_r^\star)^2 - \operatorname{dist}(U^+,V^+;X_r^\star)^2 \\
&\ge 2\widehat{\eta}\cdot\left\langle\nabla_W(f+\tfrac{1}{2}g), W - W^\star R\right\rangle - \widehat{\eta}^2\cdot\left\|\nabla_W(f+\tfrac{1}{2}g)\right\|_F^2 \\
&\ge \tfrac{\widehat{\eta}\cdot\mu_{\min}\cdot\sigma_r(W^\star)^2}{10}\operatorname{dist}(U,V;X_r^\star)^2 - \widehat{\eta}L\left\|X^\star - X_r^\star\right\|_F^2 \\
&= \tfrac{\widehat{\eta}\cdot\mu_{\min}\cdot\sigma_r(X_r^\star)}{5}\operatorname{dist}(U,V;X_r^\star)^2 - \widehat{\eta}L\left\|X^\star - X_r^\star\right\|_F^2
\end{aligned}
$$

where we use the fact that $\sigma_r(W^\star) = \sqrt{2}\cdot\sigma_r(X_r^\star)^{1/2}$.

The above lead to the following recursion:

$$
\operatorname{dist}(U^+,V^+;X_r^\star)^2 \le \gamma_t\cdot\operatorname{dist}(U,V;X_r^\star)^2 + \widehat{\eta}L\left\|X^\star - X_r^\star\right\|_F^2,
$$

where $\gamma_t = 1 - \frac{\widehat{\eta}\cdot\mu_{\min}\cdot\sigma_r(X_r^\star)}{5}$. By the definition of $\widehat{\eta}$ in (4.18), we further have:

$$\begin{aligned}\gamma_t &= 1 - \tfrac{\mu_{\min}\cdot\sigma_r(X_r^\star)}{40\cdot L_{\max}\cdot\|W\|_2^2} \\ &\overset{(i)}{\geq} 1 - \tfrac{\mu_{\min}\cdot\sigma_r(X_r^\star)}{40\cdot L_{\max}\cdot\frac{100}{81}\cdot\|W^\star\|_2^2} \\ &\overset{(ii)}{\geq} 1 - \tfrac{\mu_{\min}}{17\cdot L_{\max}}\cdot\tfrac{\sigma_r(X_r^\star)}{\sigma_1(X_r^\star)}\end{aligned}$$

where (i) is by using (C.2) that connects $\|W\|_2$ with $\|W^\star\|_2$ as $\|W\|_2 \geq \frac{9}{10}\|W^\star\|_2$, and (ii) is due to the fact $\|W^\star\|_2 = \sqrt{2}\cdot\sigma_1(X_r^\star)^{1/2}$.

C.2.1 Proof of Lemma C.1

Before we step into the proof, we require a few more notations for simpler presentation of our ideas. We use another set of stacked matrices $Y = \begin{bmatrix} U \\ -V \end{bmatrix}$, $Y^\star = \begin{bmatrix} U^\star \\ -V^\star \end{bmatrix}$. The error of the current estimate from the closest optimal point is denoted by the following $\Delta_\times$ matrix structures:

$$\Delta_U = U - U^\star R, \quad \Delta_V = V - V^\star R, \quad \Delta_W = W - W^\star R, \quad \Delta_Y = Y - Y^\star R.$$

For our proof, we can write

$$\begin{aligned}&\big\langle \nabla_W(f + \tfrac{1}{2}g), W - W^\star R\big\rangle \\ &= \underbrace{\Big\langle \nabla f(UV^\top)V, U - U^\star R\Big\rangle + \Big\langle \nabla f(UV^\top)^\top U, V - V^\star R\Big\rangle}_{(A)} \\ &\qquad + \tfrac{1}{2}\cdot\left(\underbrace{\langle U\nabla g, U - U^\star R\rangle - \langle V\nabla g, V - V^\star R\rangle}_{(B)}\right)\end{aligned}$$

For (A), we have

$$\begin{aligned}(A) &= \left\langle \nabla f(UV^\top)V, U - U^\star R \right\rangle + \left\langle \nabla f(UV^\top)^\top U, V - V^\star R \right\rangle \\ &= \left\langle \nabla f(UV^\top), UV^\top - U^\star V^{\star\top} \right\rangle + \left\langle \nabla f(UV^\top), \Delta_U \Delta_V^\top \right\rangle \qquad \text{(C.9)} \\ &\geq \frac{\mu}{2} \underbrace{\left\| UV^\top - U^\star V^{\star\top} \right\|_F^2}_{(A1)} + \frac{1}{2L} \underbrace{\left\| \nabla f(UV^\top) \right\|_F^2}_{(A2)} \\ &\qquad - \frac{L}{2} \underbrace{\|X^\star - X_r^\star\|_F^2}_{(A3)} - \underbrace{\left\| \nabla f(UV^\top) \right\|_2 \cdot \|\Delta_W\|_F^2}_{(A4)}\end{aligned}$$

where, for the second term in (C.9), we use the fact that $\left\langle \nabla f(UV^\top), \Delta_U \Delta_V^\top \right\rangle \geq -\left|\left\langle \nabla f(UV^\top), \Delta_U \Delta_V^\top \right\rangle\right| = -\left|\left\langle \nabla f(UV^\top)\Delta_V, \Delta_U \right\rangle\right|$, the Cauchy-Schwarz inequality and the fact that $\|\Delta_U\|_F, \|\Delta_V\|_F \leq \|\Delta_W\|_F$; the first term in (C.9) follows from:

$$\begin{aligned}&\left\langle \nabla f(UV^\top), UV^\top - U^\star V^{\star\top} \right\rangle \\ &\overset{(i)}{\geq} f(UV^\top) - f(U^\star V^{\star\top}) + \frac{\mu}{2} \left\| UV^\top - U^\star V^{\star\top} \right\|_F^2 \\ &\overset{(ii)}{=} (f(UV^\top) - f(X^\star)) - (f(U^\star V^{\star\top}) - f(X^\star)) + \frac{\mu}{2} \left\| UV^\top - U^\star V^{\star\top} \right\|_F^2 \\ &\overset{(iii)}{\geq} \frac{1}{2L} \left\| \nabla f(UV^\top) \right\|_F^2 - \frac{L}{2} \left\| X^\star - U^\star V^{\star\top} \right\|_F^2 + \frac{\mu}{2} \left\| UV^\top - U^\star V^{\star\top} \right\|_F^2.\end{aligned}$$

where (i) is due to the μ-strong convexity of f, (ii) is by adding and subtracting $f(X^\star)$; observe that $f(X^\star) = f(U^\star V^{\star\top})$ if and only if $\text{rank}(X^\star) = r$, and (iii) is due to the L-smoothness of f and the fact that $\nabla f(X^\star) = 0$ (for the middle term), and due to the inequality [125, eq. (2.1.7)] (for the first term):

$$f(X) + \langle \nabla f(X), Y - X \rangle + \tfrac{1}{2L} \cdot \|\nabla f(X) - \nabla f(Y)\|_F^2 \leq f(Y). \qquad \text{(C.10)}$$

For (B), we have

$$
\begin{aligned}
&(B)\\
&= \langle Y\nabla g, W - W^\star R\rangle = \left\langle \nabla g, Y^\top W - Y^\top W^\star R\right\rangle\\
&= \tfrac{1}{2}\left\langle \nabla g, Y^\top W - R^\top Y^{\star\top} W^\star R\right\rangle + \tfrac{1}{2}\left\langle \nabla g, Y^\top W - 2Y^\top W^\star R + R^\top Y^{\star\top} W^\star R\right\rangle\\
&\overset{(a)}{=} \tfrac{1}{2}\left\langle \nabla g, Y^\top W\right\rangle + \tfrac{1}{2}\left\langle \nabla g, Y^\top W - Y^\top W^\star R - R^\top Y^{\star\top} W + R^\top Y^{\star\top} W^\star R\right\rangle\\
&= \frac{1}{2}\left\langle \nabla g, U^\top U - V^\top V\right\rangle + \frac{1}{2}\left\langle \nabla g, \Delta_Y^\top \Delta_W\right\rangle \qquad \text{(C.11)}\\
&\overset{(b)}{\geq} \frac{\mu_g}{4}\underbrace{\left\|U^\top U - V^\top V\right\|_F^2}_{(B1)} + \frac{1}{4L_g}\underbrace{\|\nabla g\|_F^2}_{(B2)} - \frac{1}{2}\underbrace{\|\nabla g\|_2 \cdot \|\Delta_W\|_F \cdot \|\Delta_Y\|_F}_{(B3)}
\end{aligned}
$$

where (a) follows from the "balance" assumption in $\mathcal{X}_r^\star$:

$$Y^{\star\top} W^\star = U^{\star\top} U^\star - V^{\star\top} V^\star = 0,$$

for the first term, and the fact that ∇g is symmetric, and therefore

$$\left\langle \nabla g, Y^\top W^\star R\right\rangle = \left\langle \nabla g, R^\top W^{\star\top} Y\right\rangle = \left\langle \nabla g, R^\top Y^{\star\top} W\right\rangle,$$

for the second term; (b) follows from the fact

$$\left\langle \nabla g, \Delta_Y^\top \Delta_W\right\rangle \geq -\left|\left\langle \nabla g, \Delta_Y^\top \Delta_W\right\rangle\right| = -\left|\langle \Delta_Y \nabla g, \Delta_W\rangle\right|$$

and the Cauchy-Schwarz inequality on the second term in (C.11), and

$$
\begin{aligned}
&\left\langle \nabla g, U^\top U - V^\top V\right\rangle\\
&\overset{(i)}{\geq} g(U^\top U - V^\top V) - g(0) + \tfrac{\mu_g}{2}\left\|U^\top U - V^\top V\right\|_F^2\\
&\overset{(ii)}{\geq} \left\langle \nabla g(0), U^\top U - V^\top V\right\rangle + \tfrac{1}{2L_g}\left\|\nabla g - \nabla g(0)\right\|_F^2 + \tfrac{\mu_g}{2}\left\|U^\top U - V^\top V\right\|_F^2\\
&\overset{(iii)}{=} \tfrac{1}{2L_g}\left\|\nabla g\right\|_F^2 + \tfrac{\mu_g}{2}\left\|U^\top U - V^\top V\right\|_F^2
\end{aligned}
$$

where (i) follow from the strong convexity, (ii) is due to (C.10), and (iii) is by construction of g where $\nabla g(0) = 0$. Furthermore, (B1) can be bounded below as follows:

$$\begin{aligned}
(B1) &= \left\|U^\top U - V^\top V\right\|_F^2 = \left\|U^\top U\right\|_F^2 + \left\|V^\top V\right\|_F^2 - 2\left\langle U^\top U, V^\top V\right\rangle \\
&= \left\|UU^\top\right\|_F^2 + \left\|VV^\top\right\|_F^2 - 2\left\langle UV^\top, UV^\top\right\rangle \\
&= \left\langle WW^\top, YY^\top\right\rangle \\
&= \left\langle WW^\top - W^\star W^{\star\top}, YY^\top - Y^\star Y^{\star\top}\right\rangle + \left\langle W^\star W^{\star\top}, YY^\top\right\rangle \\
&\quad + \left\langle WW^\top - W^\star W^{\star\top}, Y^\star Y^{\star\top}\right\rangle \\
&\overset{(i)}{=} \left\langle WW^\top - W^\star W^{\star\top}, YY^\top - Y^\star Y^{\star\top}\right\rangle + \left\langle W^\star W^{\star\top}, YY^\top\right\rangle \\
&\quad + \left\langle WW^\top, Y^\star Y^{\star\top}\right\rangle \\
&\geq \left\langle WW^\top - W^\star W^{\star\top}, YY^\top - Y^\star Y^{\star\top}\right\rangle \\
&= \left\|UU^\top - U^\star U^{\star\top}\right\|_F^2 + \left\|VV^\top - V^\star V^{\star\top}\right\|_F^2 - 2\left\|UV^\top - U^\star V^{\star\top}\right\|_F^2
\end{aligned}$$

where (i) is due to the fact that

$$\left\langle W^\star W^{\star\top}, Y^\star Y^{\star\top}\right\rangle = \left\|Y^{\star\top} W^\star\right\|_F^2 = \left\|U^{\star\top} U^\star - V^{\star\top} V^\star\right\|_F^2 = 0$$

and the first inequality holds by the fact that the inner product of two PSD matrices is non-negative.

At this point, we have all the required components to compute the desired lower bound. Combining (A1) and (B1), we get

$$\begin{aligned}
&4(A1) + (B1) \\
&= \left\|UU^\top - U^\star U^{\star\top}\right\|_F^2 + \left\|VV^\top - V^\star V^{\star\top}\right\|_F^2 + 2\left\|UV^\top - U^\star V^{\star\top}\right\|_F^2 \\
&= \left\|WW^\top - W^\star W^{\star\top}\right\|_F^2 \geq \frac{4\sigma_r(W^\star)^2}{5}\left\|\Delta_W\right\|_F^2,
\end{aligned}$$

where, in order to obtain the last inequality, we borrow the following Lemma by [152]:

Lemma C.2. *For any $W, W^\star \in R^{(m+n)\times r}$, with $\Delta_W = W - W^\star R$ for some orthogonal matrix $R \in \mathbb{R}^{r\times r}$, we have:*

$$\left\|WW^\top - W^\star W^{\star\top}\right\|_F^2 \geq 2\cdot\left(\sqrt{2}-1\right)\cdot\sigma_r(W^\star)^2\cdot\|\Delta_W\|_F^2$$

For convenience, we further lower bound the right hand side of this lemma by: $2\cdot\left(\sqrt{2}-1\right)\cdot\sigma_r(W^\star)^2\cdot\|\Delta_W\|_F^2 \geq \frac{4\sigma_r(W^\star)^2}{5}\|\Delta_W\|_F^2$.

Given the definitions of $\mu_{\min}$ and $L_{\max}$, we have:

$$\begin{aligned}
&(A)+\tfrac{1}{2}(B)\\
&\geq \tfrac{\mu}{2}(A1)+\tfrac{1}{2L}(A2)-\tfrac{L}{2}(A3)-(A4)+\tfrac{\mu_g}{8}(B1)+\tfrac{1}{8L_g}(B2)-\tfrac{1}{4}(B3)\\
&\overset{(i)}{\geq} \tfrac{\mu_{\min}}{8}\left(4(A1)+(B1)\right)+\tfrac{1}{2L_{\max}}(A2)+\tfrac{1}{8L_{\max}}(B2)-(A4)-\tfrac{1}{4}(B3)-\tfrac{L}{2}(A3)\\
&\geq \tfrac{\mu_{\min}\cdot\sigma_r(W^\star)^2}{10}\|\Delta_W\|_F^2+\tfrac{1}{2L_{\max}}\left\|\nabla f(UV^\top)\right\|_F^2+\tfrac{1}{8L_{\max}}\|\nabla g\|_F^2\\
&\qquad\qquad -\left\|\nabla f(UV^\top)\right\|_2\|\Delta_W\|_F^2-\tfrac{1}{4}\|\nabla g\|_F\|\Delta_W\|_F\|\Delta_Y\|_F\\
&\qquad\qquad -\tfrac{L}{2}\|X^\star-X_r^\star\|_F^2 \qquad\qquad (C.12)
\end{aligned}$$

where in (i) we used the definitions of $\mu_{\min}$ and $L_{\max}$. Note that we have not used the condition (C.5). It follows from (C.5) that

$$\begin{aligned}
\left\|\nabla f(UV^\top)\right\|_2\cdot\|\Delta_W\|_F^2 &\leq \tfrac{\sigma_r(W^\star)}{10\sqrt{\kappa}}\left\|\nabla f(UV^\top)\right\|_2\cdot\|\Delta_W\|_F\\
&\leq \tfrac{\mu_{\min}\cdot\sigma_r(W^\star)^2}{25}\|\Delta_W\|_F^2+\tfrac{1}{4L_{\max}}\left\|\nabla f(UV^\top)\right\|_2^2 \qquad (C.13)
\end{aligned}$$

and

$$\begin{aligned}
\tfrac{1}{4}\|\nabla g\|_2\cdot\|\Delta_W\|_F\cdot\|\Delta_Y\|_F &= \tfrac{1}{4}\|\nabla g\|_2\cdot\|\Delta_W\|_F^2\\
&\leq \tfrac{\sigma_r(W^\star)}{40\sqrt{\kappa}}\|\nabla g\|_2\cdot\|\Delta_W\|_F\\
&\leq \tfrac{\mu_{\min}\cdot\sigma_r(W^\star)^2}{100}\|\Delta_W\|_F^2+\tfrac{1}{16L_{\max}}\|\nabla g\|_2^2 \qquad (C.14)
\end{aligned}$$

where we use the AM-GM inequality. Plugging (C.13) and (C.14) in (C.12), it is easy to obtain:

$$
\begin{aligned}
&(A) + \tfrac{1}{2}(B) \\
&\geq \tfrac{\mu_{\min} \cdot \sigma_r(W^\star)^2}{20} \left\|\Delta_W\right\|_F^2 + \tfrac{1}{4L_{\max}} \left\|\nabla f(UV^\top)\right\|_F^2 + \tfrac{1}{16L_{\max}} \left\|\nabla g\right\|_F^2 - \tfrac{L}{2} \left\|X^\star - X_r^\star\right\|_F^2 .
\end{aligned}
$$

C.3 Proof of (Improved) Sublinear Convergence (Theorem 4.9)

The proof follows the same framework of the sublinear convergence proof in [19]. We use the following general lemma to prove the sublinear converegence.

Lemma C.3. *Suppose that a sequence of iterates $\{W_t\}_{t=0}^T$ satisfies the following conditions*

$$
f(W_t W_t^\top) - f(W_{t+1} W_{t+1}^\top) \geq \alpha \cdot \left\|\nabla_W f(W_t W_t^\top)\right\|_F^2 , \tag{C.15}
$$

$$
f(W_t W_t^\top) - f(W^\star W^{\star\top}) \leq \beta \cdot \left\|\nabla_W f(W_t W_t^\top)\right\|_F \tag{C.16}
$$

for all $t = 0, \ldots, T-1$ and some values $\alpha, \beta > 0$ independent of the iterates. Then it is guaranteed that

$$
f(W_T W_T^\top) - f(W^\star W^{\star\top}) \leq \frac{\beta^2}{\alpha \cdot T}
$$

Proof. Define $\delta_t = f(W_t W_t^\top) - f(W^\star W^{\star\top})$. If we get $\delta_{T_0} \leq 0$ at some $T_0 < T$, the desired inequality holds because the first hypothesis guarantees $\{\delta_t\}_{t=0}^T$ to be non-increasing. Hence, we can only consider the time t where $\delta_t, \delta_{t+1} \geq 0$.

We have

$$\begin{aligned}\delta_{t+1} &\overset{(a)}{\leq} \delta_t - \alpha \cdot \left\|\nabla_W f(W_t W_t^\top)\right\|_F^2 \\ &\overset{(b)}{\leq} \delta_t - \frac{\alpha}{\beta^2} \cdot \delta_t^2 \\ &\overset{(c)}{\leq} \delta_t - \frac{\alpha}{\beta^2} \cdot \delta_t \cdot \delta_{t+1}\end{aligned}$$

where (a) follows from the first hypothesis, (b) follows from the second hypothesis, (c) follows from that $\delta_{t+1} \leq \delta_t$ by the first hypothesis. Dividing by $\delta_t \cdot \delta_{t+1}$, we obtain

$$\frac{1}{\delta_{t+1}} - \frac{1}{\delta_t} \geq \frac{\alpha}{\beta^2}$$

Then we obtain the desired result by telescoping the above inequality. □

Now it suffices to show `BFGD` provides a sequence $\{W_t\}_{t=0}^T$ satisfies the hypotheses of Lemma C.3.

Obtaining (C.15) Although f is non-convex over the factor space, it is reasonable to obtain a new estimate (with a carefully chosen steplength) which is no worse than the current one, because the algorithm takes a gradient step.

Lemma C.4. *Let f be a L-smooth convex function. Moreover, consider the recursion in Let $X = WW^\top$ and $X^+ = W^+W^{+\top}$ be two consecutive estimates of `BFGD`. Then*

$$f(WW^\top) - f(W^+W^{+\top}) \geq \frac{3\eta}{5} \cdot \left\|\nabla_W f(WW^\top)\right\|_F^2 \tag{C.17}$$

Since we can fix the steplength η based on the initial solution so that it is independent of the following iterates, we have obtained the first hypothesis of Lemma C.3.

Obtaining (C.16) Consider the following assumption.

$$(\mathrm{A}): \quad \mathrm{dist}(U, V; X_r^\star) = \min_{R \in O(r)} \|W - W^\star R\|_F \leq \frac{\sigma_r(W^\star)}{10}$$

Trivially (A) holds for U_0 and V_0. Now we provide key lemmas, and then the convergence proof will be presented.

Lemma C.5 (Suboptimality bound)**.** *Assume that (A) holds for W. Then we have*

$$f(WW^\top) - f(W^\star W^{\star\top}) \leq \frac{7}{3} \cdot \left\|\nabla_W f(WW^\top)\right\|_F \cdot \mathrm{dist}(U, V; X_r^\star)$$

Lemma C.6 (Descent in distance)**.** *Assume that (A) holds for W. If*

$$f(W^+ W^{+\top}) \geq f(W^\star W^{\star\top}),$$

then

$$\mathrm{dist}(U^+, V^+; X_r^\star) \leq \mathrm{dist}(U, V; X_r^\star)$$

Combining the above two lemmas, we obtain

$$f(WW^\top) - f(W^\star W^{\star\top}) \leq \frac{7 \cdot \mathrm{dist}(U_0, V_0; X_r^\star)}{3} \cdot \left\|\nabla_W f(WW^\top)\right\|_F \tag{C.18}$$

Plugging (C.17) and (C.18) in Lemma C.3, we obtain the desired result.

C.3.1 Proof of Lemma C.4

The L-smoothness gives

$$\begin{aligned}
&f(WW^\top) - f(W^+ W^{+\top}) \\
&\geq \left\langle \nabla f(WW^\top), WW^\top - W^+ W^{+\top} \right\rangle - \frac{L}{2} \left\| WW^\top - W^+ W^{+\top} \right\|_F^2 \\
&= \left\langle \nabla f(WW^\top), (W - W^+)W^\top + W(W - W^+)^\top \right\rangle \qquad \text{(C.19)} \\
&\quad - \left\langle \nabla f(WW^\top), (W - W^+)(W - W^+)^\top \right\rangle \\
&\quad - \frac{L}{2} \left\| WW^\top - W^+ W^{+\top} \right\|_F^2 \qquad \text{(C.20)}
\end{aligned}$$

For the first term, we have

$$\begin{aligned}
&\left\langle \nabla f(WW^\top), (W - W^+)W^\top + W(W - W^+)^\top \right\rangle \\
&= 2 \cdot \left\langle \nabla f(WW^\top)W, W - W^+ \right\rangle \\
&= \eta \cdot \left\| \nabla_W f(WW^\top) \right\|_F^2
\end{aligned} \tag{C.21}$$

Using the Cauchy-Schwarz inequality, the second term can be bounded as follows.

$$\begin{aligned}
&\left\langle \nabla f(WW^\top), (W - W^+)(W - W^+)^\top \right\rangle \\
&= \eta^2 \cdot \left\langle \nabla f(WW^\top), \nabla_W f(WW^\top) \cdot \nabla_W f(WW^\top)^\top \right\rangle \\
&= \eta^2 \cdot \left\langle \nabla f(WW^\top) \cdot \nabla_W f(WW^\top), \nabla_W f(WW^\top) \right\rangle \\
&\le \eta^2 \cdot \left\| \nabla f(WW^\top) \cdot \nabla_W f(WW^\top) \right\|_F \cdot \left\| \nabla_W f(WW^\top) \right\|_F \\
&\le \eta^2 \cdot \left\| \nabla f(WW^\top) \right\|_2 \cdot \left\| \nabla_W f(WW^\top) \right\|_F^2
\end{aligned} \tag{C.22}$$

To bound the third term of (C.19), we have

$$\begin{aligned}
&\|WW^\top - W^+{W^+}^\top\|_F \\
&\le \|WW^\top - W{W^+}^\top\|_F + \|W{W^+}^\top - W^+{W^+}^\top\|_F \\
&\le (\|W\|_2 + \left\|W^+\right\|_2) \cdot \|W - W^+\|_F \\
&\le \eta \cdot \left(2\|W\|_2 + \eta \cdot \left\|\nabla f(WW^\top)\right\|_2 \cdot \|W\|_2\right) \cdot \|\nabla_W f(WW^\top)\|_F \\
&\le \frac{7\eta}{3} \|W\|_2 \cdot \|\nabla_W f(WW^\top)\|_F
\end{aligned} \tag{C.23}$$

Plugging (C.21), (C.22), and (C.23) to (C.19), we obtain

$$\begin{aligned}
&f(WW^\top) - f(W^+{W^+}^\top) \\
&\ge \eta \cdot \left\| \nabla_W f(UV^\top) \right\|_F^2 \cdot \left(1 - \eta \frac{17L\|W\|_2^2 + 3\|\nabla f(WW^\top)\|_2}{3}\right) \\
&\ge \frac{3\eta}{5} \cdot \left\| \nabla_W f(UV^\top) \right\|_F^2
\end{aligned}$$

where the last inequality follows from the condition of the steplength η. This completes the proof.

C.3.2 Proof of Lemma C.5

We use the following lemma.

Lemma C.7 (Error bound)**.** *Assume that (A) holds for W. Then*

$$\left\langle \nabla f(WW^\top), \Delta_W \Delta_W^\top \right\rangle \leq \frac{1}{3} \cdot \left\| \nabla_W f(UV^\top) \right\|_F \cdot \mathrm{dist}(U, V; X_r^\star)$$

Now the lemma is proved as follows.

$$\begin{aligned}
& f(WW^\top) - f(W^\star W^{\star\top}) \\
& \overset{(a)}{\leq} \langle \nabla f(WW^\top), WW^\top - W^\star W^{\star\top} \rangle \\
& = \langle \nabla f(WW^\top), \Delta_W W^\top \rangle + \langle \nabla f(WW^\top), W\Delta_W^\top \rangle - \langle \nabla f(WW^\top), \Delta_W \Delta_W^\top \rangle \\
& = 2\langle \nabla f(WW^\top) W, \Delta_W \rangle - \langle \nabla f(WW^\top), \Delta_W \Delta_W^\top \rangle \\
& \overset{(b)}{\leq} 2 \cdot \left\| \nabla_W f(WW^\top) \right\|_F \cdot \|\Delta_W\|_F + |\langle \nabla f(WW^\top), \Delta_W \Delta_W^\top \rangle| \\
& \overset{(c)}{\leq} \frac{7}{3} \cdot \left\| \nabla_W f(WW^\top) \right\|_F \cdot \|\Delta_W\|_F
\end{aligned}$$

(a) follows from the convexity of f, (b) follows from the Cauchy-Schwarz inequality, and (c) follows from Lemma C.7.

C.3.3 Proof of Lemma C.7

Define Q_W, $Q_{W^\star}$, and Q_{Δ_W} as the projection matrices of the column spaces of W, $W^\star$, and $\Delta_W = W - W^\star R$, respectively. We have

$$\begin{aligned}\left\langle \nabla f(WW^\top), \Delta_W \Delta_W^\top \right\rangle &= \left\langle \nabla f(WW^\top) Q_{\Delta_W}, \Delta_W \Delta_W^\top \right\rangle \\ &\overset{(a)}{\leq} \left\| \nabla f(WW^\top) Q_{\Delta_W} \right\|_2 \cdot \|\Delta_W\|_F^2 \\ &\overset{(b)}{\leq} \left(\left\| \nabla f(WW^\top) Q_W \right\|_2 + \left\| \nabla f(WW^\top) Q_{W^\star} \right\|_2 \right) \cdot \|\Delta_W\|_F^2 \end{aligned} \tag{C.24}$$

where (a) follows from the Cauchy-Schwarz inequality and the fact $\|AB\|_F \leq \|A\|_2 \cdot \|B\|_F$, and (b) follows from that $W - W^\star$ lies on the column space spanned by W and $W^\star$. To bound the terms in (C.24), we obtain

$$\begin{aligned}\left\| \nabla f(WW^\top) Q_W \right\|_2 &= \left\| \nabla f(WW^\top) W W^\dagger \right\|_2 \\ &\leq \frac{1}{\sigma_r(W)} \left\| \nabla f(WW^\top) W \right\|_2 \\ &\leq \frac{10}{9\sigma_r(W^\star)} \left\| \nabla f(WW^\top) W \right\|_2, \\ \left\| \nabla f(WW^\top) Q_{W^\star} \right\|_2 &= \left\| \nabla f(WW^\top) W^\star W^{\star\dagger} \right\|_2 \\ &\leq \frac{1}{\sigma_r(W^\star)} \left\| \nabla f(WW^\top) W^\star \right\|_2, \\ \|\nabla f(WW^\top) W^\star\|_2 &\leq \|\nabla f(WW^\top) W\|_2 + \|\nabla f(WW^\top) \Delta_W\|_2 \\ &\leq \|\nabla f(WW^\top) W\|_2 \\ &\quad + \left(\|\nabla f(WW^\top) Q_W\|_2 + \|\nabla f(WW^\top) Q_{W^\star}\|_2 \right) \cdot \|\Delta_W\|_2 \\ &\leq \frac{10}{9} \|\nabla f(WW^\top) W\|_2 + \frac{1}{10} \cdot \|\nabla f(WW^\top) W^\star\|_2, \\ \|\nabla f(WW^\top) Q_{W^\star}\|_2 &\leq \frac{1}{\sigma_r(W^\star)} \|\nabla f(WW^\top) W^\star\|_2 \leq \frac{5}{4\sigma_r(W^\star)} \|\nabla f(WW^\top) W\|_2, \end{aligned}$$

where $W^\dagger$ and $W^{\star\dagger}$ are the Moore-Penrose pseudoinverses of W and $W^\star$. Plugging the above into (C.24), we get

$$\begin{aligned}\left\langle \nabla f(WW^\top), \Delta_W \Delta_W^\top \right\rangle &\leq \frac{95}{36\sigma_r(W^\star)} \cdot \left\| \nabla f(WW^\top) W \right\|_2 \cdot \|\Delta_W\|_F^2 \\ &\overset{(a)}{\leq} \frac{1}{3} \cdot \left\| \nabla f(WW^\top) W \right\|_2 \cdot \|\Delta_W\|_F\end{aligned}$$

where (a) follows from (A).

C.3.4 Proof of Lemma C.6

For this proof, we borrow a lemma from [19]. Although the assumption for the lemma is stronger than Assumption (A), but a slight modification of the proof leads to the following lemma from Assumption (A).

Lemma C.8 (Lemma C.2 of [19])**.** *Let $f(W^+ W^{+\top}) \geq f(W^\star W^{\star\top})$, and Assumption (A) holds. Then the following lower bound holds:*

$$\left\langle \nabla f(WW^\top), \Delta_W \Delta_W^\top \right\rangle \geq -\frac{\sqrt{2}}{\sqrt{2} - \frac{1}{10}} \cdot \frac{1}{10} \cdot \left| \left\langle \nabla f(WW^\top), WW^\top - W^\star W^{\star\top} \right\rangle \right|.$$

We have

$$
\begin{aligned}
&\operatorname{dist}(U, V; X_r^\star)^2 - \operatorname{dist}(U^+, V^+; X_r^\star)^2 \\
&\geq \|W - W^\star R\|_F^2 - \left\|W^+ - W^\star R\right\|_F^2 \\
&= 2\eta \left\langle \nabla_W f(WW^\top), \Delta_W \right\rangle - \eta^2 \left\|\nabla_W f(WW^\top)\right\|_F^2 \\
&= 4\eta \left\langle \nabla f(WW^\top) W, \Delta_W \right\rangle - \eta^2 \left\|\nabla_W f(WW^\top)\right\|_F^2 \\
&= 2\eta \left\langle \nabla f(WW^\top), WW^\top - W^\star W^{\star\top} \right\rangle \\
&\qquad + 2\eta \left\langle \nabla f(WW^\top), \Delta_W \Delta_W^\top \right\rangle - \eta^2 \left\|\nabla_W f(WW^\top)\right\|_F^2 \\
&\overset{(a)}{\geq} \frac{17\eta}{10} \left\langle \nabla f(WW^\top), WW^\top - W^\star W^{\star\top} \right\rangle - \eta^2 \left\|\nabla_W f(WW^\top)\right\|_F^2 \\
&\overset{(b)}{\geq} \frac{51\eta^2}{50} \left\|\nabla_W f(WW^\top)\right\|_F^2 - \eta^2 \left\|\nabla_W f(WW^\top)\right\|_F^2 \\
&\geq 0
\end{aligned} \tag{C.25}
$$

where (a) follows from Lemma C.8, (b) follows from the convexity of f, the hypothesis of the lemma, and Lemma C.4 as follows.

$$
\begin{aligned}
\left\langle \nabla f(WW^\top), WW^\top - W^\star W^{\star\top} \right\rangle &\geq f(WW^\top) - f(W^\star W^{\star\top}) \\
&\geq f(WW^\top) - f(W^+ W^{+\top}) \\
&\geq \frac{3\eta}{5} \cdot \left\|\nabla_W f(WW^\top)\right\|_F^2
\end{aligned}
$$

This completes the proof.

C.4 Proof of Lemma 4.10

The triangle inequality gives that

$$
\left\|U_0 V_0^\top - X_r^\star\right\|_F \leq \left\|U_0 V_0^\top - X_0\right\|_F + \|X_0 - X^\star\|_F + \|X^\star - X_r^\star\|_F \tag{C.26}
$$

Let us first obtain an upper bound on the first term. We have

$$\begin{aligned}
\left\|X_0 - U_0 V_0^\top\right\|_F &= \left\| \begin{bmatrix} \sigma_{r+1}(X_0) \\ \vdots \\ \sigma_{\min\{m,n\}}(X_0) \end{bmatrix} \right\| \\
&\overset{(a)}{\leq} \left\| \begin{bmatrix} \sigma_{r+1}(X^\star) \\ \vdots \\ \sigma_{\min\{m,n\}}(X^\star) \end{bmatrix} \right\| + \left\| \begin{bmatrix} \sigma_{r+1}(X_0) - \sigma_{r+1}(X^\star) \\ \vdots \\ \sigma_{\min\{m,n\}}(X_0) - \sigma_{\min\{m,n\}}(X^\star) \end{bmatrix} \right\| \\
&= \left\|X^\star - X_r^\star\right\|_F + \sqrt{\sum_{i=r+1}^{\min\{m,n\}} (\sigma_i(X_0) - \sigma_i(X^\star))^2} \\
&\overset{(b)}{\leq} \left\|X^\star - X_r^\star\right\|_F + \left\|X_0 - X^\star\right\|_F
\end{aligned}$$

where (a) follows from the triangle inequality, and (b) is due to Mirsky's theorem [121]. Plugging this bound into (C.26), we get

$$\left\|U_0 V_0^\top - X_r^\star\right\|_F \leq 2\left\|X_0 - X^\star\right\|_F + 2\left\|X^\star - X_r^\star\right\|_F \tag{C.27}$$

Now we bound the first term of (C.27). We have

$$\begin{aligned}
&\left\|X_0\right\|_F = \frac{1}{L}\left\|\nabla f(0)\right\|_F = \frac{1}{L}\left\|\nabla f(0) - \nabla f(X^\star)\right\|_F \overset{(a)}{\leq} \left\|0 - X^\star\right\|_F = \left\|X^\star\right\|_F, \\
&L\left\langle X_0, X^\star\right\rangle = -\left\langle \nabla f(0), X^\star\right\rangle \overset{(b)}{\geq} f(0) - f(X^\star) + \frac{\mu}{2}\left\|X^\star\right\|_F^2 \overset{(c)}{\geq} \mu\left\|X^\star\right\|_F^2
\end{aligned}$$

where (a) follows from the L-smoothness, (b) and (c) follow from the μ-strong convexity. Then it follows that

$$\left\|X_0 - X^\star\right\|_F^2 = \left\|X_0\right\|_F^2 + \left\|X^\star\right\|_F^2 - 2\left\langle X_0, X^\star\right\rangle \leq 2(1 - \frac{\mu}{L})\left\|X^\star\right\|_F^2$$

Applying this inequality to (C.27), we get the desired inequality.

Bibliography

[1] S. Aaronson. The learnability of quantum states. In *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, volume 463, pages 3089–3114. The Royal Society, 2007.

[2] A. Agarwal, S. Negahban, and M. Wainwright. Fast global convergence rates of gradient methods for high-dimensional statistical recovery. In *Advances in Neural Information Processing Systems*, pages 37–45, 2010.

[3] R. Agrawal, A. Gupta, Y. Prabhu, and M. Varma. Multi-label learning with millions of labels: Recommending advertiser bid phrases for web pages. In *Proceedings of the 22nd international conference on World Wide Web*, pages 13–24. International World Wide Web Conferences Steering Committee, 2013.

[4] Nir Ailon. Active learning ranking from pairwise preferences with almost optimal query complexity. In *Advances in Neural Information Processing Systems (NIPS)*, pages 810–818, 2011.

[5] A. Anandkumar and R. Ge. Efficient approaches for escaping higher order saddle points in non-convex optimization. *arXiv preprint arXiv:1602.05908*, 2016.

[6] H. Andrews and C. Patterson III. Singular value decomposition (SVD) image coding. *Communications, IEEE Transactions on*, 24(4):425–432, 1976.

[7] A. Aravkin, R. Kumar, H. Mansour, B. Recht, and F. Herrmann. Fast methods for denoising matrix completion formulations, with applications to robust seismic data interpolation. *SIAM Journal on Scientific Computing*, 36(5):S237–S266, 2014.

[8] E. Arias-Castro, G. Chen, and G. Lerman. Spectral clustering based on local linear approximations. *Electronic Journal of Statistics*, 5:1537–1587, 2011.

[9] J. Baglama and L. Reichel. Augmented implicitly restarted Lanczos bidiagonalization methods. *SIAM Journal on Scientific Computing*, 27(1):19–42, 2005.

[10] S. Balakrishnan, M. Wainwright, and B. Yu. Statistical guarantees for the EM algorithm: From population to sample-based analysis. *arXiv preprint arXiv:1408.2156*, 2014.

[11] Suhrid Balakrishnan and Sumit Chopra. Collaborative ranking. In *ACM International Conference on Web Search and Data Mining (WSDM)*, 2012.

[12] L. Balzano, R. Nowak, and B. Recht. Online identification and tracking of subspaces from highly incomplete information. In *Communication, Control, and Computing (Allerton), 2010 48th Annual Allerton Conference on*, pages 704–711. IEEE, 2010.

[13] R. Baraniuk. Compressive sensing. *IEEE signal processing magazine*, 24(4), 2007.

[14] S. Becker, J. Bobin, and E. Candès. NESTA: A fast and accurate first-order method for sparse recovery. *SIAM Journal on Imaging Sciences*, 4(1):1–39, 2011.

[15] S. Becker, E. Candès, and M. Grant. Templates for convex cone problems with applications to sparse signal recovery. *Mathematical Programming Computation*, 3(3):165–218, 2011.

[16] S. Becker, V. Cevher, and A. Kyrillidis. Randomized low-memory singular value projection. In *10th International Conference on Sampling Theory and Applications (Sampta)*, 2013.

[17] J. Bennett and S. Lanning. The Netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35, 2007.

[18] K. Bhatia, H. Jain, P. Kar, M. Varma, and P. Jain. Sparse local embeddings for extreme multi-label classification. In *Advances in Neural Information Processing Systems*, pages 730–738, 2015.

[19] S. Bhojanapalli, A. Kyrillidis, and S. Sanghavi. Dropping convexity for faster semi-definite optimization. *arXiv preprint arXiv:1509.03917*, 2015.

[20] P. Biswas, T.-C. Liang, K.-C. Toh, Y. Ye, and T.-C. Wang. Semidefinite programming approaches for sensor network localization with noisy distance measurements. *IEEE transactions on automation science and engineering*, 3(4):360, 2006.

[21] N. Boumal and P.-A. Absil. RTRMC: A Riemannian trust-region method for low-rank matrix completion. In *Advances in neural information processing systems*, pages 406–414, 2011.

[22] P. S. Bradley and O. L. Mangasarian. K-plane clustering. *Journal of Global Optimization*, 16(1):23–32, 2000.

[23] Ralph Allan Bradley and Milton E Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, pages 324–345, 1952.

[24] S. Burer and R. Monteiro. A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization. *Mathematical Programming*, 95(2):329–357, 2003.

[25] S. Burer and R. Monteiro. Local minima and convergence in low-rank semidefinite programming. *Mathematical Programming*, 103(3):427–444, 2005.

[26] J. Cai, E. Candès, and Z. Shen. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, 20(4):1956–1982, 2010.

[27] E. Candes, Y. Eldar, T. Strohmer, and V. Voroninski. Phase retrieval via matrix completion. *SIAM Review*, 57(2):225–251, 2015.

[28] E. Candes, X. Li, Y. Ma, and J. Wright. Robust principal component analysis? *Journal of the ACM (JACM)*, 58(3):11, 2011.

[29] E. Candes, X. Li, and M. Soltanolkotabi. Phase retrieval via Wirtinger flow: Theory and algorithms. *Information Theory, IEEE Transactions on*, 61(4):1985–2007, 2015.

[30] E. Candes and Y. Plan. Tight oracle inequalities for low-rank matrix recovery from a minimal number of noisy random measurements. *Information Theory, IEEE Transactions on*, 57(4):2342–2359, 2011.

[31] E. Candès and B. Recht. Exact matrix completion via convex optimization. *Foundations of Computational mathematics*, 9(6):717–772, 2009.

[32] G. Carneiro, A. Chan, P. Moreno, and N. Vasconcelos. Supervised learning of semantic classes for image annotation and retrieval. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(3):394–410, 2007.

[33] V. Chandrasekaran, S. Sanghavi, P. Parrilo, and A. Willsky. Sparse and low-rank matrix decompositions. In *Communication, Control, and Computing, 2009. Allerton 2009. 47th Annual Allerton Conference on*, pages 962–967. IEEE, 2009.

[34] G. Chen and G. Lerman. Spectral curvature clustering. *International Journal of Computer Vision*, 81(3):317–330, 2009.

[35] Y. Chen, S. Bhojanapalli, S. Sanghavi, and R. Ward. Coherent matrix completion. In *Proceedings of The 31st International Conference on Machine Learning*, pages 674–682, 2014.

[36] Y. Chen and C. Caramanis. Noisy and missing data regression: Distribution-oblivious support recovery. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 383–391, 2013.

[37] Y. Chen and S. Sanghavi. A general framework for high-dimensional estimation in the presence of incoherence. In *Communication, Control,*

and Computing (Allerton), 2010 48th Annual Allerton Conference on, pages 1570–1576. IEEE, 2010.

[38] Y. Chen and M. Wainwright. Fast low-rank estimation by projected gradient descent: General statistical and algorithmic guarantees. *arXiv preprint arXiv:1509.03025*, 2015.

[39] Yudong Chen and Constantine Caramanis. Noisy and missing data regression: distribution-oblivious support recovery. In *Proceedings of International Conference on Machine Learning (ICML)*, 2013.

[40] K.-Y. Chiang, C.-J. Hsieh, N. Natarajan, I. Dhillon, and A. Tewari. Prediction and clustering in signed networks: A local to global perspective. *The Journal of Machine Learning Research*, 15(1):1177–1213, 2014.

[41] A. Christoffersson. *The one component model with incomplete data*. Uppsala., 1970.

[42] M. Cohen, J. Nelson, and D. Woodruff. Optimal approximate matrix product in terms of stable rank. *arXiv preprint arXiv:1507.02268*, 2015.

[43] M. Collins, S. Dasgupta, and R. Schapire. A generalization of principal components analysis to the exponential family. In *Advances in neural information processing systems*, pages 617–624, 2001.

[44] Yan Cui, Chun-Hou Zheng, and Jian Yang. Identifying subspace gene clusters from microarray data using low-rank representation. *PLoS ONE*, 8(3), 2013.

[45] M. Davenport, Y. Plan, E. van den Berg, and M. Wootters. 1-bit matrix completion. *Information and Inference*, 3(3):189–223, 2014.

[46] Mark A Davenport, Yaniv Plan, Ewout van den Berg, and Mary Wootters. 1-bit matrix completion. *Information and Inference*, 3(3):189–223, 2014.

[47] D. DeCoste. Collaborative prediction using ensembles of maximum margin matrix factorizations. In *Proceedings of the 23rd international conference on Machine learning*, pages 249–256. ACM, 2006.

[48] D. Donoho. Compressed sensing. *Information Theory, IEEE Transactions on*, 52(4):1289–1306, 2006.

[49] P. Drineas and R. Kannan. Fast Monte-Carlo algorithms for approximate matrix multiplication. In *focs*, page 452. IEEE, 2001.

[50] P. Drineas, R. Kannan, and M. Mahoney. Fast Monte Carlo algorithms for matrices I: Approximating matrix multiplication. *SIAM Journal on Computing*, 36(1):132–157, 2006.

[51] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159, 2011.

[52] E. L. Dyer, A. C. Sankaranarayanan, and R. G. Baraniuk. Greedy feature selection for subspace clustering. *The Journal of Machine Learning Research (JMLR)*, 14(1):2487–2517, 2013.

[53] E. Elhamifar and R. Vidal. Sparse subspace clustering: Algorithm, theory, and applications. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(11):2765–2781, 2013.

[54] M. Fazel. *Matrix rank minimization with applications*. PhD thesis, PhD thesis, Stanford University, 2002.

[55] M. Fazel, E. Candes, B. Recht, and P. Parrilo. Compressed sensing and robust recovery of low rank matrices. In *Signals, Systems and Computers, 2008 42nd Asilomar Conference on*, pages 1043–1047. IEEE, 2008.

[56] M. Fazel, H. Hindi, and S. Boyd. Rank minimization and applications in system theory. In *American Control Conference, 2004. Proceedings of the 2004*, volume 4, pages 3273–3278. IEEE, 2004.

[57] S. Flammia, D. Gross, Y.-K. Liu, and J. Eisert. Quantum tomography via compressed sensing: Error bounds, sample complexity and efficient estimators. *New Journal of Physics*, 14(9):095022, 2012.

[58] A. S. Georghiades, P. N. Belhumeur, and D. J. Kriegman. From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE Trans. Pattern Anal. Mach. Intelligence*, 23(6):643–660, 2001.

[59] B. Vikrham Gowreesunker and Ahmed H. Tewfik. Learning sparse representation using iterative subspace identification. *IEEE Transactions on Signal Processing*, 58(6), Jun 2010.

[60] D. Gross, Y.-K. Liu, S. Flammia, S. Becker, and J. Eisert. Quantum state tomography via compressed sensing. *Physical review letters*, 105(15):150401, 2010.

[61] N. Gupta and S. Singh. Collectively embedding multi-relational data for predicting user preferences. *arXiv preprint arXiv:1504.06165*, 2015.

[62] B. Haeffele, E. Young, and R. Vidal. Structured low-rank matrix factorization: Optimality, algorithm, and applications to image processing. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 2007–2015, 2014.

[63] Bruce Hajek, Sewoong Oh, and Jiaming Xu. Minimax-optimal inference from partial rankings. In *Advances in Neural Information Processing Systems (NIPS)*, 2014.

[64] N. Halko, P.-G. Martinsson, and J. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288, 2011.

[65] M. Hardt and M. Wootters. Fast matrix completion without the condition number. In *Proceedings of The 27th Conference on Learning Theory*, pages 638–678, 2014.

[66] Moritz Hardt and Eric Price. Tight bounds for learning a mixture of two gaussians. In *ACM Symposium on Theory of Computing (STOC)*, 2015.

[67] E. Hazan. Sparse approximate solutions to semidefinite programs. In *LATIN 2008: Theoretical Informatics*, pages 306–316. Springer, 2008.

[68] R. Heckel and H. Bölcskei. Subspace clustering via thresholding and spectral clustering. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, May 2013.

[69] R. Heckel and H. Bölcskei. Robust subspace clustering via thresholding. *arXiv preprint arXiv:1307.4891v2*, 2014.

[70] Reinhard Heckel, Michael Tschannen, and Bölcskei. Dimensionality-reduced subspace clustering. *arXiv preprint arXiv:1507.07105v2*, 2015.

[71] Ralf Herbrich, Thore Graepel, and Klaus Obermayer. *Large Margin Rank Boundaries for Ordinal Regression*, chapter 7, pages 115–132. MIT Press, January 2000.

[72] J. Ho, M.-H. Yang, J. Lim, K.-C. Lee, and D. Kriegman. Clustering appearances of objects under varying illumination conditions. In *IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2003.

[73] Cho-Jui Hsieh, Kai-Wei Chang, Chih-Jen Lin, S. Sathiya Keerthi, and S. Sundararajan. A dual coordinate descent method for large-scale linear SVM. In *International Conference on Machine Learning (ICML)*, 2008.

[74] Cho-Jui Hsieh, Hsiang-Fu Yu, and Inderjit S. Dhillon. PASSCoDe: Parallel asynchronous stochastic dual co-ordinate descent. In *International Conference on Machine Learning (ICML)*, 2015.

[75] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *IEEE International Conference on Data Mining (ICDM)*, pages 263–272. IEEE, 2008.

[76] K. Huang, A. Wagner, and Y. Ma. Identification of hybrid linear time-invariant systems via subspace embedding and segmentation (ses). In *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, volume 3, pages 3227–3234. IEEE, 2004.

[77] T. Inglot. Inequalities for quantiles of the chi-square distribution. *Probability and Mathematical Statistics*, 30(2):339–351, 2010.

[78] M. Jaggi and M. Sulovsk. A simple algorithm for nuclear norm regularized problems. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 471–478, 2010.

[79] P. Jain, C. Jin, S. Kakade, and P. Netrapalli. Computing matrix squareroot via non convex local search. *arXiv preprint arXiv:1507.05854*, 2015.

[80] P. Jain, R. Meka, and I. Dhillon. Guaranteed rank minimization via singular value projection. In *Advances in Neural Information Processing Systems*, pages 937–945, 2010.

[81] P. Jain, P. Netrapalli, and S. Sanghavi. Low-rank matrix completion using alternating minimization. In *Proceedings of the 45th annual ACM symposium on Symposium on theory of computing*, pages 665–674. ACM, 2013.

[82] Prateek Jain, Praneeth Netrapalli, and Sujay Sanghavi. Low-rank matrix completion using alternating minimization. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing (STOC)*, 2013.

[83] Ali Jalali, Christopher Johnson, and Pradeep Ravikumar. On learning discrete graphical models using greedy methods. In *Proceedings of Neural Information Processing Systems (NIPS)*, 2011.

[84] K. G. Jamieson and R. Nowak. Active ranking using pairwise comparisons. In *Advances in Neural Information Processing Systems (NIPS)*, 2011.

[85] Kevin G. Jamieson and Robert D. Nowak. Active ranking using pairwise comparisons. In *Advances in Neural Information Processing Systems (NIPS)*, 2011.

[86] A. Javanmard and A. Montanari. Localization from incomplete noisy distance measurements. *Foundations of Computational Mathematics*, 13(3):297–345, 2013.

[87] Thorsten Joachims. Optimizing search engines using clickthrough data. In *SIGKDD*, 2002.

[88] C. Johnson. Logistic matrix factorization for implicit feedback data. *Advances in Neural Information Processing Systems*, 27, 2014.

[89] A. Kalev, R. Kosut, and I. Deutsch. Quantum tomography protocols with positivity are compressed sensing protocols. *Nature partner journals (npj) Quantum Information*, 1:15018, 2015.

[90] Raghunandan Keshavan, Andrea Montanari, and Sewoong Oh. Matrix completion from a few entries. 56(6):2980–2998, June 2010.

[91] Raghunandan Keshavan, Andrea Montanari, and Sewoong Oh. Matrix completion from noisy entries. 11:2057–2078, July 2010.

[92] Raghunandan H Keshavan and Sewoong Oh. A gradient descent algorithm on the grassman manifold for matrix completion. *arXiv preprint arXiv:0910.5260*, 2009.

[93] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, (8):30–37, 2009.

[94] F. Krahmer and R. Ward. New and improved Johnson-Lindenstrauss embeddings via the restricted isometry property. *SIAM Journal on Mathematical Analysis*, 43(3):1269–1281, 2011.

[95] H.-P. Kriegel, P. Kröger, and A. Zimek. Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 3(1):1, 2009.

[96] S. Kunis and H. Rauhut. Random sampling of sparse trigonometric polynomials, ii. orthogonal matching pursuit versus basis pursuit. *Foundations of Computational Mathematics*, 8(6):737–763, 2008.

[97] A. Kyrillidis and V. Cevher. Matrix recipes for hard thresholding methods. *Journal of mathematical imaging and vision*, 48(2):235–265, 2014.

[98] A. Kyrillidis, M. Vlachos, and A. Zouzias. Approximate matrix multiplication with application to linear embeddings. In *Information Theory (ISIT), 2014 IEEE International Symposium on*, pages 2182–2186. Ieee, 2014.

[99] A. Landgraf and Y. Lee. Dimensionality reduction for binary data through the projection of natural parameters. *arXiv preprint arXiv:1510.06112*, 2015.

[100] R. Larsen. PROPACK-Software for large and sparse SVD calculations. *Available online. URL http://sun. stanford. edu/rmunk/PROPACK*, pages 2008–2009, 2004.

[101] B. Laurent and P. Massart. Adaptive estimation of a quadratic functional by model selection. *Annals of Statistics*, 28(5):1302–1338, 2000.

[102] M. Ledoux. *The concentration of measure phenomenon*, volume 89. AMS Bookstore, 2005.

[103] Joonseok Lee, Samy Bengio, Seungyeon Kim, Guy Lebanon, and Yoram Singer. Local collaborative ranking. In *International World Wide Web Conference (WWW)*, 2014.

[104] K. Lee and Y. Bresler. ADMiRA: Atomic decomposition for minimum rank approximation. *Information Theory, IEEE Transactions on*, 56(9):4402–4416, 2010.

[105] K. C. Lee, J. Ho, and D. Kriegman. Acquiring linear subspaces for face recognition under variable lighting. *IEEE Trans. Pattern Anal. Mach. Intelligence*, 27(5):684–698, 2005.

[106] R. Lehoucq, D. Sorensen, and C. Yang. *ARPACK users' guide: solution of large-scale eigenvalue problems with implicitly restarted Arnoldi methods*, volume 6. Siam, 1998.

[107] G. Lerman and T. Zhang. Robust recovery of multiple subspaces by geometric lp minimization. *The Annals of Statistics*, 39(5):2686–2715, 2011.

[108] Z. Lin, M. Chen, and Y. Ma. The augmented Lagrange multiplier method for exact recovery of corrupted low-rank matrices. *arXiv preprint arXiv:1009.5055*, 2010.

[109] G. Liu, Z. Lin, S. Yan, J. Sun, Y. Yu, and Y. Ma. Robust recovery of subspace structures by low-rank representation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(1):171–184, 2013.

[110] Nathan N Liu, Min Zhao, and Qiang Yang. Probabilistic latent preference analysis for collaborative filtering. In *Proceedings of the 18th ACM*

conference on Information and knowledge management, pages 759–766. ACM, 2009.

[111] Tie-Yan Liu. *Learning to Rank for Information Retrieval*. Now Publishers Inc., 2009.

[112] X. Liu, Z. Wen, and Y. Zhang. Limited memory block Krylov subspace optimization for computing dominant singular value decompositions. *SIAM Journal on Scientific Computing*, 35(3):A1641–A1668, 2013.

[113] Y. Liu, M. Wu, C. Miao, P. Zhao, and X.-L. Li. Neighborhood regularized logistic matrix factorization for drug-target interaction prediction. *PLoS Computational Biology*, 12(2):e1004760, 2016.

[114] Y.-K. Liu. Universal low-rank matrix recovery from Pauli measurements. In *Advances in Neural Information Processing Systems*, pages 1638–1646, 2011.

[115] Z. Liu and L. Vandenberghe. Interior-point method for nuclear norm approximation with application to system identification. *SIAM Journal on Matrix Analysis and Applications*, 31(3):1235–1256, 2009.

[116] F. Lu, S. Keles, S. Wright, and G. Wahba. Framework for kernel regularization with application to protein clustering. *Proceedings of the National Academy of Sciences of the United States of America*, 102(35):12332–12337, 2005.

[117] Yu Lu and Sahand Negahban. Individualized rank aggregation using nuclear norm regularization. *ArXiv e-prints: 1410.0860*, Oct 2014.

[118] Duncan R Luce. *Individual Choice Behavior*. Wiley, 1959.

[119] A. Makadia, V. Pavlovic, and S. Kumar. A new baseline for image annotation. In *Computer Vision–ECCV 2008*, pages 316–329. Springer, 2008.

[120] V. D. Milman and G. Schechtman. *Asymptotic Theory of Finite Dimensional Normed Spaces: Isoperimetric Inequalities in Riemannian Manifolds*. Lecture Notes in Mathematics. Springer, 1986.

[121] L. Mirsky. Symmetric gage functions and unitarily invariant norms. *Quarterly Journal of Mathematics*, 11:50–59, 1960.

[122] Ankur Moitra and Gregory Valiant. Settling the polynomial learnability of mixtures of gaussians. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, 2010.

[123] S. Negahban and M. Wainwright. Restricted strong convexity and weighted matrix completion: Optimal bounds with noise. *The Journal of Machine Learning Research*, 13(1):1665–1697, 2012.

[124] Sahand Negahban, Sewoong Oh, and Devavrat Shah. Iterative ranking from pair-wise comparisons. In *Advances in Neural Information Processing Systems (NIPS)*, 2012.

[125] Yurii Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*. Springer, 2004.

[126] Praneeth Netrapalli, Siddhartha Banerjee, Sujay Sanghavi, and Sanjay Shakkottai. Greedy learning of markov network structure. In *Proceedings*

of 48th Annual Allerton Conference on Communication, Control, and Computing, 2010.

[127] Praneeth Netrapalli, Prateek Jain, and Sujay Sanghavi. Phase retrieval using alternating minimization. *IEEE Transactions on Signal Processing*, 63(18):4814–4826, 2015.

[128] Praneeth Netrapalli, U. N. Niranjan, Sujay Sanghavi, Animashree Anandkumar, and Prateek Jain. Non-convex robust pca, 2014.

[129] Feng Niu, Benjamin Recht, Christopher Ré, and Stephen Wright. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in Neural Information Processing Systems (NIPS)*, 2011.

[130] Sewoong Oh, Kiran K. Thekumparampil, and Jiaming Xu. Collaboratively learning preference from ordinal data. In *Proceedings of Neural Information Processing Systems (NIPS)*, pages 172–180, 2015.

[131] D. Park, A. Kyrillidis, S. Bhojanapalli, C. Caramanis, and S. Sanghavi. Provable non-convex projected gradient descent for a class of constrained matrix optimization problems. *arXiv preprint arXiv:1606.01316*, 2016.

[132] K. Pearson. Principal components analysis. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 6(2):559, 1901.

[133] Avik Ray, Sujay Sanghavi, and Sanjay Shakkottai. Greedy learning of graphical models with small girth. In *Proceedings of 50th Annual Allerton Conference on Communication, Control, and Computing*, 2012.

[134] B. Recht, M. Fazel, and P. Parrilo. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM review*, 52(3):471–501, 2010.

[135] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 452–461. AUAI Press, 2009.

[136] J. Rennie and N. Srebro. Fast maximum margin matrix factorization for collaborative prediction. In *Proceedings of the 22nd international conference on Machine learning*, pages 713–719. ACM, 2005.

[137] A. Ruhe. *Numerical computation of principal components when several observations are missing*. Univ., 1974.

[138] Andrew I. Schein, Lawrence K. Saul, and Lyle H. Ungar. A generalized linear model for principal component analysis of binary data. In *AISTATS*, 2003.

[139] Yoav Seginer. The expected norm of random matrices. *Combinatorics Probability and Computing*, 9(2):149–166, 2000.

[140] Shai Shalev-Shwartz and Tong Zhang. Stochastic dual coordinate ascent methods for regularized loss minimization. *Journal of Machine Learning Research (JMLR)*, pages 567–599, 2013.

[141] Yue Shi, Alexandros Karatzoglou, Linas Baltrunas, Martha Larson, Nuria Oliver, and Alan Hanjalic. Climf: collaborative less-is-more filtering. In *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, pages 3077–3081. AAAI Press, 2013.

[142] M. Soltanolkotabi and E. J. Candes. A geometric analysis of subspace clustering with outliers. *The Annals of Statistics*, 40(4):2195–2238, 2012.

[143] M. Soltanolkotabi, E. Elhamifar, and E. J. Candes. Robust subspace clustering. *arXiv preprint arXiv:1301.2603*, 2013.

[144] N. Srebro, J. Rennie, and T. Jaakkola. Maximum-margin matrix factorization. In *Advances in neural information processing systems*, pages 1329–1336, 2004.

[145] Nathan Srebro, Jason Rennie, and Tommi Jaakkola. Maximum margin matrix factorization. In *Advances in Neural Information Processing Systems (NIPS)*, 2004.

[146] R Sun and Z.-Q. Luo. Guaranteed matrix completion via nonconvex factorization. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015*, pages 270–289, 2015.

[147] J. Tanner and K. Wei. Normalized iterative hard thresholding for matrix completion. *SIAM Journal on Scientific Computing*, 35(5):S104–S125, 2013.

[148] M. Tipping and C. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(3):611–622, 1999.

[149] R. Tron and R. Vidal. A benchmark for the comparison of 3-d motion segmentation algorithms. In *IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[150] J. A. Tropp and A. C. Gilbert. Signal recovery from random measurements via orthogonal matching pursuit. *Information Theory, IEEE Transactions on*, 53(12):4655–4666, 2007.

[151] P. Tseng. Nearest q-flat to m points. *Journal of Optimization Theory and Applications*, 105(1):249–252, 2000.

[152] S. Tu, R. Boczar, M. Soltanolkotabi, and B. Recht. Low-rank solutions of linear matrix equations via Procrustes flow. *arXiv preprint arXiv:1507.03566v2*, 2016.

[153] Roman Vershynin. *Compressed sensing: theory and applications*, chapter Introduction to the non-asymptotic analysis of random matrices. Cambridge University Press, 2012.

[154] Koen Verstrepen. *Collaborative Filtering with Binary, Positive-only Data*. PhD thesis, University of Antwerpen, 2015.

[155] R. Vidal. Subspace clustering. *Signal Processing Magazine, IEEE*, 28(2):52–68, 2011.

[156] R. Vidal, Y. Ma, and S. Sastry. Generalized principal component analysis. In *IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2003.

[157] R. Vidal, S. Soatto, Y. Ma, and S. Sastry. An algebraic geometric approach to the identification of a class of linear hybrid systems. In *Decision and Control, 2003. Proceedings. 42nd IEEE Conference on*, volume 1, pages 167–172. IEEE, 2003.

[158] R. Vidal, R. Tron, and R. Hartley. Multiframe motion segmentation with missing data using power factorization and GPCA. *International Journal of Computer Vision*, 79(1):85–105, 2008.

[159] Maksims N. Volkovs and Richard S. Zemel. Collaborative ranking with 17 parameters. In *Advances in Neural Information Processing Systems (NIPS)*, 2012.

[160] I. Waldspurger, A. dAspremont, and S. Mallat. Phase recovery, MaxCut and complex semidefinite programming. *Mathematical Programming*, 149(1-2):47–81, 2015.

[161] C. Wang, S. Yan, L. Zhang, and H.-J. Zhang. Multi-label sparse coding for automatic image annotation. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1643–1650. IEEE, 2009.

[162] Y.-X. Wang and H. Xu. Noisy sparse subspace clutering. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, pages 89–97, 2013.

[163] Y.-X. Wang, H. Xu, and C. Leng. Provable subspace clustering: When LRR meets SSC. In *Advances in Neural Information Processing Systems (NIPS)*, December 2013.

[164] Yining Wang, Yu-Xiang Wang, and Aarti Singh. Graph connectivity in noisy sparse subspace clustering. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2016.

[165] A. Waters, A. Sankaranarayanan, and R. Baraniuk. SpaRCS: Recovering low-rank and sparse matrices from compressive measurements. In

Advances in neural information processing systems, pages 1089–1097, 2011.

[166] Fabian L. Wauthier, Michael I. Jordan, and Nebojsa Jojic. Efficient ranking from pairwise comparisons. In *International Conference on Machine Learning (ICML)*, 2013.

[167] Markus Weimer, Alexandros Karatzoglou, Quoc V. Le, and Alex Smola. Cofirank: maximum margin matrix factorization for collaborative ranking. In *Advances in Neural Information Processing Systems (NIPS)*, 2007.

[168] K. Weinberger, F. Sha, Q. Zhu, and L. Saul. Graph Laplacian regularization for large-scale semidefinite programming. In *Advances in Neural Information Processing Systems*, pages 1489–1496, 2007.

[169] Z. Wen, W. Yin, and Y. Zhang. Solving a low-rank factorization model for matrix completion by a nonlinear successive over-relaxation algorithm. *Mathematical Programming Computation*, 4(4):333–361, 2012.

[170] J. Weston, S. Bengio, and N. Usunier. WSABIE: Scaling up to large vocabulary image annotation. In *IJCAI*, volume 11, pages 2764–2770, 2011.

[171] Jason Weston, Chong Want, Ron Weiss, and Adam Berenzeig. Latent collaborative retrieval. In *International Conference on Machine Learning (ICML)*, 2012.

[172] H. Wold and E. Lyttkens. Nonlinear iterative partial least squares (NIPALS) estimation procedures. *Bulletin of the International Statistical Institute*, 43(1), 1969.

[173] A. Y. Yang, S. R. Rao, and Y. Ma. Robust statistical estimation and segmentation of multiple subspaces. In *IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2006.

[174] Jinfeng Yi, Rong Jin, Shaili Jain, and Anil Jain. Inferring users preferences from crowdsourced pairwise comparisons: A matrix completion approach. In *First AAAI Conference on Human Computation and Crowdsourcing*, 2013.

[175] Xinyang Yi, Constantine Caramanis, and Sujay Sanghavi. Alternating minimization for mixed linear regression. In *Proceedings of International Conference on Machine Learning*, 2014.

[176] Xinyang Yi, Dohyung Park, Yudong Chen, and Constantine Caramanis. Fast algorithms for robust PCA via gradient descent. *arXiv preprint arXiv:1605.07784*, 2016.

[177] Chong You, Daniel P. Robinson, and René Vidal. Scalable sparse subspace clustering by orthogonal matching pursuit. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[178] Hyokun Yun, Parameswaran Raman, and S. V. N. Vishwanathan. Ranking via robust binary classification and parallel parameter estimation in large-scale data. In *Advances in Neural Information Processing Systems (NIPS)*, 2014.

[179] Hyokun Yun, Hsiang-Fu Yu, Cho-Jui Hsieh, S. V. N. Viswanathan, and Inderjit S. Dhillon. NOMAD: Non-locking, stochastic multi-machine algorithm for asynchronous and descentralized matrix completion. In *VLDB*, 2014.

[180] A. Yurtsever, Q. Tran-Dinh, and V. Cevher. A universal primal-dual convex optimization framework. In *Advances in Neural Information Processing Systems 28*, pages 3132–3140. 2015.

[181] T. Zhang, A. Szlam, Y. Wang, and G. Lerman. Hybrid linear modeling via local best-fit flats. *International journal of computer vision*, 100(3):217–240, 2012.

[182] Tong Zhang. On the consistency of feature selection using greedy least squares regression. *Journal of Machine Learning Research*, 10:555–568, 2009.

[183] T. Zhao, Z. Wang, and H. Liu. A nonconvex optimization framework for low rank matrix estimation. In *Advances in Neural Information Processing Systems 28*, pages 559–567. 2015.

[184] Q. Zheng and J. Lafferty. A convergent gradient descent algorithm for rank minimization and semidefinite programming from random linear measurements. In *Advances in Neural Information Processing Systems*, pages 109–117, 2015.

[185] Q. Zheng and J. Lafferty. Convergence analysis for rectangular matrix completion using burer-monteiro factorization and gradient descent. *arXiv preprint arXiv:1605.07051*, 2016.

Vita

Dohyung Park received the Bachelor of Science degree and the Master of Science degree in Electrical Engineering from Korea Advanced Institute of Science and Technology (KAIST), in 2006 and 2008, respectively. From 2008 to 2011, he was a R&D staff member of Samsung Advanced Institute of Technology (SAIT) in Korea. In the summer of 2015, he worked as a software engineering intern at Facebook, Inc. His primal research interests are statistical machine learning and large-scale optimization. His doctoral research at UT Austin has been advised by Prof. Sujay Sanghavi and Prof. Constantine Caramanis.

Permanent address: dhpark@utexas.edu

This dissertation was typeset with LaTeX[†] by the author.

[†]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.