Convex Optimization Algorithms and Recovery Theories for Sparse Models in Machine Learning

Bo Huang

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy
in the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2014

ABSTRACT

Convex Optimization Algorithms and Recovery Theories for Sparse Models in Machine Learning

Bo Huang

Sparse modeling is a rapidly developing topic that arises frequently in areas such as machine learning, data analysis and signal processing. One important application of sparse modeling is the recovery of a high-dimensional object from relatively low number of noisy observations, which is the main focuses of the Compressed Sensing [14,22], Matrix Completion(MC) [13,34,68] and Robust Principal Component Analysis (RPCA) [12]. However, the power of sparse models is hampered by the unprecedented size of the data that has become more and more available in practice. Therefore, it has become increasingly important to better harnessing the convex optimization techniques to take advantage of any underlying "sparsity" structure in problems of extremely large size.

This thesis focuses on two main aspects of sparse modeling. From the modeling perspective, it extends convex programming formulations for matrix completion and robust principal component analysis problems to the case of tensors, and derives theoretical guarantees for exact tensor recovery under a framework of strongly convex programming. On the optimization side, an efficient first-order algorithm with the optimal convergence rate has been proposed and studied for a wide range of problems of linearly constraint sparse modeling problems.

In Chapter 2, we generalize matrix completion and matrix RPCA models and rigorously study tractable models for provably recovering low-rank tensors. Unlike their matrix-based predecessors, current convex approaches for recovering low-rank tensors based on incomplete (tensor completion) or/and grossly corrupted(tensor robust principal analysis) observations still suffer from a lack

of theoretical guarantees, although they have been used in various recent applications and have exhibited promising empirical performance. In Chapter 2, we attempt to fill this gap. Specifically, we propose a class of convex recovery models (including strongly convex programs) that can be proved to guarantee exact recoveries under certain conditions.

In all of the sparse models for low-rank tensor recovery problems discussed in Chapter 2, the most popular convex relaxations currently being used minimize the sum of the nuclear norms (SNN) of the unfolding matrices of the tensor. In Chapter 3, we show that this approach can be substantially suboptimal: reliably recovering a K-way tensor of length n and Tucker rank r from Gaussian measurements requires $\Omega(rn^{K-1})$ observations. In contrast, a certain (intractable) non-convex formulation needs only $O(rK + nrK)$ observations. We introduce a simple, new convex relaxation, which partially bridges this gap. Our new formulation succeeds with $O(r^{\lfloor \frac{K}{2} \rfloor} n^{\lfloor \frac{K}{2} \rfloor})$ observations. The lower bound for the SNN model follows from our new result on recovering signals with multiple structures (e.g. sparse, low rank), which demonstrates the significant sub-optimality of the common approach of minimizing the sum of individual sparsity inducing norms (e.g. l_1 norm, nuclear norm). Our new formulation for low-rank tensor recovery shows how the sample complexity can be reduced by designing convex regularizers that exploit several structures jointly.

In Chapter 4, we propose and analyze an accelerated linearized Bregman (ALB) method for solving the sparse models discussed in Chapter 2 and 3. This accelerated algorithm is based on the fact that the linearized Bregman (LB) algorithm first proposed by Stanley Osher and his collaborators is equivalent to a gradient descent method applied to a certain dual formulation. We show that the LB method requires $O(1/\varepsilon)$ iterations to obtain an ε-optimal solution and the ALB algorithm

reduces this iteration complexity to $O(1/\sqrt{\varepsilon})$ while requiring almost the same computational effort on each iteration. Numerical results on compressed sensing and matrix completion problems are presented that demonstrate that the ALB method can be significantly faster than the LB method.

In Chapter 5, we extend the arguments of Chapter 4 and apply the linearized Bregman (LB) method to solving linear programming problems. Numerical experiments show that neither the LB method or the accelerated LB method works well on linear programming problems, especially on real data sets. Inspired by the observation that the linearized Bregman, when solving linear programming problems, can be viewed as a sequence of box-constrained quadratic minimizations that are restricted to different supports of the solution variable, we propose new algorithms that combine the Conjugate Gredient/BFGS update with the linearized Bregman method. Numerical experiments on both synthetic and real data sets were conducted, and the new CGLB algorithm not only significantly outperformed the accelerated linearized Bregman proposed in Chapter 4, but was also comparable with the MATLAB solver on small-medium scale real data sets.

Contents

List of Figures

List of Tables

Acknowledgements

First of all, I feel very fortunate to have had the opportunity to work with my advisor, Professor Donald Goldfarb, and I would like to thank him for his invaluable guidance and encouragement along the way to the completion of this dissertation. It has been a wonderful experience working with him, and I could not have imagined having a better advisor and mentor for my Ph.D study. His consistent support, enthusiasm on research and a great sense of humor are unforgettable to me. He has always been a role model, from whom I have learned a lot during my Ph.D. career for being a researcher, an educator, and simply a responsible person.

I also want to express my great appreciation to my co-advisor, Professor John Wright. He is undoubtedly one of the most intelligent people I have ever worked with, and thanks to him, I have a chance to get in touch with the interesting field of low-rank and sparse recovery. John sets an good example for his students with his hard working and perfectionism on research. I have gained a lot of useful knowledge on different fields from our weekly seminar meetings. During our individual meetings, he helped me catch up with the proofs and explained every detail with great patience. I would not have been accomplished my thesis without him.

I would like give my most sincere gratitude to my other dissertation committee members, Professors Garud Iyengar, Daniel Bienstock, Arian Maleki, for their careful reading of my thesis manuscript. They have raised great questions and provided me with constructive comments, which helped improve the thesis. I have leant a lot from the courses convex optimization and optimization II taught by Graud and Dan. They were the very first optimization courses that opened up my interests in this fields, and set me a strong foundation for my research thereafter.

I am very grateful to my collaborators I have had the great pleasure to work with in the past five years. Much of my research has benefited from the collaboration and discussions with Shiqian Ma and Cun Mu, Zhiwei Qin, to whom I would like to specifically express my gratitude. The numerous fruitful discussions that we had have stimulated interesting joint papers. It was a privilege to have worked with such a group of talented people. I also thank the inspirational discussions on research with Shyam Sundar Chandramouli, Jingfei Jia, Yin Lu, Ju Cun, Yuqian Zhang and Henry Kuo.

During the past five years of my school career at Columbia, I have met a lot of people who have become very good friends and an integral part of my life. They have made my life more colorful and the office always feel like home. I have enjoyed close-knit friendship with my fellow Ph.D. students: Yin Lu, Ningyuan Chen, Cun Mu, Linan Yang, Yupeng Chen, Anran Li, Chen Chen, Chun Wang, Shyam Sundar Chardrmouli, Aya Wallwater, Juan Li, Xinyun Chen, Yan Liu, Jing Dong, Zhen Qiu, Krzysztof Choromanski, Carlos Lopez, Antoine Desir, Itai Feigenbaum, Song Hee Kim ...

I am extremely indebted to my parents, Yuanming Huang and Shuru Zhang, whose are always unconditional supportive, both financially or morally, and have helped me to fulfil my educational goals and complete my journey to the Ph.D. degree. Last but not least, I give my most special thanks to my wife, Yang Liu, who is the best gift of my life. She supported me with all her heart throughout these years, and gave me the courage to conquer any obstacle even at the toughest times. Having her by my side is the most wonderful thing that ever happened to me.

Bo Huang

April 1, 2014

To my parents and my wife

Chapter 1

Introduction

1.1 Background and Motivation

Sparse modeling is a rapidly developing topic which arises frequently in areas such as machine learning, data analysis and signal processing. One important application of the sparse modeling is to recover a high-dimensional object from relatively low number of noisy observations, which are main focuses of the Compressed Sensing [14, 22], the Matrix Completion(MC) [13, 34, 68] and the Robust Principal Component Analysis (RPCA) [12]. However, the power of the sparse models is hampered by the data of unprecedented size more and more available in practice. Therefore, it becomes to an appealing topic on better harnessing the convex optimization techniques to take advantage of the underlying "sparsity" structure of the problem.

This thesis focuses on two main aspects of the sparse modeling. From the modeling perspective, it extends convex programming formulations for the matrix completion and the robust principal component analysis problems to the case of tensors, and derives theoretical guarantees

for the exact tensor recovery under the framework of strongly convex programming. Specially, As modern computer technology keeps developing rapidly, multi-dimensional data (elements of which are addressed by more than two indices) is becoming prevalent in many areas such as compute vision [85] and information science [25, 77]. For instance, a color image is a 3-dimensional object with column, row and color modes [66]; a greyscale video is indexed by two spatial variables and one temporal variable; and 3-D face detection uses information with column, row, and depth modes. Tensor-based modeling is a natural choice in these cases because of its capability of capturing the underlying multi-linear structures. Although often residing in extremely high-dimensional spaces, the tensor of interest is frequently low-rank, or approximately so [41]. Consequently, low-rank tensor recovery or estimation is gaining significant attention in many different areas: estimating latent variable graphical models [2], classifying audio [53], mining text [18], processing radar signals [19], to name a few. Lying at the core of high-dimensional data analysis, tensor decomposition serves as a useful tool to reveal when the tensors can be modeled as lying close to a low-dimensional subspace. The two commonly used decompositions are CANDECOMP/PARAFAC(CP) [15, 36] and Tucker decomposition [83]. In particular, based on the Tucker decomposition, a convex surrogate for tensor rank, which here we refer as *sum-of-nuclear-norms (SNN)*, has been proposed in [50] and serves as a more tractable measure of the tensor rank in practical settings. This thesis focuses on the low-rank tensor estimation under partial or corrupted observations. More specifically, an underlying low-rank tensor can be recovered by minimizing the *SNN* over all tensors that obey the given data which may be incomplete or corrupted by arbitrary outliers. This idea, after first being proposed in [50], has been studied in [28, 75, 76, 79, 80], and successfully applied to various problems [27, 42, 44, 46, 73, 74]. Unlike the matrix cases, the re-

covery theory for low-rank tensor estimation problems is far from being well established. In [80], Tomioka et. al. conducted a statistical analysis on the tensor decomposition and provided the first theoretical guarantee for *SNN* minimization. The above result has been significantly enhanced in a recent paper [55], in which it is not only proved that the complexity bound obtained in [80] is tight when using the *SNN* as the convex surrogate, but also proposed a simple improvement that works much better in cases of high-order tensors. Unfortunately, both of the aforementioned results assume Gaussian measurements, while in practice the problem settings are more often similar to matrix completion [13, 34, 68] or robust PCA [12, 89] problems. Mimicking their low-dimensional predecessors, the tensor-based completion and robust PCA formulations have been applied to real applications where their empirical performances have been promising. However, to the best of our knowledge, it is still an open question that what are the theoretical guarantees for exact recovery in tensor completion and tensor RPCA problems.

Once we established a set of sufficient conditions for low-rank tensor recovery problems, the next natural question is that whether we can do better, or more formally, is the complexity bound we achieved optimal under the current convex programming formulation? The answer is "No" when we are considering recovering a low-rank tensor under the standard random Gaussian measurements. In particular, we show that although the bound is tight as long as we are using the SNN as the convex surrogate, it can be substantially suboptimal with respect to the actual "degree of freedom": reliably recovering a K-way tensor of length n and Tucker rank r from Gaussian measurements requires $\Omega(rn^{K-1})$ observations. In contrast, a certain (intractable) non-convex formulation needs only $O(rK + nrK)$ observations. We introduce a simple, new convex relaxation, which partially bridges this gap. Our new formulation succeeds with $O(r^{\lfloor \frac{K}{2} \rfloor} n^{\lfloor \frac{K}{2} \rfloor})$ observations.

The lower bound for the SNN model follows from our new result on recovering signals with multiple structures (e.g. sparse, low rank), which demonstrates the significant sub-optimality of the common approach of minimizing the sum of individual sparsity inducing norms (e.g. l_1 norm, nuclear norm). Our new formulation for low-rank tensor recovery shows how the sample complexity can be reduced by designing convex regularizers that exploit several structures jointly.

From the optimization perspective, efficient algorithms based on Augmented Lagrangian function or splitting techniques have been designed for low-rank tensor recovery problems, e.g., [28, 31, 40, 91]. However, in the matrix settings, instead of solving the original convex programming directly, some algorithms, e.g., [11, 24, 39, 89], have been proposed to solve the *strongly convex* problems by adding a small l_2 perturbation $\tau\|\cdot\|_F^2$ to the original objective. It is well known that the Lagrangian dual for a strongly convex objective is differentiable [71]. Therefore this leads to an unconstraint smooth dual problem which makes a wide class of efficient methods applicable. In [94], the L-BFGS algorithm and gradient methods with line search were studied. The major issue with the strongly convex approach is that τ needs to tend to zero in order to obtain the exact recovery. On the other hand, the empirical convergence speed of most of the aforementioned algorithms depend on τ. In general, a larger τ leads to a faster convergence rate. Fortunately, it has been proved that a finite τ is sufficient for the purpose of low-rank matrix recovery. In the thesis, we extend this conclusion to the case of tensors and provide provable convex programming formulations for tensor completion and tensor RPCA problems.

It is also well known that most of the sparse modeling problems can formulated as the following

general constraint convex programming

$$\min_{x\in\mathbb{R}^n} J(x) \quad \text{s.t.} \quad Ax = b, \tag{1.1.1}$$

where $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$ and $J(x)$ is a norm function (e.g., $\|\cdot\|_1$, $\|\cdot\|_*$ or $\sum_{i=1}^K \|\cdot\|_*$). The linearized Bregman (LB) method was proposed in [93] to solve the basis pursuit problem where $J(x) := \|x\|_1$ in (4.1.1). The method was derived by linearizing the quadratic penalty term in the augmented Lagrangian function that is minimized on each iteration of the so-called Bregman method introduced in [64] while adding a prox term to it. The linearized Bregman method was further analyzed in [8, 10, 92] and applied to solve the matrix completion problem in [8]. The linearized Bregman method depends on a single parameter $\mu > 0$ and, as the analysis in [8, 10] shows, actually solves the problem

$$\min_{x\in\mathbb{R}^n} g_\mu(x) := J(x) + \frac{1}{2\mu}\|x\|_2^2, \quad \text{s.t.} \quad Ax = b, \tag{1.1.2}$$

rather than the problem (4.1.1). It has been shown in this thesis that, for low-rank tensor recovery problem where $J(x)$ is defined as the *Sum of Nuclear Norms (SNN)*, the solution to (4.1.2) is also a solution to problem (4.1.1) as long as μ is chosen large enough. Furthermore, it can be shown that the linearized Bregman method can be viewed as a gradient descent method applied to the Lagrangian dual of problem (4.1.2). This dual problem is an unconstrained optimization problem

of the form

$$\min_{y\in\mathbb{R}^m} \quad G_\mu(y), \tag{1.1.3}$$

where the objective function $G_\mu(y)$ is differentiable since $g_\mu(x)$ is strictly convex (see, e.g., [72]). Motivated by this result, some techniques for speeding up the classical gradient descent method applied to this dual problem such as taking Barzilai-Borwein (BB) steps [3], and incorporating it into a limited memory BFGS (L-BFGS) method [48], were proposed in [92]. Numerical results on the basis pursuit problem (4.1.1) reported in [92] show that the performance of the linearized Bregman method can be greatly improved by using these techniques.

1.2 Preliminaries and Notations

Throughout the paper we denote tensors by boldface Euler script letters, e.g., $\mathcal{X}$. Matrices are denoted by boldface capital letters, e.g., X; vectors are denoted by boldface lowercase letters, e.g., x; and scalars are denoted by lowercase letters, e.g., x. For K-way tensor $\mathcal{X} \in \mathbb{R}^{n_1\times n_2\times\cdots\times n_K}$, its mode-i *fiber* is a n_i-dimensional column vector defined by fixing every index but the ith of $\mathcal{X}$. The mode-i *unfolding* (matricization) of the tensor $\mathcal{X}$ is the matrix denoted by $\mathcal{X}_{(i)} \in \mathbb{R}^{n_i\times\prod_{j\neq i} n_j}$ is obtained by concatenating all the mode-i fibers of $\mathcal{X}$ as column vectors. We denote $n_i^{(1)} = \max\{n_i, \prod_{j\neq i} n_j\}$ and $n_i^{(2)} = \min\{n_i, \prod_{j\neq i} n_j\}$. The vectorization $vec(\mathcal{X})$ is defined as $vec(\mathcal{X}_{(1)})$.

1.2.1 Tensor Norms

Here we extend norm definitions for vectors to tensors. The Frobenius norm of any tensor $\mathcal{X}$ is defined as

$$\|\mathcal{X}\|_F := \|vec(\mathcal{X})\|_2.$$

Similarly, the l_1/l_∞ norm of a tensor $\mathcal{X}$ is defined by its vectorization, i.e.,

$$\|\mathcal{X}\|_{1/\infty} := \|vec(\mathcal{X})\|_{1/\infty}.$$

Tensor-matrix Multiplication: The mode-i product (matrix) product of with tensor $\mathcal{X}$ with matrix A of compatible size is denoted as $\mathcal{Y} = \mathcal{X} \times_i A$, where the ith mode of $\mathcal{Y}$ is

$$\mathcal{Y}_{(i)} := A\mathcal{X}_{(i)}.$$

1.2.2 Linear and Projection Operators

- **[Matrization]** We denote the tensor-to-matrix operator by capital letters in calligraphic fond, e.g., $\mathcal{A}_i$, and $\mathcal{A}_i$ transforms a tensor $\mathcal{X}$ to its mode-i unfolding, i.e.,

  $$\mathcal{A}_i(\mathcal{X}) := \mathcal{X}_{(i)},$$

  and the adjoint of $\mathcal{A}_i$, denoted by $\mathcal{A}_i^*$, is defined by $\mathcal{A}_i^*(\mathcal{X}_{(i)}) = \mathcal{X}$.

- **[Support]** For a tensor $\mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \times \cdots n_K}$, let Ω be any subset of indices, i.e., $\Omega \in [n_1] \times [n_2] \times$

$\cdots \times [n_K]$. Then a projection operator on $\mathcal{P}_\Omega$ is defined by

$$(\mathcal{P}_\Omega[\mathcal{X}])_{i_1,i_2,\cdots i_K} := \begin{cases} \mathcal{X}_{i_1,i_2,\cdots i_K} & (i_1,i_2,\cdots i_K) \in \Omega \\ 0 & \text{otherwise} \end{cases}$$

The projection operator $\mathcal{P}_\Omega$ can be extended to tensor matricization. Specifically, we define the $\mathcal{P}_{\Omega_k}$ be the operator that projects the kth unfolding $\mathcal{X}_{(k)}$ onto the support Ω, i.e.,

$$\mathcal{P}_{\Omega_k}[\mathcal{X}_{(k)}] := (\mathcal{A}_k \mathcal{P}_\Omega \mathcal{A}_k^*)[\mathcal{X}_{(k)}].$$

Also for simplicity, we denote Ω_k to be the support Ω applied to the kth mode when there is no confusions in using this notation; thus

$$\mathcal{P}_{\Omega_k}[\mathcal{X}_{(k)}] = (\mathcal{P}_\Omega[\mathcal{X}])_{(k)}. \tag{1.2.1}$$

1.2.3 Tensor Decomposition

- **[CANDECOMP/PARAFAC(CP)]** The CANDECOMP/PARAFAC(CP) rank of a tensor is defined as

$$\text{rank}_{\text{cp}}(\mathcal{X}) := \min\{r \mid \mathcal{X} = \textstyle\sum_{i=1}^r a_1^{(i)} \circ \cdots \circ a_K^{(i)}\}.$$

Unlike the matrix rank definition, the numerical algebra of tensors is fraught with hardness results [38], and even computing tensor's(CP) rank is NP-hard.

- **[Tucker]** The Tucker decomposition [82] approximates $\mathcal{X}$ as

$$\mathcal{X} = \mathcal{C} \times_1 A_1 \times_2 A_2 \cdots \times_K A_K,$$

where $\mathcal{C} \in \mathbb{R}^{r_1 \times r_2 \times \cdots r_K}$ is called the *core tensor*, and the factor matrices $\{A_i \in \mathbb{R}^{n_i \times r_i}\}$ are column-wise orthonormal. Note that A_i is the matrix of the left singular vectors of the *i*-th unfolding $\mathcal{X}_{(i)}$, and the Tucker rank (also called n-rank) of $\mathcal{X}$ is a K-dimensional vector whose *i*-th entry is the (matrix) rank of the mode-i unfolding $\mathcal{X}_{(i)}$, i.e.,

$$\text{rank}_{tc}(\mathcal{X}) := (\text{rank}(\mathcal{X}_{(1)}), \text{rank}(\mathcal{X}_{(2)}), \cdots \text{rank}(\mathcal{X}_{(K)})).$$

1.2.4 Low-rank Matrix Recovery Models

In this section, we review the existing convex programming models for Matrix Completion (MC) and Robust Principal Component Analysis(RPCA) [12, 89]. Both problems demonstrate that a low rank matrix $X_0 \in \mathbb{R}^{n_1 \times n_2}$ $(n_1 \geq n_2)$ can be exactly recovered from partial or/and corrupted observations via convex programming, under certain *incoherence conditions* on X_0's row and column spaces.

- **[Matrix Incoherence Conditions]** It is well known that exact recovery becomes tractable when the matrix is not in the null space of the sampling operator. This requires the singular vectors of the low-rank component X_0 to be sufficiently spread and not highly correlated with any standard basis. This motivates the following definition.

Definition 1. *Assume that $X_0 \in \mathbb{R}^{n_1 \times n_2}$ is of rank r and has the singular value decomposition $X_0 = U\Sigma V^\top = \sum_{i=1}^{r} \sigma_i u_i v_i^\top$, where $\sigma_i,\ 1 \le i \le r$ are the singular values of X_0, and U and V are the matrices of left and right singular vectors. Then the* ***incoherence conditions*** *with parameter μ are:*

$$\max_i \left\|U^\top e_i\right\|^2 \le \frac{\mu r}{n_1}, \qquad \max_i \left\|V^\top e_i\right\|^2 \le \frac{\mu r}{n_2}, \tag{1.2.2}$$

$$\|UV^*\|_\infty \le \sqrt{\frac{\mu r}{n_1 n_2}}. \tag{1.2.3}$$

where $\{e_i\}$ is the standard matrix basis.

From (1.2.2) and (1.2.3), it follows that

$$\max_i \|\mathcal{P}_U e_i\|_2^2 \le \frac{\mu r}{n_1} \tag{1.2.4}$$

$$\max_i \|\mathcal{P}_V e_i\|_2^2 \le \frac{\mu r}{n_2} \tag{1.2.5}$$

where $\mathcal{P}_U(\mathcal{P}_V)$ is the orthogonal projection onto the column space of $U(V)$. Thus (1.2.4) and (1.2.5) indicate how spread out the singular vectors are with respect to the standard basis. Note that for any subspace, the smallest μ can be is 1, which can be achieved when U is perfectly evenly spread out. The largest possible value for μ is n_1/r when a standard basis vector lies in the subspace spanned by the columns of U and V. A well-conditioned matrix for the recovery is expected to have small incoherence parameter μ. Although other conditions such as the *rank-sparsity incoherence conditions* [17] have also been investigated, the

above conditions (1.2.2)-(1.2.3) are those most commonly used for both matrix completion and RPCA problems.

- **[Matrix Completion (MC)]** In MC problems, we would like to recover the matrix X_0, given that only entries is the support Ω are observed, where $\Omega \subseteq [n_1] \times [n_2]$. Namely, we observe $\mathcal{P}_\Omega[X_0]$ where

$$(\mathcal{P}_\Omega[X_0])_{ij} = \begin{cases} (X_0)_{ij}, & \text{if } (i,j) \in \Omega; \\ 0, & \text{otherwise.} \end{cases} \tag{1.2.6}$$

Clearly, this problem is ill-posed for general X_0. However, the low-rankness of X_0 greatly alleviates the difficulty here. Here one minimizes the nuclear norm $\|\cdot\|_*$, the sum of all the singular values, to recover the original low-rank matrix. As proposed in [13], and later further studied in [34, 68], even when the number of observed entries, i.e. $|\Omega|$, is much less than the ambient dimension $n_1 n_2$, X_0 with small rank r can still be exactly recovered by the following tractable (convex) approach:

$$\begin{aligned} \min \quad & \|X\|_* \\ \text{s.t.} \quad & \mathcal{P}_\Omega[X] = \mathcal{P}_\Omega[X_0]. \end{aligned} \tag{1.2.7}$$

Guarantees for exactly recovering $\mathcal{X}_0$ by solving (1.2.7) were first studied in [13], and later simplified and sharpened in [34, 68].

- **[Robust Principal Component Analysis (RPCA)]** In RPCA problems, the goal is to re-

cover the low-rank matrix X_0 from observations B, which is a superposition of the low-rank component X_0 and a sparse corruption component E_0. In [12], the following convex programming problem was proposed

$$\begin{aligned} \min_{X,E} \quad & \lambda \|X\|_* + \|E\|_1 \\ \text{s.t} \quad & X + E = B. \end{aligned} \tag{1.2.8}$$

It has been shown that when $\lambda = \sqrt{n_1}$, solving (1.2.8) yields the exact recovery of X_0 when it is low-rank and incoherent.

- **[Mixture Model]** Suppose in addition to being grossly corrupted, the data matrix B is observed only partially (say only entries in the support $\Omega \subseteq [n_1] \times [n_2]$ are accessible). The exact recovery of X_0, which is considered as a combination of (1.2.7) and (1.2.8), can be accomplished by solving the following problem:

$$\begin{aligned} \min \quad & \lambda \|X\|_* + \|E\|_1 \\ \text{s.t.} \quad & \mathcal{P}_\Omega[X + E] = B, \end{aligned} \tag{1.2.9}$$

where the corruption matrix E has nonzero entries only on the subset Ω of its $n_1 \times n_2$ entries, i.e., $\mathcal{P}_{\Omega^\perp}[E] = 0$. Model (1.2.9) is equivalent to MC when there is no corruption, i.e., $E = 0$, and it reduces to RPCA when Ω is the entire set of indices. This model has been studied in [12] and [45]. In particular, the bound established in [45] is consistent with the best known results for both MC and RPCA.

A strongly convex formulation is obtained by adding l_2 perturbation terms to the objective function of problem (1.2.9), i.e.,

$$\begin{aligned} \min \quad & \lambda\|X\|_* + \|E\|_1 + \frac{\tau}{2}\|X\|_F^2 + \frac{\tau}{2}\|E\|_F^2 \\ \text{s.t.} \quad & \mathcal{P}_\Omega[X+E] = B. \end{aligned} \tag{1.2.10}$$

Strongly convex models have been studied for compressed sensing, MC and RPCA problems [94–96]. The results are that, instead of vanishing to zero, τ only needs to be reasonably small for exact recovery. Since an extremely small τ often leads to an unsatisfying convergence rate, this feature of τ greatly benefits optimization algorithms that utilize the strong convexity property.

Chapter 2

Provable Low-rank Tensor Recovery

2.1 Introduction

Since, a tensor, generalizes the concept of a matrix, it arises naturally in applications of high-dimensional data analysis. The tensor-based low-rank recovery models including tensor completion [50] and tensor robust PCA [40] problems have been investigated and demonstrated encouraging performances in various applications. Besides the empirical studies, some progresses on their theoretical guarantees have been achieved recently. In [80], Tomioka et. al. conducted the statistical study of tensor decomposition and provided the first (upper)bound on the number of random Gaussian measurements required for exact low-rank tensor recovery. In a more recently work [55], Mu et. al. proved that, under the same settings, the bound obtained in [80] is tight. However, both aforementioned works are based on the constraint of random Gaussian measurements, while a rigorous study on more practical settings such as tensor completion and tensor robust PCA problems

is still left open. In this section, we extend the model (1.2.10) and propose a provable strongly convex programming model for low-rank tensor recovery problems.

2.2 Tensor recovery models

In this section, we review some most commonly used models for low-rank tensor recovery problems.

- **[Convexification for tensor ranks]** The most commonly used definitions for tensor ranks are the CANDECOMP/PARAFAC(CP) rank [15, 36] and the Tucker rank [83]. Many recent applications focus more on the Tucker rank (n-rank) because of its bases on matrix rank. Given all tensors whose corresponding elements match the given the incomplete set of observations, we would like to recovery $\mathcal{X}_0$ by minimizing some combination of the n-vector Tucker rank, i.e.,

$$\textbf{[Completion(non-convex)]} \qquad \min_{\text{w.r.t.}\mathbb{R}^K_+} \text{rank}_{tc}(\mathcal{X}) \qquad \mathcal{P}_\Omega[\mathcal{X}] = \mathcal{P}_\Omega[\mathcal{X}_0] \quad (2.2.1)$$

$$\textbf{[Robust PCA(non-convex)]} \qquad \min_{\text{w.r.t.}\mathbb{R}^K_+} \text{rank}_{tc}(\mathcal{X}) + \|\mathcal{E}\|_0 \quad \mathcal{X} + \mathcal{E} = \mathcal{B}. \quad (2.2.2)$$

To convexify the vector optimization problems (2.2.1)-(2.2.2) which are NP-hard, it is natural to replace the above n-rank's by the weighted sum of nuclear norms. This leads to the

following scalar and convex optimization problems

$$\textbf{[Completion(convex)]} \qquad \min_{\mathcal{X}} \sum_{i=1}^{K} \lambda_i \|\mathcal{X}_{(i)}\|_* \qquad \mathcal{P}_\Omega[\mathcal{X}] = \mathcal{P}_\Omega[\mathcal{X}_0] \tag{2.2.3}$$

$$\textbf{[Robust PCA(convex)]} \qquad \min_{\mathcal{X}} \sum_{i=1}^{K} \lambda_i \|\mathcal{X}_{(i)}\|_* + \|\mathcal{E}\|_1 \quad \mathcal{X} + \mathcal{E} = \mathcal{B}. \tag{2.2.4}$$

The idea of using the term $\sum_{i=1}^{K} \lambda_i \|\mathcal{X}_{(i)}\|_*$, which here we refer to as *sum-of-nuclear-norms (SNN)*, as a convex surrogate for Tucker rank was first proposed in [50]. However, in this work we put possibly different weights to each nuclear norm, while in [50] and similar works that came afterwards, considered only the heuristic of an equal weighted sum of nuclear norms. This means more model parameters in problems (2.2.3) and (2.2.4) to be tuned. Fortunately, we will provide an explicit expression for λ_i which allows for exact recovery and only depends on the dimensions of the targeting tensor $\mathcal{X}$. Moreover, the above *SNN* becomes equal weighted when $\mathcal{X}$ has the same dimension for every mode.

- **[Strongly convex programming]** On the optimization perspective, instead of directly dealing with the original convex programming problem, algorithms, e.g., [11, 24, 39, 89], have been proposed to solve a strongly convex programming problem that approximates it by adding with a small l_2 perturbation $\tau\|\cdot\|_F^2$ being added to the original objective. This is not surprising since in general the strong convexity is a favorable property for deriving faster optimization algorithms. It is well known that the Lagrangian dual to the strongly convex programming is unconstrained as well as differentiable, which makes it suitable to a much broader well established efficient algorithms. The main drawback of this class of problems

sometimes comes from the noise brought by introducing the extra l_2 perturbation. The parameter τ needs to vanish to zero for exact recovery. On the other hand, for most of the aforementioned algorithms, an infinitely small τ will significantly deteriorate the convergence speed. Fortunately, it has been proved that a finite τ is sufficient for the purpose of low-rank matrix recovery, and we show in this paper that the same conclusion also holds for tensor recovery problems.

2.3 Tensor Incoherence Conditions

As in low-rank matrix recovery problems, some incoherence conditions need to be met if recovery is to be possible for tensor-based problems. Hence, we propose a new set of incoherence conditions (2.3.1)-(2.3.2) for a tensor $\mathcal{X}_0$ by extending the matrix incoherence conditions (1.2.2)-(1.2.3) to the unfoldings of $\mathcal{X}_0$ and adding a new "mutual incoherence" condition.

Definition 2. *Suppose that for a tensor $\mathcal{X}_0 \in \mathbb{R}^{n_1 \times n_2 \times \cdots n_K}$, its unfoldings $\{\mathcal{X}_{(i)}\}_{i=1,2\cdots,K}$ have the singular value decompositions*

$$\mathcal{X}_{(i)} = U_i \Sigma_i V_i^\top \quad i = 1, 2, \cdots, K.$$

and let

$$\lambda_i := \sqrt{n_i^{(1)}} \quad \textit{and} \quad \mathcal{T} := \sum_{i=1}^{K} \lambda_i \mathcal{A}_i^* U_i V_i^\top.$$

*Then the **tensor incoherence conditions** with parameter μ are that there exists a mode k such that*

$$\textbf{\textit{[k-mode incoherence]}} \quad \begin{cases} \max_j \|U_k^\top e_j\|^2 \le \frac{\mu r_k}{n^k}, \quad \max_j \|V_k^\top e_j\|^2 \le \frac{\mu r_k}{\prod_{j=1,j\neq k}^K n_j}, \\ \\ \|\lambda_k U_k V_k^\top\|_\infty \le \sqrt{\frac{\mu r_k}{n_k^{(2)}}}, \end{cases} \tag{2.3.1}$$

$$\textbf{\textit{[mutual incoherence]}} \quad \frac{\|\mathcal{T}\|_\infty}{K} \le \sqrt{\frac{\mu r_k}{n_k^{(2)}}}. \tag{2.3.2}$$

where $\{e_i\}$ is the standard matrix basis.

Note that the first two inequalities in (2.3.1) are just the regular matrix incoherence conditions for the low-rank mode (e.g., the k-th unfolding). The second inequality of (2.3.1) is equivalent to

$$\|U_k V_k^\top\|_\infty \le \sqrt{\frac{\mu r_k}{n_k^{(1)} n_k^{(2)}}},$$

since $\lambda_k = \sqrt{n_k^{(1)}}$. Furthermore, if we define $\kappa_i := \frac{r_i}{n_i^{(2)}}$ to be the "rank-saturation" for the ith mode, then from the triangle inequality, it follows that

$$\frac{\|\mathcal{T}\|_\infty}{K} \le \sqrt{\mu\kappa}, \tag{2.3.3}$$

where $\kappa := \max_i\{\kappa_i\}$. Obviously a large κ means that the tensor $\mathcal{X}_0$ while having some mode (the kth) with low (Tucker)rank, also has modes with high ranks, i.e., $\mathcal{X}_0$ is somewhat "unbalanced" with respect to its ranks. To compare the new *mutual incoherence condition* with the original matrix incoherence conditions in (2.3.1), we see that, if $\|\lambda_k U_k V_k^\top\|_\infty \le \sqrt{\frac{\mu r_k}{n_k^{(2)}}}$ holds for some μ,

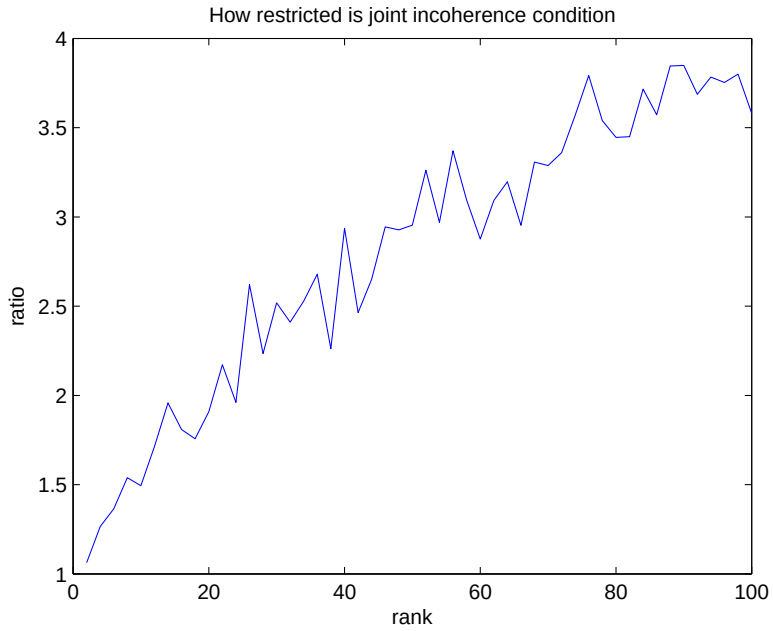


Figure 2.1: The average ratio $\frac{\|\mathcal{T}\|_\infty/K}{\|\lambda_1 U_1 V_1^\top\|_\infty}$ as a function of the rank r for randomly generated 3-way tensors $\mathcal{X} \in \mathbb{R}^{100\times100\times100}$ with Tucker rank $(1,r,r)$. For each rank $r \in [1,100]$, we run 10 independent trials and averaged their output ratios

then the bigger κ is, the harder it is for (2.3.2) to be satisfied. To demonstrate how restrictive (2.3.2) can be, we randomly generated a 3-way tensor $\mathcal{X} \in \mathbb{R}^{100\times100\times100}$ with its Tucker decomposition

$$\mathcal{X} = \mathcal{C} \times_1 A_1 \times_2 A_2 \cdots \times_K A_K,$$

where a core tensor $\mathcal{C} \in \mathbb{R}^{1\times r\times r}$ has entrees generated from i.i.d Gaussian distribution, and each A_i is a random orthogonal matrix. We gradually increased r from 2 to 100, so that while we always had $\kappa_1 = \frac{1}{100}$, κ ranged from $\frac{2}{100}$ to 1. From Figure 2.1, we observe that although r and κ were increased significantly, our mutual incoherence condition (2.3.2) appeared to be much looser than what (2.3.3) suggests, since the ratio $\frac{\|\mathcal{T}\|_\infty/K}{\|\lambda_1 U_1 V_1^\top\|_\infty}$ grew much more slowly than r and κ.

Although (2.3.2) is not that restrictive in general, as Figure 2.1 illustrated, (2.3.2) does char-

acterize a class of tensors on which SNN is a plausible model to use since it favors tensors whose Tucker rank is more balanced. More specifically, for problems where κ is significantly larger than κ_k so that (2.3.2) becomes quite restrictive, SNN minimization is more likely to perform poorly. Indeed, Figure 2.2 illustrates the difference between the SNN model and the Singleton model, which minimizes only the nuclear norm of the low-rank mode unfolding, on recovering an incomplete tensor $\mathcal{X}_0$ under different ranks and levels of observations. We notice from Figure 2.2 that the SNN model outperforms the Singleton model when $r \leq 5$, but does worse than the Singleton model when $r > 5$. Specifically, the performance of the SNN model is very good for small r but deteriorate as r is increased, while the Singleton model usually recovers $\mathcal{X}_0$ when the fraction of the elements of $\mathcal{X}_0$ that are observed is greater than 0.25, regardless of the rank of all other non-low-rank modes. This is not surprising since by minimizing the sum of nuclear norms, we are enforcing a low-rank structure for all modes simultaneously even if this may not be the case for the true solution. Therefore extra conditions are needed to ensure that all the non-low-rank modes are not too far out of line from the well-conditioned low-rank mode when we are minimizing their ranks. In particular, (2.3.2) suggests the average overall incoherence should be on a par with the incoherence of the low-rank (kth) mode as measured by the infinity norm.

2.4 Main Results

In this section, we consider recovering a low-rank tensor $\mathcal{X}_0$ under incomplete and corrupted observations. Let Ω be the set of entries accessible to us. Out of the entire set Ω, a subset $\Lambda \subset \Omega$ of the entries of $\mathcal{X}_0$ are corrupted by $\mathcal{E}_0$, and $\Gamma \subset \Omega$ are locations where data are available and clean.

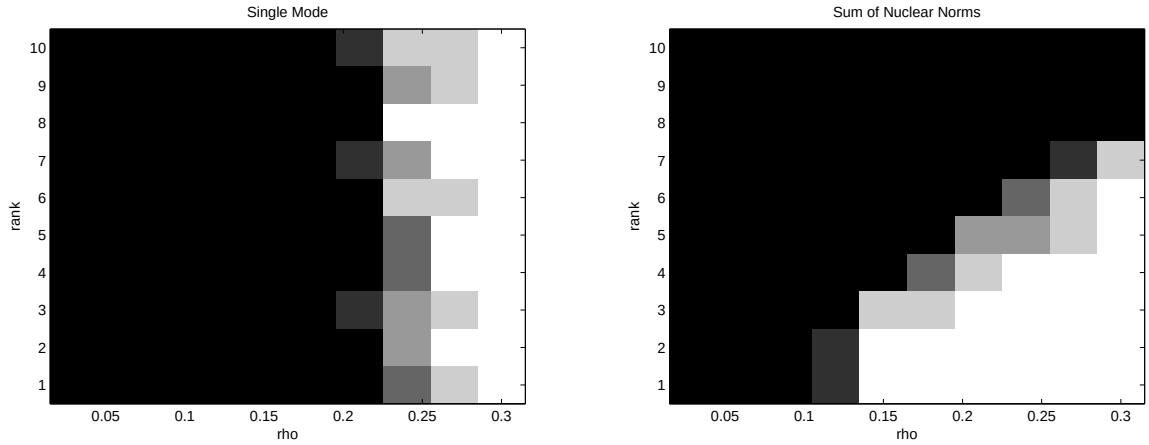


Figure 2.2: A random tensor $\mathcal{X}_0 \in \mathbb{R}^{30\times 30\times 30}$ with (Tucker)rank $(1,r,r)$ was generated. We let r increase from 1 to 10, and the portion of the observed entries, i.e., ρ, to range from 0 to 0.3. For each pair (ρ,r), we ran 5 independent trials and plotted the success rate $k/5$, where k is the number of exact recovery successes, i.e., relative error $< 10^{-3}$. The lighter a region is, the more likely exact recovery can be achieved under the given choice of ρ and r.

As is easily seen, this can be viewed as a combination of the matrix completion and the matrix RPCA, when extended to the case of tensors.

$$\begin{aligned} \min_{\mathcal{X},\mathcal{E}} \quad & f(\mathcal{X},\mathcal{E}) := \sum_{i=1}^{K} \lambda_i \|\mathcal{X}_{(i)}\|_* + \|\mathcal{E}\|_1 + \frac{\tau}{2}\|\mathcal{X}\|_F^2 + \frac{\tau}{2}\|\mathcal{E}\|_F^2 \\ \text{s.t.} \quad & \mathcal{P}_\Omega[\mathcal{X}+\mathcal{E}] = \mathcal{B} \end{aligned} \tag{2.4.1}$$

Note that the entries of $\mathcal{E}$ and $\mathcal{B}$ are nonzero only in support Ω, and by the definition of Γ, we also have

$$\mathcal{P}_\Gamma[\mathcal{E}] = 0, \quad \mathcal{P}_\Gamma[\mathcal{X}] = \mathcal{P}_\Gamma[\mathcal{B}].$$

Our main result giving conditions under which solving (2.4.1) yields the exact recovery of $\mathcal{X}_0$ is contained in the following.

Theorem 1. *Suppose $\mathcal{X}_0$ obeys the same incoherence conditions* (2.3.1)-(2.3.2) *with parameters*

μ, and the support set Ω is uniformly distributed with cardinality $m = \rho n_k^{(1)} n_k^{(2)}$. Also suppose that each observed entry is independently corrupted with probability γ. Then provided that

$$r_k \leq C_r K^{-2} \frac{\rho \mu^{-1} n_k^{(2)}}{\log^2 n_k^{(1)}}, \quad \gamma \leq C_\gamma \tag{2.4.2}$$

and

$$\tau \leq \frac{1}{2n_k^{(1)} n_k^{(2)} (1 + \frac{4}{\rho(1-2C_\gamma)}) \|\mathcal{B}\|_F}, \tag{2.4.3}$$

solving (2.4.1) *with $\lambda_i = \sqrt{n_i^{(1)}}$ yields the exact solution $\mathcal{X}_0$ with probability at least $1 - Cn^{-3}$ for where C, C_r and C_γ are positive numbers.*

2.5 Architecture of the Proof

2.5.1 Sampling Schemes and Model Randomness

Theorem 1 is established based on using a *Uniform sampling scheme without replacement* to choose a set of entrees Ω with cardinality m. However, in order simplify our proofs, it is more convenient as is commonly done to work with other sampling schemes, such as *Bernoulli sampling*. Specifically, in our proofs, we will work with *Bernoulli sampling* with a *Random sign model*.

[Bernoulli sampling] A *Bernoulli sampling scheme* has been used in previous work ([13], [12])

to facilitate the analysis of matrix completion and RPCA problems. For the *Bernoulli model*, we have $\Omega := \{(i,j) : \delta_{ij} = 1\}$, where the δ_{ij}'s are i.i.d Bernoulli variables taking value one with probability ρ and zero with probability $1-\rho$. *Bernoulli sampling* can be written as $\Omega \sim \text{Ber}(\rho)$ for short. Being a proxy for uniform sampling, the probability of failure under Bernoulli sampling with $p = \frac{m}{n_1 \times n_2 \cdots \times n_K}$ closely approximates the probability of failure under uniform sampling.

[Random sign model] A standard Bernoulli model assumes that

$$\begin{cases} \Lambda \sim \text{Ber}(\rho\gamma) \\ \Gamma \sim \text{Ber}((1-\gamma)\rho) \\ \Omega \sim \text{Ber}(\rho), \end{cases}$$

and that the signs of nonzeros entries of $\mathcal{E}_0$ are deterministic. However, it turns out that it is easier to prove Theorem 1 under the stronger assumption that the signs of the nonzeros entries of $\mathcal{E}_0$ are independent symmetric Bernoulli variables. We define two independent random subsets of Ω:

$$\begin{aligned} \Lambda' &\sim \text{Ber}(2\gamma\rho), \\ \Gamma' &\sim \text{Ber}((1-2\gamma)\rho), \end{aligned}$$

It is convenient to think of

$$\mathcal{E}_0 = \mathcal{P}_\Lambda[\mathcal{E}],$$

for some fixed tensor $\mathcal{E}$. Consider now a random sign tensor $\mathcal{W}$ with i.i.d. entries such that for

any index $\mathrm{vec}[i] \in \mathbb{R}^{i_1 \times i_2 \cdots \times i_K}$,

$$P(\mathcal{W}_{\mathrm{vec}[i]} = 1) = P(\mathcal{W}_{\mathrm{vec}[i]} = -1) = \frac{1}{2}.$$

Now $|\mathcal{E}| \circ \mathcal{W}$ has components with symmetric random signs and we define a new "noise" tensor

$$\mathcal{E}_0' := \mathcal{P}_{\Lambda'}[|\mathcal{E}| \circ \mathcal{W}].$$

By the standard derandomization theory (e.g., Theorem 2.3 in [12]), *if the recovery of* $(\mathcal{X}_0, \mathcal{E}_0')$ *is exact with high probability, then it is also exact with at least the same probability for the model with input data* $(\mathcal{X}_0, \mathcal{E}_0)$. Therefore from now on, we can equivalently work with

$$\Lambda \sim \mathrm{Ber}(2\gamma\rho), \quad \Gamma \sim \mathrm{Ber}((1-2\gamma)\rho),$$

the locations of nonzero and zero entries of $\mathcal{E}_0$, respectively, and assume that the nonzero entries of $\mathcal{E}_0$ have symmetric random signs.

2.5.2 Supporting Lemmas

Assume that the ith unfolding $\mathcal{X}_{(i)}$ has the singular value decomposition

$$\mathcal{X}_{(i)} = U_i \Sigma_i V_i^\top \qquad i = 1, 2, \cdots, K. \tag{2.5.1}$$

Define T_i to be the linear space

$$T_i := \left\{ W | W = U_i X^\top + Y V_i^\top \quad \text{for some } X, Y \right\}, \tag{2.5.2}$$

and $T_i^\perp$ to be the orthogonal complement of T_i. The orthogonal projection $\mathcal{P}_{T_i}$ on T_i is given by

$$\mathcal{P}_{T_i}(Z) = \mathcal{P}_{U_i} Z + Z \mathcal{P}_{V_i} - \mathcal{P}_{U_i} Z \mathcal{P}_{V_i}, \tag{2.5.3}$$

and $\mathcal{P}_{T_i^\perp}$ is defined as

$$\mathcal{P}_{T_i^\perp}(Z) = (I - \mathcal{P}_{U_i}) Z (I - \mathcal{P}_{V_i}), \tag{2.5.4}$$

where $\mathcal{P}_{U_i}$ and $\mathcal{P}_{V_i}$ are the orthogonal projections onto U_i and V_i respectively.

Lemma 1. *With the tensor $\mathcal{T}$ defined as in Definition 2, we have, for any mode i,*

$$\mathcal{T}_{(i)} \in T_i,$$

where the subspace T_i is defined in (2.5.2)*.*

Proof. For $\mathcal{X} = \mathcal{C} \times_1 A_1 \times_2 A_2 \cdots \times_K A_K$, let $L_k \in \mathbb{R}^{r_k \times r_k}$ and $R_k \in \mathbb{R}^{\prod_{j \neq k} r_j \times r_k}$ be matrices of the left and right singular vectors of $\mathcal{C}_{(k)}$, the mode k unfolding of $\mathcal{C}$. Then the singular value decomposition of $\mathcal{X}_{(k)}$ obeys

$$\mathcal{X}_{(k)} = (A_k L_k) \Sigma_k R_k^\top \Phi_k^\top, \tag{2.5.5}$$

where $\Sigma_k \in \mathbb{R}^{r_k \times r_k}$ is the matrix whose diagonal elements are the singular values of $\mathcal{C}_{(k)}$ and

$$\Phi_k := A_K \otimes \cdots A_{k+1} \otimes A_{k-1} \cdots \otimes A_1.$$

Therefore the subspace T_k in (2.5.2) corresponding to (2.5.5) is

$$T_k = \left\{ W | W = A_k L_k X^\top + Y R_k^\top \Phi_k^\top \quad \text{for some } X, Y \right\}. \tag{2.5.6}$$

Note that the columns of $A_k L_k$ are orthonormal since those of A_k are and L_k is an orthonormal matrix. On the other hand, $\mathcal{T}$ can be explicitly written as

$$\mathcal{T} = \mathcal{C}_T \times_1 A_1 \times_2 A_2 \cdots \times_K A_K, \tag{2.5.7}$$

where $\mathcal{C}_T := \left(\sum_{i=1}^K \lambda_i \mathcal{A}_i^* L_i R_i^\top \right)$, and its kth unfolding $T_{(k)} = A_k (\mathcal{C}_T)_{(k)} \Phi_k$ is in T_k since we can choose X and Y in (2.5.6) as

$$X = L_k^{-1} (\mathcal{C}_T)_{(k)} \Phi_k, \qquad Y = 0.$$

□

We now state three key inequalities which are crucial for the proof of the main theorem. The first and third inequalities, i.e., (2.5.8) and (2.5.10), can be found in [13] and (2.5.9) can be found in [12]. Note that all three inequalities are applied to the matricization on the kth mode where k is the low-rank mode.

Lemma 2. *Suppose Ω is sampled from the Bernoulli model with parameter ρ, Let $\mathcal{Z} \in \mathbb{R}^{n_1 \times n_2 \cdots, \times n_K}$ and k is the low-rank mode in* (2.3.1)-(2.3.2)*, and Ω_k is the support Ω applied to the kth mode as defined in* (1.2.1)*. Then with the high probability,*

$$\left\| \rho^{-1} \mathcal{P}_{T_k} \mathcal{P}_{\Omega_k} \mathcal{P}_{T_k} - \mathcal{P}_{T_k} \right\| \leq \varepsilon, \tag{2.5.8}$$

and

$$\left\| (\rho^{-1} P_{T_k} \mathcal{P}_{\Omega_k} \mathcal{P}_{T_k} - \mathcal{P}_{T_k}) \mathcal{Z}_{(k)} \right\|_\infty \leq \varepsilon \| \mathcal{P}_{T_k} \mathcal{Z}_{(k)} \|_\infty \tag{2.5.9}$$

provided that $\rho \geq C_1 \varepsilon^{-2} \frac{\mu r_k \log n_k^{(1)}}{n_k^{(2)}}$ for some positive numerical constant C_1;

$$\| (I - \rho^{-1} \mathcal{P}_{\Omega_k}) \mathcal{Z}_{(k)} \| \leq C_2' \sqrt{\frac{n_k^{(1)} \log n_k^{(1)}}{\rho}} \| \mathcal{Z}_{(k)} \|_\infty \tag{2.5.10}$$

for some $C_2' > 0$, provided that $\rho \geq C_2 \frac{\log n_k^{(1)}}{n_k^{(2)}}$ for some small constant $C_2 > 0$.

2.5.3 Dual Certificates

Lemma 3. *If there exists some unfolding $k \in [K]$ such that*

$$\| \frac{1}{(1-2\gamma)\rho} \mathcal{P}_{T_k} \mathcal{P}_{\Gamma_k} \mathcal{P}_{T_k} - \mathcal{P}_{T_k} \| \leq \frac{1}{2},$$

and a matrix $Y \in \mathbb{R}^{n_k \times \prod_{j \neq k} n_j}$ *satisfying*

$$\begin{cases} \left\| \mathcal{P}_{T_k}[Y] - \mathcal{P}_{T_k}\left[\mathcal{S}_{(k)} - \mathcal{T}_{(k)} + \tau(\mathcal{X}_0 - \mathcal{E}_0)_{(k)}\right] \right\|_F \leq \frac{1}{n_k^{(1)} n_k^{(2)}}, \\ \left\| \mathcal{P}_{T_k^\perp}[Y] - \mathcal{P}_{T_k^\perp}\left[\mathcal{S}_{(k)} + \tau(\mathcal{X}_0 - \mathcal{E}_0)_{(k)}\right] \right\| \leq \frac{\lambda_k}{2}, \\ \mathcal{P}_{\Gamma_k^\perp}[Y] = 0, \\ \|Y\|_\infty \leq \frac{1}{2} \end{cases} \tag{2.5.11}$$

where $\lambda_k = \sqrt{\rho n_k^{(1)}}$ *and* $\mathcal{S}_{(k)}$ *is the kth unfolding of*

$$\mathcal{S} := sgn(\mathcal{E}_0),$$

then $(\mathcal{X}_0, \mathcal{E}_0)$ *is the unique solution of* (2.4.1) *when* $n_k^{(1)} n_k^{(2)}$ *is sufficiently large.*

Proof. Consider a feasible perturbation $(\mathcal{X}_0 + \Delta, \mathcal{E}_0 - \mathcal{P}_\Omega[\Delta])$. We now show that the objective value $f(\mathcal{X}_0 + \Delta, \mathcal{E}_0 - \mathcal{P}_\Omega[\Delta])$ is strictly greater than $f(\mathcal{X}_0, \mathcal{E}_0$ unless $\Delta = 0$. Since

$$\begin{cases} \mathcal{A}_i^*[U_i V_i^\top + W_i^0] \in \partial \|\mathcal{A}_i[\mathcal{X}_0]\|_*, \text{ for any } i \in [K] \\ \mathcal{S} + \mathcal{F}^0 \in \partial \|\mathcal{E}_0\|_1, \end{cases}$$

where for each i

$$\mathcal{P}_{T_k}[W_i^0] = 0, \quad \left\|W_i^0\right\| \leq 1$$

$$\mathcal{P}_{\Gamma_k^\perp}[\mathcal{F}^0] = 0, \quad \left\|\mathcal{F}^0\right\|_\infty \leq 1,$$

we have

$$\begin{aligned}
& f(\mathcal{X}_0+\Delta, \mathcal{E}_0 - \mathcal{P}_\Omega[\Delta]) - f(\mathcal{X}_0, \mathcal{E}_0) \\
\geq\ & \left\langle \sum_{i=1}^K \lambda_i \mathcal{A}_i^*[U_i V_i^*] + \sum_{i=1}^K \lambda_i \mathcal{A}_i^*[W_i^0] + \tau\mathcal{X}_0, \Delta \right\rangle - \left\langle \mathcal{S} + \mathcal{F}^0 + \tau\mathcal{E}_0, \mathcal{P}_\Omega[\Delta] \right\rangle \\
=\ & \left\langle \mathcal{T} + \sum_{i=1}^K \lambda_i \mathcal{A}_i^*[W_i^0] + \tau\mathcal{X}_0, \Delta \right\rangle - \left\langle \mathcal{S} + \mathcal{F}^0 + \tau\mathcal{E}_0, \Delta \right\rangle \\
=\ & \lambda_k \left\| \mathcal{P}_{T_k^\perp}[\Delta_{(k)}] \right\|_* + \left\| \mathcal{P}_{\Gamma_k}[\Delta_{(k)}] \right\|_1 + \langle \mathcal{T} - \mathcal{S} + \tau(\mathcal{X}_0 - \mathcal{E}_0), \Delta \rangle \\
=\ & \lambda_k \left\| \mathcal{P}_{T_k^\perp}[\Delta_{(k)}] \right\|_* + \left\| \mathcal{P}_{\Gamma_k}[\Delta_{(k)}] \right\|_1 + \langle Y - \mathcal{S}_{(k)} + \mathcal{T}_{(k)} + \tau(\mathcal{X}_0 - \mathcal{E}_0)_{(k)}, \Delta_{(k)} \rangle - \langle Y, \Delta_{(k)} \rangle \\
=\ & \lambda_k \left\| \mathcal{P}_{T_k^\perp}[\Delta_{(k)}] \right\|_* + \left\| \mathcal{P}_{\Gamma_k}[\Delta_{(k)}] \right\|_1 + \langle \mathcal{P}_{T_k}\left[Y - \mathcal{S}_{(k)} + \mathcal{T}_{(k)} + \tau(\mathcal{X}_0 - \mathcal{E}_0)_{(k)}\right], \mathcal{P}_{T_k}\left[\Delta_{(k)}\right] \rangle \\
& \quad + \langle \mathcal{P}_{T_k^\perp}\left[Y - \mathcal{S}_{(k)} + \tau(\mathcal{X}_0 - \mathcal{E}_0)_{(k)}\right], \mathcal{P}_{T_k^\perp}\left[\Delta_{(k)}\right] \rangle + \langle Y, \mathcal{P}_{\Gamma_k}\left[\Delta_{(k)}\right] \rangle \\
\geq\ & \frac{\lambda_k}{2} \left\| \mathcal{P}_{T_k^\perp}[\Delta_{(k)}] \right\|_* + \frac{1}{2} \left\| \mathcal{P}_{\Gamma_k}[\Delta_{(k)}] \right\|_1 - \frac{1}{n_k^{(1)} n_k^{(2)}} \left\| \mathcal{P}_{T_k}[\Delta_{(k)}] \right\|_F, \qquad (2.5.12)
\end{aligned}$$

where the first inequality follows directly from the the convexity of $\|\cdot\|_*$ and $\|\cdot\|_1$; the second inequality holds as

$$\mathcal{P}_{\Omega^\perp}\left[\mathcal{S} + \mathcal{F}^0 + \tau\mathcal{E}_0\right] = 0;$$

The third equality requires choosing $W_i^0 = 0$ for all $i \neq k$ and picking up W_k^0 and $\mathcal{F}^0$ such that

$$\begin{aligned} \langle \mathcal{A}_k^* W_k^0, \Delta \rangle &= \langle W_k^0, \Delta_{(k)} \rangle = \| \mathcal{P}_{T_k^\perp}[\Delta_{(k)}] \|_* \\ \langle \mathcal{F}^0, \Delta \rangle &= \| \mathcal{P}_\Gamma[\Delta] \|_1 = \| \mathcal{P}_{\Gamma_k}[\Delta_{(k)}] \|_1; \end{aligned}$$

the last inequality is due to (2.5.11), thus

$$\begin{aligned} \langle \mathcal{P}_{T_k} \left[Y - \mathcal{S}_{(k)} + \mathcal{T}_{(k)} + \tau(\mathcal{X}_0 - \mathcal{E}_0)_{(k)} \right], \mathcal{P}_{T_k} \left[\Delta_{(k)} \right] \rangle &\geq -\frac{1}{n_k^{(1)} n_k^{(2)}} \left\| \mathcal{P}_{T_k}[\Delta_{(k)}] \right\|_F \\ \langle \mathcal{P}_{T_k^\perp} \left[Y - \mathcal{S}_{(k)} + \tau(\mathcal{X}_0 - \mathcal{E}_0)_{(k)} \right], \mathcal{P}_{T_k^\perp} \left[\Delta_{(k)} \right] \rangle &\geq -\frac{\lambda_k}{2} \left\| \mathcal{P}_{T_k^\perp}[\Delta_{(k)}] \right\|_* \\ \langle Y, \mathcal{P}_{\Gamma_k} \left[\Delta_{(k)} \right] \rangle &\geq -\frac{1}{2} \left\| \mathcal{P}_{\Gamma_k}[\Delta_{(k)}] \right\|_1 \end{aligned}$$

Recall that we have

$$\| \frac{1}{(1-2\gamma)\rho} \mathcal{P}_{T_k} \mathcal{P}_{\Gamma_k} \mathcal{P}_{T_k} - \mathcal{P}_{T_k} \| \leq \frac{1}{2},$$

which implies $\| \frac{1}{\sqrt{(1-2\gamma)\rho}} \mathcal{P}_{T_k} \mathcal{P}_{\Gamma_k} \| \leq \sqrt{3/2}$, then

$$\begin{aligned} \| \mathcal{P}_{T_k}[\Delta_{(k)}] \|_F &\leq 2 \left\| \frac{1}{(1-2\gamma)\rho} \mathcal{P}_{T_k} \mathcal{P}_{\Gamma_k} \mathcal{P}_{T_k}[\Delta_{(k)}] \right\|_F \\ &\leq 2 \left\| \frac{1}{(1-2\gamma)\rho} \mathcal{P}_{T_k} \mathcal{P}_{\Gamma_k} \mathcal{P}_{T_k^\perp}[\Delta_{(k)}] \right\|_F + 2 \left\| \frac{1}{(1-2\gamma)\rho} \mathcal{P}_{T_k} \mathcal{P}_{\Gamma_k}[\Delta_{(k)}] \right\|_F \\ &\leq \sqrt{\frac{6}{(1-2\gamma)\rho}} \left\| \mathcal{P}_{T_k^\perp}[\Delta_{(k)}] \right\|_F + \sqrt{\frac{6}{(1-2\gamma)\rho}} \left\| \mathcal{P}_{\Gamma_k}[\Delta_{(k)}] \right\|_F \end{aligned} \tag{2.5.13}$$

Substituting (2.5.13) into (2.5.12), we obtain

$$\begin{aligned} & f(\mathcal{X}_0+\Delta,\mathcal{E}_0-\mathcal{P}_\Omega[\Delta])-f(\mathcal{X}_0,\mathcal{E}_0) \\ \geq & \left(\frac{\lambda_k}{2}-\frac{1}{n_k^{(1)}n_k^{(2)}}\sqrt{\frac{6}{(1-2\gamma)\rho}}\right)\left\|\mathcal{P}_{T_k}[\Delta_{(k)}]\right\|_F \\ & +\left(\frac{1}{2}-\frac{1}{n_k^{(1)}n_k^{(2)}}\sqrt{\frac{6}{(1-2\gamma)\rho}}\right)\left\|\mathcal{P}_{\Gamma_k}[\Delta_{(k)}]\right\|_F. \end{aligned} \tag{2.5.14}$$

When $n_k^{(1)}n_k^{(2)}$ is large such that

$$\frac{1}{2}-\frac{1}{n_k^{(1)}n_k^{(2)}}\sqrt{\frac{6}{(1-2\gamma)\rho}}>0,$$

the inequality (2.5.14) holds if and only if $\mathcal{P}_{T_k}[\Delta_{(k)}]=\mathcal{P}_{\Gamma_k}[\Delta_{(k)}]=0$. On the other hand, when ρ is small such that

$$\|\mathcal{P}_{T_k}\mathcal{P}_{\Gamma_k}\|\leq\sqrt{\frac{3(1-2\gamma)\rho}{2}}<1,$$

which implies that $\mathcal{P}_{T_k}\mathcal{P}_{\Gamma_k}$ is injective. As a result, (2.5.14) holds if and only if $\Delta=0$. □

Proof of Theorem 1:

Proof. We apply the *Golfing Scheme* similar to that in [45] to construct the dual certificate Y that

satisfies

$$\begin{cases} \left\|\mathcal{P}_{T_k} Y - \mathcal{P}_{T_k}\left[\mathcal{S}_{(k)} - \mathcal{T}_{(k)}\right]\right\|_F \leq \frac{1}{2n_k^{(1)} n_k^{(2)}} \\ \left\|\mathcal{P}_{T_k^\perp} Y\right\| \leq \frac{\lambda_k}{8} \\ \left\|\mathcal{P}_{T_k^\perp}\left[\mathcal{S}_{(k)}\right]\right\| \leq \frac{\lambda_k}{8} \\ \|Y\|_\infty \leq \frac{1}{2} \end{cases} \tag{2.5.15}$$

and verify the following condition for τ

$$\begin{cases} \tau \cdot \left\|\mathcal{P}_{T_k}\left[(\mathcal{X}_0 - \mathcal{E}_0)_{(k)}\right]\right\|_F \leq \frac{1}{2n_k^{(1)} n_k^{(2)}} \\ \tau \cdot \left\|\mathcal{P}_{T_k^\perp}\left[(\mathcal{X}_0 - \mathcal{E}_0)_{(k)}\right]\right\| \leq \frac{\lambda_k}{4}. \end{cases} \tag{2.5.16}$$

.

[Proof of (2.5.15)**]** We construct Y, which is supported on Γ_k, by gradually increasing the size of Γ_k. Now think of $\Gamma_k \sim \mathrm{Ber}((1-2\gamma)\rho)$ as a union of sets of support Γ^j, i.e., $\Gamma_k = \bigcup_{j=1}^p \Gamma^j$ where $\Gamma^j \sim \mathrm{Ber}(q_j)$. Define $q_1 = q_2 = \frac{(1-2\gamma)\rho}{6}$ and $q_3 = \cdots = q_p = q$, which implies $q \geq C\rho/\log n_k^{(1)}$. Thus we have

$$1-(1-2\gamma)\rho = \left(1 - \frac{(1-2\gamma)\rho}{6}\right)^2 (1-q)^{p-2},$$

where $p = \lfloor 5\log n + 1 \rfloor$. Starting from $Y_0 = 0$, we define Y^L inductively

$$\begin{cases} Z^0 = \mathcal{P}_{T_k}\left[\mathcal{S}_{(k)} - \mathcal{T}_{(k)}\right], \\ Y^j = \sum_{i=1}^{j} q_i^{-1}\mathcal{P}_{\Gamma^i}[Z^{i-1}], \\ Z^j = Z^0 - \mathcal{P}_{T_k}Y^j, \end{cases}$$

which implies that

$$Z^j = \left(\mathcal{P}_{T_k} - q_j^{-1}\mathcal{P}_{T_k}\mathcal{P}_{\Gamma^j}\mathcal{P}_{T_k}\right)[Z^{j-1}].$$

Then it follows from Lemma 2 that

$$\begin{aligned} &\|Z^j\|_F \le \frac{1}{2}\|Z^{j-1}\|_F \\ &\|Z^1\|_\infty \le \frac{1}{2\sqrt{\log n_k^{(1)}}}\|Z^0\|_\infty, \quad \|Z^j\|_\infty \le \frac{1}{2^j \log n_k^{(1)}}\|Z^0\|_\infty \quad \forall j > 1, \\ &\left\|\left(I - q_j^{-1}\mathcal{P}_{\Gamma^j}\right)Z^{j-1}\right\| \le C\sqrt{\frac{n_k^{(1)}\log n_k^{(1)}}{q_j}}\|Z^{j-1}\|_\infty \end{aligned}$$

- We first bound $\left\|Z^0\right\|_F$ and $\left\|Z^0\right\|_\infty$. By the triangle inequality, we have

$$\|Z^0\|_\infty \le \|\mathcal{T}_{(k)}\|_\infty + \|\mathcal{P}_{T_k}[\mathcal{S}_{(k)}]\|_\infty.$$

Recall that for any $(i,j) \in \mathbb{R}^{n_k^{(1)} \times n_k^{(2)}}$, we have

$$\|\mathcal{P}_{T_k}[e_i e_j^\top]\|_\infty \leq \frac{2\mu r_k}{n_k^{(2)}}, \quad \|\mathcal{P}_{T_k}[e_i e_j^\top]\|_F \leq \sqrt{\frac{2\mu r_k}{n_k^{(2)}}}.$$

By Bernstein's inequality, we have

$$\begin{aligned} & \mathbb{P}\left(|\langle \mathcal{P}_{T_k}\left[\mathcal{S}_{(k)}\right], e_i e_j^\top\rangle| \geq t\right) \\ = \; & \mathbb{P}\left(|\langle \mathcal{S}_{(k)}, \mathcal{P}_{T_k}[e_i e_j^\top]\rangle| \geq t\right) \\ \leq \; & 2\exp\left(-\frac{t^2/2}{N + Mt/3}\right), \end{aligned}$$

where

$$N := 2\gamma\rho \cdot \|\mathcal{P}_{T_k} e_i e_j^\top\|_F^2 \leq C\gamma\rho \frac{\mu r_k}{n_k^{(2)}},$$

and

$$M := \left\|\mathcal{P}_{T_k} e_i e_j^\top\right\|_\infty \leq \frac{2\mu r_k}{n_k^{(2)}}.$$

Then with high probability, we have

$$\|\mathcal{P}_{T_k}[\mathcal{S}_{(k)}]\|_\infty \leq C\sqrt{\rho \frac{\mu r_k \log n_k^{(1)}}{n_k^{(2)}}},$$

and from the mutual incoherence condition (2.3.2)

$$\|\mathcal{T}_{(k)}\|_\infty = \|\mathcal{T}\|_\infty \leq K\sqrt{\frac{\mu r_k}{n_k^{(2)}}}$$

Therefore we have

$$\|Z^0\|_\infty \leq CK\sqrt{\frac{\mu r_k \log n_k^{(1)}}{n_k^{(2)}}} \tag{2.5.17}$$

$$\|Z^0\|_F \leq \sqrt{n_k^{(1)} n_k^{(2)}}\|Z^0\|_\infty \leq CK\sqrt{\mu r_k n_k^{(1)} \log n_k^{(1)}} \tag{2.5.18}$$

- Second, we bound $\|\mathcal{P}_{T_k^\perp} Y^p\|$.

$$\begin{aligned}
\|\mathcal{P}_{T_k^\perp} Y^p\| &\leq \sum_j \|q_j^{-1}\mathcal{P}_{T_k^\perp}\mathcal{P}_{\Gamma^j} Z^{j-1}\| \\
&\leq \sum_j \|q_j^{-1}(\mathcal{P}_{\Gamma^j} - I) Z^{j-1}\| \\
&\leq C\sum_j \sqrt{\frac{n_k^{(1)}\log n_k^{(1)}}{q_j}}\|Z^{j-1}\|_\infty \\
&\leq C\sqrt{n_k^{(1)}\log n_k^{(1)}}\left(\sum_{j=3}^{p}\frac{1}{2^{j-1}\log n_k^{(1)}\sqrt{q_j}} + \frac{1}{2\sqrt{\log n_k^{(1)} q_2}} + \frac{1}{\sqrt{q_1}}\right)\|Z^0\|_\infty \\
&\leq CK\sqrt{\frac{n_k^{(1)}\mu r_k\left(\log n_k^{(1)}\right)^2}{n_k^{(2)}\rho}} \\
&\leq C\sqrt{\frac{n_k^{(1)}}{C_\rho}} \\
&\leq \frac{\lambda_k}{8}.
\end{aligned}$$

The last inequality holds when C_ρ is large enough.

- Third, we bound $\left\|\mathcal{P}_{T_k^\perp}[\mathcal{S}_{(k)}]\right\|$. Since $\left\|\mathcal{P}_{T_k^\perp}[\mathcal{S}_{(k)}]\right\| \leq \|\mathcal{S}_{(k)}\|$ and the sign matrix $\mathcal{S}_{(k)} =$

$\mathrm{sgn}(\mathcal{E}_0)$ is distributed as

$$\left(\mathcal{S}_{(k)}\right)_{ij} = \begin{cases} 1, & w.p. \quad \gamma\rho \\ 0, & w.p. \quad 1-2\gamma\rho \\ -1, & w.p. \quad \gamma\rho, \end{cases}$$

standard arguments about the norm of a matrix with i.i.d entries give

$$\|\mathcal{S}_{(k)}\| \leq \frac{\lambda_k}{8},$$

when C_γ is sufficiently small.

- Fourth, we bound $\|Y^p\|_\infty$.

$$\begin{aligned}
\|Y^p\|_\infty &\leq \sum_j \left\| q_j^{-1} \mathcal{P}_{T_k^\perp} \mathcal{P}_{\Gamma^j} Z^{j-1} \right\|_\infty \\
&\leq \sum_j \left\| q_j^{-1} \left(\mathcal{P}_{\Gamma^j} - I\right) Z^{j-1} \right\|_\infty \\
&\leq C \sum_j \frac{1}{q_j} \|Z^{j-1}\|_\infty \\
&\leq \left(\sum_{j=3}^{p} \frac{1}{2^{j-1} \log n_k^{(1)} \sqrt{q_j}} + \frac{1}{2\sqrt{\log n_k^{(1)} q_2}} + \frac{1}{\sqrt{q_1}} \right) \|Z^0\|_\infty \\
&\leq K \sqrt{\frac{\mu r \log n_k^{(1)}}{n_k^{(2)} \rho}} \\
&\leq \sqrt{\frac{1}{C_\rho \log n_k^{(1)}}} \\
&\leq 1/4
\end{aligned}$$

provided C_ρ is sufficiently large.

- Last, we show that

$$\left\|\mathcal{P}_{T_k}Y^p - \mathcal{P}_{T_k}\left[\mathcal{S}_{(k)} - \mathcal{T}_{(k)}\right]\right\|_F \leq \frac{1}{2n_k^{(1)}n_k^{(2)}}.$$

Since $\mathcal{P}_{T_k}Y^p - \mathcal{P}_{T_k}\left[\mathcal{S}_{(k)} - \mathcal{T}_{(k)}\right] = \mathcal{P}_{T_k}Y^p - Z^0 = -Z^p$, we only need to bound $\|Z^p\|_F$, i.e.,

$$\begin{aligned}\left\|\mathcal{P}_{T_k}Y^p - \mathcal{P}_{T_k}\left[\mathcal{S}_{(k)} - \mathcal{T}_{(k)}\right]\right\|_F &= \|Z^p\|_F \\ &\leq C\left(\frac{1}{2}\right)^p \|Z^0\|_F \\ &\leq C\left(n_k^{(1)}\right)^{-5}\sqrt{\mu r_k n_k^{(1)}\log n_k^{(1)}} \\ &\leq \frac{1}{2n_k^{(1)}n_k^{(2)}}.\end{aligned}$$

[Proof of (2.5.16)**]** To establish the condition for τ under which (2.5.16) holds, it suffices to bound $\|\mathcal{X}_0 - \mathcal{E}_0\|_F$ since

$$\left\|\mathcal{P}_{T_k}\left[(\mathcal{X}_0 - \mathcal{E}_0)_{(k)}\right]\right\|_F \leq \|\mathcal{X}_0 - \mathcal{E}_0\|_F,$$
$$\left\|\mathcal{P}_{T_k^\perp}\left[(\mathcal{X}_0 - \mathcal{E}_0)_{(k)}\right]\right\| \leq \|\mathcal{X}_0 - \mathcal{E}_0\|_F,$$

and we require

$$\tau < \frac{1}{2n_k^{(1)}n_k^{(2)}\|\mathcal{X}_0 - \mathcal{E}_0\|_F}.$$

We observe that

$$\|\mathcal{X}_0 - \mathcal{E}_0\|_F = \|(I + \mathcal{P}_\Omega)[\mathcal{X}_0] - \mathcal{B}\|_F \leq 2\|\mathcal{X}_0\|_F + \|\mathcal{B}\|_F. \tag{2.5.19}$$

Since

$$\|\mathcal{P}_{T_k} - \frac{1}{(1-2\gamma)\rho}\mathcal{P}_{T_k}\mathcal{P}_{\Gamma_k}\mathcal{P}_{T_k}\| \leq \frac{1}{2},$$

we have

$$\|\mathcal{X}_0\|_F \leq \frac{2}{(1-2\gamma)\rho}\left\|\mathcal{P}_{T_k}\mathcal{P}_{\Gamma_k}\mathcal{P}_{T_k}[(\mathcal{X}_0)_{(k)}]\right\|_F = \frac{2}{(1-2\gamma)\rho}\left\|\mathcal{P}_{T_k}\mathcal{P}_{\Gamma_k}[\mathcal{B}_{(k)}]\right\|_F \leq \frac{2}{(1-2\gamma)\rho}\|\mathcal{B}\|_F. \tag{2.5.20}$$

Therefore we have

$$\|\mathcal{X}_0 - \mathcal{E}_0\|_F \leq \left(1 + \frac{4}{(1-2\gamma)\rho}\right)\|\mathcal{B}\|_F,$$

and it suffices to have

$$\tau \leq \frac{1}{2n_k^{(1)}n_k^{(2)}\left(1+\frac{4}{p_0(1-\gamma_s)}\right)\|\mathcal{B}\|_F}, \tag{2.5.21}$$

□

Chapter 3

Square Deal: Lower Bounds and Improved Relaxations for Tensor Recovery

3.1 Introduction

In the previous chapter, we consider the problem of recovering a K-way tensor $\mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_K}$ from linear measurements $z = \mathcal{G}[\mathcal{X}] \in \mathbb{R}^m$. Typically, $m \ll N = \prod_{i=1}^{K} n_i$, and so the problem of recovering $\mathcal{X}$ from z is *ill-posed*. In the past few years, tremendous progress has been made in understanding how to exploit structural assumptions such as sparsity for vectors [14] or low-rankness for matrices [69] to develop computationally tractable methods for tackling ill-posed inverse problems. In many situations, convex optimization can estimate a structured object from near-minimal sets of observations [1, 16, 56]. For example, an $n \times n$ matrix of rank r can, with high probability, be exactly recovered from Cnr generic linear measurements, by minimizing the nuclear norm $\|X\|_* = \sum_i \sigma_i(X)$. Since a rank r matrix has $r(2n-r)$ degrees of freedom, this is nearly optimal. In

contrast, we show in this chapter that the correct generalization of these results to low-rank tensors is not true. The theoretical guarantees we achieved in the previous chapter for tensor recovery problems can be quite sub-optimal in general. For ease of stating results, suppose $n_1 = \cdots = n_K = n$, and $\text{rank}_{\text{tc}}(\mathcal{X}) \preceq (r, r, \cdots, r)$. Let $\mathfrak{T}_r$ denote the set of all such tensors. We will consider the problem of estimating an element $\mathcal{X}_0$ of $\mathfrak{T}_r$ from Gaussian measurements $\mathcal{G}$ (i.e., $z_i = \langle \mathcal{G}_i, \mathcal{X} \rangle$, where $\mathcal{G}_i$ has i.i.d. standard normal entries). To describe a generic tensor in $\mathfrak{T}_r$, we need at most $r^K + rnK$ parameters. We also show that a certain nonconvex strategy can recover all $\mathcal{X} \in \mathfrak{T}_r$ exactly when $m > (2r)^K + 2nrK$. In contrast, the best known theoretical guarantee for SNN minimization, due to Tomioka et al. [80], shows that $\mathcal{X}_0 \in \mathfrak{T}_r$ can be recovered (or accurately estimated) from Gaussian measurements $\mathcal{G}$, provided $m = \Omega(rn^{K-1})$. It can be shown that this number of measurements is also *necessary*: accurate recovery is unlikely unless $m = \Omega(rn^{K-1})$. Thus, there is a substantial gap between an ideal nonconvex approach and the best known tractable surrogate. Lastly, we introduce a simple alternative, which we call the *square reshaping* model, which reduces the required number of measurements to $O(r^{\lfloor K/2 \rfloor} n^{\lceil K/2 \rceil})$. For $K > 3$, this improves by a multiplicative factor polynomial in n.

3.2 Bounds for Non-Convex and Convex Recovery Models

3.2.1 Non-convex models

In this section, we introduce a non-convex model for tensor recovery, and show that it recovers low-rank tensors from near-minimal number of measurements. While our nonconvex formulation is computationally intractable, it gives a baseline for evaluating tractable (convex) approaches.

For a tensor of low Tucker rank, the matrix unfolding along each mode is low-rank. Suppose we observe $\mathcal{G}[\mathcal{X}_0] \in \mathbb{R}^m$. We would like to attempt to recover $\mathcal{X}_0$ by minimizing some combination of the ranks of the unfoldings, over all tensors $\mathcal{X}$ that are consistent with our observations. This suggests a *vector optimization* problem:

$$\min_{(\text{w.r.t. } \mathbb{R}^K_+)} \text{rank}_{\text{tc}}(\mathcal{X}) \qquad \text{s.t.} \quad \mathcal{G}[\mathcal{X}] = \mathcal{G}[\mathcal{X}_0]. \tag{3.2.1}$$

The recovery performance of program (3.2.1) depends heavily on the properties of $\mathcal{G}$. Suppose (3.2.1) fails to recover $\mathcal{X}_0 \in \mathfrak{T}_r$. Then there exists another $\mathcal{X}' \in \mathfrak{T}_r$ such that $\mathcal{G}[\mathcal{X}'] = \mathcal{G}[\mathcal{X}_0]$. To guarantee that (3.2.1) recovers *any* $\mathcal{X}_0 \in \mathfrak{T}_r$, a necessary and sufficient condition is that $\mathcal{G}$ is injective on $\mathfrak{T}_r$, which is implied by the condition $\text{null}(\mathcal{G}) \cap \mathfrak{T}_{2r} = \{0\}$. So, if $\text{null}(\mathcal{G}) \cap \mathfrak{T}_{2r} = \{0\}$, (3.2.1) will recover any $\mathcal{X}_0 \in \mathfrak{T}_r$. We expect this to occur when the number of measurements significantly exceeds the number of intrinsic degrees of freedom of a generic element of $\mathfrak{T}_r$, which is $O(r^K + nrK)$. The following theorem shows that when m is approximately twice this number, with probability one, $\mathcal{G}$ is injective on $\mathfrak{T}_r$:

Theorem 2. *Whenever* $m \geq (2r)^K + 2nrK + 1$, *with probability one,* $\text{null}(\mathcal{G}) \cap \mathfrak{T}_{2\text{r}} = \{0\}$, *and hence* (3.2.1) *recovers every* $\mathcal{X}_0 \in \mathfrak{T}_r$.

3.2.2 Convex models

[Sum of Nuclear Norms]

Since the nonconvex problem (3.2.1) is NP-hard for general $\mathcal{G}$, it is tempting to seek a convex surrogate. We have discussed the SNN convexification for the Tucker rank in the previous chapter, and it leads to the following convex recovery model:

$$\min_{\mathcal{X}} \quad \sum_{i=1}^{K} \lambda_i \|\mathcal{X}_{(i)}\|_* \quad \text{s.t.} \quad \mathcal{G}[\mathcal{X}] = \mathcal{G}[\mathcal{X}_0], \tag{3.2.2}$$

The optimization (3.2.2) was first introduced by [50] and has been used successfully in applications in imaging [27, 42, 44, 46, 73, 74]. Similar convex relaxations have been considered in a number of theoretical and algorithmic works [28, 75, 76, 79, 80]. It is not too surprising, then, that (3.2.2) provably recovers the underlying tensor $\mathcal{X}_0$, when the number of measurements m is sufficiently large. The following is a (simplified) corollary of results of Tomioka et. al. [79][1]:

Corollary 3 (of [79], Theorem 3)**.** *Suppose that $\mathcal{X}_0$ has Tucker rank $(r,\ldots,r)$, and $m \geq Crn^{K-1}$. With high probability, $\mathcal{X}_0$ is an optimal solution to* (3.2.2)*, with each $\lambda_i = 1$. Here, C is numerical.*

This result shows that there *is* a range in which (3.2.2) succeeds: loosely, when we undersample by at most a factor of $m/N \sim r/n$. However, the number of observations $m \sim rn^{K-1}$ is significantly larger than the number of degrees of freedom in $\mathcal{X}_0$, which is on the order of $r^K + nrK$. Is it possible to prove a better bound for this model? Unfortunately, we show that in general $O(rn^{K-1})$ measurements are also *necessary* for reliable recovery using (3.2.2):

[1] Tomioka et. al. also show noise stability when $m = \Omega(rn^{K-1})$ and give extensions to the case where the $\text{rank}_{tc}\mathcal{X}_0 = (r_1,\ldots,r_K)$ differs from mode to mode.

Theorem 4. *Let $\mathcal{X}_0 \in \mathfrak{T}_r$ be nonzero. Set $\kappa = \min_i \left\{ \left\| (\mathcal{X}_0)_{(i)} \right\|_*^2 / \|\mathcal{X}_0\|_F^2 \right\} \times n^{K-1}$. Then if the number of measurements $m \leq \kappa - 2$, $\mathcal{X}_0$ is not the unique solution to* (3.2.2)*, with probability at least $1 - 4\exp(-\frac{(\kappa-m-2)^2}{16(\kappa-2)})$. Moreover, there exists $\mathcal{X}_0 \in \mathfrak{T}_r$ for which $\kappa = rn^{K-1}$.*

This implies that Corollary 3 (and other results of [79]) is essentially tight. Unfortunately, it has negative implications for the efficacy of the sum of nuclear norms in (3.2.2): although a generic element $\mathcal{X}_0$ can be described using at most $r^K + nrK$ real numbers, we require $\Omega(rn^{K-1})$ observations to recover it using (3.2.2). Theorem 4 is a direct consequence of a much more general principle underlying multi-structured recovery, which is elaborated next. After that, in the next section, we demonstrate that for low-rank tensor recovery, better convexifying schemes are available.

[Square Deal]

The number of measurements promised by Corollary 3 and Theorem 4 is actually the same (up to constants) as the number of measurements required to recover a tensor $\mathcal{X}_0$ which is low-rank along just one mode. Since matrix nuclear norm minimization correctly recovers a $n_1 \times n_2$ matrix of rank r when $m \geq Cr(n_1 + n_2)$ [16], solving

$$\text{minimize } \|\mathcal{X}_{(1)}\|_* \quad \text{subject to} \quad \mathcal{G}[\mathcal{X}] = \mathcal{G}[\mathcal{X}_0] \tag{3.2.3}$$

also recovers $\mathcal{X}_0$ w.h.p. when $m \geq Crn^{K-1}$.

This suggests a more mundane explanation for the difficulty with (3.2.2): the term rn^{K-1} comes from the need to reconstruct the right singular vectors of the $n \times n^{K-1}$ matrix $\mathcal{X}_{(1)}$. If we had some

way of matricizing a tensor that *produced a more balanced (square) matrix* and also *preserved the low-rank property*, we could remedy this effect, and reduce the overall sampling requirement. In fact, this is possible when the order K of $\mathcal{X}_0$ is four or larger.

For $A \in \mathbb{R}^{m_1 \times n_1}$, and integers m_2 and n_2 satisfying $m_1 n_1 = m_2 n_2$, the reshaping operator $\text{reshape}(A, m_2, n_2)$ returns a $m_2 \times n_2$ matrix whose elements are taken columnwise from A. This operator rearranges elements in A and leads to a matrix of different shape. In the following, we reshape matrix $\mathcal{X}_{(1)}$ to a more square matrix while preserving the low-rank property. Let $\mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_K}$. Select $j \in [K] := \{1, 2, \cdots, K\}$. Then we define matrix $\mathcal{X}_{[j]}$ as [2]

$$\mathcal{X}_{[j]} = \text{reshape}\Big(\mathcal{X}_{(1)}, \prod_{i=1}^{j} n_i, \prod_{i=j+1}^{K} n_i\Big). \tag{3.2.4}$$

We can view $\mathcal{X}_{[j]}$ as a natural generalization of the standard tensor matricization. When $j = 1$, $\mathcal{X}_{[j]}$ is nothing but $\mathcal{X}_{(1)}$. However, when some $j > 1$ is selected, $\mathcal{X}_{[j]}$ could become a more balanced matrix. This reshaping also preserves some of the algebraic structures of $\mathcal{X}$. In particular, we will see that if $\mathcal{X}$ is a low-rank tensor (in either the CP or Tucker sense), $\mathcal{X}_{[j]}$ will be a low-rank matrix.

Lemma 4. *(1) If $\mathcal{X}$ has CP decomposition $\mathcal{X} = \sum_{i=1}^{r} \lambda_i a_i^{(1)} \circ a_i^{(2)} \circ \cdots \circ a_i^{(K)}$, then*

$$\mathcal{X}_{[j]} = \sum_{i=1}^{r} \lambda_i (a_i^{(j)} \otimes \cdots \otimes a_i^{(1)}) \circ (a_i^{(K)} \otimes \cdots \otimes a_i^{(j+1)}).$$

[2] You can also think of (3.2.4) as embedding tensor $\mathcal{X}$ into the matrix $\mathcal{X}_{[j]}$ as follows: $\mathcal{X}_{i_1, i_2, \cdots, i_K} = \big(\mathcal{X}_{[j]}\big)_{a,b}$, where

$$\begin{aligned} a &= 1 + \sum_{m=1}^{j} \left((i_m - 1) \prod_{l=1}^{m-1} n_l \right) \\ b &= 1 + \sum_{m=j+1}^{K} \left((i_m - 1) \prod_{l=j+1}^{m-1} n_l \right). \end{aligned}$$

(2) If $\mathcal{X}$ has Tucker decomposition[3] $\mathcal{X} = \mathcal{C} \times_1 U_1 \times_2 U_2 \times_3 \cdots \times_K U_K$*, then*

$$\mathcal{X}_{[j]} = (U_j \otimes \cdots \otimes U_1)\, \mathcal{C}_{[j]}\, (U_K \otimes \cdots \otimes U_{j+1})^*.$$

Using Lemma 4 and the fact that $\operatorname{rank}(A \otimes B) = \operatorname{rank}(A)\operatorname{rank}(B)$, we obtain:

Lemma 5. *Let* $\operatorname{rank}_{tc}\mathcal{X} = (r_1, r_2, \cdots, r_K)$*, and* $\operatorname{rank}_{\text{cp}}(\mathcal{X}) = r_{\text{cp}}$*. Then* $\operatorname{rank}(\mathcal{X}_{[j]}) \le r_{\text{cp}}$*, and* $\operatorname{rank}(\mathcal{X}_{[j]}) \le \min\Big\{ \prod_{i=1}^{j} r_i, \prod_{i=j+1}^{K} r_i \Big\}$.

Thus, $\mathcal{X}_{[j]}$ is not only more balanced but also maintains the low-rank property of tensor $\mathcal{X}$, which motivates us to recover $\mathcal{X}_0$ by solving

$$\text{minimize} \quad \big\|\mathcal{X}_{[j]}\big\|_* \quad \text{subject to} \quad \mathcal{G}[\mathcal{X}] = \mathcal{G}[\mathcal{X}_0]. \tag{3.2.5}$$

Using Lemma 5 and [16], we can prove that this relaxation exactly recovers $\mathcal{X}_0$, when the number of measurements is sufficiently large:

Theorem 5. *Consider a K-way tensor with the same length (say n) along each mode. (1) If $\mathcal{X}_0$ has CP rank r, using* (3.2.5) *with $j = \lceil \frac{K}{2} \rceil$, $m \ge Crn^{\lceil \frac{K}{2} \rceil}$ is sufficient to recover $\mathcal{X}_0$ with high probability. (2) If $\mathcal{X}_0$ has Tucker rank $(r, r, \cdots, r)$, using* (3.2.5) *with $j = \lceil \frac{K}{2} \rceil$, $m \ge Cr^{\lfloor \frac{K}{2} \rfloor} n^{\lceil \frac{K}{2} \rceil}$ is sufficient to recover $\mathcal{X}_0$ with high probability.*

The number of measurements $O(r^{\lfloor \frac{K}{2} \rfloor} n^{\lceil \frac{K}{2} \rceil})$ required to recover $\mathcal{X}$ with square reshaping (3.2.5), is always within a constant of the number $O(rn^{K-1})$ with the sum-of-nuclear-norms model, and is

[3] The *mode-i (matrix) product* $\mathcal{A} \times_i B$ of tensor $\mathcal{A}$ with matrix B of compatible size is the tensor $\mathcal{C}$ such that $\mathcal{C}_{(i)} = B\mathcal{A}_{(i)}$.

significantly smaller when r is small and $K \geq 4$. E.g., we obtain an improvement of a multiplicative factor of $n^{\lfloor K/2 \rfloor - 1}$ when r is a constant. This is a significant improvement.

Our square reshaping can be slightly generalized to group any j modes (say modes $i_1, i_2, \cdots, i_j$) together rather than the first j rows. Denote $I = \{i_1, i_2, \cdots, i_j\}$ and $\mathcal{J} = [K] \backslash I = \{i_{j+1}, i_{j+2}, \cdots, i_K\}$. Then the embedded matrix $\mathcal{X}_I \in \mathbb{R}^{\prod_{k=1}^{j} n_{i_k} \times \prod_{k=j+1}^{K} n_{i_k}}$ can be defined similarly as in (3.2.4) but with a relabeling preprocessing. For $1 \leq k \leq K$, we relabel mode k as the original mode i_k. Denote the relabeled tensor as $\widehat{\mathcal{X}}$. Then we can define

$$\mathcal{X}_I := \widehat{\mathcal{X}}_{[j]} = \text{reshape}\Big(\widehat{\mathcal{X}}_{(1)}, \prod_{k=1}^{j} n_{i_k}, \prod_{k=j+1}^{K} n_{i_k}\Big).$$

Lemma 5 and Theorem 5 can be easily modified. As suggested by Theorem 5 (after modification), in practice, we would like to set I that minimizes the quantity,

$$\text{rank}(\mathcal{X}_I) \cdot \max\{\prod_{k=1}^{j} n_{i_k}, \prod_{k=j+1}^{K} n_{i_k}\}. \tag{3.2.6}$$

For tensors with different lengths or ranks, the comparison between sum-of-nuclear-norms and our square reshaping becomes more subtle. It is possible to construct examples for which the square reshaping model does not have an advantage over the SNN model, even for $K > 3$. Nevertheless, for a large class of tensors, our square reshaping is capable of reducing the number of generic measurements required by SNN model, both in theory and in numerical experiments.

3.3 Numerical Experiments

We corroborate the improvement of square reshaping with numerical experiments on *tensor completion* for both noise-free case (synthetic data) and noisy case (real data). Tensor completion attempts to reconstruct the (approximately) low-rank tensor $\mathcal{X}_0$ from a subset Ω of its entries. By imposing appropriate incoherence conditions (and modifying slightly arguments in [33]), it is possible to prove exact/stable recovery guarantees for both our square deal formulation and the SNN model for tensor completion. However, unlike the recovery problem under Gaussian random measurements, due to the lack of sharp upper bounds, we have no proof that our square reshaping model is guaranteed to outperform the SNN model here. Nonetheless, numerical results below indicate clearly the advantage of our square approach, which much complements our theoretical results established in previous sections.

3.3.1 Simulation

We generate a 4-way tensor $\mathcal{X}_0 \in \mathbb{R}^{n\times n\times n\times n}$ as $\mathcal{X}_0 = \mathcal{C}_0 \times_1 U_1 \times_2 U_2 \times_3 U_3 \times_4 U_4$, where the core tensor $\mathcal{C}_0 \in \mathbb{R}^{1\times 1\times 2\times 2}$ has i.i.d. standard Gaussian entries, and matrices U_1, $U_2 \in \mathbb{R}^{n\times 1}$ and matrices U_3, $U_4 \in \mathbb{R}^{n\times 2}$, satisfying $U_i^* U_i = I$, are drawn uniformly at random. The observed entries are chosen uniformly with ratio ρ. We compare the recovery performances between

$$\min_{\mathcal{X}} \ \sum_{i=1}^{K} \|\mathcal{X}_{(i)}\|_* \qquad \text{s.t.} \ \ \mathcal{P}_\Omega[\mathcal{X}] = \mathcal{P}_\Omega[\mathcal{X}_0], \tag{3.3.1}$$

$$\min_{\mathcal{X}} \ \left\|\mathcal{X}_{\{1,2\}}\right\|_* \qquad \text{s.t.} \ \ \mathcal{P}_\Omega[\mathcal{X}] = \mathcal{P}_\Omega[\mathcal{X}_0]. \tag{3.3.2}$$

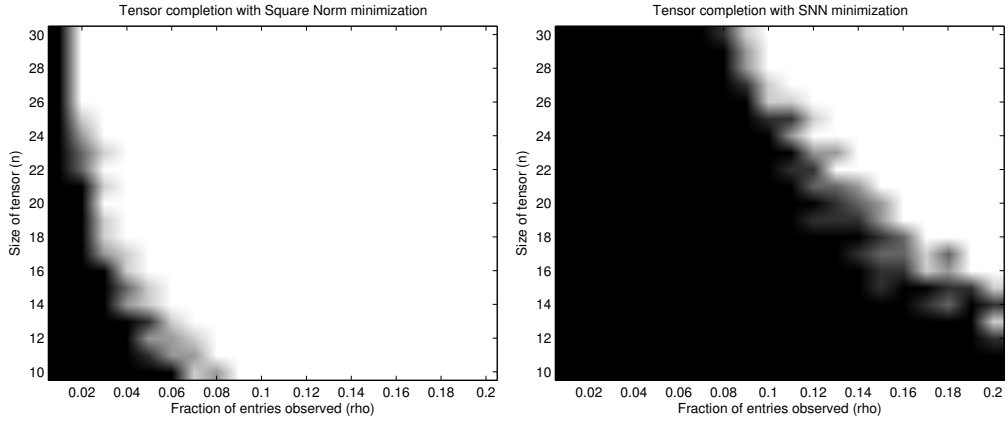


Figure 3.1: **Tensor completion.** The colormap indicates the fraction of correct recovery, which increases with brightness from certain failure (black) to certian success (white). **Left:** square reshaping model. **Right:** SNN model.

We increase the problem size n from 10 to 30 with increment 1, and the observation ratio ρ from 0.01 to 0.2 with increment 0.01. For each (ρ, n)-pair, we simulate 5 test instances and declare a trial to be successful if the recovered $\mathcal{X}^\star$ satisfies $\|\mathcal{X}^\star - \mathcal{X}_0\|_F / \|\mathcal{X}_0\|_F \leq 10^{-2}$.

The optimization problems are solved using efficient first-order methods. Since (3.3.2) is equivalent to standard matrix completion, we use the Augmented Lagrangian Method (ALM) proposed in [47] to solve it. For the sum of nuclear norms minimization (3.3.1), we implement the accelerated linearized Bregman algorithm [39] to solve it (which we will discuss in the appendix).

Figure 3.1 plots the fraction of correct recovery for each pair (black $= 0\%$ and white $= 100\%$). Clearly much larger white region is produced by square norm, which empirically suggests that (3.3.2) outperforms (3.3.1) for tensor completion problem.

3.3.2 Video Completion

Color videos can be naturally represented as 4-mode tensors (length×width×channels×frames). In this part, we compare the performances of our square formulation and SNN model on video data completion from randomly missing pixels. We consider three video datasets here:

1. The first video, referred to as the Ocean video is of size $112 \times 160 \times 3 \times 32$. It records the movements of ocean and has been used in [50] to demonstrate the efficacy of the SNN model.

2. The second video, referred to as the Face video is of size $96 \times 65 \times 3 \times 994$. It is a YOUTUBE video that records the face of a lady aging from young to old.

3. The third video, referred to as the Escalator video is of size $96 \times 128 \times 3 \times 30$. It records an escalator in the airport with passengers travelling along the escalator.

For our square reshaping, we set $I = \{1,4\}$ for the Ocean and the Escalator videos, and set $I = \{1,2\}$ for the Face video, to form the embedded matrix $\mathcal{X}_I$. Due to the existence of noise in real data, as in [50], we would solve the norm regularized least square problems:

$$\min_{\mathcal{X}} \quad \frac{1}{2}\|\mathcal{P}_\Omega[\mathcal{X}] - \mathcal{D}\|_F^2 + \sum_{i=1}^{4} \lambda_i \left\|\mathcal{X}_{(i)}\right\|_* \tag{3.3.3}$$

$$\min_{\mathcal{X}} \quad \frac{1}{2}\|\mathcal{P}_\Omega[\mathcal{X}] - \mathcal{D}\|_F^2 + \lambda \|\mathcal{X}_I\|_*, \tag{3.3.4}$$

where $\mathcal{D} = \mathcal{P}_\Omega[\mathcal{X}_0]$ is the observed tensor, $\lambda_i \geq 0$ and $\lambda \geq 0$ are tuning parameters. Since the purpose of our experiment is to compare the efficacy between SNN and square reshaping, to make the comparison fair and meaningful, we should tune those parameters as optimally as possible.

That is not an easy task, especially for SNN model (3.3.3), which involves four tuning parameters. As a remedy, here we solve the nuclear norm constrained optimization problems. Specifically, we compare the recovery performances between

$$\min_{\mathcal{X}} \quad \tfrac{1}{2}\|\mathcal{P}_\Omega[\mathcal{X}] - \mathcal{D}\|_F^2 \quad \text{s.t.} \left\|\mathcal{X}_{(i)}\right\|_* \leq \beta_i, \forall i \in [4], \tag{3.3.5}$$

$$\min_{\mathcal{X}} \quad \tfrac{1}{2}\|\mathcal{P}_\Omega[\mathcal{X}] - \mathcal{D}\|_F^2 \quad \text{s.t.} \|\mathcal{X}_I\|_* \leq \beta. \tag{3.3.6}$$

In our experiments, we set β_i and β to their oracle values, i.e. $\beta_i = \left\|(\mathcal{X}_0)_{(i)}\right\|_*$ and $\beta = \|(\mathcal{X}_0)_I\|_*$. By doing that, the solutions to (3.3.5) and (3.3.6) can be respectively regarded as the best achievable results from (3.3.3) and (3.3.4) in terms of recovering $\mathcal{X}_0$.

Figures 3.2 and 3.3 display the results of recovery using (3.3.5) and (3.3.6). For the Ocean and the Face videos, our square reshaping clearly outperforms the SNN model. We expect the benefits of using our square formulation will be much magnified in multi-spectral video data, where the number of channels could be much larger than 3. In such data, the video tensor tends to be low rank in both the wavelength and the temporal modes. Thus we can group these two modes to form our low-rank matrix $\mathcal{X}_I$. When the technique of taking multi-spectral data becomes mature in the future, we believe our square reshaping model will be more useful and more significant for the completion task.

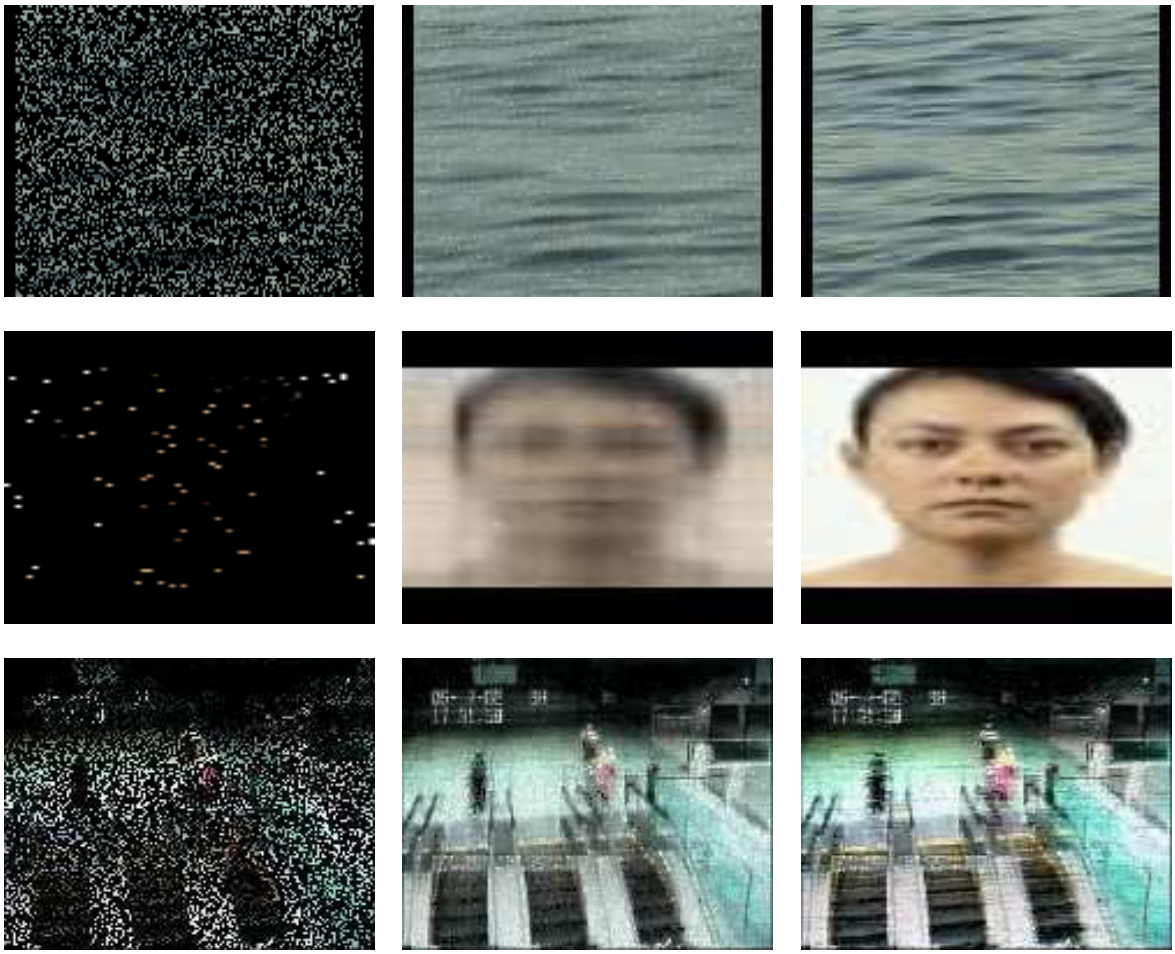
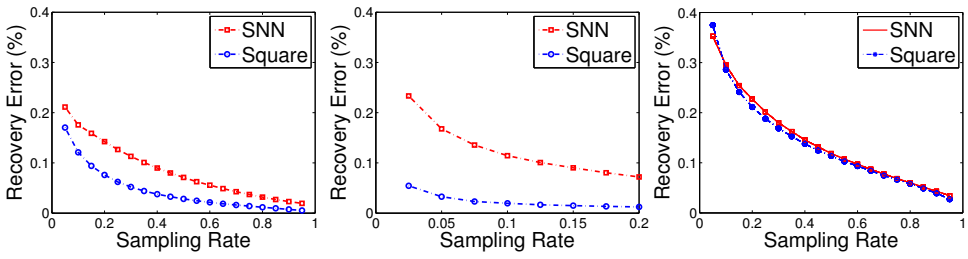


Figure 3.2: Sample snapshots from our datasets: Ocean, Face, Escalator. **Left**: sampled video (20 % for the Ocean video, 2% for the face video and 30% for the Escalator video). **Middle**: video recovered by SNN model. **Right**: video recovered by square reshaping.



Ocean Face Escalator

Figure 3.3: **Video Completion.** (relative) recovery error vs. sampling rate, for the three videos in Figure 3.2.

Chapter 4

Accelerated Linearized Bregman Method for Sparse Modeling problems

4.1 Introduction

It is well known that many the sparse modeling problems can formulated and solved as the following convex program

$$\min_{x\in\mathbb{R}^n} J(x) \quad \text{s.t.} \quad Ax = b, \tag{4.1.1}$$

where $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$ and $J(x)$ is a norm function (e.g., $\|\cdot\|_1$, $\|\cdot\|_*$ or $\sum_{i=1}^{K}\|\cdot\|_*$). The linearized Bregman (LB) method was proposed in [93] to solve the basis pursuit problem where $J(x) := \|x\|_1$ in (4.1.1). The method was derived by linearizing the quadratic penalty term in the augmented Lagrangian function that is minimized on each iteration of the so-called Bregman

method introduced in [64] while adding a prox term to it. The linearized Bregman method was further analyzed in [8, 10, 92] and applied to solve the matrix completion problem in [8]. The linearized Bregman method depends on a single parameter $\mu > 0$ and, as the analysis in [8, 10] shows, actually solves the problem

$$\min_{x\in\mathbb{R}^n} g_\mu(x) := J(x) + \frac{1}{2\mu}\|x\|_2^2, \quad \text{s.t.} \quad Ax = b, \tag{4.1.2}$$

rather than the problem (4.1.1). It has been shown in this thesis that, for low-rank tensor recovery problem where $J(x)$ is defined as the *Sum of Nuclear Norms (SNN),* the solution to (4.1.2) is also a solution to problem (4.1.1) as long as μ is chosen large enough. Furthermore, it can be shown that the linearized Bregman method can be viewed as a gradient descent method applied to the Lagrangian dual of problem (4.1.2). This dual problem is an unconstrained optimization problem of the form

$$\min_{y\in\mathbb{R}^m} \quad G_\mu(y), \tag{4.1.3}$$

where the objective function $G_\mu(y)$ is differentiable since $g_\mu(x)$ is strictly convex (see, e.g., [72]). Motivated by this result, some techniques for speeding up the classical gradient descent method applied to this dual problem such as taking Barzilai-Borwein (BB) steps [3], and incorporating it into a limited memory BFGS (L-BFGS) method [48], were proposed in [92]. Numerical results on the basis pursuit problem (4.1.1) reported in [92] show that the performance of the linearized Bregman method can be greatly improved by using these techniques.

Our starting point is also motivated by the equivalence between applying the linearized Bregman method to (4.1.1) and solving the Lagrangian dual problem (4.1.3) by the gradient descent method. Since the gradient of $G_\mu(y)$ can be shown to be Lipschitz continuous, it is well-known that the classical gradient descent method with a properly chosen step size will obtain an ε-optimal solution to (4.1.3) (i.e., an approximate solution y^k such that $G_\mu(y^k) - G_\mu(y^*) \leq \varepsilon$) in $O(1/\varepsilon)$ iterations. In [58], Nesterov proposed a technique for accelerating the gradient descent method for solving problem of the form (4.1.3) (see, also, [59]), and proved that using this accelerated method, the number of iterations needed to obtain an ε-optimal solution is reduced to $O(1/\sqrt{\varepsilon})$ with a negligible change in the work required at each iteration. Nesterov also proved that the $O(1/\sqrt{\varepsilon})$ complexity bound is the best bound that one can get if one uses only the first-order information. Based on the above discussion, we propose an accelerated linearized Bregman (ALB) method for solving (4.1.2) which is equivalent to an accelerated gradient descent method for solving the Lagrangian dual (4.1.3) of (4.1.2). As a by-product, we show that the basic and the accelerated linearized Bregman methods require $O(1/\varepsilon)$ and $O(1/\sqrt{\varepsilon})$ iterations, respectively, to obtain an ε-optimal solution with respect to the Lagrangian for (4.1.2).

4.2 Bregman and Linearized Bregman Methods

The Bregman method was introduced to the image processing community by Osher et al. in [64] for solving the total-variation (TV) based image restoration problems. The Bregman distance [7]

with respect to convex function $J(\cdot)$ between points u and v is defined as

$$D_J^p(u,v) := J(u) - J(v) - \langle p, u-v\rangle, \tag{4.2.1}$$

where $p \in \partial J(v)$, the subdifferential of J at v. The Bregman method for solving (4.1.1) is given below as Algorithm 4.1. Note that the updating formula for p^k (Step 4 in Algorithm 4.1) is based on the optimality conditions of Step 3 in Algorithm 4.1:

$$0 \in \partial J(x^{k+1}) - p^k + A^\top(Ax^{k+1} - b).$$

This leads to

$$p^{k+1} = p^k - A^\top(Ax^{k+1} - b).$$

It was shown in [64,93] that the Bregman method (Algorithm 4.1) converges to a solution of (4.1.1) in a finite number of steps.

Algorithm 4.1 Original Bregman Iterative Method

1: Input: $x^0 = p^0 = 0$.
2: **for** $k = 0, 1, \cdots$ **do**
3: $\quad x^{k+1} = \arg\min_x D_J^{p^k}(x, x^k) + \frac{1}{2}\|Ax - b\|^2$;
4: $\quad p^{k+1} = p^k - A^\top(Ax^{k+1} - b)$;
5: **end for**

It is worth noting that for solving (4.1.1), the Bregman method is equivalent to the augmented Lagrangian method [37, 67, 70] in the following sense.

Theorem 6. *The sequences $\{x^k\}$ generated by Algorithm 4.1 and by the augmented Lagrangian method, which computes for $k = 0, 1, \cdots$*

$$\begin{cases} x^{k+1} & := & \arg\min_x J(x) - \langle \lambda^k, Ax - b \rangle + \frac{1}{2}\|Ax - b\|^2 \\ \lambda^{k+1} & := & \lambda^k - (Ax^{k+1} - b) \end{cases} \tag{4.2.2}$$

starting from $\lambda^0 = 0$ are exactly the same.

Proof. From Step 4 of Algorithm 4.1 and the fact that $p^0 = 0$, it follows that

$$p^k = -\sum_{j=1}^{k} A^\top (Ax^j - b).$$

From the second equation in (4.2.2) and using $\lambda^0 = 0$, we get $\lambda^k = -\sum_{j=1}^{k}(Ax^j - b)$. Thus, $p^k = A^\top \lambda^k$ for all k. Hence it is easy to see that Step 3 of Algorithm 4.1 is exactly the same as the first equation in (4.2.2) and that the x^{k+1} computed in Algorithm 1 and (4.2.2) are exactly the same. Therefore, the sequences $\{x^k\}$ generated by both algorithms are exactly the same. □

Note that for $J(x) := \alpha\|x\|_1$, Step 3 of Algorithm 4.1 reduces to an ℓ_1-regularized problem:

$$\min_x \quad \alpha\|x\|_1 - \langle p^k, x \rangle + \frac{1}{2}\|Ax - b\|^2. \tag{4.2.3}$$

Although there are many algorithms for solving the subproblem (4.2.3) such as FPC [35], SPGL1 [84], FISTA [4] etc., it often takes them many iterations to do so. The linearized Bregman method was proposed in [93], and used in [9, 10, 65] to overcome this difficulty. The linearized Bregman method replaces the quadratic term $\frac{1}{2}\|Ax - b\|^2$ in the objective function that is minimized

in Step 3 of Algorithm 4.1 by its linearization $\langle A^\top(Ax^k - b), x\rangle$ plus a proximal term $\frac{1}{2\mu}\|x - x^k\|^2$. Consequently the updating formula for p^k is changed since the optimality conditions for this minimization step become:

$$0 \in \partial J(x^{k+1}) - p^k + A^\top(Ax^k - b) + \frac{1}{\mu}(x^{k+1} - x^k).$$

In Algorithm 4.2 below we present a slightly generalized version of the original linearized Bregman method that includes an additional parameter τ that corresponds to the length of a gradient step in a dual problem.

Algorithm 4.2 Linearized Bregman Method

1: Input: $x^0 = p^0 = 0$, $\mu > 0$ and $\tau > 0$.
2: **for** $k = 0, 1, \cdots$ **do**
3: $\quad x^{k+1} = \arg\min_x D_J^{p^k}(x, x^k) + \tau\langle A^\top(Ax^k - b), x\rangle + \frac{1}{2\mu}\|x - x^k\|^2$;
4: $\quad p^{k+1} = p^k - \tau A^\top(Ax^k - b) - \frac{1}{\mu}(x^{k+1} - x^k)$;
5: **end for**

In [92], it is shown that when $\mu\|A\|^2 < 2$, where $\|A\|$ denotes the largest singular value of A, the iterates of the linearized Bregman method (Algorithm 2 with $\tau = 1$) converge to the solution of the following regularized version of problem (4.1.1):

$$\min_x \quad J(x) + \frac{1}{2\mu}\|x\|^2 \quad \text{s.t.} \quad Ax = b. \tag{4.2.4}$$

We prove in Theorem 7 below an analogous result for Algorithm 2 for a range of values of τ. However, we first prove, as in [92], that the linearized Bregman method (Algorithm 2) is equivalent

to a gradient descent method

$$y^{k+1} := y^k - \tau \nabla G_\mu(y^k) \tag{4.2.5}$$

applied to the Lagrangian dual

$$\max_y \min_w \{J(w) + \frac{1}{2\mu}\|w\|^2 - \langle y, Aw - b\rangle\}$$

of (4.2.4), which we express as the following equivalent minimization problem:

$$\min_y \quad G_\mu(y) := -\{J(w^*) + \frac{1}{2\mu}\|w^*\|^2 - \langle y, Aw^* - b\rangle\}, \tag{4.2.6}$$

where

$$w^* := \arg\min_w \{J(w) + \frac{1}{2\mu}\|w\|^2 - \langle y, Aw - b\rangle\}.$$

To show that $G_\mu(y)$ is continuously differentiable, we rewrite $G_\mu(y)$ as

$$G_\mu(y) = -\Phi_\mu(\mu A^\top y) + \frac{\mu}{2}\|A^\top y\|^2 - b^\top y,$$

where

$$\Phi_\mu(v) \equiv \min_w \{J(w) + \frac{1}{2\mu}\|w - v\|^2\}$$

is strictly convex and continuously differentiable with gradient $\nabla\Phi_\mu(v) = \frac{v-\hat{w}}{\mu}$, and $\hat{w} = \arg\min_w\{J(w) + \frac{1}{2\mu}\|w - v\|^2\}$ (e.g., see Proposition 4.1 in [5]). From this it follows that $\nabla G_\mu(y) = Aw^* - b$. Hence the gradient method (4.2.5) corresponds to Algorithm 4.3 below.

Algorithm 4.3 Linearized Bregman Method (Equivalent Form)

1: Input: $\mu > 0$, $\tau > 0$ and $y^0 = \tau b$.
2: **for** $k = 0, 1, \cdots$ **do**
3: $\quad w^{k+1} := \arg\min_w \{J(w) + \frac{1}{2\mu}\|w\|^2 - \langle y^k, Aw - b\rangle\}$;
4: $\quad y^{k+1} := y^k - \tau(Aw^{k+1} - b)$.;
5: **end for**

Lemma 6 and Theorem 7 below generalize Theorem 2.1 in [92] by allowing a step length choice in the gradient step (4.2.5) and show that Algorithms 4.2 and 4.3 are equivalent. Our proof closely follows the proof of Theorem 2.1 in [92].

Lemma 6. *x^{k+1} computed by Algorithm 4.2 equals w^{k+1} computed by Algorithm 4.3 if and only if*

$$A^\top y^k = p^k - \tau A^\top (Ax^k - b) + \frac{1}{\mu} x^k. \tag{4.2.7}$$

Proof. By comparing Step 3 in Algorithms 4.2 and 4.3, it is obvious that w^{k+1} is equal to x^{k+1} if and only if (4.2.7) holds. □

Theorem 7. *The sequences $\{x^k\}$ and $\{w^k\}$ generated by Algorithms 4.2 and 4.3 are the same.*

Proof. We prove by induction that equation (4.2.7) holds for all $k \geq 0$. Note that (4.2.7) holds for $k = 0$ since $p^0 = x^0 = 0$ and $y^0 = \tau b$. Now let us assume that (4.2.7) holds for all $0 \leq k \leq n-1$; thus by Lemma 6 $w^{k+1} = x^{k+1}$ for all $0 \leq k \leq n-1$. By iterating Step 4 in Algorithm 4.3 we get

$$y^n = y^{n-1} - \tau(Aw^n - b) = -\sum_{j=0}^{n} \tau(Ax^j - b). \tag{4.2.8}$$

By iterating Step 4 in Algorithm 4.2 we get

$$p^n = -\sum_{j=0}^{n-1} \tau A^\top (Ax^j - b) - \frac{1}{\mu} x^n,$$

which implies that

$$p^n - \tau A^\top (Ax^n - b) + \frac{1}{\mu} x^n = -\sum_{j=0}^{k} \tau A^\top (Ax^j - b) = A^\top y^n,$$

where the last equality follows from (4.2.8); thus by induction (4.2.7) holds for all $k \geq 0$, which implies by Lemma 6 that $x^k = w^k$ for all $k \geq 0$. □

Before analyzing Algorithms 4.2 and 4.3, we note that by defining $v^k = A^\top y^k$ and algebraically manipulating the last two terms in the objective function in Step 3 in Algorithm 4.3, Steps 3 and 4 in that algorithm can be replaced by

$$\begin{cases} w^{k+1} & := & \arg\min_w J(w) + \frac{1}{2\mu}\|w - \mu v^k\|^2 \\ v^{k+1} & := & v^k - \tau A^\top (Aw^{k+1} - b) \end{cases} \tag{4.2.9}$$

if we set $v^0 = \tau A^\top b$. Because Algorithms 4.2 and 4.3 are equivalent, convergence results for the gradient descent method can be applied to both of them. Thus we have the following convergence result.

Theorem 8. *Let $J(w) \equiv \|w\|_1$. Then $G_\mu(y)$ in the dual problem* (4.2.6) *is continuously differentiable and its gradient is Lipschitz continuous with the Lipschitz constant $L \leq \mu\|A\|^2$. Conse-*

quently, if the step length $\tau < \frac{2}{\mu\|A\|^2}$*, the sequences* $\{x^k\}$ *and* $\{w^k\}$ *generated by Algorithms 4.2 and 4.3 converge to the optimal solution of* (4.2.4)*.*

Proof. When $J(x) = \|x\|_1$, w^{k+1} in (4.2.9) reduces to

$$w^{k+1} = \mu \cdot \text{shrink}(v^k, 1),$$

where the ℓ_1 shrinkage operator is defined as

$$\text{shrink}(z, \alpha) := \text{sgn}(z) \circ \max\{|z| - \alpha, 0\}, \forall z \in \mathbb{R}^n, \alpha > 0. \tag{4.2.10}$$

$G_\mu(y)$ is continuously differentiable since $g_\mu(x)$ is strictly convex. Since for any point y, $\nabla G_\mu(y) = Aw - b$, where $w = \mu \cdot \text{shrink}(A^\top y, 1)$, it follows from the fact that the shrinkage operator is non-expansive, i.e.,

$$\|\text{shrink}(s, \alpha) - \text{shrink}(t, \alpha)\| \leq \|s - t\|, \ \forall s, t, \alpha$$

that

$$\begin{aligned}
\|\nabla G_\mu(y^1) - \nabla G_\mu(y^2)\| &= \|\mu \cdot A \cdot \text{shrink}(A^\top y^1, 1) - \mu \cdot A \cdot \text{shrink}(A^\top y^2, 1)\| \\
&\leq \mu \cdot \|A\| \cdot \|A^\top (y^1 - y^2)\| \\
&\leq \mu \|A\|^2 \|y^1 - y^2\|,
\end{aligned}$$

for any two points y^1 and y^2. Thus the Lipschitz constant L of $\nabla G_\mu(\cdot)$ is bounded above by $\mu\|A\|^2$.

When $\tau < \frac{2}{\mu\|A\|^2}$, we have $\tau L < 2$ and thus $|1 - \tau L| < 1$. It then follows that the gradient descent

method $y^{k+1} = y^k - \tau \nabla G_\mu(y^k)$ converges and therefore Algorithms 4.2 and 4.3 converge to x_μ^*, the optimal solution of (4.2.4). □

Before developing an accelerated version of the LB algorithm in the next section. We would like to comment on the similarities and differences between the LB method and Nesterov's composite gradient method [61] and the ISTA method [4] applied to problem (4.1.1) and related problems. The latter algorithms iterate Step 3 in the LB method (Algorithm 4.2) with $p^k = 0$, and never compute or update the subgradient vector p^k. More importantly, their methods solve the unconstrained problem

$$\min_{x \in \mathbb{R}^n} \quad \|x\|_1 + \frac{1}{2\mu}\|Ax - b\|^2.$$

Hence, while these methods and the LB method both linearize the quadratic term $\|Ax-b\|^2$ while handling the nonsmooth term $\|x\|_1$ directly, they are very different.

Similar remarks apply to the accelerated LB method presented in the next section and fast versions of ISTA and Nesterov's composite gradient method.

Remark 1. *The Algorithm 4.2 and 4.3 can be viewed as the primal and dual implementations for the linearized Bregman. The variables x and p are primal variable while y is the dual variable.*

4.3 The Accelerated Linearized Bregman Algorithm

Based on Theorem 7, i.e., the equivalence between the linearized Bregman method and the gradient descent method, we can accelerate the linearized Bregman method by techniques used to accelerate the classical gradient descent method. In [92], Yin considered several techniques such as line

search, BB step and L-BFGS, to accelerate the linearized Bregman method. Here we consider the acceleration technique proposed by Nesterov in [58, 59]. This technique accelerates the classical gradient descent method in the sense that it reduces the iteration complexity significantly without increasing the per-iteration computational effort. For the unconstrained minimization problem (4.1.3), Nesterov's accelerated gradient method replaces the gradient descent method (4.2.5) by the following iterative scheme:

$$\begin{cases} x^{k+1} & := \quad y^k - \tau \nabla G_\mu(y^k) \\ y^{k+1} & := \quad \alpha_k x^{k+1} + (1-\alpha_k) x^k, \end{cases} \tag{4.3.1}$$

where the scalars α_k are specially chosen weighting parameters. A typical choice for α_k is $\alpha_k = \frac{3}{k+2}$. If τ is chosen so that $\tau \leq 1/L$, where L is the Lipschitz constant for $\nabla G_\mu(\cdot)$, Nesterov's accelerated gradient method (4.3.1) obtains an ε-optimal solution of (4.1.3) in $O(1/\sqrt{\varepsilon})$ iterations, while the classical gradient method (4.2.5) takes $O(1/\varepsilon)$ iterations. Moreover, the per-iteration complexities of (4.2.5) and (4.3.1) are almost the same since computing the gradient $\nabla G_\mu(\cdot)$ usually dominates the computational cost in each iteration. Nesterov's acceleration technique has been studied and extended by many others for nonsmooth minimization problems and variational inequalities, e.g., see [4, 29, 30, 32, 57, 60, 61, 81].

Our accelerated linearized Bregman method is given below as Algorithm A.1. The main difference between it and the basic linearized Bregman method (Algorithm 4.2) is that the latter uses the previous iterate x^k and subgradient p^k to compute the new iterate x^{k+1}, while Algorithm A.1 uses extrapolations $\tilde{x}^k$ and $\tilde{p}^k$ that are computed as linear combinations of the two previous iterates

and subgradients, respectively. Carefully choosing the sequence of weighting parameters $\{\alpha_k\}$ guarantees an improved rate of convergence.

Algorithm 4.4 Accelerated Linearized Bregman Method

Input: $x^0 = \tilde{x}^0 = \tilde{p}^0 = p^0 = 0$, $\mu > 0$, $\tau > 0$.
for $k = 0, 1, \cdots$ **do**
  $x^{k+1} = \arg\min_x D_J^{\tilde{p}^k}(x, \tilde{x}^k) + \tau\langle A^\top(A\tilde{x}^k - b), x\rangle + \frac{1}{2\mu}\|x - \tilde{x}^k\|^2$;
  $p^{k+1} = \tilde{p}^k - \tau A^\top(A\tilde{x}^k - b) - \frac{1}{\mu}(x^{k+1} - \tilde{x}^k)$;
  $\tilde{x}^{k+1} = \alpha_k x^{k+1} + (1-\alpha_k)x^k$;
  $\tilde{p}^{k+1} = \alpha_k p^{k+1} + (1-\alpha_k)p^k$.
end for

In the following, we first establish the equivalence between the accelerated linearized Bregman method and the corresponding accelerated gradient descent method (4.3.1), which we give explicitly as (4.3.2) below applied to the dual problem (4.2.6). Based on this, we then present complexity results for both basic and accelerated linearized Bregman methods. Not surprisingly, the accelerated linearized Bregman method improves the iteration complexity from $O(1/\varepsilon)$ to $O(1/\sqrt{\varepsilon})$.

Theorem 9. *The accelerated linearized Bregman method (Algorithm A.1) is equivalent to the accelerated dual gradient descent method* (4.3.2) *starting from* $\tilde{y}^0 = y^0 = \tau b$*:*

$$\begin{cases} w^{k+1} &:= \arg\min J(w) + \frac{1}{2\mu}\|w\|^2 - \langle \tilde{y}^k, Aw - b\rangle \\ y^{k+1} &:= \tilde{y}^k - \tau(Aw^{k+1} - b) \\ \tilde{y}^{k+1} &:= \alpha_k y^{k+1} + (1-\alpha_k)y^k. \end{cases} \tag{4.3.2}$$

More specifically, the sequence $\{x^k\}$ *generated by Algorithm A.1 is exactly the same as the sequence* $\{w^k\}$ *generated by* (4.3.2)*.*

Proof. Note that the Step 3 of Algorithm A.1 is equivalent to

$$x^{k+1} := \arg\min J(x) - \langle \tilde{p}^k, x\rangle + \tau\langle A^\top(A\tilde{x}^k - b), x\rangle + \frac{1}{2\mu}\|x - \tilde{x}^k\|^2. \tag{4.3.3}$$

Comparing (4.3.3) with the first equation in (4.3.2), it is easy to see that $x^{k+1} = w^{k+1}$ if and only if

$$A^\top \tilde{y}^k = \tilde{p}^k + \tau A^\top(b - A\tilde{x}^k) + \frac{1}{\mu}\tilde{x}^k. \tag{4.3.4}$$

We will prove (4.3.4) in the following by induction. Note that (4.3.4) holds for $k = 0$ since $\tilde{y}^0 = \tau b$ and $\tilde{x}^0 = \tilde{p}^0 = 0$. As a result, we have $x^1 = w^1$. By defining $w^0 = 0$, we also have $x^0 = w^0$,

$$A^\top \tilde{y}^1 = A^\top(\alpha_0 y^1 + (1-\alpha_0)A^\top y^0) = \alpha_0 A^\top \tilde{y}^0 + \alpha_0 \tau A^\top(b - Aw^1) + (1-\alpha_0)A^\top y^0. \tag{4.3.5}$$

On the other hand,

$$p^1 = \tilde{p}^0 + \tau A^\top(b - A\tilde{x}^0) - \frac{1}{\mu}(x^1 - \tilde{x}^0) = A^\top \tilde{y}^0 - \frac{1}{\mu}x^1, \tag{4.3.6}$$

where for the second equality we used (4.3.4) for $k = 0$. Expressing $\tilde{p}^1$ and $\tilde{x}^1$ in terms of their affine combinations of p^1, p^0, x^1 and x^0, then substituting for p^1 using (4.3.6) and using the fact

that $x^0 = p^0 = 0$, and finally using $\tilde{y}^0 = \tau b$ and (4.3.5), we obtain,

$$\begin{aligned}
\tilde{p}^1 + \tau A^\top (b - A\tilde{x}^1) + \frac{1}{\mu}\tilde{x}^1 &= \alpha_0 p^1 + (1-\alpha_0)p^0 + \alpha_0 \tau A^\top (b - Ax^1) + (1-\alpha_0)\tau A^\top (b - Ax^0) \\
&\quad + \frac{1}{\mu}(\alpha_0 x^1 + (1-\alpha_0)x^0) \\
&= \alpha_0 (A^\top \tilde{y}^0 - \frac{1}{\mu}x^1) + \alpha_0 \tau A^\top (b - Ax^1) + (1-\alpha_0)\tau A^\top b + \frac{1}{\mu}\alpha_0 x^1 \\
&= \alpha_0 A^\top \tilde{y}^0 + \alpha_0 \tau A^\top (b - Ax^1) + (1-\alpha_0)A^\top y^0 \\
&= \alpha_0 A^\top \tilde{y}^0 + \alpha_0 \tau A^\top (b - Aw^1) + (1-\alpha_0)A^\top y^0 \\
&= A^\top \tilde{y}^1.
\end{aligned}$$

Thus we proved that (4.3.4) holds for $k = 1$. Now let us assume that (4.3.4) holds for $0 \leq k \leq n-1$, which implies $x^k = w^k, \forall 0 \leq k \leq n$ since $x^0 = w^0$. We will prove that (4.3.4) holds for $k = n$.

First, note that

$$p^n = \tilde{p}^{n-1} + \tau A^\top (b - A\tilde{x}^{n-1}) - \frac{1}{\mu}(x^n - \tilde{x}^{n-1}) = A^\top \tilde{y}^{n-1} - \frac{1}{\mu}x^n, \tag{4.3.7}$$

where the first equality is from Step 4 of Algorithm A.1 and the second equality is from (4.3.4) for $k = n-1$. From Step 6 of Algorithm A.1 and (4.3.7), we have

$$\begin{aligned}
\tilde{p}^n &= \alpha_{n-1} p^n + (1-\alpha_{n-1})p^{n-1} \\
&= \alpha_{n-1}(A^\top \tilde{y}^{n-1} - \tfrac{1}{\mu}x^n) + (1-\alpha_{n-1})(A^\top \tilde{y}^{n-2} - \tfrac{1}{\mu}x^{n-1}) \\
&= \alpha_{n-1} A^\top \tilde{y}^{n-1} + (1-\alpha_{n-1})A^\top \tilde{y}^{n-2} - \tfrac{1}{\mu}\tilde{x}^n,
\end{aligned} \tag{4.3.8}$$

where the last equality uses Step 5 of Algorithm A.1. On the other hand, from (4.3.2) we have

$$\begin{aligned} A^\top \tilde{y}^n &= A^\top(\alpha_{n-1}y^n + (1-\alpha_{n-1})y^{n-1}) \\ &= \alpha_{n-1}A^\top(\tilde{y}^{n-1} + \tau(b - Aw^n)) + (1-\alpha_{n-1})A^\top(\tilde{y}^{n-2} + \tau(b - Aw^{n-1})) \\ &= \alpha_{n-1}A^\top\tilde{y}^{n-1} + (1-\alpha_{n-1})A^\top\tilde{y}^{n-2} + \tau A^\top[b - A(\alpha_{n-1}x^n + (1-\alpha_{n-1})x^{n-1})] \\ &= \alpha_{n-1}A^\top\tilde{y}^{n-1} + (1-\alpha_{n-1})A^\top\tilde{y}^{n-2} + \tau A^\top(b - A\tilde{x}^n), \end{aligned} \tag{4.3.9}$$

where the third equality is from $w^n = x^n$ and $w^{n-1} = x^{n-1}$, the last equality is from Step 5 of Algorithm A.1. Combining (4.3.8) and (4.3.9) we get that (4.3.4) holds for $k = n$. □

Like the linearized Bregman, we can also use a simpler implementation for accelerated linearized Bregman method in which the main computation at each step is a proximal minimization. Specifically, (4.3.2) is equivalent to the following three steps.

$$\begin{cases} w^{k+1} &:= \arg\min J(w) + \frac{1}{2\mu}\|w - \mu\tilde{v}^k\|^2 \\ v^{k+1} &:= \tilde{v}^k - \tau A^\top(Aw^{k+1} - b) \\ \tilde{v}^{k+1} &:= \alpha_k v^{k+1} + (1-\alpha_k)v^k \end{cases} \tag{4.3.10}$$

As before this follows from letting $v^k = A^\top y^k$ and $\tilde{v}^k = A^\top \tilde{y}^k$ and completing the square in the objective function in the first equation of (4.3.2).

Next we prove iteration complexity bounds for both basic and accelerated linearized Bregman algorithms. Since these algorithms are standard gradient descent methods applied to the Lagrangian dual function and these results have been well established. For the completeness, we provide the proofs in the appendix.

Theorem 10. *Let the sequence* $\{x^k\}$ *be generated by the linearized Bregman method (Algorithm 4.2) and* (x^*, y^*) *be the pair of optimal primal and dual solutions for Problem* (4.2.4)*. Let* $\{y^k\}$ *be the sequence generated by Algorithm 4.3 and suppose the step length* $\tau \leq \frac{1}{L}$*, where* L *is the Lipschitz constant for* $\nabla G_\mu(y)$*. Then for the Lagrangian function*

$$\mathcal{L}_\mu(x,y) = J(x) + \frac{1}{2\mu}\|x\|^2 - \langle y, Ax - b\rangle, \tag{4.3.11}$$

we have

$$\mathcal{L}_\mu(x^*,y^*) - \mathcal{L}_\mu(x^{k+1},y^k) \leq \frac{\|y^* - y^0\|^2}{2\tau k}. \tag{4.3.12}$$

Thus, if we further have $\tau \geq \beta/L$*, where* $0 < \beta \leq 1$*, then* (x^{k+1}, y^k) *is an* ε*-optimal solution to Problem* (4.2.4) *with respect to the Lagrangian function if* $k \geq \lceil C/\varepsilon \rceil$*, where* $C := \frac{L\|y^* - y^0\|^2}{2\beta}$.

Before we analyze the iteration complexity of the accelerated linearized Bregman method, we introduce a lemma from [81] that we will use in our analysis.

Lemma 7 (Property 1 in [81])**.** *For any proper lower semicontinuous function* $\psi : \mathbb{R}^n \to (-\infty, +\infty]$ *and any* $z \in \mathbb{R}^n$*, if*

$$z_+ = \arg\min_x \{\psi(x) + \frac{1}{2}\|x - z\|^2\},$$

then

$$\psi(x) + \frac{1}{2}\|x - z\|^2 \geq \psi(z_+) + \frac{1}{2}\|z_+ - z\|^2 + \frac{1}{2}\|x - z_+\|^2, \quad \forall x \in \mathbb{R}^n.$$

The following theorem gives an iteration-complexity result for the accelerated linearized Bregman method. Our proof of this theorem closely follows the proof of Proposition 2 in [81].

Theorem 11. *Let the sequence $\{x^k\}$ be generated by accelerated linearized Bregman method (Algorithm A.1) and (x^*,y^*) be the optimal primal and dual variable for Problem* (4.2.4)*. Let $\{\alpha_k\}$ be chosen as*

$$\alpha_{k-1} \quad = \quad 1+\theta_k(\theta_{k-1}^{-1}-1), \tag{4.3.13}$$

where

$$\theta_{-1} := 1, \text{ and } \theta_k = \frac{2}{k+2}, \forall k \geq 0. \tag{4.3.14}$$

Let the sequence $\{y^k\}$ be defined as in (4.3.2) *and the step length $\tau \leq \frac{1}{L}$, where L is the Lipschitz constant of $\nabla G_\mu(y)$ and $G_\mu(\cdot)$ is defined by* (A.1)*. We have*

$$G_\mu(y^k) - G_\mu(y^*) \leq \frac{2\|y^*-y^0\|^2}{\tau k^2}. \tag{4.3.15}$$

Thus, if we further have $\tau \geq \beta/L$, where $0<\beta\leq 1$, then (x^{k+1},y^k) is an ε-optimal solution to Problem (4.2.4) *with respect to the Lagrangian function* (4.3.11) *if $k \geq \lceil\sqrt{C/\varepsilon}\rceil$, where $C := \frac{2L\|y^*-y^0\|^2}{\beta}$.*

Remark 2. *The proof technique and the choice of θ_k used here are suggested in [81] for acceler-*

ating the basic algorithm. Other choices of θ_k can be found in [4,58,59,81]. They all work here and give the same order of iteration complexity.

4.4 Extension Problems with Additional Convex Constraints

We now consider extensions of both the LB and ALB methods to problems of the form

$$\min_{x \in X} \quad J(x) \quad \text{s.t } Ax = b, \tag{4.4.1}$$

where X is a nonempty closed convex set in $\mathbb{R}^n$. It is not clear how to extend the LB and ALB methods (Algorithms 4.2 and A.1) to problem (4.4.1) since we can no longer rely on the relationship

$$0 \in \partial J(x^{k+1}) - p^k + A^\top (Ax^k - b) + \frac{1}{\mu}(x^{k+1} - x^k)$$

to compute a subgradient $p^{k+1} \in \partial J(x^{k+1})$. Fortunately, the Lagrangian dual gradient versions of these algorithms do not suffer from this difficulty. All that is required to extend them to problem (4.4.1) is to include the constraint $w \in X$ in the minimization step in these algorithms. Note that the gradient of

$$\hat{\Phi}_\mu(v) = \min_{w \in X} \{J(w) + \frac{1}{2\mu}\|w - v\|^2\}$$

remains the same. Also it is clear that the iteration complexity results given in Theorems 10 and 11 apply to these algorithms as well.

Being able to apply the LB and ALB methods to problems of the form of (4.4.1) greatly expands their usefulness. One immediate extension is to compressed sensing problems in which the signal is required to have nonnegative components. Also (4.4.1) directly includes all linear programs. Applying the LB and ALB to such problems, with the goal of only obtaining approximated optimal solutions, will be the subject of a future paper.

4.5 Numerical Experiments

In this section, we report some numerical results that demonstrate the effectiveness of the accelerated linearized Bregman algorithm. All numerical experiments were run in MATLAB 7.3.0 on a Dell Precision 670 workstation with an Intel Xeon(TM) 3.4GHZ CPU and 6GB of RAM.

4.5.1 Numerical Results on Compressed Sensing Problems

In this subsection, we compare the performance of the accelerated linearized Bregman method against the performance of the basic linearized Bregman method on a variety of compressed sensing problems of the form (4.1.1).

We use three types of sensing matrices $A \in \mathbb{R}^{m\times n}$. Type (i): A is a standard Gaussian matrix generated by the $randn(m,n)$ function in MATLAB. Type (ii): A is first generated as a standard Gaussian matrix and then normalized to have unit-norm columns. Type (iii): The elements of A are sampled from a Bernoulli distribution as either $+1$ or -1. We use two types of sparse solutions

$x^* \in \mathbb{R}^n$ with sparsity s (i.e., the number of nonzeros in x^*). The positions of the nonzero entries of x^* are selected uniformly at random, and each nonzero value is sampled either from (i) standard Gaussian (the *randn* function in MATLAB) or from (ii) $[-1,1]$ uniformly at random ($2 * rand - 1$ in MATLAB).

For compressed sensing problems, where $J(x) = \|x\|_1$, the linearized Bregman method reduces to the two-line algorithm:

$$\begin{cases} x^{k+1} & := & \mu \cdot \text{shrink}(v^k, 1) \\ v^{k+1} & := & v^k + \tau A^\top (b - Ax^{k+1}), \end{cases}$$

where the ℓ_1 shrinkage operator is defined in (4.2.10). Similarly, the accelerated linearized Bregman can be written as:

$$\begin{cases} x^{k+1} & := & \mu \cdot \text{shrink}(\tilde{v}^k, 1) \\ v^{k+1} & := & \tilde{v}^k + \tau A^T (b - Ax^{k+1}) \\ \tilde{v}^{k+1} & := & \alpha_k v^{k+1} + (1 - \alpha_k) v^k. \end{cases}$$

Both algorithms are very simple to program and involve only one Ax and one $A^\top y$ matrix-vector multiplication in each iteration.

We ran both LB and ALB with the *seed* used for generating random number in MATLAB setting as 0. Here we set $n = 2000, m = 0.4 \times n, s = 0.2 \times m, \mu = 5$ for all data sets. We set $\tau = \frac{2}{\mu \|A\|^2}$. We terminated the algorithms when the stopping criterion

$$\|Ax^k - b\| / \|b\| < 10^{-5} \tag{4.5.1}$$

was satisfied or the number of iterations exceeded 5000. Note that (4.5.1) was also used in [92]. We report the results in Table 4.5.1.

Table 4.1: Compare linearized Bregman (LB) with accelerated linearized Bregman (ALB)

Standard Gaussian matrix A		Number of Iterations		Relative error $\|x-x^*\|/\|x^*\|$	
Type of x^*	$n(m=0.4n, s=0.2m)$	LB	ALB	LB	ALB
Gaussian	2000	5000+	330	5.1715e-3	1.4646e-5
Uniform	2000	1681	214	2.2042e-5	1.5241e-5
Normalized Gaussian matrix A		Number of Iterations		Relative error $\|x-x^*\|/\|x^*\|$	
Type of x^*	$n(m=0.4n, s=0.2m)$	LB	ALB	LB	ALB
Gaussian	2000	2625	234	3.2366e-5	1.2664e-5
Uniform	2000	5000+	292	1.2621e-2	1.5629e-5
Bernoulli +1/-1 matrix A		Number of Iterations		Relative error $\|x-x^*\|/\|x^*\|$	
Type of x^*	$n(m=0.4n, s=0.2m)$	LB	ALB	LB	ALB
Gaussian	2000	2314	222	4.2057e-5	1.0812e-5
Uniform	2000	5000+	304	1.6141e-2	1.5732e-5

In Table 4.5.1, we see that for three out of six problems, LB did not achieve the desired convergence criterion within 5000 iterations, while ALB satisfied this stopping criterion in less than 330 iterations on all six problems. To further demonstrate the significant improvement the ALB achieved over LB, we plot in Figures 4.1, 4.2 and 4.3 the Euclidean norms of the residuals and the relative errors as a function of the iteration number that were obtained by LB and ALB applied to the same data sets. These figures also depict the non-monotonic behavior of the ALB method.

ALB and other fast implementations

We compare our results with the implementations used in [92], i.e, (BB+LS) kicking+BB line-search, (LB) L-BFGS. We use the randomly generated A and x^*. The stopping criterion (4.5.1) is used.

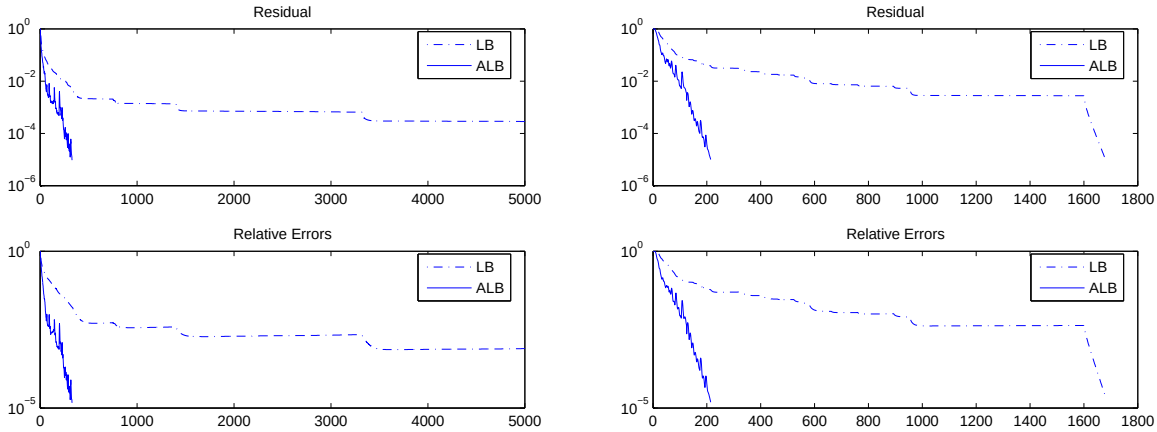


Figure 4.1: Gaussian matrix A, Left: Gaussian x^*, Right: Uniform x^*

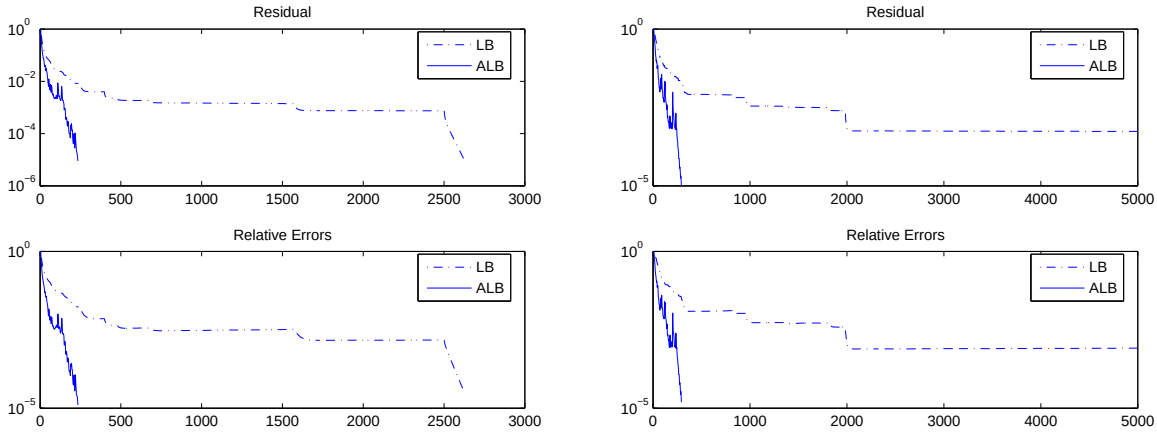


Figure 4.2: Normalized Gaussian matrix A, Left: Gaussian x^*, Right: Uniform x^*

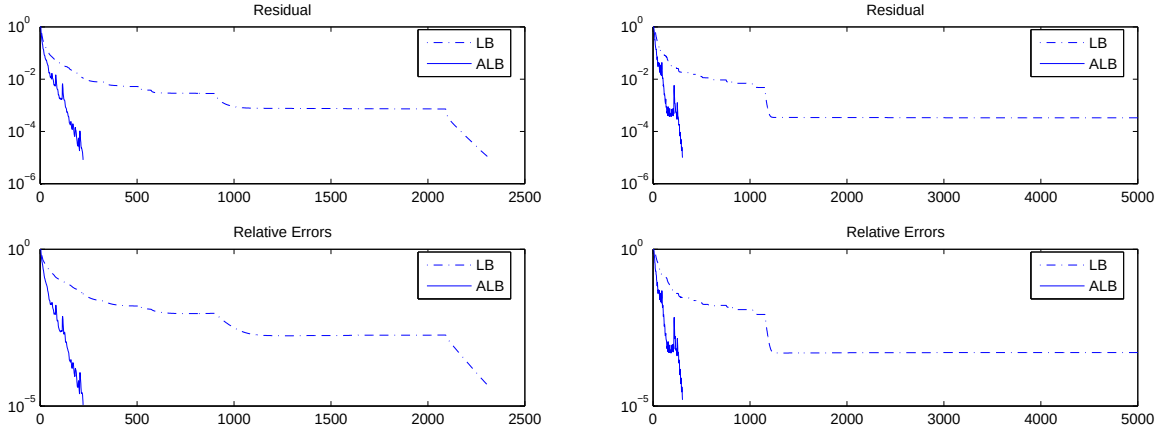


Figure 4.3: Bernoulli matrix A, Left: Gaussian x^*, Right: Uniform x^*

In Table 4.5.1, we report the number of iterations before reaching the stopping criterion. For fairness of comparison, we record the number of A and A^T multiplications at each iteration. The choices of A and x^* are stated in the table. We set $\mu = 5$ uniformly for all tests and exact regularization holds for all of the tested problem. This can be seen from the low relative errors of $O(10^{-6})$ achieved by ALB method. Increasing α will maintain exact regularization but make the algorithms take more iterations

4.5.2 Numerical Results on Matrix Completion Problems

There are fast implementations of linearized Bregman [8] and other solvers [51,52,78,88] for solving matrix completion problems. We do not compare the linearized Bregman and our accelerated linearized Bregman algorithms with these fast solvers here. Rather our tests are focused only on comparing ALB with LB and verifying that the acceleration actually occurs in practice for matrix completion problems. The nuclear norm matrix completion problem (1.2.7), i.e.,

$$\min_X \quad \|X\|_* \quad \text{s.t.} \quad \mathcal{P}_\Omega(X) = \mathcal{P}_\Omega(M), \tag{4.5.2}$$

where $[\mathcal{P}_\Omega(X)]_{ij} = X_{ij}$ if $(i,j) \in \Omega$ and $[\mathcal{P}_\Omega(X)]_{ij} = 0$ otherwise. When the convex function $J(\cdot)$ is the nuclear norm of matrix X, the Step 3 of Algorithm 4.2 with inputs X^k, P^k can be reduced to

$$X^{k+1} := \arg\min_{X \in \mathbb{R}^{m\times n}} \mu\|X\|_* + \frac{1}{2}\|X - (X^k - \mu(\tau\mathcal{P}_\Omega(\mathcal{P}_\Omega X^k - \mathcal{P}_\Omega(M)) - P^k))\|_F^2. \tag{4.5.3}$$

It is known (see, e.g., [8, 52]) that (4.5.3) has the closed-form solution,

$$X^{k+1} = \text{Shrink}(X^k - \mu(\tau\mathcal{P}_\Omega(\mathcal{P}_\Omega X^k - \mathcal{P}_\Omega(M)) - P^k), \mu),$$

where the matrix shrinkage operator is defined as

$$\text{Shrink}(Y, \gamma) := U\text{Diag}(\max(\sigma - \gamma, 0))V^\top,$$

and $Y = U\text{Diag}(\sigma)V^\top$ is the singular value decomposition (SVD) of matrix Y. Thus, a typical iteration of the linearized Bregman method (Algorithm 4.2), with initial inputs $X^0 = P^0 = 0$, for solving the matrix completion problem (4.5.2) can be summarized as

$$\begin{cases} X^{k+1} & := & \text{Shrink}(X^k - \mu(\tau\mathcal{P}_\Omega(\mathcal{P}_\Omega X^k - \mathcal{P}_\Omega(M)) - P^k), \mu) \\ P^{k+1} & := & P^k - \tau(\mathcal{P}_\Omega X^k - \mathcal{P}_\Omega M) - (X^{k+1} - X^k)/\mu. \end{cases} \tag{4.5.4}$$

Similarly, a typical iteration of the accelerated linearized Bregman method (Algorithm A.1), with initial inputs $X^0 = P^0 = \tilde{X}^0 = \tilde{P}^0 = 0$, for solving the matrix completion problem (4.5.2) can be summarized as

$$\begin{cases} X^{k+1} & := & \text{Shrink}(X^k - \mu(\tau\mathcal{P}_\Omega(\mathcal{P}_\Omega X^k - \mathcal{P}_\Omega(M)) - P^k), \mu) \\ P^{k+1} & := & \tilde{P}^k - \tau(\mathcal{P}_\Omega \tilde{X}^k - \mathcal{P}_\Omega M) - (X^{k+1} - \tilde{X}^k)/\mu \\ \tilde{X}^{k+1} & := & \alpha_k X^{k+1} + (1 - \alpha_k)X^k \\ \tilde{P}^{k+1} & := & \alpha_k P^{k+1} + (1 - \alpha_k)P^k, \end{cases} \tag{4.5.5}$$

where the sequence α_k is chosen according to Theorem 11. We compare the performance of LB and ALB on a variety of matrix completion problems. We created matrices $M \in \mathbb{R}^{n\times n}$ with rank r by the following procedure. We first created standard Gaussian matrices $M_L \in \mathbb{R}^{n\times r}$ and $M_R \in \mathbb{R}^{n\times r}$ and then we set $M = M_L M_R^\top$. The locations of the p known entries in M were sampled uniformly, and the values of these p known entries were drawn from an iid Gaussian distribution. The ratio p/n^2 between the number of measurements and the number of entries in the matrix is denoted by "SR" (sampling ratio). The ratio between the dimension of the set of $n \times n$ rank r matrices, $r(2n-r)$, and the number of samples p, is denoted by "FR". In our tests, we fixed FR to 0.2 and 0.3 and r to 10. We tested five matrices with dimension $n = 100, 200, 300, 400, 500$ and set the number p to $r(2n-r)/FR$. The random seed for generating random matrices in MATLAB was set to 0. μ was set to $5n$ (a heuristic argument for this choice can be found in [8]). We set the step length τ to $1/\mu$ since for matrix completion problems $\|\mathcal{P}_\Omega\| = 1$. We terminated the code when the relative error between the residual and the true matrix was less than 10^{-4}, i.e.,

$$\|\mathcal{P}_\Omega(X^k) - \mathcal{P}_\Omega(M)\|_F / \|\mathcal{P}_\Omega(M)\|_F < 10^{-4}. \tag{4.5.6}$$

Note that this stopping criterion was used in [8]. We also set the maximum number of iteration to 2000.

We report the number of iterations needed by LB and ALB to reach (4.5.6) in Table 4.5.2. Note that performing the shrinkage operation, i.e., computing an SVD, dominates the computational cost in each iteration of LB and ALB. Thus, the per-iteration complexities of LB and ALB are almost the same and it is reasonable to compare the number of iterations needed to reach the stopping

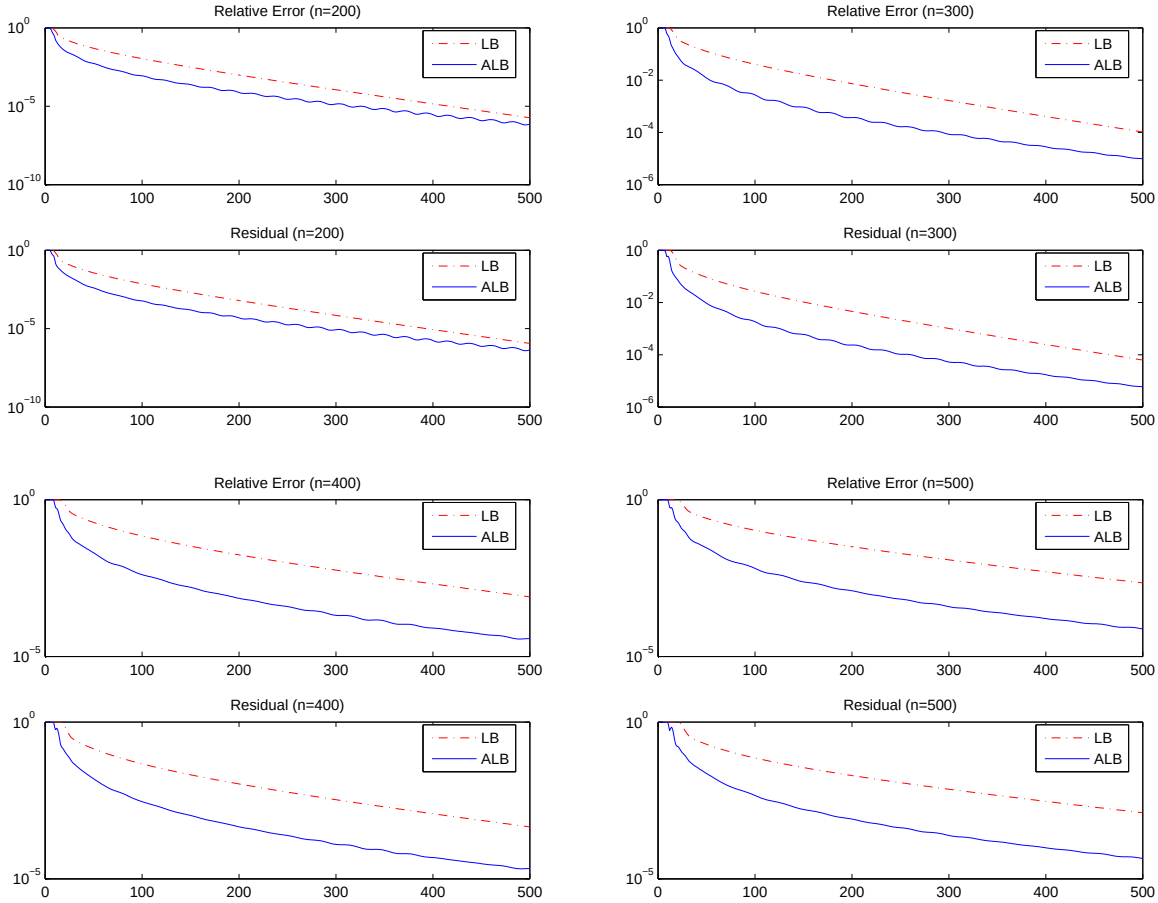


Figure 4.4: Comparison of LB and ALB on matrix completion problems with rank $= 10, FR = 0.2$ criterion. We report the relative error $err := \|X^k - M\|_F / \|M\|_F$ between the recovered matrix X^k and the true matrix M in Table 4.5.2. We see from Table 4.5.2 that ALB needed significantly fewer iterations to meet the stopping criterion (4.5.6).

In Figures 4.4 and 4.5, we plot the Frobenius norms of the residuals and the relative errors obtained by LB and ALB for iteration 1-500 for the tests involving matrices with dimension $n = 200, 300, 400$ and 500. Note that the non-monotonicity of ALB is far less pronounced on these problems.

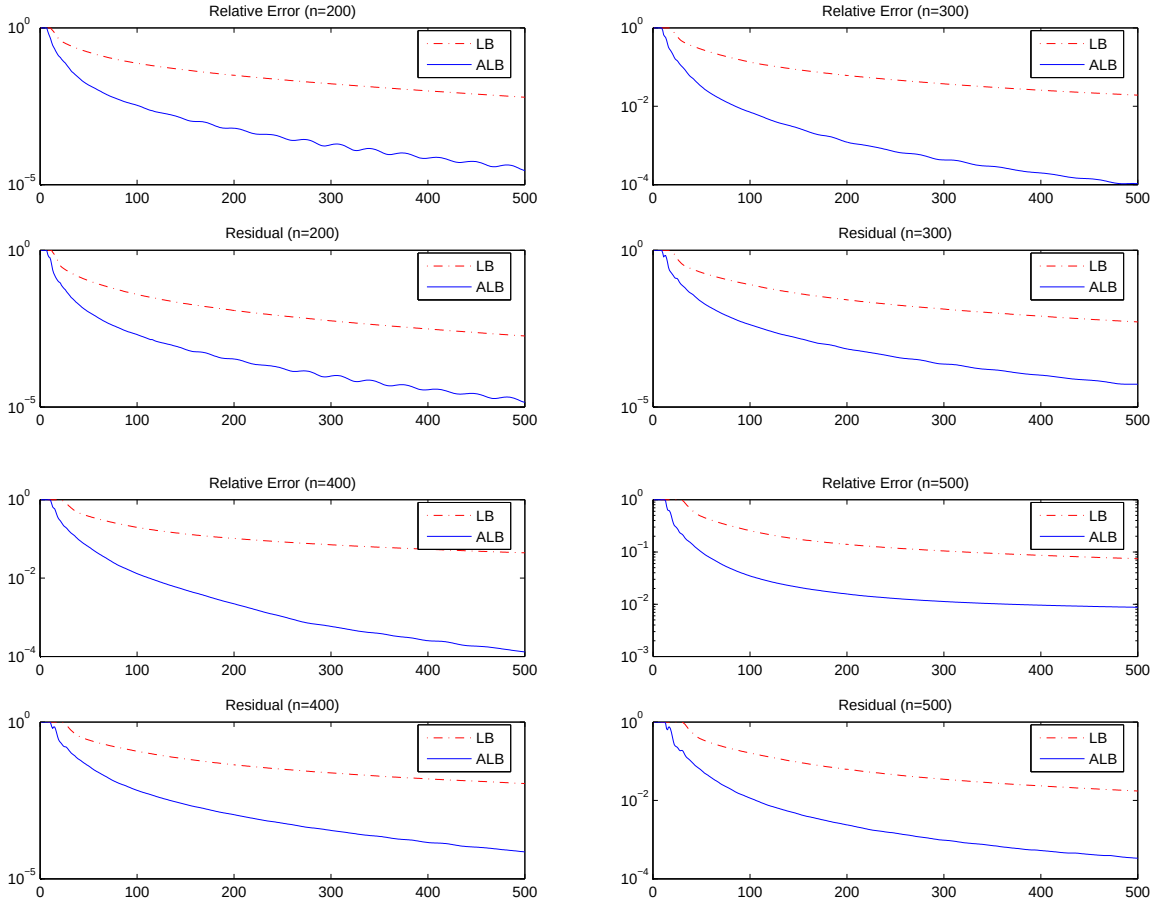


Figure 4.5: Comparison of LB and ALB on matrix completion problems with rank $= 10, FR = 0.3$

Table 4.2: Compare accelerated linearized Bregman (ALB) with BB line search (BB+LS) and the original linearized Bregman (LB)

$\mu = 5$							
Normalized Gaussian matrix A		Total A and A^T multiplications			Relative error $\|x-x^*\|/\|x^*\|$		
Type of x^*	$n(m=0.4n, k=0.2m)$	LB	BB+LS	ALB	LB	BB+LS	ALB
Uniform	1000	2000+	1014	**538**	9.999e-003	5.755e-007	**1.538e-006**
Uniform	2000	2000+	990	**582**	8.304e-003	4.309e-007	**1.638e-006**
Uniform	4000	2000+	1852	**702**	7.202e-003	2.997e-007	**1.563e-006**
Gaussian	1000	1486	242	**320**	3.747e-006	1.066e-006	**1.403e-006**
Gaussian	2000	2000+	632	**374**	1.475e-003	2.976e-006	**1.436e-006**
Gaussian	4000	2000+	2000+	**888**	2.817e-003	5.839e-004	**1.229e-006**
Gaussian matrix A		Total A and A^T multiplications			Relative error $\|x-x^*\|/\|x^*\|$		
Type of x^*	$n(m=0.4n, k=0.2m)$	LB	BB+LS	ALB	LB	BB+LS	ALB
Uniform	1000	2000+	1768	**622**	4.688e-003	1.169e-006	**1.474e-006**
Uniform	2000	2000+	2000+	**1116**	7.061e-003	1.526e-004	**1.429e-006**
Uniform	4000	2000+	1626	**532**	5.488e-003	1.268e-006	**1.504e-006**
Gaussian	1000	2000+	492	**448**	1.347e-004	3.845e-006	**1.419e-006**
Gaussian	2000	2000+	1832	**736**	9.450e-003	4.329e-006	**1.317e-006**
Gaussian	4000	2000+	2000+	**738**	3.396e-003	2.079e-004	**1.189e-006**
$\mu = 10$							
Normalized Gaussian matrix A		Total A and A^T multiplications			Relative error $\|x-x^*\|/\|x^*\|$		
Type of x^*	$n(m=0.4n, k=0.2m)$	LB	BB+LS	ALB	LB	BB+LS	ALB
Uniform	1000	2000+	1772	**624**	2.641e-002	6.812e-007	**1.420e-006**
Uniform	2000	2000+	1728	**616**	1.652e-002	4.393e-007	**1.443e-006**
Uniform	4000	2000+	2000+	**778**	1.122e-002	2.151e-003	**1.406e-006**
Gaussian	1000	1580	222	**302**	2.498e-006	7.572e-007	**1.481e-006**
Gaussian	2000	2000+	752	**420**	1.169e-003	6.956e-007	**1.435e-006**
Gaussian	4000	2000+	2000+	**808**	2.503e-003	4.236e-004	**1.383e-006**
Gaussian matrix A		Total A and A^T multiplications			Relative error $\|x-x^*\|/\|x^*\|$		
Type of x^*	$n(m=0.4n, k=0.2m)$	LB	BB+LS	ALB	LB	BB+LS	ALB
Uniform	1000	2000+	2000+	**690**	3.142e-002	2.778e-003	**1.504e-006**
Uniform	2000	2000+	2000+	**1562**	1.220e-002	4.865e-005	**1.276e-006**
Uniform	4000	2000+	1618	**730**	1.463e-002	6.519e-004	**1.561e-006**
Gaussian	1000	1622	264	**426**	3.218e-006	1.369e-006	**1.425e-006**
Gaussian	2000	2000+	1874	**584**	9.626e-003	8.747e-007	**1.502e-006**
Gaussian	4000	2000+	2000+	**836**	2.796e-003	6.113e-004	**1.251e-006**

Table 4.3: Comparison between LB and ALB on Matrix Completion Problems

	$FR = 0.2, \text{rank} = 10$					$FR = 0.3, \text{rank} = 10$				
n	SR	iter-LB	err-LB	iter-ALB	err-ALB	SR	iter-LB	err-LB	iter-ALB	err-ALB
100	0.95	85	1.07e-4	63	1.11e-4	0.63	294	1.75e-4	163	1.65e-4
200	0.49	283	1.62e-4	171	1.58e-4	0.33	1224	3.76e-4	289	1.83e-4
300	0.33	466	1.64e-4	261	1.60e-4	0.22	2000+	3.59e-3	406	1.93e-4
400	0.25	667	1.79e-4	324	1.65e-4	0.17	2000+	1.12e-2	455	1.80e-4
500	0.20	831	1.76e-4	398	1.65e-4	0.13	2000+	3.14e-2	1016	7.49e-3

Chapter 5

Fast Linearized Bregman Algorithms On Linear Programming Problems

In this chapter, we discuss how to apply the linearized Bregman method to linear programming problems. Consider the following standard form linear programming problem:

$$\begin{aligned} \min \quad & c^\top x \\ s.t \quad & Ax = b \\ & x \geq 0 \end{aligned} \tag{5.0.1}$$

It is equivalent to (4.1.1) with

$$J(x) := c^\top x + \mathbf{1}(x \geq 0),$$

where $\mathbf{1}(x \geq 0)$ is the indicator function:

$$\mathbf{1}(x \geq 0) = \begin{cases} 0, & x \geq 0 \\ +\infty, & o.w. \end{cases}$$

Under the framework of the linearized Bregman method, the iterative steps are

$$\begin{cases} x^{k+1} &= (x^k + \mu p^k - \mu c - \mu\tau A^\top (Ax^k - b))^+ \\ p^{k+1} &= p^k - \tau A^\top (Ax^k - b) - \frac{1}{\mu}(x^{k+1} - x^k). \end{cases} \tag{5.0.2}$$

where $p^k \in \partial J(x^k)$. If we define $\tilde{p}^k = p^k - c$, then (5.0.2) can be written as

$$\begin{cases} x^{k+1} &= (x^k + \mu \tilde{p}^k - \mu\tau A^\top (Ax^k - b))^+ \\ \tilde{p}^{k+1} &= \tilde{p}^k - \tau A^\top (Ax^k - b) - \frac{1}{\mu}(x^{k+1} - x^k), \end{cases} \tag{5.0.3}$$

where $\tilde{p}^k$ is the subgradient of the indicator function $\mathbf{1}(x \geq 0)$. We also have the corresponding simpler implementation with $\tilde{v}^k := \tilde{p}^k + \frac{1}{\mu}x^k$.

$$\begin{cases} x^{k+1} &= \mu(\tilde{v}^k)^+ \\ \tilde{v}^{k+1} &= \tilde{v}^k - \tau A^\top (Ax^{k+1} - b). \end{cases} \tag{5.0.4}$$

5.1 A New Convergence Analysis for the LB Method

Although the convergence for the general LB method has been well studied, we propose a new way of analyzing the LB method on LP problems. It becomes clearer in this analysis why the regular

LB method converges slowly, and how we can improve it. First we introduce the notation to be used throughout this section:

[Notation]

- $u^k := A^\top(Ax^k - b)$;
- $a \geq^v (\leq^v) b$ for two vectors $a, b \in \mathcal{R}^n$ if and only if $|a_j| \geq (\leq)|b_j|$ and $\mathrm{sgn}(a_j) = \mathrm{sgn}(b_j)$ for all j;
- Let I_k be the support of the variable x^k and $\bar{I_k}$ be the complementary set of I_k.
- For simplicity, we assume that the support I_k is the first $|I_k|$ elements $(1, 2, \cdots |I_k|)$ of $\{1, 2, \cdots m\}$.
- Let $A_k := A_{I_k}$ be the sub-matrix of A that contains only columns in I_k, and $\bar{A_k}$ be the rest of the sub-matrix of A other than A_k. Hence we have

$$A^\top = \begin{pmatrix} A_k^\top \\ \bar{A_k}^\top \end{pmatrix}.$$

- Let the range(A) and range$(A^\top)$ be the subspaces spanned by the columns of A and $A^\top$, respectively.

Lemma 8. *If the linearized Bregman method (e.g.* (4.2.9)*) is applied to the compressed sensing and the linear programming problems, we have*

$$x^k - x^{k+1} \leq^v \tau\mu u^k \tag{5.1.1}$$

Proof. For the compressed sensing problems, we have

$$x^{k+1} = \mu \cdot \text{shrink}(v^k, 1).$$

Using the update

$$\begin{aligned} v^k &= v^{k-1} - \tau A^\top (Ax^k - b) \\ &= v^{k-1} - \tau u^k, \end{aligned}$$

we get

$$\begin{aligned} x^k - x^{k+1} &= \mu(\text{shrink}(v^{k-1}, 1) - \text{shrink}(v^k, 1)) \\ &\leq^v \mu(v^{k-1} - v^k) \\ &\leq^v \mu\tau u^k. \end{aligned}$$

Similarly for linear programming, we have

$$x^{k+1} = \mu(v^k)^+.$$

Therefore

$$\begin{aligned} x^k - x^{k+1} &= \mu((v^{k-1})^+ - (v^k)^+) \\ &\leq^v \mu(v^{k-1} - v^k) \\ &\leq^v \mu\tau u^k. \end{aligned}$$

□

Theorem 12. *When applied to the compressed sensing and linear programming problems with* $\tau < \frac{2}{\mu\|A\|^2}$*, the linearized Bregman algorithm (Algorithm 4.2) has the following properties.*

1. $v^k = p^k + \frac{1}{\mu}x^k \in span\{A^\top\}$
2. $\|Ax^{k+1} - b\|^2 < \|Ax^k - b\|^2$
3. *If* $\{x^k\} \to x^*$ *and* $Ax^* = b$*, then* x^* *is the optimal solution to*

$$\begin{aligned} \min \quad & J(x) + \frac{1}{2\mu}\|x\|^2 \\ s.t \quad & Ax = b. \end{aligned}$$

Proof. 1. From step 4 of Algorithm 4.2, we have

$$p^{k+1} + \frac{1}{\mu}x^{k+1} = p^k + \frac{1}{\mu}x^k - \tau A^\top(Ax^k - b).$$

Furthermore, since $p^0 + \frac{1}{\mu}x^0 = 0 \in span\{A^\top\}$, the claim is true since the increment to $p^k + \frac{1}{\mu}x^k$ on each iteration is also in the subspace spanned by $A^\top$.

2. Let $\mu\tau\tilde{u}^k$ be the actual increment of x^{k+1} over x^k, i.e.,

$$x^{k+1} = x^k - \mu\tau\tilde{u}^k.$$

From Lemma 8, we know that

$$\tilde{u}^k \leq^v u^k, \tag{5.1.2}$$

which means the magnitude of each component of $\tilde{u}^k$ is dominated by that of u^k.

Next we show that if

$$\tau < 2/(\mu\|A\|^2),$$

the update

$$x^{k+1} = x^k - \mu\tau\tilde{u}^k$$

will lead to a decrease in $\|Ax - b\|^2$, namely,

$$\|Ax^{k+1} - b\|^2 < \|Ax^k - b\|^2.$$

If we define

$$F(\alpha) = \|A(x^k - \alpha\mu\tilde{u}^k) - b\|^2,$$

then we have

$$\begin{aligned} F'(\alpha) &= 2(A\tilde{u}^k)^\top(A(x^k - \alpha\mu\tilde{u}^k) - b) \\ &= 2(A\tilde{u}^k)^\top(Ax^k - b) - 2\alpha\mu(A\tilde{u}^k)^\top A\tilde{u}^k \\ &= 2(\tilde{u}^k)^\top u^k - 2\alpha\mu(\tilde{u}^k)^\top(A^\top A)\tilde{u}^k. \end{aligned}$$

Therefore $F'(\alpha) = 0$ implies

$$\begin{aligned} \alpha &= (\tilde{u}^k)^\top u^k / (\mu(\tilde{u}^k)^\top(A^\top A)\tilde{u}^k) \\ &\geq \frac{\|\tilde{u}^k\|^2}{\mu(\tilde{u}^k)^\top(A^\top A)\tilde{u}^k} \\ &\geq \frac{1}{\mu\|A\|^2} \end{aligned}$$

where the second equation is true because $\tilde{u}^k \leq^v u^k$ which implies

$$\begin{aligned} (\tilde{u}^k)^\top u^k &\geq (\tilde{u}^k)^\top(\tilde{u}^k) \\ &= \|\tilde{u}^k\|^2. \end{aligned}$$

Therefore, if the step length τ is chosen to be no more than 2α, i.e.,

$$\tau < \frac{2}{\mu \|A\|^2}$$

and $x^{k+1} = x^k - \tau\mu\tilde{u}^k$, we will have

$$\|Ax^{k+1} - b\|^2 < \|Ax^k - b\|^2.$$

3. For any other feasible solution z such that $Az = b$, let us check the optimality conditions

$$\langle \partial J(x^*), x^* - z \rangle = \langle p^* + \frac{1}{\mu} x^*, x^* - z \rangle,$$

where p^* is the subgradient of $J(x^*)$ with $\{p^k\} \to p^*$ and $\{x^k\} \to x^*$. Since $p^* + \frac{1}{\mu} x^* \in span\{A^\top\}$, i.e.,

$$p^* + \frac{1}{\mu} x^* = A^\top y^*,$$

for some y^*, then the optimality conditions can be written as

$$\begin{aligned} \langle \partial J(x^*), x^* - z \rangle &= \langle p^* + \frac{1}{\mu} x^*, x^* - z \rangle \\ &= \langle y^*, Ax^* - Az \rangle \\ &= \langle y^*, 0 \rangle, \end{aligned}$$

which implies that x^* is the optimal solution.

□

Remark 3. *In the original linearized Bregman method, the iterative update for $v^k = p^k + \frac{1}{\mu}x^k$ is*

$$v^{k+1} = v^k - \tau A^\top (Ax^k - b).$$

On the other hand, from Theorem 12, we know that any update of the form

$$v^{k+1} = v^k - d^k, \quad x^{k+1} = \mu(v^{k+1})^+$$

leads to the optimal solution as long as

$$d^k \in span(A^\top),$$

and

$$Ax^k \to b.$$

Indeed, we could have a much better choice for d^k than $\tau A^\top (Ax^k - b)$. In the next section, we demonstrate scenarios where $\tau A^\top (Ax^k - b)$ results in an inefficient update, and how a smarter choice of d^k can significantly accelerate the convergence rate.

5.2 A new fast linearized Bregman method for the linear programming

In this section, we first discuss the Kicking technique which was originally proposed for solving the compressed sensing problem. Here we show that it can be easily applied to the linear programming problem. Besides the Kicking technique, we also consider combining it with the *Conjugate Gradient (CG)* and *BFGS* methods to further accelerate the convergence rate.

5.2.1 The Kicking technique

The so-called Kicking technique was proposed by Osher et.al. [65] to accelerate the linearized Bregman method for solving the compressed sensing problems. In the following, we present some important observations that motivate the use of the Kicking technique, as well as the features of the Kicking technique that enable it to be extended to solving more general problems. These observations are mostly based on the primal implementations (Algorithm 4.2 and A.1).

1. The relationship between the primal variables x and p and the dual variables y and v is:

$$v^k = A^\top y^k = p^{k+1} + \frac{1}{\mu} x^{k+1}. \tag{5.2.1}$$

2. The dual variable v^k is the subgradient of the objective function in the regularized problem (4.2.4), i.e,

$$v^k = p^{k+1} + \frac{1}{\mu} x^{k+1} = \partial_x (J(x) + \frac{1}{2\mu}\|x\|)|_{x=x^{k+1}}. \tag{5.2.2}$$

3. The current iterate x^k is the unique optimal solution of (4.2.4) whenever it satisfies primal feasibility, i.e., $Ax^k = b$. Therefore the convergence rate of the linearized Bregman method highly depends on how fast the residual $\|Ax^k - b\|^2$ decreases.

4. In step 3 of Algorithms 4.2 and A.1, when the difference between x^{k+1} and x^k ($\tilde{x}^k$ in Algorithm A.1) is small and they share the same support, i.e., $I_k = I_{k+1}$, then

$$J(x^{k+1}) - J(x^k) - \langle p^k, x^{k+1} - x^k \rangle = 0. \tag{5.2.3}$$

Therefore at that iteration, the linearized Bregman iteration reduces to a gradient descent step for minimizing the residual $\|A_{I_k}x - b\|^2$, i.e.,

$$x^{k+1}_{I_k} = x^k_{I_k} - \mu\tau A^\top_{I_k}(Ax^k - b),$$

where $I_k = \{i : x^k_i > 0\}$ denotes the set of indices of the nonzero components of x^k, x_{I_k} denotes a vector in $R^{|I_k|}$ containing only the elements of x in I_k and A_{I_k} is the sub-matrix of A corresponding to $x^k_{I_k}$.

However, when the magnitude of descent direction $A^\top_{I_k}(Ax^k - b)$ is so small that

$$x^{k+1}_j \approx x^k_j, \quad j \in I_k$$

$$x^{k+1}_j = x^k_j = 0, \quad j \in I^c_k,$$

and

$$\|Ax^{k+1} - b\| \approx \|Ax^k - b\|,$$

stagnation will occur. Therefore it is not efficient to continue to solve for x within the same support I_k. To get around this problem, the Kicking technique creates a short-cut leading to the new active set I_{k+1}. More specifically, in Algorithm 4.2 the Kicking technique fixes the variable $x^{k+1} := x^k$ while updating only the p variable by

$$p_j^{k+1} := p_j^k - \Delta t \cdot \mu \left[A^\top (Ax^k - b)\right]_j \qquad j \in I_k^c \tag{5.2.4}$$

$$p_j^{k+1} := p_j^k \qquad j \in I_k \tag{5.2.5}$$

where Δt is defined as

$$\Delta t := \max\{t : p^{k+1} \in \partial J(x^k)\}. \tag{5.2.6}$$

In other words, Δt is the time when condition (5.2.3) is about to be violated. The computation on the r.h.s of (5.2.6) is easily performed when $J(x)$ is a piecewise linear function such as $\|x\|_1$. More broadly speaking, besides the compressed sensing problems, the Kicking technique is also applicable to problems with *piecewise linear* objective which naturally includes the linear programming problem, i.e.,

$$J(x) := c^\top x + \mathbf{1}\{x \geq 0\},$$

where $\mathbf{1}\{\cdot\}$ is the indicator function.

As explained above, the Kicking technique is invoked whenever the support set I_k of the iterate x^k does not change for many iterations due to the the small magnitude of the gradient descent step. Generally speaking, the slower the support I_k changes, the more likely the Kicking technique will be invoked. For linear programming problems, simple simulations presented in the next section show that the support set I_k changes even less frequently than in compressed sensing problems. This implies that the Kicking technique can be also useful for linear programming problems to avoid stagnation in the vicinity of the optimal solution of $\min \|A_{I_k} x - b\|^2$.

Remark 4. *Note that by calling the Kicking technique, extra small errors will be introduced. This is because, when applying the Kicking technique, we are violating the condition*

$$v^{k+1} \in span\{A^\top\}$$

since we added v^k by the partial vector $u^k_{I_k}$ instead of the whole vector u^k. This introduces an error to the final solution x^*. On the other hand, the error can be neglected if we use the Kicking only when $u^k_{I_k}$ is very small, which is exactly the case when $x^{k+1} \approx x^k$. Sometimes it limits the use of the Kicking technique which means that, in order to make the error negligible, it can not be called freely unless the residual is decreasing extremely slowly.

5.2.2 The linearized Bregman method with conjugate gradient steps

In [65], it has been demonstrated that the Kicking technique can be effective for compressed sensing problems. In contrast, for linear programming problems, the numerical experiments suggest that "stagnation" is in general not the only reason for the slow convergence. Therefore accelera-

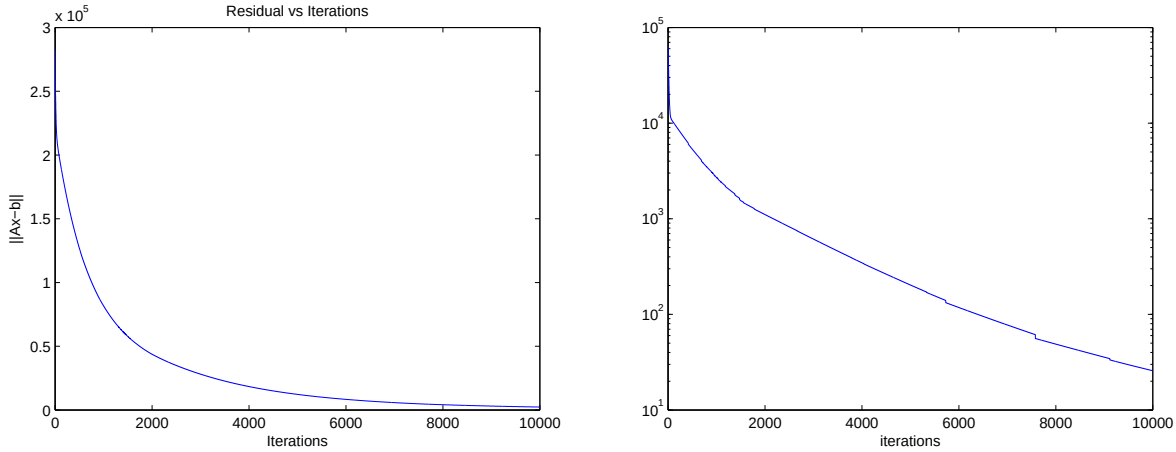


Figure 5.1: Linearized Bregman method on Netlib data 'ADLITTLT.mat'. Left: $\|Ax^k - b\|_2^2$, Right: $\|x^{k+1} - x^k\|_2^2$

tion techniques other than "Kicking" need to be developed. An illustration of this is the following. We applied the linearized Bregman to the data set of *ADLITTLE.mat* in Netlib, and we set the following criterion for calling the Kicking technique.

$$\|x^{k+1} - x^k\| \leq 0.1.$$

From Figure 5.1, we see that the overall convergence rate was fairly slow. As the main indicator of the convergence, $\|Ax^k - b\|_2^2$, decayed slowly to around 10^{-3} after 10^4 iterations. Surprisingly, within 10^4 iterations, which is much larger than the criterion we set to call the Kicking procedures, the Kicking technique was never invoked since the difference between two consecutive iterates, i.e., $\|x^{k+1} - x^k\|_2^2$, was always 30. This implied that there was no "stagnation" despite of the slow convergence rate. On the other hand, for the above problem, the active set I_k change only 38 times throughout 10,000 iterations. As is mentioned in the previous subsection, within the same active

set I_k, the linearized Bregman method reduces to the straight gradient descent method with the fixed step length τ for minimizing the residual $\|A_{I_k}x-b\|^2$. Therefore we know the main reason for the slow convergence is due to the inefficient gradient steps.

The above arguments provide us with the important clues on how to speed up the convergence. Instead of using the gradient as the descent direction, we can use the conjugate gradient steps which converge in n steps to the minimizer of an unconstraint quadratic function of n variables. Algorithm 5.1 is the standard *Conjugate Gradient Method* for minimizing $\|Ax-b\|^2$. The algorithm keeps

Algorithm 5.1 Conjugate Gradient Method

1: $r^0 = -A^\top(Ax^0-b)$,
2: $u^0 = r^0$,
3: $\alpha_k = \frac{\|r^k\|^2}{\|Au^k\|^2}$,
4: $x^{k+1} = x^k + \alpha_k u^k$,
5: $r^{k+1} = r^k - \alpha_k A^\top A u^k$,
6: **if** u^{k+1} is small **then**
7: EXIT
8: **else**
9: $\beta_k = \frac{\|r^{k+1}\|^2}{\|r^k\|^2}$
10: $u^{k+1} = r^{k+1} + \beta_k u^k$
11: **end if**

track of the last descent direction $\{u^k\}$ and residual(gradient) $\{r^k\}$. The steps (3) and (4) perform a conjugate gradient descent step. Note that (3) and (4) are crucial to make r^k perpendicular to past iterates so that the same property will hold for the next descent direction u^{k+1}. The steps (9) and (10) generate the new descent direction u^{k+1} which is conjugate to all the previous descent directions and residuals.

Recall that our goal is to accelerate the decrease of the residual $\|A_{I_k}x-b\|_2^2$ with I_k being the support of x; therefore steps (3) and (9) in Algorithm 5.1 need to be slightly modified when

combined with the linearized Bregman. Suppose that x^{k+1} has the same support as x^k. Then although we update v^k by

$$v^{k+1} = v^k + \tau u^k,$$

the actual increment in x^k is $u^k_{I_k}$ with a step size τ, i.e.,

$$x^{k+1}_{I_k} = x^k_{I_k} + \tau\mu u^k_{I_k}.$$

Therefore to generate a conjugate gradient direction, we compute α_k and β_k in Algorithm 5.1 by

$$\alpha_k = \frac{\|r^k_{I_k}\|^2}{\|Au^k_{I_k}\|^2} \tag{5.2.7}$$

$$\beta_k = \frac{\|r^{k+1}_{I_k}\|^2}{\|r^k_{I_k}\|^2}. \tag{5.2.8}$$

If we can perform the above CG updates (5.2.7)-(5.2.8) without changing the support I_k, the residual $\|Ax-b\|$ will decrease quickly in the subspace $x \in I_k$. However, the major issue that limits its effectiveness is that the support sets of x and p change. Note that in the update of x_{k+1} and p_{k+1}, i.e.,

$$\begin{aligned} x_{k+1} &= x_k + \alpha_k \begin{pmatrix} u^k_{I_k} \\ 0 \end{pmatrix} \\ p_{k+1} &= p_k + \frac{\alpha_k}{\mu} \begin{pmatrix} 0 \\ u^k_{I_k} \end{pmatrix}, \end{aligned}$$

Next we describe in detail the procedures of incorporating conjugate gradient steps along with the Kicking technique in the linearized Bregman method.

1. We set $\tau_0 = \frac{2}{\mu\|A\|^2}$(or a lower estimate of it), and start with $v^0 = 0$; thus x^0 and p^0 can be also computed. The initial descent direction u^0 and the residual r^0 are

$$r^0 = u^0 = A^\top(b - Ax^0)$$

2. Suppose at the kth iteration, we have obtained a descent direction u^k, then the step size τ is computed as

$$\tau = \begin{cases} \alpha_k & \text{if } \alpha_k \leq \Delta t_k \\ \max\{\Delta t_k, \tau_0\} & \text{o.w.} \end{cases} \tag{5.2.9}$$

where α_k is computed by (5.2.7) and Δt_k is the time when some nonzero component of v^k is about to change its sign, i.e.,

$$\Delta t_k := \max\{s | v_j^k \cdot (v_j^k + s u_j^k) > 0 \ \text{ for } \ \forall j : v_j^k \neq 0\} \tag{5.2.10}$$

The idea behind (5.2.9) is that we try to take a full exact line search step α_k without changing the supports of x^k and p_k characterized by the step length Δt_k. If this is possible, i.e.,

$$\alpha_k \leq \Delta t_k,$$

we use the next descent direction u^{k+1} as a conjugate gradient direction. Otherwise when $\alpha_k > \Delta t_k$, we take the largest step length without violating

$$\|Ax^{k+1} - b\|^2 < \|Ax^k - b\|^2. \tag{5.2.11}$$

It is obvious that Δt_k leads to (5.2.11) since $\Delta t_k < \alpha_k$ by definition, and it has been shown in the second conclusion of Theorem 12 that (5.2.11) holds as long as the step size is less than $\frac{2}{\mu\|A\|^2}$.

3. Perform the regular linearized Bregman steps with the new step length τ, i.e.,

$$\begin{aligned} x^{k+1} &= \mu \cdot \max(v^k, 0) \\ v^{k+1} &= v^k + \tau u^k, \end{aligned}$$

and the residual is computed as

$$r^{k+1} = A^\top (b - Ax^{k+1}).$$

4. When $\|r^{k+1}_{I_{k+1}}\| \leq \varepsilon$, where ε is a pre-defined small value, we know that x^{k+1} is close to the optimal solution to

$$\min_{x \in I_{k+1}} \|Ax - b\|^2.$$

Therefore we switch to the new active set by applying the Kicking technique.

5. Now we compute the next descent direction u^{k+1} as follow.

   - If $\tau = \alpha_k$, this implies that a full steepest step is taken without changing the active set of x^k. Therefore the next descent direction should be conjugate to the previous one, i.e.,

   $$u^{k+1} \quad = \quad r^{k+1} + \beta_k u^k$$

   where β_k is computed by (5.2.8).

   - If $\tau \neq \alpha_k$, we cannot generate the conjugate gradient u^{k+1} and we set u^{k+1} to the regular gradient r^{k+1}, i.e.,

   $$u^{k+1} = A^\top (b - Ax^{k+1}).$$

The above procedure is described in Algorithm 5.2. From Figure 5.1, we already see that, without

Algorithm 5.2 Conjugate Gradient Linearized Bregman (CGLB)

Set $v^0 = 0$ and $x^0 = (-\mu \cdot c)^+$.
Set $r^0 = u^0 = A^\top (b - Ax^0)$, $\tau_0 = 2/(\mu \|A\|^2)$ and $\varepsilon = 10^{-8}$.
for $k = 1, 2, \cdots$ **do**
  τ is computed by (5.2.9).
  $x^{k+1} = \mu \cdot (v^k)^+$,
  $v^{k+1} = v^k + \tau u^k$,
  $r^{k+1} = A^\top (b - Ax^{k+1})$,
  if $\|r^{k+1}_{I_{k+1}}\|^2 \leq \varepsilon$ **then**
    Call the **Kicking procedure**.
  else
    if $\tau = \alpha_k$ **then**
      $u^{k+1} = r^{k+1} + \beta_k u^k$ where β_k is defined as (5.2.8).
    else
      $u^{k+1} = r^{k+1}$
    end if
  end if
end for

the CG updates we discussed above, the linearized Bregman method can be fairly inefficient even with the Kicking technique. Next, we apply our CGLB method (Algorithm 5.2) to both synthetic and real data sets and demonstrate the great improvement of the new CGLB algorithm over the original linearized Bregman method. In Table 5.1, we compare the CGLB method with the Matlab solver on the synthetic data sets. The components of A, b and c are randomly generated from the standard Gaussian distribution. In order to make the best use of the linearized Bregman method which performs only matrix and vector multiplications, we make A sparse by zeroing out all but a fraction k of its components.

The Random sparsity matrix A	Cpu time(sec)		Residual $\|Ax-b\|_2$	
(n,m,k)	Matlab	CGLB	Matlab	CGLB
$(10^4,1000,0.05)$	11.7	5.1	4.851432e-011	7.068521e-012
$(10^4,2000,0.02)$	86.2	17.5	3.330402e-011	1.146507e-011
$(5000,1000,0.02)$	11.6	3.0	1.437937e-011	0.911005e-011

Table 5.1: CGLB v.s. Matlab on synthetic data sets with large and sparse A's.

In Figure 5.2, we demonstrate the significant inprovment that the CGLB method exhibits over the straight linearized Bregman method on several real data sets from the Netlib data base. As we know, the residual $\|Ax-b\|_2^2$ is the major indicator of the convergence for both methods, we compare their decay rate on $\|Ax-b\|_2^2$ in Figure 5.2.

In Table 5.2, we compare our CGLB method with the Matlab solver on several real Netlibs data sets. It is worth mentioning that, because of the extra l_2 perturbation $\frac{1}{2\mu}\|x\|_2^2$, the CGLB method is not solving the original linear programming. In order to produce the correct solution, μ is required to be large enough. However, if the original linear programming problem is unbounded, CGLB fails to give the correct answer no matter how large μ is.

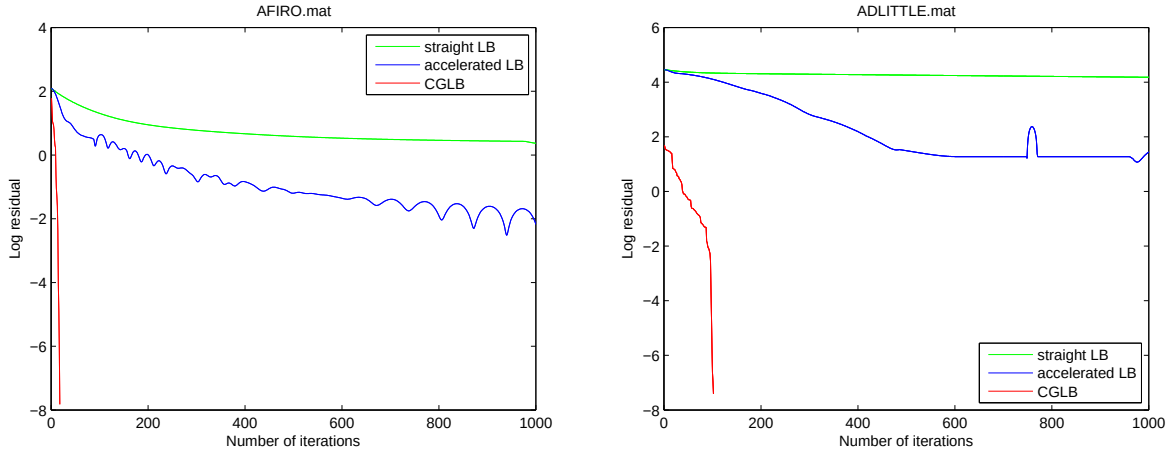


Figure 5.2: The decay of the residual $\log_{10}\|Ax-b\|_2^2$ over the iterations. Here we compare three different algorithms. The green curve is the straight linearized Bregman method, the blue curve is the accelerated linearized Bregman from the previous section, which is based on the Nesterov accelerating technique, the red curve is the CGLB method.

Table 5.2: CGLB v.s. Matlab on Netlib data sets

Netlib Datas	Matrix Size	Cpu time		Residual $\|Ax-b\|$		Error
Name	(n,m)	Matlab	CGLB	Matlab	CGLB	$\frac{\|c^\top(x_{CGLB}-x_{matlab})\|}{\|c^\top x_{matlab}\|}$
AFIRO.mat	(51, 27)	0.3	0.1	3.940887e-014	9.740180e-013	6.1483e-010
ADLITTLE.mat	(138, 56)	0.5	0.4	3.153577e-014	4.297251e-005	0.0065
ADLITTLE.mat	(138, 56)	0.5	0.4	3.153577e-014	4.297251e-005	0.0065
SC50B.mat	(78, 50)	0.3	0.1	1.611010e-014	1.898189e-011	3.8775e-014
SC50A.mat	(78, 50)	0.3	0.1	8.534108e-013	9.489085e-005	4.6645e-006
GROW22.mat	(946, 440)	0.5	1.1	3.266052e-008	1.789332e-010	NA
FIT1D.mat	(24, 1049)	0.4	0.5	9.581569e-008	9.355395e-005	NA
BEACONFD.mat	(173, 295)	0.4	0.3	2.572157e-012	3.547562e-005,	0.0078

Remark 5. *In our algorithm, we use a step size as* $\tau_0 = \frac{2}{\mu\|A\|^2}$ *when a full step conjugate gradient update is not achievable, since this always guarantees the decrease on the norm of the residual* $\|Ax-b\|_2^2$. *Moreover, it changes the support by one or more indices. There are other possible support transition schemes. For instance, one could choose* τ_0 *to be* Δt_k *so that each time only one index is allowed to enter or leave the support.*

Remark 6. *In Algorithm 5.2, within each support* I_k, *we solve a nonnegativity-constrained (i.e.,* $x \geq 0$*) quadratic minimization problem. At the mean while, the support* I_k *changes over iterations.*

There are similar algorithms on box-constrained quadratic programming problems such as [20, 21, 26, 54, 63, 90]. The major difference in our work is the extra requirement that all the descent directions must lie the subspace spanned by columns of $A^\top$.

5.2.3 The BFGS Linearized Bregman Method

As is discussed in the previous section, within in the subspace determined by I_k, the linearized Bregman method reduces to the quadratic subspace minimization, i.e.,

$$\min \|A_k x - b\|_2^2.$$

In this section, we discuss how to apply BFGS steps instead of the conjugate gradient steps to accelerate the decrease in the residual $\|Ax_k - b\|^2$ within the support I_k.

The BFGS method is given by Algorithm 5.3.

Algorithm 5.3 BFGS Algorithm

1: Set $x^0 = 0$ and $H^0 = I$.
2: **for** $k = 1, 2, \cdots$ **do**
3: $g^k = \nabla f(x^k)$,
4: $p^k = -H^k g^k$,
5: Compute the step length α_k through either the exact or inexact line search.
6: $x^{k+1} = x^k + \alpha_k p^k$,
7: $s^k = x^{k+1} - x^k$,
8: $y^k = \nabla f(x^{k+1}) - \nabla f(x^k)$.
9: Update the inverse Hessian H^k by

$$H^{k+1} = H^k + \frac{s^k (s^k)^\top}{y^k \cdot s^k}\left[\frac{(y^k)^\top H^k y^k}{y^k \cdot s^k} + 1\right] - \frac{s^k (y^k)^\top H^k + H^k y^k (s^k)^\top}{y^k \cdot s^k}$$

10: **end for**

Next we apply the BFGS updates in the exactly the same way as presented in Algorithm 5.3,

which solves the unconstraint least square problem with $A = A_k$ where A_k contains the columns of A that correspond to I_k. However, when the support I_k changes, we propose a novel approach to proceed with the BFGS Hessian update within the new support. Note that the variable v can be written as

$$v = x/\mu + p = \begin{pmatrix} x_+/\mu \\ 0 \\ \vdots \\ 0 \\ p_- \end{pmatrix}$$

where x_+ and p_- are positive and negative sub-vectors of variables x and p, respectively, i.e,

$$x^k = \mu \cdot (v^k)^+ = \begin{pmatrix} x_+^k \\ 0 \end{pmatrix},$$

and

$$p^k = (v^k)^- = \begin{pmatrix} 0 \\ p_-^k \end{pmatrix}.$$

We denote ***v-null*** to be the set of components of v that are equal to 0. We first discuss the case where *v-null* is empty. It is more complicated when *v-null* is not empty and we leave a discussion of this to the end of this section. As we show below, like CGLB, the BFGS update also generates a sequence of $\{x^k\}$ such that

$$x^k \in \text{range}(A^\top).$$

Theorem 13. *The BFGS algorithm (Algorithm 5.3) generates a sequence* $\{x^k\}$ *and approximate inverse Hessian matrices* $\{H^k\}$ *such that*

1. *For* $\forall k$, $x^k \in range\{A^\top\}$.

2. *There exists a sequence of symmetric matrices* $\{M^k\}$, *such that* $M^0 = 0$ *and*

$$H^k = I + A^\top M^k A.$$

3. *The update for* M^k *satisfies*

$$M^{k+1} = M^k + C_1 \cdot t^k (t^k)^\top - \frac{N^k + (N^k)^\top}{C_2},$$

where

$$\begin{aligned} t^k &= \alpha_k[(b - Ax^k) + M^k AA^\top (b - Ax^k)] \\ N^k &= t^k (As^k)^\top (I + AA^\top M^k) \\ C_2 &= y^k \cdot s^k \\ C_1 &= [\frac{(y^k)^\top H^k y^k}{y^k \cdot s^k} + 1]/(y^k \cdot s^k) \end{aligned}$$

[How to guarantee that $v \in range\{A^\top\}$**?]**

We assume that *v-null* is empty so at the kth iteration with the current support $I_{k'}$, we have

$$v^k = \begin{pmatrix} x_+^k/\mu \\ p_-^k \end{pmatrix}$$

where $x_+^k \in \mathcal{R}^{|I_{k'}|}$ and $p_-^k \in \mathcal{R}^{|\bar{I}_{k'}|}$. Within the same support $I_{k'}$, we update the variable x_+^k by minimizing $\|A_{k'}x-b\|^2$ using the BFGS method (Algortihm 5.3), i.e.,

$$\begin{cases} g^k &= A_{k'}^\top (A_{k'}x^k - b) \\ d^k &= -(I + A_{k'}^\top M^k A_{k'}) \cdot g^k \\ x_+^{k+1} &= x_+^k + \alpha_k d^k \end{cases} \tag{5.2.12}$$

where $M \in \mathcal{R}^{m\times m}$ and we have implicitly used the third conclusion of Theorem 13 to represent the approximate inverse Hessian H^k.

Note that d^k is obviously in the range space of $A_{k'}^\top$, i.e.,

$$d^k = A_{k'}^\top \cdot [(b - A_{k'}x^k) - M^k A_{k'} g^k] \tag{5.2.13}$$

Therefore in order to make sure that $v^{k+1} \in range\{A^\top\}$ assuming that $v^k \in range\{A^\top\}$, we update p_-^k by replacing $A_{k'}^\top$ in d^k by $\bar{A}_{k'}^\top$, i.e.,

$$p_-^{k+1} = p_-^k + \alpha_k \bar{A}_{k'}^\top \cdot [(b - A_{k'}x^k) - M^k A_{k'} g^k]$$

where $A = [A_{k'}|\bar{A}_{k'}]$. Hence the we have

$$v^{k+1} = v^k + \alpha_k A^\top [(b - A_{k'}x^k) - M^k A_{k'} g^k], \tag{5.2.14}$$

and starting with $v^0 = 0$, we always have $v^k \in range\{A^\top\}$ due to Theorem 13.

[How to choose the step length α_k?]

Next we discuss how to determine the step length α_k in the update for the variable v, i.e.,

$$\begin{aligned} v^{k+1} &= v^k + \alpha_k A^\top [(b - A_{k'}x^k) - M^k A_{k'} g^k] \\ &= v^k + \alpha_k d_v^k \end{aligned}$$

In the case of the unconstrained least square problem, BFGS with exact line search behaves better than the one with inexact line search, and the exact line search itself is faster than inexact line search for the quadratic objective functions. Therefore, we use exact line search with respect to the function $\|A_{k'}x - b\|^2$ whenever it does not change the current support. However when an exact line search is about to change the current support $I_{k'}$, we proceed to the next iteration in the following way.

(P1) We perform the inexact line search whenever the exact line search fails to maintain the current support. We can always choose the step size to be

$$\alpha_k := \frac{2}{\mu \|A\|^2},$$

and, as discussed in the previous section, it will always guarantee a decrease in $\|Ax-b\|_2^2$.

(P2) Because of the non-smoothness of the function $\|A_{(v>0)}x-b\|^2$ at the turning point between different supports $I_{k'}$ and $I_{k''}$ (or more precisely, when *v-null* is not empty and v has components equal 0), inexact line search may have difficulty in finding a point that satisfies the *Curvature condition*. The other idea which does not require using inexact line search when the support is about to changes is to let

$$\alpha_k := \max\{t : (v^k + t \cdot d_v^k > 0) = I_{k'}\}.$$

Hence the support just starts to change at α_k with the set of positive components either expanding or shrinking. Then within this new support, we continue proceeding with the BFGS steps and exact line search.

[How to choose the starting inverse Hessian matrix H when the support changes?] Note that the inverse Hessian matrix we are talking about here is the one used to update x_+^k, i.e.,

$$x_+^{k+1} = x^k - \alpha_k(I + A_k^\top M^k A_k)g^k,$$

so we have $H = I + A_k^\top M^k A_k \in \mathcal{R}^{|I_k|\times|I_k|}$ and $M^k = \mathcal{R}^{m\times m}$. Suppose at the kth iteration, the support changes from I_k to I_{k+1}, and the current inverse Hessian matrix is written as

$$H^k = I + A_k^\top M^k A_k.$$

Then for the next iteration, we keep M^k unchanged and start with the

$$H^{k+1} = I + A_{k+1}^\top M^k A_{k+1}.$$

This update ensures that H^{k+1} inherits most of the "curvature information" from H^k. For instance, if I_{k+1} expands I_k by one component, then H^k is exactly the sub-matrix of H^{k+1} that corresponds to I_k. Similarly, if I_{k+1} shrinks I_k by one component, then H^{k+1} truncates from H^k the information that corresponds to $I_{k+1} \backslash I_k$.

In short, the rule of updating the inverse Hessian matrix $H = I + A_k^\top M^k A_k$ is that

1. Update M^k if the support I_k is unchanged;

2. Update A_k if the support changes.

[The BFGS Linearized Bregamn Method] Our approach for applying BFGS steps within the linearized Bregman method is described in Algorithm 5.4. Steps 1-3 initialize the variables as well as the inverse Hessian matrix $H^0 = I + A_0^\top M^0 A_0 = I$. The while loop terminates when we obtain a feasible solution x^k. Step 6 computes the gradient of the function of the partial residual $\|A_k x - b\|^2$. When $\|g^k\|^2$ is small, then the current x^k is already nearly optimal within the current support I_k, so Kicking needs to be applied in Step 7 and the support will be changed. Steps 10-11 implemented an exact line search with respect to the function $\|A_k x - b\|^2$ and d_x^k corresponds to the sub-vector of d^k that affects x_+^k. Steps 12-15 apply when the exact line search does not change the support, thus a full BFGS step is taken and M^k is updated accordingly. Steps 17-19 apply when an exact

line search is about to change the current support, therefore $\bar{\alpha}_k$ that can be computed by either (P1) or (P2) which will possibly change the support of v^{k+1}.

Algorithm 5.4 BFGS Linearized Bregman

Given v^0 and $x^0 = \mu \cdot \max(v^0 - c, 0)$.
The support $I_0 := (v^0 > 0)$ and $\varepsilon = 10^{-8}$.
Set $M^0 = 0$ ($H^0 = I$).
while $\|Ax^k - b\|^2 \geq \varepsilon$ **do**
  Compute the gradient: $g^k = A_k^\top (Ax^k - b)$;
  if $\|g^k\|^2 \leq \varepsilon$ **then**
    Apply the Kicking technique.
  else
    Compute the descent direction: $d^k = A^\top[(b - A_k x^k) - M^k A_k g^k]$ and $d_x^k = d_{I_k}^k$;
    Compute the exact step length: $\alpha_k = \frac{(d_x^k)^\top g^k}{\|A_k d_x^k\|^2}$;
    if $(v^k + \alpha_k \cdot d^k > 0) = I_k$ **then**
      $v^{k+1} = v^k + \alpha_k \cdot d^k$;
      Compute M^{k+1} from M^k as stated in Theorem 13.
      (The support I_{k+1} remains unchanged)
    else
      Compute $\bar{\alpha}_k$ by either **(P1)** or **(P2)** stated in section 5.2.2.
      $v^{k+1} = v^k + \bar{\alpha}_k \cdot d^k$.
      Determine the new support I_{k+1} depending on v^{k+1}.
    end if
  end if
end while

[The nonempty set of *v-null* and restarting the BFGS method]

When the set of *v-null* is not empty, which means that v has components exactly equal to 0, the situation becomes much more complicated. For instance, suppose that we have a zero iterate $v^k = 0$ and some inverse Hessian matrix is given as H^k, then with the current gradient vector g^k, the direction generated by the BFGS method is

$$d^k = -H^k g^k,$$

which is descent direction if H^k is positive definite, i.e.,

$$d^k \cdot g^k < 0.$$

However only the positive components of d^k will contribute to $x^{k+1} := \mu \cdot \max(v^k + \alpha_k d^k, 0)$ since all other components will be eventually zeroed out. Therefore the *real* increment that is made to the variable x^k is

$$d_x^k = (d^k)^+.$$

But it is likely that after zeroing out all the negative component, d_x^k is no longer a descent direction!

In fact, this has been an issue for most of the projected Newton's type of algorithms for a long time. A possible solution to this, proposed in [6], is to rewrite the inverse Hessian matrix in the form that is positive definite and diagonal with respect to the set *v-null*, i.e.,

$$H^k = \begin{pmatrix} \bar{H}^k & 0 & \dots & 0 \\ 0 & d^{r+1} & \dots & 0 \\ \vdots & 0 & \ddots & 0 \\ 0 & 0 & \dots & d^n \end{pmatrix}$$

where $v_{r+1}^k = \cdots v_n^k = 0$. However, if we use the H^k of this form and update v^k by

$$v^{k+1} := v^k - H^k g^k,$$

then it will be in general extremely hard to make sure that $v^k \in range\{A^\top\}$ unless we choose

$H^k = I$, which is equivalent to abandoning all previous curvature information and restarting the BFGS steps.

For a non-sparse A, the *v-null* is mostly empty since $v^k \in range\{A^\top\}$. Therefore in this case, we use inexact line search for $\bar{\alpha}_k$ (P1) when exact line search will change the current support. However, if we use (P2) to compute $\bar{\alpha}_k$, then we have at least one component of v^k equal to 0 every time the support changes. In practice, when *v-null* is not empty and the projected direction d_x^k is no longer a descent direction, we restart the BFGS by setting $M = 0(H = I)$. In general, the set *v-null* can be large in the beginning and decreases quickly as we restart the BFGS several times, but we can also be extremely unlucky that *v-null* never quickly shrinks and our algorithm degenerates to the steepest descent method.

5.2.4 Comparison with the CGLB method

Finally in Figure 5.3, we compare the Linearized Bregman based the BFGS update with the CGLB method. Since each iteration for BFGS update is much more time consuming, we only compare their number of iterations Note that to accelerate run time of each iteration, we can mimic the update in the limited memory BFGS algorithm [49]. However, the generalization is not straight forward since we also need to guarantee that

$$v^k \in \text{span}(A^\top).$$

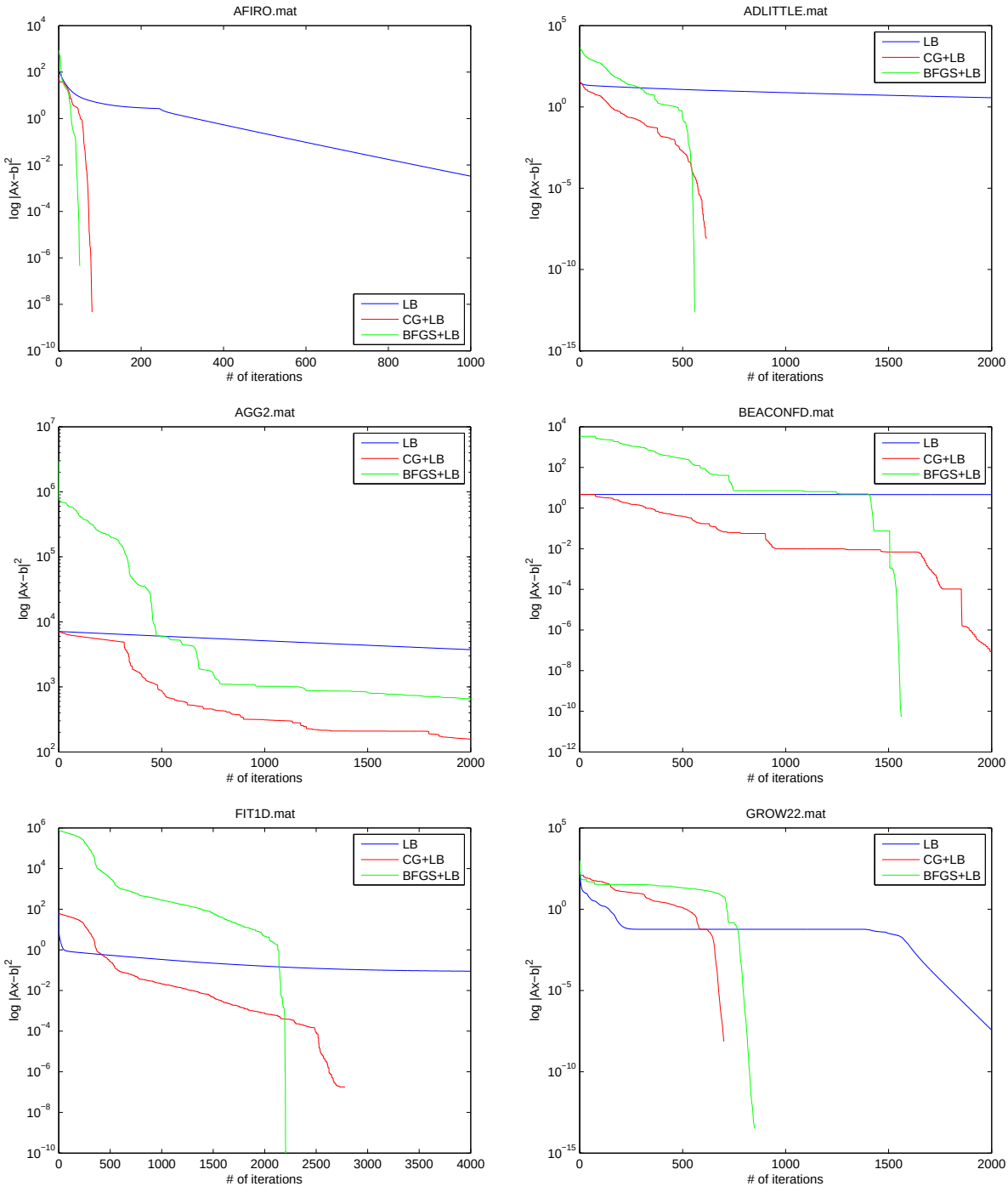


Figure 5.3: The decay of the residual $\log_{10} \|Ax-b\|_2^2$ over the iterations. Here we compare three different algorithms. The blue curve is the straight linearized Bregman method, the red curve is the LB+CG method, and the green curve is the LB+BFGS method.

On the other hand, it can be expected its number of iterations will be slightly larger than that of the above BFGS based method since the L-BFGS algorithm maintains a coarser approximation for the Hessian matrix.

Chapter 6

Conclusions

We established provable models for several sparse tensor modeling problems as well as theoretical guarantees for exact recovery. We also proposed efficient optimization algorithms for solving large-scale sparse modeling and linear programming problems.

In Chapter 2, we established a theoretical bound for the low-rank tensor recovery model, i.e., (2.4.1), based on SNN convexification. The model (2.4.1), extending both matrix completion and matrix RPCA models to the case of tensors, employs a strongly convex programming formulation. Its reduced form, i.e., $\tau = 0$, has been repeatedly used in practice with an excellent empirical performance. This is, to the best of our knowledge, the first rigorous study on theoretical guarantees for the aforementioned problems. Simulations suggest that the tensor model (2.4.1) is not superior to the matrix model unconditionally; thus we propose a new set of tensor incoherence conditions under which using high-order tensor models based on SNN minimization is plausible.

Although we established a set of sufficient conditions for exactly recovering a low-rank tensor, it remains unclear how to derive necessary conditions except for the generic settings with random

Gaussian measurements, which is discussed in detail in Chapter 3. It would be more interesting to extend the similar arguments to the more practical tensor completion and tensor RPCA models discussed in this work. Furthermore, the bound achieved in Theorem 1 does not explain why the SNN model is superior to the Singleton model, as suggested by numerical simulations, when the Tucker ranks of all modes are simultaneously low. This requires a sharper bound for the SNN model, which is an unaddressed issue in this work and serves as an interesting topic for future research.

In Chapter 3, we establish several theoretical bounds for the problem of low-rank tensor recovery using random Gaussian measurements. For the nonconvex model (3.2.1), we show that $\left((2r)^K + 2nrK + 1\right)$ measurements are sufficient to recover any $\mathcal{X}_0 \in \mathfrak{T}_r$ almost surely. We highlight that though the nonconvex recovery program is NP-hard in general, it can serve a baseline for evaluating tractable (convex) approaches. For the conventional convex surrogate sum-of-nuclear-norms (SNN) model (3.2.2), we prove a necessary condition that $\Omega(rn^{K-1})$ Gaussian measurements are required for reliable recovery. This lower bound is derived from our study on multi-structured object recovery under a very general setting, which can be applied to many other scenarios (e.g. signal processing, metric learning, computer vision). To narrow the apparent gap between the non-convex model and the SNN model, we unfold the tensor into a more balanced matrix while preserving its low-rank property, leading to our square reshaping model (3.2.5). We prove that $O(r^{\lfloor \frac{K}{2} \rfloor} n^{\lceil \frac{K}{2} \rceil})$ measurements are sufficient to recover a tensor $\mathcal{X}_0 \in \mathfrak{T}_r$ with high probability. Though the theoretical results only pertain to Gaussian measurements, our numerical experiments for tensor completion also suggests that the square reshaping model outperforms the SNN model. Compared with $\Omega(rn^{K-1})$ measurements required by the sum-of-nuclear-norms model, the sample

Table 6.1: Theoretical guarantees for different models.

Model	**sample complexity**
non-convex	$(2r)^K + 2nrK + 1$
SNN	$\Omega(rn^{K-1})$
square reshaping	$O(r^{\lfloor \frac{K}{2} \rfloor} n^{\lceil \frac{K}{2} \rceil})$

complexity, $O(r^{\lfloor \frac{K}{2} \rfloor} n^{\lceil \frac{K}{2} \rceil})$, required by the square reshaping (3.2.5), is always within a constant of it, much better for small r and $K \geq 4$. Although this is a significant improvement, compared with the nonconvex model (3.2.1), the improved sample complexity achieved by square norm model is still suboptimal. It remails an open problem to obtain near-optimal convex relaxations for all $K > 2$.

In Chapter 4, we analyzed the iteration complexity of the linearized Bregman method, which applies naturally to sparse modeling problems such as compressed sensing and matrix(tensor) completion. Specifically, we showed that for a suitably chosen step length, the method achieves a value of the Lagrangian of a quadratically regularized version of the basis pursuit problem that is within ε of the optimal value in $O(1/\varepsilon)$ iterations. We also derived an accelerated version of the linearized Bregman method whose iteration complexity is reduced to $O(1/\sqrt{\varepsilon})$. Numerical experiments results demonstrate the great improvement of the accelerated linearized Bregman method over the original method.

In Chapter 5, we extended the arguments of Chapter 4 and applied the linearized Bregman method to linear programming problems. We demonstrated by numerical experiments that the previous accelerating techniques including "Kicking" can be quite ineffective when handling linear programming problems. Inspired by the fact that the linearized Bregman method can be viewed as a sequence of box-constrained quadratic minimizations that are restricted on different supports

of the solution variable, we proposed new algorithms that combines the Conjugate Gredient/BFGS update with the linearized Bregman method. Numerical experiments on both synthetic and real data sets are conducted, and the new CGLB algorithm not only significantly outperformed the accelerated linearized Bregman proposed in Chapter 4, but also was comparable with the MATLAB solver on small-medium scale real data sets. We also established some theoretical fundamentals on combining the BFGS updates with linearized Bregman. However, the empirical performance is not quite satisfying due to some unaddressed issues such as what to do if there is a the non-empty v-null set and choosing the step size at the support transition.

Bibliography

[1] Dennis Amelunxen, Martin Lotz, Michael B McCoy, and Joel A Tropp, *Living on the edge: A geometric theory of phase transitions in convex optimization*, arXiv preprint arXiv:1303.6672 (2013).

[2] Anima Anandkumar, Rong Ge, Daniel Hsu, Sham M. Kakade, and Matus Telgarsky, *Tensor decompositions for learning latent variable models*, CoRR **abs/1210.7559** (2012).

[3] J. Barzilai and J. Borwein, *Two point step size gradient methods*, IMA Journal of Numerical Analysis **8** (1988), 141–148.

[4] A. Beck and M. Teboulle, *A fast iterative shrinkage-thresholding algorithm for linear inverse problems*, SIAM J. Imaging Sciences **2** (2009), no. 1, 183–202.

[5] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and distributed computation: numerical methods*, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1989.

[6] Dimitri P Bertsekas, *Projected newton methods for optimization problems with simple constraints*, SIAM Journal on control and Optimization **20** (1982), no. 2, 221–246.

[7] L. Bregman, *The relaxation method of finding the common points of convex sets and its application to the solution of problems in convex programming*, USSR Computational Mathematics and Mathematical Physics **7** (1967), 200–217.

[8] J. Cai, E. J. Candès, and Z. Shen, *A singular value thresholding algorithm for matrix completion*, SIAM J. on Optimization **20** (2010), no. 4, 1956–1982.

[9] J.-F. Cai, S. Osher, and Z. Shen, *Convergence of the linearized Bregman iteration for ℓ_1-norm minimization*, Mathematics of Computation **78** (2009), no. 268, 2127–2136.

[10] ———, *Linearized Bregman iterations for compressed sensing*, Mathematics of Computation **78** (2009), no. 267, 1515–1536.

[11] Jian-Feng Cai, Emmanuel J Candès, and Zuowei Shen, *A singular value thresholding algorithm for matrix completion*, SIAM Journal on Optimization **20** (2010), no. 4, 1956–1982.

[12] Emmanuel J Candès, Xiaodong Li, Yi Ma, and John Wright, *Robust principal component analysis?*, arXiv preprint arXiv:0912.3599 (2009).

[13] Emmanuel J Candès and Benjamin Recht, *Exact matrix completion via convex optimization*, Foundations of Computational mathematics **9** (2009), no. 6, 717–772.

[14] Emmanuel J. Candès, Justin K. Romberg, and Terence Tao, *Stable signal recovery from incomplete and inaccurate measurements*, Communications on Pure and Applied Mathematics **59**, no. 8.

[15] J Douglas Carroll and Jih-Jie Chang, *Analysis of individual differences in multidimensional scaling via an n-way generalization of eckart-young decomposition*, Psychometrika **35** (1970), no. 3, 283–319.

[16] Venkat Chandrasekaran, Benjamin Recht, Pablo A Parrilo, and Alan S Willsky, *The convex geometry of linear inverse problems*, Foundations of Computational Mathematics **12** (2012), no. 6, 805–849.

[17] Venkat Chandrasekaran, Sujay Sanghavi, Pablo A Parrilo, and Alan S Willsky, *Rank-sparsity incoherence for matrix decomposition*, SIAM Journal on Optimization **21** (2011), no. 2, 572–596.

[18] S. B. Cohen and M. Collins, *Tensor decomposition for fast latent-variable PCFG parsing*, Proceedings of NIPS, 2012.

[19] N.D. Sidiropoulos D. Nion, *Tensor algebra and multi-dimensional harmonic retrieval in signal processing for mimo radar*, IEEE Trans. on Signal Processing **58** (2010), no. 11, 5693–5705.

[20] Yu-Hong Dai and Roger Fletcher, *Projected barzilai-borwein methods for large-scale box-constrained quadratic programming*, Numerische Mathematik **100** (2005), no. 1, 21–47.

[21] Ron S Dembo and Ulrich Tulowitzki, *On the minimization of quadratic functions subject to box constraints*, Yale University, Department of Computer Science, 1984.

[22] D. Donoho, *Compressed sensing*, IEEE Trans. Info. Theory **52** (2006), no. 4, 1289–1306.

[23] Yonina C Eldar, Deanna Needell, and Yaniv Plan, *Unicity conditions for low-rank matrix recovery*, arXiv preprint arXiv:1103.5479 (2011).

[24] Kimon Fountoulakis and Jacek Gondzio, *A second-order method for strongly convex l1-regularization problems*, arXiv preprint arXiv:1306.5386 (2013).

[25] Thomas Franz, Antje Schultz, Sergej Sizov, and Steffen Staab, *Triplerank: Ranking semantic web data by tensor decomposition*, The Semantic Web-ISWC 2009, Springer, 2009, pp. 213–228.

[26] Ana Friedlander, José Mario Martínez, and Marcos Raydon, *A new method for large-scale box constrained convex quadratic minimization problems*, Optimization Methods and Software **5** (1995), no. 1, 57–74.

[27] N. Hao G. Ely, S. Aeron and M. E. Kilmer, *5d and 4d pre-stack seismic data completion using tensor nuclear norm (tnn)*, preprint (2013).

[28] Silvia Gandy, Benjamin Recht, and Isao Yamada, *Tensor completion and low-n-rank tensor recovery via convex optimization*, Inverse Problems **27** (2011), no. 2, 025010.

[29] D. Goldfarb and S. Ma, *Fast multiple splitting algorithms for convex optimization*, Tech. report, Department of IEOR, Columbia University. Preprint available at http://arxiv.org/abs/0912.4570, 2009.

[30] D. Goldfarb, S. Ma, and K. Scheinberg, *Fast alternating linearization methods for minimizing the sum of two convex functions*, Tech. report, Department of IEOR, Columbia University. Preprint available at http://arxiv.org/abs/0912.4571, 2010.

[31] D. Goldfarb and Z. Qin, *Robust low-rank tensor recovery: Models and algorithms*, preprint (2012).

[32] D. Goldfarb and K. Scheinberg, *Fast first-order methods for composite convex optimization with line search*, preprint (2011).

[33] D. Gross, *Recovering low-rank matrices from few coefficients in any basis*, IEEE Trans. Info. Theory **57** (2011), no. 3, 1548–1566.

[34] David Gross, *Recovering low-rank matrices from few coefficients in any basis*, Information Theory, IEEE Transactions on **57** (2011), no. 3, 1548–1566.

[35] E. T. Hale, W. Yin, and Y. Zhang, *Fixed-point continuation for ℓ_1-minimization: Methodology and convergence*, SIAM Journal on Optimization **19** (2008), no. 3, 1107–1130.

[36] Richard A Harshman, *Foundations of the parafac procedure: models and conditions for an" explanatory" multimodal factor analysis*, (1970).

[37] M. R. Hestenes, *Multiplier and gradient methods*, Journal of Optimization Theory and Applications **4** (1969), 303–320.

[38] Christopher J. Hillar and Lek-Heng Lim, *Most tensor problems are np hard*, CoRR **abs/0911.1393** (2009).

[39] Bo Huang, Shiqian Ma, and Donald Goldfarb, *Accelerated linearized bregman method*, Journal of Scientific Computing **54** (2013), no. 2-3, 428–453.

[40] Bo Jiang, Shiqian Ma, and Shuzhong Zhang, *Tensor principal component analysis via convex optimization*, arXiv preprint arXiv:1212.2702 (2012).

[41] Tamara G Kolda and Brett W Bader, *Tensor decompositions and applications*, SIAM Rev. **51** (2009), no. 3, 455–500.

[42] Stanton A. Kreimer, N. and M. D. Sacchi, *Nuclear norm minimization and tensor completion in exploration seismology*, International Conference on Acoustics, Speech and Signal Processing, 2013.

[43] Ming-Jun Lai and Wotao Yin, *Augmented l1 and nuclear-norm models with a globally linearly convergent algorithm*, arXiv preprint arXiv:1201.4615 (2012).

[44] Nan Li and Baoxin Li, *Tensor completion for on-board compression of hyperspectral images*, Image Processing (ICIP), 2010 17th IEEE International Conference on, 2010, pp. 517–520.

[45] Xiaodong Li, *Compressed sensing and matrix completion with constant proportion of corruptions*, Constructive Approximation **37** (2013), no. 1, 73–99.

[46] Yin Li, Junchi Yan, Yue Zhou, and Jie Yang, *Optimum subspace learning and error correction for tensors*, ECCV, 2010, pp. 790–803.

[47] Zhouchen Lin, Minming Chen, and Yi Ma, *The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices*, arXiv preprint arXiv:1009.5055 (2010).

[48] D. C. Liu and J. Nocedal, *On the limited memory BFGS method for large scale optimization*, Mathematical Programming, Series B **45** (1989), no. 3, 503–528.

[49] Dong C Liu and Jorge Nocedal, *On the limited memory bfgs method for large scale optimization*, Mathematical programming **45** (1989), no. 1-3, 503–528.

[50] Ji Liu, Przemyslaw Musialski, Peter Wonka, and Jieping Ye, *Tensor completion for estimating missing values in visual data*, ICCV, 2009, pp. 2114–2121.

[51] Y. Liu, D. Sun, and K.-C. Toh, *An implementable proximal point algorithmic framework for nuclear norm minimization*, To appear in Mathematical Programming (2009).

[52] S. Ma, D. Goldfarb, and L. Chen, *Fixed point and Bregman iterative methods for matrix rank minimization*, Mathematical Programming Series A **128** (2011), 321–353.

[53] N. Mesgarani, M. Slaney, and S.A. Shamma, *Discrimination of speech from nonspeech based on multiscale spectro-temporal modulations*, IEEE Trans. Audio, Speech, and Language Processing **14** (2006), no. 3, 920–930.

[54] Jorge J Moré and Gerardo Toraldo, *Algorithms for bound constrained quadratic programming problems*, Numerische Mathematik **55** (1989), no. 4, 377–400.

[55] Cun Mu, Bo Huang, John Wright, and Donald Goldfarb, *Square deal: Lower bounds and improved relaxations for tensor recovery*, arXiv preprint arXiv:1307.5870 (2013).

[56] S. Negahban, P. Ravikumar, M. Wainwright, and B. Yu, *A unified framework for high-dimensional analysis of m-estimators with decomposable regularizers*, Stat. Sci. **27** (2012), no. 4, 528–557.

[57] A. Nemirovski, *Prox-method with rate of convergence $O(1/t)$ for variational inequalities with Lipschitz continuous monotone operators and smooth convex-concave saddle point problems*, SIAM Journal on Optimization **15** (2005), no. 1, 229–251.

[58] Y. E. Nesterov, *A method for unconstrained convex minimization problem with the rate of convergence $O(1/k^2)$*, Dokl. Akad. Nauk SSSR **269** (1983), 543–547.

[59] ———, *Introductory lectures on convex optimization*, **87** (2004), xviii+236, A basic course. MR MR2142598 (2005k:90001)

[60] ———, *Smooth minimization for non-smooth functions*, Math. Program. Ser. A **103** (2005), 127–152.

[61] ———, *Gradient methods for minimizing composite objective function*, CORE Discussion Paper 2007/76 (2007).

[62] Yurii Nesterov, *A method for unconstrained convex minimization problem with the rate of convergence o (1/k2)*, Doklady AN SSSR, vol. 269, 1983, pp. 543–547.

[63] Dianne Prost O'Leary, *A generalized conjugate gradient algorithm for solving a class of quadratic programming problems*, Linear Algebra and its Applications **34** (1980), 371–399.

[64] S. Osher, M. Burger, D. Goldfarb, J. Xu, and W. Yin, *An iterative regularization method for total variation-based image restoration*, SIAM Journal on Multiscale Modeling and Simulation **4** (2005), no. 2, 460–489.

[65] S. Osher, Y. Mao, B. Dong, and W. Yin, *Fast linearized Bregman iteration for compressive sensing and sparse denoising*, Communications in Mathematical Sciences **8** (2010), no. 1, 93–?11.

[66] Konstantinos N Plataniotis and Anastasios N Venetsanopoulos, *Color image processing and applications*, Springer, 2000.

[67] M. J. D. Powell, *A method for nonlinear constraints in minimization problems*, Optimization (R. Fletcher, ed.), Academic Press, New York, 1972, pp. 283–298.

[68] Benjamin Recht, *A simpler approach to matrix completion*, arXiv preprint arXiv:0910.0651 (2009).

[69] Benjamin Recht, Maryam Fazel, and Pablo A Parrilo, *Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization*, SIAM review **52** (2010), no. 3, 471–501.

[70] R. T. Rockafellar, *Augmented Lagrangians and applications of the proximal point algorithm in convex programming*, Math. Oper. Res. **1** (1976), no. 2, 97–116. MR MR0418919 (54 #6954)

[71] R Tyrell Rockafellar, *Convex analysis*, vol. 28, Princeton university press, 1997.

[72] R.T. Rockafellar, *Convex analysis*, Princeton University Press, Princeton, 1970.

[73] O. Semerci, N. Hao, M. Kilmer, and E. Miller, *Tensor based formulation and nuclear norm regularizatin for multienergy computed tomography*, IEEE Transactions on Image Processing (2013).

[74] M. Signoretto, R. Van de Plas, B. De Moor, and J. A K Suykens, *Tensor versus matrix completion: A comparison with application to spectral data*, IEEE SPL **18** (2011), no. 7, 403–406.

[75] Marco Signoretto, Lieven De Lathauwer, and Johan AK Suykens, *Nuclear norms for tensors and their use for convex multilinear estimation*, Submitted to Linear Algebra and Its Applications **43** (2010).

[76] Marco Signoretto, Quoc Tran Dinh, Lieven Lathauwer, and JohanA.K. Suykens, *Learning with tensors: a framework based on convex optimization and spectral regularization*, Machine Learning (2013), 1–49.

[77] Jian-Tao Sun, Hua-Jun Zeng, Huan Liu, Yuchang Lu, and Zheng Chen, *Cubesvd: a novel approach to personalized web search*, Proceedings of the 14th international conference on World Wide Web, ACM, 2005, pp. 382–390.

[78] K.-C. Toh and S. Yun, *An accelerated proximal gradient algorithm for nuclear norm regularized least squares problems*, Pacific J. Optimization **6** (2010), 615–640.

[79] Ryota Tomioka, Kohei Hayashi, and Hisashi Kashima, *Estimation of low-rank tensors via convex optimization*, arXiv preprint arXiv:1010.0789 (2010).

[80] Ryota Tomioka, Taiji Suzuki, Kohei Hayashi, and Hisashi Kashima, *Statistical performance of convex tensor decomposition*, Advances in Neural Information Processing Systems (NIPS) (2011), 137.

[81] P. Tseng, *On accelerated proximal gradient methods for convex-concave optimization*, submitted to SIAM J. Optim. (2008).

[82] L. Tucker, *Some mathematical notes on three-mode factor analysis*, Psychometrika **31** (1966), no. 3, 279–311.

[83] Ledyard R Tucker, *Some mathematical notes on three-mode factor analysis*, Psychometrika **31** (1966), no. 3, 279–311.

[84] E. van den Berg and M. P. Friedlander, *Probing the Pareto frontier for basis pursuit solutions*, SIAM J. on Scientific Computing **31** (2008), no. 2, 890–912.

[85] M Alex O Vasilescu and Demetri Terzopoulos, *Multilinear subspace analysis of image ensembles*, Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on, vol. 2, IEEE, 2003, pp. II–93.

[86] Roman Vershynin, *Math 280 lectue notes*, http://www-personal.umich.edu/ romanv/teaching/2006-07/280/lec6.pdf, 2007.

[87] ———, *Introduction to the non-asymptotic analysis of random matrices*, arXiv preprint arXiv:1011.3027 (2010).

[88] Z. Wen, W. Yin, and Y. Zhang, *Solving a low-rank factorization model for matrix completion by a nonlinear successive over-relaxation algorithm*, preprint (2010).

[89] John Wright, Arvind Ganesh, Shankar Rao, Yigang Peng, and Yi Ma, *Robust principal component analysis: Exact recovery of corrupted low-rank matrices via convex optimization*, Advances in neural information processing systems **22** (2009), 2080–2088.

[90] Eugene K Yang and Jon W Tolle, *A class of methods for solving large, convex quadratic programs subject to box constraints*, Mathematical Programming **51** (1991), no. 1-3, 223–228.

[91] L. Yang, Z.-H. Huang, and X. Shi, *A fixed point iterative method for low -rank tensor pursuit*, Signal Processing, IEEE Transactions on **61** (2013), no. 11, 2952–2962.

[92] W. Yin, *Analysis and generalizations of the linearized Bregman method*, SIAM Journal on Imaging Sciences **3** (2010), no. 4, 856?877.

[93] W. Yin, S. Osher, D. Goldfarb, and J. Darbon, *Bregman iterative algorithms for ℓ_1-minimization with applications to compressed sensing*, SIAM Journal on Imaging Sciences **1** (2008), no. 1, 143–168.

[94] Wotao Yin, *Analysis and generalizations of the linearized bregman method*, SIAM Journal on Imaging Sciences **3** (2010), no. 4, 856–877.

[95] Qingshan You, Qun Wan, and Yipeng Liu, *Strongly convex programming for principal component pursuit*, arXiv preprint arXiv:1209.4405 (2012).

[96] Hui Zhang, Jian-Feng Cai, Lizhi Cheng, and Jubo Zhu, *Strongly convex programming for exact matrix completion and robust principal component analysis*, arXiv preprint arXiv:1112.3946 (2011).

Appendix A

Appendix

A.1 Proof for Theorem 2

The proof of Theorem 1 follows from a covering argument, which we establish in several steps as below.

Let

$$\mathfrak{S}_{2r} = \{\mathcal{D} \mid \mathcal{D} \in \mathfrak{T}_{2r}, \|\mathcal{D}\|_F = 1\}. \tag{A.1}$$

The following lemma shows that the required number of measurements can be bounded in terms of the exponent of the covering number for $\mathfrak{S}_{2r}$, which can be considered as a proxy for dimensionality:

Lemma 9. *Suppose that the covering number for $\mathfrak{S}_{2r}$ with respect to Frobenius norm, satisfies*

$$N(\mathfrak{S}_{2r}, \|\cdot\|_F, \varepsilon) \leq (\beta/\varepsilon)^d, \tag{A.2}$$

for some integer d and scalar β that does not depend on ε. Then if $m \geq d+1$, with probability one $\text{null}(\mathcal{G}) \cap \mathfrak{S}_{2r} = \emptyset$, *which implies that* $\text{null}(\mathcal{G}) \cap \mathfrak{T}_{2r} = \{0\}$.

Proof. The arguments we used below are primarily adapted from [23], where their interest is to establish the number of Gaussian measurements required to recover a low rank matrix by rank minimization.

Notice that every $\mathcal{D} \in \mathfrak{S}_{2r}$, and every i, $\langle \mathcal{G}_i, \mathcal{D} \rangle$ is a standard Gaussian random variable, and so

$$\forall t > 0, \quad \mathbb{P}\left[\left| \langle \mathcal{G}_i, \mathcal{D} \rangle \right| < t \right] < 2t \cdot \frac{1}{\sqrt{2\pi}} = t\sqrt{\frac{2}{\pi}}. \tag{A.3}$$

Let $\mathfrak{N}$ be an ε-net for $\mathfrak{S}_{2r}$ in terms of $\|\cdot\|_F$. Because the measurements are independent, for any fixed $\bar{\mathcal{D}} \in \mathfrak{S}_{2r}$,

$$\mathbb{P}\left[\left\| \mathcal{G}[\bar{\mathcal{D}}] \right\|_\infty < t \right] \; < \; \left(t\sqrt{2/\pi} \right)^m. \tag{A.4}$$

Moreover, for any $\mathcal{D} \in \mathfrak{S}_{2r}$, we have

$$\begin{aligned} \|\mathcal{G}[\mathcal{D}]\|_\infty &\geq \max_{\bar{\mathcal{D}} \in \mathfrak{N}} \left\{ \left\| \mathcal{G}[\bar{\mathcal{D}}] \right\|_\infty - \|\mathcal{G}\|_{F\to\infty} \left\| \bar{\mathcal{D}} - \mathcal{D} \right\|_F \right\} && \text{(A.5)} \\ &\geq \min_{\bar{\mathcal{D}} \in \mathfrak{N}} \left\{ \left\| \mathcal{G}[\bar{\mathcal{D}}] \right\|_\infty \right\} - \varepsilon \|\mathcal{G}\|_{F\to\infty}. && \text{(A.6)} \end{aligned}$$

Hence,

$$\begin{aligned}
&\mathbb{P}\left[\inf_{\mathcal{D}\in\mathfrak{S}_{2r}} \|\mathcal{G}[\mathcal{D}]\|_\infty < \varepsilon\log(1/\varepsilon)\right] \\
&\quad\leq\ \mathbb{P}\left[\min_{\mathcal{D}\in\mathfrak{N}} \|\mathcal{G}[\mathcal{D}]\|_\infty < 2\varepsilon\log(1/\varepsilon)\right] + \mathbb{P}\left[\|\mathcal{G}\|_{F\to\infty} > \log(1/\varepsilon)\right] \\
&\quad\leq\ \#\mathfrak{N}\times\left(2\sqrt{2/\pi}\times\varepsilon\log(1/\varepsilon)\right)^m + \mathbb{P}\left[\|\mathcal{G}\|_{F\to\infty} > \log(1/\varepsilon)\right] \\
&\quad\leq\ \beta^d(2\sqrt{2/\pi})^m\varepsilon^{m-d}\log(1/\varepsilon)^m + \mathbb{P}\left[\|\mathcal{G}\|_{F\to\infty} > \log(1/\varepsilon)\right]. \qquad \text{(A.7)}
\end{aligned}$$

Since $m \geq d+1$, (A.7) goes to zero as $\varepsilon \searrow 0$. Hence, taking a sequence of decreasing ε, we can show that $\mathbb{P}\left[\inf_{\mathcal{D}\in\mathfrak{S}_{2r}} \|\mathcal{G}[\mathcal{D}]\|_\infty = 0\right] \leq t$ for every positive t, establishing the result. □

Following Lemma 9, it just remains to find the covering number of $\mathfrak{S}_{2r}$. We use the following lemma, which uses the triangle inequality to control the effect of perturbations in the factors of the Tucker decomposition

$$[[\mathcal{C}; U_1, U_2, \cdots, U_K]] := \mathcal{C}\times_1 U_1 \times_2 U_2 \times_3 \cdots \times_K U_K, \qquad \text{(A.8)}$$

where the *mode-i (matrix) product* of tensor $\mathcal{A}$ with matrix B of compatible size, denoted as $\mathcal{A}\times_i B$, outputs a tensor $\mathcal{C}$ such that $\mathcal{C}_{(i)} = B\mathcal{A}_{(i)}$.

Lemma 10. *Let $\mathcal{C}, \mathcal{C}' \in \mathbb{R}^{r_1,\ldots,r_K}$, and $U_1, U_1' \in \mathbb{R}^{n_1\times r_1}, \ldots, U_K, U_K' \in \mathbb{R}^{n_K\times r_K}$ with $U_i^* U_i = U_i'^* U_i' = I$, and $\|\mathcal{C}\|_F = \left\|\mathcal{C}'\right\|_F = 1$. Then*

$$\left\|[[\mathcal{C}; U_1, \ldots, U_K]] - [[\mathcal{C}'; U_1', \ldots, U_K']]\right\|_F \ \leq\ \left\|\mathcal{C} - \mathcal{C}'\right\|_F + \sum_{i=1}^K \left\|U_i - U_i'\right\|. \qquad \text{(A.9)}$$

Proof. This follows from the basic fact that for any tensor $\mathcal{X}$ and matrix U of compatible size,

$$\|\mathcal{X} \times_k U\|_F \leq \|U\| \|\mathcal{X}\|_F, \tag{A.10}$$

which can be established by direct calculation. Write

$$\begin{aligned}
&\left\|[[\mathcal{C}; U_1, \ldots, U_K]] - [[\mathcal{C}'; U'_1, \ldots, U'_K]]\right\|_F \\
&\leq \left\|[[\mathcal{C}; U_1, \ldots, U_K]] - [[\mathcal{C}'; U_1, \ldots, U_K]]\right\|_F \\
&\quad + \left\|\sum_{i=1}^{K} [[\mathcal{C}'; U'_1, \ldots, U'_i, U_{i+1}, \ldots U_k]] - [[\mathcal{C}'; U'_1, \ldots, U'_{i-1}, U_i, \ldots U_K]]\right\|_F \\
&\leq \left\|\mathcal{C} - \mathcal{C}'\right\|_F + \sum_{i=1}^{K} \left\|U_i - U'_i\right\|,
\end{aligned}$$

where the first inequality follows from triangle inequality and the second inequality follows from the fact that $\|\mathcal{C}\|_F = 1$, $\left\|U_j\right\| = 1$, $U_i^* U_i = I$ and $U_i'^* U'_i = I$. □

Using this result, we construct an ε-net for $\mathfrak{S}_{2r}$ by building $\varepsilon/(K+1)$-nets for each of the $K+1$ factors $\mathcal{C}$ and $\{U_i\}$. The total size of the resulting ε net is thus bounded by the following lemma:

Lemma 11. $N(\mathfrak{S}_{2r}, \|\cdot\|_F, \varepsilon) \leq (3(K+1)/\varepsilon)^{(2r)^K + 2nrK}$

Proof. The idea of this proof is to construct a net for each component of the Tucker decomposition and then combine those nets to form a *compound* net with the desired cardinality.

Denote $\mathfrak{C} = \{\mathcal{C} \in \mathbb{R}^{2r \times 2r \times \cdots \times 2r} \mid \|\mathcal{C}\|_F = 1\}$ and $O = \{U \in \mathbb{R}^{n \times r} \mid U^* U = I\}$. Clearly, for any $\mathcal{C} \in \mathfrak{C}$, $\|\mathcal{C}\|_F = 1$, and for any $U \in O$, $\|U\| = 1$. Thus by Prop. 4 of [86] and Lemma 5.2 of [87], there exists an $\frac{\varepsilon}{K+1}$-net $\mathfrak{C}'$ covering $\mathfrak{C}$ with respect to the Frobenius norm such that

$\#\mathfrak{C}' \leq (\frac{3(K+1)}{\varepsilon})^{(2r)^K}$, and there exists an $\frac{\varepsilon}{K+1}$-net O' covering O with respect to the operator norm such that $\#O' \leq (\frac{3(K+1)}{\varepsilon})^{2nr}$. Construct

$$\mathfrak{S}'_{2r} = \{[[\mathcal{C}'; U'_1, \ldots, U'_K]] \mid \mathcal{C}' \in \mathfrak{C}', U'_i \in O'\}.$$

Clearly $\#\mathfrak{S}'_{2r} \leq \left(\frac{3(K+1)}{\varepsilon}\right)^{(2r)^K + 2nrK}$. The rest is to show that $\mathfrak{S}'_{2r}$ is indeed an ε-net covering $\mathfrak{S}_{2r}$ with respect to the Frobenius norm.

For any fixed $\mathcal{D} = [[\mathcal{C}; U_1, \cdots, U_K]] \in \mathfrak{S}_{2r}$ where $\mathcal{C} \in \mathfrak{C}$ and $U_i \in O$, by our constructions above, there exist $\mathcal{C}' \in \mathfrak{C}'$ and $U'_i \in O'$ such that $\left\|\mathcal{C} - \mathcal{C}'\right\|_F \leq \frac{3(K+1)}{\varepsilon}$ and $\|U_i - U'_i\| \leq \frac{3(K+1)}{\varepsilon}$. Then $\mathcal{D}' = [[\mathcal{C}'; U'_1, \cdots, U'_K]] \in \mathfrak{S}'_{2r}$ is within ε-distance from $\mathcal{D}$, since by the triangle inequality derived in Lemma 2, we have

$$\left\|\mathcal{D} - \mathcal{D}'\right\|_F = \left\|[[\mathcal{C}; U_1, \ldots, U_K]] - [[\mathcal{C}'; U'_1, \ldots, U'_K]]\right\|_F \leq \left\|\mathcal{C} - \mathcal{C}'\right\|_F + \sum_{i=1}^{K} \left\|U_i - U'_i\right\| \leq \varepsilon.$$

This completes the proof. □

With these observations in hand, Theorem 1 follows immediately.

A.2 Proof for Lemma 4

Proof. (1) By the definition of $\mathcal{X}_{[j]}$, it is sufficient to prove that the vectorization of the right hand side of (3.2.4) equals $\text{vec}(\mathcal{X}_{(1)})$.

Since $\mathcal{X} = \sum_{i=1}^{r} \lambda_i a_i^{(1)} \circ a_i^{(2)} \circ \cdots \circ a_i^{(K)}$, we have

$$\begin{aligned} \text{vec}(\mathcal{X}_{(1)}) &= \text{vec}\big(\sum_{i=1}^{r} \lambda_i a_i^{(1)} \circ (a_i^{(K)} \otimes a_i^{(K-1)} \otimes \cdots \otimes a_i^{(2)})\big) \\ &= \sum_{i=1}^{r} \lambda_i \text{vec}\big(a_i^{(1)} \circ (a_i^{(K)} \otimes a_i^{(K-1)} \otimes \cdots \otimes a_i^{(2)})\big) \\ &= \sum_{i=1}^{r} \lambda_i (a_i^{(K)} \otimes a_i^{(K-1)} \otimes \cdots \otimes a_i^{(2)} \otimes a_i^{(1)}), \end{aligned}$$

where the last equality follows from the fact that $\text{vec}(a \circ b) = b \otimes a$. Similarly, we can derive that the vectorization of the right hand side of (3.2.4),

$$\begin{aligned} & \text{vec}(\sum_{i=1}^{r} \lambda_i (a_i^{(j)} \otimes a_i^{(j-1)} \otimes \cdots \otimes a_i^{(1)}) \circ (a_i^{(K)} \otimes a_i^{(K-1)} \cdots \otimes a_i^{(j+1)})) \\ = & \sum_{i=1}^{r} \lambda_i \text{vec}\big((a_i^{(j)} \otimes a_i^{(j-1)} \otimes \cdots \otimes a_i^{(1)}) \circ (a_i^{(K)} \otimes a_i^{(K-1)} \cdots \otimes a_i^{(j+1)})\big) \\ = & \sum_{i=1}^{r} \lambda_i (a_i^{(K)} \otimes a_i^{(K-1)} \otimes \cdots \otimes a_i^{(2)} \otimes a_i^{(1)}) \\ = & \text{vec}(\mathcal{X}_{(1)}). \end{aligned}$$

Thus, equation (3.2.4) is valid.

(2) The above argument can be easily adapted to prove the second claim. Since $\mathcal{X} = \mathcal{C} \times_1$

$U_1 \times_2 U_2 \times_3 \cdots \times_K U_K$, we have

$$\begin{aligned} \text{vec}(\mathcal{X}_{(1)}) &= \text{vec}\Big(U_1 \, \mathcal{C}_{(1)} \, (U_K \otimes U_{K-1} \otimes \cdots \otimes U_2)^* \Big) \\ &= (U_K \otimes U_{K-1} \otimes \cdots \otimes U_1) \, \text{vec}(\mathcal{C}_{(1)}), \end{aligned}$$

where the last equality follows from the fact that $\text{vec}(ABC) = (C^* \otimes A)\text{vec}(B)$. Similarly, we can derive that the vectorization of the right hand side of (3.2.5),

$$\begin{aligned} & \text{vec}\Big((U_j \otimes U_{j-1} \otimes \cdots \otimes U_1) \, \mathcal{C}_{[j]} \, (U_K \otimes U_{K-1} \otimes \cdots \otimes U_{j+1})^* \Big) \\ = \;& (U_K \otimes U_{K-1} \otimes \cdots \otimes U_1) \, \text{vec}(\mathcal{C}_{[j]}) \\ = \;& (U_K \otimes U_{K-1} \otimes \cdots \otimes U_1) \, \text{vec}(\mathcal{C}_{(1)}) \\ = \;& \text{vec}(\mathcal{X}_{(1)}). \end{aligned}$$

Thus, equation (3.2.5) is valid. □

A.3 Accelerated Linearized Bregman Method

1. The proof of Theorem 10.

Proof. From (4.2.6) we get

$$G_\mu(y^k) = -\mathcal{L}_\mu(x^{k+1}, y^k). \tag{A.1}$$

By using the convexity of function $G_\mu(\cdot)$ and the Lipschitz continuity of the gradient $\nabla G_\mu(\cdot)$, we get for any y,

$$\begin{aligned} G_\mu(y^k) - G_\mu(y) &\leq G_\mu(y^{k-1}) + \langle \nabla G_\mu(y^{k-1}), y^k - y^{k-1} \rangle + \tfrac{L}{2}\|y^k - y^{k-1}\|^2 - G_\mu(y) \\ &\leq G_\mu(y^{k-1}) + \langle \nabla G_\mu(y^{k-1}), y^k - y^{k-1} \rangle + \tfrac{1}{2\tau}\|y^k - y^{k-1}\|^2 - G_\mu(y) \\ &\leq \langle \nabla G_\mu(y^{k-1}), y^{k-1} - y \rangle + \langle \nabla G_\mu(y^{k-1}), y^k - y^{k-1} \rangle \\ &\quad + \tfrac{1}{2\tau}\|y^k - y^{k-1}\|^2 \qquad \text{(A.2)} \\ &= \langle \nabla G_\mu(y^{k-1}), y^k - y \rangle + \tfrac{1}{2\tau}\|y^k - y^{k-1}\|^2 \\ &= \tfrac{1}{\tau}\langle y^{k-1} - y^k, y^k - y \rangle + \tfrac{1}{2\tau}\|y^k - y^{k-1}\|^2 \\ &\leq \tfrac{1}{2\tau}(\|y - y^{k-1}\|^2 - \|y - y^k\|^2). \end{aligned}$$

Setting $y = y^{k-1}$ in (A.2), we obtain $G_\mu(y^k) \leq G_\mu(y^{k-1})$ and thus the sequence $\{G_\mu(y^k)\}$ is non-increasing. Moreover, summing (A.2) over $k = 1, 2, \ldots, n$ with $y = y^*$ yields

$$n(G_\mu(y^n) - G_\mu(y^*)) \leq \sum_{k=1}^{n} (G_\mu(y^k) - G_\mu(y^*)) \leq \frac{1}{2\tau}(\|y^* - y^0\|^2 - \|y^* - y^n\|^2) \leq \frac{1}{2\tau}\|y^* - y^0\|^2,$$

and this implies (4.3.12). □

2. The proof of Theorem 11.

Proof. Let

$$z^k = y^{k-1} + \theta_{k-1}^{-1}(y^k - y^{k-1}) \qquad \text{(A.3)}$$

and denote the linearization of $G_\mu(y)$ as

$$l_{G_\mu}(x;y) := G_\mu(y) + \langle \nabla G_\mu(y), x-y\rangle \le G_\mu(x). \tag{A.4}$$

Therefore the second equality in (4.3.2) is equivalent to

$$\begin{aligned} y^{k+1} &:= \arg\min_y \quad G_\mu(\tilde{y}^k) + \langle \nabla G_\mu(\tilde{y}^k), y-\tilde{y}^k\rangle + \frac{1}{2\tau}\|y-\tilde{y}^k\|^2 \\ &= \arg\min_y \quad l_{G_\mu}(y;\tilde{y}^k) + \frac{1}{2\tau}\|y-\tilde{y}^k\|^2. \end{aligned}$$

Define $\hat{y}^k := (1-\theta_k)y^k + \theta_k y^*$, we have

$$\begin{aligned} G_\mu(y^{k+1}) &\le G_\mu(\tilde{y}^k) + \langle \nabla G_\mu(\tilde{y}^k), y^{k+1}-\tilde{y}^k\rangle + \frac{L}{2}\|y^{k+1}-\tilde{y}^k\|^2 \\ &\le l_{G_\mu}(y^{k+1};\tilde{y}^k) + \frac{1}{2\tau}\|y^{k+1}-\tilde{y}^k\|^2 \\ &\le l_{G_\mu}(\hat{y}^k;\tilde{y}^k) + \frac{1}{2\tau}\|\hat{y}^k-\tilde{y}^k\|^2 - \frac{1}{2\tau}\|\hat{y}^k-y^{k+1}\|^2 \\ &= l_{G_\mu}((1-\theta_k)y^k+\theta_k y^*;\tilde{y}^k) + \frac{1}{2\tau}\|(1-\theta_k)y^k+\theta_k y^*-\tilde{y}^k\|^2 \\ &\quad - \frac{1}{2\tau}\|(1-\theta_k)y^k+\theta_k y^*-y^{k+1}\|^2 \\ &= l_{G_\mu}((1-\theta_k)y^k+\theta_k y^*;\tilde{y}^k) + \frac{\theta_k^2}{2\tau}\|y^*+\theta_k^{-1}(y^k-\tilde{y}^k)-y^k\|^2 \\ &\quad - \frac{\theta_k^2}{2\tau}\|y^*+\theta_k^{-1}(y^k-y^{k+1})-y^k\|^2 \\ &= l_{G_\mu}((1-\theta_k)y^k+\theta_k y^*;\tilde{y}^k) + \frac{\theta_k^2}{2\tau}\|y^*-z^k\|^2 - \frac{\theta_k^2}{2\tau}\|y^*-z^{k+1}\|^2 \\ &= (1-\theta_k)l_{G_\mu}(y^k;\tilde{y}^k) + \theta_k l_{G_\mu}(y^*;\tilde{y}^k) + \frac{\theta_k^2}{2\tau}\|y^*-z^k\|^2 - \frac{\theta_k^2}{2\tau}\|y^*-z^{k+1}\|^2 \\ &\le (1-\theta_k)G_\mu(y^k) + \theta_k G_\mu(y^*) + \frac{\theta_k^2}{2\tau}\|y^*-z^k\|^2 - \frac{\theta_k^2}{2\tau}\|y^*-z^{k+1}\|^2, \end{aligned} \tag{A.5}$$

where the second inequality is from (A.4) and $\tau \leq 1/L$, the third inequality uses Lemma 7 with $\psi(x) := \tau l_{G_\mu}(x; \tilde{y}^k)$, the third equality uses (A.3), (4.3.2) and (4.3.13) and the last inequality uses (A.4).

Therefore we get

$$\frac{1}{\theta_k^2}(G_\mu(y^{k+1}) - G_\mu(y^*)) \leq \frac{1-\theta_k}{\theta_k^2}(G_\mu(y^k) - G_\mu(y^*)) + \frac{1}{2\tau}\|y - z^k\|^2 - \frac{1}{2\tau}\|y - z^{k+1}\|^2.$$

From (4.3.14), it is easy to show that $\frac{1-\theta_k}{\theta_k^2} \leq \frac{1}{\theta_{k-1}^2}$ for all $k \geq 0$. Thus (A.5) implies that

$$\frac{1-\theta_{k+1}}{\theta_{k+1}^2}(G_\mu(y^{k+1}) - G_\mu(y^*)) \leq \frac{1-\theta_k}{\theta_k^2}(G_\mu(y^k) - G_\mu(y^*)) + \frac{1}{2\tau}\|y - z^k\|^2 - \frac{1}{2\tau}\|y - z^{k+1}\|^2. \quad \text{(A.6)}$$

Summing (A.6) over $k = 0, 1, \ldots, n-1$, we get

$$\frac{1-\theta_n}{\theta_n^2}(G_\mu(y^n) - G_\mu(y^*)) \leq \frac{1}{2\tau}\|y^* - z^0\|^2 = \frac{1}{2\tau}\|y^* - y^0\|^2,$$

which immediately implies (4.3.15). □

A.4 Accelerated linearized Bregman algorithm for low-rank tensor recovery problems

In the section, we will address our numerical algorithms for solving optimization problems such as (3.3.1) and (3.3.2). For model (3.3.1)

$$\text{minimize}_{\mathcal{X}} \quad \sum_{i=1}^{K} \|\mathcal{X}_{(i)}\|_* \quad \text{subject to} \quad \mathcal{P}_\Omega[\mathcal{X}] = \mathcal{P}_\Omega[\mathcal{X}_0]. \tag{A.1}$$

By introducing auxiliary variable $\mathcal{W}$ and splitting $\mathcal{X}$ into $\mathcal{X}_1, \mathcal{X}_2, \cdots, \mathcal{X}_K$, it can be easily verified that problem (A.1) is equivalent to

$$\begin{aligned} \min_{(\{\mathcal{X}_i\},\mathcal{W})} \quad & \sum_{i=1}^{K} \|(\mathcal{X}_i)_{(i)}\|_* \\ \text{s.t.} \quad & \mathcal{X}_i = \mathcal{W}, \quad i = 1,2,\cdots,K, \\ & \mathcal{P}_\Omega[\mathcal{W}] = \mathcal{P}_\Omega[\mathcal{X}_0], \end{aligned} \tag{A.2}$$

whose objective function is now separable.

The accelerated linearized Bregman (ALB) algorithm, proposed in [39], is an efficient first-order method designed for solving convex optimization problems with nonsmooth objective functions and linear constraints. It has been successfully applied to solve ℓ^1 and nuclear norm minimization problems [39]. The ALB algorithm solves nonsmooth problem by firstly smoothing the objective function (e.g. adding a small l_2 perturbation), and then exploiting Nesterov's accelerated scheme [62] to the dual problem, which can be verified to be unconstrained and Lipschitz differ-

Algorithm A.1 Accelerated linearized Bregman algorithm for SNN model (5.1)

Initialization: $\mathcal{Y}_i^0 = \tilde{\mathcal{Y}}_i^0 = 0$ for each $i \in [K]$, $\mathcal{Z}^0 = \tilde{\mathcal{Z}}^0 = 0$, $\mu > 0$, $\tau > 0$, $t_0 = 1$;
for $k = 0,\ 1,\ 2,\ \cdots$ **do**
  for $i = 1,\ 2,\ \cdots,\ K$ **do**
    $\mathcal{X}_i^{k+1} = \mu \cdot \text{Shrinkage}(\mathcal{Y}_i^k, 1)$;
  end for
  $\mathcal{W}^{k+1} = \mu \cdot \left(\mathcal{P}_\Omega\left[\mathcal{Z}^k\right] - \Sigma_i \mathcal{Y}_i^k\right)$;
  for $i = 1,\ 2,\ \cdots,\ K$ **do**
    $\tilde{\mathcal{Y}}_i^k = \mathcal{Y}_i^k - \tau \cdot \left(\mathcal{X}_i^{k+1} - \mathcal{W}^{k+1}\right)$;
  end for
  $\tilde{\mathcal{Z}}^k = \mathcal{Z}^k - \tau \cdot \mathcal{P}_\Omega\left[\mathcal{W}^{k+1} - \mathcal{X}_0\right]$;
  $t_{k+1} = \frac{1+\sqrt{1+4t_k^2}}{2}$;
  for $i = 1,\ 2,\ \cdots,\ K$ **do**
    $\mathcal{Y}_i^{k+1} = \tilde{\mathcal{Y}}_i^k + \frac{t_k - 1}{t_{k+1}}\left(\tilde{\mathcal{Y}}_i^k - \tilde{\mathcal{Y}}_i^{k-1}\right)$;
  end for
  $\mathcal{Z}^{k+1} = \tilde{\mathcal{Z}}^k + \frac{t_k - 1}{t_{k+1}}\left(\tilde{\mathcal{Z}}^k - \tilde{\mathcal{Z}}^{k-1}\right)$;
end for

entiable. In Algorithm A.1, we describe our ALM algorithm adapted to problem (A.2). Algorithm A.1 solves exactly the smoothed version of problem (A.2):

$$\begin{aligned} \min_{(\{\mathcal{X}_i\},\mathcal{W})} \quad & \sum_{i=1}^{K}\left(\|(\mathcal{X}_i)_{(i)}\|_* + \frac{1}{2\mu}\|(\mathcal{X}_i)_{(i)}\|_F^2\right) + \frac{1}{2\mu}\|\mathcal{W}\|_F^2 \\ \text{s.t.} \quad & \mathcal{X}_i = \mathcal{W}, \quad i = 1,2,\cdots,K, \\ & \mathcal{P}_\Omega[\mathcal{W}] = \mathcal{P}_\Omega[\mathcal{X}_0], \end{aligned} \tag{A.3}$$

where we denote $\mathcal{Y}_i$ as the dual variable for the constraint $\mathcal{X}_i = \mathcal{W}$ and denote $\mathcal{Z}$ as the dual variable for the last constraint $\mathcal{P}_\Omega[\mathcal{W}] = \mathcal{P}_\Omega[\mathcal{X}_0]$. Since the objective function in (A.3) is separable, each setup of the ALB algorithm is easy to solve as we can see from Algorithm A.1 [1].

[1] The Shrinkage operator in line 4 of Algorithm A.1 performs the regular shrinkage on the singular values of the ith unfolding matrix of $\mathcal{Y}_i^k$, i.e. $(\mathcal{Y}_i^k)_{(i)}$, and then folds the resulting matrix back into tensor.

For our numerical experiment ($K = 4$), we choose smoothing parameter $\mu = 50\|\mathcal{X}_0\|_F$ and step size $\tau = \frac{1}{5\mu}$. Empirically, we observe that larger values of μ do not result in a better recovery performance. This is consistent with the theoretical results established in [43, 96].