Optimization for High-Dimensional Data Analysis

by

Derek T. Driggs

B.S., University of Colorado Boulder, 2017

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Master of Science

Department of Applied Mathematics

2017

This thesis entitled:
Optimization for High-Dimensional Data Analysis
written by Derek T. Driggs
has been approved for the Department of Applied Mathematics

_____

Prof. Stephen Becker

_____

Prof. Lijun Chen

_____

Prof. Jem Corcoran

Date _____

The final copy of this thesis has been examined by the signatories, and we find that both the content and the form meet acceptable presentation standards of scholarly work in the above mentioned discipline.

Driggs, Derek T. (M.S., Applied Mathematics)

Optimization for High-Dimensional Data Analysis

Thesis directed by Prof. Stephen Becker

As modern datasets continue to grow in size, they are also growing in complexity. Data are more often being recorded using multiple sensors, creating large, multidimensional datasets that are difficult to analyze. In this thesis, we explore methods to accelerate low-rank recovery algorithms for data analysis, with an emphasis on Robust Principal Component Analysis (RPCA). We also develop a tensor-based approach to RPCA that preserves the inherent structure of multidimensional datasets, allowing for improved analysis. Both of our approaches use nuclear-norm regularization with Burer-Monteiro factorization (or higher-order generalizations thereof) to transform convex but expensive programs into non-convex programs that can be solved efficiently. We supplement our non-convex programs with a certificate of optimality that can be used to bound the suboptimality of each iterate. We demonstrate that both of these approaches allow for new applications of RPCA in fields involving multidimensional datasets; for example, we show that our methods can be used for real-time video processing as well as the analysis of fMRI brain-scans. Traditionally, these tasks have been considered too demanding for low-rank recovery algorithms.

Dedication

I would like to dedicate this work to all my family members, friends, instructors, mentors, and advisors who showed me the excitement in even the smallest discoveries.

Acknowledgements

Foremost, I would like to acknowledge my advisor, Prof. Stephen Becker, for introducing me to what has become my field of research. Every hour of work I put into this thesis was matched by two hours of Prof. Becker's guidance.

I would also like to acknowledge Prof. Aleksandr Aravkin from the Applied Mathematics department of the University of Washington for the work he contributed as coauthor to the paper that has resulted from the first chapter of this thesis. I have learned much from Prof. Aravkin's instruction as well, and his insights have significantly improved the quality of this thesis.

Finally, I would like to thank Prof. Tor Wager and Dr. Stephan Geuter from the Neuroscience department of the University of Colorado Boulder for providing the fMRI data that appears in this work.

Contents

Chapter

List of Tables

Table

List of Figures

Figure

Introduction

A single MRI scan of a patient's brain produces over fifty-million data points, representing four dimensions of neurological activity. These images contain a wealth of knowledge that could lead to a diagnosis, improved treatments, or a better understanding of the human brain, but much of this information is clouded by noise. Problems like these are ubiquitous, accompanying large-scale data analysis in fields as diverse as machine learning, genome sequencing, and video processing. Low-rank recovery algorithms attempt to extract patterns and detect outliers in these data sets; representing the data as a matrix and using the well-studied tools of linear algebra to decompose the matrix into meaningful components. Robust Principal Component Analysis (RPCA) is a famous example of these matrix-decompositional algorithms, and since its introduction in 2009, it has become widely used as a method for natural language processing, facial recognition, and background subtraction, just to name a few of its numerous applications [17].

While matrix-based models are powerful tools for data analysis, they are being increasingly challenged by the growing size and complexity of modern datasets. Less than a decade earlier, a dataset would generally contain information from a single sensor, or a single spatial dimension, or a single trial of an experiment [58]. Today, it is easy for researchers to collect and store datasets from multiple sources on a single machine. A single fMRI scan, for example, records information for three dimensions of space over time, yielding an enormous four-dimensional dataset. It is natural to expect data-analytic algorithms to progress to be able to use the structure of these multidimensional datasets to their advantage and discover deeper patterns in the data. However, this is currently not the case.

The size of these multidimensional datasets makes matrix-based methods, which are often notoriously slow [27], impractical. Furthermore, all matrix-based algorithms inherently assume that the data is best represented in a two-dimensional structure, which is often untrue. In this thesis, we explore solutions to both of these problems. In Chapter 1, we present a general method for adapting low-rank recovery algorithms, such as RPCA, to take advantage of parallelized computational architectures. GPU and multiple-CPU architectures are becoming prolific and popular throughout the data analysis community, and we show that their use can accelerate matrix-based models enough to expand their domain of application. Specifically, we show how our approach can allow algorithms like RPCA to be applied to decompose fMRI datasets and perform real-time background subtraction in video streams. Before this work, matrix-based algorithms were often seen as too slow to be applied to problems involving datasets as large as an fMRI brain scan, or problems requiring such speed as real-time video processing.

In Chapter 2, we address the problem of data's increasing dimensionality. The success of matrices in data analysis has been largely due to their extensive theoretical support using results from linear algebra. Higher-order matrices, or tensors, do not enjoy the same strong theoretical foundations, so many data scientists have been reluctant to generalize matrix models to work with high-dimensional datasets. Instead, it is more common to "unwrap" the data, force it into a two-dimensional structure, and use traditional matrix-based algorithms for analysis. We present a tensor-based approach to RPCA, and supplement this approach with theoretical performance guarantees. This analysis generalizes many popular tools in matrix-based analysis to higher orders, including nuclear-norm regularization, Burer-Monteiro factorization, and an alternating-minimization algorithm to fit our factorized model. We provide both theoretical and experimental evidence to show that tensor-based algorithms that preserve the structure of high-dimensional datasets are much more effective than models that force the data into two-dimensional matrices.

Our results argue for the pursuit of several new directions of research. Matrices benefit from a strong theoretical foundation in linear algebra, but tensors do not have this privilege. Most theoretical support for tensors is obtained from studying their behavior after they are "unwrapped"

into matrices. Our results demonstrate that the future of multidimensional algorithms relies on the direct analysis of tensors as multidimensional structures.

Chapter 1

Adapting Regularized Low Rank Recovery Algorithms for Parallel Architectures

Chapter Summary

We introduce two new methods to parallelize low rank recovery models in order to take advantage of GPU, multiple CPU, and hybridized architectures. Using Burer-Monteiro splitting and marginalization, we develop a smooth, non-convex formulation of regularized low rank recovery models that can be solved with first-order optimizers. Using L-BFGS on the GPU, the resulting non-convex programs offer enormous speedup over existing methods.

Our second development is a communication-avoiding implementation of the randomized singular value decomposition (rSVD) that ameliorates the heavy communication accompanying tall-skinny matrix structures that arise in intermediate calculations of low rank recovery problems. We demonstrate, using synthetic data, surveillance video, and data from fMRI brain scans, that both algorithms offer significant speedup over traditional low rank recovery models.

1.1 Introduction

Low rank matrix decompositions are an effective tool for large-scale data analytics. Background subtraction, facial recognition, document indexing, and collaborative filtering all use low rank matrix recovery algorithms [17]. However, a low rank structure is only a part of the story — signals often exhibit additional structure. *Regularized* low rank recovery models seek to decompose a matrix $X \in \mathbb{R}^{m\times n}$ into the sum of a low rank component $L \in \mathbb{R}^{m\times n}$ and another well-structured

matrix $S \in \mathbb{R}^{m\times n}$, where S could be sparse, clustered, non-negative, or have some other useful attribute (see e.g. [69]).

Searching for sparse and low rank models gives rise to combinatorial formulations, which are intractable. Fortunately, relaxations often work well, and are amenable to modern optimization techniques. Consider finding an approximate decomposition of a data matrix X into a sum $X = L + S$, where S is sparse and L low rank. We are given partial linear measurements $b = \mathcal{A}(X)$. We can formulate the problem

$$\min_{L,S} \quad \text{rank}(L) + \text{card}(S), \quad \text{subject to:} \quad \frac{1}{2}\|\mathcal{A}(L+S) - b\|_2^2 \leq \epsilon, \tag{1.1}$$

where the cardinality function $\text{card}(\cdot)$ counts the number of nonzero entries of S, and the parameter ϵ accounts for noise or model inconsistencies. Rank is analogous to sparsity, since it is precisely the $\text{card}(\cdot)$ function applied to the singular values of the argument. Problem (1.1) is NP-hard [55] and intractable for large problem sizes. However, both terms admit nonsmooth convex relaxations. $\text{card}(\cdot)$ can be replaced with the ℓ_1-norm, and the rank function can be replaced with the nuclear norm, $\|\cdot\|_*$, which is equal to the sum of the singular values of the argument:

$$\min_{L,S} \quad \|L\|_* + \|S\|_1, \quad \text{subject to:} \quad \frac{1}{2}\|\mathcal{A}(L+S) - b\|_2^2 \leq \epsilon \tag{1.2}$$

When the parameter ϵ is set to zero, (1.2) is called *robust PCA* (RPCA), and can recover the lowest-rank L and the sparsest S under mild conditions. Setting $\epsilon > 0$ in (1.2) is more suitable for most applications, since it allows for noise in the measurements; the problem is then called *stable principal component pursuit* (SPCP) [73, 77]. SPCP stably recovers L and S with error bounded by ϵ [77]. Both RPCA and SPCP are convex problems, much easier to solve than the original problem (1.1). While (1.2) is an important problem, it is just one example of regularized low rank models.

Problem class. We are interested in general regularized low rank approaches:

$$\min_{L,S} \quad \|L\|_* + \mathcal{L}(b, \mathcal{A}(L+S)) + r(S), \tag{1.3}$$

where $\mathcal{L}$ is a loss function and r is a convex regularizer. $\mathcal{L}$, which can be *infinite-valued*, measures how well $L+S$ agrees with measured values of X. In RPCA, $\mathcal{L}(v) = \delta_b(v)$ is the indicator function ensuring we match observed data, while in SPCP, $\mathcal{L}(v) = \delta_{2\sqrt{\epsilon}\mathbb{B}}(v-b)$ ensures we are close to b. $\mathcal{L}$ can also be finite valued — for example, a modified SPCP is given by solving

$$\min_{L,S} \quad \lambda_L \|L\|_* + \frac{1}{2}\|\mathcal{A}(L+S) - b\|_2^2 + \lambda_S \|S\|_1, \tag{1.4}$$

where λ_L, λ_S are tuning parameters, and $\mathcal{L}(\cdot) = \frac{1}{2}\|\cdot\|_2^2$.

The regularizer $r(S)$ is a convex penalty that promotes desired structure of S. The ℓ_1 norm $r(S) = \|S\|_1$ promotes sparsity in S; other important examples include the ordered-weighted ℓ_1 norm (OWL norm) ([76, 25, 13], and the elastic net [78], which both enforce sparsely correlated observations (clusters) in S. Other convex regularizers promote known group-sparse and hierarchical structures in S; see [7] for a survey.

Marginalization in S. In all of the applications we consider, $r(S)$ is simple enough that we can efficiently minimize (1.3) in S for a given L. In the motivating SPCP example (1.4), we have

$$\min_S \left\{ \lambda_L \|L\|_* + \frac{1}{2}\|\mathcal{A}(L+S) - b\|_2^2 + \lambda_S \|S\|_1 \right\} = \lambda_L \|L\|_* + \rho_{\lambda_S}(\mathcal{A}(L) - b),$$

where ρ_{λ_S} is the Huber function (for a detailed derivation, see e.g. [6]). This motivates the definition

$$\varphi : L \mapsto \min_S \left\{ \mathcal{L}(\mathcal{A}(L+S) - b) + \lambda_S \cdot r(S) \right\}, \tag{1.5}$$

where φ is smooth (see section 1.3.2) with value and derivative either explicitly available as in the case of SPCP, or efficiently computable. After marginalizing out S, (1.3) becomes

$$\min_L \ \lambda_L \|L\|_* + \varphi(L). \tag{1.6}$$

Proximal gradient methods are applicable, but require computing the expensive singular value decomposition (SVD) at each iteration. To avoid this bottleneck, we develop and compare two alternatives for accelerating the method: parallelizing SVD computations using the tall-skinny QR algorithm, and non-convex factorized representations that avoid the SVD entirely. The smooth

structure of φ allows a computable *optimality certificate* for the non-convex case, which is developed and explored in section 1.3.

Accelerating SVD computation. The SVD is notoriously costly to compute, and its high communication costs make it difficult to parallelize. Making matters worse, in many applications the data matrix has many more rows than columns, and these "tall-skinny" matrices make the communication costs of a parallel SVD much higher.

To accelerate the SVD step, the randomized SVD (rSVD) [36] is often used to efficiently compute a partial SVD (see e.g. [4]). While the rSVD algorithm requires fewer operations than traditional SVDs, intermediate steps of the algorithm exacerbate the tall-skinny problem, which makes the rSVD even more difficult to parallelize. In order to use parallelized architectures such as the GPU to accelerate regularized low rank recovery, the communication cost of the SVD step must be reduced. To do this, we use the tall-skinny QR algorithm (TSQR) of [26]. [3] explored using TSQR to accelerate the SVD step of RPCA for tall-skinny matrices, but we address an inherent communication problem in the rSVD algorithm that arises even for square matrices.

Non-convex reformulation. To avoid the SVD entirely, we consider non-convex reformulations for regularized low rank recovery models, using the factorization $L = UV^T$, where $U \in \mathbb{R}^{m\times k}$, $V \in \mathbb{R}^{n\times k}$, and $k \ll m, n$. This factorization imposes the rank-constraint $\text{rank}(L) \leq k$. In addition, we can further penalize rank using the representation (see e.g. [64, 59, 8])

$$\|L\|_* \equiv \inf_{L=UV^T} \frac{1}{2}(\|U\|_F^2 + \|V\|_F^2). \tag{1.7}$$

This equivalence only holds if the ranks of U and V are high enough, and makes it possible to maintain a low rank $L = UV^T$ without requiring SVDs. The non-convex analogue to (1.6) is given by

$$\arg\min_{U,V} \quad \varphi(UV^T) + \frac{\lambda_L}{2}(\|U\|_F^2 + \|V\|_F^2). \tag{1.8}$$

Roadmap. We start with a survey of related work in section 1.2. We then consider two theoretical issues related to the smooth reformulation (1.8) in section 1.3 . First, using the factorization $L = UV^T$ gives a non-convex program that could potentially create new stationary

points and local minima. Second, it is important to understand the properties of the map φ obtained by marginalizing over S in the context of specific optimization algorithms. In section 1.4, we study heuristic initialization methods, which are important for non-convex formulations. We present detailed numerical studies, comparing the non-convex method with competing alternatives, in section 1.5. The convex formulation (1.6) and TSQR method are presented in section 1.6, along with additional numerical experiments, illustrating the parallelism of TSQR and comparing it to the non-convex method. We end with final thoughts in section 1.7.

1.2 Prior Work and Contributions

Several authors have used alternating minimization to solve non-convex, factorized matrix completion models [38, 45, 46, 47, 48, 35, 43]. This line of research is supported by the existence of conditions that guarantee fast convergence to a globally optimal solution [38, 46, 48, 47, 35], sometimes at a linear rate [43, 69]. These models are a special class of (1.8), when $r(S) = 0$.

In [69], the authors review the use of alternating minimization to solve regularized PCA problems of the following form:

$$\min_{U,V} \quad \mathcal{L}(UV^T, X) + r(U) + \tilde{r}(V). \tag{1.9}$$

Matrix completion models and RPCA can adopt the split-form in (1.9), but SPCP and other regularized models cannot. Our analysis includes more general regularizers and offers an alternative to alternating minimization.

[62] develop a split RPCA program and solve it using alternating minimization. While their technique can be an order of magnitude faster than competitive convex solvers (such as the inexact augmented Lagrangian method (IALM) from [51]), its performance suffers when the magnitude of the sparse component is large compared to the magnitude of the low rank component.

[67] follow techniques introduced in [18] and consider provable recovery of a low rank solution to a system of linear equations using a non-convex, factorized program. In [74], the authors extend these ideas to investigate a non-convex formulation of robust PCA that can be solved with gradient

descent.

Another non-convex approach to RPCA is considered in [57]. In contrast to the research discussed previously, their method does not use a low rank factorization of L. Instead, these authors develop an algorithm of alternating projections, where each iteration sequentially projects L onto a set of low rank matrices and thresholds the entries of S. Their technique also comes with recovery and performance guarantees.

We develop a general non-convex formulation for regularized low rank recovery models that can be solved with general first-order optimization algorithms. We address some problems associated with local minima and spurious stationary points that accompany non-convexity, by showing that all local minima of our model have the same objective value, and by providing a certificate to prove that a given solution is not a spurious stationary point. We also show that our method is particularly well-suited for the GPU.

We also develop a communication-efficient solver for convex models by limiting the communication costs in the rSVD, using TSQR to replace the orthogonalization step. [3] briefly discusses using TSQR to accelerate the full SVD step in RPCA, but our rSVD algorithm offers greater speedup and applies to general matrices, not just tall and skinny matrices. In [53], the authors propose a blocked algorithm to accelerate the rSVD. Their blocking scheme reduces communication costs in a way similar to TSQR, but TSQR also offers additional parallelism. Both of our approaches offer significant speedup over existing methods, as illustrated in section 1.5.

1.3 Theoretical Considerations

We discuss three issues related to the approach (1.8). First, we study the convexity, smoothness, and differentiability of the marginal function (1.6) in section 1.3.2. Next, moving from (1.6) to (1.8) by factorizing L, we transform a convex problem into a non-convex one. This creates two issues: an implicit rank constraint $\text{rank}(L) \leq k$ that is not present in (1.8), and potential for local minima and spurious stationary points in the non-convex variant that do not correspond to a solution of the convex problem. We address these issues in section 1.3.3.

Section 1.3.4 introduces a computable certificate that can be used to check whether the non-convex model has converged to a global solution. Using the smoothness of φ, we derive a simple optimality certificate that can be tracked while optimizing (1.8) to identify either when the implicit rank of the factors is too small or when we have converged to a minimizer of (1.6).

1.3.1 Convexity of φ

The convexity of the marginal function (1.6) is a well-known result in convex analysis [60, Proposition 2.22]. We include a short, self-contained proof of this fact.

Lemma 1. *Let $f : \mathbb{R}^{m\times n} \times \mathbb{R}^{m\times n} \to \mathbb{R}$ be a jointly convex function, and let $L \in \mathbb{R}^{m\times n}$. The function $g : \mathbb{R}^{m\times n} \to \mathbb{R} : L \mapsto \min_{S\in\mathbb{R}^{m\times n}} f(L, S)$ is convex.*

Proof. For all $L_1, L_2, S_1, S_2 \in \mathbb{R}^{m\times n}$ and $t \in (0, 1)$,

$$\begin{aligned} g(tL_1 + (1-t)L_2) &= \min_{S\in\mathbb{R}^{m\times n}} f(tL_1 + (1-t)L_2, S) \\ &\leq f(tL_1 + (1-t)L_2, tS_1 + (1-t)S_2) \\ &\leq tf(L_1, S_1) + (1-t)f(L_2, S_2). \end{aligned}$$

As this inequality holds for arbitrary S_1 and S_2, it holds for any minimizing $S^\star$:

$$\begin{aligned} g(tL_1 + (1-t)L_2) &\leq tf(L_1, S^\star) + (1-t)f(L_2, S^\star) \\ &= tg(L_1) + (1-t)g(L_2). \end{aligned}$$

□

It follows immediately that φ as defined in (1.5) is convex when $\mathcal{L}$ and r are convex.

1.3.2 Smoothness of φ

When $\mathcal{L}$ is the least-squares error and

$$\mathcal{A}(L + S) = \mathcal{A}_L(L) + \mathcal{A}_S(S),$$

with $\mathcal{A}_S$ an injective linear operator, Lipschitz differentiability of φ is immediate from the strong convexity of $\mathcal{L}$. Support for this claim is given, for example, in [9], where the relevant proposition for the specific case where $\mathcal{A}_S$ is a reshaping operator is as follows:

Proposition ***[9, Prop. 12.29]:*** *Let r be a proper lower-semicontinuous convex function, and let ${}^{\mu}r(\mathcal{A}(L) - \mathrm{vec}(S) - b) = \inf_S \; r(S) + \frac{1}{2\mu}\|\mathcal{A}(L) - \mathrm{vec}(S) - b\|_2^2$ be the Moreau envelope of f with parameter μ. Then ${}^{\mu}r$ is differentiable and its gradient is μ^{-1}-Lipschitz continuous.*

More generally, $\varphi(L)$ is differentiable as long as $r(S)$ has a unique minimizer and the loss function $\mathcal{L}$ is convex and smooth. By [60, Theorem 10.58], we have

$$\partial\varphi(\cdot) = \mathcal{A}^*\nabla\mathcal{L}\left(\mathcal{A}(\cdot) - b - \mathcal{A}(S^\star)\right), \tag{1.10}$$

where $S^\star$ is any minimizer of (1.4). The subdifferential is a singleton if and only if the minimizing $S^\star$ is unique, and a unique minimum is guaranteed if the sum $\mathcal{L}(\mathcal{A}(L+S) - b) + r(S)$ in (1.4) is strictly convex.

Using SPCP as an example, we have

$$\varphi(L) = \min_S \left\{ \frac{1}{2}\|\mathcal{A}(L) + \mathrm{vec}(S) - b\|_2^2 + \lambda_S\|S\|_1 \right\},$$

where sparse outliers S are fully observed. The objective is strongly convex in S, and has the unique minimizer

$$S^\star = \mathrm{prox}_{\lambda_S\|\cdot\|_1}(\mathcal{A}(L) - b).$$

Therefore, $\varphi(L)$ is smooth, with

$$\nabla\varphi(L) = \mathcal{A}^*\left(\mathcal{A}(L) + \mathrm{vec}(S^\star) - b\right).$$

Note we used (1.10) rather than the parametric form of $\varphi(L)$ to obtain its value and gradient. Using (1.7) to replace the nuclear norm and replacing the non-smooth regularizer with φ, we have created a smooth optimization problem amenable to first-order methods.

For minimizing smooth objectives, quasi-Newton methods are often faster than algorithms that require only first-order smoothness, such as gradient-descent. The convergence rates of quasi-Newton methods depend on second-order smoothness, which does not hold in problems such as

SPCP, since e.g. the Huber function is only $\mathcal{C}^1$. However, empirical results (including those presented in section 1.5) suggest that these methods are still effective.

1.3.3 Rank and Local Minima

The factorized problem (1.8) is not equivalent to (1.6) because of the implicit rank constraint. However, we can show that (1.8) is equivalent to the rank-constrained problem

$$\min_{L} \quad \lambda_L \|L\|_* + \varphi(L) \quad \text{subject to:} \quad \operatorname{rank}(L) \leq k. \tag{1.11}$$

This follows immediately from (1.7). In order for the split problem to recover the solution to (1.1), it must be initialized with a k larger than the rank of the minimizing L. However, a larger k slows computation, so it must be chosen with care. This issue is considered in more depth in section 1.4.

We want to be sure that any local minimum of (1.8) corresponds to a (global) minimum of (1.6). The following theorem combines ideas from [15] and [5] to show this holds provided that k is larger than the rank of the minimizer for (1.6).

Theorem 1. *Consider an optimization problem of the following form:*

$$\min_{X \succeq 0} \quad f(X), \quad \textit{such that} \quad \operatorname{rank}(X) \leq k, \tag{1.12}$$

where $X \in \mathbb{R}^{n \times n}$ is a positive semidefinite real matrix, and f is a lower semi-continuous function mapping to $[-\infty, \infty]$ and has a non-empty domain over the set of positive semi-definite matrices. Using the change of variable $X = PP^T$, take $P \in \mathbb{R}^{n \times k}$, and consider the problem

$$\min_{P} \quad g(P) \stackrel{def}{=} f(PP^T). \tag{1.13}$$

Let $\bar{X} = \bar{P}\bar{P}^T$, where $\bar{X}$ is feasible for (1.12). Then $\bar{X}$ is a local minimizer of (1.12) if and only if $\bar{P}$ is a local minimizer of (1.13).

The proof of Theorem 1 is deferred to Appendix A. Using the SDP formulation of the nuclear norm presented in [28], our problem can be recast as a semi-definite program so that we can apply

Theorem 1. Define

$$Z = \begin{bmatrix} U \\ V \end{bmatrix} \begin{bmatrix} U \\ V \end{bmatrix}^T = \begin{bmatrix} UU^T & L \\ L^T & VV^T \end{bmatrix}. \tag{1.14}$$

The matrix Z is positive semi-definite, and has form $Z = PP^T$, with $P = \begin{bmatrix} U \\ V \end{bmatrix}$. Let $\mathcal{R}(\cdot)$ be the function that extracts the upper-right block of a matrix (so that $\mathcal{R}(Z) = L$), and let

$$f(Z) = \frac{\lambda_L}{2} \operatorname{trace}(Z) + \varphi(\mathcal{R}(Z)). \tag{1.15}$$

Using equation (3), we now see that the rank-constrained problem is equivalent to

$$\min_{Z \succeq 0} \quad f(Z), \quad \text{such that} \quad \operatorname{rank}(Z) \leq k. \tag{1.16}$$

Applying Theorem 1, we can now be assured that $\overline{P}$ is a local minimizer to the split program if and only if $\overline{Z}$ is local minimizer of the original problem. This is equivalent to the statement that the point $(\overline{U}, \overline{V})$ is a local minimizer of the split problem if and only if $\overline{L} = \overline{U}\ \overline{V}^T$ is a local minimizer of the original, rank-constrained program.

1.3.4 A Certificate for Convergence

Although Theorem 1 asserts that the split problem and the rank-constrained problem have the same local minima, the rank-constrained problem is itself non-convex, so we have not guaranteed that every stationary point of the split problem solves the convex program. Using an approach building on work of [69], we can develop a certificate to check whether a recovered solution to (6) corresponds to a spurious stationary point when φ is convex (in particular when $\mathcal{L}$ and r are convex). This technique can also be used on the convex formulation as a method to check the distance to optimality.

We base our certificate on the following convex analysis. Let F be any proper convex function, then Fermat's rule states that the set of minimizers of F are points L such that $0 \in \partial F(L)$, where $\partial(\cdot)$ denotes the subdifferential. One could take a candidate point L, evaluate $F(L)$, and if this

is zero, conclude that L is the minimizer, but such expectations are unrealistic in floating point arithmetic, much less from an iterative method. Suppose instead we find

$$\mathcal{E} \in \partial F(L) \tag{1.17}$$

where $\mathcal{E}$ is small. Let $L^\star$ denote any minimizer of F, then from the definition of a subgradient, $F(L^\star) \geq F(L) + \langle \mathcal{E}, L^\star - L \rangle$. Then

$$F(L) - F(L^\star) \leq \langle \mathcal{E}, L - L^\star \rangle \leq \|\mathcal{E}\|_p \cdot \|L - L^\star\|_d \tag{1.18}$$

for any pair of primal and dual norms $\|\cdot\|_p$ and $\|\cdot\|_d$. We discuss bounding $\|\mathcal{E}\|_p$ and $\|L - L^\star\|_d$ in the next two subsections.

Finding an approximate zero $\mathcal{E}$ The key to the certificate is analyzing the marginalized problem (1.6) in terms of L, rather than the problem (1.3) in terms of (L, S). The point $L = UV^T$ is the optimal solution of (1.6) if and only if

$$0 \in \partial\big(\|L\|_* + \varphi(L)\big). \tag{1.19}$$

We have set $\lambda_L = 1$ for convenience, or alternatively one can absorb λ_L^{-1} into φ. Since both the nuclear norm and φ are proper lower semi-continuous and the intersection of the interior of their domains is non-empty (in particular they are finite valued), then by [9, Cor. 16.38], we have that

$$0 \in \partial\big(\|L\|_* + \varphi(L)\big) \Leftrightarrow 0 \in \big(\partial\|L\|_*\big) + \big(\partial\varphi(L)\big).$$

Both of these subdifferentials are computable. Let

$$L = \widetilde{U}\Sigma\widetilde{V}^T = \begin{bmatrix} U_1 & U_2 \end{bmatrix} \begin{bmatrix} \Sigma_1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} V_1^T \\ V_2^T \end{bmatrix} \tag{1.20}$$

be the (full) SVD of L, where $U_1 \in \mathbb{R}^{m\times r}$ and $V_1 \in \mathbb{R}^{n\times r}$. The subdifferentials of the nuclear norm at L comprises matrices of the form $X = U_1V_1^T + W$, where U_1, V_1 contain the left and right singular vectors of L that correspond to non-zero singular values,[1] and W satisfies the conditions

[1]The orthogonal matrices U_1 and V_1 should not be confused with the variables U and V that are used as a

$U_1^T W = 0$, $W V_1 = 0$ and $\|W\|_2 \leq 1$. Equivalently,

$$X \in \partial \|L\|_* \quad \text{if and only if} \quad \exists W' \in \mathbb{R}^{m-r \times n-r} \text{ s.t. } \widetilde{U}^T X \widetilde{V} = \begin{bmatrix} I & 0 \\ 0 & W' \end{bmatrix}, \quad \|W'\| \leq 1. \tag{1.21}$$

We can find an explicit matrix $D \in \partial\varphi(L)$ using (1.10), and it is unique under the smoothness conditions described in Section 1.3.2.

Altogether, we can guarantee a point $\mathcal{E} \in \partial\|L\|_* + \partial\varphi(L)$ such that

$$\begin{aligned} \|\mathcal{E}\|_F^2 &= \min_{X \in \partial \|L\|_*} \|X + D\|_F^2 \\ &= \min_{X \in \partial \|L\|_*} \|\widetilde{U}^T (X + D) \widetilde{V}\|_F^2 \\ &= \min_{\|W'\| \leq 1} \left\| \begin{bmatrix} I & 0 \\ 0 & W' \end{bmatrix} + \widetilde{U}^T D \widetilde{V} \right\|_F^2 \\ &= \|I - U_1^T D V_1\|_F^2 + \|U_1^T D V_2\|_F^2 + \|U_2^T D V_1\|_F^2 + \min_{\|W'\| \leq 1} \|U_2^T D V_2 - W\|_F^2. \end{aligned} \tag{1.22}$$

In the first equation, the minimum is achieved since the squared-norm is continuous and the set is compact, and (1.21) is used in the third equation. Each term of the fourth equation can be calculated explicitly, with the last term obtained via projection onto the unit spectral-norm ball, which requires the SVD of the small matrix $U_2^T D V_2$. This bound on $\|\mathcal{E}\|_F$ can then be used in (1.18).

Most terms above can be computed efficiently, in the sense that U_2 and V_2 never need to be explicitly computed (which is most important when r is small, since then U_2 and V_2 have $m - r$ and $n - r$ columns, respectively), and therefore the computation consists only of matrix multiplies, thin-QR factorizations, and an $r \times r$ SVD, leading to a complexity of $\mathcal{O}(rmn + r^2 \cdot (m + n) + r^3)$. For example, we compute

$$\|U_1^T D V_2\|_F^2 = \|U_1^T D\|_F^2 - \|U_1^T D V_1\|_F^2$$

factorization of L, as U and V are not necessarily orthogonal. In fact, U_1 and V_1 can be efficiently computed from the factorization $L = UV^T$ by taking the QR-decompositions $U = Q_U R_U$ and $V = Q_V R_V$ so $L = Q_U (R_U R_V^T) Q_V^T$. Perform a SVD on the small inner matrix to write $(R_U R_V^T) = U_R \Sigma V_R^T$, and hence Σ are the nonzero singular values of L and $U_1 = Q_U U_R$ and $V_1 = Q_V V_R$ are the corresponding left and right singular vectors.

and $\|U_2^T D V_1\|_F^2$ is computed analogously. The final term requires an unavoidable $(m-r)\times(n-r)$ SVD factorization, so for large m, n this is not cheap, and in this case we suggest only computing the certificate as a final check at the end rather than at every point to check convergence.

Bounding the distance to the feasible set We seek a bound on $\|L - L^\star\|_d$ in an appropriate norm, which we combine with the bound on $\|\mathcal{E}\|_F$ in (1.18) to bound the error on the objective function. Since the bound on $\mathcal{E}$ is in the Frobenius norm, ideally we set $\|\cdot\|_d = \|\cdot\|_F$, but bounds in any norm will work using $\|L\|_F \leq \sqrt{\text{rank}(L)}\|L\|$ and $\|L\|_F \leq \|L\|_*$.

Letting $F(L) = \|L\|_* + \varphi(L)$, we first bound $F(L^\star)$ by computing $F(L)$ for explicit choices of L. To be concrete, in this section we assume $r(S) = \|S\|_1$ (and again choose $\lambda_S = 1$ for simplicity) and a squared-quadratic loss function. Choosing L such that $\mathcal{A}(L) = b$ means that $S = 0$ in the definition of φ (1.5), and hence $F(L) = \|L\|_*$. Another choice is $L = 0$, and then choosing $S = 0$ (which may be sub-optimal) leads to $F(L) \leq \frac{1}{2}\|b\|^2$. A third choice is to explicitly compute $F(L)$ at the iterates in the algorithm and record the best.

Denoting $F^\star = \min_L F(L)$ and using $F^\star \leq F_{\text{bound}}$, non-negativity of φ immediately implies $\|L^\star\|_* \leq F_{\text{bound}}$. Hence $\|L - L^\star\|_F \leq \|L\|_F + \|L^\star\|_F \leq \|L\|_F + \|L^\star\|_* \leq \|L\|_F + F_{\text{bound}}$, and $\|L\|_F$ is explicitly computable.

Results Figure 1.1 shows how the distance to the optimal subgradient decreases over time for the convex solver LagQN, the non-convex Split-SPCP with a rank bound that is too strict, and Split-SPCP with a rank bound large enough to reach the global optimum. When the rank bound is too restrictive, Split-SPCP cannot recover the optimal L. Measuring the distance to the optimal L reveals this. Figure 1.1 shows that the distance plateaus far from zero when the rank of L is bounded above by 30, and the rank of the optimizing L is 58. In practice, this measure can be used to indicate that the rank bound should be increased.

1.4 Initialization

The non-convex formulation (1.8) is sensitive to the point of initialization, and requires a bound on the rank of L. The performance of this solver can vary greatly for different initializations

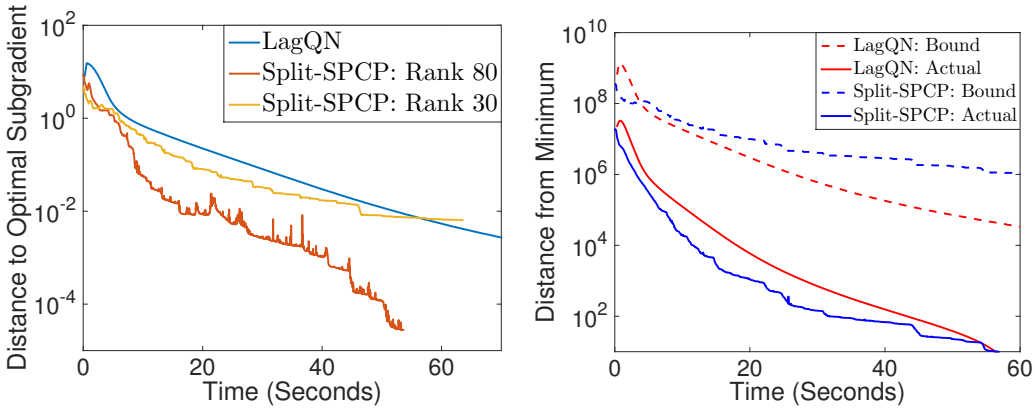


Figure 1.1: (Left): Distance from the optimal subgradient for the convex solver LagQN and the non-convex Split-SPCP with different rank bounds. The rank of the optimal L is 58. (Right): Bounded distance from minimal objective value, as given by $\|\mathcal{E}\|_F(\|L\|_F + F_{\text{bound}})$, with F_{bound} given by the objective value at the current iterate. The actual distance from the minimal objective value is shown for comparison. The rank bound for the Split-SPCP test is 80.

of these parameters. In this section, we present some heuristics for choosing a suitable initialization.

1.4.1 Dynamically Increasing k

Figure 1.2 demonstrates the sensitivity of the Split-SPCP program to the factor rank k. These tests were performed on the surveillance video data described in section 1.5, where the true rank of the low rank component is 58. We see that when $k \leq \text{rank}(L)$, Split-SPCP does not converge to the correct solution, but if k is much larger than the rank of the minimizing L, then the computation is slowed significantly.

Because our solver is oblivious to the individual dimensions of U and V, columns can be added to both matrices on the fly. Dynamically updating the rank bound can help when (1.8) is initialized with a k that is too small. Suppose a solver reaches some convergence criterion at iteration i. To see if this is a solution to (1.6), we add a single column to U and V:

$$\begin{bmatrix} U_i & u \end{bmatrix} \begin{bmatrix} V_i & v \end{bmatrix}^T = L_i + uv^T, \tag{1.23}$$

and observe whether this rank-one allows for a lower objective value or certificate value. Dynamically increasing k, even aggressively, is still more efficient than overestimating k from the start.

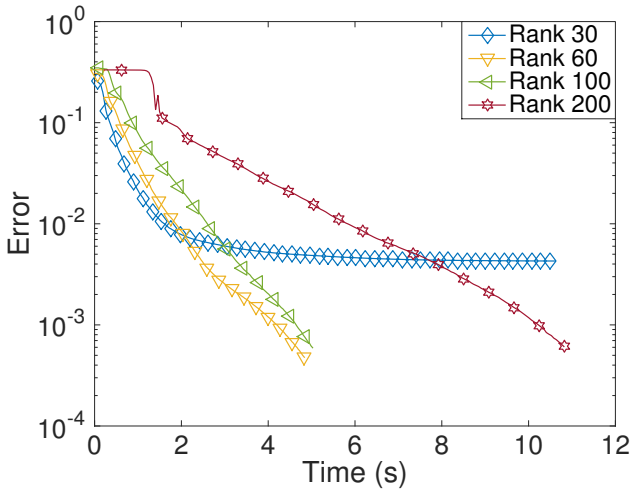


Figure 1.2: Varying the rank bound k in Split-SPCP affects computation time and the accuracy of the final solution. One marker represents 10 iterations.

1.4.2 Choosing U_0 and V_0

Choosing an initialization point (U_0, V_0) has received considerable attention in factorized low rank recovery problems. Most of this work has centered around matrix completion using alternating minimization, see, for example, [38, 46, 47, 48, 43, 69]. Provably good choices for U_0 and V_0 are $U_0 = U\Sigma^{\frac{1}{2}}$ and $V_0 = V\Sigma^{\frac{1}{2}}$, where U and V come from the SVD of $\mathcal{A}^\dagger X$, where X is the data-matrix, $\mathcal{A}$ is a linear operator, and $\dagger$ denotes the Moore-Penrose pseudo-inverse. [38] showed that these initial points lie in the basin of attraction of globally optimal minima, yielding fast convergence to an optimal solution.

As mentioned in [69], it is sometimes not necessary to perform a full SVD to form U_0, V_0. Once k is chosen, only a partial SVD is necessary to calculate the first k singular values and vectors, which can be done efficiently using the randomized SVD (rSVD) [36]. Although using the rSVD is significantly faster than a full SVD, the values and vectors it returns are not always accurate, especially when the singular values of the underlying data do not decay rapidly.

Figure 1.3 shows the performance of Split-SPCP using various initial points. The tests were performed on a 5,000 $\times$ 5,000 matrix $X = L + S$, where $\text{rank}(L) = 1000$ and $\text{sparsity}(S) = 9.43\%$. The rank bound was $k = 1{,}050$. All 5,000 singular value and vector triples of X were calculated

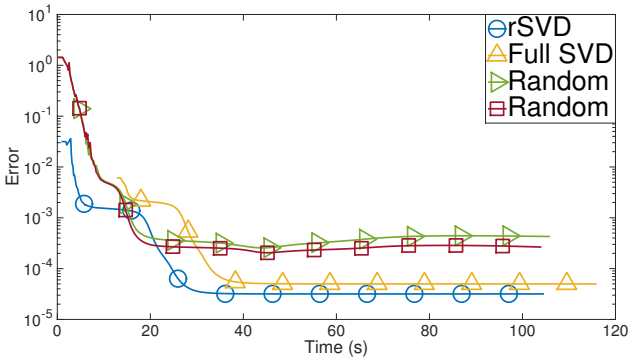


Figure 1.3: The performance of Split-SPCP with various initialization points. All tests were run on a 5,000 × 5,000 matrix $X = L + S$, where $\text{rank}(L) = 1000$ and $\text{sparsity}(S) = 9.43\%$. The rank bound was $k = 1{,}050$. One marker is 50 iterations.

for the SVD test, and U_0 and V_0 were formed from the first $k = 1{,}050$ triples. For the rSVD test, only the first 1,050 triples were approximated, U_0 and V_0 were formed from these.

Figure 1.3 shows that initializing U_0 and V_0 with the first 1,050 singular value and vector triples returned by the full SVD yields the smallest initial error, but the extra computational cost is not worth the better start. The rSVD allows quicker convergence to a similar solution. The two random initializations converge to a stationary point that is not globally optimal.

1.5 Numerical Experiments

In this section, we present numerical experiments that illustrate our SPCP accelerations. Recall that the Split-SPCP program is given by

$$\min_{U,V} \quad \frac{\lambda_L}{2}(\|U\|_F^2 + \|V\|_F^2) + \varphi(UV^T), \tag{1.24}$$

where

$$\varphi(UV^T) = \min_S \frac{1}{2}\|UV^T + S - X\|_F^2 + \lambda_S\|S\|_1. \tag{1.25}$$

Since the objective of Split-SPCP is smooth, any first-order method can be used to solve it. Also, since the communication-heavy SVD step is no longer a limitation, we are motivated to choose the solver that is most suited for the GPU. To find this solver, we compared the performance of several first-order methods as they solved a logistic regression problem on the GPU. The time it took for

different solvers to minimize a logistic loss with a Tikhonov regularization term for 10^4 labels is shown in Table 1.1 below. The computation time is defined as the time it took the solver to come within 10^{-8} of the optimal objective value.

Solver	Normalized Computation Time
Cyclic Steepest Descent	3.46
Barzilai-Borwein	1.24
Conjugate Gradient	1.44
Scaled Conjugate Gradient	1.53
Preconditioned Conjugate Gradient	3.20
L-BFGS (10 iterations in memory)	**1**
L-BFGS (50 iterations in memory)	1.82
BFGS	2.39
Hessian-Free Newton's Method	3.05

Table 1.1: First-order methods solving logistic loss program with 10^4 labels on the GPU. All times are reported as a ratio with respect to the fastest solver (L-BFGS with 10 iterations in memory). Time is recorded when the solution is within 10^{-8} of the optimal value.

From our analysis, we decided that the L-BFGS solver, storing a small number of iterations in memory, yields the best performance. The times shown in 1.1 depend on the stopping tolerance as well as the specific problem, but there is further evidence in the literature supporting L-BFGS as a competitively efficient solver [72]. To solve Split-SPCP, we use L-BFGS as implemented in [61], with our own slight modifications so that it will run on the GPU.

The value of the objective is calculated by first computing the minimizing S in the definition of φ, which we call $S_{U,V}$, using soft thresholding. Explicitly, the soft-thresholding (or shrinkage) operator is defined so that

$$(\text{shrink}(Y, \lambda))_{i,j} = \text{sign}(Y_{i,j})(|Y_{i,j}| - \lambda|)_+,$$

where $(\cdot)_+$ is the zero operator on negative reals and is the identity operator on non-negative reals.

This shrinkage operator gives us a closed-form representation of $S_{U,V}$ and $\varphi(UV^T)$:

$$S_{U,V} = \text{shrink}(X - UV^T, \lambda_S),$$
$$\varphi(UV^T) = \frac{1}{2}\|L + S_{U,V} - X\|_F^2 + \lambda_S\|S_{U,V}\|_1$$

Once $S_{U,V}$ and $\varphi(UV^T)$ are found, we can compute the objective value, $\frac{\lambda_L}{2}(\|U\|_F^2+\|V\|_F^2)+\varphi(UV^T)$. We can also compute the gradients of the objective, which are given below. We let $f(U,V)$ denote the objective for convenience.

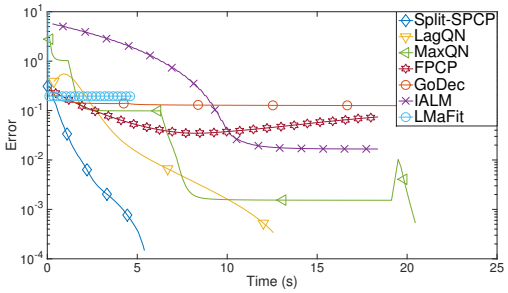
$$\nabla_U f = \lambda_L U + (UV^T + S_{U,V} - X)V,$$
$$\nabla_V f = \lambda_L V + (UV^T + S_{U,V} - X)^T U.$$

In [14] and [63], the authors provide a review of the fastest algorithms in the family of robust PCA and principal component pursuit for background subtraction. We chose some of the fastest of these to compare to our Split-SPCP algorithm. Each of these methods runs faster on the GPU, so we ran all of them on the GPU for our comparisons. We found that our non-convex solver is particularly well-suited for the GPU, and **outperforms all other solvers in almost every case.** This is shown on data sets from applications in section 1.5.1 and synthetic data in section 1.5.2. In section 1.5.3, we show that Split-SPCP on the GPU can be used for real-time background subtraction, even for high-resolution videos. Finally, in section 1.5.4, we demonstrate how marginalizing the sparse component in low rank models leads to an accelerated Frank-Wolfe method for optimization. We also discuss how applying marginalization to the Frank-Wolfe problem generalizes recent results published in [54].

1.5.1 SPCP for Background Subtraction

For our background subtraction tests, we use the escalator surveillance video provided by [50]. We want to identify the people in the video while ignoring the moving escalators and the stationary background. This problem is particularly difficult for many low rank recovery algorithms, such as PCA, because the motion of the escalator is a confounder. SPCP is less sensitive to outliers, so

it can overcome this challenge. Our SPCP method and several others are compared in Figure 1.4. We measured performance in terms of the objective value over time. All of the algorithms were run on the GPU. To find a reference solution, we hand-tuned the parameters λ_L and λ_S



(a) Comparing SPCP solvers for background subtraction. One marker corresponds to 40 iterations.

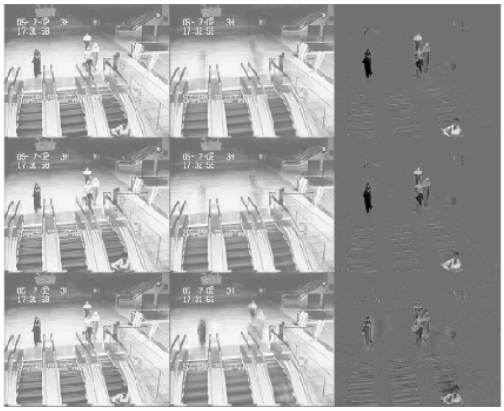
Algorithm	Reference
Split-SPCP	**This Work**
Lagrangian Quasi-Newton (LagQN)	Aravkin et al. (2014a)
Max Quasi-Newton (MaxQN)	Aravkin et al. (2014a)
Fast PCP (FPCP)	Rodrígues & Wohlberg (2013)
Go Decomposition (GoDec)	Zhou & Tao (2011)
Inexact ALM (IALM)	Lin et al. (2011)
LMaFit (LaMaFit)	Wen et al. (2010)

(b) References for the algorithms used in testing. Among these are the fastest models for background subtraction, as determined in [14] and [63].

Figure 1.4: A Comparison of SPCP Solvers

in the (convex) quasi-Newton Lagrangian SPCP algorithm (LagQN) until it found a qualitatively accurate decomposition. The L we chose has rank 58, and the optimal S is 58.03% sparse. The optimal parameters for both Split-SPCP and LagQN are $\lambda_L = 115$ and $\lambda_S = 0.825$. We tuned the parameters in the other solvers to recover this solution as closely as possible. The error is measured as the normalized difference between the objective and the objective at the optimal solution. Since the solvers minimize different objectives, we calculated the Split-SPCP objective value at each iteration for every algorithm. We did not include these calculations in our time measurements. To initialize Split-SPCP, we used the first 100 singular values and vectors from the randomized SVD of the data matrix.

Several of the algorithms in Figure 1.4 did not converge to the same solution, despite considerable effort in parameter tuning. These algorithms were designed to quickly find approximate solutions to the SPCP problem, and might have suffered from the large amount of noise present in the data. The approximate solutions recovered by these algorithms are qualitatively different from the solution found by Split-SPCP, see Figure 1.5. We also found that a lower objective value generally corresponds to a qualitatively superior solution.



(a) X,L, and S matrices found by (from top to bottom) Split-SPCP, LagQN, and GoDec.

(b) X,L, and S matrices found by (from top to bottom) FPCP, IALM, and LMaFit.

Figure 1.5: Background subtraction using various SPCP solvers on surveillance video data from [50] (frame 10 is shown). We see that Split-SPCP, LagQN, and IALM best locate the people while ignoring the escalators.

We measured the quality of the solutions in terms of the *Corrected Akaike Information Criterion*, or AIC_C. The AIC_C measures the fit of a statistical model to a certain data set. Given a model with p parameters and a log-likelihood function ℓ, the value of the AIC_C is

$$\text{AIC}_\text{C} = 2(p - \log(\ell_{\max})) + \frac{2p(p+1)}{m \cdot n - p - 1},$$

where $m \cdot n$ is the size of the data set and $\ell_{\max}$ is the maximum of ℓ. The preferred statistical model is the one that minimizes the AIC_C. The AIC_C does not only favor models that maximize the likelihood function, but it also penalizes complex models with many parameters, guarding against overfitting. Using AIC_C as a measure of quality avoids both overemphasis of objective values and the inherent ambiguity of visual comparisons.

To compute the AIC_C value, we must formulate SPCP as a statistical model. It is well-known that the least-squares loss term assumes that the data is of the form $X = L + S + Z$, where the entries of Z are iid Gaussian random variables with $\mu = 0$ and variance estimated using the sample variance, $\hat{\sigma}^2 = \frac{\|X\|_F^2}{m \cdot n}$. Similarly, the ℓ_1-regularizer assumes that the entries of S are drawn iid from the Laplace distribution with mean 0 and variance estimated as $\hat{b} = \frac{\|S\|_1}{m \cdot n}$. The nuclear norm of L is the ℓ_1-norm of its singular values, so its corresponding prior assumes that the singular values of

L follow the Laplace distribution. The log-likelihood function is then

$$\ell(L,S,X) = -\left(\frac{m\cdot n}{2}\right)\log(2\pi\sigma^2) - \frac{\|L+S-X\|_F^2}{2\sigma^2} - (m\cdot n)\log(2b) - \frac{\|S\|_1}{2b} - \operatorname{rank}(L)\log(2b_*) - \frac{\|L\|_*}{2b_*}, \quad (1.26)$$

where σ^2, b and b_* are computed according to their respective estimator. We must also define the number of parameters of this model, which is equal to its degrees of freedom. Each term in SPCP provides the following degrees of freedom:

$$\begin{aligned} \operatorname{rank}(L) = k \quad &\rightarrow \quad k(m+n-k) \text{ degrees of freedom,} \\ \|S\|_1 \quad &\rightarrow \quad nnz(S) \text{ degrees of freedom,} \\ \tfrac{1}{2}\|L+S-X\|_F^2 \quad &\rightarrow \quad \left(\frac{\|L+S-X\|_F^2}{\|X\|_F^2}\right)(m\cdot n) \text{ degrees of freedom.} \end{aligned}$$

Note that $nnz(S)$ counts the number of non-zero entries in S. Finding the degrees of freedom in a rank-k matrix or a sparse matrix are both standard calculations. For the loss term, we use the residual effective degrees of freedom as an estimate of the flexibility that it introduces. This is detailed further in [52]. The number of parameters p is equal to the total degrees of freedom.

The AIC_C values for the solutions shown in Figure 1.5 are listed in Table 1.2. The AIC_C value for the reference solution is listed under "oracle." Recall that the reference solution was found using the LagQN solver with a tight tolerance. All the other values in the table correspond to solutions that met comparable tolerances. Also, since the IALM model does not use an ℓ_1-norm regularizer, many of the values in the returned S matrix were small but not exactly zero. To make accurate comparisons, we applied the shrinkage operator to the S matrix returned by IALM to set small values equal to zero before calculating the degrees of freedom. The values for IALM and LMaFit are high because these solvers are not generally robust to large amounts of noise. With this metric as well, we see that Split-SPCP discovers the best solution, and in a much shorter time compared to the other algorithms.

Split-SPCP has an advantage because it parallelizes well on GPU architectures. This is due to the relative simplicity of the L-BFGS algorithm, see also [29]. The algorithm avoids complex linear

Solver	AIC_C ($\times 10^6$)
Oracle	*7.37*
Split-SPCP	**7.64**
GoDec	10.95
LagQN	11.30
MaxQN	11.64
FPCP	15.70
IALM	2.43×10^7
LMaFit	3.99×10^9

Table 1.2: Degrees of freedom in solutions returned by various solvers.

algebraic decompositions that require heavy communication, such as the QR and SVD. Figure 1.6 shows that although Split-SPCP on the CPU is slower than LagQN on the CPU, Split-SPCP enjoys enormous speedup when it is implemented on the GPU.

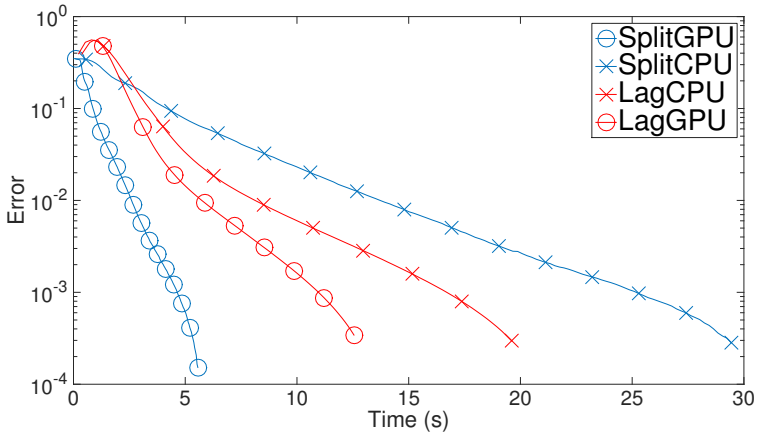


Figure 1.6: Split-SPCP and LagQN on the CPU and the GPU. Tests were performed on the synthetic data described in section 1.5.2

1.5.2 Synthetic Data

To test Split-SPCP further, we created a synthetic problem consisting of a noisy data-matrix that can be represented as the sum of a low rank and a sparse component. We created two $1{,}000 \times 150$ random matrices with each entry drawn from independently from the univariate normal distribution, and defined our low rank reference matrix, L_{ref}, as the product of these two matrices. To form the sparse reference matrix, S_{ref}, we filled 50% of a $1{,}000 \times 1{,}000$ matrix with numbers

drawn independently from the univariate normal distribution, and the other entries we set equal to zero. We defined X to be the sum $L_{ref} + S_{ref}$ with added white Gaussian noise, so that $\frac{\|L_{ref}+S_{ref}-X\|_F}{\|X\|_F} = 8.12\times10^{-5}$. Compared to data sets that are normally encountered in applications of SPCP, the sparse component of S has a large number of non-zero entries, the rank of L is large, and the noise in X is small.

We initialized Split-SPCP with the first 200 singular value and vector triples of X, so that k was larger than the rank of L. These triples were found using the rSVD. The tuning parameters were set to $\lambda_L = 2.95$ and $\lambda_S = 0.1$. As before, we measured performance based on the normalized difference from the true Split-SPCP objective value. The results are shown in Figure 1.7.

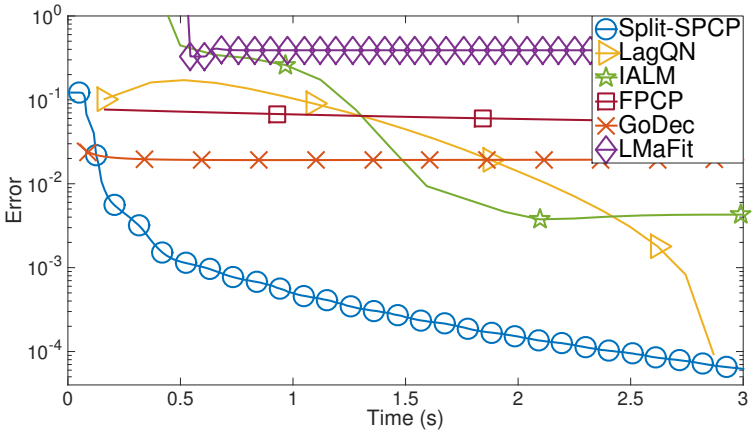


Figure 1.7: Performance of SPCP solvers on synthetic data. One marker represents six iterations.

We see that FPCP and GoDec discover approximate solutions quickly, but the approximations are not within 10^{-2} of the true objective. LMaFit exhibits poor performance. This could be due to the fact that S is about 50% sparse, and LMaFit struggles with problems that have relatively many non-zero entries [71]. Split-SPCP significantly outperforms all solvers in speed and accuracy, except for LagQN. While Split-SPCP is much faster than LagQN initially, LagQN catches up in later iterations.

1.5.3 Dimensional Scaling and Real-Time Video Processing

One useful application of accelerated low rank recovery algorithms is real-time background subtraction in video. The experiments in section 1.5.1 show that Split-SPCP is faster and more accurate than the methods reviewed in [14] and [63] for background subtraction, and in this section, we apply Split-SPCP to the real-time processing of high-resolution videos with standard frame rates. We also explore how Split-SPCP scales as the resolution and number of frames increases.

To measure the performance of Split-SPCP on more diverse video datasets, we used videos from the Background Model Challenge [70]. The experiments shown in Figure 1.8 used a video with 240×320 pixels per frame and 32,964 frames, which is substantially larger than the low-resolution escalator video with 130×160 pixels per frame and only 200 frames. We unfolded the three-dimensional video data into a matrix with each column containing a frame, and we partitioned the matrix columnwise into blocks comprising a fixed number of frames. With the rank bound fixed at 60, we then recorded the computation time required to perform 200 iterations of Split-SPCP on one block as a function of the block size. Larger block sizes yield better solutions, but they also require more time for SPCP to converge. We found that after 200 iterations, Split-SPCP had converged to within 10^{-3} of the optimal objective value for all block sizes, and the returned solution was qualitatively no better than the optimal solution.

Figure 1.8 shows how Split-SPCP scales as the video resolution and the number of frames per block increase. These two qualities correspond to the number of rows and the number of columns of each block. We increased the resolution by rescaling each frame so that the aspect ratio stayed close to constant, and we used interpolation to impute the missing pixel values. It is easy to discern the $\mathcal{O}(n)$ scaling as the number of frames (columns) is increased. Increasing the resolution affected the computation time linearly, but beyond a "critical resolution" of 332×446 pixels per frame, the slope of the linear scaling underwent a shift in regime. This linear scaling is expected because the bottleneck of Split-SPCP is the matrix-matrix multiply UV^T, which requires $\mathcal{O}(mnk)$ operations, so with k fixed, the computation time of Split-SPCP grows linearly with the number of data points.

The results of the background subtraction are shown in Figure 1.9.

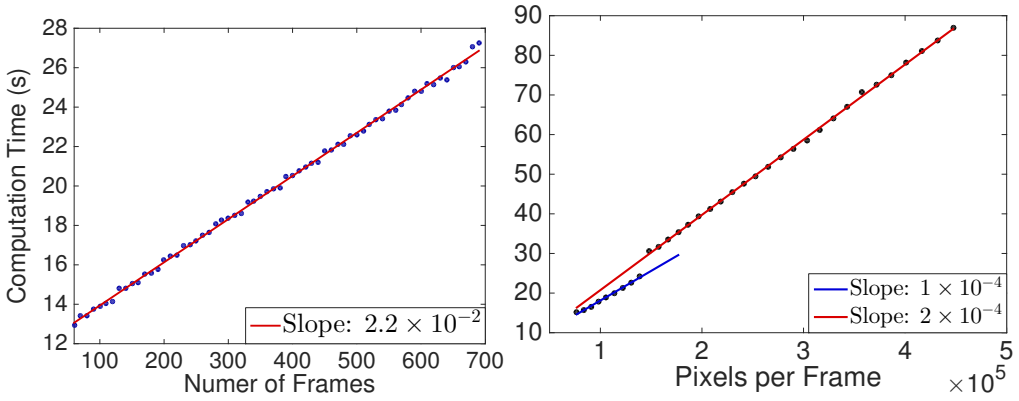


Figure 1.8: The scaling of Split-SPCP (left) as the number of columns (or frames) increases and (right) as the number of rows (or pixels per frame) increases. Each point is the average of twenty trials. For the column-scaling tests, the resolution was fixed at 240×320 pixels per frame, and for the row-scaling tests, the number of frames in a block was fixed at 100. The tuning parameters were set to $\lambda_L = 200, \lambda_S = 5$.

These experiments suggest conditions under which Split-SPCP can decompose a video in real-time. Assuming a resolution of 240×320 pixels per frame and a frame rate of 24 frames-per-second, which is standard in many applications of video processing, the video must be partitioned blocks of about 600 frames or more. With this partitioning, the algorithm will have finished decomposing the given block before the next block is recorded. This number increases linearly with resolution, following the trend line given in the rightmost plot of Figure 1.8. Following these guidelines, Split-SPCP can decompose the videos from [70] in real-time without sacrificing quality. An example is shown in Figure 1.9.

1.5.4 Accelerating Frank-Wolfe

Our approach to accelerating low rank recovery models is closely connected to an approach presented in [54] to accelerate the Frank-Wolfe method for RPCA. The Frank-Wolfe method, first proposed in [30], has recently gained attention for its ability to yield scalable algorithms for optimization problems involving structure-encouraging regularizers [41, 54]. Frank-Wolfe is a method of optimizing a Lipschitz-differentiable function over a convex set by solving a sequence of linear

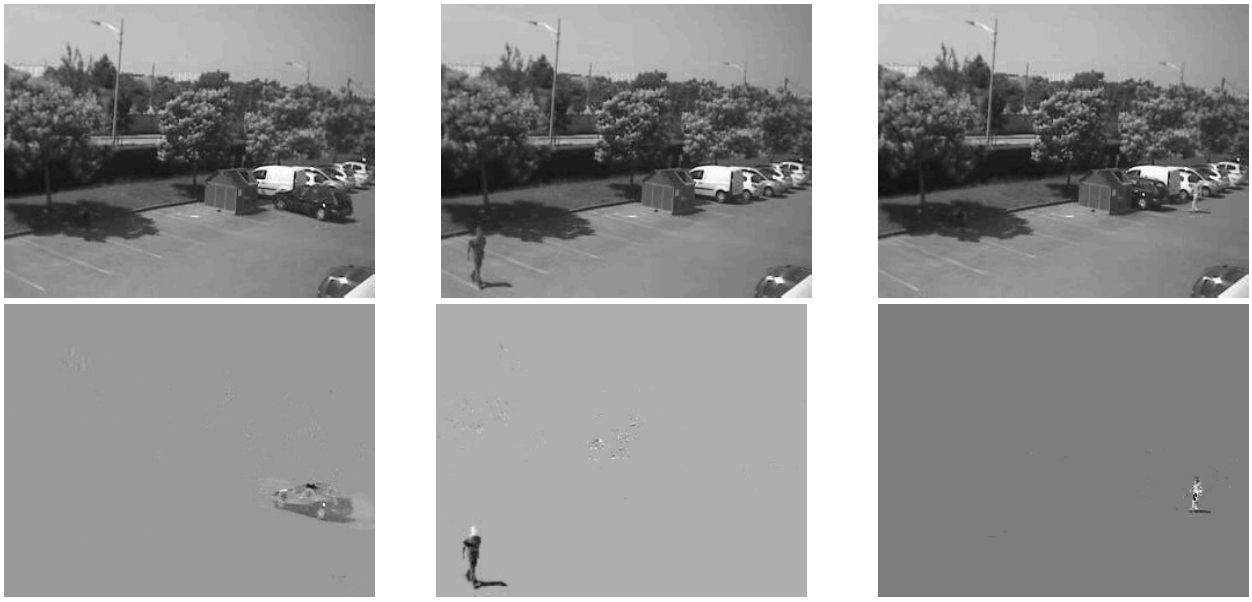


Figure 1.9: Real-time background subtraction using Split-SPCP. Split-SPCP correctly identifies (left) a car in the middle-ground, (center) a person in the foreground, and (right) a person in the background. The resolution is 240×320 pixels per frame, 600 frames were decomposed at one time, and only 200 iterations were allowed.

subproblems. For example, if f is Lipschitz-differentiable and $\mathcal{C}$ is a convex set, Frank-Wolfe solves the problem

$$\min_{x \in \mathcal{C}} \quad f(x) \tag{1.27}$$

by linearizing f about each iteration x_k: $f(y) \approx f(x_k) + \langle \nabla f(x), y - x \rangle$, minimizing the linear problem to find a decent direction $y_k - x_k$, and then stepping in this direction with a step size η: $x_{k+1} = x_k + \eta(y_k - x_k)$. Traditionally, $\eta = \frac{2}{k+2}$ is fixed for convergence guarantees, but better performance can be achieved with adjustable step sizes [54]. A summary of this process is shown in Algorithm 1.

Algorithm 1 Using the Frank-Wolfe Method to Solve (1.27) (c.f. [30, 41, 54])

Input: $x_0 \in \mathcal{C}$

while (Not Converged) **do**
  $y_k \leftarrow \arg\min_{y \in \mathcal{C}} \langle y, \nabla f(x_k) \rangle$
  $\eta = \frac{2}{k+2}$
  $x_{k+1} \leftarrow x_k + \eta(y_k - x_k)$
end while

The Frank-Wolfe method has recently become popular in machine learning, statistics, image processing, and related fields due to the simple form of the linear subproblem when handling structure-inducing regularizers [41, 42, 37]. For example, for any optimization problem involving the nuclear norm, we can write an equivalent problem where the nuclear norm term is represented in the constraint [42, 54]:

$$\min_{X} \quad f(X) + \lambda \|X\|_* \Leftrightarrow \min_{\|X\|_* \leq t} \quad f(X) + \lambda_L t. \tag{1.28}$$

When $\mathcal{C}$ in Algorithm 1 is the scaled nuclear norm unit ball, then the linear subproblem corresponds to finding the leading eigenspace of the iterate L_k. This is much cheaper than computing the full SVD at each iteration, which is required by proximal methods.

However, it is often difficult to scale the Frank-Wolfe method for problems involving ℓ_1 regularization. For these problems, each Frank-Wolfe iteration updates iterate S_k with only a single non-zero element [54]. This makes Frank-Wolfe unfeasible when the sparse component of the data-matrix has several thousand non-zero elements or more. The authors of [54] adapt the Frank-Wolfe method to solve SPCP, using the traditional Frank-Wolfe update for the low rank component and using a projected gradient update for the sparse component. Their adaptations allow for significant speedup and better convergence compared to the traditional Frank-Wolfe.

The techniques used in [54] to accelerate the update of the sparse component can be seen as a special case of the marginalization we present in this work. With the sparsely structured component marginalized, the program depends only upon the low rank component, so the benefits of the Frank-Wolfe scheme are preserved and its drawbacks are negated. Algorithm 2 shows how marginalization can be used to extend the adaptations presented in Algorithms 5 and 6 of [54]. Algorithm 2 solves the following program, which is equivalent to SPCP:

$$\min_{\|L\|_* \leq t} \quad \lambda_L t + \min_{S} \frac{1}{2} \|\mathcal{P}_\Omega(L + S - X)\|_F^2 + \lambda_S \|S\|_1. \tag{1.29}$$

This problem is smooth, and as mentioned earlier, the constraint set is amenable to acceleration by applying the linear subproblems introduced in the Frank-Wolfe scheme. Each iteration of Algorithm

2, then, updates the low rank component and the nuclear norm bound, t, while the sparse component remains only implicitly defined. More details on this approach can be found in [54].

Algorithm 2 Frank-Wolfe Method with Marginalization

Input: $\mathbf{L_0}, \lambda_L, \lambda_S, t_0 = 0, U_0 = \frac{1}{2\lambda_L}\|\mathcal{P}_\Omega(\mathbf{L_0} + \mathbf{S_0} - \mathbf{X})\|_F^2$

while Not Converged **do**
  $\mathbf{S_k} \leftarrow \text{shrink}(\mathbf{X} - \mathbf{L_k}, \lambda_S)$
  $\mathbf{Y_k} \leftarrow \underset{\|\mathbf{Y}\|_* \leq 1}{\arg\min} \langle \mathcal{P}_\Omega(\mathbf{L_k} + \mathbf{S_k} - \mathbf{X}), \mathbf{Y} \rangle$
  if $\lambda_L \geq -\langle \mathcal{P}_\Omega(\mathbf{L_k} + \mathbf{S_k} - \mathbf{X}), \mathbf{Y_k} \rangle$ **then**
    $\mathbf{V_k} \leftarrow \mathbf{0}$
    $V_{t_k} \leftarrow 0$
  else
    $\mathbf{V_k} \leftarrow U_k \mathbf{Y_k}$
    $V_{t_k} \leftarrow U_k$
  end if
  $\eta^\star \leftarrow \underset{\eta \in (0,1)}{\arg\min} \quad \frac{1}{2}\|\mathcal{P}_\Omega((1-\eta)\mathbf{L_k} + \eta \mathbf{V_k} + \mathbf{S_k} - \mathbf{X})\|_F^2 + (1-\eta)\lambda_L t_k + \eta \lambda_L t_k$
  $\mathbf{L_{k+1}} \leftarrow (1-\eta^\star)\mathbf{L_k} + \eta^\star \mathbf{V_k}$
  $t_{k+1} \leftarrow (1-\eta^\star) t_k + \eta^\star V_{t_k}$
  $U_{k+1} \leftarrow \frac{1}{2\lambda_L}\|\mathcal{P}_\Omega(\mathbf{L_0} + \mathbf{S_0} - \mathbf{X})\|_F^2 + t_k$
end while

While Algorithm 2 is similar to the methodology in [54], fully marginalizing the sparse component eliminates the linear subproblem required in [54] to update the iterate S_k. We show the performance of this adapted Frank-Wolfe scheme on SPCP in Figure 1.10. Although its performance is better than the traditional Frank-Wolfe (see [54]), it is still much slower than quasi-Newton methods applied to the smoothed problem.

1.6 Low Rank Recovery Models with a Parallel rSVD

In section 1.5, we saw that solving low rank recovery models without the SVD step required by proximal methods allows the GPU to offer greater acceleration. In this section, we explore whether a communication-avoiding rSVD can provide the convex problem (1.3) with similar acceleration on the GPU.

Many problems in data analysis involve matrices of a "tall-skinny" structure. Tall-skinny matrices arising in applications could have twice as many rows as columns, or several-thousand-

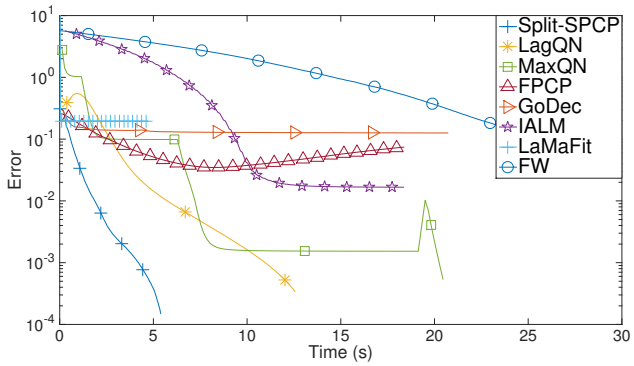


Figure 1.10: Comparing Frank-Wolfe with marginalization (denoted "FW") with other SPCP solvers for background subtraction on the escalator video of [50]. One marker corresponds to 40 iterations.

times more rows than columns [26]. In the psychometric data that we consider in section 1.6.4, the matrices have about 230-times more rows than columns. The surveillance video from the previous section has dimensions $20{,}800 \times 200$, so it is also tall-skinny. The prevalence of this structure is due to the fact that high-dimensional data sets must be "unwrapped" into a two-dimensional matrix before matrix-based algorithms can analyze the data. Data from video are three-dimensional: there are two dimensions of space, and one dimension of time. When applying a low rank recovery model to this data, each frame of the video is stored as one column of the matrix, so a two-hundred frame video with a resolution of 120×160 pixels per frame produces a $20{,}800 \times 200$ matrix for decomposition. The problem is exacerbated when data is even higher dimensional, as it is for fMRI data.

Unfortunately, tall-skinny matrices pose a challenge for matrix decompositions, especially on computational architectures where communication is costly (such as a GPU) [26]. This is why transferring low rank recovery models to the GPU is a challenge: to minimize the nuclear norm, an SVD is computed at every step, and this is extremely costly on the GPU.

Even for problems involving matrices that are not originally tall-skinny, many popular algorithms for data-analysis produce tall-skinny matrices during intermediate calculations, which significantly slows computation. The randomized SVD [36] is one of these algorithms. The rSVD creates a tall-skinny matrix structure that amplifies communication costs, even if the input matrix

is square. The effect of this increased communication is notable: on the CPU, the rSVD step in LagQN accounts for about 55% of the algorithm's total running time, but when this solver is run on the GPU, the rSVD accounts for about 90% of SPCP's total running time. The following sections demonstrate how the Tall-Skinny QR algorithm of [26] can provide a convex option for accelerated low rank recovery models. While TSQR has been used to accelerate the SVD step in RPCA for tall-skinny matrices [3], we extend this idea to the rSVD algorithm on general matrices.

1.6.1 Motivation

A basic description of the rSVD algorithm is presented in Algorithm 3. To decompose the $m \times n$ matrix X, the rSVD algorithm first multiplies X on the right by the $n \times k$ random matrix Ω to project X onto a low-dimensional subspace. This projection creates an $m \times k$ matrix Y, where $k \ll m$, The orthogonalization steps (3 through 8) perform QR decompositions on these tall-skinny matrices. If the singular values of X decay slowly (which is often the case in applications), then several power iterations are performed, which increases the number of QR decompositions. Regardless of input X, Y is always tall-skinny when the target rank k is small.

Algorithm 3 Randomized Singular Value Decomposition (rSVD) [36]

Input: Matrix X, rank estimate k, and power-parameter q

1: Form Ω, an $n \times k$ Gaussian matrix
2: $Y_0 \leftarrow X\Omega$, so that Y is an $m \times k$ tall-skinny matrix
3: $Q_0 \leftarrow \texttt{qr}(Y_0)$
4: **for** $j \in \{1, 2, \cdots q\}$ **do**
5: $\quad \tilde{Y}_j \leftarrow X^T Q_{j-1}$
6: $\quad \tilde{Q}_j \leftarrow \texttt{qr}(\tilde{Y}_j)$
7: $\quad Y_j \leftarrow X\tilde{Q}_j$
8: $\quad Q_j \leftarrow \texttt{qr}(Y_j)$
9: **end for**
10: $Q \leftarrow Q_q$
11: $B \leftarrow Q^T X$, (where B is a small $k \times n$ matrix)
12: $[\hat{U}, \Sigma, V] \leftarrow \texttt{svd}(B)$
13: $U \leftarrow Q\hat{U}$

The tall-skinny structure notably slows the QR decomposition on the CPU, and when performing this algorithm on the GPU, the sub-optimality of the tall-skinny structure is amplified [26].

To avoid this problem, we replace the QR decompositions in the rSVD algorithm with a tall-skinny QR decomposition developed in [26].

1.6.2 The TSQR Algorithm

It is important to see why tall-skinny matrix structures are so hazardous in the first place. This analysis can be found in [26], but we include it here for completeness. Consider the Householder QR decomposition on the tall-skinny matrix in Figure 1.11:

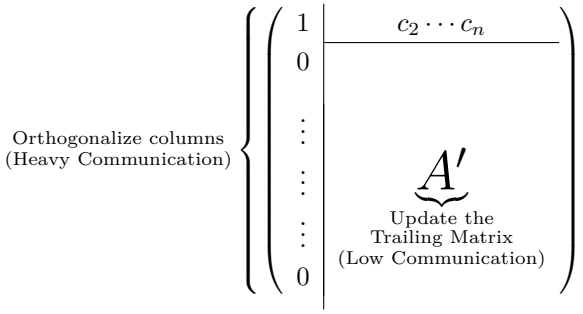


Figure 1.11: The first iteration in the Householder QR Decomposition of a Tall-Skinny Matrix.

Orthogonalization of columns is a particularly communication-heavy task, especially when compared to the matrix-update step that can be performed using BLAS3 matrix-matrix multiplies that are easily parallelizable. As the columns of the matrix get taller, the Householder QR must spend more time orthogonalizing columns and less time performing the matrix-update. This causes deceleration in CPUs, and in GPUs, where communication between cores is even more time consuming, this causes significant inefficiency. The TSQR decomposition reduces this inefficiency. An outline of the algorithm is presented on the following page.

The TSQR Algorithm [26]:

(1) Divide the matrix vertically into 2^n blocks, which we illustrate with $n = 2$:

$$A = \begin{bmatrix} A_{1,1} \\ A_{1,2} \\ A_{1,3} \\ A_{1,4} \end{bmatrix}$$

(2) Perform a QR decomposition on each block in parallel:

$$\begin{bmatrix} A_{1,1} \\ A_{1,2} \\ A_{1,3} \\ A_{1,4} \end{bmatrix} = \left[\begin{array}{c|c|c|c} Q_{1,1} & & & \\ \hline & Q_{1,2} & & \\ \hline & & Q_{1,3} & \\ \hline & & & Q_{1,4} \end{array}\right] \left[\begin{array}{c} R_{1,1} \\ \hline R_{1,2} \\ \hline R_{1,3} \\ \hline R_{1,4} \end{array}\right]$$

(3) Concatenate the matrix of R's vertically, forming the matrices $A_{2,i}$, for $i = 1, 2, 3, \cdots, 2^{n-1}$:

$$\begin{bmatrix} A_{2,1} \\ A_{2,2} \end{bmatrix} \leftarrow \left[\begin{array}{c} \begin{pmatrix} R_{1,1} \\ R_{1,2} \end{pmatrix} \\ \hline \begin{pmatrix} R_{1,3} \\ R_{1,4} \end{pmatrix} \end{array}\right]$$

(4) Repeat steps 2 through 4 for the matrices $A_{2,i}$. Do this process $n - 1$ times. For $n = 2$, we form the matrices $Q_{2,1}, Q_{2,2}, R_{2,1}$, and $R_{2,2}$:

$$\begin{bmatrix} A_{1,1} \\ A_{1,2} \\ A_{1,3} \\ A_{1,4} \end{bmatrix} = \left[\begin{array}{c|c|c|c} Q_{1,1} & & & \\ \hline & Q_{1,2} & & \\ \hline & & Q_{1,3} & \\ \hline & & & Q_{1,4} \end{array}\right] \left[\begin{array}{c|c} Q_{2,1} & \\ \hline & Q_{2,2} \end{array}\right] \left[\begin{array}{c} R_{2,1} \\ \hline R_{2,2} \end{array}\right]$$

(5) Concatenate the matrix of R's to form $A_{3,1}$:

$$A_{3,1} \leftarrow \begin{bmatrix} \begin{pmatrix} R_{2,1} \\ R_{2,2} \end{pmatrix} \end{bmatrix}$$

(6) Performing the final QR decomposition on $A_{3,1}$ we have

$$\begin{bmatrix} A_{1,1} \\ A_{1,2} \\ A_{1,3} \\ A_{1,4} \end{bmatrix} = \left[\begin{array}{c|c|c|c} Q_{1,1} & & & \\ \hline & Q_{1,2} & & \\ \hline & & Q_{1,3} & \\ \hline & & & Q_{1,4} \end{array}\right] \left[\begin{array}{c|c} Q_{2,1} & \\ \hline & Q_{2,2} \end{array}\right] Q_{3,1}R_{3,1}$$

(7) The upper-triangular matrix R is equal to $R_{3,1}$, and the orthogonal matrix Q is the product

$$Q = \left[\begin{array}{c|c|c|c} Q_{1,1} & & & \\ \hline & Q_{1,2} & & \\ \hline & & Q_{1,3} & \\ \hline & & & Q_{1,4} \end{array}\right] \left[\begin{array}{c|c} Q_{2,1} & \\ \hline & Q_{2,2} \end{array}\right] Q_{3,1},$$

but since R is upper-triangular, it is often more efficient to calculate $Q = AR^{-1}$.

The QR decomposition in steps 2,4, and 6 is generally chosen to be Householder's, but any method could be used without changing the high-level flow of the algorithm. Figure 1.12 clearly summarizes the binary-tree structure of TSQR. The optimal number of vertical partitions depends on the height-to-width ratio of the input matrix and on the particular architecture that is used to execute the algorithm.

It is easy to see the speedup in the early steps of the TSQR algorithm due to the vertical partitioning and parallelization, but it is not obvious that the final steps of TSQR maintain the algorithm's efficiency. In the first step, the vertical partitions ameliorate the tall-skinny communication problem, and the "short" QR decompositions can be performed in parallel, which is another

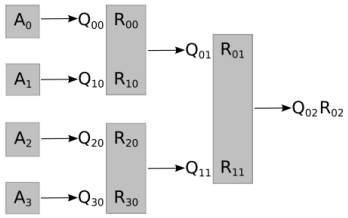


Figure 1.12: The TSRQ algorithm's binary tree structure. Figure from [26].

source of speedup. However, as we descend the TSQR tree, the matrices become taller and the parallelization disappears, so these steps become more costly. As long as the input matrix has sufficiently more rows than columns, then the intermediate R submatrices shrink, so that the final QR performed without parallelization is smaller than the original matrix. This means that the final QR is still inexpensive compared to performing a traditional QR on the original matrix.

To exhibit this shrinking, suppose we perform a two-split TSQR on a 64×4 matrix. The first step is to perform QR's on the two 32×4 matrices that result from the split. This produces two 4×4 matrices, $R_{1,1}$ and $R_{1,2}$. The final step is to concatenate these two matrices and perform a QR decomposition on the resulting 8×4 matrix. In total, the number of operations required to do this TSQR decomposition is equal to the number of operations required to find the QR of one 32×4 matrix and one 8×4 matrix, and this costs less than performing a QR on the original 64×4 matrix. In general, if the input matrix is size $m \times n$ and the number of splits in the TSQR algorithm is s, then the final QR will be less expensive than the initial QR if $m \geq n^s$.

1.6.3 Numerical Experiments on rSVD

To test the ability of TSQR to accelerate rSVD, we built implementations of TSQR that run on (1) multiple CPU cores and on (2) multiple blocks of a single GPU card. The multiple CPU code uses MATLAB's Parallel Computing Toolbox, while the GPU code uses Magma libraries [66]. Our code can also be used to run on multiple GPU cards, but due to limited resources, we exclude this case from our tests.

We test four different rSVD implementations: rSVD on the CPU, rSVD on the GPU, rSVD

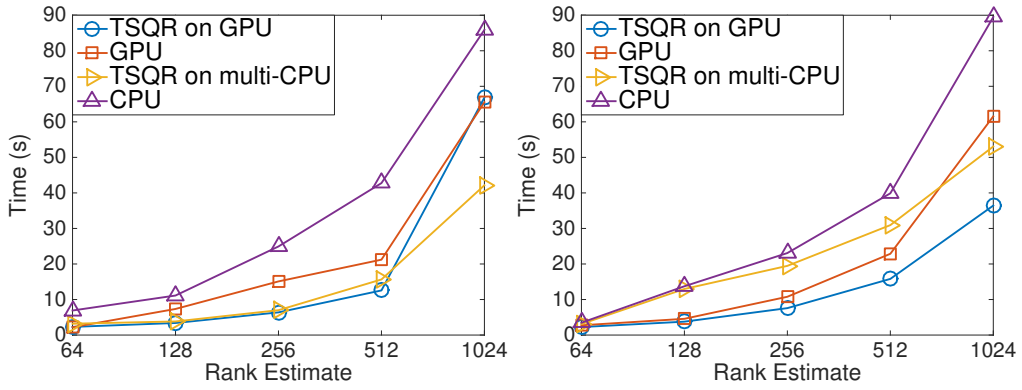


Figure 1.13: Timing the rSVD algorithm for various rank estimates and matrix sizes: on the left, the matrix is $65{,}536 \times 8{,}192$. On the right, the matrix is $131{,}072 \times 4{,}096$ (twice the number of rows and half the number of columns).

on the GPU with a multi-CPU-core TSQR, and rSVD on the GPU with a single-GPU TSQR. The matrices in our tests were randomly generated with each entry drawn independently from the univariate normal distribution, so they were all of full rank. The rank estimate is the number of columns that the matrix Y has at the time of the QR decomposition, so the speedup that TSQR offers would be the same even if the test matrices were square. Both tests (and both the multi-CPU and GPU TSQR algorithms) employed sixteen vertical partitions for the TSQR step, which we determined to be near optimal through testing. We see from the tests that both TSQR on the GPU and on multi-CPU architectures provide significant speedup. TSQR on the GPU is most effective for extremely tall-skinny matrices, while TSQR on the CPU outperforms its GPU counterpart when the matrix is closer to square.

1.6.4 Numerical Experiments on fMRI Brain Scan Data

Functional MRI brain scan data sets are inherently four-dimensional, so unwrapping the dataset into a two-dimensional array creates a tall-skinny matrix with hundreds of times more rows than columns. The data set used in the following experiments has 230-times more rows than columns. Also, the enormity of these data sets makes low rank recovery models intractable without parallelization. Figure 1.14 shows the total time spent performing SPCP on one of these data sets,

as well as the total time spent performing rSVDs, performing the QR step of the rSVD, and moving data (labeled as the "overhead" cost). The results of this test are included in Figures 1.15 and 1.16.

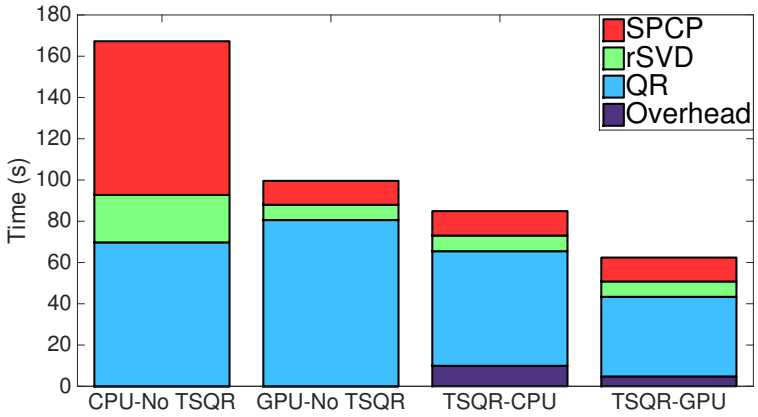


Figure 1.14: The total time spent performing various tasks during a run of SPCP on fMRI brain scan data. The test "TSQR-CPU" uses a hybridized architecture, running TSQR on multiple CPU's but running the rest of SPCP on the GPU. Both tests "GPU-No TSQR" and "TSQR-GPU" were run entirely on the GPU.

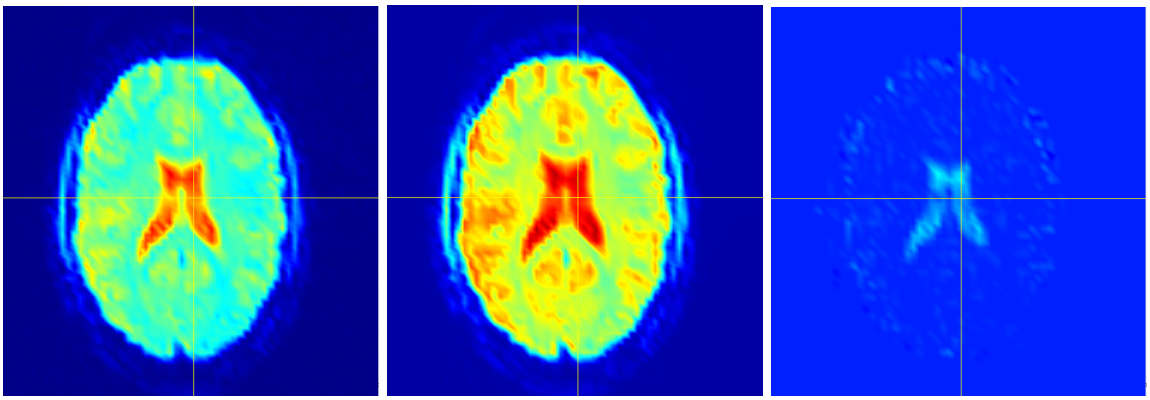


Figure 1.15: The results of SPCP on fMRI brain scan data. Activity is measured in Blood-oxygen-level dependent (BOLD) signal. (From left to right: original image, low rank image recovered by SPCP, and sparse image recovered by SPCP.)

The datasets used for testing were taken from a study of the human brain's response to uncomfortable stimuli, specifically, the feeling of submerging a hand in hot or cold water. Analyzing these scans to find common neurological responses to these stimuli is difficult due to the enormous amount of error in the data. There is uniformly distributed noise due to constant ancillary physiological activity, and there are also sparsely distributed groups representing neurological structures

that should all exhibit the same behavior. The ventricles, for example, are filled with cerebrospinal fluid (CSF), which does not contribute to neurological communication, so they should not be active. All signals observed in the ventricles should be treated as sparsely structured outliers. SPCP removes the uniform noise and, most remarkably, correctly identifies signals in the brain's ventricles as outliers. In Figure 1.15, the largest ventricles are the two structures in the center of the brain. The rightmost image shows that the majority of the noise contained in S is from these ventricles.

The other two major components of the brain are white and gray matter. The activity we are hoping to observe takes place in the gray matter, so ideally, SPCP would remove most signals from the white matter regions. However, the regions of white matter are more difficult to distinguish than the regions of CSF, and SPCP removes about equal amounts of noise from the white matter as it does from the gray. If we let S_{gm} be the gray-matter component of S, and define S_{wm} and S_{csf} similarly, Figure 1.16 shows the average BOLD signal in S_{gm}, S_{wm}, and S_{csf} for each frame in time. These data were normalized by the average original signal in the corresponding regions.

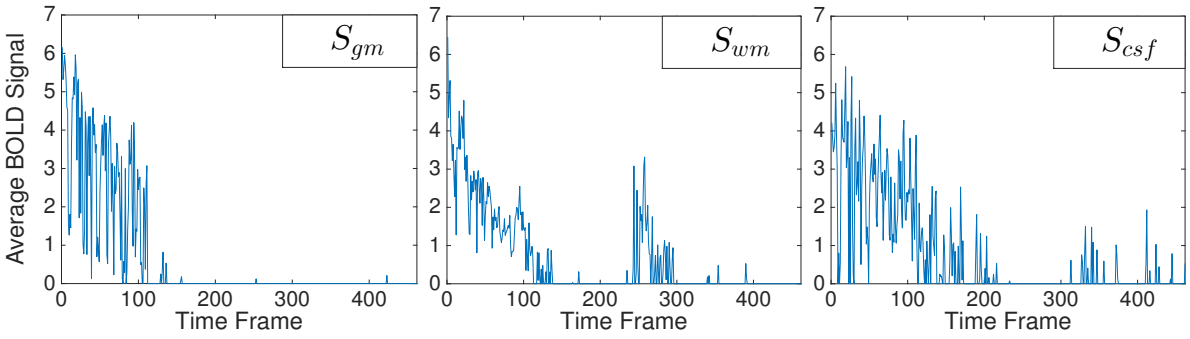


Figure 1.16: The average BOLD signal in different regions of S. The averages were normalized by the average original signal in the corresponding regions.

It is clear that S_{csf} contains more signal than the other two regions. For S_{gm}, SPCP detects noise in only the first 100 time slices. The removed signal from the white matter is more distributed over time, and the total amount of noise in S_{gm} and S_{wm} is comparable. These results suggest that SPCP correctly identifies outliers in the fMRI data, especially within the regions of CSF.

Because of the size and tall-skinny structure of fMRI brain scan data, this analysis using SPCP is only feasible using parallel programming. Figure 1.14 shows how incorporating the TSQR

algorithm into SPCP on the GPU more than doubles the speed of the algorithm. While the GPU offers enormous speedup over the CPU implementation of SPCP, the rSVD step on the GPU is about 90% of the algorithm by time, and it is only about 55% of the algorithm on the CPU. This discrepancy is ameliorated by TSQR.

1.6.5 Comparing the Non-Convex and Communication-Avoiding Approaches

Although reducing communication in the low rank recovery algorithm provides noticeable speedup, the non-convex approach is often even faster. In Figure 1.17, we compare our non-convex SPCP algorithm to a convex solver using TSQR for the rSVD step. Both solvers were decomposing the same $106{,}496 \times 462$ brain-scan data-matrix used in section 6.4, and both were run on the GPU.

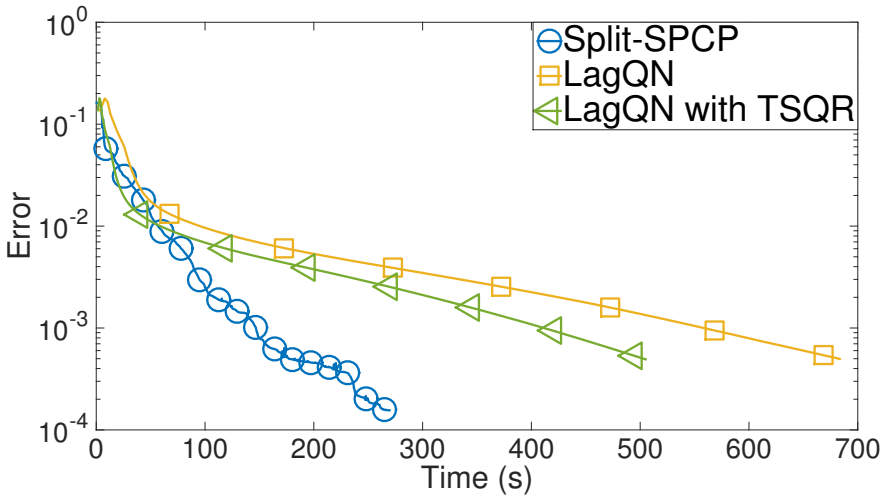


Figure 1.17: Comparing Split-SPCP with the convex solver LagQN with TSQR and without TSQR. Split-SPCP was initialized with the rank bound $k = 462$, the maximum possible rank of the low rank component. One marker represents 35 iterations.

As with previous tests, a reference solution (L_{ref}, S_{ref}) was found by solving LagQN to high accuracy, and the error is measured as the normalized distance from the reference objective value. The rank of L_{ref} was 423, and Split-SPCP was initialized with the rank bound $k = 462$. Since X had 462 columns, this was the largest possible rank bound, and Split-SPCP still greatly outperformed the convex solvers. Also, both solvers found solutions of similar quality. At an error of 5×10^{-4}, the low rank component found by LagQN had rank 427, and the low rank component

found by Split-SPCP had rank 425. Similarly, the sparse component of LagQN had sparsity 58.5%, and the sparse component of Split-SPCP had sparsity 58.3%, while S_{ref} was 58.5% sparse. Non-convex SPCP recovered nearly the same solution as the parallelized convex algorithm at a much smaller cost.

1.7 Conclusion

In this manuscript, we developed two new methods to accelerate regularized low rank recovery models. Both of these methods offer significant speedup over existing approaches, and both models are amenable to further speedup using parallelized computational architectures.

For our first approach (Split-SPCP), we use the factorization $L = UV^T$ to induce an upper-bound on the rank of L and eliminate the iterative SVDs normally required for nuclear norm minimization. By marginalizing the regularized S variable, we create an objective that is Lipschitz-differentiable when the loss function is strongly convex. Using L-BFGS on the GPU, we showed that our non-convex SPCP model converges much faster than existing methods. We also developed a certificate to determine whether our model had converged to a minimum or to a spurious stationary point, showing empirically and theoretically that local minima do not pose a serious problem for our non-convex model.

Our second approach (TSQR) maintained the convexity of the low rank recovery model and accelerated the iterative SVDs by reducing the algorithm's communication. Combining the tall-skinny QR decomposition with the rSVD algorithm, we solved an inherent communication problem that exists when running the rSVD on a GPU. This reduction in communication accelerated convex SPCP solvers by 40% in our experiments. While the non-convex model outperformed TSQR, it is an important contribution, showing how to better parallelize the original convex formulation on GPU architectures.

Our accelerated solvers provide regularized low rank recovery models with new applications. The non-convex SPCP approach can process video streams in real-time. Both of our models allow for faster decomposition of extremely large data sets, such as fMRI brain scans, which makes the

low rank recovery of these data sets feasible.

Chapter 2

Tensor RPCA Outperforms Matrix RPCA for High-Dimensional Data Recovery

2.1 Chapter Summary

We derive performance guarantees for tensor-based RPCA using atomic-norm regularization. These results improve upon existing guarantees and show that using tensor RPCA yields better recovery of the low-rank and sparse components of a tensor than using a sum-of-nuclear-norms approach or using matrix RPCA on the matricized tensor. We then present a non-convex relaxation of the atomic norm using a higher-order generalization of Burer-Monteiro factorization that is popular in matrix recovery. We provide several algorithms to solve this non-convex model, and we supplement them with a certificate to bound the distance from a recovered solution to the global optimum. The strong performance of our approach is demonstrated in numerical experiments, where we show that our non-convex model can reliably recover tensors with ranks larger than all of their side lengths.

2.2 Introduction

Tensors, or multidimensional arrays, are generalizations of matrices to higher order. A vector is an order-1 tensor, a matrix is an order-2 tensor, and an order-K tensor contains K indices of information. Tensors were first used as tools for data analysis in the psychometics community, where researchers used tensor decompositions to study fMRI datasets, which are more naturally represented as tensors rather than matrices [49, 68]. Since then, tensors have established a place in

chemometrics, computer vision, compressed sensing, data mining, and higher-order statistics [49]. With modern datasets growing quickly in both size and complexity, tensor-based algorithms offer more natural approaches analyzing multidimensional data.

However, this is not to say that tensor-based methods only offer convenience. Higher-order tensors and their decompositions have many useful mathematical properties that their matrix correlates do not. Matrix decompositions often enforce orthogonality constraints on their factors (for example, the singular value decomposition), and these constraints are generally not present in the information we would like to extract from the data matrix. Tensor decompositions have no such constraints, and they also enjoy uniqueness under weak conditions [49]. These qualities are what give higher-order tensors their advantage over matrices. As a particularly influential example, researchers in machine learning have recently exploited these properties of tensor decompositions to use them for parameter estimation in latent-variable models. This work has lead to advances in Gaussian mixture models, hidden Markov models, and some of the most general guarantees for neural-network performance [2, 44].

In this chapter, we will focus on low-rank tensor recovery through a new formulation of tensor RPCA. Low-rank matrix recovery is a versatile tool used for collaborative filtering, dimension reduction, and background subtraction; it is also supplemented with many mathematical guarantees [17]. Low-rank tensor recovery can be used for the same applications, and, as we will show, tensor recovery is often better suited for handling multidimensional datasets. Our tensor RPCA algorithm can also be interpreted as computing a low-rank decomposition of a tensor with gross, sparsely distributed errors.

The rest of this chapter is outlined as follows. In section 2.3, we will provide an overview of some analytic and algebraic properties of tensors. This will include a description of existing methods for tensor RPCA. In section 2.4, we will present recovery guarantees for our formulation of tensor RPCA using atomic-norm regularization. We will also compare these guarantees to existing guarantees for both tensor and matrix RPCA. Sections 2.4 through 2.6 contain the proof for our main result.

In section 2.7, we will develop a non-convex representation of the tensor atomic norm that can be seen as a higher-order generalization of the Burer-Monteiro-factorization approach that is popular in low-rank matrix recovery algorithms. We will develop three algorithms to fit our non-convex model. These include an alternating minimization approach that provably converges to a stationary point, as well as a smoothing technique that allows our model to be fit using first-order methods. In section 2.8, we will provide a certificate for optimality for our model, which includes a bound on the suboptimality of any point. Finally, we present numerical experiments in section 2.9 that demonstrate the superiority of our model.

Taken together, our results suggest that preserving the structure of multidimensional datasets allows for significantly improved recovery. Hence, atomic-norm regularization outperforms existing methods methods for RPCA involving matricization.

2.3 Tensor Preliminaries

This section introduces definitions and properties of tensors that we will use. All of this information and more can be found in [31, 49, 75], for example.

Let $X \in \mathbb{R}^{d_1 \times \cdots \times d_K}$ be an order-K tensor with side lengths $d_1, \cdots, d_K$. X can be described in terms of its entries and indices: $X := [[x_{i_1,\cdots,i_K}]]$, where the brackets $[[\cdot]]$ indicate that we are considering the tensor defined by these entries. The *fibres* of X *along its* k^{th} *mode* are the vectors obtained by holding all but one of the indices of X fixed and varying the k^{th} index. If X is an order-2 tensor (or, matrix), the fibres of X along its first mode are the columns of X, and the fibres along its second mode are its rows.

In some cases, it is useful to *matricize* a tensor, so that it is represented as a matrix. The matricization of an order-K tensor X along its k^{th} mode is denoted $X_{(k)}$. $X_{(k)}$ is formed by taking the mode-k fibres of X and making them the columns of $X_{(k)}$.

There are several products that are useful when working with tensors. The outer product,

which we will denote $\otimes$, is the map

$$\otimes : \mathbb{R}^{d_1} \times \cdots \times \mathbb{R}^{d_K} \to \mathbb{R}^{d_1 \times \cdots \times d_K} : (u_1, \cdots u_K) \mapsto [[(u_1)_{i_1}, (u_2)_{i_2}, \cdots, (u_1)_{i_K}]], \tag{2.1}$$

where $(u_1)_{i_1}$ is the i_1^{th} component of u_1, for example. For a simple example, we see that for $u \in \mathbb{R}^m$, $v \in \mathbb{R}^n$, $u \otimes v$ is the $m \times n$ matrix uv^T. We will also use the *Khatri-Rao* product, which we denote $\odot$. For two matrices $A \in \mathbb{R}^{m \times n}$. $B \in \mathbb{R}^{p \times n}$ with the same number of columns, we have

$$A \odot B := \begin{pmatrix} a_{1,1}\mathbf{b_1} & \cdots & a_{1,n}\mathbf{b_n} \\ \vdots & \vdots & \vdots \\ a_{m,1}\mathbf{b_1} & \cdots & a_{m,n}\mathbf{b_n} \end{pmatrix} \in \mathbb{R}^{m \cdot p \times n}, \tag{2.2}$$

where $\mathbf{b_i}$ is the i^{th} column of B. A matricized tensor can be expressed neatly using the Khatri-Rao product [49]. We will revisit this in section 2.7.

Every tensor $X \in \mathbb{R}^{d_1 \times \cdots d_K}$ admits a *CP-decomposition* of the form

$$\begin{aligned} X &= \sum_{r=1}^{R} \gamma_r (u_r^{(1)} \otimes u_r^{(2)} \otimes \cdots \otimes u_r^{(K)}), \\ &= \sum_{r=1}^{R} (a_r^{(1)} \otimes a_r^{(2)} \otimes \cdots \otimes a_r^{(K)}) \end{aligned} \tag{2.3}$$

where R is minimal. We call R the *CP-rank* of X. Here $u_r^{(k)}$ are unit vectors with respect to the Euclidean norm, and $\gamma_r = \prod_{k=1}^{K} \|a_r^{(k)}\|$. The *factor matrices* of X are then the matrices $A^{(1)}, \cdots, A^{(K)}$ that have $a_r^{(1)}, \cdots, a_r^{(K)}$ as their columns, respectively. It is also sometimes convenient to write $X = [[A^{(1)}, \cdots, A^{(K)}]]$ when the factor matrices of X are known; this representation of X is using *Kruskal's notation*.

The tensor *atomic norm*, which we will denote as $\|\cdot\|_*$ is defined as follows:

$$\|X\|_* = \inf \quad \left\{ \sum_{r=1}^{R} |\gamma_r| : X = \sum_{r=1}^{R} \gamma_r (w_r^{(1)} \otimes \cdots \otimes w_r^{(K)}),\ \|w_r^{(1)}\| = \cdots = \|w_r^{(K)}\| = 1 \right\}.$$

This norm is also known as the *higher-order nuclear norm* [75], but we will follow the convention of [23], where $\|\cdot\|_*$ is the norm induced by the Minkowski functional (or gauge) on the convex hull of the set of rank-1 tensors with unit-Euclidean norm. In the case $K = 2$, the atomic norm is the

matrix nuclear norm (also known as the trace norm), which is equal to the sum of the singular values of a matrix. In section 2.4, we will show that atomic-norm regularization encourages low-rank structure.

The dual to the nuclear norm is the spectral norm, $\|\cdot\|$, which is defined as [31, 75]:

$$\|X\| = \max_{\|w_r^{(1)}\|=\cdots=\|w_r^{(K)}\|=1} \langle X, w_r^{(1)} \otimes \cdots \otimes w_r^{(K)} \rangle. \tag{2.4}$$

There are two interesting qualities of this norm that we will use. First, there exists a rank-1, unit-Euclidean-norm tensor that maximizes this inner-product, so the maximum is well-defined [75]. Throughout this paper, we will denote this maximizing tensor as W. Next, suppose a third-order tensor $\mathcal{X} = [[A, B, C]]$ (using Kruskal's notation). If we let $W = w_1 \otimes w_2 \otimes w_3$ be the tensor that maximizes the above inner product, then w_1 lies in the columnspace of A, and the analogous results holds true for w_2 and w_3 as well [75]. We will be working with the columnspaces of factor matrices often, so we will define projection operators onto these spaces in the sequel.

2.3.1 Projection Operators and the Tucker-Rank

Let $X = [[U, V, W]]$, and let $\mathcal{Z} = [[A, B, C]]$ be order-3 tensors. Define the projection operator $\mathcal{P}_{U,V,W} : \mathcal{Z} \mapsto [[\mathcal{P}_U(A), \mathcal{P}_V(B), \mathcal{P}_W(C)]]$, where $\mathcal{P}_U$, for example, projects matrices onto the columnspace of U. We will use operators of this form often. For convenience, we will use the notation of [75] to define the following projections:

$$\begin{aligned}
\mathcal{P}_{X^0} &:= \mathcal{P}_{U,V,W} \\
\mathcal{P}_X &:= \mathcal{P}_{U,V,W} + \mathcal{P}_{U^\perp,V,W} + \mathcal{P}_{U,V^\perp,W} + \mathcal{P}_{U,V,W^\perp} \\
\mathcal{P}_{X_1} &:= \mathcal{P}_{U^\perp,V^\perp,W} \\
\mathcal{P}_{X_2} &:= \mathcal{P}_{U^\perp,V,W^\perp} \\
\mathcal{P}_{X_3} &:= \mathcal{P}_{U,V^\perp,W^\perp} \\
\mathcal{P}_{X_4} &:= \mathcal{P}_{U^\perp,V^\perp,W^\perp} \\
\mathcal{P}_{X^\perp} &:= \mathcal{P}_{X_1} + \mathcal{P}_{X_2} + \mathcal{P}_{X_3} + \mathcal{P}_{X_4}.
\end{aligned}$$

Analogous projections exist for tensors of higher order, but they will not be useful to us.

We will also need to define the *Tucker-rank* of a tensor. Suppose we have tensor $X = [[U^{(1)}, \cdots, U^{(K)}]] \in \mathbb{R}^{d_1 \times \cdots \times d_K}$. The Tucker rank of X is the tuple $(\text{rank}(U^{(1)}), \cdots, \text{rank}(U^{(K)})) =: (r_1, \cdots, r_K)$. It follows that r_k is the dimension of the range of $\mathcal{P}_{U^{(k)}}$. Another useful measure of am order-3 tensor's complexity is the weighted average of its Tucker ranks:

$$\overline{r}(T) := \sqrt{\frac{r_1 r_2 d_3 + r_1 r_3 d_2 + r_2 r_3 d_1}{d_1 + d_2 + d_3}}.$$

2.3.2 Norms for Tensors and Operators on Tensors

We will be working with linear operators that act on tensors, and we will need to define a norm for these operators. Let $\mathcal{Q} : \mathbb{R}^{d_1 \times d_2 \times d_3} \to \mathbb{R}^{d_1' \times d_2' \times d_3'}$ be an operator mapping one tensor product of vector spaces to another. We will define the norm of $\mathcal{Q}$ as

$$\|\mathcal{Q}\| := \sup_{\substack{X \in \mathbb{R}^{d_1 \times d_2 \times d_3} \\ \|X\|_F \leq 1}} \|\mathcal{Q}X\|_F. \tag{2.5}$$

This definition is a trivial generalization of the commonly used norm for operators acting on matrices.

Finally, we will need higher-dimensional generalizations of the matrix ℓ_1 and ℓ_∞ norms. For tensor $X \in \mathbb{R}^{d_1 \times d_2 \times d_3}$, we can define the useful norms

$$\|X\|_{\text{sum}} = \sum_{(i,j,k) \in [d_1] \times [d_2] \times [d_3]} |X_{i,j,k}|, \tag{2.6}$$

and

$$\|X\|_{\max} = \max_{(i,j,k) \in [d_1] \times [d_2] \times [d_3]} |X_{i,j,k}|. \tag{2.7}$$

Because these tensor-norms are not induced by the topologies of the ℓ_1 or ℓ_∞ spaces, it is more appropriate to use the notation above rather than denote the norms as $\|\cdot\|_1$ and $\|\cdot\|_\infty$, respectively.

2.3.3 Coherence

Exact recovery of a tensor through our RPCA model relies on the tensor having low coherence. For our proofs, we will adopt the measures of tensor coherence introduced in [75]. Recall that the

coherence of an r-dimensional linear subspace U of $\mathbb{R}^k$ is defined to be [19, 75]

$$\mu(U) := \frac{k}{r} \max_{1 \le i \le k} \|\mathcal{P}_U e_i\|^2 = \frac{\max_{1 \le i \le k} \|\mathcal{P}_U e_i\|^2}{k^{-1} \sum_{i=1}^k \|\mathcal{P}_U e_i\|^2}. \tag{2.8}$$

For tensor $X = [[U, V, W]] \in \mathbb{R}^{d_1 \times d_2 \times d_3}$, we can define one measure of coherence as

$$\mu(X) := \max\{\mu(\text{range}(\mathcal{P}_U)), \mu(\text{range}(\mathcal{P}_V)), \mu(\text{range}(\mathcal{P}_W))\}, \tag{2.9}$$

where $\mathcal{P}_U$, for example, is the projection onto the column space of U. The following lemma will be used in the sequel:

Lemma 2. ***(Lemma 2 in [75]):*** *Let $X \in \mathbb{R}^{d_1 \times d_2 \times d_3}$ be a third-order tensor, let $d_+ := d_1 + d_2 + d_3$, and let $\overline{r}(X) := \sqrt{(r_1(X) r_2(X) d_3 + r_1(X) r_3(X) d_2 + r_2(X) r_3(X) d_1)/d_+}$. Then*

$$\max_{i,j,k} \|\mathcal{P}_X(e_i \otimes e_j \otimes e_k)\|_F^2 \le \frac{\overline{r}(X) d_+}{d_1 d_2 d_3} \mu(X)^2. \tag{2.10}$$

The other measure of coherence introduced in [75] is

$$\alpha(X) := \sqrt{d_1 d_2 d_3 / \overline{r}(X)} \|W\|_{\max}, \tag{2.11}$$

where W is the one define by the spectral norm of X, as described in (2.4), so $W = \mathcal{P}_X^0 W$, $\|W\| = 1$, and $\langle X, W \rangle = \|X\|_*$. For our recovery results, we will assume that the low-rank component $X \in \mathbb{R}^{d_1 \times d_2 \times d_3}$ has low coherence, so it satisfies $\mu(X) \le \mu_0$ and $\alpha(X) \le \alpha_0$.

2.3.4 Further Notation

We will often use the notation $a \wedge b$ to mean $\min(a, b)$, and $a \vee b := \max(a, b)$ for scalars a and b. Also, we will let $d_+ := d_1 + d_2 + d_3$.

Furthermore, our main result involves the Lambert-W function of large inputs, so we will describe the asymptotic behavior of $W(x)$ for large x. Let $f(x) := xe^x$. The Lambert-W function is defined as f^{-1}. It has the following asymptotic behavior [16]:

$$\text{For } x \gg 1: \quad W(x) \sim \log(x) \left(1 - \frac{\log(\log(x))}{\log(x) + 1}\right). \tag{2.12}$$

We are now prepared to state our results.

2.4 Tensor RPCA

We will show that the existence of a approximate dual certificate is enough to ensure that the following program

$$\begin{aligned} \min_{X,S} \quad & \|X\|_* + \lambda \|S\|_{\text{sum}} \\ \text{subject to:} \quad & X + S = \mathcal{Z} \end{aligned} \tag{2.13}$$

recovers X and S exactly. Our main result is Theorem 2.

Theorem 2. *Suppose tensor $X \in \mathbb{R}^{d_1 \times d_2 \times d_3}$ satisfies $\mu(X) \leq \mu_0$, $\alpha(X) \leq \alpha_0$, and $\overline{r}(X) = r$, and let tensor $S \in \mathbb{R}^{d_1 \times d_2 \times d_3}$ have a support set Ω that is uniformly distributed among all sets of cardinality m. There then exists a positive constant β so that (2.13) with $\lambda = (d_+)^{-1/2}$ exactly recovers X and S with probability $1 - (d_+)^{-1-\beta}$, provided that*

$$\overline{r}(X) \leq \rho_r \left(\frac{d_1 d_2 d_3 - m}{(d_1 + d_2 + d_3) W(d_1 d_2 d_3 - m) \mu^2} \right)^{1/2} \quad \text{and} \quad m \leq \left(\rho_s d_1 d_2 d_3 \wedge \frac{\sqrt{\rho_r \log(n) W(n)} + W(n)}{\rho_r \log(n)} \cdot \right) \tag{2.14}$$

Here, ρ_r, ρ_s are numerical constants, and $W(\cdot)$ is the Lambert-W function.

These are the best guarantees that have been proven for a tensor RPCA, and they are near optimal. For large d_1, d_2, and d_3, (2.12) implies that removing a factor of $\log(d_1 d_2 d_3)^{-(1/2-\epsilon)}$ for some small $\epsilon > 0$ would imply that tensors of all Tucker-ranks could be exactly recovered using the suggested program. In fact, our numerical experiments in section 2.9 indicate that our non-convex reformulation of (2.13) reliably recovers tensors of arbitrarily large Tucker-ranks. While this does indicate the strength of our approach to tensor RPCA, this also implies that the Tucker-rank is a poor measure of the complexity of high-rank tensors. For more precise performance guarantees, we must formulate bounds on the CP-rank of recoverable tensors, which will require new theoretical tools.

Outside of this work, the best guarantees for a tensor RPCA are from [40]. The relevant theorem from this paper is listed below.

Theorem 3. ***(Theorem 1 from [40])****: Consider the following sum-of-nuclear-norms model for tensor RPCA:*

$$\min_{X,S} \quad \sum_{i=1}^{K} \lambda_i \|X_{(i)}\|_* + \|S\|_1, \tag{2.15}$$

$$\text{subject to:} \quad X + S = \mathcal{Z}, \tag{2.16}$$

where $X_{(i)}$ is the unfolding of tensor X along its i^{th} dimension. Let $X = [[U_1, U_2, \cdots, U_K]]$, and define r_k to be the rank of the linear space spanned by the columns of U_k (so that r_k is the k^{th} component of the Tucker rank of X). Let $d_i^{(1)} = \max(d_i, \Pi_{j \neq i} d_j)$, and $d_i^{(2)} = \min(d_i, \Pi_{j \neq i} d_j)$. For some constant C, and with $\lambda_i = \sqrt{d_i^{(1)}}$, (2.15) exactly recovers X and S with probability $1 - C d_1^{-3}$ as long as

$$r_k \leq C_r K^{-2} \frac{\mu'^{-1} d_k^{(2)}}{\log^2 d_k^{(1)}}, \quad \text{and} \quad |\Omega| \leq \rho d_k^{(1)} d_k^{(2)}, \tag{2.17}$$

for some constants C_r and ρ, and incoherence parameter μ'.

For a direct comparison of the rank bounds required by these two theorems, consider the case where T is an order-3 tensor with $d_1 = d_2 = d_3 = d$. Our result says that exact recovery is likely when

$$r_1 r_2 + r_1 r_3 + r_2 r_3 \leq \rho_r \left(\frac{d^3 - m}{3dW(d^3 - m)\mu^2} \right), \tag{2.18}$$

and the result from [40] says that exact recovery is likely when

$$r_k \lesssim C_r \left(\frac{d}{\log^2(d)\mu'} \right). \tag{2.19}$$

It should be noted that for an order-3 tensor with $d_1 = d_2 = d_3 = d$, the maximum possible Tucker rank is $r_1 = r_2 = r_3 = d$, so our program allows for exact recovery of a tensor with arbitrary Tucker rank to within a factor of less than $\log(d)^{-1/2}$. This means that we have exhausted the utility of the Tucker-rank approximation to a tensor's CP-rank, and future work should investigate tighter bounds using the CP-rank of a tensor.

Now suppose further that $r_1 = r_2 = r_3 = r$, m is small, and d is large. Our guarantees require that

$$r \lesssim \rho_r^{1/2} \left(\frac{d}{(\log(d) - \log(\log(d)))^{1/2}} \right). \tag{2.20}$$

This is only a factor of $\log(d)^{3/2}$ better than (2.17) (and, because (2.17) agrees with the guarantees for matrix RPCA on the matricized tensor, we are a factor or $\log(d)^{3/2}$ better than matrix RPCA as well). However, this is because a rank bound of $r \lesssim d$ is the best we can possibly prove using the Tucker-rank measure. In section 2.9, we show that our model for tensor RPCA can reliably recover tensors of arbitrarily large Tucker-rank.

To prove Theorem 2, we follow a strategy similar to the one used in [17]. In Lemma 3, we will show that Theorem 2 holds provided that a certain dual certificate exists. In sections 2.5 and 2.6, we will show that the dual certificate given by Lemma 3 can be constructed with high probability.

Lemma 3. *Suppose* $\|\mathcal{P}_\Omega \mathcal{P}_X\| < \frac{1}{2}$ *and* $\lambda \in (0,1)$. *Then* (X,S) *is the unique (and exact) solution to* (2.13) *if there exists tensors* $W^\perp$ *and* F *satisfying*

$$W + \mathcal{P}_{X^\perp} W^\perp = \lambda(\operatorname{sgn}(S) + F + \mathcal{P}_\Omega D)$$

$$\text{subject to:} \quad M = X + S$$

where $\|W^\perp\| \leq \frac{1}{4}$, $\mathcal{P}_\Omega F = 0$, $\|F\|_{\max} \leq \frac{1}{4}$, *and* $\|\mathcal{P}_\Omega D\|_F \leq \frac{1}{8}$. *(As before and throughout this paper,* W *satisfies* $\mathcal{P}_X^0 W = W$, $\|W\| = 1$, *and* $\langle W, X\rangle = \|X\|_*$. *Such a* W *always exists, so we do not need to prove its existence for our certificate.)*

Proof. Let Δ be a perturbation away from the supposed optimal point (X,S). Let $W + \mathcal{P}_{X^\perp} W'^\perp$ be an (arbitrary) element of the subdifferential of the nuclear norm at L, and let $\operatorname{sgn}(S) + F'$ be an element of the subdifferential of the ℓ_1-norm at S. We then have that $(X+\Delta, S-\Delta)$ is a feasible point of (2.13), and

$$\|X+\Delta\|_* + \lambda\|S-\Delta\|_{\text{sum}} \geq \|X\|_* + \langle \Delta, W + \mathcal{P}_{X^\perp} W'^\perp\rangle + \lambda\left(\|S\|_{\text{sum}} - \langle \Delta, \operatorname{sgn}(S) + F'\rangle\right) \quad (2.21)$$

The projection $\mathcal{P}_{X^\perp}$ is orthogonal, so $\langle \Delta, \mathcal{P}_{X^\perp} W^\perp\rangle = \langle \mathcal{P}_{X^\perp}\Delta, W^\perp\rangle$, and by the duality of the tensor nuclear and spectral norms, there exists a $W'^\perp$ satisfying $\|W'^\perp\| = \frac{1}{4}$ and $\langle \mathcal{P}_{X^\perp}\Delta, W^\perp\rangle = \frac{1}{4}\|\mathcal{P}_{X^\perp}\Delta\|_*$. Similarly, we can choose F' so that $\langle F', \Delta\rangle = \frac{1}{4}\|\mathcal{P}_{\Omega^\perp}\Delta\|_{\text{sum}}$. Hence,

$$\|X+\Delta\|_* - \|X\|_* + \lambda(\|S-\Delta\|_{\text{sum}} - \|S\|_{\text{sum}}) \geq \frac{1}{4}\left(\|\mathcal{P}_{X^\perp}\Delta\|_* + \lambda\|\mathcal{P}_{\Omega^\perp}\Delta\|_{\text{sum}}\right) + \langle \Delta, W - \operatorname{sgn}(S)\rangle \quad (2.22)$$

We would like to show that the right side of this inequality is positive unless $\Delta = 0$. To this end, we can bound the magnitude of the last term.

$$|\langle \Delta, W - \operatorname{sgn}(S)\rangle| = |\langle \lambda F + \lambda \mathcal{P}_\Omega D - \mathcal{P}_{T^\perp} W^\perp\rangle| \tag{2.23}$$

$$\leq |\langle F, \Delta\rangle| + |\langle \mathcal{P}_\Omega D, \Delta\rangle| + |\langle \mathcal{P}_{X^\perp} W^\perp, \Delta\rangle| \tag{2.24}$$

$$< \frac{1}{4}\left(\|\mathcal{P}_{X^\perp}\Delta\|_* + \lambda\|\mathcal{P}_{\Omega^\perp}\Delta\|_{\text{sum}}\right) + \frac{\lambda}{8}\|\mathcal{P}_\Omega \Delta\|_F, \tag{2.25}$$

where we used the fact that $\mathcal{P}_\Omega$ is self-adjoint. This yields

$$\|X+\Delta\|_* - \|X\|_* + \lambda(\|S-\Delta\|_{\text{sum}} - \|S\|_{\text{sum}}) \geq \frac{1}{4}\left(\|\mathcal{P}_{X^\perp}\Delta\|_* + \lambda\|\mathcal{P}_{\Omega^\perp}\Delta\|_{\text{sum}}\right) - \frac{\lambda}{8}\|\mathcal{P}_\Omega\Delta\|_F \tag{2.26}$$

The last term can be bounded.

$$\|\mathcal{P}_\Omega \Delta\|_F = \|\mathcal{P}_\Omega(\mathcal{P}_X + \mathcal{P}_{X^\perp})\Delta\|_F \tag{2.27}$$

$$\leq \|\mathcal{P}_\Omega \mathcal{P}_X \Delta\|_F + \|\mathcal{P}_{X^\perp}\Delta\|_F \tag{2.28}$$

$$\leq \frac{1}{2}\|\Delta\|_F + \|\mathcal{P}_{X^\perp}\Delta\|_F \tag{2.29}$$

$$\leq \frac{1}{2}\|\mathcal{P}_\Omega\Delta\|_F + \frac{1}{2}\|\mathcal{P}_{\Omega^\perp}\Delta\|_F + \|\mathcal{P}_{X^\perp}\Delta\|_F, \tag{2.30}$$

where we have used the fact that $\|\mathcal{P}_\Omega \mathcal{P}_X\| \leq \frac{1}{2}$

$$\|\mathcal{P}_\Omega\Delta\|_F \leq \|\mathcal{P}_{\Omega^\perp}\Delta\|_F + 2\|\mathcal{P}_{X^\perp}\Delta\|_F. \tag{2.31}$$

We now have

$$\|X+\Delta\|_* - \|X\|_* + \lambda(\|S-\Delta\|_{\text{sum}} - \|S\|_{\text{sum}}) \tag{2.32}$$

$$\geq \frac{1}{4}\left(\|\mathcal{P}_{X^\perp}\Delta\|_* + \lambda\|\mathcal{P}_{\Omega^\perp}\Delta\|_{\text{sum}}\right) - \frac{\lambda}{8}\left(\|\mathcal{P}_{\Omega^\perp}\Delta\|_F + 2\|\mathcal{P}_{X^\perp}\Delta\|_F\right) \tag{2.33}$$

$$\geq \frac{1}{4}(1-\lambda)\|\mathcal{P}_{X^\perp}\Delta\|_* + \frac{3\lambda}{8}\|\mathcal{P}_{\Omega^\perp}\Delta\|_{\text{sum}}. \tag{2.34}$$

This shows that the perturbation Δ leads to a strict increase in the objective, unless $\Delta = 0$. $\square$

To summarize, to ensure exact recovery, it suffices to find a tensor $W^\perp$ satisfying

$$\begin{cases} W^\perp \in \text{range}(X^\perp), \\ \|W^\perp\| < \frac{1}{4}, \\ \|\mathcal{P}_\Omega(W - \lambda \text{sgn}(S) + W^\perp)\|_F \leq \frac{\lambda}{8}, \\ \|\mathcal{P}_{\Omega^\perp}(W + W^\perp)\|_{\max} < \frac{\lambda}{4}. \end{cases} \tag{2.35}$$

Equivalently, we will find tensors W^L and W^S satisfying $W^L + W^S \in \text{range}(\mathcal{P}_{X^\perp})$ and the following:

$$\begin{cases} (a) \quad \|W^L\| < \frac{1}{8}, \\ (b) \quad \|\mathcal{P}_\Omega(W + W^L)\|_F \leq \frac{\lambda}{8}, \\ (c) \quad \|\mathcal{P}_{\Omega^\perp}(W + W^L)\|_{\max} < \frac{\lambda}{8}, \end{cases} \tag{2.36}$$

$$\begin{cases} (d) \quad \|W^S\| < \frac{1}{8}, \\ (e) \quad \|\mathcal{P}_{\Omega^\perp} W^S\|_{\max} < \frac{\lambda}{8}. \end{cases} \tag{2.37}$$

Similarly to the argument in [17], we will construct W^L using a golfing scheme described in section 2.5, and we will construct W^S as a the solution to a certain least-squares problem which is outlined in section 2.6.

2.5 Constructing W^L

Our construction of W^L uses a variation of the golfing scheme developed in [33, 34], and mirrors the adaptations used in [17, 19, 75]. Let $n := |\Omega^c| = d_1 d_2 d_3 - m$. We will create an i.i.d. uniformly distributed ordered set of triples in $[d_1] \times [d_2] \times [d_3]$, call it $\{(a_i, b_i, c_i) : 1 \leq i \leq n\}$. This set is created by sampling with replacement from Ω^c using the following process:

(1) Initialize $S_0 = \emptyset$.

(2) For $i = 1, 2, \cdots, n$, sample the triple (a_i, b_i, c_i) from S_{i-1} uniformly with probability $|S_{i-1}|/d_1 d_2 d_3$, and sample (a_i, b_i, c_i) uniformly from $\Omega^c \backslash S_{i-1}$ with probability $1 - |S_{i-1}|/d_1 d_2 d_3$.

(3) Set $S_i = S_{i-1} \cup \{(a_i, b_i, c_i)\}$.

The same scheme is used to construct a dual certificate for the tensor completion problem in [75], but in our case, we are sampling from Ω^c to analyze the support of S.

The utility of the golfing scheme comes from the fact that $\mathbb{P}((a_i, b_i, c_i) \in S_{i-1} | S_{i-1})$ is equal to the probability of the same event when the triples (a_i, b_i, c_i) are drawn as i.i.d. random variables. Also, the conditional distribution of (a_i, b_i, c_i) given S_{i-1} and the event $(a_i, b_i, c_i) \in S_{i-1}^c$ is uniform. Together, these properties imply that the points (a_i, b_i, c_i) are drawn unifromly from $[d_1] \times [d_2] \times [d_3]$ as i.i.d. random variables [75].

We split this set into n_2 subsets of cardinality n_1, creating a partition of the set $\Omega^c = \bigcup_{k=1}^{n_2} \Omega_k$, where $|\Omega_k| = n_1$. Notice that $n_1 n_2 \leq n$ due to non-empty intersections among the Ω_k. The constants n_1 and n_2 must be chosen appropriately; this is addressed in later subsections.

With the sets Ω_k defined, we can define the corresponding projections $\mathcal{P}_{\Omega_k}$. Finally, with $Y_0 = 0$, define the recursion

$$Y_j = Y_{j-1} + \frac{d_1 d_2 d_3}{n_1} \mathcal{P}_{\Omega_j} \mathcal{P}_X (W - Y_{j-1}), \tag{2.38}$$

where here and throughout, W is a tensor satisfying $\mathcal{P}_X^0 W = W$, $\|W\| \leq 1$, and $\langle W, X \rangle = \|X\|_*$. We will set $W^L = \mathcal{P}_{X^\perp} Y_{n_2}$, and prove that this choice of W^L satisfies conditions $(a), (b)$, and (c) in (2.36).

2.5.1 Proof of (a)

Fortunately, the authors of [75] have already provided the deviation bounds necessary to prove that the W^L defined above has low spectral norm with high probability. This section consists of an adaptation of their proof. The following two lemmas will be used:

Lemma 4. ***(Lemma 6 in [75])****: Let $\{(a_i, b_i, c_i)\}$ be an ordered set of independently and uniformly distributed samples from $[d_1] \times [d_2] \times [d_3]$. Assume that $\mu(X) \leq \mu_0$. Define $r := \overline{r}(X)$ and $d_+ :=$*

$d_1 + d_2 + d_3$ *for convenience. Then for any fixed* $k \in \{1, 2, \cdots, n_2\}$ *and for all* $\tau > 0$,

$$\mathbb{P}\left\{\left\|\mathcal{P}_X - \frac{d_1 d_2 d_3}{n_1}\mathcal{P}_X\mathcal{P}_{\Omega_j}\mathcal{P}_X\right\| \geq \tau\right\} \leq 2r^2 d_+ \ \exp\left(-\frac{n_1(\tau^2/2)}{(1+2\tau/3)(\mu_0^2 r^2 d_+)}\right). \tag{2.39}$$

Also,

$$\max_{\|X\|_{\max}=1} \mathbb{P}\left\{\left\|\left(\mathcal{P}_X - \frac{d_1 d_2 d_3}{n_1}\mathcal{P}_X\mathcal{P}_{\Omega_j}\mathcal{P}_X\right)X\right\|_{\max} \geq \tau\right\} \leq 2d_1 d_2 d_3 \exp\left(-\frac{n_1(\tau^2/2)}{(1+2\tau/3)\mu_0^2 r^2 d_+}\right) \tag{2.40}$$

Lemma 5. ***(Lemma 7 in [75]):*** *Let* $\alpha(X) \leq \alpha_0$, $r := \overline{r}(X)$, $d_+ = d_1 + d_2 + d_3$, *and* $q_1^* = (\beta + \log(d_+))^2 \alpha_0^2 r \log(d_+)$. *There exists a positive constant* c_1 *so that for any constants* $\beta > 0$ *and* $\delta_1 \in [1/(\log(d_+)), 1)$,

$$n_1 \geq c_1\left[q_1^* d_+^{1+\delta_1} + \sqrt{q_1^*(1+\beta)\delta_1^{-1} d_1 d_2 d_3}\right] \tag{2.41}$$

implies

$$\max_{\substack{X: X = \mathcal{P}_X X \\ \|X\|_{\max} \leq \|W\|_{\max}}} \mathbb{P}\left(\left\|(\mathcal{P}_X - \frac{d_1 d_2 d_3}{n_1}\mathcal{P}_X\mathcal{P}_{\Omega_1}\mathcal{P}_X)X\right\| \geq \frac{1}{16}\right) \leq d_+^{-\beta-1}, \tag{2.42}$$

where W *satisfies* $W = \mathcal{P}_X^0 W$, $\|W\| = 1$, *and* $\langle X, W\rangle = \|X\|_*$.

Notice that Lemma 5 puts a lower bound on n_1, and because $n_1 n_2 \leq n$, this also puts an upper bound on n_2.

Instead of working directly with the sequence $\{Y_j\}$, it will be easier to work with the recursive sequence

$$Z_j := W - \mathcal{P}_X Y_j \tag{2.43}$$

instead. Notice that for a fixed j, this recursion satisfies

$$\begin{aligned} Z_j &= \left(\mathcal{P}_X(I - \frac{d_1 d_2 d_3}{n_1}\mathcal{P}_{\Omega_j})\mathcal{P}_X\right) Z_{j-1}, \\ &= \left(\mathcal{P}_X(I - \frac{d_1 d_2 d_3}{n_1}\mathcal{P}_{\Omega_j})\mathcal{P}_X\right)\left(\mathcal{P}_X(I - \frac{d_1 d_2 d_3}{n_1}\mathcal{P}_{\Omega_{j-1}})\mathcal{P}_X\right)\cdots\left(\mathcal{P}_X(I - \frac{d_1 d_2 d_3}{n_1}\mathcal{P}_{\Omega_0})\mathcal{P}_X\right) W. \end{aligned}$$

We see that $Y_{n_2} = \sum_j \frac{d_1 d_2 d_3}{n_1}\mathcal{P}_{\Omega_j} Z_{j-1}$ and $Z_j \in \text{range}(\mathcal{P}_X)$ for all j. This implies

$$W^L = \mathcal{P}_{X^\perp} Y_{n_2} = \mathcal{P}_{X^\perp}\sum_j (I - \frac{d_1 d_2 d_3}{n_1}\mathcal{P}_{\Omega_j})Z_j. \tag{2.44}$$

We would like to show that $\|W^L\| < \frac{1}{8}$. For convenience, let $\mathcal{R}_j = \mathcal{P}_X(I - \frac{d_1 d_2 d_3}{n_1}\mathcal{P}_{\Omega_j})\mathcal{P}_X$. Using an adaptation of the argument presented in [75], section 3.3,

$$\mathbb{P}\left(\|\mathcal{P}_{X^\perp} Y_{n_2}\| \geq \frac{1}{8}\right) \leq \mathbb{P}\left(\left\|\sum_j \mathcal{R}_j Z_{j-1}\right\| \geq \frac{1}{8}\right) \tag{2.45}$$

$$\leq \mathbb{P}\left(\|\mathcal{R}_1 W\| \geq \frac{1}{16}\right) + \mathbb{P}\left(\|Z_1\|_{\max} \geq \|W\|_{\max}/4\right) \tag{2.46}$$

$$+ \mathbb{P}\left(\left\|\sum_{j=2}^{n_2} \mathcal{R}_j Z_{j-1}\right\| \geq \frac{1}{16},\ \|Z_1\|_{\max} < \|W\|_{\max}/4\right) \tag{2.47}$$

$$\leq \mathbb{P}\left(\|\mathcal{R}_1 W\| \geq \frac{1}{16}\right) + \mathbb{P}\left(\|Z_1\|_{\max} \geq \|W\|_{\max}/4\right) \tag{2.48}$$

$$+ \mathbb{P}\left(\|\mathcal{R}_2 Z_1\| \geq \frac{1}{32},\ \|Z_1\|_{\max} < \|W\|_{\max}/4\right) \tag{2.49}$$

$$+ \mathbb{P}\left(\|Z_2\|_{\max} \geq \|W\|_{\max}/8, \|Z_1\|_{\max} < \|W\|/4\right) \tag{2.50}$$

$$+ \mathbb{P}\left(\left\|\sum_{j=3}^{n_2} \mathcal{R}_j Z_{j-1}\right\| \geq \frac{1}{32}, \|Z_2\|_{\max} < \|W\|_{\max}/8\right) \tag{2.51}$$

$$\leq \sum_{j=1}^{n_2-1} \mathbb{P}\left(\|\mathcal{P}_X \mathcal{R}_j \mathcal{P}_X Z_{j-1}\|_{\max} \geq \|W\|_{\max}/2^{j+1},\ \|Z_{j-1}\|_{\max} \leq \|W\|_{\max}/2^j\right) \tag{2.52}$$

$$+ \sum_{j=1}^{n_2} \mathbb{P}\left(\|\mathcal{R}_j Z_{j-1}\| \geq 2^{-3-j},\ \|Z_{j-1}\|_{\max} \leq \|W\|_{\max}/2^j\right) \tag{2.53}$$

Using the fact that $\mathcal{R}_j$ and Z_j are i.i.d, Lemma 4 with the maximizing $X = Z_{j-1}/\|Z_{j-1}\|_{\max}$ and $\tau = \frac{1}{8}$ gives the bound

$$\mathbb{P}\left(\|\mathcal{P}_{X^\perp} Y_{n_2}\| \geq \frac{1}{8}\right) \leq n_2 \max_{\substack{X: X = \mathcal{P}_X X \\ \|X\|_{\max} \leq 1}} \mathbb{P}\left(\|\mathcal{P}_X \mathcal{R}_1 \mathcal{P}_X X\|_{\max} > \frac{1}{4}\right) + \mathbb{P}\left(\|\mathcal{R}_1 X\| > \frac{1}{16\|W\|_{\max}}\right) \tag{2.54}$$

$$\leq 2n_2 d_1 d_2 d_3 \exp\left(-\frac{(3/112)n_1}{\mu_0^2 r^2 d}\right) + n_2 \max_{\substack{X: X = \mathcal{P}_X X \\ \|X\|_{\max} \leq \|W\|_{\max}}} \mathbb{P}\left(\|\mathcal{R}_1 X\| > \frac{1}{16}\right). \tag{2.55}$$

Applying Lemma 5, to bound the rightmost term with probability no more than $(d_1 + d_2 + d_3)^{-1-\beta}$ for some positive constant β, we are finished.

2.5.2 Proof of (b)

We would like to prove that $\|\mathcal{P}_\Omega(W + W^L)\|_F < \frac{\lambda}{8}$. By the definition of the operator norm (2.5), Lemma 4 implies

$$\left\| Z - \frac{d_1 d_2 d_3}{n_1} \mathcal{P}_X \mathcal{P}_{\Omega_j} Z \right\|_F \leq \tau \|Z\|_F \tag{2.56}$$

and

$$\left\| Z - \frac{d_1 d_2 d_3}{n_1} \mathcal{P}_X \mathcal{P}_{\Omega_j} Z \right\|_{\max} \leq \tau \|Z\|_{\max} \tag{2.57}$$

with high probability for all $Z \in \text{range}(\mathcal{P}_X)$. Consequentially, using the independence of Ω_j and Z_{j-1}, we have

$$\|Z_j\|_F \leq \tau \|Z_{j-1}\|_F \implies \|Z_{n_2}\|_F \leq \tau^{n_2} \|W\|_F \tag{2.58}$$

and, using our bound on the coherence of X,

$$\|W\|_{\max} \leq \alpha_0 \left(\frac{\overline{r}}{d_1 d_2 d_3} \right)^{1/2} \tag{2.59}$$

and

$$\|Z_{n_2}\|_F \leq \tau^{n_2} \alpha_0 \sqrt{\overline{r}}. \tag{2.60}$$

Hence,

$$\|\mathcal{P}_\Omega(W + W^L)\|_F = \|\mathcal{P}_\Omega(W + (I - \mathcal{P}_X) Y_{n_2})\|_F \tag{2.61}$$

$$= \|\mathcal{P}_\Omega Z_{n_2}\|_F \tag{2.62}$$

$$\leq \tau^{n_2} \alpha_0 \sqrt{\overline{r}}, \tag{2.63}$$

where we used the fact that $\mathcal{P}_\Omega Y_{n_2} = 0$. Choosing $\tau = \mathcal{O}\left(\frac{r d_+}{n_1}\right)^{\frac{1}{2}}$ and $n_2 = \log(d_+)$, we are assured that our bound on $\overline{r}$ allows the quantity above to be less than $\frac{\lambda}{8}$ as long as ρ_r and ρ_s are sufficiently small. In fact, for the sequel, we will actually require that $\|\mathcal{P}_\Omega(W + W^L)\|_F < \frac{\lambda}{16}$, and it is clear that this bound also holds with high probability.

2.5.3 Proof of (c)

We would like to prove that $\|\mathcal{P}_{\Omega^\perp}(W + W^L)\|_{\max} < \frac{\lambda}{8}$. We have that $W + W^L = Y_{n_2} + Z_{n_2}$, so $\|\mathcal{P}_{\Omega^\perp}(W + W^L)\|_{\max} \leq \|Y_{n_2}\|_{\max} + \|Z_{n_2}\|_{\max}$. From the previous section we have already the

bound $\|Z_{n_2}\|_{\max} \leq \|Z_{n_2}\|_F \leq \frac{\lambda}{16}$, so we must only bound $\|Y_{n_2}\|_{\max}$.

$$\|Y_{n_2}\|_{\max} = \left(\frac{d_1 d_2 d_3}{n_1}\right) \left\| \sum_j \mathcal{P}_{\Omega_j} Z_{j-1} \right\|_{\max} \tag{2.64}$$

$$\leq \left(\frac{d_1 d_2 d_3}{n_1}\right) \sum_j \left\| \mathcal{P}_{\Omega_j} Z_{j-1} \right\|_{\max} \tag{2.65}$$

$$\leq \left(\frac{d_1 d_2 d_3}{n_1}\right) \sum_j \|Z_{j-1}\|_{\max} \tag{2.66}$$

$$\leq \left(\frac{d_1 d_2 d_3}{n_1}\right) \sum_j \tau^j \|W\|_{\max} \tag{2.67}$$

$$\leq \left(\frac{d_1 d_2 d_3}{n_1}\right) \sum_j \tau^j \alpha_0 \left(\frac{\overline{r}}{d_1 d_2 d_3}\right)^{1/2} \tag{2.68}$$

$$\leq \left(\frac{\alpha_0 \sqrt{\overline{r} d_1 d_2 d_3}}{n_1}\right) \sum_j \tau^j \tag{2.69}$$

$$= \left(\frac{\alpha_0 \sqrt{\overline{r} d_1 d_2 d_3}}{n_1}\right) \left(\frac{\tau(1-\tau^{n_2})}{1-\tau}\right) \tag{2.70}$$

Because we have used Lemma 5, we must choose $n_1 \geq \mathcal{O}(\sqrt{d_1 d_2 d_3})$. We will use $n_1 = \mathcal{O}\left(\frac{n}{\log(d_+)}\right)$. With

$$\overline{r} \leq \mathcal{O}\left(\frac{n}{(d_+) W(n) \alpha_0^2 \mu_0^2}\right)^{1/2}, \quad \tau = \mathcal{O}\left(\frac{\mu^2 r^2 d_+}{n_1}\right)^{\frac{1}{2}}$$

we have that $\|Y_{n_2}\| \leq \frac{\lambda}{16}$ when ρ_r and ρ_s are small enough, so the desired result holds.

2.6 Constructing W^S

Following the construction of the dual certificate for matrix RPCA [17], we will let

$$W^S = \lambda \mathcal{P}_{X^\perp} (\mathcal{P}_\Omega - \mathcal{P}_\Omega \mathcal{P}_X \mathcal{P}_\Omega)^{-1} \text{sgn}(S). \tag{2.71}$$

Notice that $\mathcal{P}_\Omega W^S = \lambda \mathcal{P}_\Omega (I - \mathcal{P}_X)(\mathcal{P}_\Omega - \mathcal{P}_\Omega \mathcal{P}_X \mathcal{P}_\Omega)^{-1} \text{sgn}(S) = \lambda \text{sgn}(S)$. As in the matrix case, W^S can be interpreted as the tensor with minimum Frobenius norm in the set $\{\mathcal{T} : \mathcal{T} \in \text{range}(\mathcal{P}_{X^\perp}),\ \mathcal{P}_\Omega W^S = \lambda \text{sgn}(S)\}$.

2.6.1 Proof of (d)

With $W^S = \lambda \mathcal{P}_{T^\perp}(\mathcal{P}_\Omega - \mathcal{P}_\Omega \mathcal{P}_T \mathcal{P}_\Omega)^{-1}\text{sgn}(S)$, we would like to show that $\|W^S\| < \frac{1}{8}$. Let $G = \text{sgn}(S)$ for convenience. The elements of G follow the distribution

$$G_{i,j} = \begin{cases} 1 & \text{with probability } \frac{\rho_S}{2}, \\ 0 & \text{with probability } 1 - \rho_S, \\ -1 & \text{with probability } \frac{\rho_S}{2}. \end{cases} \tag{2.72}$$

We can then write

$$\|W^S\| = \lambda \left\| \mathcal{P}_{X^\perp}(\mathcal{P}_\Omega - \mathcal{P}_\Omega \mathcal{P}_X \mathcal{P}_\Omega)^{-1} G \right\| \tag{2.73}$$

$$= \lambda \left\| \mathcal{P}_{X^\perp} \sum_{k=0}^{\infty} (\mathcal{P}_\Omega \mathcal{P}_X \mathcal{P}_\Omega)^k G \right\| \tag{2.74}$$

$$\leq \lambda \left\| \mathcal{P}_{X^\perp} G \right\| + \lambda \left\| \sum_{k=1}^{\infty} (\mathcal{P}_\Omega \mathcal{P}_X \mathcal{P}_\Omega)^k G \right\|, \tag{2.75}$$

where we have used the Neumann series expansion of the operator $(\mathcal{P}_\Omega - \mathcal{P}_\Omega \mathcal{P}_X \mathcal{P}_\Omega)^{-1}$ (viewing the operator as a map from range$(\mathcal{P}_\Omega)$ to itself, so that $\mathcal{P}_\Omega$ is the identity map).

The first term can be bounded using existing tail bounds on the spectral norm of random tensors. The distribution of G is subgaussian (with parameter 1), and applying the result from [65] is sufficient for our purposes, although a much tighter bound is likely to exist.

Lemma 6. ***(Theorem 1, Tamoika):*** *The following holds with probability at least* $1 - \delta$*:*

$$\|G\| \leq \sqrt{8 \log(\rho_S)\,(d_1 + d_2 + d_3) \log(6/\log(3/2)) + \log(2/\delta)} \tag{2.76}$$

With $\lambda = (d_+)^{-\frac{1}{2}}$ and ρ_s sufficiently small, Lemma 6 bounds the first term of (2.75) with large probability. To bound the second term, we will use an ϵ-net covering argument and some results from [75]. Define the following set of "digitalized" vectors:

$$\mathcal{B}_{m_j,d_j} = \{0, \pm 1, \pm 2^{-1/2}, \cdots, \pm 2^{-m_j/2}\}^{d_j} \cap \{u \in \mathbb{R}^{d_j} : \|u\| \leq 1\}. \tag{2.77}$$

Let Q be the operator $\sum_{k=1}^{\infty} (\mathcal{P}_\Omega \mathcal{P}_X \mathcal{P}_\Omega)^k$. By Lemma 9 in [75],

$$\|Q(G)\| = \max_{u_j \in \mathbb{S}^{d_j}} \langle Q(G), u_1 \otimes u_2 \otimes u_3 \rangle \leq 8 \max_{u_j \in \mathcal{B}_{m_j,d_j}} \langle Q(G), u_1 \otimes u_2 \otimes u_3 \rangle, \tag{2.78}$$

where $m_j = \lceil \log_2(d_j) \rceil$. (A similar result holds for the matrix spectral norm, where the set corresponding to $B_{m,D}$ is an ϵ-net for the unit sphere.) Because Q is self-adjoint, this also implies

$$\|Q(G)\| = \max_{u_j \in \mathbb{S}^{d_j}} \langle Q(u_1 \otimes u_2 \otimes u_3), G \rangle \leq 8 \max_{u_j \in \mathcal{B}_{m_j,d_j}} \langle Q(u_1 \otimes u_2 \otimes u_3), G \rangle . \tag{2.79}$$

Let $X(u,v,w) := \langle Q(u_1 \otimes u_2 \otimes u_3), G \rangle$ for convenience. Because the signs of G are i.i.d. symmetric, we can apply Hoeffding's inequality, conditional on the event that the support of G is exactly Ω:

$$\mathbb{P}(|X(u_1,u_2,u_3)| > t \mid \Omega) \leq 2 \exp\left(-\frac{2t^2}{\|Q(u_1 \otimes u_2 \otimes u_3)\|_F^2} \right). \tag{2.80}$$

Because $\|Q(u_1 \otimes u_2 \otimes u_3)\|_F^2 \leq \|Q\|^2 \cdot \|(u_1 \otimes u_2 \otimes u_3)\|_F^2 = \|Q\|^2$,

$$\mathbb{P}(|X(u_1,u_2,u_3)| > t \mid \Omega) \leq 2 \exp\left(-\frac{2t^2}{\|Q\|^2} \right), \tag{2.81}$$

and

$$\mathbb{P}\left(\max_{u_j \in \mathcal{B}_{m_j,d_j}} |X(u_1,u_2,u_3)| > t \mid \Omega \right) \leq 2 \left(\prod_{j=1,2,3} |\mathcal{B}_{m_j,d_j}| \right) \exp\left(-\frac{2t^2}{\|Q\|^2} \right), \tag{2.82}$$

$$\mathbb{P}(\|Q(E)\| > t \mid \Omega) \leq \left(\prod_{j=1,2,3} |\mathcal{B}_{m_j,d_j}| \right) \exp\left(-\frac{t^2}{32\|Q\|^2} \right). \tag{2.83}$$

We can easily bound $\|Q\|$ conditional on the event that $\|\mathcal{P}_\Omega \mathcal{P}_X\| \leq \sigma$. (See section 2.6.3 for a proof that this event holds with high probability.) Recall that

$$\|Q\| = \left\| \sum_{k=1}^{\infty} (\mathcal{P}_\Omega \mathcal{P}_X \mathcal{P}_\Omega)^k \right\| = \left\| \sum_{k=1}^{\infty} ((\mathcal{P}_\Omega \mathcal{P}_X)(\mathcal{P}_\Omega \mathcal{P}_X)^*)^k \right\| \leq \sum_{k=1}^{\infty} \left\| ((\mathcal{P}_\Omega \mathcal{P}_X)(\mathcal{P}_\Omega \mathcal{P}_X)^*)^k \right\| \leq \frac{\sigma^2}{1-\sigma^2}. \tag{2.84}$$

We finally have the unconditional bound

$$\mathbb{P}(\lambda \|Q(E)\| > t) \leq 2 \left(\prod_{j=1,2,3} |\mathcal{B}_{m_j,d_j}| \right) \exp\left(-\frac{t^2(1-\sigma^2)^2}{32\sigma^4\lambda^2} \right) + \mathbb{P}(\|\mathcal{P}_\Omega \mathcal{P}_X\| > \sigma) \tag{2.85}$$

$$= 2 \left(e^{\frac{21 d_+}{4}} \right) \exp\left(-\frac{t^2(1-\sigma^2)^2}{32\sigma^4\lambda^2} \right) + \mathbb{P}(\|\mathcal{P}_\Omega \mathcal{P}_X\| > \sigma) \tag{2.86}$$

$$= 2 \exp\left(-\frac{t^2(1-\sigma^2)^2}{32\sigma^4\lambda^2} + \frac{21 d_+}{4} \right) + \mathbb{P}(\|\mathcal{P}_\Omega \mathcal{P}_X\| > \sigma). \tag{2.87}$$

Here, we used a bound on the cardinality of $\mathcal{B}_{m_j,d_j}$ given in equation (21) of [75]. This shows that if λ is chosen to be on the order of $(d_+)^{-1/2}$, and if σ is small enough (order-unity), then $\|W^S\| < \frac{1}{8}$ with high probability.

Remark 1. *We would like to note that we can derive a similar bound using a more traditional covering argument (see, for example, [65]) instead of using the set of "digitalized" vectors* $\mathcal{B}_{m_j,d_j}$.

2.6.2 Proof of (e)

We would like to show that $\|\mathcal{P}_{\Omega^\perp} W^S\|_{\max} < \frac{\lambda}{8}$. To do so, we will use an adaptation of Lemma 5, the proof of which will parallel the argument outlined in section 5 of [75] to a large extent.

Define the set

$$\mathcal{U}(\eta) := \{X \in \mathbb{R}^{d_1 \times d_2 \times d_3} : X \in \text{range}(\mathcal{P}_{X^\perp}),\ \|X\|_{\max} \leq \eta/\sqrt{d_1 d_2 d_3}\}. \tag{2.88}$$

We will eventually take $\eta = \lambda\left(\frac{\sqrt{1+4m}-1}{2m}\right)$ for reasons that will become apparent later. We are interested in bounding

$$\max_{X \in \mathcal{U}(\eta)} \|\mathcal{P}_{\Omega^\perp} X\|_{\max} = \max_{X \in \mathcal{U}(\eta)} \|(\mathcal{P}_\Omega - I)X\|_{\max}. \tag{2.89}$$

To do so, we will first determine the set $\mathcal{U}(\eta)$ by bounding $\|X\|_{\max}$. Notice that

$$W^S = -\lambda \mathcal{P}_{X^\perp}(\mathcal{P}_\Omega - \mathcal{P}_\Omega \mathcal{P}_X \mathcal{P}_\Omega)^{-1} G \tag{2.90}$$

$$= \lambda (I - \mathcal{P}_X)(\mathcal{P}_\Omega - \mathcal{P}_\Omega \mathcal{P}_X \mathcal{P}_\Omega)^{-1} G \tag{2.91}$$

$$= -\lambda \mathcal{P}_X (\mathcal{P}_\Omega - \mathcal{P}_\Omega \mathcal{P}_X \mathcal{P}_\Omega)^{-1} G \tag{2.92}$$

Choosing $(i, j, k) \in [d_1] \times [d_2] \times [d_3]$, we have

$$W^S_{i,j,k} = \left\langle W^S, e_i \otimes e_j \otimes e_k \right\rangle \tag{2.93}$$

$$= \lambda \left\langle (I - \mathcal{P}_X)(\mathcal{P}_\Omega - \mathcal{P}_\Omega \mathcal{P}_T \mathcal{P}_\Omega)^{-1} G, e_i \otimes e_j \otimes e_k \right\rangle \tag{2.94}$$

$$= \lambda \left\langle (I - \mathcal{P}_X)(\mathcal{P}_\Omega - \mathcal{P}_\Omega \mathcal{P}_X \mathcal{P}_\Omega)^{-1} \mathcal{P}_\Omega G, e_i \otimes e_j \otimes e_k \right\rangle \tag{2.95}$$

$$= -\lambda \left\langle (\mathcal{P}_\Omega - \mathcal{P}_\Omega \mathcal{P}_X \mathcal{P}_\Omega)^{-1} \mathcal{P}_\Omega \mathcal{P}_X (e_i \otimes e_j \otimes e_k), G \right\rangle, \tag{2.96}$$

where we used the fact that $\mathcal{P}_\Omega$, $\mathcal{P}_X$, and $(\mathcal{P}_\Omega - \mathcal{P}_\Omega \mathcal{P}_X \mathcal{P}_\Omega)^{-1}$ are all self-adjoint. For convenience, let

$$R(i, j, k) = (\mathcal{P}_\Omega - \mathcal{P}_\Omega \mathcal{P}_X \mathcal{P}_\Omega)^{-1} \mathcal{P}_\Omega \mathcal{P}_X (e_i \otimes e_j \otimes e_k). \tag{2.97}$$

Now we will bound the maximum entry of W^S with high probability conditional on the event that the support of G (denoted supp(G)) is exactly Ω, and that $\|\mathcal{P}_\Omega \mathcal{P}_X\| \leq \sigma$ (again, we refer the reader to section 2.6.3 for a proof that this holds with high probability). Because the entries of G are i.i.d. symmetric, Hoeffding's inequality gives

$$\mathbb{P}\left(\left|W^S_{i,j,k}\right| \geq \eta \mid \Omega\right) \leq 2\exp\left(-\frac{2\eta^2}{\lambda^2 \|R(i,j,k)\|_F^2}\right), \tag{2.98}$$

so

$$\mathbb{P}\left(\max_{i,j,k}\left|W^S_{i,j,k}\right| \geq \eta \mid \Omega\right) \leq 2d_1d_2d_3\exp\left(-\frac{2\eta^2}{\lambda^2 \max_{i,j,k}\|R(i,j,k)\|_F^2}\right). \tag{2.99}$$

Now we will bound the operator norm of $(\mathcal{P}_\Omega - \mathcal{P}_\Omega\mathcal{P}_X\mathcal{P}_\Omega)^{-1}\mathcal{P}_\Omega\mathcal{P}_X$. First, by Lemma 2,

$$\|\mathcal{P}_T(e_i \otimes e_j \otimes e_k)\|_F \leq \mu_0 \overline{r}\left(\frac{d_1+d_2+d_3}{d_1d_2d_3}\right)^{1/2}. \tag{2.100}$$

Hence,

$$\|\mathcal{P}_\Omega\mathcal{P}_T(e_i \otimes e_j \otimes e_k)\|_F = \|\mathcal{P}_\Omega\mathcal{P}_X\|\|\mathcal{P}_X(e_i \otimes e_j \otimes e_k)\|_F \tag{2.101}$$

$$\leq \sigma\mu_0\overline{r}\left(\frac{d_1+d_2+d_3}{d_1d_2d_3}\right)^{1/2} \tag{2.102}$$

Furthermore, using the Neumann series representation of $(\mathcal{P}_\Omega - \mathcal{P}_\Omega\mathcal{P}_X\mathcal{P}_\Omega)^{-1}$, we have

$$\|(\mathcal{P}_\Omega - \mathcal{P}_\Omega\mathcal{P}_X\mathcal{P}_\Omega)^{-1}\| = \|(\mathcal{P}_\Omega - (\mathcal{P}_\Omega\mathcal{P}_X)(\mathcal{P}_\Omega\mathcal{P}_X)^*)^{-1}\| \tag{2.103}$$

$$\leq (1-\sigma^2)^{-1}, \tag{2.104}$$

so

$$\|R(i,j,k)\|_F^2 = \|(\mathcal{P}_\Omega - \mathcal{P}_\Omega\mathcal{P}_X\mathcal{P}_\Omega)^{-1}\mathcal{P}_\Omega\mathcal{P}_X(e_i \otimes e_j \otimes e_k)\|_F^2 \tag{2.105}$$

$$\leq \left(\frac{\sigma\mu_0\overline{r}}{1-\sigma^2}\right)^2\left(\frac{d_1+d_2+d_3}{d_1d_2d_3}\right). \tag{2.106}$$

Finally, we have derived that

$$\mathbb{P}\left(\max_{i,j,k}\left|W^S_{i,j,k}\right| \geq \eta\right) \leq 2d_1d_2d_3\exp\left(-\frac{2\eta^2(1-\sigma^2)^2d_1d_2d_3}{(\sigma\lambda\mu_0\overline{r})^2(d_1+d_2+d_3)}\right) + \mathbb{P}\left(\|\mathcal{P}_\Omega\mathcal{P}_X\| > \sigma\right). \tag{2.107}$$

With this bound in place, we can now develop a bound on $\max_{X\in\mathcal{U}(\eta)}\|(\mathcal{P}_\Omega - I)X\|_{\max}$ by adapting some results from [75]. The following lemma simplifies our problem using a standard symmetrization argument.

Lemma 7. ***(Adapted from Lemma 8 from [75]):*** *Let ϵ_i be a Rademacher sequence. Then, for $\Omega = \{(a_i, b_i, c_i) \in [d_1] \times [d_2] \times [d_3] : i = 1, 2, \cdots, m\}$,*

$$\max_{X \in \mathcal{U}(\eta)} \mathbb{P}\left\{\|\mathcal{P}_\Omega X - X\| \geq t\right\} \tag{2.108}$$

$$\leq \max_{X \in \mathcal{U}(\eta)} \mathbb{P}\left\{\left\|\sum_{i=1}^m \epsilon_i \mathcal{P}_{(a_i,b_i,c_i)} X\right\| \geq \frac{t}{2}\right\} + 4\exp\left(-\frac{t^2/8}{m\eta^2 + 2\eta t/3}\right) \tag{2.109}$$

Proof. Using symmetrization,

$$\max_{X \in \mathcal{U}(\eta)} \mathbb{P}\left\{\|\mathcal{P}_\Omega X - X\| \geq t\right\} \tag{2.110}$$

$$\leq 4 \max_{X \in \mathcal{U}(\eta)} \mathbb{P}\left\{\left\|\sum_{i=1}^m \epsilon_i \mathcal{P}_{(a_i,b_i,c_i)} X\right\| \geq \frac{t}{2}\right\} \tag{2.111}$$

$$2 \max_{X \in \mathcal{U}(\eta)} \max_{\|u\|=\|v\|=\|w\|=1} \mathbb{P}\left\{\left\langle u \otimes v \otimes w, \sum_{i=1}^m \mathcal{P}_{(a_i,b_i,c_i)} X - X \right\rangle > \frac{t}{2}\right\}. \tag{2.112}$$

To bound the rightmost quantity, let

$$\xi_i = \left\langle u \otimes v \otimes w, \mathcal{P}_{(a_i,b_i,c_i)} X - X \right\rangle, \quad \|u\| = \|v\| = \|w\| = 1, \quad X \in \mathcal{U}(\eta). \tag{2.113}$$

It is clear that the ξ_i are i.i.d. random variables satisfying

$$\mathbb{E}[\xi_i] = 0, \quad |\xi_i| \leq 2\sqrt{d_1 d_2 d_3}\|X\|_{\max} \leq 2\eta, \quad \mathbb{E}[\xi_i^2] \leq (d_1 d_2 d_3)\|X\|_{\max}^2 \leq \eta^2, \tag{2.114}$$

where we used the fact that $\|X\| \leq \sqrt{d_1 d_2 d_3}\|X\|_{\max}$. The desired result is then an application of Bernstein's inequality. □

The next lemma we take directly from [75].

Lemma 8. ***(Lemma 13 in [75]):*** *For some constant β and $\delta \in [1/\log(d_+), 1]$, define the following:*

$$p^* = \max(d_1, d_2, d_3)/(d_1 d_2 d_3), \qquad \nu_1 = (e d_+^{\delta_1} m p^*) \wedge (3 + \beta)/\delta_1$$

$$J(a, b, c) := (b + 2)\sqrt{a 2^{c-1}} L(\sqrt{a 2^{c-1}}, (b + 2)d_+), \qquad L_0 = 1 \vee \log(e d_+ (m_* + 2)/\sqrt{\nu_1 2^{-1}},$$

$$m_* = \min\{x : x \geq m^* \text{ or } J(\nu_1, x, x) \geq d_+\}, \qquad L(a', b') := \max\{1, \log(e b'/a')\}.$$

Choose x and t_1 to satisfy

$$x \geq 1, \quad mt_1 \geq 24\eta(m_* + 1)^{\frac{3}{2}}\sqrt{\nu_1 d_1 d_2 d_3}, \quad mxt_1^2(2\log(2) - 1) \geq 12\eta^2(m_* + 2)^2\sqrt{e}(d_1 + d_2 + d_3)L_0.$$

Then,

$$\max_{X \in \mathcal{U}(\eta)} \mathbb{P}\left(\left\|\sum_{i=1}^{m} \epsilon_i \mathcal{P}_{(a_i,b_i,c_i)} X\right\| \geq \frac{xt_1 m}{d_1 d_2 d_3}\right) \tag{2.115}$$

$$\leq \quad \left[\binom{m_* + 1}{2} + m^* - m_*\right][ed_+(m_* + 2)]^{(-6x-21/4)(m_*+2)} + d_+^{-\beta-1}/3. \tag{2.116}$$

The conditions on x and t_1 listed in Lemma 8 can be interpreted as upper bounds on η and m. With $xt_1 \leq \frac{\lambda d_1 d_2 d_3}{16m}$, and taking $x = 1$, the first inequality implies

$$m \geq 24\eta(m_* + 2)^{3/2}\left(\frac{16m\sqrt{\nu_1 d_1 d_2 d_3}}{\lambda d_1 d_2 d_3}\right) \tag{2.117}$$

$$\implies \eta \leq \frac{1}{384}(m_* + 2)^{-3/2}\left(\frac{\lambda\sqrt{d_1 d_2 d_3}}{\sqrt{\nu_1}}\right) \tag{2.118}$$

The second inequality implies

$$m \geq 12\eta^2(m_* + 2)^2\sqrt{e}d_+ L_0\left(\frac{16m}{\lambda d_1 d_2 d_3\sqrt{2\log(2) - 1}}\right)^2 \tag{2.119}$$

$$\implies m \leq \left(12(m_* + 2)^2\sqrt{e}d_+ L_0\right)^{-1}\left(\frac{\lambda d_1 d_2 d_3\sqrt{2\log(2) - 1}}{16\eta}\right)^2. \tag{2.120}$$

Lemma 9. *With ρ_r sufficiently small, m and η satisfying*

$$m \leq \frac{\sqrt{\rho_r \log(n) W(n)} + W(n)}{\rho_r \log(n)}, \quad \eta = \lambda\left(\frac{\sqrt{1 + 4m} - 1}{2m}\right),$$

and the other parameters defined as above,

$$\mathbb{P}\left\{\|\mathcal{P}_\Omega X - X\|_{\max} \geq \tfrac{\lambda}{8}\right\} \leq d_+^{-1-\beta}. \tag{2.121}$$

Proof. With m and η chosen in this way, the probability in (2.107) is sufficiently small. Also, as outlined in section 5.6 of [75], the choice of the parameters in Lemma 8 ensures that the probability in Lemma 8 is of order $d_+^{-\beta-1}$. □

In light of Lemma 9, we are assured that $\|(\mathcal{P}_\Omega X - X)\| = \|(\mathcal{P}_\Omega - I)X\| < \frac{\lambda}{8}$ with probability at most $d_+^{-1-\beta}$. Hence, we have proven the desired result, and completed our proof of Theorem 2.

2.6.3 Bounding the Operator Norm of $\mathcal{P}_\Omega \mathcal{P}_X$

The past two results were conditional on the event that $\|\mathcal{P}_\Omega \mathcal{P}_X\| \leq \sigma$ with high probability. Here, we will show that this assumption is justified.

Lemma 10. ***(Lemma 5 in [75])****: Let $\mu(X) \leq \mu_0$, $\overline{r}(X) = r$, and Ω with $|\Omega| = m$ be uniformly sampled from $[d_1] \times [d_2] \times [d_3]$ without replacement. Then, for any $\tau > 0$*

$$\mathbb{P}\left(\|\mathcal{P}_X(\tfrac{d_1 d_2 d_3}{m}\mathcal{P}_\Omega - I)\mathcal{P}_X\| \geq \sigma\right) \leq 2r^2(d_1+d_2+d_3)\exp\left(-\frac{m\sigma^2/2}{(1+2\sigma/3)\mu_0^2 r^2(d_1+d_2+d_3)}\right). \tag{2.122}$$

We have that $r = \mathcal{O}\left(\left(\frac{n}{d_+ W(n)\mu^2}\right)^{\frac{1}{2}}\right)$, so with $\sigma = \frac{1}{2}$, Lemma 10 shows that as an operator in the range of $\mathcal{P}_X$, $\frac{d_1 d_2 d_3}{m}\|\mathcal{P}_X \mathcal{P}_\Omega \mathcal{P}_X\| \in [1/2, 3/2]$. Using the fact that $\|\mathcal{P}_\Omega \mathcal{P}_X\|^2 = \|(\mathcal{P}_\Omega \mathcal{P}_X)^*(\mathcal{P}_\Omega \mathcal{P}_X)\| = \|\mathcal{P}_X \mathcal{P}_\Omega \mathcal{P}_X\|$, we have also bounded $\|\mathcal{P}_\Omega \mathcal{P}_X\|$ with high probability.

2.7 A Non-Convex Approach to Atomic Norm Minimization

Although we have proven that the program (2.13) can exactly recover a low Tucker-rank tensor and a sparse tensor from their superposition, (2.13) is NP-hard to solve in general due to the intractability of the atomic norm [39]. For low-rank matrix recovery using nuclear norm regularization, it is common to accelerate computation by replacing the nuclear norm with a non-convex surrogate based on a factorization of X. Specifically, it can be shown that [19, 27, 28, 64]

$$\|X\|_* = \inf_{UV^T = X} \quad \tfrac{1}{2}(\|U\|_F^2 + \|V\|_F^2). \tag{2.123}$$

We will derive a similar non-convex, factorized formulation of the tensor atomic norm that can be used to develop a tractable program that is equivalent to (2.13).

Our development builds on previous work on tensor completion using various extensions of the nuclear norm to a higher-order setting. In [10], the authors present a non-convex algorithm for tensor completion based on a higher-order generalization of the Schatten-2/3 norm. We will extend this idea to develop a non-convex program that is equivalent to (2.13) and offer several algorithms

that provably converge to a stationary point of the non-convex program. Of course, the results of Theorem 2 apply only to the global minimizer of (2.13), which is NP-hard to find. However, in section 2.8, we provide a certificate that bounds the suboptimality of a recovered solution to our non-convex program. In section 2.9, we demonstrate the superior performance of our non-convex program. Most importantly, we will see that our tensor RPCA can recover tensors whose low-rank component has a CP-rank that is larger than each of its side lengths. These experiments demonstrate that unfolding tensors for low-rank recovery leads to sub-optimal performance, and that the complexity of a high-dimensional dataset is better measured by its CP-rank rather than its Tucker-rank.

2.7.1 Burer-Monteiro Factorization in Higher-Orders

Contrary to the previous sections, we will now explicitly work with tensors of order-K, with $K \geq 3$. We will write the CP-decomposition of a rank-R tensor $X \in \mathbb{R}^{d_1 \times \cdots \times d_K}$ as

$$X = \sum_{r=1}^{R} \gamma_r (a_r^{(1)} \otimes \cdots \otimes a_r^{(K)}), \tag{2.124}$$

$$= [[A^{(1)}, \cdots, A^{(K)}]], \tag{2.125}$$

using Kruskal's notation with the factor matrices $A^{(1)}, \cdots, A^{(k)}, \cdots, A^{(K)} \in \mathbb{R}^{d_k \times R}$.

Instead of establishing equivalence to the program (2.13) directly, we will work with its Lagrangian formulation:

$$\min_{X,S} \quad \frac{1}{2}\|X + S - \mathcal{Z}\|_F^2 + \mu\|X\|_* + \lambda\|S\|_{\text{sum}}. \tag{2.126}$$

As in the matrix case [77], this formulation can be used to adjust the amount of noise we expect to observe in our data. When μ and λ are small, the solution of (2.126) is the same as the solution to (2.13), with $\|X + S - \mathcal{Z}\|_F = 0$. If we expect our observed data $\mathcal{Z}$ to contain noise, we can increase μ and λ to allow a disparity between $\mathcal{Z}$ and $X + S$.

Using a Burer-Monteiro factorization approach to develop a non-convex program equivalent

to (2.126), we will implicitly introduce a bound on the rank of X, yielding the constrained problem

$$\begin{aligned} \min_{X,S} \quad & \frac{1}{2}\|X+S-\mathcal{Z}\|_F^2 + \mu\|X\|_* + \lambda\|S\|_{\text{sum}}. \\ \text{s.t.} \quad & \text{rank}_{\text{CP}}(X) \leq R. \end{aligned} \tag{2.127}$$

Although (2.126) is non-convex, it has the same global optima as the convex program (2.13) as long as the rank-bound R is non-restrictive at the solution [27, 69]. The derivation of our non-convex model begins with the following proposition:

Proposition 1. *Suppose $\gamma^*, \{u_r^{*(1)}\}, \cdots, \{u_r^{*(K)}\}, S^*$ are optimal for the non-convex program*

$$\min_{\gamma,\{u_r^{(1)}\},\cdots,\{u_r^{(K)}\}} \quad \frac{1}{2}\left\|\sum_{r=1}^{R}\gamma_r(u_r^{(1)}\otimes\cdots\otimes u_r^{(K)}) + S - \mathcal{Z}\right\|_F^2 + \lambda\|S\|_{\text{sum}} + \frac{\mu}{K}\|\gamma\|_1, \tag{2.128}$$

$$\text{s.t.} \quad \|u_r^{(1)}\|, \cdots, \|u_r^{(K)}\| \leq 1, \tag{2.129}$$

Let $X^* = \sum_{r=1}^{R}\gamma_r^*(u_r^*\otimes v_r^*\otimes w_r^*)$. Then the point (X^*, S^*) is optimal for the problem (2.126). Conversely, if (X^*, S^*) is the minimizer of (2.126), then the terms $\gamma^*, \{u_r^*\}, \{v_r^*\}, \{w_r^*\}$ from a CP-decomposition of X^* are optimal for (2.128).

Proof. Using the definition of the atomic norm, we can rewrite (2.13) as

$$\min_{X} \quad \frac{1}{2}\|X+S-\mathcal{Z}\|_F^2 + \lambda\|S\|_{\text{sum}} + \min_{\gamma,\{u_r^{(1)}\},\cdots,\{u_r^{(K)}\}} \mu\|\gamma\|_1, \tag{2.130}$$

$$\text{s.t.} \quad X = \sum_{r=1}^{R}\gamma_r(u_r^{(1)}\otimes\cdots\otimes u_r^{(K)}), \tag{2.131}$$

$$\|u_r^{(1)}\| = \cdots = \|u_r^{(K)}\| = 1. \tag{2.132}$$

Due to the coerciveness of norms, replacing the norm constraints with inequalities does not change the global optima. This adjustment yields (2.128). □

If R is chosen large enough, then the program (2.128) is a tractable, non-convex reformulation of (2.13). However, instead of replacing the atomic norm with a smooth, non-convex regularizer as we would in the matrix case, we have introduced a non-smooth term and multiple constraints. We would like a non-convex representation of the atomic norm that more closely generalizes (2.123). Proposition 2 provides this.

Proposition 2. *Consider the non-convex program*

$$\min_{\{a_r^{(1)}\},\cdots,\{a_r^{(K)}\}} \quad \frac{1}{2}\left\|\sum_{r=1}^{R}(a_r^{(1)}\otimes\cdots\otimes a_r^{(K)})+S-\mathcal{Z}\right\|_F^2+\lambda\|S\|_{\text{sum}}+\frac{\mu}{K}\sum_{r=1}^{R}\sum_{k=1}^{K}\|a_r^{(k)}\|^K. \tag{2.133}$$

Let (X^*, S^*) *be the solution to* (2.13)*. The program* (2.133) *and* (2.13) *are equivalent, in that they share the same set of global optima.*

Proof. We will use an argument similar to the proof in Appendix II of [10]. We can rewrite (2.133) as

$$\min_{\gamma,\{a_r^{(1)}\},\cdots,\{a_r^{(K)}\},\{u_r^{(1)}\},\cdots,\{u_r^{(K)}\}} \quad \frac{1}{2}\|X+S-\mathcal{Z}\|_F^2+\lambda\|S\|_{\text{sum}}+\frac{\mu}{K}\sum_{r=1}^{R}\sum_{k=1}^{K}\|a_r^{(k)}\|^K \tag{2.134}$$

$$\text{s.t.} \quad X=\sum_{r=1}^{R}\gamma_r(u_r^{(1)}\otimes\cdots\otimes u_r^{(K)}), \tag{2.135}$$

$$\gamma_r=\|a_r^{(1)}\|\cdots\|a_r^{(K)}\|. \tag{2.136}$$

Minimizing over $\gamma, \{a_r\}, \{b_r\}$, and $\{c_r\}$ first, we must solve

$$\min_{\gamma} \quad \sum_{r=1}^{R}\sum_{k=1}^{K}\|a_r^{(k)}\|^K \tag{2.137}$$

$$\text{s.t.} \quad \gamma_r=\|a_r^{(1)}\|\cdots\|a_r^{(K)}\|. \tag{2.138}$$

The AM-GM inequality tells us

$$(\|a_r^{(1)}\|^K\cdots\|a_r^{(K)}\|^K)^{\frac{1}{K}}\leq\tfrac{1}{K}(\|a_r^{(1)}\|^K+\cdots+\|a_r^{(K)}\|^K), \tag{2.139}$$

with equality when $\|a_r^{(1)}\|=\cdots=\|a_r^{(K)}\|=\gamma^{\frac{1}{K}}$, so the optimal γ satisfies

$$\|\gamma\|_1=\frac{1}{K}\sum_{r=1}^{R}\sum_{k=1}^{K}\|a_r^{(k)}\|^K. \tag{2.140}$$

Using these optimal values in (2.134), we see that (2.134) is equivalent to

$$\min_{\gamma,\{u_r^{(1)}\},\cdots,\{u_r^{(K)}\}} \quad \frac{1}{2}\left\|\sum_{r=1}^{R}\gamma_r(u_r^{(1)}\otimes\cdots\otimes u_r^{(K)})+S-\mathcal{Z}\right\|_F^2+\lambda\|S\|_{\text{sum}}+\frac{\mu}{K}\|\gamma\|_1, \tag{2.141}$$

$$\text{s.t.} \quad \|u_r^{(1)}\|,\cdots,\|u_r^{(K)}\|\leq 1, \tag{2.142}$$

which is equivalent to (2.13) by Proposition 1. □

The program (2.133) can be seen as a higher-order generalization of Burer-Monteiro factorization, which is popular in low-rank matrix recovery [19, 27, 28, 64, 69]. Notice for $K = 2$, the low-rank regularizing term of (2.133) reduces to $\frac{\mu}{2}(\|A\|_F^2 + \|B\|_F^2)$, which is equivalent to standard nuclear-norm regularization (2.123). We will compare a number of algorithms that solve (2.128) and (2.133).

Remark 2. *For matrices, this type of factorization carries with it a useful classification of the stationary points of the resulting non-convex program. We do not claim that these properties generalize to higher-orders, but this would be an interesting direction for future research.*

For low-rank matrix recovery, it is popular to use alternating minimization to solve the factorized problem (2.133) [38, 69]. There are many techniques, both heuristic and theoretically supported, to improve the rate of convergence and the quality of the solution for this well-studied approach. We will show that alternating minimization is guaranteed to converge to a stationary point of (2.133). We will also briefly analyze the performance of various implementations of FISTA and quasi-Newton methods with smoothing, as presented in [27]. In section 2.9, we show that even though the performance of quasi-Newton methods is not guaranteed in this setting, they have the strongest performance in application.

2.7.2 Alternating Minimization

Alternating minimization is a workhorse for fitting low-rank matrix models due to its efficiency and easy of implementation [38, 69]. In this section, we will develop an alternating-minimization approach to solving (2.133), and prove that our algorithm converges to a stationary point.

2.7.2.1 Simplifying the Subproblems

We would like to solve the problem

$$\min_{\{a_r^{(1)}\},\cdots,\{a_r^{(K)}\}} \quad \frac{1}{2}\left\|\sum_{r=1}^{R}(a_r^{(1)} \otimes \cdots \otimes a_r^{(K)}) + S - \mathcal{Z}\right\|_F^2 + \lambda\|S\|_{\text{sum}} + \frac{\mu}{K}\sum_{r=1}^{R}\sum_{k=1}^{K}\|a_r^{(k)}\|^K, \tag{2.143}$$

by sequentially updating each vector $a_r^{(k)}$ and holding the other vectors fixed. In the matrix case (where $K = 2$), the low-rank regularizing term is $\frac{\mu}{2}(\|A\|_F^2 + \|B\|_F^2)$, and optimizing one factor matrix, say, A, with B fixed is a convex problem that can be solved for (at most) the cost of an SVD. For $K \geq 3$, there are several problems. Most importantly, the objective is columnwise seperable for each factor matrix, but it is not separable by row. This means that we must update each factor matrix columnwise, rather than all at once, contrary to the matrix case. Also, although the subproblem is convex, more theoretical work is required to show that we attain the optimum efficiently. Despite these problems, we show that each subproblem of our alternating minimization scheme for $K \geq 3$ recovers the unique, optimal factor matrix $A^{(k)}$ for roughly the cost of an SVD, just as in the matrix case. Furthermore, we prove that our alternating minimization algorithm converges to a stationary point of (2.143).

For the k^{th} alternating minimization step, we solve for a factor matrix $A^{(k)}$ columnwise, so we iteratively compute the optimal vectors $a_r^{(k)}$. We use matricization to isolate a single factor matrix. Notice that (2.128) is equivalent to

$$\min_{\{a_r^{(1)}\},\cdots,\{a_r^{(K)}\}} \quad \frac{1}{2}\left\| \left(\sum_{r=1}^{R} (a_r^{(1)} \otimes \cdots \otimes a_r^{(K)}) \right)_{(k)} + S_{(k)} - \mathcal{Z}_{(k)} \right\|_F^2 + \lambda \|S\|_{\text{sum}} + \frac{\mu}{K} \sum_{r=1}^{R} \sum_{k=1}^{K} \|a_r^{(k)}\|^K, \tag{2.144}$$

where the subscript $\mathcal{Z}_{(k)}$ denotes the unfolding of the tensor $\mathcal{Z}$ along the k^{th} mode. We can rewrite these unfoldings in a more illuminating form to obtain the following equivalent problem [1]:

$$\min_{\{a_r^{(1)}\},\cdots,\{a_r^{(K)}\}} \quad \frac{1}{2}\left\| \left(A^{(K)} \odot \cdots \odot A^{(k+1)} \odot A^{(k-1)} \odot \cdots \odot A^{(1)} \right) (A^{(k)})^T + S_{(k)}^T - \mathcal{Z}_{(k)}^T \right\|_F^2 + \lambda \|S\|_{\text{sum}} + \frac{\mu}{K} \sum_{r=1}^{R} \sum_{k=1}^{K} \|a_r^{(k)}\|^K. \tag{2.145}$$

where $\odot$ denotes the Khatri-Rao product (c.f. (2.2)). For convenience, define

$$C := \left(A^{(K)} \odot \cdots \odot A^{(k+1)} \odot A^{(k-1)} \odot \cdots \odot A^{(1)} \right). \tag{2.146}$$

To make the objective in (2.145) columnwise separable in $A^{(k)}$, we need to invert the transpose operator acting on it. To do this, we vectorize $(A^{(k)})^T$, $S_{(k)}^T$ and $\mathcal{Z}_{(k)}^T$ in the loss term. We also must

define the matrix

$$L := \left(\overbrace{\begin{array}{cccc} C & \mathbf{0}_{\prod_{i\neq k} d_i \times R} & \cdots & \mathbf{0}_{\prod_{i\neq k} d_i \times R} \\ \mathbf{0}_{\prod_{i\neq k} d_i \times R} & C & \cdots & \mathbf{0}_{\prod_{i\neq k} d_i \times R} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0}_{\prod_{i\neq k} d_i \times R} & \mathbf{0}_{\prod_{i\neq k} d_i \times R} & \cdots & C \end{array}}^{d_i-\text{blocks}} \right\} d_i - \text{blocks}, \tag{2.147}$$

where $\mathbf{0}_{m\times n}$ is an $m \times n$ matrix of zeros. This yields the equivalent problem

$$\min_{\{a_r^{(1)}\},\cdots,\{a_r^{(K)}\}} \quad \frac{1}{2}\left\|Ly_T + s_T - z_T\right\|_F^2 + \lambda\|S\|_{\text{sum}} + \frac{\mu}{K}\sum_{r=1}^{R}\sum_{k=1}^{K}\|a_r^{(k)}\|^K, \tag{2.148}$$

where $y_T = \text{vec}\left((A^{(k)})^T\right)$, and s_T, z_T are defined analogously. Finally, define the linear operator $\mathcal{T} : \mathbb{R}^{d_k \cdot R} \to \mathbb{R}^{d_k \cdot R} : \text{vec}(M) \mapsto \text{vec}(M^T)$ for all matrices $M \in \mathbb{R}^{d_k \times R}$. This operator has a simple structure:

$$\mathcal{T} = \begin{pmatrix} \overbrace{1\ 0\ \cdots\ 0}^{n} & & & & & \\ & 1\ 0\ \cdots\ 0 & & & & \\ & & \ddots & & & \\ & & & 1\ 0\ \cdots\ 0 & & \\ 0\ 1\ \cdots\ 0 & & & & & \\ & 0\ 1\ \cdots\ 0 & & & & \\ & & \ddots & & & \\ & & & 0\ 1\ \cdots\ 0 & & \\ \vdots & \vdots & & \vdots & & \\ 0\ \cdots\ 0\ 1 & & & & & \\ & 0\ \cdots\ 0\ 1 & & & & \\ & & \ddots & & & \\ & & & 0\ \cdots\ 0\ 1 & & \end{pmatrix}. \tag{2.149}$$

Our problem now has the form

$$\min_{\{a_r^{(1)}\},\cdots,\{a_r^{(K)}\}} \quad \frac{1}{2}\left\|(L\mathcal{T})y + s_T - z_T\right\|_F^2 + \lambda\|S\|_{\text{sum}} + \frac{\mu}{K}\sum_{r=1}^{R}\sum_{k=1}^{K}\|a_r^{(k)}\|^K, \tag{2.150}$$

where

$$y = \mathcal{T}(y_T) = \begin{pmatrix} a_1^{(k)} \\ \vdots \\ a_R^{(k)} \end{pmatrix}. \tag{2.151}$$

We would like to solve (2.150) for $a_r^{(k)}$, with all other variables fixed. Without loss of generality, suppose we are optimizing over $a_1^{(k)}$. Define

$$y' = \begin{pmatrix} \mathbf{0}_{d_1 \times 1} \\ a_2^{(k)} \\ \vdots \\ a_R^{(k)} \end{pmatrix}, \quad x' = \begin{pmatrix} a_1^{(k)} \\ \mathbf{0}_{d_2 \times 1} \\ \vdots \\ \mathbf{0}_{d_K \times 1} \end{pmatrix}, \tag{2.152}$$

so that $y = y' + x'$. We have now reduced our problem to the much simpler form

$$\min_{x'} \frac{1}{2}\|(L\mathcal{T})x' - b\|^2 + \frac{\mu}{K}\|x'\|^K, \tag{2.153}$$

where $b := z_T - s_T - (L\mathcal{T})y'$. To reduce the problem size, we can ignore the zero entries of x' and replace the operator $(L\mathcal{T})$ with the block of size $\left(\prod_{i=1}^K d_i\right) \times d_k$ that multiplies the non-zero entries of x'. Call this block A, and define $x := a_1^{(k)}$. We now have the smaller problem

$$\min_{x} \frac{1}{2}\|Ax - b\|^2 + \frac{\mu}{K}\|x\|^K. \tag{2.154}$$

Notice that this problem is convex because $z \mapsto z^K$ is increasing on $\mathbb{R}_+$, and composition with an increasing function preserves convexity. However, the fact that (2.154) is convex does not necessarily mean that it can be solved efficiently.

With $K = 3$, this is a cubic-regularized least-squares problem. Similar problems have been considered previously in, for example, [20, 21, 22, 56]. We will generalize results from [56] to show that the minimizer of (2.154) can be found as the solution of a one-dimensional equation.

2.7.2.2 Solving (2.154)

We will solve (2.154) using a dual program. Before formally presenting the dual, we will provide some motivating intuition. For the rest of this section, we will assume that the minimizer

of (2.154) is bounded away from zero. We will relax this requirement when we give a more formal treatment of the problem in section 2.7.2.3.

Let $f(x)$ be the objective of (2.154), and enforce that $\mu > 0$. The critical point can be found by solving

$$\nabla f(x) = A^T(Ax - b) + \mu\|x\|^{K-2}x = 0, \text{ i.e.}$$
$$\begin{cases} (A^TA + \mu_x I)x = A^Tb \\ \mu_x = \mu\|x\|^{K-2} \end{cases} \tag{2.155}$$

We examine the first equation in detail. Let $A = U\Sigma V^T$ be the SVD of A so that $A^TA = V\Sigma^2V^T$. Then

$$(A^TA + \mu_x I) = V(\Sigma^2 + \mu_x I)V^T. \tag{2.156}$$

Under the constraint that $(A^TA + \mu_x I)$ is invertible (which is always true if $\mu_x > 0$ because A^TA is PSD), we have

$$\begin{aligned} x = (A^TA + \mu_x)^{-1}A^Tb &= V(\Sigma^2 + \mu_x I)^{-1}V^TA^Tb, \\ &= V(\Sigma^2 + \mu_x I)^{-1}\widetilde{b}. \end{aligned} \tag{2.157}$$

where $\widetilde{b} = V^TA^Tb = \Sigma U^Tb$. We see now that the constraint that $A^TA+\mu_x$ is invertible is equivalent to the constraint $(\Sigma^2 + \mu_x I) \succ 0$. We now have that $\|x\| = \|V(\Sigma^2 + \mu_x I)^{-1}\widetilde{b}\| = \|(\Sigma^2 + \mu_x I)^{-1}\widetilde{b}\|$ (since V is orthogonal). Because the inverse is of a diagonal matrix, we can simplify this to

$$\|x\|^2 = \sum_{i=1}^{n}\left(\frac{\widetilde{b}_i}{\sigma_i^2 + \mu\|x\|^{K-2}}\right)^2, \tag{2.158}$$

where σ_i are the singular values of A. We also used the definition $\mu_x = \mu\|x\|^{K-2}$. If we let $\alpha = \|x\|$, then we see that we must satisfy the one-dimensional equation

$$\sum_{i=1}^{n}\left(\frac{\widetilde{b}_i}{\sigma_i^2 + \mu\alpha^{K-2}}\right)^2 - \alpha^2 = 0, \tag{2.159}$$

Thus the optimal x for (2.154) is given by (2.157) using $\mu_x = \mu\alpha^*$, where α^* is a solution of (2.159).

This analysis is sufficient when the optimal $x \neq 0$, but it does not explain why our problem can be transformed into a one-dimensional, constrained program. As we will prove in the next section, this one-dimensional program and (2.154) are actually dual problems, and this procedure will yield the solution to (2.154) even when it is not bounded away from zero.

2.7.2.3 Constructing the Dual and Proving Strong Duality

This section will use some arguments from [20], and especially Theorem 10 in [56]. Let us rewrite the objective of (2.154) as the following function:

$$g(x) := \frac{1}{2}\langle A^T A x, x\rangle - \langle A^T b, x\rangle + \frac{\mu}{K}\|x\|^K. \tag{2.160}$$

We will define $h : \mathbb{R} \to \mathbb{R}$ to be the function

$$h(\alpha) := -\frac{1}{2}\left\langle (A^T A + \tfrac{\mu\alpha^{K-2}}{2} I)^{-1} A^T b, A^T b\right\rangle + \frac{(2-K)\mu}{2K}\alpha^K, \tag{2.161}$$

and define a feasible set as

$$\mathcal{C} := \{\alpha \in \mathbb{R}_+ : A^T A + \tfrac{\mu\alpha^{K-2}}{2} I \succ 0\}. \tag{2.162}$$

The following theorem states that minimizing g and maximizing h are dual problems. Furthermore, strong duality holds, and the solution to the primal problem can be solved in closed-form given a non-trivial solution to the dual.

Theorem 4. *Let $\mu > 0$ and $K \geq 3$. The two programs*

$$(P) \quad \min_{x\in\mathbb{R}^n} g(x), \qquad \text{and} \qquad (D) \quad \sup_{\alpha\in\mathcal{C}} h(\alpha). \tag{2.163}$$

are dual problems, and strong duality holds. Furthermore, define the direction

$$x(\alpha) = (A^T A + \tfrac{\mu\alpha^{K-2}}{2} I)^{-1} A^T b. \tag{2.164}$$

For any $\alpha \in \mathcal{C}$, we have that

$$0 \leq g(x(\alpha)) - h(\alpha) = \left(\frac{-2\alpha^3(K-2) - \alpha K\|x(\alpha)\|^2 + 4\alpha^{3-K}\|x(\alpha)\|^K}{2K(\alpha^{K-1}(2-K) + (K-2)\|x(\alpha)\|^2)}\right) h'(\alpha). \tag{2.165}$$

Hence, if the solution to the dual problem (D) is non-trivial, then $x(\alpha)$ is the solution to the primal (P).

Proof. This argument uses ideas from the proof of Theorem 10 in [56]. Let g^* be the optimal value of (2.160).

$$g^* = \min_x \quad \frac{1}{2}\langle A^TAx, x\rangle - \langle A^Tb, x\rangle + \frac{\mu}{K}\|x\|^K \tag{2.166}$$

$$= \min_{\substack{x \\ \tau=\|x\|^2}} \quad \frac{1}{2}\langle A^TAx, x\rangle - \langle A^Tb, x\rangle + \frac{\mu}{K}(\tau^{K/2})_+ \tag{2.167}$$

$$= \min_{\substack{x \\ \tau\in\mathbb{R}}} \sup_{\alpha\in\mathbb{R}} \quad \frac{1}{2}\langle A^TAx, x\rangle - \langle A^Tb, x\rangle + \frac{\mu}{K}(\tau^{K/2})_+ + \frac{\mu}{2}\alpha^{K-2}\left(\|x\|^2 - \tau\right). \tag{2.168}$$

We see that this last equality holds because if $\tau \neq \|x\|^2$, then the objective can be made arbitrarily large by taking $\alpha \to \pm\infty$. Continuing with the saddle-point inequality,

$$\geq \sup_{\alpha\in\mathcal{C}} \min_{\substack{x \\ \tau\in\mathbb{R}}} \quad \frac{1}{2}\langle A^TAx, x\rangle - \langle A^Tb, x\rangle + \frac{\mu}{K}(\tau^{K/2})_+ + \frac{\mu}{2}\alpha^{K-2}\left(\|x\|^2 - \tau\right) \tag{2.169}$$

$$= h^*, \tag{2.170}$$

where h^* is the minimum of (2.161). The final step deserves more attention. Let ∇_τ denote the gradient of the objective with respect to τ, and let ∇_x be defined analogously. The inner-minimization problem can then be solved analytically:

$$\nabla_\tau : \frac{\mu}{2}\tau^{K/2-1} - \frac{\mu}{2}\alpha^{K-2} = 0 \implies \tau = \alpha^2, \tag{2.171}$$

$$\nabla_x : A^TAx - A^Tb + \frac{\mu}{2}\alpha x = 0 \implies x = (A^TA + \tfrac{\mu\alpha^{K-2}}{2}I)^{-1}A^Tb. \tag{2.172}$$

Hence, (2.169) reduces to

$$\sup_{\alpha\in\mathcal{C}} \quad -\frac{1}{2}\left\langle (A^TA + \tfrac{\mu\alpha^{K-2}}{2}I)^{-1}A^Tb, A^Tb\right\rangle + \frac{(2-K)\mu}{2K}\alpha^K, \tag{2.173}$$

$$= h^*. \tag{2.174}$$

Now we must show that equality holds. For an arbitrary α, denote the direction

$$x(\alpha) = (A^TA + \tfrac{\mu\alpha^{K-2}}{2})^{-1}A^Tb. \tag{2.175}$$

as in (2.164). Hence,

$$A^T b = -(A^T A + \tfrac{\mu\alpha^{K-2}}{2})x(\alpha). \tag{2.176}$$

We now have

$$\begin{aligned}
g(x(\alpha)) &= \langle A^T b, x(\alpha)\rangle + \tfrac{1}{2}\langle A^T A x(\alpha), x(\alpha)\rangle + \tfrac{\mu}{K}\|x(\alpha)\|^K && (2.177)\\
&= -\tfrac{1}{2}\langle A^T A x(\alpha), x(\alpha)\rangle - \tfrac{\mu\alpha^{K-2}}{2}\|x(\alpha)\|^2 + \tfrac{\mu}{K}\|x(\alpha)\|^K && (2.178)\\
&= -\frac{1}{2}\left\langle (A^T A + \frac{\mu\alpha^{K-2}}{2})x(\alpha), x(\alpha)\right\rangle - \tfrac{\mu\alpha^{K-2}}{4}\|x(\alpha)\|^2 + \tfrac{\mu}{K}\|x(\alpha)\|^K && (2.179)\\
&= -\frac{1}{2}\left\langle (A^T A + \frac{\mu\alpha^{K-2}}{2})x(\alpha), x(\alpha)\right\rangle - \tfrac{\mu\alpha^{K-2}}{4}\|x(\alpha)\|^2 + \tfrac{\mu}{K}\|x(\alpha)\|^K && (2.180)\\
&= h(\alpha) - \frac{(2-K)\mu\alpha^K}{2K} - \tfrac{\mu\alpha^{K-2}}{4}\|x(\alpha)\|^2 + \tfrac{\mu}{K}\|x(\alpha)\|^K && (2.181)\\
&= h(\alpha) + \left(\frac{-2\alpha^3(K-2) - \alpha K\|x(\alpha)\|^2 + 4\alpha^{3-K}\|x(\alpha)\|^K}{2K(\alpha^{K-1}(2-K) + (K-2)\|x(\alpha)\|^2)}\right) h'(\alpha), && (2.182)
\end{aligned}$$

where

$$h'(\alpha) = \tfrac{\mu\alpha^{K-3}(K-2)}{2}\left(\|x(\alpha)\|^2 - \alpha^2\right). \tag{2.183}$$

Therefore,

$$g(x(\alpha)) - h(\alpha) = \left(\frac{-2\alpha^3(K-2) - \alpha K\|x(\alpha)\|^2 + 4\alpha^{3-K}\|x(\alpha)\|^K}{2K(\alpha^{K-1}(2-K) + (K-2)\|x(\alpha)\|^2)}\right) h'(\alpha). \tag{2.184}$$

The derivative in (2.183) clearly shows that if $h'(\alpha) = 0$ for some $\alpha > 0$, then strong duality holds, and we have recovered a globally optimal solution. If the optimal α^* of the dual problem occurs at $\alpha^* = 0$, then strong duality holds as a consequence of the continuity of g [56]. □

Theorem 4 reduces (2.154) to finding a solution to

$$\alpha = \|(A^T A + \tfrac{\mu\alpha^{K-2}}{2})^{-1} A^T b\|, \quad \alpha \geq -\frac{2}{\mu}\sigma_n^2, \tag{2.185}$$

where σ_n is the smallest singular value of A. Problems of this form arise often in applications of trust region methods (see, for example, [56] or Chapter 7 of [24]) In fact, this problem is equivalent to finding an α that satisfies

$$\sum_{i=1}^{n}\left(\frac{\widetilde{b}_i}{\sigma_i^2 + \mu\alpha^{K-2}}\right)^2 - \alpha^2 = 0, \quad \alpha \geq -\frac{\sigma_n^2}{\mu} \tag{2.186}$$

giving more theoretical support for (2.159).

As explained in [56], the solution to this problem is undefined when $\widetilde{b}_n = 0$, because this would imply that $\alpha = -\frac{2\sigma_n^2}{\mu} = 0$. In this case, the continuity of $g(x(\alpha);\alpha)$ allows us to approximate the optimal α arbitrarily well, and we refer the reader to [56], and Chapter 7 of [24] for more information on this case.

Suppose $\widetilde{b}_i \neq 0$ for all $i < k$, and $\widetilde{b}_i = 0$ for $i \geq k$. Once the optimal α^* is found, the solution to (2.154) is then

$$x^* = \sum_{i=1}^{k-1} \frac{\widetilde{b}_i v_i}{\sigma_i^2 + \mu\alpha^*} + cv_n, \tag{2.187}$$

where v_i is the i^{th} eigenvector of A^TA (corresponding to the eigenvalue σ_i^2), and the constant c is chosen so that $\|x^*\| = \alpha^*$. Because the subproblem (2.154) is convex, the optimal solution we find for the subproblem is unique.

We can now prove that our alternating minimization scheme converges to a stationary point of (2.133) using the following result:

Proposition 3. ***Proposition 2.7.1 from [12]*** *Consider a block coordinate descent approach to solving* $\min_x \ f(x)$*, where* f *is not necessarily convex. If the minimizer of each block is unique, then the algorithm converges to a stationary point of* f*.*

An application of Proposition 3 to our problem proves our claim. Despite these performance guarantees, we have found that the first-order methods developed in section 2.7.3 significantly outperform alternating minimization in practice.

2.7.3 First-Order Methods

One of the drawbacks of (2.133) is that the non-smooth ℓ_1-regularizer prevents the use of gradient-based solvers. However, as suggested in [27], the structure of the program allows us to smooth the problem through marginalization. Define

$$\varphi : X \mapsto \min_S \frac{1}{2}\|X + S - \mathcal{Z}\|_F^2 + \lambda\|S\|_{\text{sum}}. \tag{2.188}$$

Because we are using the least-squares loss and ℓ_1-regularization, φ is the (shifted) Moreau envelope of the ℓ_1-norm, also known as the Huber loss [6, 27]. The objective in (2.188) is strongly convex, so the minimum exists and is unique. In fact, it can be written in closed-form using the shrinkage operator:

$$(\text{shrink}(Y, \lambda))_{i_1,\cdots,i_K} := \text{sign}(Y_{i_1,\cdots,i_K})(|Y_{i_1,\cdots,i_K}| - \lambda|)_+,$$

(where $(a)_+$ denotes the non-negative part of a), so that

$$\arg\min_S \frac{1}{2}\|X + S - \mathcal{Z}\|_F^2 + \lambda\|S\|_{\text{sum}} = \text{shrink}(S, \lambda). \tag{2.189}$$

Incorporating φ into the convex program (2.13) preserves convexity and introduces differentiability. Combining φ with the non-convex program (2.133) yields a tractable, Lipschitz-differentiable problem that is susceptible to first-order solvers. Formally, consider the program

$$\min_{\{a_r^{(1)}\},\cdots,\{a_r^{(K)}\}} \quad \frac{\mu}{K}\sum_{r=1}^{R}\sum_{k=1}^{K}\|a_r^{(k)}\|^K + \varphi\left(a_1^{(1)},\cdots,a_R^{(1)},a_1^{(2)},\cdots,a_R^{(K)}\right), \tag{2.190}$$

where

$$\varphi\left(a_1^{(1)},\cdots,a_R^{(K)}\right) = \min_S \frac{1}{2}\left\|\left(\sum_{r=1}^{R}(a_r^{(1)}\otimes\cdots\otimes a_r^{(K)})\right) + S - \mathcal{Z}\right\|_F^2 + \lambda\|S\|_{\text{sum}}. \tag{2.191}$$

Let $f(a_1^{(1)},\cdots,a_R^{(K)})$ be the objective in (2.190), and let $S^\star$ be the minimizer in (2.191). Then f is differentiable with gradient given by

$$\nabla_{a_r^{(k)}} f = (\mu\|a_r^{(k)}\|^{K-2})a_r^{(k)} + \nabla_{a_r^{(k)}}\varphi\left(a_1^{(1)},\cdots,a_R^{(K)}\right)\Big|_{S^\star} \tag{2.192}$$

$$= (\mu\|a_r^{(k)}\|^{K-2})a_r^{(k)} + \left(a_r^{(k)}C^T + S^\star - \mathcal{Z}\right)C, \tag{2.193}$$

where $C = \left(A^{(K)} \odot \cdots \odot A^{(k+1)} \odot A^{(k-1)} \odot \cdots \odot A^{(1)}\right)$. We direct the reader to [60], Theorem 10.58, for proof of the first equality and to [1] for a derivation of the second. The smoothed problem (2.190) is not only easier to solve, it also has theoretical properties that the non-smoothed problem does not, making (2.190) easier to analyze. In section 2.8, we will use (2.190) to develop a certificate of optimality for (2.133).

For minimizing smooth objectives, quasi-Newton methods are often faster than algorithms that require only first-order smoothness, such as gradient-descent. The provable convergence rates of quasi-Newton methods rely on second-order smoothness, which does not hold for (2.190) because, for example, the Huber loss is in $\mathcal{C}^1$ and not in $\mathcal{C}^2$. However, numerical experiments (including those presented in section 2.9) suggest that quasi-Newton methods perform well even when only first-order smoothness is guaranteed. In section 2.9, we will see that L-BFGS solves (2.190) particularly quickly and reliably.

2.7.4 FISTA Solvers

There are many parallels between the procedure presented in the previous section and iterative thresholding algorithms. In this section, we explore several ways that FISTA can be applied to solve (2.133). FISTA was introduced in [11] as an accelerated solver for optimization problems of the form

$$\min_x \quad f(x) + g(x), \tag{2.194}$$

where f is first-order differentiable with an L-Lipschitz continuous gradient, and g is possibly non-smooth. The FISTA algorithm is often used in machine learning and compressed sensing to solve ℓ_1-regularized problems, and our problem (2.143) is of this form.

It is often the case with FISTA-like methods that some terms in the objective can be grouped into either the differentiable f or the non-smooth g, and different partitions can lead to different convergence behavior [11]. We will present two different applications of FISTA and briefly comment

on their differences. For the first, we define

$$f : \left(a_1^{(1)}, \cdots, a_R^{(K)}\right) \mapsto \frac{1}{2}\left\|\sum_{r=1}^{R}(a_r^{(1)} \otimes \cdots \otimes a_r^{(K)}) + S - \mathcal{Z}\right\|_F^2 + \frac{\mu}{K}\sum_{r=1}^{R}\sum_{k=1}^{K}\|a_r^{(k)}\|^K$$
$$g : S \mapsto \lambda\|S\|_{\text{sum}} \tag{2.195}$$

Notice that we could transfer either of the smooth terms in f to g and still have a problem definition that is consistent with the requirements of FISTA, but this partition is the most natural. An application of FISTA to minimize $f\left(a_1^{(1)}, \cdots, a_R^{(K)}\right) + g(S)$ as they are defined in (2.195) is outlined in Algorithm 4.

Algorithm 4 FISTA: Using the Function Definitions Given in (2.195)

Input: Lipschitz constant L, maximum iterations m, parameters λ, μ, tensor $\mathcal{Z}$, and initial points $A_0^{(1)}, \cdots, A_0^{(K)}, S_0$, and $t_0 = 1$

1: **while** $i < m$ **do**
2: $\quad t_{i+1} \leftarrow \frac{1+\sqrt{1+4t_i^2}}{2}$
3: $\quad$ **for** $(r,k) \in [R] \times [K]$ **do**
4: $\quad\quad C \leftarrow \left(A^{(K)} \odot \cdots \odot A^{(k+1)} \odot A^{(k-1)} \odot \cdots \odot A^{(1)}\right)$
5: $\quad\quad (x_r^{(k)})_i \leftarrow (a_r^{(k)})_i - \frac{1}{L}\left[\left(\mu\|(a_r^{(k)})_i\|^{K-2}\right)(a_r^{(k)})_i + \left((a_r^{(k)})_i C^T + S_{(k)} - \mathcal{Z}_{(k)}\right)C\right]$
6: $\quad\quad (a_r^{(k)})_{i+1} \leftarrow (x_r^{(k)})_i + \frac{t_i - 1}{t_{i+1}}\left[(x_r^{(k)})_i - (x_r^{(k)})_{i-1}\right]$
7: $\quad$ **end for**
8: $\quad S_i' \leftarrow \text{soft\_thresh}(S_i, \frac{2\lambda}{L})$
9: $\quad S_{i+1} \leftarrow S_i' + \frac{t_i - 1}{t_{i+1}}\left[S_i' - S_{i-1}'\right]$
10: **end while**

The terms can also be grouped so that

$$\tilde{f} : \left(a_1^{(1)}, \cdots, a_R^{(K)}\right) \mapsto \frac{1}{2}\left\|\sum_{r=1}^{R}(a_r^{(1)} \otimes \cdots \otimes a_r^{(K)}) + S - \mathcal{Z}\right\|_F^2$$
$$\tilde{g} : S \mapsto \lambda\|S\|_{\text{sum}} + \frac{\mu}{K}\sum_{r=1}^{R}\sum_{k=1}^{K}\|a_r^{(k)}\|^K. \tag{2.196}$$

In this form, applying FISTA requires iteratively solving the subproblem

$$\min_x \quad \frac{\mu}{K}\|x\|^K + \frac{L}{2}\left[x - \left(y - \frac{1}{L}\left(xC^T + S^\star - \mathcal{Z}\right)C\right)\right]. \tag{2.197}$$

Finding a minimizing x would replace step (5) in Algorithm 4. While this subproblem can be solved

using the techniques presented in section 2.7.2, we have found that FISTA in this form is inferior to Algorithm 4.

2.8 Bounding Sub-Optimality

Although the program (2.133) is non-convex, we can provide a certificate that bounds the suboptimality of any stationary point. For simplicity, we will derive this certificate for only the case where $K = 3$, but it is clear that this analysis immediately extends to tensors of arbitrary order. Recall that our non-convex program for tensor RPCA is the following:

$$\min_{\{a_r\},\{b_r\},\{c_r\}} \quad \frac{\mu}{3}\sum_{r=1}^{R} \|a_r\|^3 + \|b_r\|^3 + \|c_r\|^3 + \varphi\left(a_1, \cdots, c_R\right), \tag{2.198}$$

where

$$\varphi\left(a_1, \cdots, c_R\right) = \min_{S} \frac{1}{2}\left\| \left(\sum_{r=1}^{R} (a_r \otimes b_r \otimes c_r \right) + S - \mathcal{Z} \right\|_F^2 + \lambda \|S\|_{\text{sum}}. \tag{2.199}$$

For the matrix completion problem, it has been shown that a stationary point of a similar non-convex algorithm exists whenever the spectral norm of a certain matrix is small enough [10]. The following lemma extends this result to the RPCA problem, for matrices and higher-order tensors.

Lemma 11. *Let* $\{a_r^*\}, \{b_r^*\}, \{c_r^*\}$ *be stationary points of* (2.198)*, let* $X^* = \sum_{r=1}^{R} (a_r^* \otimes b_r^* \otimes c_r^*)$*, and let* S^* *be the optimal* S *given by the function* φ*. If* $\|\mathcal{P}_{X^{*\perp}}\left(S^* - \mathcal{Z}\right)\| \leq \frac{\mu}{2}$*, then* X^* *is the solution to the convex problem* (2.126)*.*

Proof. The point (X^*, S^*) is the solution to (2.126) if and only if zero is in the subdifferential of the objective. Using the (partial) characterization of the subdifferential of the tensor atomic norm given in [75], we know that (X^*, S^*) is the global optimum of (2.126) if there exists a matrix $W^{\perp}$ satisfying

$$(X^* + S^* - \mathcal{Z}) + \mu(W + \mathcal{P}_{X^{\perp}} W^{\perp}) = 0,$$

and $\|W^{\perp}\| \leq \frac{1}{2}$. (Here, W is the unique tensor $\mathcal{P}_{X^*}^0 W = W$, $\|W\| = 1$, and $\langle X^*, W \rangle = \|X^*\|_*$;

recall (2.4)). We have that

$$\mathcal{P}_{X^{*\perp}} W^{\perp} = \mathcal{P}^2_{X^{*\perp}} W^{\perp} \tag{2.200}$$

$$= -\tfrac{1}{\lambda}\, \mathcal{P}_{X^{*\perp}} \left((X^* + S^* - \mathcal{Z}) + \lambda W \right) \tag{2.201}$$

$$= -\tfrac{1}{\lambda}\, \mathcal{P}_{X^{*\perp}} (S^* - \mathcal{Z}). \tag{2.202}$$

Hence, both conditions on $W^{\perp}$ are satisfied if $\| \mathcal{P}_{X^{*\perp}} (S^* - \mathcal{Z}) \| \leq \frac{\mu}{2}$.

□

Although this provides a foundation for the development of a certificate of optimality, Lemma 11 is limited in several respects. The conditions of Lemma 11 could be unnecessarily strong because we only have access to a partial characterization of the subdifferential of the tensor atomic norm, namely, the subdifferential derived in [75].

However, the most important drawback of Lemma 11 is that $\| \mathcal{P}_{X^{*\perp}} (S^* - \mathcal{Z}) \|$ is NP-hard to compute in general, so once a stationary point (X^*, S^*) is recovered, there is no clear way to prove that $\| \mathcal{P}_{X^{*\perp}} (S^* - \mathcal{Z}) \|$ is bounded above. The following results will uncover a computationally feasible method to check whether the non-convex program has recovered the global optimum.

Lemma 12. *Let $(\{a_r^*\}, \{b_r^*\}, \{c_r^*\})$ be a stationary point of (2.198), and let $a_r^{\perp}$ be any vector orthogonal to a_r^*. Then*

$$\left\langle \left(\sum_{r=1}^{R} (a_r^* \otimes b_r^* \otimes c_r^*) + S^* - \mathcal{Z} \right), (a_r^{\perp} \otimes b_r^* \otimes c_r^*) \right\rangle = 0.$$

The same can be said for vectors $b_r^{\perp}$ and $c_r^{\perp}$ defined analogously.

Proof. Let $f(a_1, \cdots, c_R)$ be the objective of (2.198). The point $(\{a_r^*\}, \{b_r^*\}, \{c_r^*\})$ is a stationary point of (2.198) if and only if it nulls the gradient

$$(\nabla_{a_r} f)_i = \left\langle \left(\sum_{r=1}^{R} (a_r \otimes b_r \otimes c_r) + S - \mathcal{Z} \right), (e_i \otimes b_r \otimes c_r) \right\rangle + 3\|a_r\|(a_r)_i = 0.$$

Here, e_i is the i^{th} canonical basis vector, and $(a_r)_i$ denotes the i^{th} component of a_r. Multiplying

$(\nabla_{a_r} f)_i$ by $(a_r^\perp)_i$ and summing over the index i, we then have

$$\left\langle \left(\sum_{r=1}^{R} (a_r \otimes b_r \otimes c_r) \right), (a_r^\perp \otimes b_r \otimes c_r) \right\rangle = -3\|a_r\| \langle a_r^\perp, a_r \rangle,$$
$$= 0.$$

An analogous result holds for vectors $b_r^\perp$ and $c_r^\perp$. □

With this result established, we can now demonstrate how to efficiently compute a bound for $\|\mathcal{P}_{X^{*\perp}} (X^* + S^* - \mathcal{Z})\|$.

Lemma 13. *Let $(\{a_r^*\}, \{b_r^*\}, \{c_r^*\})$ be a stationary point of (2.198), let $X^* = \sum_{r=1}^{R} (a_r^* \otimes b_r^* \otimes c_r^*)$, and let S^* be the optimal S given by the function φ. The following inequality holds:*

$$\|\mathcal{P}_{X^{*\perp}}(S^* - \mathcal{Z})\| \leq \left\| \sum_{r=1}^{R} (a_r^* \otimes b_r^* \otimes c_r^*) + S^* - \mathcal{Z} \right\|_F - \left\| \mathcal{P}^0_{X^*} \left(\sum_{r=1}^{R} (a_r^* \otimes b_r^* \otimes c_r^*) + S^* - \mathcal{Z} \right) \right\| \quad (2.203)$$

Proof. Because $I = \mathcal{P}_{X^{*\perp}} + \mathcal{P}_{X^*}$ and $(\mathcal{P}_{X^{*\perp}})(\mathcal{P}_{X^*}) = 0$, we have that

$$\|\mathcal{P}_{X^{*\perp}}(S^* - \mathcal{Z})\| = \left\| \sum_{r=1}^{R} (a_r^* \otimes b_r^* \otimes c_r^*) + S^* - \mathcal{Z} \right\| - \left\| \mathcal{P}_{X^*} \left(\sum_{r=1}^{R} (a_r^* \otimes b_r^* \otimes c_r^*) + S^* - \mathcal{Z} \right) \right\|. \quad (2.204)$$

Using Lemma 12, we can reduce the rightmost term. Recall that $\mathcal{P}_{X^*} = \mathcal{P}_{U,V,W} + \mathcal{P}_{U^\perp,V,W} + \mathcal{P}_{U,V^\perp,W} + \mathcal{P}_{U,V,W^\perp}$, where U, V, and W are the linear spaces spanned by the vectors $\{a_r\}, \{b_r\}$, and $\{c_r\}$, respectively. Also, for arbitrary vectors q, r, s, we have, for example,

$$\left\langle \mathcal{P}_{U^\perp,V,W} \left(\sum_{r=1}^{R} (a_r^* \otimes b_r^* \otimes c_r^*) + S^* - \mathcal{Z} \right), q \otimes r \otimes s \right\rangle \quad (2.205)$$
$$= \left\langle \sum_{r=1}^{R} (a_r^* \otimes b_r^* \otimes c_r^*) + S^* - \mathcal{Z}, \mathcal{P}_{U^\perp,V,W}(q \otimes r \otimes s) \right\rangle \quad (2.206)$$
$$= \left\langle \sum_{r=1}^{R} (a_r^* \otimes b_r^* \otimes c_r^*) + S^* - \mathcal{Z}, a_r^\perp \otimes r \otimes s \right\rangle \quad (2.207)$$
$$= 0 \quad (2.208)$$

by Lemma 12. The same holds for the operators $\mathcal{P}_{U,V^\perp,W}$ and $\mathcal{P}_{U,V,W^\perp}$. Therefore, we can replace the projection $\mathcal{P}_{X^*}$ in the rightmost term with $\mathcal{P}^0_{X^*}$. The desired inequality follows from the fact that $\|\mathcal{T}\| \leq \|\mathcal{T}\|_F$ for any tensor $\mathcal{T}$. □

To use this bound in practice, one would first find a stationary point $(\{a_r\}^*, \{b_r\}^*, \{c_r\}^*)$, and then use, for example, alternating minimization on q, r and s to find a large value for

$$\left\langle \mathcal{P}_X^0 \left(\sum_{r=1}^R (a_r^* \otimes b_r^* \otimes c_r^*) + S^* - \mathcal{Z} \right), (q \otimes r \otimes s) \right\rangle \tag{2.209}$$

$$= \left\langle \left(\sum_{r=1}^R (a_r^* \otimes b_r^* \otimes c_r^*) + S^* - \mathcal{Z} \right), \mathcal{P}_X^0 (q \otimes r \otimes s) \right\rangle. \tag{2.210}$$

We only have to search over tensors $q \otimes r \otimes s$ in the range of $\mathcal{P}_X^0$, and we have a basis for this space given by our stationary point $(\{a_r\}^*, \{b_r\}^*, \{c_r\}^*)$. If it is possible to make this quantity large enough so that

$$\left\| \sum_{r=1}^R (a_r^* \otimes b_r^* \otimes c_r^*) + S^* - \mathcal{Z} \right\|_F - \left\| \mathcal{P}_{X^*}^0 \left(\sum_{r=1}^R (a_r^* \otimes b_r^* \otimes c_r^*) + S^* - \mathcal{Z} \right) \right\| \leq \frac{\mu}{2}, \tag{2.211}$$

then this stationary point is globally optimal. To be clear, it is possible for a stationary point to be globally optimal but fail to satisfy this certificate.

Just as in [27], we can extend this certificate to provide a bound on the distance to optimality. Let $X = \sum_{r=1}^R (a_r \otimes b_r \otimes c_r)$, let $X^\star$ be a globally optimal point, and let $f(a_1, \cdots, c_R)$ be the objective in (2.198). Following the analysis in [27], we would like to find an element of the subdifferential

$$\mathcal{E} \in \partial f(X) \tag{2.212}$$

that is close to zero. We can bound $\|\mathcal{E}\|_F^2$ with the following:

$$\begin{aligned} \|\mathcal{E}\|_F^2 &= \min_{\mathcal{Y} \in \partial \|X\|_*} \|\mathcal{Y} + \tfrac{1}{\mu}(X + S - \mathcal{Z})\|_F^2 \\ &= \min_{\|W^\perp\| \leq \frac{1}{2}} \|W + \mathcal{P}_{X^\perp}(W^\perp) + \tfrac{1}{\mu}(X + S - \mathcal{Z})\|_F^2 \\ &\leq 1 + \tfrac{1}{\mu}\|X + S - \mathcal{Z}\|_F^2. \end{aligned}$$

We can also use the bound

$$\begin{aligned} \|X - X^*\|_F &\leq \|X\|_F + \|X^*\|_F \\ &\leq \|X\|_F + \|X^*\|_* \\ &\leq \|X\|_F + f(a_1, \cdots, c_R). \end{aligned}$$

Combining these bounds with the process outlined in [27], we have the following bound on the sub-optimality of the point X:

$$f(X) - f(X^*) \leq \langle \mathcal{E}, X - X^* \rangle \tag{2.213}$$

$$\leq \|\mathcal{E}\|_F \cdot \|X - X^*\|_F \tag{2.214}$$

$$\leq (1 + \tfrac{1}{\mu}\|X + S - \mathcal{Z}\|_F^2)(\|X\|_F + f(a_1, \cdots, c_R)). \tag{2.215}$$

While a bit crude, this bound allows us to measure the convergence of our non-convex model with respect to the global minimum.

2.9 Numerical Experiments

We compare our model to existing methods for tensor and matrix RPCA on synthetic data and the escalator-video dataset of [50]. Our experiments demonstrate that our model significantly outperforms all existing methods for tensor- and matrix-based RPCA. We also see that we perform much better than the guarantees given in Theorem 2. Most remarkably, our model is able to recover tensors whose rank is much larger than its side lengths. In this regime, the Tucker-rank of the tensor is no longer an appropriate measure of the complexity of the data, and all existing methods for tensor RPCA are ineffective. These results suggest that algorithms designed to recover low Tucker-rank tensors are suboptimal, and that new theoretical tools must be developed to better understand tensor-based algorithms involving a tensor's CP-rank.

2.9.1 Experiments on Synthetic Data

For each trial of our synthetic-data tests, we created an order-3, cubic dataset that could be represented as the sum of a low-rank tensor and a sparse tensor. To form the low-rank component, we randomly generated three factor matrices $A, B, C \in \mathbb{R}^{20\times R}$, with each entry drawn i.i.d. from the univariate Gaussian distribution with mean zero and variance one. We then formed the low-rank component as

$$X^* = \sum_{r=1}^{R} (a_r \otimes b_r \otimes c_r). \tag{2.216}$$

For each trial, we set the rank bound to be $R + 10$.

To form the sparse component, we generated 20 matrices with sparsity $\frac{\rho_S}{20}$ and non-zero entries drawn i.i.d. from the standard univariate Gaussian distribution. We concatenated each of these matrices along the tubular dimension to form the tensor S^*. Our "observed" dataset was then $\mathcal{Z} = X^* + S^*$. Both the rank R and the sparsity ρ_S were varied.

For each test, we performed 16 trials in parallel and measured the error between the recovered low-rank component X and the actual low-rank component X^* using the relative least-squares loss: $\frac{\|X - X^*\|_F}{\|X^*\|_F}$. If the error was below 10^{-3}, we called the recovery exact. We fit our model using L-BFGS [61], maintaining 10 iterations in memory, and we stopped each trial after 1,000 iterations. For our parameters, we set $\mu = 10^{-5}$ and $\lambda = 10^{-3}$. Both are small because we do not expect there to be any noise in $\mathcal{Z}$. Our results are shown in Figure 2.1, along with the results of the same experiment using matrix RPCA and two existing tensor RPCA methods.

The ranks reported in the figure are upper bounds, as it is possible for a certain $X = \sum_{r=1}^{R}(a_r \otimes b_r \otimes c_r)$ to admit a lower-rank CP-decomposition. However, this is only a concern for $R \geq 30$, as it has been shown that if $\text{rank}(A) + \text{rank}(B) + \text{rank}(C) \geq 2(R+1)$, then the CP-decomposition is unique [49]. Furthermore, the Tucker-rank of the low-rank component is (R, R, R) for $R \leq 20$, and it is $(20, 20, 20)$ for $R \geq 20$. This was explicitly checked for each trial.

The two other tensor-based models, `HoRPCA-C` [32] and `HoRPCA-S` [32, 40], use the sum-of-nuclear-norms (SNN) regularizer, and `HoRPCA-S` [40] is one of the only provable method for recovering low-rank tensors outside of this work [40]. `HoRPCA-S` has the form

$$\begin{aligned} \min_{X,S} \quad & \sum_{i=1}^{K} \lambda_i \|X_{(i)}\|_* + \|S\|_{\text{sum}}, \\ \text{subject to:} \quad & X + S = \mathcal{Z}, \end{aligned} \tag{2.217}$$

where $X_{(i)}$ is the unfolding of tensor X along its i^{th} mode. In [40], the authors prove that with $d_i^{(1)} = \max(d_i, \Pi_{j \neq i} d_j)$, $d_i^{(2)} = \min(d_i, \Pi_{j \neq i} d_j)$, and $\lambda_i = \sqrt{d_i^{(1)}}$, this program exactly recovers X

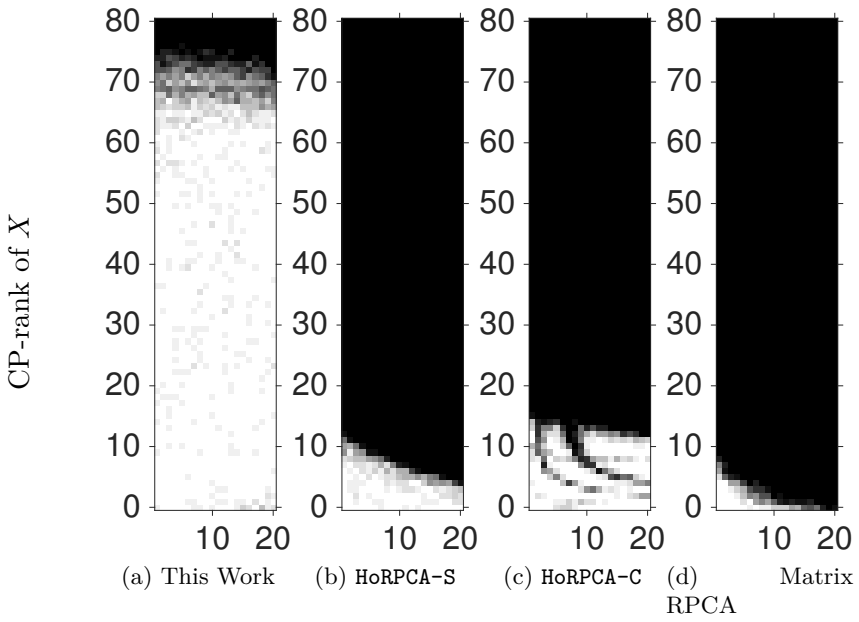


Figure 2.1: A comparison of RPCA methods for recovering the decomposition $\mathcal{Z} = X + S$ with $\mathcal{Z} \in \mathbb{R}^{20\times 20\times 20}$. A pixel is colored white if X is recovered exactly. Each pixel represents the average of 16 trials. Matrix RPCA was applied to $X_{(1)}$. Algorithms (b), (c), and (d) are ill-posed when the CP-rank is greater than 20.

and S with probability $1 - Cd_1^{-3}$ as long as

$$r_k \leq C_r K^{-2} \frac{\rho \mu'^{-1} d_k^{(2)}}{\log^2 d_k^{(1)}}, \quad \textit{and} \quad |\Omega| \leq \rho d_k^{(1)} d_k^{(2)}, \tag{2.218}$$

for some constants C, C_r, and ρ, and incoherence parameter μ'. Here, r_k is the k^{th} component of the Tucker-rank of X. These results mirror the performance guarantees for matrix RPCA applied to the matricized tensor $X_{(1)}$. Our implementation of `HoRPCA-S` uses the parameters $\lambda_i = \sqrt{d_i} = \sqrt{20}$ for all i. In our experiments, we see that `HoRPCA-S` performs better than matrix RPCA, but only marginally. When $R \geq 20$, `HoRPCA-S` is an ill-posed problem, because each matricization of X has full rank. The poor performance of `HoRPCA-S` in this regime is predicted by its guarantees (2.218).

`HoRPCA-C` faces similar problems. This program is defined as

$$\begin{aligned} \min_{X,S} \quad & \|S\|_{\text{sum}} \\ \text{subject to:} \quad & X + S = \mathcal{Z}, \\ & \text{rank}(X_{(i)}) \leq r_i. \end{aligned} \tag{2.219}$$

Although non-convex, it has been shown to outperform other methods for tensor RPCA, including `HoRPCA-S` [32]. However, this model still suffers from the effects of tensor matricization. For $R \geq 20$ in our experiments, `HoRPCA-C` is an ill-posed problem. Using atomic-norm regularization makes it possible to recover tensors with large rank, and Figure 2.1 demonstrates this. For our implementation of `HoRPCA-C`, we set $r_i = R + 1$ for all i. These rank bounds are much tighter than the rank bound $(R + 10)$ we used for our model, but because r_i cannot be larger than 20, it would not make sense to use the rank bound $R + 10$ for the components of the Tucker-rank.

For the matrix RPCA in our experiments, we used the variational approach developed in [4]. The program we solve is

$$\min_{X,S} \quad \max(\|X_{(1)}\|_*, \lambda S_{(1)}) \tag{2.220}$$

$$\text{s.t.} \quad \frac{1}{2}\|X_{(1)} + S_{(1)} - \mathcal{Z}_{(1)}\|_F^2 \leq \epsilon. \tag{2.221}$$

One of the benefits of this approach is that we can choose λ optimally because we know the matrices we would like to recover. We set $\lambda = \frac{\|X^*_{(1)}\|_*}{\|S^*_{(1)}\|_1}$, and because we do not expect there to be any noise in $\mathcal{Z}$, we chose $\epsilon = 10^{-5}$.

2.9.2 Tensor RPCA for Background Subtraction

One of the most natural applications for RPCA is in background subtraction. In this section, we will use our model to identify subjects in the "escalator-video" dataset provided by [50]. Comparing its performance to matrix RPCA, we see that our tensor RPCA more accurately separates the subjects from the background.

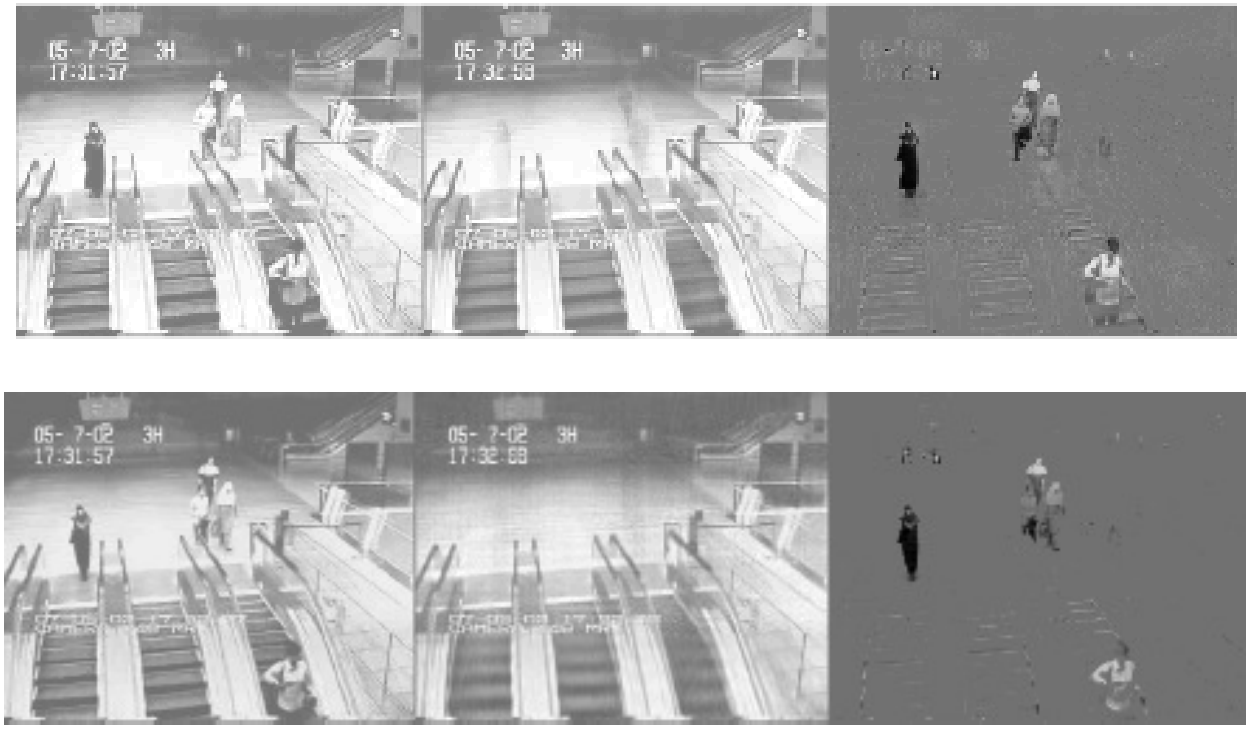


Figure 2.2: The results of using (top) matrix RPCA and (bottom) tensor RPCA for background subtraction. From left to right: original image, low-rank component (X), and sparse component (S). We see that tensor RPCA can more precisely extract the subjects into the sparse component and leave the moving escalator in the low-rank component. This dataset is taken from [50], and frame 5 is shown.

The escalator-video dataset is challenging for background-subtraction models because it contains three moving parts: a time-stamp, escalators, and the subjects. A strong model would be able to recognize that the motion of the escalators and the time-stamp is periodic, so these features belong to the low-rank component of the dataset, and the unpredictable motion of the subjects should be extracted into the sparse component. Using a matrix-based approach to this problem, the video is usually matricized along the temporal mode, so each column of the resulting matrix represents one frame of the original video. It makes intuitive sense that unraveling the dataset in this way would make it more difficult to distinguish temporal patterns, so the periodic motion of the escalators and the time-stamp would not be recognized.

In Figure 2.2, we compare the performance of our tensor RPCA model and matrix RPCA for identifying the background in surveillance video. For tensor RPCA, we set chose the parameters

$\mu = 30,\ \lambda = 0.1$, and the rank-bound $R = 50$. For matrix RPCA, we chose $\lambda = 0.02$ and $\epsilon = 10^4$. All of these parameters were chosen after careful tuning. We see in Figure 2.2 that tensor RPCA recovers a qualitatively superior decomposition, with the sparse component containing the subjects and very little of the stairs, and the low-rank component sufficiently "sharp." In contrast, the low-rank component found by matrix RPCA appears more smoothed and has "ghosts" where the subjects should be removed, while the sparse component contains a significant amount of the stairs.

Even more impressive is the quantitative difference between the decompositions. The numerical rank of the low-rank component found using tensor RPCA is 48, and the sparse component is 5.5%-sparse. For matrix RPCA, the recovered low-rank component has rank 58, and the sparse component is 53%-sparse. Tensor RPCA also extracts more noise from the data, with $\|X^* + S^* - \mathcal{Z}\|_F = 2.50 \times 10^4$ in the tensor case and $\|X^* + S^* - \mathcal{Z}\|_F = 1.34 \times 10^3$ for in the matrix case. These differences, especially between the sparse components, are substantial.

2.10 Conclusion

We have provided theoretical and empirical support showing that using atomic-norm regularization for low-rank tensor recovery is superior to using matrix-based regularizers, like the nuclear norm. Although the atomic norm is generally intractable, our development of a higher-order generalization of Burer-Monteiro factorization allows us to derive a non-convex representation of atomic norm. We have described several algorithms to fit our non-convex model including first-order methods and an alternating-minimization procedure with guaranteed convergence to a stationary point.

In practice, our tensor RPCA significantly outperforms matrix-based RPCA as well as existing implementations of RPCA that use sum-of-nuclear-norm regularization. Most remarkably, our approach to tensor RPCA is able to recover tensors whose CP-rank greatly exceeds all of its side lengths. Our results show that matricization approaches to tensor-based problems do not perform well, and new theoretical tools must be developed to analyze the performance of tensor-based algorithms with respect to their CP-rank, not their Tucker-rank.

As datasets continue to grow in size and complexity, it is necessary for data-analytic algorithms to evolve with them. When high-dimensional datasets are forced into two-dimensional matrix representations, much of the data's structure is lost. It is difficult to abandon matrix-based methods for high-order generalizations because the theoretical foundations of multilinear algebra are extremely underdeveloped, especially compared to the well-studied theory of linear algebra. In order to fully understand the performance benefits that tensor-based methods offer over their order-2 correlates, we must continue to investigate performance guarantees for algorithms that do not rely on matricizing datasets.

Bibliography

[1] E. Acar, D. M. Dunlavy, and T. G. Kolda. A scalable optimization approach for fitting canonical tensor decompositions. Journal of Chemometrics, 25(2):67–86, 2011.

[2] Animashree Anandkumar, Rong Ge, Daniel Hsu, Sham M. Kakade, and Matus Telgarsky. Tensor decompositions for learning latent variable models. JMLR, pages 2773–2832, 2014.

[3] Michael Anderson, Grey Ballard, James Demmel, and Kurt Keutzer. Communication-avoiding QR decomposition for GPUs. In GPU technology conference, 2010.

[4] A. Aravkin, S. Becker, V. Cevher, and P. Olsen. A variational approach to stable principal component pursuit. In Uncertainty in Artificial Intelligence, Quebec City, 2014.

[5] Aleksandr Aravkin, Rajiv Kumar, Hassan Mansour, Ben Recht, and Felix J. Herrmann. Fast methods for denoising matrix completion formulations, with applications to robust seismic data interpolation. SIAM Journal of Scientific Computing, 36(5):S237?–S266, 2014.

[6] A.Y. Aravkin, J.V. Burke, D. Drusvyatskiy, M.P. Friedlander, and S. Roy. Level-set methods for convex optimization. arXiv preprint, 2016.

[7] F. Bach, R. Jenatton, J. Mairal, and G. Obozinski. Optimization with sparsity-inducing penalties. Found. Trends Mach. Learn., 4(1):1–106, January 2012.

[8] Francis Bach, Julien Mairal, and Jean Ponce. Convex sparse matrix factorizations. Technical report, preprint arXiv:0812.1869, 2008.

[9] H. H. Bauschke and P. L. Combettes. Convex Analysis and Monotone Operator Theory in Hilbert Spaces. Springer-Verlag, New York, 2011.

[10] J. Bazerque, G. Mateos, and G. Giannakis. Rank regularization and bayesian inference for tensor completion and extrapolation. IEEE Signal Processing, 2013.

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

[12] D. P. Bertsekas. Nonlinear Programming. Athena Scientific, 1999.

[13] Malgorzata Bogdan, Ewout van den Berg, Chiara Sabatti, Weijie Su, and Emmanuel Candès. SLOPE-adaptive variable selection via convex optimization. The Annals of Applied Statistics, 9(3):1103–1140, 2015.

[14] Thierry Bouwmans and El-hadi Zahzah. Robust PCA via principal component pursuit: A review for a comparative evaluation in video surveillance. Computer Vision and Image Understanding, 2014.

[15] S. Burer and R.D.C. Monteiro. Local minima and convergence in low-rank semidefinite programming. Math. Prog. (series A), 103(3):427–444, 2005.

[16] W. Börsh-Supan. On the evaluation of the function ... for real values of lambda. J. Res. Nat. Bur. Stand., 65(245), 1961.

[17] E. J. Candès, X. Li, Y. Ma, and J. Wright. Robust principal component analysis? Journal of ACM, 58(1):1–37, 2011.

[18] E. J. Candès, X. Li, and M. Soltanolkotabi. Phase retrieval via wirtinger flow: Theory and algorithms. IEEE Transactions on Information Theory, 61(4):1985–2007, 2015.

[19] E. J. Candès and B. Recht. Exact matrix completion via convex optimzation. Foundations of Computational Mathematics, 9:717—772, 2009.

[20] C. Cartis, N. I. M. Gould, and P. L. Toint. Adaptive cubic regularisation methods for unconstrained optimization. part I: motivation, convergence and numerical results. Math. Program., Ser. A, (127):245–295, 2011.

[21] C. Cartis, N. I. M. Gould, and P. L. Toint. Adaptive cubic regularisation methods for unconstrained optimization. part ii: worst-case function- and derivative-evaluation complexity. Math. Program., 130, 2011.

[22] C. Cartis, N. I. M. Gould, and P. L. Toint. On the evaluation complexity of cubic regularization methods for potentially rank-deficient nonlinear least-squares problems and its relevance to constrained nonlinear optimization. SIAM journal on optimization, 23(3):1553–1574, 2013.

[23] Venkat Chandrasekaran, Benjamin Recht, Pablo A. Parrilo, and Alan S. Willsky. The convex geometry of linear inverse problems. Foundations of Computational Mathematics, 12(6):805–849, 2012.

[24] A. B. Conn, N. I. M. Gould, and P. L. Toint. Trust Region Methods. SIAM, Philadelphia, 2000.

[25] Damek Davis. An $O(n \log(n))$ algorithm for projecting onto the ordered weighted ℓ_1 norm ball. Technical Report CAM 15-32, University of California, Los Angeles, 2015.

[26] J. Demmel, L. Grigori, M. Hoemmen, and J. Langou. Communication-optimal parallel and sequential QR and LU factorizations. SIAM J. Sci. Comput., 34:A206?–A239, 2012.

[27] D. Driggs, S. Becker, and A. Aravkin. Adapting regularized low rank recovery models for parallel architectures.

[28] Maryam Fazel, Haitham Hindi, and Stephen P Boyd. A rank minimization heuristic with application to minimum order system approximation. In American Control Conference, 2001. Proceedings of the 2001, volume 6, pages 4734–4739. IEEE, 2001.

[29] Yun Feia, Guodong Rongb, Bin Wanga, and Wenping Wangc. Parallel L-BFGS-B algorithm on gpu. Elsevier, 40, 2014.

[30] M. Frank and P. Wolfe. An algorithm for quadratic programming. Naval Research Logistics Quarterly, 3(1):95–?110, 1956.

[31] S. Friedland and L.-H. Lim. Nuclear norm of higher-order tensors. Mathematics of Computation, 2016.

[32] Donald Goldfarb and Zhiwei Qin. Robust low-rank tensor recovery: Models and algorithms. SIAM Journal on Matrix Analysis and Applications, to appear.

[33] D. Gross. Recovering low-rank matrices from few coefficients in any basis. IEEE Trans. Inf. Theory, 57(3):1548—-1566, 2011.

[34] D. Gross, Yi-Kai Liu, Steven T. Flammia, S. Becker, and Jens Eisert. Quantum state tomography via compressed sensing. Physical Review Letters, 105(15), 2010.

[35] S. Gunasekar, A. Acharya, N. Gaur, and J. Ghosh. Noisy matrix completion using alternating minimization. In Machine Learning and Knowledge Discovery in Databases, pages 194?–209. Springer, 2013.

[36] N. Halko, P. G. Martinsson, and J. A. Tropp. Finding structure with randomness: Stochastic algorithms for constructing approximate matrix decompositions. SIAM Rev., 53(2):217–288, 2011.

[37] Z. Harchaoui, A. Juditsky, and A. Nemirovski. Conditional gradient algorithms for norm-regularized smooth convex optimization. Mathematical Programming, 152(1):75–112, 2015.

[38] M. Hardt. On the provable convergence of alternating minimization for matrix completion. arXiv preprint, 2013.

[39] C.J. Hillar and L.-H. Lim. Most tensor problems are np-hard. J. ACM, (6), 2013.

[40] Bo Huang, Cun Mu, Donald Goldfarb, and John Wright. Provable low-rank tensor recovery. arXiv preprint.

[41] M. Jaggi. Revisiting Frank-Wolfe: Projection-free sparse convex optimization. In International Conference on Machine Learning, 2013.

[42] M. Jaggi and M. Sulovsk. A simple algorithm for nuclear norm regularized problems. In International Conference on Machine Learning (ICML), 2010.

[43] P. Jain, P. Netrapalli, and S. Sanghavi. Low-rank matrix completion using alternating minimization. In Proceedings of the 45th annual ACM Symposium on the Theory of Computing, pages 665?–674. ACM, 2013.

[44] Majid Janzamin, Hanie Sedghi, and Anima Anandkumar. Beating the perils of non-convexity: Guaranteed training of neural networks using tensor methods. arXiv preprint, 2015.

[45] R. Keshavan. Efficient algorithms for collaborative filtering. PhD thesis, Stanford University, 2012.

[46] R. Keshavan and A. Montanari. Regularization for matrix completion. In IEEE International Symposium on Information Theory Proceedings (ISIT), pages 1503?–1507, 2010.

[47] R. Keshavan, A. Montanari, and S. Oh. Matrix completion from noisy entries. In Advances in Neural Information Processing Systems, pages 952?–960, 2009.

[48] R. Keshavan, A. Montanari, and S. Oh. Matrix completion from a few entries. IEEE Transactions on Information Theory, 56:2980–2998, 2010.

[49] T. Kolda and B. Bader. Tensor decompositions and application. SIAM Review, pages 455–500, 2009.

[50] L. Li, W. Huang, I. Gu, and Q. Trian. Statistical modeling of complex backgrounds for foreground object detection. IEEE Transaction on Image Processing, 2004.

[51] Zhouchen Lin, Risheng Liu, and Zhixun Su. Linearized alternating direction method with adaptive penalty for low-rank representation. In Advances in Neural Information Processing Systems, 2011.

[52] Clive Loader. Local Regression and Likelihood. Springer, 1999.

[53] Per-Gunnar Martinsson and Sergey Voronin. A randomized blocked algorithm for efficiently computing rank-revealing factorizations of matrices. arXiv, 2015.

[54] Cun Mu, Yuqian Zhang, John Wright, and Donald Goldfarb. Scalable robust matrix recovery: Frank-wolfe meets proximal methods. SIAM Journal on Scientific Computing, 2016.

[55] B. K. Natarajan. Sparse approximate solutions to linear systems. SIAM Journal of Computing, 24:227?–234, 1995.

[56] Y. Nesterov and B. T. Polyak. Cubic regularization of newton method and its global performance. Math. Program., 108:177–205, 2006.

[57] Praneeth Netrapalli, U. N. Niranjan, Sujay Sanghavi, Animashree Anandkumar, and Prateek Jain. Non-convex robust PCA. In Neural Information Processing Systems, 2014.

[58] Evangelos E. Papalexakis, Christos Faloutsos, and Nicholas D. Sidiropoulos. Tensors for data mining and data fusion: Models, applications, and scalable algorithms. ACM Transactions on Intelligent Systems and Technology, 8(16), 2017.

[59] B. Recht, M. Fazel, and P. Parrilo. Guaranteed minimum rank solutions of matrix equations via nuclear norm minimization. SIAM Review, 52(3):471–501, 2010.

[60] R. Rockafellar and R. Wets. Variational Analysis. Springer Berlin Heidelberg, 2009.

[61] M. Schmidt. minFunc: unconstrained differentiable multivariate optimization in Matlab. 2005.

[62] Y. Shen, Z. Wen, and Y. Zhang. Augmented Lagrangian alternating direction method for matrix separation based on low-rank factorization. Optimization Methods and Software, 2012.

[63] Andrews Sobral and Antoine Vacavant. A comprehensive review of background subtraction algorithms evaluated with synthetic and real videos. Computer Vision and Image Understanding, 122:4–21, 2014.

[64] N. Srebro, J. Rennie, and T. Jaakkola. Maximum-margin matrix factorization. In NIPS, 2005.

[65] Ryota Tomioka and Taiji Suzuki. Spectral norm of random tensors. arXiv preprint, 2014.

[66] Stanimire Tomov, Jack Dongarra, and Marc Baboulin. Towards dense linear algebra for hybrid GPU accelerated manycore systems. Parallel Computing, 36(5-6):232–240, June 2010.

[67] Stephen Tu, Ross Boczar, Mahdi Soltanolkotabi, and Benjamin Recht. Low-rank solutions of linear matrix equations via procrustes flow. In International Conference on Machine Learning, 2016.

[68] L. R. Tucker. Some mathematical notes on three-mode factor analysis. Psychometrika, 1966.

[69] Madeleine Udell, Corinne Horn, Reza Zadeh, and Stephen Boyd. Generalized low rank models. Foundations and Trends in Machine Learning, 9(1):1–118, 2015.

[70] Antoine Vacavant, Thierry Chateau, Alexis Wilhelm, and Laurent Lequiévre. A benchmark dataset for foreground/background extraction. In ACCV 2012, Workshop: Background Models Challenge, pages 291–300, 2012.

[71] Z. Wen, W. Yin, and Y. Zhang. Solving a low-rank factorization model for matrix completion by a nonlinear successive over-relaxation algorithm. Mathematical Programming Computation, pages 1–29, 2010.

[72] S. J. Wright. Primal-Dual Interior-Point Methods. SIAM, 1997.

[73] Y. Wright, A. Ganesh, S. Rao, and Y. Ma. Robust principal component analysis: Exact recovery of corrupted low-rank matrices by convex optimization. In Neural Information Processing Systems (NIPS), 2009.

[74] Xinyang Yi, Dohyung Park, Yudong Chen, and Constantine Caramanis. Fast algorithms for robust PCA via gradient descent. In Advances in neural information processing systems, 2016.

[75] Ming Yuan and Cun-Hui Zhang. On tensor completion via nuclear norm minimization. Foundations of Computational Mathematics, 16:1031–1068.

[76] Xiangrong Zeng and Mário A. T. Figueiredo. The ordered weighted l1 norm: Atomic formulation, dual norm, and projections. arXiv, 2014.

[77] Zihan Zhou, Xiaodong Li, John Wright, Emmanuel J. Candès, and Yi Ma. Stable principal component pursuit. In IEEE International Symposium on Information Theory (ISIT), 2010.

[78] H. Zou and T. Hastie. Regularization and variable selection via the elastic net. Journal of the Royal Statistical Society (B), 67:301–320, 2005.

Appendix A

Factorization Theorem Details

Both problems share the same set of local minimizers. This was first observed in the case of SDP in [15] and later generalized in [5].

Our variant of the theorem:

Theorem 5. *Consider an optimization problem of the following form*

$$\min_{X \succeq 0} \quad f(X), \quad \textit{such that} \quad \operatorname{rank}(X) \leq k \tag{A.1}$$

where $X \in \mathbb{R}^{n \times n}$ is a positive semidefinite real matrix, and f is a lower semi-continuous (lsc) function mapping to $[-\infty, \infty]$ and has a non-empty domain over the positive semi-definite matrices. Using the change of variable $X = PP^T$, take $P \in \mathbb{R}^{n \times k}$, and consider the problem

$$\min_{P} \quad g(P) \stackrel{def}{=} f(PP^T) \tag{A.2}$$

Let $\bar{X} = \bar{P}\bar{P}^T$, where $\bar{X}$ is feasible for (A.1). Then $\bar{X}$ is a local minimizer of (A.1) if and only if $\bar{P}$ is a local minimizer of (A.2).

Proof. We follow ideas from both [15] and [5]. From Lemma 2.1 in [15], if both P and K are $n \times k$ matrices, then $PP^T = KK^T$ if and only if $P = QK$ for some orthogonal matrix $Q \in \mathbb{R}^{k \times k}$. The objective in (A.2) depends only on PP^T, so it is clear that P is a local minimizer of (A.2) if and only if PQ is a local minimizer for all orthogonal Q.

Note that g defined by $P \mapsto f(PP^T)$ is also lsc since it is the composition of f with the continuous function $P \mapsto PP^T$. We require the functions to be lsc so that the notion of "local minimizer" is well-defined.

Suppose $\bar{X}$ is a local minimizer of (A.1), i.e., for some $\epsilon > 0$ there is no better feasible point, and factor it as $\bar{X} = \bar{P}\bar{P}^T$ for some $n \times k$ matrix $\bar{P}$. Then we claim $\bar{P}$ is a local minimizer of (A.2). If it is not, then there exists a sequence $P_n \to \bar{P}$ with $g(P_n) < g(\bar{P})$ for all n. By continuity of the map $P \mapsto PP^T$, there is some n large enough such that $X_n \stackrel{\text{def}}{=} P_n P_n^T$ is within ϵ of $\bar{X}$, with $f(X_n) < f(\bar{X})$, contradicting the local optimality of $\bar{X}$.

We prove the other direction using contrapositive. Suppose $\bar{X}$ is not a local minimizer of (A.1), so there is a sequence $X_n \to \bar{X}$ with $f(X_n) < f(\bar{X})$. Factor each X_n as $X_n = P_n P_n^T$, and observe that it is not true that P_n converges. However, (X_n) converges and hence is bounded, thus (P_n) is bounded as well (for the spectral norm, $\|X_n\| = \|P_n\|^2$, and over the space of matrices, all norms are equivalent). Since P_n are in a finite dimensional space, so the Bolzano-Weierstrass theorem guarantees that there is a sub-sequence of (P_n) that converges. Let the limit of the sub-sequence be $\bar{P} = \lim_{k\to\infty} P_{n_k}$, and note $\bar{P}\bar{P}^T = \bar{X}$ since $X_{n_k} \to \bar{X}$. Then $g(P_{n_k}) = f(X_{n_k}) < f(\bar{X}) = g(\bar{P})$, so $\bar{P}$ is not a local solution. It also follows from the first paragraph of the proof that there can not be another local solution $\tilde{P}$ that also satisfies $\tilde{P}\tilde{P}^T = \bar{X}$. □

Remark 3. *We recover the settings of both [5, Thm. 4.1] and [15], since allowing f to be extended valued and lsc encompasses constraints.*

Remark 4. *[5, Cor. 4.2] can be clarified to state that the two problems listed there are "equivalent" in the sense that they share the same local minimizers (as well as global), using similar continuity arguments on the mapping $\mathcal{R}$ and its adjoint. Furthermore, since the constraints are compact, solutions exist in both formulations.*