Literature Review

Nonconvex Optimization Methods & Applications: Low-Rank Matrix Factorization [1] [2]

Abstract

While nonconvex optimization methods are not ideal due to local minima and saddle points, various iterative methods work in practice under specific conditions. This is possible given conditions like a benign landscape (as an example) that can provide theoretical convergence guarantees in a provable manner. Moreover, two different methodologies are discussed to solve nonconvex problems under matrix factorization applications.

1 Overview

1. Overview & Motivation
2. Canonical Matrix Factorization Problems
3. A Short Discussion on Nonconvexity & Convexity
4. Two-Stage Algorithms
    - Gradient Descent Method
    - Alternating Minimization Method
    - Iterative Hard Thresholding Method
    - Initialization w/ Spectral Method
5. Global Landscape Algorithms
    - Gradient-based Methods
    - Hessian-based Methods
    - Almost-Convex-AGD & Negative-Curvature-Descent Method
6. Conclusion & References
7. Appendices (Various Proofs)

Convex optimization methods, such as gradient descent methods, have been extremely popular to solve optimization problems due to theoretical guarantees of global convergence, fast convergence rate, and ease of use. However, some research problems are inherently nonconvex meaning multiple local minima exist, and thus algorithms may not necessarily converge to the best global solution. Despite this fact, nonconvex algorithms exist that are efficient in a provable manner under certain conditions. This report will include discussion on various nonconvex methods applied to matrix factorization applications. Some examples of matrix factorization problems include problems such as matrix sensing, phase retrieval, matrix completion, blind deconvolution, robust principal component analysis, phase synchronization, and joint alignment. The nonconvex methods are efficient and achieve theoretical convergence guarantees due to the use of statistical models and specific conditions that are met like restricted isometry property. This report will mainly discuss two different approaches to the nonconvex problem: 1) two-part algorithms with a particular initialization step with iterative refinement of the initial estimate; and 2) initialization-free algorithms and optimization landscape analysis. Before doing so, various canonical matrix factorization problems are introduced to further motivate discussion of nonconvex methods.

2 Canonical Problems

2.1 Matrix Sensing [3]

Assume m observations are made and denoted as $y_i \in \mathbb{F}, \forall i \in 1, ..., m$. The observation model is described as $y_i = \langle \boldsymbol{A_i}, \boldsymbol{M_*} \rangle = \text{trace}(\boldsymbol{A_i}\boldsymbol{M_*}^T)$ where we wish to reconstruct the unprocessed measurement matrix $\boldsymbol{M_*}$ (assumed rank r) given the set of sensing matrices known beforehand denoted $\boldsymbol{A_i}$. A simple example can be considered where $\boldsymbol{M_*}$ is a rank-1 matrix (represented as $\boldsymbol{M_*} = \boldsymbol{X_*}\boldsymbol{X_*}^T \in \mathbb{R}^{n \times n}$). In this case, the minimization problem reduces to the classical least-squares setup

$$\min_{\boldsymbol{X}} \frac{1}{4m} \sum_{i}^{m} (\langle \boldsymbol{A_i}, \boldsymbol{X}\boldsymbol{X}^T \rangle - y_i)^2 \tag{2.1.1}$$

One important observation to make is that the trace operator is unitary invariant and thus multiple solutions exist for the minimization problem since the solution can be multiplied by a unitary matrix. Meaning, this is a nonconvex optimization problem (albeit easily solvable and well understood). Generally, for the rank-r case, we describe $\boldsymbol{M_*} = \boldsymbol{L_*}\boldsymbol{R_*} = \boldsymbol{U_*}\boldsymbol{\Sigma_*}^{1/2}\boldsymbol{\Sigma_*}^{1/2}\boldsymbol{V_*}^T$ where $\boldsymbol{U_*}, \boldsymbol{V_*}, \boldsymbol{\Sigma_*}$ represent the singular value decomposition of $\boldsymbol{M_*}$ to solve:

$$\min_{\boldsymbol{L},\boldsymbol{R}} \frac{1}{4m} \sum_{i=1}^{m} (\langle \boldsymbol{A_i}, \boldsymbol{L}\boldsymbol{R}^T \rangle - y_i)^2 \tag{2.1.2}$$

Some examples of practical matrix sensing problems include facial recognition and quantum state tomography. This matrix sensing problem will be used extensively in the coming chapters when discussing theoretical guarantees of different methods. The rest of this chapter is included to show variety of matrix factorization problems but may not necessarily show up in subsequent chapters.

2.2 Robust Principle Component Analysis (RPCA) [4]

Classical PCA boils down to taking a singular value decomposition of a matrix in order to create a low-rank approximation of the original matrix. However, under non-ideal conditions like outlier data, the SVD of a matrix will incorporate the outlier data as part of the low-rank construction. To avoid this, we wish to decompose data matrix $\boldsymbol{\Gamma_*} = \boldsymbol{M_*} + \boldsymbol{S_*}$ where $\boldsymbol{M_*}$ is the low-rank matrix to recover but is essentially corrupted by $\boldsymbol{S_*}$ the sparse outliers. For the general r-rank case ($\boldsymbol{M_*} = \boldsymbol{L}\boldsymbol{R}^T$), the minimization problem is the following assuming all entries of $\boldsymbol{\Gamma_*}$ are observed (no missing entries):

$$\min_{\boldsymbol{L},\boldsymbol{R},\boldsymbol{S}} \frac{1}{4p} \|\boldsymbol{\Gamma_*} - \boldsymbol{L}\boldsymbol{R}^T - \boldsymbol{S}\|_F^2 \text{ where } \boldsymbol{S} \in \boldsymbol{S}_\alpha = \{\boldsymbol{S} : \|(\boldsymbol{S_*})_{i,\cdot}\|_0 \leq \alpha n_2, \|(\boldsymbol{S_*})_{\cdot,j}\|_0 \leq \alpha n_1 \; \forall i, j\}$$

This is in contrast to the convex relaxation version which is written as follows:

$$\min_{\boldsymbol{M},\boldsymbol{S}} \|\boldsymbol{M}\|_* + \lambda \|\boldsymbol{S}\|_1 \text{ such that } \boldsymbol{M} + \boldsymbol{S} = \boldsymbol{\Gamma}$$

In this case, sparsity is promoted for the outlier matrix since there is an implicit assumption that outlier data is relatively uncommon and represents a small portion of the overall data matrix. Moreover, the $\boldsymbol{L}\boldsymbol{R}$ term is now replaced by the low-rank matrix $\boldsymbol{M}$ that has been relaxed by the nuclear norm constraint. These kinds of RPCA problems appear in computer vision, medical imaging, and surveillance to provide a few examples.

2.3 Phase Retrieval [5]

For the rank-1 case, suppose we have m measurements y_i, $i \in \{1, ..., m\}$ and $\boldsymbol{M_*} = \boldsymbol{X_*}\boldsymbol{X_*}^T$. The problem is now to reconstruct $\boldsymbol{X_*}$ given y_i. These parameters are related via $y_i = \|\boldsymbol{a_i}^T \boldsymbol{X_*}\|_2^2 = \boldsymbol{a_i}\boldsymbol{M_*}\boldsymbol{a_i}$ where $\boldsymbol{a_i}$ is the design vector that is known beforehand. Given this setup, the optimization problem is described as:

$$\min_{\boldsymbol{X}} \frac{1}{4m} \sum_{i=1}^{m} (\|\boldsymbol{a_i}^T \boldsymbol{X}\|_2^2 - y_i)^2 \tag{2.3.1}$$

An astute observation is the similarity with matrix sensing in that $\boldsymbol{A_i} = \boldsymbol{a_i}\boldsymbol{a_i}^T$ so phase retrieval is a particular setup of the general matrix sensing problem. Phase retrieval shows up in applications such as X-ray crystallography. The main idea to this problem is to figure out phase information given only magnitude information since this is what the transducers are able to detect.

3 A Short Discussion on Nonconvexity & Convexity

Before discussing the nonconvex methods, a short discussion will be provided to set context on the two chapters to follow. Usually, within a matrix factorization problem, it is desirable to solve $\min_{\boldsymbol{L},\boldsymbol{R}} f(\boldsymbol{L}, \boldsymbol{R})$ subject to $\boldsymbol{M} = \boldsymbol{L}\boldsymbol{R}^T$. More often than not, the problem is inherently nonconvex and may seem impossible to solve for the global solution. Classically, the approach taken is to use *convex relaxation* to replace $\boldsymbol{M} = \boldsymbol{L}\boldsymbol{R}^T$ by a nuclear norm restraint as an example and instead solve for $\boldsymbol{M}$ instead of $\boldsymbol{L}\boldsymbol{R}$. These kinds of methods have been incredible successful in practice and have performance guarantees when talking about things like sample complexity and robustness (against noise). However, the computational cost is quite large $\mathcal{O}(n^3)$ depending on the size of the matrix which may even far exceed the time it takes to read the data. Moreover, extremely large matrices will require large storage allocation both in storage and memory. In this paper, low-rank matrix estimation will be based on nonconvex methods $(\boldsymbol{L}, \boldsymbol{R})$ instead of convex approaches. It may not seem intuitive, but it turns out that the nonconvex approach has low memory/storage requirements, per-iteration cost significantly improves (e.g. no nuclear norm computation), malleability towards parallelization, and scalability to large-scale problems [1]. A fair question one may ask is whether it is even possible to solve a nonconvex problem. On the surface it would appear to be a daunting task. Remarkably, many important nonconvex problems are not difficult to solve. In the case of matrix factorization problems, straightforward first-order methods are guaranteed to succeed with not too many iterations and thus have low computational cost [6]. The main idea is to analyze the estimation problem which could have a benign landscape that would make problematic regions like local minima nonexistent. Specifically, the two chapters to follow will focus on two key ideas:

- **Basin of attraction**
  For many problems, one can find a large basin of attraction (e.g. L_2 sphere) centered around the global solution. This set is convex around the minimizer so clearly, if the initialization begins within the basin, then iterative methods like gradient descent are guaranteed to converge and do so quickly. The idea of how to initialize within the basin is discussed in the section about the spectral method.

- **Benign (Harmless) Landscape**
  Many problems have a "benign global landscape" when the sample size is large enough. This means that there are no false local solutions (i.e. all local solutions are global solutions) and that the only unwanted critical point situations involve strict saddle points. As will be seen later on, it is surprisingly simple to escape saddle points and continue the descent in many situations.

These two key ideas inspire two different branches of nonconvex methods: 1) two-stage approaches and 2) global landscape analysis methods. The next chapter will focus on two-stage algorithms.

4 Two-Stage Algorithms

Two-stage algorithms are inspired by the idea of a cost function having a basin of attraction. Stage one involves initialization which will locate a guess that belongs to the basin. Stage two involves refining the solution iteratively without leaving the basin. Clearly, this is very efficient and essentially reduces to a convex problem (locally). There are three important two-stage algorithms that will be discussed in this paper: gradient descent, alternating minimization, and iterative hard thresholding.

4.1 Gradient Descent [7]

Gradient descent in a non-convex setting is expressed similarly to the convex setting. For example, with matrix sensing, using the matrix sensing cost function EQ 2.1.1 and assuming symmetry of apriori sensing matrix ($\boldsymbol{A}_i = \boldsymbol{A}_i^T$), the gradient rule update is expressed as the following:

$$\boldsymbol{X}_{k+1} = \boldsymbol{X}_k - \alpha_k \underbrace{\frac{1}{m}\sum_{i=1}^{m}(\langle \boldsymbol{A}_i, \boldsymbol{X}_k\boldsymbol{X}_k^T\rangle - y_i)\boldsymbol{A}_i\boldsymbol{X}_k}_{\nabla f(\boldsymbol{X}_k)}$$

However, this rule will only converge if the sensing operator $\mathcal{A}(\cdot)$ satisfies a condition known as RIP condition (will be discussed later on this page). The sensing operator is the "function" that maps the input matrix of interest $\boldsymbol{X}$ to the observation domain processed by the sensing matrix. Stated mathematically, for matrix sensing cost function EQ 2.1.1, the operator is defined as:

$$\mathcal{A}(\boldsymbol{X}) = [m^{-1/2}\langle \boldsymbol{A}_i, \boldsymbol{X}\rangle]_{1\leq i\leq m} \tag{4.1.1}$$

$\mathcal{A}(\boldsymbol{X})$ must satisfy the RIP condition for convergence. The definition for RIP is written as follows.

Definition 4.1.1 (Restricted Isometry Property (RIP))**.** *An operator $\mathcal{A} : \mathbb{R}^{n_1\times n_2} \to \mathbb{R}^m$ (EQ 4.1.1) is said to satisfy the r-RIP condition with constant $\delta_r < 1$ if:*

$$(1-\delta_r)\|\boldsymbol{X}\|_F^2 \leq \|\mathcal{A}(\boldsymbol{X})\|_2^2 \leq (1+\delta_r)\|\boldsymbol{X}\|_F^2 \quad \forall \boldsymbol{X} \text{ where } rank(\boldsymbol{X}) \leq r \tag{4.1.2}$$

This definition states that the operator preserves the norm of the matrix when dealing with a rank-r matrix. To put it more simply, imagine a triangle on the Cartesian 2D plane. The triangle can move, rotate, and be reflected but the distance between the vertices is preserved. The triangle is not allowed to be scaled or distorted in shape. This is what an isometry describes to give a conceptual example. Surprisingly, many random matrix sensing matrices satisfy this r-RIP condition if $\boldsymbol{A}_i \sim \mathcal{N}(0,1)$. This is also usually the case in phase retrieval for $\boldsymbol{a}_i$ as well. The RIP condition implies smoothness is well controlled within the basin and local strong convexity is achieved within the basin. Before discussing convergence for matrix sensing, due to the trace operator being unitary invariant, a more suitable error metric is the following:

$$\text{dist}(\boldsymbol{X}, \boldsymbol{X}_*) = \min_H \|\boldsymbol{X}\boldsymbol{H} - \boldsymbol{X}_*\|_F \text{ where H is a unitary matrix}$$

Given this metric above, the convergence of gradient descent for the matrix sensing problem is stated below.

Theorem 4.1.2 (Matrix sensing (rank-r) convergence of GD)**.** *The matrix sensing problem, with cost function EQ 2.1.1, must have the $\mathcal{A}(\boldsymbol{X})$ operator (EQ 4.1.1) satisfy the 6r-RIP condition (EQ 4.1.2) with constant $\delta_{6r} \leq \frac{1}{10}$. Then, $\exists c_0, c_1 > 0$ such that if dist$(\boldsymbol{X_0}, \boldsymbol{X}_*) \leq \frac{\sigma_r(\boldsymbol{M}_*)}{16}$ then with $\alpha_k = \frac{c_0}{\sigma_1(\boldsymbol{M}_*)}$ obeys convergence:*

$$dist(\boldsymbol{X}_k, \boldsymbol{X}_*) \leq (1-\frac{c_1}{\kappa})^k dist^2(\boldsymbol{X}_0, \boldsymbol{X_*}) \; \forall k = 0, 1, ...$$

Proof: Appendix A: Matrix Sensing Gradient Method Proof

Thus, the distance between the initial solution $\boldsymbol{X}_0$ and the global minima $\boldsymbol{X}_*$ becomes increasingly smaller with convergence rate on par with classical gradient descent for convex problems. If the nonconvex problem does not satisfy RIP (e.g. phase retrieval), then the problem becomes challenging since the smoothness condition is not well-controlled and local strong convexity may fail so vanilla GD will get stuck in local minima. To resolve this dilemma, we wish to identify a constrained direction within a sphere that has nice geometry and then apply a regularized GD method to ensure the iterates stay in the constrained region set. This means that the descent direction must be along a specific trajectory to achieve smoothness and strong convexity even though the initalizer is inside the local ball. The theorem below discusses desired smoothness conditions for phase retrieval since this usually does not satisfy RIP.

Lemma 4.1.3 (Restricted smoothness (phase retrieval)). *For $\boldsymbol{x} \in \mathbb{R}^n$, $\exists c_0, ..., c_3 > 0$ such that for arbitrary $m \geq c_0 n \log(n)$ obeys the following for phase retrieval cost function EQ 2.3.1:*

$$\nabla^2 f(\boldsymbol{x}) \preceq c_1 \log(n) \boldsymbol{I}_n$$

but specifically $\forall \boldsymbol{x}$ that meet $\|\boldsymbol{x} - \boldsymbol{x}_*\|_2 \leq c_2$ and $\max_j |\boldsymbol{a}_j^T(\boldsymbol{x} - \boldsymbol{x}_*)| \leq c_3\sqrt{\log(n)}$.

This means that given an initializer within an c_2-ball, the hessian $\nabla^2 f(\boldsymbol{x})$ will be well-behaved and smooth so long as the iterates are nearly orthogonal ($\langle ., . \rangle \leq c_3$) to the sampling vectors $\boldsymbol{a}_j$. In the literature, this means $\boldsymbol{x}$ is "incoherent" with $\boldsymbol{a}$ and keeping the iterates nearly orthogonal ($\leq c_3$) to sampling vectors results in convergence. Thus, the iterates must be kept in this set of near-orthogonality. Surprisingly, vanilla GD does this already for the problems mentioned above without any kind of explicit regularization. In other words, vanilla GD will force its iterates to stay incoherent with sampling vectors and matrices. Therefore, the vanilla GD trajectory, with high probability, already belongs to a region that is smooth and achieves strong convexity. However, there does exist explicit regularized GD methods to promote incoherence (via an additional term in the cost function) for various problems to improve computational performance. The basic structure looks like $\min_x f_{\text{reg}}(\boldsymbol{x}) = f(\boldsymbol{x}) + \lambda G(\boldsymbol{x})$ with λ as the regularization parameter to force incoherence. The function $G(\boldsymbol{x})$ is different for each application and there is no "one size fits all" approach. For matrix sensing specifically, the following cost function has been proposed by [8]:

$$G(\boldsymbol{L}, \boldsymbol{R}) = G_0(c_1\|\boldsymbol{L}\|_F^2) + G_0(c_2\|\boldsymbol{R}\|_F^2) + \sum_{i=1}^{rows} G_0(c_3\|\boldsymbol{L}_{i,\cdot}\|_2^2) + \sum_{i=1}^{rows} G_0(c_4\|\boldsymbol{R}_{i,\cdot}\|_2^2)$$

for selected non-zero scalars $c_1, ..., c_4$ and $G_0(x) = \max\{x - 1, 0\}^2$. By doing this, the incoherence can be encouraged although as mentioned before, it is not required since vanilla GD does a great job already at ensuring incoherence.

4.2 Alternating Minimization [9]

Alternating minimization, under a matrix sensing situation, involves solving between the two sub-problems:

$$\boldsymbol{R}_{k+1} = \min_{\boldsymbol{R}} f(\boldsymbol{L}_k, \boldsymbol{R}) = \|\mathcal{A}(\boldsymbol{L}_k\boldsymbol{R}^T - \boldsymbol{M}_*)\|_F^2 \tag{4.2.1}$$

$$\boldsymbol{L}_{k+1} = \min_{\boldsymbol{L}} f(\boldsymbol{L}, \boldsymbol{R}_k) = \|\mathcal{A}(\boldsymbol{L}\boldsymbol{R}_k^T - \boldsymbol{M}_*)\|_F^2 \tag{4.2.2}$$

where both matrices are updated sequentially given initialization matrices $\boldsymbol{L}_0, \boldsymbol{R}_0$. Since minimization occurs with respect to one variable, each substep simplifies to the classical least-squares setup. Each step is then solved using conjugate gradient algorithm. If the reader is not familiar with this method, it is fairly similar to vanilla gradient descent except the current search direction is required to be orthogonal to the last search direction and so forth. Interestingly, if L_0 is initialized to be based on the singular vectors of $\mathcal{A}^*(\boldsymbol{y})$, then convergence is guaranteed. The following theorem describes this succinctly.

Theorem 4.2.1 (Matrix Sensing AltMin Convergence)**.** *Under the context of the general rank-r cost function EQ 2.1.2, suppose the sensing operator satisfies 2r-RIP (EQ 4.1.2) with RIP constant* $\delta_{2r} \leq \frac{1}{100\kappa^2 r}$*. If* $\boldsymbol{L}_0$ *represents the first r left singular vectors of* $\mathcal{A}^*(\boldsymbol{y})$*, then AltMin has the following convergence:*

$$\|\boldsymbol{M}_* - \boldsymbol{L}_k \boldsymbol{R}_k^T\|_F \leq \epsilon \,,\, \forall k \geq 2\log(\frac{\|\boldsymbol{M}_*\|_F}{\epsilon})$$

As seen above, AltMin has better iteration complexity compared to vanilla GD. To be clear, κ above is the condition number for matrix $\boldsymbol{M}_*$ and $\mathcal{A}^*(\boldsymbol{y})$ is the adjoint operator that satisfies $\langle \mathcal{A}(\boldsymbol{x}), \boldsymbol{y} \rangle = \langle \boldsymbol{x}, \mathcal{A}^*(\boldsymbol{y}) \rangle$.

4.3 Iterative Hard Thresholding [10]

Iterative Hard Thresholding (IHT), while similar to gradient descent, iteratively updates in the full matrix space $\boldsymbol{M}$ instead of $\boldsymbol{L}, \boldsymbol{R}^T$ (where $\boldsymbol{M} = \boldsymbol{L}\boldsymbol{R}^T$)and then performs hard threshold on the singular values to retain low-rank conditions. Notice that this problem remains nonconvex since $\|\boldsymbol{\sigma}(\boldsymbol{M})\|_0 = r$. Explicitly, we have that:

$$\boldsymbol{M}_{k+1} = \mathcal{P}_r(\boldsymbol{M}_k - \alpha_k \nabla f(\boldsymbol{M}_k)) \quad \forall k = 0, 1, ...$$

and for matrix sensing specifically we have that $f(\boldsymbol{M}) = \frac{1}{2}\|\mathcal{A}(\boldsymbol{M}) - \mathcal{A}(\boldsymbol{M}_*)\|_F^2$. The $\mathcal{P}_r(\cdot)$ operator will return the best r-rank approximation by setting $\sigma_{r+1}, ..., \sigma_n = 0$. One advantage is that the iterates are low rank so $\boldsymbol{M}_k$ can be stored efficiently (memory size) through the use of a compact SVD. One important note to make is that a full SVD is not performed here when thresholding the singular values. Instead, in practice, a partial SVD is computed via Krylov-based subspace methods like Lanczos algorithm or other linear algebra algorithms. This is done for efficiency as computing a full SVD can be computationally expensive. Moreover, for matrix sensing, there are convergence guarantees for this IHT method.

Theorem 4.3.1 (Matrix sensing convergence for IHT)**.** *For the matrix sensing problem EQ 2.1.2, assume the sensing operator* $\mathcal{A}$ *(EQ 4.1.1) satisfies 2r-RIP (EQ 4.1.2) with constant* $\delta_{2r} \leq \frac{1}{3}$*. For initial iterate* $\boldsymbol{M}_0 = \boldsymbol{0}$ *and step size* $\alpha = \frac{1}{(1+\delta_{2r})}$*, then IHT achieves the following:*

$$\|\boldsymbol{M}_k - \boldsymbol{M}_*\|_F \leq \epsilon \text{ but only when } k \geq c_1 \log(\|\boldsymbol{M}_*\|_F/\epsilon) \text{ for some } c_1$$

Thus, it is possible to work with the full matrix space instead of each $\boldsymbol{L}, \boldsymbol{R}^T$ matrix individually in the nonconvex setting. Now that these two two-stage algorithms have been discussed, the question remains on how to initialize such that the first guess is inside the basin of attraction. The next section will discuss this important question.

4.4 Spectral Method (Initialization Technique) [11]

Given the methods in the previous section, initialization plays a key role for basin of attraction methods. One well known approach involves the spectral method. Given a data sample matrix $\boldsymbol{Y}$, this is decomposed to the following form $\boldsymbol{Y} = \boldsymbol{Y}_* + \boldsymbol{\Delta}$ where $\boldsymbol{Y}_*$ represents data whose eigenspace represents the truth and $\boldsymbol{\Delta}$ contains fluctuations due to finite-sample effect since the matrix is of finite size. The goal is to estimate the truth by analyzing the subspace information of $\boldsymbol{Y}$ assuming the fluctuations are "well controlled". Assume $\boldsymbol{Y}$ is the data matrix with samples and $\boldsymbol{Y}_*$ is a symmetric matrix with real eigenvalues such that $\text{SVD}(\boldsymbol{Y}_*) \to \boldsymbol{U}_*$ the true subspace of the global minimizer. Say $\boldsymbol{Y}_*$ is rank-r, then it is assumed that the spectral-gap $\lambda_r(\boldsymbol{Y}_*) - \lambda_{r+1}(\boldsymbol{Y}_*)$ is nonzero. As an example, if $\boldsymbol{Y}_*$ is positive semi definite, then the spectral gap will clearly be nonzero for some r. It turns out that the subspace $\boldsymbol{U}_Y$ is close to subspace $\boldsymbol{U}_{Y_*}$ by the following theorem.

Theorem 4.4.1 (Davis-Kahan $\sin\Theta$ Theorem)**.** *Given the positive spectral gap, the error between subspaces for the true matrix and perturbed true matrix are expressed as follows:*

$$dist(\boldsymbol{U}, \boldsymbol{U}_*) = \|\boldsymbol{U}\boldsymbol{U}^T - \boldsymbol{U}_*\boldsymbol{U}_*^T\|_F \leq \frac{\|\boldsymbol{\Delta}\|_2}{\lambda_r(\boldsymbol{Y}_*) - \lambda_{r+1}(\boldsymbol{Y}_*)}$$

Therefore, the perturbed subspace $\boldsymbol{U}$ of $\boldsymbol{Y}$ is bounded by the spectral norm of the fluctuation matrix $\boldsymbol{\Delta}$ and the spectral gap. This is great news because it provides sound theory as to why the subspace of the data matrix would work as an initialization for two-stage algorithms since it is close to the true subspace $\boldsymbol{U}_*$ of the global minimizer. To provide an example, the spectral theory can be applied to matrix sensing. Let the surrogate matrix be expressed as $\boldsymbol{Y} = \frac{1}{m}\sum_{i=1}^m y_i \boldsymbol{A}_i$ where the $\boldsymbol{A}_i$ are the sensing matrices known beforehand and y_i is the observation data. Recall that we wish to recover the set of measurements $\boldsymbol{M}_*$ but this data is "hidden" by the sensing matrices and only y_i is collected. This surrogate matrix $\boldsymbol{Y}$ is decomposed via SVD such that $\boldsymbol{L}_0 = \boldsymbol{U}\boldsymbol{\Sigma}^{1/2}$ and $\boldsymbol{R}_0 = \boldsymbol{V}\boldsymbol{\Sigma}^{1/2}$. Then, the end goal is to find $\boldsymbol{L}_*\boldsymbol{R}_* = \boldsymbol{M}_*$. If the sensing operator satisfies 2r-RIP then it is possible to bound the initializer with the global solution.

Lemma 4.4.2 (Matrix Sensing Spectral Initialization)**.** *Given that $\boldsymbol{M}_*$ is rank-r matrix and $\mathcal{A}$ (EQ 4.1.1) satisfies 2r-RIP condition (EQ 4.1.2) with constant $\delta_{2r} < 1$, then the following holds:*

$$\|\boldsymbol{Y} - \boldsymbol{M}_*\| \leq \delta_{2r}\|\boldsymbol{M}_*\|_F \leq \delta_{2r}\sqrt{r}\|\boldsymbol{M}_*\|$$

Proof: Appendix B: Matrix Sensing Spectral Method Proof

Thus, the surrogate matrix $\boldsymbol{Y}$ is bounded in distance to $\boldsymbol{M}_*$. Moreover, many surrogate matrices have been designed for various matrix factorization problems where the sensing operator satisfies RIP condition (EQ 4.1.2). For the case where RIP condition is not satisfied, it turns out that the subspace of a surrogate matrix can still be useful for initialization. Recall that phase retrieval (a specific matrix sensing problem) does not satisfy RIP but it is possible to initialize regardless so this will be used as an example as to why the method can still work. For phase retrieval, the sensing matrix is $\boldsymbol{A}_i = \boldsymbol{a}_i\boldsymbol{a}_i^T$ so let the surrogate matrix be expressed similarly in that $\boldsymbol{Y} = \frac{1}{m}\sum_{i=1}^m (y_i\boldsymbol{a}_i\boldsymbol{a}_i^T)$. For simplicity, assume the measurement matrix is rank-1 $\boldsymbol{M}_* = \boldsymbol{x}_*\boldsymbol{x}_*^T$ such that our initializer is a vector (specifically the first eigenvector of $\boldsymbol{Y}$). The goal now becomes to figure out whether this surrogate is bounded with the final solution $\boldsymbol{x}_*$. The surrogate matrix, under a different interpretation, is really an average of m i.i.d. random rank-1 matrices. When the number of observations is large, using probability theory, the following is achieved:

$$\mathbb{E}[\boldsymbol{Y}] = \mathbb{E}[y_i\boldsymbol{a}_i\boldsymbol{a}_i^T] = 2\boldsymbol{x}_*\boldsymbol{x}_*^T + \|\boldsymbol{x}_*\|_2^2\boldsymbol{I}_n$$

This is fantastic since this tells us that the surrogate matrix $\boldsymbol{Y}$ contains information about $\boldsymbol{x}_*$ on average even though $\boldsymbol{Y}$ does not contain any information about $\boldsymbol{x}$ in the surrogate expression written above. Therefore, $\boldsymbol{x}_0 = \sqrt{\lambda_1/3}\boldsymbol{u}_1$ is a valid initializer since first eigenvector $\boldsymbol{u}_1$ of $\boldsymbol{Y}$ contains information about $\boldsymbol{x}_*$.

5 Global Landscape Algorithms

Previously, the goal was to find a basin of attraction and initialize within it. In the global landscape setting, the entire parameter space is considered and the curvature properties of the cost function are considered. This is done by avoiding two kinds of situations: 1) local minima that are not the same value as the global minima and 2) saddle points that confuse convex algorithms. In order to discuss nonconvex methods, the notion of a benign landscape is introduced to differentiate from badly behaved landscapes.

Definition 5.0.1 (Benign Landscape)**.** *A function $f(\cdot)$ is said to have a benign (meaning harmless) landscape iff all the critical points of $f(\cdot)$ are either global minima or strict saddle points.*

The reason why only strict saddle points are considered (instead of degenerate saddle points) is because they are relatively easy to escape during the descent process with only second order derivative information whereas degenerate saddle points require higher order information. Besides matrix factorization, there are many popular problems where benign landscapes appear. For example, dictionary learning and statistical estimation have a nonconvex but benign landscape. An example of a nonconvex problem that does **not** have a benign landscape would be neural networks with nonlinear activation functions. In the matrix factorization context, the nonconvexity usually comes from symmetry of some kind. For example, matrix sensing operator involves the Frobenious norm (trace operation) which is unitary invariant so we have a "ring" of global minima versus one single minima. To take this further, matrix sensing convergence is guaranteed in a global landscape setting by the following theorem:

Theorem 5.0.2 (Matrix Sensing Global Landscape Theorem)**.** *Consider the matrix sensing operator $\mathcal{A}$ from EQ 4.1.1. If $\mathcal{A}$ satisfies 2r-RIP condition (EQ 4.1.2) with constant $\delta_{2r} < 1/10$, then the global landscape meets the following:*

- *All local minima are global minima*
- *All critical points that are not minima are strict saddle points, specifically for critical point $\boldsymbol{X}$, $\lambda_{min}(\nabla^2 f(\boldsymbol{X})) \leq -4\sigma_r(\boldsymbol{M}_*)/5$*

Proof: Appendix C: Matrix Sensing Benign Landscape Proof

In other words, the matrix sensing problem that satisfies RIP condition (EQ 4.1.2) will have a benign landscape (DF 5.0.1). If matrix sensing has a benign landscape, then the main goal becomes how to escape saddle points efficiently. Let us consider the famous gradient descent method.

5.1 Gradient-based Methods [12]

Before discussing gradient theory, it is important to consider what is known as the strict saddle property. Putting it simply, it states that every point in some function $f(\cdot)$ either has a large gradient, a negative curvature, or a local minimum when there is a benign landscape. Mathematically, the strict saddle property is defined as follows:

Definition 5.1.1 (Strict Saddle Property)**.** *The function $f(\cdot)$ satisfies $(\epsilon, \gamma, \zeta)$-strict saddle property for said constants if one of the following is met for each $\boldsymbol{x}$:*

- *(strong gradient magnitude) $\|f(\boldsymbol{x})\|_2 \geq \epsilon$*
- *(negative curvature) $\lambda_{min}(\nabla^2 f(\boldsymbol{x})) \leq -\gamma$*
- *(local minimum) For local minimizer $\boldsymbol{x}_*$, $\|\boldsymbol{x} - \boldsymbol{x}_*\|_2 \leq \zeta$*

It turns out that gradient descent can converge for functions that meet this condition but only if initialized randomly. If the initialization occurs at a location where $\nabla f(\boldsymbol{x}) = \boldsymbol{0}$ then gradient descent will get stuck and unable to escape. However, if the initialization is random, then it can avoid saddle points during the descent since the probability of this event is zero. This powerful idea is expressed in the following theorem.

Theorem 5.1.2 (GD Convergence - Random Initialization Case)**.** *Assume $f(\cdot)$ (EQ 2.1.1) satisfies the strict saddle property (DF 5.1.1). This implies a twice continuously differentiable function meaning it is β-smooth for some β. If the step size is such that $\alpha_k < 1/\beta$ then GD under random initialization converges (in the almost surely sense) to a local minimizer or $-\infty$.*

Conceptually, GD iterations will not even reach the saddle point (under random initialization) since there is always negative curvature around that area that forces the GD trajectory away from any saddle point. Since phase retrieval does not satisfy RIP condition (EQ 4.1.2) but satisfies the strict saddle property (DF 5.1.1), then convergence is guaranteed under the random initialization setting.

Theorem 5.1.3 (Phase retrieval convergence for randomized GD method)**.** *For a sufficiently large iteration number k, under a random initialization* $\boldsymbol{x}_0 \sim \mathcal{N}(\boldsymbol{0}, \frac{\|\boldsymbol{x}_*\|_2^2}{n}\boldsymbol{I}_n)$*, and step size* $\alpha = \frac{1}{c_1\|\boldsymbol{x}_*\|_2^2}$*, the following bound is achieved (for cost function EQ 2.3.1):*

$$\|\boldsymbol{x}_k - \boldsymbol{x}_*\|_2 \leq (1-c_2)^k\|\boldsymbol{x}_0 - \boldsymbol{x}_*\|_2 \;\; \textit{with a very high probability for some constants } c_1, c_2 > 0$$

This particular GD method is useful for large-scale problems due to fast convergence. Designing other minimization algorithms besides gradient descent is a topic for the next section.

5.2 Hessian-based Methods [13] [14]

Before talking about trust-region methods, the reader should know that both trust-region methods and cubic regularization fall under a broad umbrella of algorithms known as generic saddle-escaping algorithms. To explain what these methods do, it is important to consider the Taylor expansion of any saddle point. Given a saddle point $\boldsymbol{x}$, the following Taylor expansion

$$f(\boldsymbol{x}+\boldsymbol{\epsilon}) \approx f(\boldsymbol{x}) + \frac{1}{2}\boldsymbol{\epsilon}^T\nabla^2 f(\boldsymbol{x})\boldsymbol{\epsilon}$$

shows that $f(\boldsymbol{x}+\boldsymbol{\epsilon}) < f(\boldsymbol{x})$ since the $\nabla^2 f(\boldsymbol{x})$ term will be negative. This is remarkable since it is possible to identify a direction where the curvature is negative and continue to decrease the objective function (therefore escape saddle points). One way to use this property is to use an algorithm that involves the Hessian $\nabla^2 f(\boldsymbol{x})$ which is the basis for trust-region methods. The trust-region method is written as follows for some $\boldsymbol{x}$:

$$\boldsymbol{x}_{k+1} = \min_{\boldsymbol{z}:\|\boldsymbol{z}-\boldsymbol{x}_k\|\leq\Delta} \{\langle\nabla f(\boldsymbol{x}_k), \boldsymbol{z}-\boldsymbol{x}_k\rangle + \frac{1}{2}\langle\nabla^2 f(\boldsymbol{x}_k)(\boldsymbol{z}-\boldsymbol{x}_k), \boldsymbol{z}-\boldsymbol{x}_k\rangle\}$$

where Δ is a user chosen radius known as the "trust region". From the Taylor expansion written above, we can expect a reasonable approximation only if the trust region is small. If the current iterate happens to be the saddle point then the method above will move in the direction of the negative curvature since $\nabla^2 f(\boldsymbol{x})(\boldsymbol{z}-\boldsymbol{x})$ is negative. Thus, the method will escape a saddle point more efficiently than GD would if the current iterate happens to meet the condition. On the other hand, cubic regularization explicitly adds a cubic term to the cost function so that the minimization problem becomes unconstrained with respect to $\boldsymbol{z}$. This cubic regularized minimization expression can be written as follows:

$$\boldsymbol{x}_{k+1} = \min_{\boldsymbol{z}} \{\langle\nabla f(\boldsymbol{x}_k), \boldsymbol{z}-\boldsymbol{x}_k\rangle + \frac{1}{2}\langle\nabla^2 f(\boldsymbol{x}_k)(\boldsymbol{z}-\boldsymbol{x}_k), \boldsymbol{z}-\boldsymbol{x}_k\rangle + \frac{L_2}{6}\|\boldsymbol{z}-\boldsymbol{x}_k\|_2^3\}$$

where L_2 corresponds to the Lipschitz constant of the Hessian of the cost function. In this case, the Lipschitz constant is selected since it will allow the new cubic-regularized function to *majorize* the objective function $f(\boldsymbol{x})$. The majorization definition is written below.

Definition 5.2.1 (Majorization)**.** *For vectors* $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^d$*,* $\boldsymbol{a}$ *is said to majorize b iff* $\Sigma_{i=1}^k \boldsymbol{a}_i \geq \Sigma_{i=1}^k \boldsymbol{b}_i$ *for* $k = 1, ..., d$*.*

Without diving into majorization theory, functions that preserve the ordering of majorization (e.g. this cubic-regularized function) are Schur-convex functions. While a Schur-convex function does not imply a convex function, gradient descent can be used on Schur-convex functions to efficiently solve the minimization problem.

5.3 Almost-Convex-AGD & Negative-Curvature-Descent [15]

Both Almost-Convex-AGD and Negative-Curvature-Descent make up one complete algorithm that consists of the two subroutines. Almost-Convex-AGD uses Nesterov's accelerated gradient method to minimize an almost convex function (hence the name) that combines the objective function with a distance term. Negative-Curvature-Descent will find a direction that decreases the function value by calculating the smallest eigenvector of the Hessian and thus move in the negative curvature direction and escape saddle points. The overall scheme of the algorithm is that for each iterate, it will first calculate the smallest eigenvalue for the Hessian and decide whether to move in the direction computed by Negative-Curvature-Descent (to avoid saddle points) or either to apply Almost-Convex-AGD to optimize an almost convex function. Almost-Convex-AGD is exactly the same as Nesterov's method except the cost function becomes $f(\boldsymbol{z}) + \gamma\|\boldsymbol{z} - \boldsymbol{z}_i\|^2$ where $\gamma < L_1$ (Lipschitz constant of the gradient) and accelerated GD is performed on the modified ("convex") cost function. This is done by coming up with a momentum term $\boldsymbol{p}_{k+1} = \boldsymbol{z}_k + \beta_k(\boldsymbol{z} - \boldsymbol{z}_{k-1})$ and using that in vanilla GD such that $\boldsymbol{z}_{k+1} = \boldsymbol{p}_{k+1} - \tau\nabla f(\boldsymbol{p}_{k+1})$ where β is the momentum weight and τ is the step size. For the other subroutine, the goal of Negative-Curvature-Descent is to find a vector $\boldsymbol{v}_j$ where $\|\boldsymbol{v}_j\| \leq 1$ and meets the following condition for current iterate $\boldsymbol{x}_k$:

$$\lambda_{\min}(\nabla^2 f(\boldsymbol{x}_k)) \geq \boldsymbol{v}_j^T \nabla^2 f(\boldsymbol{x}_k)\boldsymbol{v}_j$$

Then if $\boldsymbol{v}_j^T \nabla^2 f(\boldsymbol{x}_k)\boldsymbol{v}_j < 0$ we will either take a step in the negative direction computed (i.e. $\boldsymbol{v}_j$) of size proportional to $\boldsymbol{v}_j^T \nabla^2 f(\boldsymbol{x}_k)\boldsymbol{v}_j$ or otherwise retain current iterate $\boldsymbol{x}_k$. In other words, it will stay at the current iterate if a negative descent direction cannot be calculated (since it is away from saddle points) or descend based on negative curvature if near a saddle point. Specifically, the new iterate is written as follows:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \frac{2|\boldsymbol{v}_j^T \nabla^2 f(\boldsymbol{x}_k)\boldsymbol{v}_j|}{L_2}\text{sign}(\boldsymbol{v}_j^T \nabla f(\boldsymbol{x}_k))\boldsymbol{v}_j$$

where L_2 represents the Lipschitz constant of the Hessian. It is important to note that, unlike trust region method and cubic regularization, this method does not use the Hessian directly. Instead, it approximates it by calculating the following:

$$\nabla^2 f(\boldsymbol{x})\boldsymbol{v} = \lim_{h \to 0} \frac{\nabla f(\boldsymbol{x} + h\boldsymbol{v}) - \nabla f(\boldsymbol{x})}{h}$$

6 Conclusion

In this paper, nonconvex optimization algorithms are discussed within the context of matrix factorization problems. Canonical problems like matrix sensing, phase retrieval, and robust PCA were introduced to show a representative sample of the world of matrix factorization. For the sake of simplicity and consistency, special attention was placed to matrix sensing for each topic. Thereafter, two-stage algorithms and global landscape algorithms were discussed that take advantage of two key ideas: basin of attraction and benign landscape. Specifically, for two-stage algorithms (basin of attraction), gradient descent method and alternating minimization methods are useful in practice so long as the initialization process is carefully designed via the spectral method. Moreover, for global landscape algorithms (benign landscape), gradient methods were discussed for strict saddle functions and generic saddle-escaping algorithms such as trust-region methods and AGD-NCD. Throughout these sections, theorems are also included for theoretical guarantees under a matrix factorization setting. Nonconvex methods are extremely important for a wide range of problems as shown in this paper. However, some attention should be placed on whether it is optimal for the specific problem. It may be that a convex relaxation approach would be more appropriate instead of a nonconvex formulation depending on the situation. Each specific problem has different conditions that determine whether or not a nonconvex approach would be ideal so individual analysis is required before applying any such methods discussed.

References

[1] Y. Chi, Y. M. Lu, and Y. Chen, "Nonconvex optimization meets low-rank matrix factorization: An overview," *CoRR*, vol. abs/1809.09573, 2018.

[2] J. Wright and Y. Ma, *High-Dimensional Data Analysis with Low-Dimensional Models: Principles, Computation, and Applications*. Cambridge University Press, 2022.

[3] E. J. Candès and Y. Plan, "Tight oracle bounds for low-rank matrix recovery from a minimal number of random measurements," *CoRR*, vol. abs/1001.0339, 2010.

[4] N. Vaswani, T. Bouwmans, S. Javed, and P. Narayanamurthy, "Robust PCA and robust subspace tracking," *CoRR*, vol. abs/1711.09492, 2017.

[5] Y. Shechtman, Y. C. Eldar, O. Cohen, H. N. Chapman, J. Miao, and M. Segev, "Phase retrieval with application to optical imaging," *CoRR*, vol. abs/1402.7350, 2014.

[6] Y. Chen and E. J. Candès, "Solving random quadratic systems of equations is nearly as easy as solving linear systems," *CoRR*, vol. abs/1505.05114, 2015.

[7] S. Tu, R. Boczar, M. Simchowitz, M. Soltanolkotabi, and B. Recht, "Low-rank solutions of linear matrix equations via procrustes flow," in *International Conference on Machine Learning*, pp. 964–973, PMLR, 2016.

[8] R. Sun and Z.-Q. Luo, "Guaranteed matrix completion via non-convex factorization," *IEEE Transactions on Information Theory*, vol. 62, no. 11, pp. 6535–6579, 2016.

[9] P. Jain, P. Netrapalli, and S. Sanghavi, "Low-rank matrix completion using alternating minimization," in *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pp. 665–674, 2013.

[10] P. Jain, R. Meka, and I. Dhillon, "Guaranteed rank minimization via singular value projection," *Advances in Neural Information Processing Systems*, vol. 23, 2010.

[11] C. Davis and W. M. Kahan, "The rotation of eigenvectors by a perturbation. iii," *SIAM Journal on Numerical Analysis*, vol. 7, no. 1, pp. 1–46, 1970.

[12] J. Sun, *When are nonconvex optimization problems not scary?* Columbia University, 2016.

[13] A. R. Conn, N. I. Gould, and P. L. Toint, *Trust region methods*. SIAM, 2000.

[14] Y. Nesterov and B. T. Polyak, "Cubic regularization of newton method and its global performance," *Mathematical Programming*, vol. 108, no. 1, pp. 177–205, 2006.

[15] Y. Carmon, J. C. Duchi, O. Hinder, and A. Sidford, "Accelerated methods for nonconvex optimization," *SIAM Journal on Optimization*, vol. 28, no. 2, pp. 1751–1772, 2018.

[16] S. Bhojanapalli, B. Neyshabur, and N. Srebro, "Global optimality of local search for low rank matrix recovery," *Advances in Neural Information Processing Systems*, vol. 29, 2016.

7 Appendix A: Matrix Sensing Gradient Method Proof [1]

Theorem 7.0.1 (Matrix sensing (rank-r) convergence of GD)**.** *The matrix sensing problem (EQ 2.1.2) must have the $\mathcal{A}(\boldsymbol{X})$ operator (EQ 4.1.1) satisfy the 6r-RIP condition (EQ 4.1.2) with constant $\delta_{6r} \leq \frac{1}{10}$. Then, $\exists c_0, c_1 > 0$ such that if dist$(\boldsymbol{X_0}, \boldsymbol{X}_*) \leq \frac{\sigma_r(\boldsymbol{M}_*)}{16}$ then with $\alpha_k = \frac{c_0}{\sigma_1(\boldsymbol{M}_*)}$ obeys convergence:*

$$dist(\boldsymbol{X}_k, \boldsymbol{X}_*) \leq (1 - \frac{c_1}{\kappa})^k dist^2(\boldsymbol{X}_0, \boldsymbol{X_*}) \; \forall k = 0, 1, ...$$

Proof (rank-1 case):
For arbitrary vectors $\boldsymbol{x}$ and $\boldsymbol{\Delta} \in \mathbb{R}^n$ and $\boldsymbol{D} = \boldsymbol{x}\boldsymbol{x}^T - \boldsymbol{x_*}\boldsymbol{x_*}^T$ we have that the quadratic form of matrix sensing cost function stated as:

$$\boldsymbol{\Delta}^T \nabla^2 f(\boldsymbol{x}) \boldsymbol{\Delta} = \frac{1}{m} \sum_{i=1}^{m} \langle \boldsymbol{A}_i, \boldsymbol{D} \rangle (\boldsymbol{\Delta}^T \boldsymbol{A}_i \boldsymbol{\Delta}) + 2(\boldsymbol{\Delta}^T \boldsymbol{A}_i \boldsymbol{x})^2$$

This is a lemma from [16]. Introducing the sensing operator $\mathcal{A}(\cdot)$, the expression becomes the following:

$$\boldsymbol{\Delta}^T \nabla^2 f(\boldsymbol{x}) \boldsymbol{\Delta} = \langle \mathcal{A}(\boldsymbol{D}), \mathcal{A}(\boldsymbol{\Delta}\boldsymbol{\Delta}^T) \rangle + \frac{1}{2} \langle \mathcal{A}(\boldsymbol{\Delta}\boldsymbol{x}^T + \boldsymbol{x}\boldsymbol{\Delta}^T), \mathcal{A}(\boldsymbol{\Delta}\boldsymbol{x}^T + \boldsymbol{x}\boldsymbol{\Delta}^T) \rangle$$

The goal is to show that

$$\boldsymbol{\Delta}^T \nabla^2 f(\boldsymbol{x}) \boldsymbol{\Delta} \approx \langle \boldsymbol{D}, \boldsymbol{\Delta}\boldsymbol{\Delta}^T \rangle + \frac{1}{2} \| \boldsymbol{\Delta}\boldsymbol{x}^T + \boldsymbol{x}\boldsymbol{\Delta}^T \|_F^2 = g(\boldsymbol{x}, \boldsymbol{\Delta})$$

The reason is that $g(\cdot)$ has local strong convexity and is smooth. This means that convergence is guaranteed. This is expressed by the following lemma.
Lemma 7.0.2 (GD Convergence (Strong Convexity & Smoothness))**.** *Suppose that function $f(\cdot)$ is α-strongly convex and β-smooth within an L2 sphere $\mathcal{B}[\boldsymbol{x}_*, \zeta]$. If $\boldsymbol{x}_0 \in \mathcal{B}[\cdot]$ and $\tau = \frac{1}{\beta}$, then the following is achieved:*

$$\| \boldsymbol{x}_k - \boldsymbol{x}_* \|_2 \leq (1 - \frac{\alpha}{\beta})^k \| \boldsymbol{x}_0 - \boldsymbol{x}_* \|_2 \;\; \forall k = 0, 1, ...$$

Going back to the problem at hand, when $\mathcal{A}(\cdot)$ satisfies 4-RIP (EQ 4.1.2), then by definition, we have that

$$|\langle \mathcal{A}(\boldsymbol{D}), \mathcal{A}(\boldsymbol{\Delta}\boldsymbol{\Delta}^T) \rangle - \langle \boldsymbol{D}, \boldsymbol{\Delta}\boldsymbol{\Delta}^T \rangle| \leq \delta \|\boldsymbol{D}\|_F \|\boldsymbol{\Delta}\boldsymbol{\Delta}^T\|_F \leq 3\delta \|\boldsymbol{x}_*\|_2^2 \|\boldsymbol{\Delta}\|_2^2 \text{ assuming } \|\boldsymbol{x} - \boldsymbol{x}_*\|_2 \leq \|\boldsymbol{x}_*\|_2$$

Likewise, applying the same logic, we can get another bound:

$$|\langle \mathcal{A}(\boldsymbol{\Delta}\boldsymbol{x}^T + \boldsymbol{x}\boldsymbol{\Delta}^T), \mathcal{A}(\boldsymbol{\Delta}\boldsymbol{x}^T + \boldsymbol{x}\boldsymbol{\Delta}^T) \rangle - \|\boldsymbol{\Delta}\boldsymbol{x}^T + \boldsymbol{x}\boldsymbol{\Delta}^T\|_F^2| \leq 4\delta \|\boldsymbol{x}\|_2^2 \|\boldsymbol{\Delta}\|_2^2 \leq 16\delta \|\boldsymbol{x}_*\|_2^2 \|\boldsymbol{\Delta}\|_2^2$$

So long as $\delta \leq \frac{1}{44}$, then with some norm inequality properties, it is possible to get the result:

$$\frac{1}{4} \|\boldsymbol{\Delta}\|_2^2 \|\boldsymbol{x}_*\|_2^2 \leq \boldsymbol{\Delta}^T \nabla^2 f(\boldsymbol{x}) \boldsymbol{\Delta} \leq 3 \|\boldsymbol{\Delta}\|_2^2 \|\boldsymbol{x}_*\|_2^2 \;\; \forall \boldsymbol{\Delta}$$

Thus, we have bounds on (α, β) the smoothness and convexity parameters. Because of this, through the lemma stated above, matrix sensing is guaranteed to converge using gradient descent method so long as the sensing operator satisfies the RIP condition (EQ 4.1.2).

8 Appendix B: Matrix Sensing Spectral Method Proof [1]

Lemma 8.0.1 (Matrix Sensing Spectral Initialization)**.** *Given that $\boldsymbol{M}_*$ is rank-r matrix and $\mathcal{A}$ (EQ 4.1.1) satisfies 2r-RIP condition (EQ 4.1.2) with constant $\delta_{2r} < 1$, then the following holds for surrogate matrix $\boldsymbol{Y}$:*

$$\|\boldsymbol{Y} - \boldsymbol{M}_*\| \leq \delta_{2r}\|\boldsymbol{M}_*\|_F \leq \delta_{2r}\sqrt{r}\|\boldsymbol{M}_*\|_F$$

Proof:
For arbitrary vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ we have that,

$$\begin{aligned}
&\boldsymbol{x}^T(\boldsymbol{M}_* - \boldsymbol{Y})\boldsymbol{y} \\
&= \boldsymbol{x}^T(\boldsymbol{M}_* - \mathcal{A}^*\mathcal{A}(\boldsymbol{M}_*))\boldsymbol{y} && \text{(per definition of the surrogate matrix)} \\
&= \langle \boldsymbol{x}\boldsymbol{y}^T, \boldsymbol{M}_* - \mathcal{A}^*\mathcal{A}(\boldsymbol{M}_*)\rangle && \text{(per inner product definition)} \\
&= \langle \boldsymbol{x}\boldsymbol{y}^T, \boldsymbol{M}_*\rangle - \langle \mathcal{A}(\boldsymbol{x}\boldsymbol{y}^T), \mathcal{A}(\boldsymbol{M}_*)\rangle && \text{(per inner product properties)} \\
&\leq \delta_2 r(\|\boldsymbol{x}\|_2 \cdot \|\boldsymbol{y}\|_2)\|\boldsymbol{M}_*\|_F && \text{(per 2r-RIP property definition)} \\
&\implies \|\boldsymbol{M}_* - \boldsymbol{Y}\| = \max_{\|\boldsymbol{x}\|_2=\|\boldsymbol{y}\|_2=1} \boldsymbol{x}^T(\boldsymbol{M}_* - \boldsymbol{Y})\boldsymbol{y} && \text{(spectral norm definition)} \\
&\leq \delta_2 r\|\boldsymbol{M}_*\|_F \leq \delta_2 r\sqrt{r}\|\boldsymbol{M}_*\| && \text{(using above bound)} \\
&\implies \|\boldsymbol{Y} - \boldsymbol{M}_*\| \leq \delta_{2r}\|\boldsymbol{M}_*\|_F \leq \delta_{2r}\sqrt{r}\|\boldsymbol{M}_*\| && \square
\end{aligned}$$

Thus, the surrogate matrix $\boldsymbol{Y}$ is bounded to the minimizer $\boldsymbol{M}_*$ by some distance proportional to $\|\boldsymbol{M}_*\|_F$. Because of this, the surrogate matrix is a valid initialization for any two-stage algorithm.

9 Appendix C: Matrix Sensing Benign Landscape Proof [1]

Theorem 9.0.1 (Matrix Sensing Global Landscape Theorem)**.** *Consider the matrix sensing operator $\mathcal{A}$ (EQ 4.1.1). If $\mathcal{A}$ satisfies 2r-RIP condition (EQ 4.1.2) with constant $\delta_{2r} < 1/10$, then the global landscape is benign (DF 5.0.1) and meets the following conditions for cost function EQ 2.1.1:*

- *All local minima are global minima*
- *All critical points that are not minima are strict saddle points, specifically for critical point $\boldsymbol{X}$, $\lambda_{min}(\nabla^2 f(\boldsymbol{X})) \leq -4\sigma_r(\boldsymbol{M}_*)/5$*

Proof:
For rank-1 case with arbitrary vector $\boldsymbol{x}$ that is a critical point for the matrix sensing problem we have that,

$$\nabla f(\boldsymbol{x}) = \frac{1}{m}\sum_{i=1}^{m}\langle \boldsymbol{A}_i, \boldsymbol{x}\boldsymbol{x}^T - \boldsymbol{x}_*\boldsymbol{x}_*^T\rangle \boldsymbol{A}_i\boldsymbol{x} = \boldsymbol{0} \qquad \text{(first order optimality condition)}$$

$$\implies \|(\boldsymbol{x}\boldsymbol{x}^T - \boldsymbol{x}_*\boldsymbol{x}_*^T)\boldsymbol{x}\boldsymbol{x}^T\|_F \leq \delta\|\boldsymbol{x}\boldsymbol{x}^T - \boldsymbol{x}_*\boldsymbol{x}_*^T\|_F \cdot \|\boldsymbol{x}\|_2^2 \qquad \text{(direct consequence from above)}$$

$$\nabla^2 f(\boldsymbol{x}) = \frac{1}{m}\sum_{i=1}^{m} 2\boldsymbol{A}_i\boldsymbol{x}\boldsymbol{x}^T\boldsymbol{A}_i + \langle \boldsymbol{A}_i, \boldsymbol{x}\boldsymbol{x}^T - \boldsymbol{x}_*\boldsymbol{x}_*^T\rangle \boldsymbol{A}_i \succeq \boldsymbol{0} \qquad \text{(second order optimality condition)}$$

Let $\boldsymbol{\Delta} = \boldsymbol{x} - \boldsymbol{x}_*$ where $\boldsymbol{x}_*$ is the minimizer. Then,

$$\boldsymbol{\Delta}^T\nabla^2 f(\boldsymbol{x})\boldsymbol{\Delta} \qquad \text{(quadratic form of function)}$$

$$= \frac{1}{m}\sum_{i=1}^{m} 2\langle \boldsymbol{A}_i, \boldsymbol{x}\boldsymbol{\Delta}^T\rangle^2 + \langle \boldsymbol{A}_i, \boldsymbol{x}\boldsymbol{x}^T - \boldsymbol{x}_*\boldsymbol{x}_*^T\rangle\langle \boldsymbol{A}_i, \boldsymbol{\Delta}\boldsymbol{\Delta}^T\rangle$$

$$= \frac{1}{m}\sum_{i=1}^{m} 2\langle \boldsymbol{A}_i, \boldsymbol{x}\boldsymbol{\Delta}^T\rangle^2 - \langle \boldsymbol{A}_i, \boldsymbol{x}\boldsymbol{x}^T - \boldsymbol{x}_*\boldsymbol{x}_*^T\rangle^2$$

$$\implies \boldsymbol{\Delta}^T\nabla^2 f(\boldsymbol{x})\boldsymbol{\Delta} \leq 2(1+\delta)\|\boldsymbol{x}\boldsymbol{\Delta}^T\|_F^2 - (1-\delta)\|\boldsymbol{D}\|_F^2 \qquad \text{(using 2r-RIP condition)}$$

Where $\boldsymbol{D} = \boldsymbol{x}\boldsymbol{x}^T - \boldsymbol{x}_*\boldsymbol{x}_*^T$. Without proof, the bound is stated:

$$\|\boldsymbol{x}\boldsymbol{\Delta}^T\|_F^2 \leq \frac{1}{8}\|\boldsymbol{D}\|_F^2 + \frac{34}{8\|\boldsymbol{x}\|_2^2}\|\boldsymbol{D}\boldsymbol{x}\boldsymbol{x}^T\|_F^2 \qquad \text{(Lemma from [16])}$$

$$\implies \boldsymbol{\Delta}^T\nabla^2 f(\boldsymbol{x})\boldsymbol{\Delta}$$

$$\leq \frac{5\delta - 3}{4}\|\boldsymbol{D}\|_F^2 + \frac{34 + 34\delta}{4\|\boldsymbol{x}\|_2^2}\|\boldsymbol{D}\boldsymbol{x}\boldsymbol{x}^T\|_F^2 \qquad \text{(using bound above on quadratic form)}$$

$$\leq -\gamma\|\boldsymbol{D}\|_F^2 \text{ where } 0 < \gamma < 1 \qquad \text{(using first order optimality condition)}$$

$$\implies \boldsymbol{\Delta}^T\nabla^2 f(\boldsymbol{x})\boldsymbol{\Delta} < 0 \;\; \forall \boldsymbol{\Delta} \text{ unless } \boldsymbol{\Delta} = \boldsymbol{0}$$

$$\boldsymbol{\Delta} = \boldsymbol{0} \implies \boldsymbol{x} = \boldsymbol{x}_* \qquad \square$$

Therefore, either $\boldsymbol{x}$ (a critical point) is the minimizer $\boldsymbol{x}_*$ or it is a strict saddle point since the quadratic form is negative as shown above.