Journal of Machine Learning Research 21 (2020) 1-43 Submitted 6/19; Revised 10/20; Published 11/20

Fast Exact Matrix Completion: A Unified Optimization Framework for Matrix Completion

Dimitris Bertsimas DBERTSIM@MIT.EDU
Sloan School of Management and Operations Research Center
Massachusetts Institute of Technology
Cambridge, MA 02139, USA

Michael Lingzhi Li MLLI@MIT.EDU
Operations Research Center
Massachusetts Institute of Technology
Cambridge, MA 02139, USA

Editor: Tong Zhang

Abstract

We formulate the problem of matrix completion with and without side information as a non-convex optimization problem. We design fastImpute based on non-convex gradient descent and show it converges to a global minimum that is guaranteed to recover closely the underlying matrix while it scales to matrices of sizes beyond $10^5 \times 10^5$. We report experiments on both synthetic and real-world datasets that show fastImpute is competitive in both the accuracy of the matrix recovered and the time needed across all cases. Furthermore, when a high number of entries are missing, fastImpute is over 75% lower in MAPE and 15 times faster than current state-of-the-art matrix completion methods in both the case with side information and without.

Keywords: Matrix Completion, Projected Gradient Descent, Stochastic Approximation

1. Introduction

Low-rank matrix completion is one of the most studied problems after its successful application in the Netflix Competition. It has been used in computer vision (Candes and Plan (2010)), signal processing (Ji et al. (2010)), and control theory (Boyd et al. (1994)) to generate a completed matrix from partially observed entries, among several other areas. Given a data matrix $\boldsymbol{A} \in \mathbb{R}^{n \times m}$, the low-rank assumption assumes that rank($\boldsymbol{A}$) is small - in other words there are only a few, but still unknown, common linear factors that affect A_{ij}.

In recent years, as noted by Nazarov et al. (2018), there has been a rise in interest for *inductive matrix completion*, where the common linear factors are chosen from a set of given vectors in the form of side information.

In this paper, we present an optimization based approach that improves upon the state of the art in matrix completion with and without side information. We next review the literature in both the general matrix completion and the inductive matrix completion areas.

Literature

General Matrix Completion

Matrix completion has been applied successfully for many tasks, including recommender systems Koren et al. (2009), social network analysis Chiang et al. (2014) and clustering Chen et al. (2014b). After Candès and Recht (2009) first proved a theoretical guarantee for the retrieval of the exact matrix under the nuclear norm convex relaxation, a lot of methods have focused on the nuclear norm problem (see Mazumder et al. (2010), Beck and Teboulle (2009), Jain et al. (2010), and Tanner and Wei (2013) for examples). Alternative methods include alternating projections by Recht and Ré (2013) and Grassmann manifold optimization by Keshavan et al. (2009). There has also been work where the uniform distributional assumptions required by the theoretical guarantees are violated, such as Negahban and Wainwright (2012) and Chen et al. (2014a).

Despite the non-convexity of the problem, many gradient-descent based approaches have also been proposed. Many works, including Koren et al. (2009), Jain and Netrapalli (2015), Zheng and Lafferty (2016) and Jin et al. (2016), utilize the Burer-Monteiro factorization ($\boldsymbol{A} = \boldsymbol{U}\boldsymbol{V}^T$, where $\boldsymbol{U} \in \mathbb{R}^{n\times k}$ and $\boldsymbol{V} \in \mathbb{R}^{m\times k}$) and conduct various forms (projected, stochastic, lifted, etc) of gradient descent on $\boldsymbol{U}$ and $\boldsymbol{V}$. Numerical experiments suggest that such algorithms can often converge to the global optimal solution despite their local nature.

Recently, a line of work, including Chen and Wainwright (2015), Ge et al. (2016), and Ma et al. (2019), investigated the reasons behind this uncanny efficiency of gradient descent on matrix completion. Collectively, they showed that for the case where $\boldsymbol{A}$ is positive semidefinite (PSD), gradient descent on the symmetric factorization $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{U}^T$ can converge linearly to the global optimum under restricted isometry conditions.

This work has also been reproduced in the non-PSD case considered here ($\boldsymbol{A} = \boldsymbol{U}\boldsymbol{V}^T$) if special regularization terms are added (see for example Sun and Luo (2016)).

Our work differs from previous work in one significant way: After writing $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{V}^T$, instead of performing gradient descent on both $\boldsymbol{U}$ and $\boldsymbol{V}$, we derive $\boldsymbol{U}$ as a function of $\boldsymbol{V}$, $g(\boldsymbol{V})$, and then directly perform gradient updates for $\boldsymbol{V}$, with a projection to a fixed-norm hypersphere $\|\boldsymbol{V}\|_2 = 1$ to ensure scaling invariance.

Using this formulation, we are able to prove convergence guarantees without special regularization terms on the objective. Numerical experiments show that we are able to retrieve the matrix faster than the top performing methods discussed here.

Inductive Matrix Completion

Interest in inductive matrix completion intensified after Xu et al. (2013) showed that given predictive side information, one only needs $O(\log n)$ samples to retrieve the full matrix. Thus, most of this work (see Xu et al. (2013), Jain and Dhillon (2013), Farhat et al. (2013), Natarajan and Dhillon (2014)) have focused on the case in which the side information is assumed to be perfectly predictive so that the theoretical bound of $O(\log n)$ sample complexity Xu et al. (2013) can be achieved. Chiang et al. (2015) explored the case in which the side information is corrupted with noise, while Shah et al. (2017) and Si et al. (2016) incorporated nonlinear combination of factors into the side information. As pointed out by a recent article Nazarov et al. (2018), there is a considerable lack of effort to introduce sparsity into inductive matrix completion, with Lu et al. (2016), Soni et al. (2016), Nazarov et al. (2018) as examples. Bertsimas and Li (2018) introduced a convex binary formulation of the sparse inductive matrix completion problem, and constructed randomized algorithms for scaling.

Our work differs from the work in Lu et al. (2016), Soni et al. (2016), and Nazarov et al. (2018) as it does not consider the heuristic convex relaxation of sparsity in the nuclear norm, but rather the exact sparse problem. Bertsimas and Li (2018) consider the problem where the low-rank factorization needs to be selected from features in the side information, while we allow the factorization to be any linear combination of the features in the side information. This greatly increases the flexibility of the algorithm, provides higher modeling power, and leads to stronger matrix recovery.

Contributions and Structure

Our contributions in this paper are as follows:

1. We reformulate the low-rank matrix completion problem, both with side information and without, as a separable optimization problem. We show that the general matrix completion problem is in fact a special case of the matrix completion problem with side information.

2. We propose a novel algorithm using projected gradient descent and Nesterov's accelerated gradient to solve the reformulated matrix completion problem. We prove that the algorithm is guaranteed to converge to the optimal solution under mild conditions.

3. We present computational results on both synthetic and real-world datasets that shows the algorithm outperforms current state-of-the-art methods in scalability and accuracy with and without side information. In particular, for the general matrix completion

problem, we show the algorithm is about 15 times faster than the fastest algorithm available, while achieving on average a 75% decrease in error of retrieval on synthetic datasets.

The structure of the paper is as follows. In Section 2, we introduce the separable reformulation of the low-rank matrix completion problem. In Section 3, we introduce the base projected gradient descent method, projImpute. In Section 4, we introduce fastImpute, the stochastic version of projImpute designed for use with side information. We report on its computational complexity, derive theoretical results, and compare it with other algorithms. in Section 5, we report results on both synthetic and real-world datasets with side information using fastImpute and compare its performance with other algorithms. In Section 6, we show how the calculations can be further simplified when there is no side information. We report results on both synthetic and real-world datasets without side information using fastImpute and compare its performance with other algorithms. In Section 7, we provide our conclusions.

2. Reformulation of Matrix Completion

The classical matrix completion problem considers a matrix $\boldsymbol{A} \in \mathbb{R}^{n \times m}$ in which $\Omega = \{(i,j) \mid A_{ij} \text{ is known}\}$ is the set of known values. We aim to recover a matrix $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{V}^T$ of rank k that minimizes the distance between $\boldsymbol{X}$ and $\boldsymbol{A}$ on the known entries $\boldsymbol{A}$:

$$\min_{\boldsymbol{X}} \frac{1}{nm} \sum_{(i,j)\in\Omega} (X_{ij} - A_{ij})^2 \quad \text{subject to} \quad \text{Rank}(\boldsymbol{X}) = k.$$

The problem we consider here is that for every column $j = 1, \ldots, m$, we have a given p-dimensional feature vector $\boldsymbol{B}_j$ with $p \geq k$ that contains the information we have on column j. In the Netflix example, column j corresponds to movie j, and thus the feature vector $\boldsymbol{B}_j$ includes information about the jth movie: Budget, Box Office revenue, IMDB rating, etc. We represent all this side information with a matrix $\boldsymbol{B} \in \mathbb{R}^{m \times p}$. Given side data $\boldsymbol{B}$ we postulate that $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{S}^T\boldsymbol{B}^T$, where $\boldsymbol{U} \in \mathbb{R}^{n \times k}$ is the matrix of feature exposures, and

$$\boldsymbol{S} = \begin{pmatrix} s_{11} & s_{12} & \cdots & s_{1k} \\ s_{21} & s_{22} & \cdots & s_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ s_{p1} & s_{p2} & \cdots & s_{pk} \end{pmatrix} \in \mathbb{R}^{p \times k}.$$

Then, the matrix completion problem with side data $\boldsymbol{B}$ can be written as:

$$\min_{\boldsymbol{S}} \ \min_{\boldsymbol{U}} \frac{1}{nm} \sum_{(i,j)\in\Omega} (X_{ij} - A_{ij})^2 \quad \text{subject to} \quad \boldsymbol{X} = \boldsymbol{U}\boldsymbol{S}^T\boldsymbol{B}^T, \ \|\boldsymbol{S}\|_2 = 1.$$

The norm constraint on $\boldsymbol{S}$ reduces the space of feasible solutions without loss of generality, as the problem is invariant under the transformation $\boldsymbol{U} \to \boldsymbol{U}\boldsymbol{D}^{-1}$ and $\boldsymbol{S}^T \to \boldsymbol{D}\boldsymbol{S}^T$ for any diagonal matrix $\boldsymbol{D} \in \mathbb{R}^{k\times k}$ that is invertible. Throughout the paper $\|\cdot\|_2$ is the Frobenius norm. We note that since $\boldsymbol{S}$ is a $p \times k$ matrix, the rank of matrix $\boldsymbol{X}$ is indeed k. Further,we note that if $p = m$, and $\boldsymbol{B} = \boldsymbol{I}_m$, then the problem is reduced back to the general low-rank matrix completion problem with no feature information.

Similar to linear regression and for robustness purposes as shown in Bertsimas and Copenhaver (2018), we address in this paper the problem with a Tikhonov regularization term. Specifically, the matrix completion problem with side information and regularization we address is

$$\min_{\|\boldsymbol{S}\|_2=1} \ \min_{\boldsymbol{U}} \frac{1}{nm}\left(\sum_{(i,j)\in\Omega}(X_{ij}-A_{ij})^2 + \frac{1}{\gamma}\|\boldsymbol{U}\|_2^2\right) \quad \text{subject to} \quad \boldsymbol{X} = \boldsymbol{U}\boldsymbol{S}^T\boldsymbol{B}^T, \tag{1}$$

where $\gamma > 0$ is a given parameter that controls the strength of the regularization term. We can reformulate Problem (1) as followed:

Proposition 1 *Problem (1) can be reformulated as a separable optimization problem:*

$$\min_{\|\boldsymbol{S}\|_2=1} c(\boldsymbol{S}) = \frac{1}{nm}\sum_{i=1}^{n} \overline{\boldsymbol{a}}_i \left(\boldsymbol{I}_m - \boldsymbol{V}\left(\frac{\boldsymbol{I}_k}{\gamma} + \boldsymbol{V}^T\boldsymbol{W}_i\boldsymbol{V}\right)^{-1}\boldsymbol{V}^T\right)\overline{\boldsymbol{a}}_i^T, \tag{2}$$

where $\boldsymbol{V} = \boldsymbol{B}\boldsymbol{S}$, $\boldsymbol{W}_1, \cdots, \boldsymbol{W}_n \in \mathbb{R}^{m\times m}$ are diagonal matrices:

$$(\boldsymbol{W}_i)_{jj} = \begin{cases} 1, & (i,j) \in \Omega, \\ 0, & (i,j) \notin \Omega, \end{cases}$$

and $\overline{\boldsymbol{a}}_i = \boldsymbol{a}_i\boldsymbol{W}_i$, $i = 1\dots, n$, so that $\overline{\boldsymbol{a}}_i \in \mathbb{R}^{1\times m}$ is the ith row of $\boldsymbol{A}$ with unknown entries taken to be 0.

Proof With the diagonal projection matrices $\boldsymbol{W}_i$ defined above, we can rewrite the sum in (1) over known entries of $\boldsymbol{A}$, $\sum_{(i,j)\in\Omega}(X_{ij}-A_{ij})^2$, as a sum over the rows of $\boldsymbol{A}$:

$$\sum_{i=1}^{n} \|(\boldsymbol{x}_i - \boldsymbol{a}_i)\boldsymbol{W}_i\|_2^2,$$

where $\boldsymbol{x}_i$ is the ith row of $\boldsymbol{X}$. Using $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{S}^T\boldsymbol{B}^T$, then $\boldsymbol{x}_i = \boldsymbol{u}_i\boldsymbol{S}^T\boldsymbol{B}^T$ where $\boldsymbol{u}_i$ is the ith row of $\boldsymbol{U}$. Moreover,

$$\|\boldsymbol{U}\|_2^2 = \sum_{i=1}^{n}\|\boldsymbol{u}_i\|_2^2.$$

Then, Problem (1) becomes:

$$\min_{\|\boldsymbol{S}\|_2=1} \ \min_{\boldsymbol{U}} \frac{1}{nm}\left(\sum_{i=1}^{n}\left(\|(\boldsymbol{u}_i\boldsymbol{S}^T\boldsymbol{B}^T - \boldsymbol{a}_i)\boldsymbol{W}_i\|_2^2 + \frac{1}{\gamma}\|\boldsymbol{u}_i\|_2^2\right)\right).$$

We then notice that within the sum $\sum_{i=1}^{n}$ each row of $\boldsymbol{U}$ can be optimized separately, leading to:

$$\min_{\|\boldsymbol{S}\|_2=1} \quad \frac{1}{nm}\left(\sum_{i=1}^{n}\min_{\boldsymbol{u}_i}\left(\|(\boldsymbol{u}_i\boldsymbol{S}^T\boldsymbol{B}^T-\boldsymbol{a}_i)\boldsymbol{W}_i\|_2^2+\frac{1}{\gamma}\|\boldsymbol{u}_i\|_2^2\right)\right). \tag{3}$$

The inner optimization problem $\min_{\boldsymbol{u}_i}\|(\boldsymbol{u}_i\boldsymbol{S}^T\boldsymbol{B}^T-\boldsymbol{a}_i)\boldsymbol{W}_i\|_2^2+\frac{1}{\gamma}\|\boldsymbol{u}_i\|_2^2$ can be solved in closed form given $\boldsymbol{S}$, as it is a weighted linear regression problem with Tikhonov regularization. The closed form solution is:

$$\boldsymbol{a}_i\boldsymbol{W}_i(\boldsymbol{I}_m+\gamma\boldsymbol{W}_i\boldsymbol{B}\boldsymbol{S}\boldsymbol{S}^T\boldsymbol{B}^T\boldsymbol{W}_i^T)^{-1}\boldsymbol{W}_i\boldsymbol{a}_i^T=\overline{\boldsymbol{a}}_i(\boldsymbol{I}_m+\gamma\boldsymbol{W}_i\boldsymbol{B}\boldsymbol{S}\boldsymbol{S}^T\boldsymbol{B}^T\boldsymbol{W}_i)^{-1}\overline{\boldsymbol{a}}_i^T.$$

So Problem (3) can be simplified to:

$$\min_{\|\boldsymbol{S}\|_2=1} \frac{1}{nm}\left(\sum_{i=1}^{n}\overline{\boldsymbol{a}}_i(\boldsymbol{I}_m+\gamma\boldsymbol{W}_i\boldsymbol{B}\boldsymbol{S}\boldsymbol{S}^T\boldsymbol{B}^T\boldsymbol{W}_i^T)^{-1}\overline{\boldsymbol{a}}_i^T\right).$$

Then, let us define $\boldsymbol{V}=\boldsymbol{B}\boldsymbol{S}$, and write:

$$\begin{aligned}
&\frac{1}{nm}\left(\sum_{i=1}^{n}\overline{\boldsymbol{a}}_i(\boldsymbol{I}_m+\gamma\boldsymbol{W}_i\boldsymbol{B}\boldsymbol{S}\boldsymbol{S}^T\boldsymbol{B}^T\boldsymbol{W}_i^T)^{-1}\overline{\boldsymbol{a}}_i^T\right)\\
&=\frac{1}{nm}\left(\sum_{i=1}^{n}\overline{\boldsymbol{a}}_i(\boldsymbol{I}_m+\gamma\boldsymbol{W}_i\boldsymbol{V}\boldsymbol{V}^T\boldsymbol{W}_i^T)^{-1}\overline{\boldsymbol{a}}_i^T\right)\\
&=\frac{1}{nm}\sum_{i=1}^{n}\overline{\boldsymbol{a}}_i\left(\boldsymbol{I}_m-\boldsymbol{V}\left(\frac{\boldsymbol{I}_k}{\gamma}+\boldsymbol{V}^T\boldsymbol{W}_i\boldsymbol{V}\right)^{-1}\boldsymbol{V}^T\right)\overline{\boldsymbol{a}}_i^T,
\end{aligned}$$

which is the needed form in (2). The second equality comes from the matrix inversion lemma, as derived in Woodbury (1950):

Lemma 1 *For matrices $\boldsymbol{U}\in\mathbb{R}^{n\times k}$ and $\boldsymbol{V}\in\mathbb{R}^{k\times n}$, we have the following equivalence:*

$$(\boldsymbol{I}_n+\boldsymbol{U}\boldsymbol{V})^{-1}=\boldsymbol{I}_n-\boldsymbol{U}(\boldsymbol{I}_k+\boldsymbol{V}\boldsymbol{U})^{-1}\boldsymbol{V}.$$

$\blacksquare$

3. A Gradient Descent Algorithm

In this section, we describe a gradient descent algorithm to solve the separable optimization problem (2), and examine its properties. First, we introduce

$$\alpha_i(\boldsymbol{S})=\overline{\boldsymbol{a}}_i\gamma_i(\boldsymbol{S})=\overline{\boldsymbol{a}}_i\left(\boldsymbol{I}_m-\boldsymbol{V}\left(\frac{\boldsymbol{I}_k}{\gamma}+\boldsymbol{V}^T\boldsymbol{W}_i\boldsymbol{V}\right)^{-1}\boldsymbol{V}^T\right)\overline{\boldsymbol{a}}_i^T,\quad i=1,\ldots,n. \tag{4}$$

Here $\alpha_i(\boldsymbol{S})$ is a scalar and $\boldsymbol{\gamma}_i(\boldsymbol{S})$ is a $m \times 1$ vector. Then the function $c(\boldsymbol{S})$ in (2) can be expressed as

$$c(\boldsymbol{S}) = \frac{1}{nm}\sum_{i=1}^{n} \alpha_i(\boldsymbol{S}) = \frac{1}{nm}\sum_{i=1}^{n} \overline{\boldsymbol{a}}_i \boldsymbol{\gamma}_i(\boldsymbol{S}).$$

Using this notation, we then can calculate the derivative of $c(\boldsymbol{S})$:

Lemma 2

$$\nabla c(\boldsymbol{S}) = -\frac{2\gamma}{n}\sum_{i=1}^{n} \boldsymbol{B}^T \boldsymbol{\gamma}_i(\boldsymbol{S})\boldsymbol{\gamma}_i(\boldsymbol{S})^T \boldsymbol{V}. \tag{5}$$

Proof By the standard result of the derivative of the inverse matrix, we have:

$$\begin{aligned}
\nabla \alpha_i(\boldsymbol{S}) &= \nabla \overline{\boldsymbol{a}}_i \boldsymbol{\gamma}_i(\boldsymbol{S}) \\
&= -2\gamma \boldsymbol{B}^T \boldsymbol{\gamma}_i(\boldsymbol{S}) \overline{\boldsymbol{a}}_i (\boldsymbol{I}_m + \gamma \boldsymbol{W}_i \boldsymbol{V}\boldsymbol{V}^T \boldsymbol{W}_i^T)^{-1} \boldsymbol{B}\boldsymbol{S} \\
&= -2\gamma \boldsymbol{B}^T \boldsymbol{\gamma}_i(\boldsymbol{S})\boldsymbol{\gamma}_i(\boldsymbol{S})^T \boldsymbol{V}.
\end{aligned}$$

Then, since $c(\boldsymbol{S}) = \frac{1}{nm}\sum_{i=1}^{n} \alpha_i(\boldsymbol{S})$, we have the required result. ■

Using Lemma 2, we apply a projected gradient descent algorithm (as discussed in Bertsekas (1997)) on the hypersphere $\|\boldsymbol{S}\|_2 = 1$ and obtain in Algorithm 1. (Note that $\mathbf{0}^{p\times k}$ denotes a zero matrix of dimension $p \times k$).

We explain the reasoning behind some key steps of Algorithm 1.

Nesterov Step This is Step 7 of Algorithm 1. We update the gradient with the accelerated formula in Nesterov (1983) by adding the gradient of the current step to $\frac{t-1}{t+2}$ times the previous gradient, which introduces damping in the resulting gradient and enables faster convergence.

Projection and Update Steps This is Step 8 and 9 of Algorithm 1. There are two reasons why we are optimizing over the $\|\boldsymbol{S}\|_2 = 1$: (a) as explained, rescaling $\boldsymbol{S}$ does not change the objective, so the restriction can guarantee a smaller set of feasible solutions; (b) such restriction enables us to effectively approximate the objective value, and its derivative using random sampling, see Theorem 1.

Because we are optimizing on the hypersphere $\|\boldsymbol{S}\|_2 = 1$, our gradient updates need to be projected to the tangent plane of the sphere at the current point $\boldsymbol{S}_t$. Thus, we project the raw gradient, $\nabla\tilde{\boldsymbol{S}}_{t+1}$ onto the tangent plane to get $\overline{\nabla \boldsymbol{S}_{t+1}}$. Then the update step ensures the norm of the updated $\boldsymbol{S}$ is renormalized to 1.

Note that updating the gradient on the hypersphere is a rotation on the great circle formed by $\boldsymbol{S}_t$ and $\overline{\nabla \boldsymbol{S}_{t+1}}$, so the update formula is $\boldsymbol{S}_t \cos\theta + \frac{\overline{\nabla \boldsymbol{S}_{t+1}}}{\|\overline{\nabla \boldsymbol{S}_{t+1}}\|_2}\sin\theta$.

Algorithm 1 Gradient Descent algorithm for matrix completion with side information.

```
1:  procedure projImpute($\boldsymbol{A}$, $\boldsymbol{B}$, $k$, $\theta$, $t_{max}$)    ▷ $\boldsymbol{A} \in \mathbb{R}^{n\times m}$ the masked matrix, $\boldsymbol{B} \in \mathbb{R}^{p\times m}$ the feature matrix, $k$ the desired rank, $\theta$ the step size, and $t_{max}$ the number of steps
2:      $\boldsymbol{S}_1 \leftarrow$ random initial matrix with $\|\boldsymbol{S}_1\|_2 = 1$    ▷ Randomized Start
3:      $\eta_0 \leftarrow \infty$    ▷ Initialize objective value
4:      $\eta_1, \boldsymbol{G}_1 \leftarrow c(\boldsymbol{S}_1), \nabla c(\boldsymbol{S}_1)$    ▷ Initialize objective value and gradient
5:      $\nabla\tilde{\boldsymbol{S}}_1 \leftarrow \boldsymbol{0}^{p\times k}$    ▷ Initialize accelerated gradient
6:      while $t < t_{max}$ do    ▷ While we have not reached $t_{max}$ iterations
7:          $\nabla\tilde{\boldsymbol{S}}_{t+1} = \boldsymbol{G}_t + \frac{t-1}{t+2}\nabla\tilde{\boldsymbol{S}}_t$    ▷ Nesterov accelerated gradient update step
8:          $\overline{\nabla\boldsymbol{S}_{t+1}} = -\nabla\tilde{\boldsymbol{S}}_{t+1} + (\nabla\tilde{\boldsymbol{S}}_{t+1}\cdot\boldsymbol{S}_t)\boldsymbol{S}_t$    ▷ Project gradient to tangent plane of $\boldsymbol{S}_t$
9:          $\boldsymbol{S}_{t+1} \leftarrow \boldsymbol{S}_t\cos\theta + \frac{\overline{\nabla\boldsymbol{S}_{t+1}}}{\|\overline{\nabla\boldsymbol{S}_{t+1}}\|_2}\sin\theta$    ▷ Update $\boldsymbol{S}_t$ by projected gradient
10:         $\eta_{t+1}, \boldsymbol{G}_{t+1} \leftarrow c(\boldsymbol{S}_{t+1}), \nabla c(\boldsymbol{S}_{t+1})$    ▷ Update the new cost and derivative.
11:         $t \leftarrow t+1$
12:     end while
13:     $\boldsymbol{S}^* \leftarrow \boldsymbol{S}_t$
14:     $i \leftarrow 1$
15:     for $i < n$ do
16:         $\boldsymbol{a}_i \leftarrow \overline{\boldsymbol{a}}_i\boldsymbol{B}\boldsymbol{S}^*(\boldsymbol{B}\boldsymbol{S}^*\boldsymbol{W}_i\boldsymbol{B}\boldsymbol{S}^{*T}\boldsymbol{B}^T)^{-1}\boldsymbol{S}^{*T}\boldsymbol{B}^T$    ▷ Calculate the final $\boldsymbol{A}$ matrix
17:     end for
18:     return $\boldsymbol{A}$    ▷ Return the filled matrix $\boldsymbol{A}$
19: end procedure
```

3.1 Discussion on Computational Complexity

There are two key computational steps of the algorithm - Step 10, where the cost and the derivative is calculated in every gradient update, and Step 16 where the final matrix is calculated. We next derive the asymptotic complexity of such steps.

Proposition 2 *The computation complexity of Step 10 in projImpute is*

$$O\left(|\Omega|\left(p + k^2 + \frac{pk + k^3}{m\alpha}\right)\right),$$

where $|\Omega|$ is the number of samples given in the original matrix $\boldsymbol{A}$ and $\alpha = \frac{|\Omega|}{mn}$ is the percentage not missing. The computational complexity of Step 16 is

$$O\left(|\Omega|\left(k^2 + \frac{k}{\alpha} + \frac{k^3}{m\alpha}\right)\right).$$

Proof We analyze Steps 10 and 16 separately.

Computational Complexity of Step 10

Recall we have that:

$$\alpha_i(\boldsymbol{S}) = \overline{\boldsymbol{a}}_i\boldsymbol{\gamma}_i(\boldsymbol{S}) = \overline{\boldsymbol{a}}_i\left(\boldsymbol{I}_m - \boldsymbol{V}\left(\frac{\boldsymbol{I}_k}{\gamma} + \boldsymbol{V}^T\boldsymbol{W}_i\boldsymbol{V}\right)^{-1}\boldsymbol{V}^T\right)\overline{\boldsymbol{a}}_i^T, \quad i = 1, \dots, n,$$

First let us denote m_i as the number of non-zero entries of $\boldsymbol{W}_i$. This is the number of known entries per row. Then define $\boldsymbol{V}_{W_i} \in \mathbb{R}^{m_i \times k}$ as the matrix of $\boldsymbol{W}_i\boldsymbol{V}$ after removing the all-zero columns, as illustrated below:

$$\begin{aligned}\boldsymbol{W}_i\boldsymbol{V} &= \mathrm{Diag}(1,0,1,\cdots,0) \times \begin{pmatrix} — & \boldsymbol{v}_1 & — \\ — & \boldsymbol{v}_2 & — \\ — & \boldsymbol{v}_3 & — \\ \vdots & \vdots & \vdots \\ — & \boldsymbol{v}_k & — \end{pmatrix} \\ &= \begin{pmatrix} — & \boldsymbol{v}_1 & — \\ — & \boldsymbol{0} & — \\ — & \boldsymbol{v}_3 & — \\ \vdots & \vdots & \vdots \\ — & \boldsymbol{0} & — \end{pmatrix} \rightsquigarrow \begin{pmatrix} — & \boldsymbol{v}_1 & — \\ — & \boldsymbol{v}_3 & — \\ \vdots & \vdots & \vdots \end{pmatrix} := \boldsymbol{V}_{W_i}. \end{aligned} \tag{6}$$

Note that $\boldsymbol{V}_{W_i}$ can be created efficiently through subsetting, and its creation does not impact the asymptotic running time. Then similarly, we denote $\boldsymbol{a}_{W_i} \in \mathbb{R}^{1\times m_i}$ as $\boldsymbol{a}\boldsymbol{W}_i$ with all the zero elements removed. Then we have the following equality, using (4):

$$\begin{aligned} c(\boldsymbol{S}) &= \frac{1}{nm}\sum_{i=1}^{n}\overline{\boldsymbol{a}}_i\boldsymbol{\gamma}_i(\boldsymbol{S}) \\ &= \frac{1}{nm}\sum_{i=1}^{n}\overline{\boldsymbol{a}}_i\left(\boldsymbol{I}_m - \boldsymbol{V}\left(\frac{\boldsymbol{I}_k}{\gamma} + \boldsymbol{V}^T\boldsymbol{W}_i\boldsymbol{V}\right)^{-1}\boldsymbol{V}^T\right)\overline{\boldsymbol{a}}_i^T \\ &= \frac{1}{nm}\sum_{i=1}^{n}\boldsymbol{a}_{W_i}\left(\boldsymbol{I}_{m_i} - \boldsymbol{V}_{W_i}\left(\frac{\boldsymbol{I}_k}{\gamma} + \boldsymbol{V}_{W_i}^T\boldsymbol{V}_{W_i}\right)^{-1}\boldsymbol{V}_{W_i}^T\right)\boldsymbol{a}_{W_i}^T \qquad (7) \\ &= \frac{1}{nm}\sum_{i=1}^{n}\boldsymbol{a}_{W_i}\boldsymbol{a}_{W_i}^T - \boldsymbol{a}_{W_i}\boldsymbol{V}_{W_i}\left(\frac{\boldsymbol{I}_k}{\gamma} + \boldsymbol{V}_{W_i}^T\boldsymbol{V}_{W_i}\right)^{-1}\boldsymbol{V}_{W_i}^T\boldsymbol{a}_{W_i}^T. \end{aligned}$$

The equality (7) follows directly from the definition of $\boldsymbol{V}\boldsymbol{W}_i$ and $\boldsymbol{a}_{W_i}$ as all the removed elements/columns are zero.

Then, the complexity of each term in the sum (7) is $O(m_ik^2 + k^3)$. Summing over n terms and noting that:

$$\sum_{i=1}^{n} nm_i = |\Omega|,$$

we obtain that the full complexity of the objective function is thus

$$O(|\Omega|k^2 + nk^3) = O\left(|\Omega|\left(k^2 + \frac{1}{m\alpha}k^3\right)\right).$$

Now let us further define:

$$\boldsymbol{\gamma}_{W_i}(\boldsymbol{S}) = \left(\boldsymbol{I}_{m_i} - \boldsymbol{V}_{W_i}\left(\frac{\boldsymbol{I}_k}{\gamma} + \boldsymbol{V}_{W_i}^T\boldsymbol{V}_{W_i}\right)^{-1}\boldsymbol{V}_{W_i}^T\right)\boldsymbol{a}_{W_i}^T, \tag{8}$$

and $\boldsymbol{B}_{W_i}$ in the same fashion as defined in (6). Then we can similarly show that the derivative (5) is equivalent to the following expression:

$$\begin{aligned}\nabla c(\boldsymbol{S}) &= -\frac{2\gamma}{n}\sum_{i=1}^{n}\boldsymbol{B}^T\boldsymbol{\gamma}_i(\boldsymbol{S})\boldsymbol{\gamma}_i(\boldsymbol{S})^T\boldsymbol{V} &(9)\\ &= -\frac{2\gamma}{n}\sum_{i=1}^{n}\boldsymbol{B}_{W_i}^T\boldsymbol{\gamma}_{W_i}(\boldsymbol{S})\boldsymbol{\gamma}_{W_i}(\boldsymbol{S})^T\boldsymbol{V}_{W_i}. &(10)\end{aligned}$$

The matrix multiplication in the final expression (10) has computational complexity of $O(m_i(p+k)+pk)$ (every individual term has been previously calculated), so summing over n terms the computational complexity is

$$O\left(|\Omega|\left(p + k + \frac{pk}{m\alpha}\right)\right).$$

Thus, the computational complexity of the entire step is:

$$O\left(|\Omega|\left(p + k^2 + \frac{pk + k^3}{m\alpha}\right)\right).$$

Computational Complexity of Step 16

Recall that Step 16 of projImpute does the following calculation for every $i \in \{1, \cdots, n\}$:

$$\boldsymbol{a}_i \leftarrow \overline{\boldsymbol{a}}_i\boldsymbol{B}\boldsymbol{S}^*(\boldsymbol{B}\boldsymbol{S}^*\boldsymbol{W}_i\boldsymbol{B}\boldsymbol{S}^{*T}\boldsymbol{B}^T)^{-1}\boldsymbol{S}^{*T}\boldsymbol{B}^T, \tag{11}$$

where $\boldsymbol{S}^*$ is the imputed $\boldsymbol{S}$. Now define $\boldsymbol{V}^* = \boldsymbol{B}\boldsymbol{S}^*$, and define $\boldsymbol{V}_{W_i}^*$ in the same fashion as defined in (6). Then we have that:

$$\begin{aligned}\overline{\boldsymbol{a}}_i\boldsymbol{B}\boldsymbol{S}^*(\boldsymbol{B}\boldsymbol{S}^*\boldsymbol{W}_i\boldsymbol{B}\boldsymbol{S}^{*T}\boldsymbol{B}^T)^{-1}\boldsymbol{S}^{*T}\boldsymbol{B}^T &= \overline{\boldsymbol{a}}_i\boldsymbol{V}^*(\boldsymbol{V}^{*T}\boldsymbol{W}_i\boldsymbol{V}^*)^{-1}\boldsymbol{V}^{*T}\\ &= \boldsymbol{a}_{W_i}\boldsymbol{V}^*(\boldsymbol{V}_{W_i}^{*T}\boldsymbol{V}_{W_i}^*)^{-1}\boldsymbol{V}_{W_i}^{*T}. &(12)\end{aligned}$$

The final expression (12) has a matrix multiplication complexity of $O(km_i + k^2m_i + k^3 + mk)$, so summing over n samples the total computational complexity of Step 16 is:

$$O\left(|\Omega|\left(k^2 + \frac{k}{\alpha} + \frac{k^3}{m\alpha}\right)\right).$$

■

Using these results we can easily arrive at the following corollary:

Corollary 1 *The projImpute algorithm terminates in t_{max} number of steps. Furthermore, it has complexity*

$$O\left(|\Omega|\left(p+k^2+\frac{pk+k^3}{m\alpha}\right)\right).$$

Proof The computational complexity follows immediately from Proposition 2. ■

Note that with $p \gg k$ sufficiently large, the computational complexity of Step 10 dominates that of Step 16. In particular, such step scales scales linearly in m, n, and p, which is undesirable when all of these are large, as it is commonly in real-world settings. In the next section, we design an algorithm that reduces the computational complexity on Step 10.

4. fastImpute: An Adaptive Stochastic Projected Gradient Descent Algorithm for Matrix Completion

In the previous section, we introduced the projImpute algorithm, which conducted full projected gradient updates for every step $t \in \{1, \cdots, t_{max}\}$ using all n rows and m columns. However, it is computationally prohibitively expensive, especially when n, m, p is large.

In this section, we introduce fastImpute that uses random sampling of rows and columns to estimate the gradient update at step t. Specifically, at each step, we randomly select n_0 rows and *for each row*, we randomly select m_0 columns to calculate the sampled derivative for that row. Let $[n_t]$ denote the random set of numbers from $\{1, \cdots, n\}$ of size n_0 at step t, and $[m_t^i]$ denote a set of numbers from $\{1, \cdots, m\}$ of size m_0 at step t corresponding to the random sample for row i. Then the objective function evaluated with rows $[n_t]$ and columns $[m_t^i]$ is:

$$c_t(\boldsymbol{S}) = \frac{1}{n_0 m_0}\sum_{i\in[n_t]} \overline{\boldsymbol{a}}_{im_t^i}\left(\boldsymbol{I}_{m_t^i} - \boldsymbol{V}_{m_t^i}\left(\frac{\boldsymbol{I}_k}{\gamma} + \boldsymbol{V}_{m_t^i}^T \boldsymbol{W}_{im_t^i}\boldsymbol{V}_{m_t^i}\right)^{-1}\boldsymbol{V}_{m_t^i}^T\right)\overline{\boldsymbol{a}}_{im_t^i}^T,$$

where $\boldsymbol{V}_{m_t^i}$ is the submatrix of $\boldsymbol{V}$ formed with $[m_t^i]$ columns, and similarly with other terms. We use the notation m_t^i to explicitly indicate that the columns are selected for each row $i \in [n_t]$ and independent across rows. Similarly, using Lemma 2, the derivative evaluated with rows $[n_t]$ and columns $[m_t^i]$ is:

$$\begin{aligned}\nabla c_t(\boldsymbol{S}) = \frac{1}{n_0 m_0}\sum_{i\in[n_t]} -2\gamma \boldsymbol{B}_{m_t^i}^T\left(I_{m_t^i} - \boldsymbol{V}_{m_t^i}\left(\frac{\boldsymbol{I}_k}{\gamma} + \boldsymbol{V}_{m_t^i}^T \boldsymbol{W}_{im_t^i}\boldsymbol{V}_{m_t^i}\right)^{-1}\boldsymbol{V}_{m_t^i}^T\right)\overline{\boldsymbol{a}}_{im_t^i}^T \\ \times\, \overline{\boldsymbol{a}}_{im_t^i}\left(I_{m_t^i} - \boldsymbol{V}_{m_t^i}\left(\frac{\boldsymbol{I}_k}{\gamma} + \boldsymbol{V}_{m_t^i}^T \boldsymbol{W}_{im_t^i}\boldsymbol{V}_{m_t^i}\right)^{-1}\boldsymbol{V}_{m_t^i}^T\right)^T \boldsymbol{V}_{m_t^i}.\end{aligned}$$

We then present the algorithm as Algorithm 2. It is the same procedure as projImpute, except with stochastic gradient updates of n_0 rows and m_0 columns. For this algorithm to

Algorithm 2 Gradient Descent algorithm for matrix completion with side information.

```
procedure fastImpute($\boldsymbol{A}, \boldsymbol{B}, k, \theta, t_{max}$)    ▷ $\boldsymbol{A} \in \mathbb{R}^{n\times m}$ the masked matrix, $\boldsymbol{B} \in \mathbb{R}^{p\times m}$ the feature matrix, $k$ the desired rank, $\theta$ the step size, and $t_{max}$ the number of gradient steps
    $t \leftarrow 1$
    $\alpha \leftarrow \frac{|\Omega|}{mn}$    ▷ Define existing percentage of $\boldsymbol{A}$. $|\Omega|$ is the set of non-zero entries in $\boldsymbol{A}$
    $\boldsymbol{S}_1 \leftarrow$ random initial matrix with $\|\boldsymbol{S}_1\|_2 = 1$    ▷ Randomized Start
    $\eta_0 \leftarrow \infty$    ▷ Initialize objective value
    $q_1 \leftarrow 0$    ▷ Initialize counter for non-improving steps
    $m_0 \leftarrow \min(2p, m)$    ▷ Define initial gradient update size for columns
    $n_0 \leftarrow \left\lfloor \frac{k\sqrt{nm}\log(\sqrt{nm})}{8m_0\alpha} \right\rfloor$    ▷ Define initial gradient update size for rows
    $[n_0] \subset \{1, \cdots, n\}$    ▷ Initialize rows selected
    for $i \in [n_0]$ do
        $[m_0^i] \subset \{1, \cdots, m\}$    ▷ Initialize columns selected
    end for
    $\eta_1, \boldsymbol{G}_1 \leftarrow c_t(\boldsymbol{S}_1), \nabla c_t(\boldsymbol{S}_1)$    ▷ Initialize objective value and gradient
    $\nabla\tilde{\boldsymbol{S}}_1 \leftarrow \boldsymbol{0}^{k\times p}$    ▷ Initialize accelerated gradient
    while $t < t_{max}$ do    ▷ While we have not reached $t_{max}$ iterations
        $[n_t] \subset \{1, \cdots, n\}$    ▷ Select new rows
        for $i \in [n_t]$ do
            $[m_t^i] \subset \{1, \cdots, m\}$    ▷ Select new columns
        end for
        $\nabla\tilde{\boldsymbol{S}}_{t+1} = \boldsymbol{G}_t + \frac{t-1}{t+2}\nabla\tilde{\boldsymbol{S}}_t$    ▷ Nesterov accelerated gradient update step
        $\overline{\nabla\boldsymbol{S}_{t+1}} = -\nabla\tilde{\boldsymbol{S}}_{t+1} + (\nabla\tilde{\boldsymbol{S}}_{t+1} \cdot \boldsymbol{S}_t)\boldsymbol{S}_t$    ▷ Project gradient to the tangent plane of $\boldsymbol{S}_t$
        $\boldsymbol{S}_{t+1} \leftarrow \boldsymbol{S}_t \cos\theta + \frac{\overline{\nabla\boldsymbol{S}_{t+1}}}{\|\overline{\nabla\boldsymbol{S}_{t+1}}\|_2}\sin\theta$    ▷ Update $\boldsymbol{S}_t$ based on projected gradient
        $\eta_{t+1}, \boldsymbol{G}_{t+1} \leftarrow c_t(\boldsymbol{S}_{t+1}), \nabla c_t(\boldsymbol{S}_{t+1})$    ▷ Update the cost and derivative.
        $t \leftarrow t + 1$
    end while
    $\boldsymbol{S}^* \leftarrow \boldsymbol{S}_t$
    $i \leftarrow 1$
    for $i < n$ do
        $\boldsymbol{a}_i \leftarrow \overline{\boldsymbol{a}}_i\boldsymbol{B}\boldsymbol{S}^*(\boldsymbol{B}\boldsymbol{S}^*\boldsymbol{W}_i\boldsymbol{B}\boldsymbol{S}^{*T}\boldsymbol{B}^T)^{-1}\boldsymbol{S}^{*T}\boldsymbol{B}^T$    ▷ Calculate the final $\boldsymbol{A}$ matrix
    end for
    return $\boldsymbol{A}$    ▷ Return the filled matrix $\boldsymbol{A}$
end procedure
```

work, we need the stochastic gradient to be "close" to the true gradient, which is true using a theorem adapted from Bertsimas and Li (2018):

Theorem 1 *Let $\boldsymbol{A}$ be a partially known matrix, $\boldsymbol{B}$ a known feature matrix, and $\boldsymbol{W}_i$ as defined in Proposition 1. Then, with probability at least $1-\epsilon$, for any t, and any $\boldsymbol{S}$ satisfying*

$\|\boldsymbol{S}\| = 1$, *we have:*

$$|c_t(\boldsymbol{S}) - c(\boldsymbol{S})| \leq \sqrt{\frac{Ak \log\left(\frac{k}{\epsilon}\right)}{n_0}},$$
$$\|\nabla c_t(\boldsymbol{S}) - \nabla c(\boldsymbol{S})\|_2 \leq \sqrt{\frac{Bk \log\left(\frac{k}{\epsilon}\right)}{n_0}}.$$

Inverting the statements, we have that:

$$\mathbb{P}(|c_t(\boldsymbol{S}) - c(\boldsymbol{S})| > \epsilon) \leq ke^{-\frac{n_0 \epsilon^2}{Ak}},$$
$$\mathbb{P}(\|\nabla c_t(\boldsymbol{S}) - \nabla c(\boldsymbol{S})\| > \epsilon) \leq ke^{-\frac{n_0 \epsilon^2}{Bk}}.$$

Theorem 1 shows that the probability the stochastic objective value and its derivative is ϵ away from the true value is exponentially vanishing with increasing n_0. This property affects how we choose n_0 and m_0, which is discussed in Section 4.2.

In the next section, we show that fastImpute recovers the true solution for matrix completion under relatively mild conditions.

4.1 Convergence Rates and Guarantees

In this section, we derive results that show Algorithm 2 converges to the global minimum of $c(\boldsymbol{S})$. for the case $1/\gamma = 0$ and without side information ($p = m$ and $\boldsymbol{B} = \boldsymbol{I}_m$). Under such assumptions, our objective can be written as:

$$c(\boldsymbol{S}) = \frac{1}{nm} \sum_{i=1}^{n} \overline{\boldsymbol{a}}_i \left(\boldsymbol{I}_m - \boldsymbol{S} \left(\boldsymbol{S}^T \boldsymbol{W}_i \boldsymbol{S} \right)^{-1} \boldsymbol{S}^T \right) \overline{\boldsymbol{a}}_i^T.$$

The analysis can be easily extended to the case where $\boldsymbol{B}$ is a general feature matrix, at the expense of increased notational complexity. The analysis below also holds for $\frac{1}{\gamma} < \epsilon_0$ for some sufficiently small ϵ_0 with minimal changes. To facilitate the theoretical analysis, we would further assume that the final projected gradient is perturbed by some asymptotically vanishing Gaussian noise (immediately before Step 22):

$$\overline{\nabla \boldsymbol{S}_{t+1}} \leftarrow \overline{\nabla \boldsymbol{S}_{t+1}} + O\left(\frac{1}{\sqrt{n_0 m}}\right) \boldsymbol{E}$$

Where $\boldsymbol{E} \in \mathbb{R}^{k \times p}$ has iid entries with the standard normal distribution. As n_0 and m is large, this does not significantly change the resultant gradient, and numerical experiments suggest that this has no appreciable impact on the final solution.

Over the last few years, various researchers have proven related results using different formulations for matrix completion (see Ge et al. (2016); Ma et al. (2019) for the case where the recovered matrix is semi-definite, and Zheng and Lafferty (2016) for the general case using a lifting formulation). The proofs follow a two-step process:

- Prove the proposed algorithm converges to a local minimum of the objective function.
- Prove all local minima of the objective function are global minima (equivalently, the objective function has "no spurious local minima"). This condition has been shown to be true for various formulations of nonconvex low rank problems, as explored in e.g. Ge et al. (2017).

In our new formulation of matrix completion, we show how proving the first result is equivalent to checking the "strict saddle" condition on the objective function introduced in Ge et al. (2015). We then prove the objective function $c(\boldsymbol{S})$ does indeed satisfy the strict saddle condition, and moreover has no spurious local minima.

Before we layout our detailed results, we need to specify a random sampling model for the known entries of $\boldsymbol{A}$. Generally in the literature (see e.g. Candès and Tao (2010); Jin et al. (2016); Ge et al. (2016) for examples), the uniform sampling model is used, where every element is assumed to be present independently with a probability r_0. To simplify the proof, we specify a row-sampling model: For every row $\boldsymbol{a}_i$ of $\boldsymbol{A}$, we randomly sample (without replacement) l out of m entries to be known. Thus, each entry has a probability of $r = \frac{l}{m}$ being selected. By results in Candès and Tao (2010) and Gross and Nesme (2010), it is known that convergence results under row-sampling models with sampling rate r hold true under the uniform sampling model with a rate $r_0 = O(\log n)r$, where the factor $\log n$ is essentially due to the coupon collector effect (for more discussion please see Candès and Tao (2010)).

We next show that proving Algorithm 2 converges to a local minimum is equivalent to verifying $c(\boldsymbol{S})$ satisfies the "strict saddle" condition. We first formally define such concept, as introduced in Ge et al. (2015):

Definition 1 *A function $f : \mathbb{R}^d \to \mathbb{R}$ is (θ, ζ, η)-strict saddle if for every $\boldsymbol{x}$, at least one of the following hold for constants $\theta, \zeta, \eta > 0$:*

1. $\|\nabla f(\boldsymbol{x})\| \geq \theta$.
2. $\lambda_{min}(\nabla^2 f(\boldsymbol{x})) \leq -\zeta$.
3. *There exists a local minimum $\boldsymbol{x}^*$ such that $\|\boldsymbol{x} - \boldsymbol{x}^*\| \leq \eta$.*

Using this definition, the following theorem from Fang et al. (2019) (see also Daneshmand et al. (2018); Ge et al. (2015)) establishes that a stochastic gradient algorithm converges to a local minimum if the function f being optimized over satisfies the strict saddle condition:

Theorem 2 *Assume that $f(\boldsymbol{x})$ is a function of $\boldsymbol{x} \in \mathbb{R}^d$ that is a (θ, ζ, η)-strict saddle. Then, a stochastic gradient descent algorithm with stochastic gradient $\tilde{\nabla} f(\boldsymbol{x})$ and added*

Gaussian noise $\frac{\sigma}{\sqrt{d}}\boldsymbol{\epsilon}$ *where* $\boldsymbol{\epsilon} \sim N(0, \boldsymbol{I}_d)$ *converges in* $O\left(\text{poly}\left(\frac{1}{\delta}\right)\right)$ *iterations to a point* δ *close to a local minimum, where we have:*

$$\mathbb{P}(\|\tilde{\nabla} f(\boldsymbol{x}) - \nabla f(\boldsymbol{x})\| > \epsilon) \leq e^{-\frac{\epsilon^2}{2\sigma^2}}.$$

In the current problem, Theorem 1 shows that a valid bound on the standard deviation is $\sqrt{\frac{Bk\log(k)}{n_0}}$. Therefore, as $d = mk$, a stochastic gradient descent algorithm with noise of $O(\frac{1}{\sqrt{n_0 m}})$ would converge to a local minimum of $c(\boldsymbol{S})$. However, Algorithm 2 is a *projected* stochastic gradient descent algorithm (with such added perturbation), so we cannot immediately apply Theorem 2. Thus, we next connect Algorithm 2 with the standard stochastic gradient algorithm. Note that for the case $1/\gamma = 0$, the stochastic gradient is always orthogonal to $\boldsymbol{S}$:

Lemma 3 *Let* $1/\gamma = 0$. *Then for all* $\boldsymbol{S}$ *and any random sample* $[n_t], [m_t]$ *of rows and columns, we have*

$$\boldsymbol{S}^T \nabla c_t(\boldsymbol{S}) = 0.$$

The statement follows immediately from Lemma 4 in Appendix A. Therefore, the projection step (Step 21 in Algorithm 2) is trivial and reduces to:

$$\overline{\nabla \boldsymbol{S}_{t+1}} = -\nabla \tilde{\boldsymbol{S}}_{t+1}$$

In other words, the projected gradient is exactly the original gradient, and the projection step does not change the derivative. Thus, Algorithm 2 under $1/\gamma = 0$ reduces to a stochastic gradient algorithm.

Thus, if we can establish that $c(\boldsymbol{S})$ is a (θ, ζ, η)-strict saddle, we can indeed use Theorem 2 to prove that Algorithm 2 converges to a local minimum. Therefore, to establish the global convergence of Algorithm 2, we now equivalently need to establish the following two results:

1. The function $c(\boldsymbol{S})$ is a (θ, ζ, η)-strict saddle.
2. All local minima of $c(\boldsymbol{S})$ are global minima.

These properties and Theorem 2 will then establish that Algorithm 2 converges to a point δ-close to the global minimum in $O\left(\text{poly}\left(\frac{1}{\delta}\right)\right)$ iterations. We prove these two results simultaneously. To do so, we introduce a few regularity conditions that are necessary for these statements to be true. First, the matrix $\boldsymbol{A}$ we want to recover cannot be the following:

$$\boldsymbol{A} = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix}. \tag{13}$$

This is because to recover such matrix truthfully, we need to have to know A_{11} as knowing any other element would give us no information about the 1 in the top left corner. To know A_{11} we would require basically all entries to be known under a random sampling model, which is undesirable; see Candès and Recht (2009) for further discussion. To prevent this from happening, we introduce the following concept:

Definition 2 *We define a matrix $\boldsymbol{S} \in \mathbb{R}^{m\times k}$, for $m > k$, to be (α, p)-submatrix full rank if every $p \times k$ submatrix of $\boldsymbol{S}$ is full column rank and the minimum singular value is at least α.*

We introduce the following assumption:

Assumption 1 *The desired matrix to be recovered, $\boldsymbol{A}$, is of rank k and admits a decomposition $\boldsymbol{A} = \boldsymbol{U}^*\boldsymbol{S}^{*T}$ where $\boldsymbol{U}^* \in \mathbb{R}^{n\times k}, \boldsymbol{S}^* \in \mathbb{R}^{m\times k}$, $\|\boldsymbol{S}^*\| = 1$, and $\boldsymbol{U}^*$, $\boldsymbol{S}^*$ is (α, l)-submatrix full rank, where l/m is the sampling rate and $\alpha > 0$.*

Qualitatively, this assumption requires the decomposition of $\boldsymbol{A}$ to be "spread out" enough so that no particular row and column is special. In particular, it excludes the pathological matrices like the one presented in Eq. (13), as $\boldsymbol{A} = (1, 0, \cdots, 0)(1, 0, \cdots, 0)^T = \boldsymbol{U}^*\boldsymbol{S}^{*T}$, and most of the submatrices of $\boldsymbol{U}^* = \boldsymbol{S}^* = (1, 0, \cdots, 0)$ have rank 0, save for those that includes the first element. This is because the first element is special in the decomposition of such $\boldsymbol{A}$, and if the first element is missing, no information for $\boldsymbol{A}$ can be deduced. Assumption 1 excludes these matrices so that no element is special enough such that if it is missing, we fail to recover $\boldsymbol{A}$.

We now introduce a second technical assumption. If $\boldsymbol{W_i S}$ has rank $< k$, then the inverse $(\boldsymbol{S}^T\boldsymbol{W}_i\boldsymbol{S})^{-1}$ in the objective function $c(\boldsymbol{S})$ would not be well-defined, so the objective function is not defined. Therefore, the second assumption is that we restrict our domain to the set of $\boldsymbol{S}$ where the objective function is defined:

Assumption 2 *The set $\boldsymbol{S}$ is restricted to matrices that are (α, l)-submatrix full rank, where l/m is the sampling rate and $\alpha > 0$.*

In real-world situations this assumption is almost never violated as numerical matrices almost never have *perfectly* singular submatrices. If it is, any small random perturbation around $\boldsymbol{S}$ would almost surely make the condition true.

Now, with these regularity conditions, we establish that for r sufficiently large, the two results we aim to prove are true:

Theorem 3 *Assume $\boldsymbol{A}$ follows the row-sampling model, and each element is known with probability $r = \frac{l}{m} > \frac{Ck}{m}$ for sufficiently large C. Further assume Assumptions 1-2 hold. Then, with probability $1 - O(\frac{1}{n})$, for $1/\gamma = 0$ and $\boldsymbol{B} = \boldsymbol{I}_m$, $c(\boldsymbol{S})$ is $(\epsilon, O(\frac{\epsilon}{k}), O(\sqrt{\epsilon}))$-strict saddle for sufficiently small ϵ. Furthermore, all local minima of $c(\boldsymbol{S})$ are global minima.*

The proof is contained in Appendix A. Then combining Theorem 3 with Theorem 2, we have the desired global convergence result:

Corollary 2 *Consider the minimization problem defined in (2) under $1/\gamma = 0$ and $\boldsymbol{B} = \boldsymbol{I}_m$. Assume that Assumptions 1-2 hold. Then, under the row-sampling model, for $r > \frac{Ck}{m}$ with C sufficiently large, Algorithm 2 (with added perturbation) outputs a point that is δ-close to a global minimum in $\mathrm{poly}(\frac{1}{\delta})$ iterations with probability $1 - O(\frac{1}{n})$.*

As explained above, we can transfer results in the row-sampling model to the commonly utilized uniform sampling (Bernoulli) model:

Corollary 3 *Consider the minimization problem defined in (2) under $1/\gamma = 0$ and $\boldsymbol{B} = \boldsymbol{I}_m$. Assume that Assumptions 1-2 hold. Then, under the uniform sampling model, for a sampling rate $r_0 > \frac{Cnk\log(n)}{mn}$ with C sufficiently large, Algorithm 2 (with added perturbation) outputs a point that is δ-close to a global minimum in $\mathrm{poly}(\frac{1}{\delta})$ iterations with probability $1 - O(\frac{1}{n})$.*

The conclusion is consistent with other algorithms which require $O(nk^\alpha \log(n))$ samples under the uniform sampling model.

4.1.1 Necessity of Regularization

The previous results deal with the case when there is no regularization ($1/\gamma = 0$) though it can be readily extended to any sufficiently small regularization term. Therefore, it is natural to question the purpose of the ℓ_2 regularization term in the formulation. The answer is two-fold:

1. The proof above is for the case when the matrix $\boldsymbol{A}$ is perfectly low rank. However, in real-world cases, the data might be corrupted in unseen ways, and it is shown in Bertsimas and Copenhaver (2018) that ℓ_2 regularization provides robustness properties that could guard against such corruption.
2. There are cases in the real-world in which the the number of known entries in a row $\boldsymbol{A}_i$ of the partially-known matrix is less than the desired rank k, and in such cases the regularization becomes necessary to recover the true solution, as $\mathrm{Rank}(\boldsymbol{W}_i) < k$, so $\boldsymbol{S}\boldsymbol{W}_i\boldsymbol{S}^T$ is not invertible. Adding a regularization term guards against such cases.

4.2 Computational Complexity of fastImpute

Here we first discuss how m_0 and n_0 is chosen. By Theorem 1 on the approximation of the stochastic gradient and objective, the error in such approximation drops exponentially with increasing n_0. Therefore, we want to ensure n_0 is never too small.

Now, by our convergence result in Theorem 3, we need $O(nk \log n)$ samples to recover the true solution under the uniform sampling model, when n is sufficiently large. Since α is the empirical probability that any element is known, we need to have:

$$m_0 n_0 \alpha = O(nk \log n).$$

The complexity required to discern the p factors we can choose from is $O(p)$, thus, our selection of n_0 and m_0 is:

$$m_0 = O(p), \qquad n_0 = \max\left\{O\left(\frac{nk\log(n)}{m_0\alpha}\right), \underline{n}\right\}.$$

where $\underline{n}$ is some lower bound on n_0 to ensure that Theorem 3 can be applied. We now show that such m_0 and n_0 reduces the computational complexity of the gradient update step asymptotically:

Corollary 4 *The computational complexity of Step 10 in Algorithm 2 is*

$$O\left(\frac{(pk+k^3)n\log(n)}{\alpha}\right).$$

Proof For the gradient update step, we replace m with $O(p)$ and n with $O\left(\frac{kn\log(n)}{p\alpha}\right)$ in the original formula for Step 10 in projImpute to get that its computational complexity is:

$$\begin{aligned} &O\left(\frac{kn\log(n)}{\alpha}\left(p + k^2 + \frac{k}{\alpha} + \frac{k^3+pk}{p\alpha}\right)\right) \\ &= O\left(\frac{(pk+k^3)n\log(n)}{\alpha}\right). \end{aligned}$$

Where we have suppressed the last two terms as they are dominated by the first two. ■

We note that the dependence of m is completely removed (for sufficiently large n) when compared with Algorithm 1. This, however, does not reduce the asymptotic complexity with regards to m for the full algorithm as eventually the step to fill the matrix $\boldsymbol{A}$ dominates, and that scales linearly with m. Our dependence on k is cubic, but k is usually small in real-world applications, so it is not a large concern.

In the next section, we discuss experiments for fastImpute.

5. Experiments on fastImpute with Side Information

In this section, we compare fastImpute with other inductive matrix completion algorithms on both synthetic datasets and real-world datasets to explore its performance and scaling behavior.

5.1 Synthetic Data Experiments

For synthetic data experiments, we assume that the underlying matrix satisfies the form $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{S}^T\boldsymbol{B}^T$, where $\boldsymbol{U} \in \mathbb{R}^{n\times k}$, $\boldsymbol{S} \in \mathbb{R}^{p\times k}$, and $\boldsymbol{B} \in \mathbb{R}^{m\times p}$. The elements of $\boldsymbol{U}$, $\boldsymbol{S}$, and $\boldsymbol{B}$ are selected from a uniform distribution of $[0, 1]$, where a fraction μ is missing. We report statistics on various combinations of (m, n, p, k, μ).

All algorithms are tested on a server with 16 CPU cores. For each combination (m, n, p, k, μ), we ran 10 tests and report the average value for every statistic. The algorithms tested are:

- **fastImpute**: We use the sampling parameters:

$$m_0 = \min(2p, m), \qquad n_0 = \max\left\{\frac{nk\log(n)}{8m_0\alpha}, 100\right\}.$$

  with $t_{max} = 50$, and $\theta = \frac{\pi}{64}$, and regularization parameter $\gamma = 10^6$. We explicitly stress here that no parameter tuning is done on fastImpute as we intend to show that the algorithm is not parameter sensitive. We implement our algorithm in Julia 0.6 with only the base packages.

- **IMC**: This algorithm is a well-accepted benchmark for testing Inductive Matrix Completion algorithms developed by Natarajan and Dhillon (2014). For each combination of (m, n, p, k, μ), we tune the regularization parameter λ by imputing random matrices with such combination and find the λ that gives the best results. We utilize the implementation provided by the authors in Matlab.

To further understand the benefits of the stochastic algorithm, we also compare against projImpute, the full gradient descent algorithm. We report the following statistics for each algorithm:

- n, m - the dimensions of $\boldsymbol{A}$.
- p - the number of features in the feature matrix.
- k - the true number of features.
- μ - The fraction of missing entries in $\boldsymbol{A}$.
- T - the total time of algorithm execution.
- MAPE - the Mean Absolute Percentage Error (MAPE) for the retrieved matrix $\hat{\boldsymbol{A}}$:

$$\text{MAPE} = \frac{1}{nm}\sum_{i=1}^{n}\sum_{j=1}^{m}\frac{|\hat{A}_{ij} - A_{ij}|}{|A_{ij}|}.$$

We have the following observations:

	$\boldsymbol{n}$	$\boldsymbol{m}$	$\boldsymbol{p}$	$\boldsymbol{k}$	$\boldsymbol{\mu}$%	**projImpute**		**fastImpute**		**IMC**	
						$\boldsymbol{T}$	**MAPE**	$\boldsymbol{T}$	**MAPE**	$\boldsymbol{T}$	**MAPE**
n	10^3	10^3	100	5	95%	5.1s	0.2%	3.1s	0.4%	4.4s	2.9%
	10^4	10^3	100	5	95%	56.7s	0.1%	12.0s	0.1%	18s	2.5%
	10^5	10^3	100	5	95%	580s	0.2%	54.8s	0.2%	173s	1.0%
	10^6	10^3	100	5	95%	6035s	0.2%	390s	0.2%	1400s	0.3%
m	10^4	10^3	100	5	95%	56.7s	0.1%	12.0s	0.1%	18s	2.5%
	10^4	10^4	100	5	95%	605s	0.2%	39s	0.4%	144s	0.8%
	10^4	10^5	100	5	95%	5874s	0.1%	360s	0.2%	1265s	0.4%
	10^4	10^6	100	5	95%	N/A	0.1%	3056s	0.1%	13891s	0.2%
p	10^4	10^3	100	5	95%	56.7s	0.1%	12.0s	0.1%	18s	2.5%
	10^4	10^3	200	5	95%	104s	0.05%	23s	0.1%	30s	2.7%
	10^4	10^3	500	5	95%	230s	0.06%	40s	0.05%	87s	2.1%
	10^4	10^3	1000	5	95%	417s	0.02%	85s	0.01%	160s	1.9%
k	10^4	10^3	100	5	95%	56.7s	0.1%	12.0s	0.1%	18s	2.5%
	10^4	10^3	100	10	95%	194s	0.1%	37.6s	0.2%	32s	2.1%
	10^4	10^3	100	20	95%	460s	0.2%	90s	0.4%	47s	2.2%
	10^4	10^3	100	30	95%	1055s	0.3%	290s	0.5%	55s	2.6%
μ	10^4	10^3	100	5	20%	107.4s	0.5%	25.1s	0.7%	6.2s	0.01%
	10^4	10^3	100	5	50%	92.6s	0.2%	19.2s	0.4%	8.9s	0.03%
	10^4	10^3	100	5	80%	70.4s	0.1%	15.7s	0.1%	14s	0.7%
	10^4	10^3	100	5	95%	56.7s	0.1%	12.0s	0.1%	18s	2.5%
	10^4	10^3	100	5	95%	56.7s	0.1%	12.0s	0.1%	18s	2.5%
	10^5	10^3	100	10	95%	1095s	0.1%	84s	0.1%	203s	0.8%
	10^5	10^4	200	10	95%	14506s	0.1%	560s	0.2%	2430s	0.4%
	10^5	10^5	200	10	95%	N/A	N/A	3170s	0.2%	25707s	0.2%
	10^6	10^4	500	20	95%	N/A	N/A	6516s	0.2%	N/A	N/A

Table 1: Comparison of fastImpute and IMC on synthetic data. N/A means the algorithm did not complete running in 20 hours, corresponding to 72000 seconds.

- fastImpute significantly improves the scaling behavior of projImpute in n and m, and reduces the running time by over 10 times on real-world datasets.

- For small n, m, fastImpute scales roughly linearly in n and is not sensitive towards m. Eventually it scales as $O(nm)$ when the matrix filling step dominates, as predicted. IMC scales similarly with slightly worse accuracy.

- Both algorithms roughly exhibit linear scaling in p, but IMC retrieves a matrix that has much higher MAPE than fastImpute.

- fastImpute roughly shows the $O(k^3)$ dominant behavior as expected, while IMC seems to scale linearly in k.

- IMC's running time increases with more missing data while it decreases for fastImpute. Both algorithms achieve roughly the same performance, with IMC dropping significantly as the number of missing entries increased.

5.2 Real-World Experiments

For real-world experiments, we utilize the Netflix Prize Dataset. This dataset was released in a competition to predict ratings of customers on unseen movies, given over 10 million ratings scattered across $500,000$ people and $16,000$ movies. Thus, when presented in a matrix $\boldsymbol{A}$ where A_{ij} represents the rating of individual i on movie j, the goal is to complete the matrix $\boldsymbol{A}$ under a low-rank assumption.

For this experiment, we included movies where people who had at least 5 ratings present. This gives a matrix of $471,268$ people and $14,538$ movies. To observe the scalability of fastImpute, we created five data sets (in similar format to Bertsimas and Li (2018)):

1. Base - $\boldsymbol{A}_1$ has dimensions $3,923 \times 103$.

2. Small - $\boldsymbol{A}_2$ has dimensions $18,227 \times 323$.

3. Medium - $\boldsymbol{A}_3$ has dimensions $96,601 \times 788$.

4. Large - $\boldsymbol{A}_4$ has dimensions $471,268 \times 1760$.

5. Full - $\boldsymbol{A}$ has dimensions $471,268 \times 14,538$.

These sizes are constructed such that the total number of elements in $\boldsymbol{A}$ in the successive sizes are approximately different by approximately an order of magnitude.

The feature matrix $\boldsymbol{B}$ is constructed using data from the TMDB Database, and covers 59 features that measure geography, popularity, top actors/actresses, box office, runtime, genre and more. The full list of 59 features is contained in Appendix C.

For comparison, we test against IMC. We split the training set in 80%/20%, where the latter group is used for validation of the rank in IMC and fastImpute. We then report the time taken, T, the MAPE, and the optimal chosen rank k^* for each algorithm: We see that fastImpute is able to outperform IMC on the Netflix dataset across the different n and m values, while enjoying competitive scalability.

n	m	p	$\mu\%$	**fastImpute**			**IMC**		
				T	k^*	**MAPE**	T	k^*	**MAPE**
3,923	103	59	92.6%	1.7s	5	32.9%	0.8s	5	34.1%
18,227	323	59	94.8%	11	6	28.0%	7.5s	6	29.0%
96,601	788	59	94.2%	75s	7	25.7%	49s	8	28.5%
471,268	1,760	59	93.6%	460s	8	22.9%	870s	10	24.1%
471,268	14,538	59	94.1%	2934s	8	20.7%	7605s	10	21.0%

Table 2: Comparison of methods on Netflix data for fastImpute.

6. fastImpute without Side Information

In this section, we explore fastImpute in the special case where there is no side information ($p = m$, and $\boldsymbol{B} = \boldsymbol{I}_m$). In such case, the objective function becomes

$$c(\boldsymbol{S}) = \frac{1}{nm} \sum_{i=1}^{n} \overline{\boldsymbol{a}}_i \left(\boldsymbol{I}_m - \boldsymbol{S} \left(\boldsymbol{S}^T \boldsymbol{W}_i \boldsymbol{S} \right)^{-1} \boldsymbol{S}^T \right) \overline{\boldsymbol{a}}_i^T.$$

This special cases give rises to two additional optimizations:

- Define $d = \min\{m, n\}$. Since the problem is now symmetric in n and m, instead of requiring $O(nk \log(n))$ samples, we only require $O(dk \log(d))$ as we can similarly perform the matrix completion on $\boldsymbol{A}^T$.
- The multiplications involving $\boldsymbol{B}$ are no longer needed as $\boldsymbol{B}$ is the identity matrix and thus does not contribute.

With these two observations, we can update our corollary about the computational complexity of the gradient update step:

Corollary 5 *For $m = p$, and $\boldsymbol{B} = \boldsymbol{I}_m$, the computational complexity of Step 10 in Algorithm 2 is*

$$O\left(\frac{k^3 d \log(d)}{\alpha} \right).$$

Using this further optimization, we compare fastImpute with multiple general matrix completion algorithms on both synthetic datasets and real-world datasets to explore its performance and scaling behavior.

6.1 Synthetic Data Experiments

For synthetic data experiments, we assume that the underlying matrix satisfies the form $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{S}^T$, where $\boldsymbol{U} \in \mathbb{R}^{n \times k}$, $\boldsymbol{S} \in \mathbb{R}^{m \times k}$. Then the elements of $\boldsymbol{U}$ and $\boldsymbol{S}$ are selected from a uniform distribution of $[0, 1]$, where a fraction μ is missing. We report statistics on various combinations of (m, n, k, μ).

The algorithms tested are:

- **fastImpute**: We use the sampling parameters:

  $$m_0 = m, \qquad n_0 = \max\left\{\frac{nk\log(n)}{4m_0\alpha}, 100\right\}.$$

  with $t_{max} = 50$, and $\theta = \frac{\pi}{64}$, and regularization parameter $\gamma = 10^6$. We explicitly stress here that no parameter tuning is done on fastImpute as we intend to show that the algorithm is not parameter sensitive. We implement our algorithm in Julia 0.6 with only the base packages.

- **softImpute-ALS (SIALS)**: Developed by Hastie et al. (2015), this is widely recognized as a state-of-the-art matrix completion method without feature information. It has among the best scaling behavior across all classes of matrix completion algorithms as it utilizes fast alternating least squares to achieve scalability. For each combination of (m, n, μ, k), we tune the regularization parameter λ by imputing random matrices with such combination and find the λ that gives the best results. We utilize the implementation in the softImpute package in R for testing.

- **softImpute-SVD (SISVD)**: Developed by Mazumder et al. (2010), this is the original softImpute algorithm that utilizes truncated SVDs and spectral regularization to impute the matrix. This method is used as a fast benchmark for SVD-type methods. For each combination of (m, n, μ, k), we tune the regularization parameter λ by imputing random matrices with such combination and find the λ that gives the best results. We utilize the implementation in the softImpute package in R for testing.

- **Matrix Factorization Stochastic Gradient Descent (MFSGD)**: This is a popular stochastic gradient descent algorithm (discussed in Jin et al. (2016)) which separates $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{V}$, where $\boldsymbol{U} \in \mathbb{R}^{n\times k}$ and $\boldsymbol{V} \in \mathbb{R}^{k\times m}$), and perform gradient updates for $\boldsymbol{U}$ and $\boldsymbol{V}$. We utilize the implementation in the Fancyimpute package of python that utilizes Tensorflow and the latest available speed optimizations for such algorithm.

To further illustrate the favorable scaling behavior of fastImpute, we compare fastImpute against *online* matrix completion algorithms. Online matrix completion algorithms complete the matrix by updating the factorization $\boldsymbol{U}$ and $\boldsymbol{V}$ with sequential data input from $\boldsymbol{A}$ over time. By their sequential nature, they are usually much faster than offline matrix completion algorithms which consider all the data jointly. The specific online algorithm we compare to is:

- **GROUSE**: This is a popular and efficient online matrix completion algorithm based on sequential gradient updates on the Grassmann manifold, as set out in Balzano et al.. We utilize the official implementation in MATLAB along with the latest optimizations in MATLAB R2020a.

We note that online matrix completion is usually used in different settings than offline algorithms and such comparison is only to note the favorable scaling behavior of fastImpute.

All of the algorithms are executed on a server with 16 CPU cores. Each combination (m, n, k, μ) was ran 10 times, and we report the average value of every statistic. The statistics reported are as followed:

- n, m - the dimensions of $\boldsymbol{A}$.
- k - the true number of features.
- μ - The fraction of missing entries in $\boldsymbol{A}$.
- T - the total time of algorithm execution.
- MAPE - the Mean Absolute Percentage Error (MAPE) for the retrieved matrix $\hat{\boldsymbol{A}}$:
$$\text{MAPE} = \frac{1}{nm}\sum_{i=1}^{n}\sum_{j=1}^{m}\frac{|\hat{A}_{ij} - A_{ij}|}{|A_{ij}|}.$$

The results are separated into sections in Table 3. The first four sections investigate fastImpute's scalability with respect to each of the 4 parameters m, n, μ, k, with the parameter under investigation denoted in the leftmost column.The final section of the results compares the different algorithms' performance on large realistic combinations of (m, n, p, k).

We see that on the final set of large realistic combinations, fastImpute outperforms all comparison algorithms in all cases. Table 4 records the average difference in time and MAPE between fastImpute and the other algorithms, on the final set of combinations. On average fastImpute takes 10% of the time of comparison while achieving $\sim 40 - 70\%$ reduction in MAPE at the same time.

Even when compared to the online algorithm GROUSE, fastImpute is on average over 80% faster on the final set of combinations.

For scaling behavior, we have the following observations:

- n, m - We see that fastImpute scales sublinearly for low n, m as the gradient descent step dominates, and as we move to $n \sim 10^6$ it starts to scale as $O(nm)$ as the step of completing the final matrix starts to dominate. The MAPE steadily decreases as we have more entries. In contrast, SIALS, SISVD, and MFSGD all roughly scale linearly with n, m from the start. Interestingly, the MFSGD algorithm has increasing error with increasing number of entries - we hypothesize this may be due to the gradient descent in factorized form $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{V}^T$ failing to capture non-linear dynamics of the interactions between $\boldsymbol{U}$ and $\boldsymbol{V}$ at high levels.
- k - Somewhat surprisingly, fastImpute scales as $O(k)$ even though theoretically it scales at $O(k^3)$. We believe this is due to the small constant factor in front of the k^2 and k^3 terms.

	n	m	k	μ%	fastImpute		SIALS		SISVD		MFSGD		GROUSE	
					T	MAPE	T	MAPE	T	MAPE	T	MAPE	T	MAPE
n	10^3	10^3	5	95%	0.9s	3.5%	2.3s	19.9%	213s	21.3%	9.4s	2.1%	1.2s	5.0%
	10^4	10^3	5	95%	4.2s	2.4%	25.5s	12.7%	1780s	17.5%	72.1s	2.8%	9.6s	2.6%
	10^5	10^3	5	95%	18.9s	2.2%	443s	8.1%	23650s	12.1%	709s	6.1%	153s	3.1%
	10^6	10^3	5	95%	104s	2.1%	6270s	6.7%	N/A	N/A	7605s	7.0%	1480s	2.7%
m	10^4	10^3	5	95%	4.2s	2.4%	25.5s	12.7%	1780s	17.5%	72.1s	2.8%	9.6s	2.6%
	10^4	10^4	5	95%	19s	4.0%	227s	6.2%	15070s	8.9%	840s	5.4%	75s	5.1%
	10^4	10^5	5	95%	140s	3.1%	3170s	9.1%	N/A	N/A	8010s	7.5%	748s	3.6%
	10^4	10^6	5	95%	1052s	3.5%	30542s	8.0%	N/A	N/A	N/A	N/A	6309s	3.1%
k	10^4	10^3	5	95%	4.2s	2.4%	25.5s	12.7%	1780s	17.5%	72.1s	2.8%	9.6s	2.6%
	10^4	10^3	10	95%	14.1s	2.8%	41.0s	9.8%	3120s	11.7%	80.7s	2.6%	17.6s	2.5%
	10^4	10^3	20	95%	29.8s	3.7%	80.4s	8.2%	3609s	14.6%	83.7s	5.9%	80.2s	1.9%
	10^4	10^3	30	95%	49.5s	5.0%	122s	9.3%	3670s	20.9%	82.1s	4.9%	196s	1.1%
μ	10^4	10^3	5	20%	8.4s	1.1%	3.6s	0.6%	204s	0.6%	140s	4.9%	21s	1.9%
	10^4	10^3	5	50%	7.5s	1.6%	6.7s	1.0%	370s	1.1%	109s	1.2%	15.8s	2.1%
	10^4	10^3	5	80%	5.7s	2.4%	13.1s	4.0%	1340s	2.7%	85s	1.9%	12.5s	2.4%
	10^4	10^3	5	95%	4.2s	2.6%	25.5s	12.7%	1780s	17.5%	72.1s	2.8%	9.6s	2.6%
	10^4	10^3	5	95%	4.2s	2.5%	25.5s	12.7%	1780s	17.5%	72.1s	2.8%	9.6s	2.6%
	10^5	10^3	10	95%	29.0s	2.5%	403s	10.7%	25049s	14.6%	708s	6.0%	270s	2.9%
	10^5	10^4	10	95%	317s	2.1%	4470s	8.9%	N/A	N/A	8215s	7.3%	2076s	2.4%
	10^5	10^5	10	95%	3260s	2.0%	52690s	4.1%	N/A	N/A	N/A	N/A	27043s	2.2%
	10^6	10^4	20	95%	5070s	1.9%	N/A	N/A	N/A	N/A	N/A	N/A	48740s	2.9%

Table 3: Comparison of fastImpute, SIALS, SISVD, MFSGD, and GROUSE on synthetic data. *N/A* means the algorithm did not complete running in 20 hours, corresponding to 72000 seconds.

	SIALS		SISVD		MFSGD		GROUSE	
	ΔT	ΔMAPE	ΔT	ΔMAPE	ΔT	ΔMAPE	ΔT	ΔMAPE
fastImpute vs.	−90%	−71%	−99%	−95%	−96%	−45%	−81%	−15%

Table 4: Average performance of fastImpute, SIALS, SISVD, MFSGD and GROUSE on synthetic trials. Percentages are computed by averaging over the set of realistic combinations.

- μ - We see that in accordance to the linear dependence on $|\Omega|$, the number of known entries, fastImpute runs slower as we have more filled elements, while in contrast SIALS and SISVD both run faster. MFSGD is similar to fastImpute in that it runs slower with more filled elements (a construct of the gradient descent method).

6.2 Real-World Experiments

For real-world experiments, we again utilize the Netflix datasets created in Section 5.2 without the feature matrix.

$\boldsymbol{n}$	$\boldsymbol{m}$	$\boldsymbol{\mu}$%	**fastImpute**			**SIALS**		
			$\boldsymbol{T}$	$\boldsymbol{k^*}$	**MAPE**	$\boldsymbol{T}$	$\boldsymbol{k^*}$	**MAPE**
3,923	103	92.6%	2s	5	23.5%	4s	5	30.6%
18,227	323	94.8%	18s	8	19.8%	47s	7	27.5%
96,601	788	94.2%	109s	10	18.2%	620s	10	24.0%
471,268	1,760	93.6%	370s	10	16.5%	2837s	12	22.5%
471,268	14,538	94.1%	2098s	12	13.8%	38256s	14	20.1%

Table 5: Comparison of methods on Netflix data for fastImpute

We test against SIALS as it is the only algorithm capable of scaling to such size. We split the training set in 80%/20%, where the latter group is used for validation of the rank in SIALS and fastImpute. We then report the time taken, T, the MAPE, and the optimal chosen rank k^* for each algorithm: We see that fastImpute is able to outperform SIALS on the Netflix dataset across the different n and m values, while enjoying superior scalability especially as we approach the full matrix.

7. Conclusion

In conclusion, we have designed a unified optimization framework that is able to conduct state-of-the-art matrix completion with and without side information. Using the factorization approach $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{S}^T\boldsymbol{B}^T$, we wrote $\boldsymbol{U} = f(\boldsymbol{S})$ as a function of $\boldsymbol{S}$, and derived the cost and gradient expressions with respect to $\boldsymbol{S}$ through a separable reformulation of the problem. By then conducting non-convex gradient descent on $\boldsymbol{S}$, our synthetic and real-world data experiments show the competitiveness of the method in both scalability and accuracy against a multitude of comparison algorithms.

References

Laura Balzano, Robert Nowak, and Benjamin Recht. Online identification and tracking of subspaces from highly incomplete information. In *2010 48th Annual allerton conference on communication, control, and computing (Allerton)*, pages 704–711. IEEE.

Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1):183–202, 2009.

Dimitri P Bertsekas. Nonlinear programming. *Journal of the Operational Research Society*, 48(3):334–334, 1997.

Dimitris Bertsimas and Martin S Copenhaver. Characterization of the equivalence of robustification and regularization in linear and matrix regression. *European Journal of Operational Research*, 270(3):931–942, 2018.

Dimitris Bertsimas and Michael Lingzhi Li. Interpretable matrix completion: A discrete optimization approach. *arXiv preprint arXiv:1812.06647*, 2018.

Stephen Boyd, Laurent El Ghaoui, Eric Feron, and Venkataramanan Balakrishnan. *Linear matrix inequalities in system and control theory*, volume 15. Siam, 1994.

Emmanuel J Candes and Yaniv Plan. Matrix completion with noise. *Proceedings of the IEEE*, 98(6):925–936, 2010.

Emmanuel J Candès and Benjamin Recht. Exact matrix completion via convex optimization. *Foundations of Computational mathematics*, 9(6):717, 2009.

Emmanuel J Candès and Terence Tao. The power of convex relaxation: Near-optimal matrix completion. *IEEE Transactions on Information Theory*, 56(5):2053–2080, 2010.

Yudong Chen and Martin J Wainwright. Fast low-rank estimation by projected gradient descent: General statistical and algorithmic guarantees. *arXiv preprint arXiv:1509.03025*, 2015.

Yudong Chen, Srinadh Bhojanapalli, Sujay Sanghavi, and Rachel Ward. Coherent matrix completion. In *International Conference on Machine Learning*, pages 674–682, 2014a.

Yudong Chen, Ali Jalali, Sujay Sanghavi, and Huan Xu. Clustering partially observed graphs via convex optimization. *The Journal of Machine Learning Research*, 15(1):2213–2238, 2014b.

Kai-Yang Chiang, Cho-Jui Hsieh, Nagarajan Natarajan, Inderjit S Dhillon, and Ambuj Tewari. Prediction and clustering in signed networks: a local to global perspective. *The Journal of Machine Learning Research*, 15(1):1177–1213, 2014.

Kai-Yang Chiang, Cho-Jui Hsieh, and Inderjit S Dhillon. Matrix completion with noisy side information. In *Advances in Neural Information Processing Systems*, pages 3447–3455, 2015.

Hadi Daneshmand, Jonas Kohler, Aurelien Lucchi, and Thomas Hofmann. Escaping saddles with stochastic gradients. *arXiv preprint arXiv:1803.05999*, 2018.

Cong Fang, Zhouchen Lin, and Tong Zhang. Sharp analysis for nonconvex sgd escaping from saddle points. *arXiv preprint arXiv:1902.00247*, 2019.

Maha R Farhat, B Jesse Shapiro, Karen J Kieser, Razvan Sultana, Karen R Jacobson, Thomas C Victor, Robin M Warren, Elizabeth M Streicher, Alistair Calver, Alex Sloutsky, et al. Genomic analysis identifies targets of convergent positive selection in drug-resistant mycobacterium tuberculosis. *Nature genetics*, 45(10):1183, 2013.

Rong Ge, Furong Huang, Chi Jin, and Yang Yuan. Escaping from saddle points—online stochastic gradient for tensor decomposition. In *Conference on Learning Theory*, pages 797–842, 2015.

Rong Ge, Jason D Lee, and Tengyu Ma. Matrix completion has no spurious local minimum. In *Advances in Neural Information Processing Systems*, pages 2973–2981, 2016.

Rong Ge, Chi Jin, and Yi Zheng. No spurious local minima in nonconvex low rank problems: A unified geometric analysis. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1233–1242. JMLR. org, 2017.

David Gross and Vincent Nesme. Note on sampling without replacing from a finite collection of matrices. *arXiv preprint arXiv:1001.2738*, 2010.

Trevor Hastie, Rahul Mazumder, Jason D Lee, and Reza Zadeh. Matrix completion and low-rank svd via fast alternating least squares. *The Journal of Machine Learning Research*, 16(1):3367–3402, 2015.

Prateek Jain and Inderjit S Dhillon. Provable inductive matrix completion. *arXiv preprint arXiv:1306.0626*, 2013.

Prateek Jain and Praneeth Netrapalli. Fast exact matrix completion with finite samples. In *Conference on Learning Theory*, pages 1007–1034, 2015.

Prateek Jain, Raghu Meka, and Inderjit S Dhillon. Guaranteed rank minimization via singular value projection. In *Advances in Neural Information Processing Systems*, pages 937–945, 2010.

Hui Ji, Chaoqiang Liu, Zuowei Shen, and Yuhong Xu. Robust video denoising using low rank matrix completion. In *Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, 2010.

Chi Jin, Sham M Kakade, and Praneeth Netrapalli. Provable efficient online matrix completion via non-convex stochastic gradient descent. In *Advances in Neural Information Processing Systems*, pages 4520–4528, 2016.

Raghunandan H Keshavan, Sewoong Oh, and Andrea Montanari. Matrix completion from a few entries. In *Information Theory, 2009. ISIT 2009. IEEE International Symposium on*, pages 324–328. IEEE, 2009.

Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, pages 30–37, 2009.

Jin Lu, Guannan Liang, Jiangwen Sun, and Jinbo Bi. A sparse interactive model for matrix completion with side information. In *Advances in neural information processing systems*, pages 4071–4079, 2016.

Cong Ma, Kaizheng Wang, Yuejie Chi, and Yuxin Chen. Implicit regularization in nonconvex statistical estimation: Gradient descent converges linearly for phase retrieval, matrix completion, and blind deconvolution. *Foundations of Computational Mathematics*, pages 1–182, 2019.

Rahul Mazumder, Trevor Hastie, and Robert Tibshirani. Spectral regularization algorithms for learning large incomplete matrices. *Journal of machine learning research*, 11(Aug): 2287–2322, 2010.

Nagarajan Natarajan and Inderjit S Dhillon. Inductive matrix completion for predicting gene–disease associations. *Bioinformatics*, 30(12):i60–i68, 2014.

Ivan Nazarov, Boris Shirokikh, Maria Burkina, Gennady Fedonin, and Maxim Panov. Sparse group inductive matrix completion. *arXiv preprint arXiv:1804.10653*, 2018.

Sahand Negahban and Martin J Wainwright. Restricted strong convexity and weighted matrix completion: Optimal bounds with noise. *Journal of Machine Learning Research*, 13(May):1665–1697, 2012.

Yurii E Nesterov. A method for solving the convex programming problem with convergence rate o (1/kˆ 2). In *Dokl. akad. nauk Sssr*, volume 269, pages 543–547, 1983.

Benjamin Recht and Christopher Ré. Parallel stochastic gradient algorithms for large-scale matrix completion. *Mathematical Programming Computation*, 5(2):201–226, 2013.

Vatsal Shah, Nikhil Rao, and Weicong Ding. Matrix factorization with side and higher order information. *stat*, 1050:4, 2017.

Si Si, Kai-Yang Chiang, Cho-Jui Hsieh, Nikhil Rao, and Inderjit S Dhillon. Goal-directed inductive matrix completion. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1165–1174. ACM, 2016.

Akshay Soni, Troy Chevalier, and Swayambhoo Jain. Noisy inductive matrix completion under sparse factor models. *arXiv preprint arXiv:1609.03958*, 2016.

Gilbert W Stewart. Matrix perturbation theory. 1990.

Ruoyu Sun and Zhi-Quan Luo. Guaranteed matrix completion via non-convex factorization. *IEEE Transactions on Information Theory*, 62(11):6535–6579, 2016.

Jared Tanner and Ke Wei. Normalized iterative hard thresholding for matrix completion. *SIAM Journal on Scientific Computing*, 35(5):S104–S125, 2013.

Joel A Tropp. User-friendly tail bounds for sums of random matrices. *Foundations of computational mathematics*, 12(4):389–434, 2012.

Max A Woodbury. Inverting modified matrices. *Memorandum report*, 42(106):336, 1950.

Miao Xu, Rong Jin, and Zhi-Hua Zhou. Speedup matrix completion with side information: Application to multi-label learning. In *Advances in neural information processing systems*, pages 2301–2309, 2013.

Qinqing Zheng and John Lafferty. Convergence analysis for rectangular matrix completion using burer-monteiro factorization and gradient descent. *arXiv preprint arXiv:1605.07051*, 2016.

Appendix A. Proof of Theorem 3

For the case where we have no side information, our objective function is:

$$c(\boldsymbol{S}) = \frac{1}{nm}\sum_{i=1}^{n} \overline{\boldsymbol{a}}_i \left(\boldsymbol{I}_m - \boldsymbol{S}\left(\boldsymbol{S}^T\boldsymbol{W}_i\boldsymbol{S}\right)^{-1}\boldsymbol{S}^T\right)\overline{\boldsymbol{a}}_i^T.$$

We first outline the roadmap to prove the two results required. As a reminder, a function $f : \mathbb{R}^d \to \mathbb{R}$ is (θ, ζ, η)-strict saddle if for every $\boldsymbol{x}$, at least one of the following hold for constants $\theta, \zeta, \eta > 0$:

1. $\|\nabla f(\boldsymbol{x})\| \geq \theta$.
2. $\lambda_{\min}(\nabla^2 f(\boldsymbol{x})) \leq -\zeta$.
3. $\boldsymbol{x}$ is η-close to a local minimum.

We want to prove that our objective function $c(\boldsymbol{S})$ is a (θ, ζ, η)-strict saddle. Furthermore, we want to prove that there are no spurious local minima.

To do so, we claim that proving the following statement would imply both results.

"For all $\boldsymbol{S}$ such that $\|\nabla c(\boldsymbol{S})\| \leq \epsilon$ and $\boldsymbol{S}$ more than η-away from a *global* minimum, we have that $\lambda_{\min}(\nabla^2 f(\boldsymbol{x})) \leq -\zeta$ for some suitable ζ."

By construction, this statement shows that for $c(\boldsymbol{S})$, if Condition 1 and 3 of the strict saddle definition is not true, then 2 is always true, which implies $c(\boldsymbol{S})$ is (θ, ζ, η)-strict saddle.

Furthermore, this statement implies that every stationary point (which satisfies $\|\nabla c(\boldsymbol{S})\| \leq \epsilon$) that is not a global minimum has a negative eigenvalue in its Hessian. Thus, every stationary point that is not a global minimum is a saddle point, since local minima have no negative eigenvalues in the Hessian. Therefore, there are no spurious local minima - every local minimum is a global minimum.

Thus, proving this statement would give us the two required results we need. Therefore, we now set out to prove the statement above.

To ease the notation burden, we define the following :

$$\begin{aligned}
\boldsymbol{S}_i &= \boldsymbol{W}_i\boldsymbol{S}, \\
\boldsymbol{P}_{\boldsymbol{S}} &= \boldsymbol{I}_m - \boldsymbol{S}(\boldsymbol{S}^T\boldsymbol{S})^{-1}\boldsymbol{S}^T, \\
\boldsymbol{P}_{\boldsymbol{S}_i} &= \boldsymbol{W}_i - \boldsymbol{S}_i(\boldsymbol{S}_i^T\boldsymbol{S}_i)^{-1}\boldsymbol{S}_i^T.
\end{aligned}$$

Note that the projection matrices are setup so that $\boldsymbol{P}_{\boldsymbol{S}_i}\boldsymbol{S}_i = \boldsymbol{0}$ and $\boldsymbol{P}_{\boldsymbol{S}}\boldsymbol{S} = \boldsymbol{0}$. Furthermore, let us denote $\boldsymbol{U}^* \in \mathbb{R}^{n\times k}, \boldsymbol{S}^* \in \mathbb{R}^{m\times k}$ as the true solution to the matrix completion problem (i.e., $\boldsymbol{A} = \boldsymbol{U}^* \boldsymbol{S}^{*T}$). Then we can write

$$c(\boldsymbol{S}) = \frac{1}{nm} \sum_{i=1}^{n} \boldsymbol{a}_i \boldsymbol{P}_{\boldsymbol{S}_i} \overline{\boldsymbol{a}}_i^T. \tag{A1}$$

We calculate its gradient as follows.

Lemma 4

$$\nabla c(\boldsymbol{S}) = \frac{1}{nm} \sum_{i=1}^{n} \boldsymbol{P}_{\boldsymbol{S}_i} \boldsymbol{a}_i^T \boldsymbol{a}_i \boldsymbol{S}_i (\boldsymbol{S}_i^T \boldsymbol{S}_i)^{-1}.$$

Proof This follows from direct calculation of the derivative. In particular, we note that $\boldsymbol{S}^T \boldsymbol{P}_{\boldsymbol{S}_i} = \boldsymbol{0}$ for all i by definition, and thus *every term* of the derivative in the sum is orthogonal to $\boldsymbol{S}$. ■

To calculate the Hessian, note that we expressed $\boldsymbol{S}$ as a $m \times k$ matrix, so the Hessian has dimensions $(m \times k) \times (m \times k)$. Therefore, to calculate the second derivative in a specific direction $\boldsymbol{M}$, we use the following notation to represent the quadratic form:

$$\boldsymbol{M} : \nabla^2 c(\boldsymbol{S}) : \boldsymbol{M} = \sum_{i,j,k,l} M_{ij} \nabla^2 c(\boldsymbol{S})_{ijkl} M_{kl}.$$

Where $\nabla^2 c(\boldsymbol{S})_{ijkl} = \frac{\partial^2 c(\boldsymbol{S})}{\partial \boldsymbol{S}_{ij} \partial \boldsymbol{S}_{kl}}$.

Lemma 5 *For any matrix $\boldsymbol{M} \in \mathbb{R}^{m \times k}$, the Hessian in the direction of $\boldsymbol{M}$ is*

$$\begin{aligned} \boldsymbol{M} : \nabla^2 c(\boldsymbol{S}) : \boldsymbol{M} = \frac{2}{nm} \sum_{i=1}^{n} \Big(& \boldsymbol{a}_i \boldsymbol{S}_i (\boldsymbol{S}_i^T \boldsymbol{S}_i)^{-1} \boldsymbol{M}^T \boldsymbol{P}_{\boldsymbol{S}_i} \boldsymbol{M} (\boldsymbol{S}_i^T \boldsymbol{S}_i)^{-1} \boldsymbol{a}_i^T \\ & + 2 \boldsymbol{a}_i \boldsymbol{S}_i (\boldsymbol{S}_i^T \boldsymbol{S}_i)^{-1} \boldsymbol{M}^T \boldsymbol{S}_i (\boldsymbol{S}_i^T \boldsymbol{S}_i)^{-1} \boldsymbol{M}^T \boldsymbol{P}_{\boldsymbol{S}_i} \boldsymbol{a}_i^T \\ & - \overline{\boldsymbol{a}}_i \boldsymbol{M} (\boldsymbol{S}_i^T \boldsymbol{S}_i)^{-1} \boldsymbol{M}^T \boldsymbol{P}_{\boldsymbol{S}_i} \boldsymbol{a}_i^T \\ & + \boldsymbol{a}_i \boldsymbol{S}_i (\boldsymbol{S}_i^T \boldsymbol{S}_i)^{-1} \boldsymbol{S}_i^T \boldsymbol{M} (\boldsymbol{S}_i^T \boldsymbol{S}_i)^{-1} \boldsymbol{M}^T \boldsymbol{P}_{\boldsymbol{S}_i} \boldsymbol{a}_i^T \Big). \end{aligned}$$

Proof The result follows through by direct calculation. ■

As a reminder, our goal is to prove that, given $\|\nabla c(\boldsymbol{S})\| \leq \epsilon$ and $\|\boldsymbol{S} - \boldsymbol{S}^*\| \geq \eta$ for any global minima $\boldsymbol{S}^*$, there exists a negative eigenvalue for the Hessian $\nabla^2 c(\boldsymbol{S})$.

Note the existence of a negative eigenvalue for the Hessian $\nabla^2 c(\boldsymbol{S})$ is equivalent to existence of a matrix $\boldsymbol{M}$ such that $\boldsymbol{M} : \nabla^2 c(\boldsymbol{S}) : \boldsymbol{M} < 0$. Now let us look at the expression for the second derivative.

$$\begin{aligned}\boldsymbol{M} : \nabla^2 c(\boldsymbol{S}) : \boldsymbol{M} = \frac{2}{nm} \sum_{i=1}^{n} \Big(& \underbrace{\boldsymbol{a}_i \boldsymbol{S}_i (\boldsymbol{S}_i^T \boldsymbol{S}_i)^{-1} \boldsymbol{M}^T \boldsymbol{P}_{\boldsymbol{S}_i} \boldsymbol{M} (\boldsymbol{S}_i^T \boldsymbol{S}_i)^{-1} \boldsymbol{S}_i^T \boldsymbol{a}_i^T}_{\text{(T1)}} \\ & + \underbrace{2 \boldsymbol{a}_i \boldsymbol{S}_i (\boldsymbol{S}_i^T \boldsymbol{S}_i)^{-1} \boldsymbol{M}^T \boldsymbol{S}_i (\boldsymbol{S}_i^T \boldsymbol{S}_i)^{-1} \boldsymbol{M}^T \boldsymbol{P}_{\boldsymbol{S}_i} \boldsymbol{a}_i^T}_{\text{(T2)}} \\ & - \underbrace{\overline{\boldsymbol{a}}_i \boldsymbol{M} (\boldsymbol{S}_i^T \boldsymbol{S}_i)^{-1} \boldsymbol{M}^T \boldsymbol{P}_{\boldsymbol{S}_i} \boldsymbol{a}_i^T}_{\text{(T3)}} \\ & + \underbrace{\boldsymbol{a}_i \boldsymbol{S}_i (\boldsymbol{S}_i^T \boldsymbol{S}_i)^{-1} \boldsymbol{S}_i^T \boldsymbol{M} (\boldsymbol{S}_i^T \boldsymbol{S}_i)^{-1} \boldsymbol{M}^T \boldsymbol{P}_{\boldsymbol{S}_i} \boldsymbol{a}_i^T}_{\text{(T4)}} \Big).\end{aligned}$$

We first prove that for *any* $\boldsymbol{M}$, the terms (T2) and (T4) are small. Then, we exhibit a specific $\boldsymbol{M}$, such that (T1) is close to 0, (T3) is large and positive, and so that the resulting quadratic form $\boldsymbol{M} : \nabla^2 c(\boldsymbol{S}) : \boldsymbol{M}$ is negative.

Now let us prove that terms (T2) and (T4) are small (on the order of $O(\epsilon)$) for any $\boldsymbol{M}$.

Lemma 6 *Let $\boldsymbol{S}$ satisfy Assumptions 1-2, and be such that $\|\nabla c(\boldsymbol{S})\| \le \epsilon$. Then we have that, for $r > \frac{Ck}{m}$ for sufficiently large C, with probability at least $1 - O(\frac{1}{n})$:*

$$\left\| \frac{1}{nm} \sum_{i=1}^{n} \boldsymbol{a}_i \boldsymbol{S}_i (\boldsymbol{S}_i^T \boldsymbol{S}_i)^{-1} \boldsymbol{M}^T \boldsymbol{S}_i (\boldsymbol{S}_i^T \boldsymbol{S}_i)^{-1} \boldsymbol{M}^T \boldsymbol{P}_{\boldsymbol{S}_i} \boldsymbol{a}_i^T \right\| \le K r \epsilon \|\boldsymbol{M}\|^2,$$
$$\left\| \frac{1}{nm} \sum_{i=1}^{n} \boldsymbol{a}_i \boldsymbol{S}_i (\boldsymbol{S}_i^T \boldsymbol{S}_i)^{-1} \boldsymbol{S}_i^T \boldsymbol{M} (\boldsymbol{S}_i^T \boldsymbol{S}_i)^{-1} \boldsymbol{M}^T \boldsymbol{P}_{\boldsymbol{S}_i} \boldsymbol{a}_i^T \right\| \le L r \epsilon \|\boldsymbol{M}\|^2,$$

for some absolute constants K, L.

Proof Note that we are trying to bound the second derivative when the first derivative is approximately 0. Using Lemma 4, we can rewrite $\|\nabla c(\boldsymbol{S})\| \le \epsilon$ as

$$\left\| \frac{1}{nm} \sum_{i=1}^{n} \boldsymbol{a}_i \boldsymbol{S}_i (\boldsymbol{S}_i^T \boldsymbol{S}_{\boldsymbol{i}})^{-1} \boldsymbol{M}^T \boldsymbol{P}_{\boldsymbol{S}_i} \boldsymbol{a}_i^T \right\| \le \epsilon \|\boldsymbol{M}\|. \tag{A2}$$

We only prove for term (T2) as the proof for term (T4) follows the same steps. First, we bound the difference between the first derivative in (A2) and its "non-stochastic" version:

$$\left\| \frac{1}{nm} \sum_{i=1}^{n} \boldsymbol{a}_i \boldsymbol{S}_i (\boldsymbol{S}_i^T \boldsymbol{S}_i)^{-1} \boldsymbol{M}^T \boldsymbol{P}_{\boldsymbol{S}_i} \boldsymbol{a}_i^T - \frac{r}{nm} \sum_{i=1}^{n} \boldsymbol{a}_i \boldsymbol{S} (\boldsymbol{S}^T \boldsymbol{S})^{-1} \boldsymbol{M}^T \boldsymbol{P}_{\boldsymbol{S}} \boldsymbol{a}_i^T \right\|. \tag{A3}$$

Then we focus on each individual term inside the sum (taking $\frac{1}{m}$ inside). We have

$$\begin{aligned}
&\left\|\frac{\boldsymbol{a}_i\boldsymbol{S}_i(\boldsymbol{S}_i^T\boldsymbol{S}_i)^{-1}\boldsymbol{M}^T\boldsymbol{P}_{\boldsymbol{S}_i}\boldsymbol{a}_i^T}{m} - \frac{l}{m}\boldsymbol{a}_i\boldsymbol{S}(\boldsymbol{S}^T\boldsymbol{S})^{-1}\boldsymbol{M}^T\boldsymbol{P}_{\boldsymbol{S}}\boldsymbol{a}_i^T\right\| \\
&\leq r\left\|\left(\frac{\boldsymbol{a}_i\boldsymbol{S}_i}{l} - \frac{\boldsymbol{a}_i\boldsymbol{S}}{m}\right)\left(\frac{\boldsymbol{S}_i^T\boldsymbol{S}_i}{l}\right)^{-1}\frac{\boldsymbol{M}^T\boldsymbol{P}_{\boldsymbol{S}_i}\boldsymbol{a}_i^T}{l}\right\| \\
&+ r\left\|\frac{\boldsymbol{a}_i\boldsymbol{S}}{m}\left(\left(\frac{\boldsymbol{S}_i^T\boldsymbol{S}_i}{l}\right)^{-1} - \left(\frac{\boldsymbol{S}^T\boldsymbol{S}}{m}\right)^{-1}\right)\frac{\boldsymbol{M}^T\boldsymbol{P}_{\boldsymbol{S}_i}\boldsymbol{a}_i^T}{l}\right\| \\
&+ r\left\|\frac{\boldsymbol{a}_i\boldsymbol{S}}{m}\left(\frac{\boldsymbol{S}^T\boldsymbol{S}}{m}\right)^{-1}\left(\frac{\boldsymbol{M}^T\boldsymbol{P}_{\boldsymbol{S}_i}\boldsymbol{a}_i^T}{l} - \frac{\boldsymbol{M}^T\boldsymbol{P}_{\boldsymbol{S}}\boldsymbol{a}_i^T}{m}\right)\right\|.
\end{aligned}$$

We utilize Hoeffding's theorem for the first and third term, along with Lemma 11 for the second term to obtain that with probability $1-\delta$ the sum of the three terms is less than or equal to $r\sqrt{\frac{Dk\log(\frac{1}{\delta})}{l}}\|\boldsymbol{M}\|$, where D is an absolute constant.

Then, we have, by Lemma 10, with probability $1-\delta$

$$\begin{aligned}
&\left\|\frac{1}{nm}\sum_{i=1}^n \boldsymbol{a}_i\boldsymbol{S}_i(\boldsymbol{S}_i^T\boldsymbol{S}_i)^{-1}\boldsymbol{M}^T\boldsymbol{P}_{\boldsymbol{S}_i}\boldsymbol{a}_i^T - \frac{l}{nm}\sum_{i=1}^n \boldsymbol{a}_i\boldsymbol{S}(\boldsymbol{S}^T\boldsymbol{S})^{-1}\boldsymbol{M}^T\boldsymbol{P}_{\boldsymbol{S}}\boldsymbol{a}_i^T\right\| \\
&\leq r\sqrt{\frac{D'k\log(\frac{1}{\delta})}{ln}}\|\boldsymbol{M}\|,
\end{aligned}$$

where D' is an absolute constant. We let $\delta = \frac{1}{n}$, and $l \geq Ck$ for some constant C. Then, for n sufficiently large, we have

$$r\sqrt{\frac{D'k\log(\frac{1}{\delta})}{ln}}\|\boldsymbol{M}\| \leq r\epsilon\|\boldsymbol{M}\|.$$

Therefore, we know that, with probability at least $1-\frac{1}{n}$, we have

$$\left\|\frac{l}{nm}\sum_{i=1}^n \boldsymbol{a}_i\boldsymbol{S}(\boldsymbol{S}^T\boldsymbol{S})^{-1}\boldsymbol{M}^T\boldsymbol{P}_{\boldsymbol{S}}\boldsymbol{a}_i^T\right\| \leq 2r\epsilon\|\boldsymbol{M}\|, \tag{A4}$$

for any $\boldsymbol{M}$. Now we do the same for term (T2) by bounding the difference between term (T2) and its non-stochastic variant. The details are omitted as they follow the same logic as above. The final result is that with probability at least $1-\frac{1}{n}$, we have

$$\begin{aligned}
&\| \underbrace{\frac{1}{nm}\sum_{i=1}^n \boldsymbol{a}_i\boldsymbol{S}_i(\boldsymbol{S}_i^T\boldsymbol{S}_i)^{-1}\boldsymbol{M}^T\boldsymbol{S}_i(\boldsymbol{S}_i^T\boldsymbol{S}_i)^{-1}\boldsymbol{M}^T\boldsymbol{P}_{\boldsymbol{S}_i}\boldsymbol{a}_i^T - \frac{l}{nm}\sum_{i=1}^n \boldsymbol{a}_i\boldsymbol{S}(\boldsymbol{S}^T\boldsymbol{S})^{-1}\boldsymbol{M}^T\boldsymbol{S}(\boldsymbol{S}^T\boldsymbol{S})^{-1}\boldsymbol{M}^T\boldsymbol{P}_{\boldsymbol{S}}\boldsymbol{a}_i^T}_{\text{(S1)}} \| \leq r\epsilon\|\boldsymbol{M}\|^2.
\end{aligned} \tag{A5}$$

Now let us focus on the term (S1). Such term is exactly (A4) substituting $\boldsymbol{M}$ with $\boldsymbol{M}^T\boldsymbol{S}(\boldsymbol{S}^T\boldsymbol{S})^{-1}\boldsymbol{M}^T$. Therefore, we have

$$\left\| \frac{l}{nm} \sum_{i=1}^{n} \boldsymbol{a}_i \boldsymbol{S}(\boldsymbol{S}^T\boldsymbol{S})^{-1}\boldsymbol{M}^T\boldsymbol{S}(\boldsymbol{S}^T\boldsymbol{S})^{-1}\boldsymbol{M}^T\boldsymbol{P}_{\boldsymbol{S}}\boldsymbol{a}_i^T \right\| \leq Kr\epsilon\|\boldsymbol{M}\|^2, \tag{A6}$$

for some constant K. Then combining (A5) and (A6) gives the required result. ■

We next utilize the bounds on (T2) and (T4) to explicitly construct a $\boldsymbol{M}$ to bound (T1) and (T3). Specifically, we define $\boldsymbol{M} = \boldsymbol{z}^T\boldsymbol{x}$, $\|\boldsymbol{M}\| = 1$, where $\boldsymbol{z}^T \in \mathbb{R}^{m\times 1}$ is a non-zero (rescaled) column of $\boldsymbol{P}_{\boldsymbol{S}}\boldsymbol{S}^*$ (it exists because $\boldsymbol{S}$ is not optimal, so $\boldsymbol{P}_{\boldsymbol{S}}\boldsymbol{S}^* \neq \boldsymbol{0}$) and $\boldsymbol{x}^T \in \mathbb{R}^{k\times 1}$ is an eigenvector of $\boldsymbol{S}^{*T}\boldsymbol{S}(\boldsymbol{S}^T\boldsymbol{S})^{-1}$ that has an eigenvalue of size $O(\epsilon)$. We utilize the row vector convention to be consistent with our other notations. We prove the existence of the eigenvector $\boldsymbol{x}^T$ in Lemma 7 below.

Lemma 7 *Let $\boldsymbol{S}$ satisfy Assumptions 1-2, and be such that $\|\nabla c(\boldsymbol{S})\| \leq \epsilon$ for some ϵ sufficiently small. Assume that $\min_{\boldsymbol{R}^T\boldsymbol{R}=\boldsymbol{I}} \|\boldsymbol{S} - \boldsymbol{S}^*\boldsymbol{R}\| \geq \eta$. Then $\boldsymbol{S}^{*T}\boldsymbol{S}(\boldsymbol{S}^T\boldsymbol{S})^{-1}$ has an eigenvector with an eigenvalue of size $O(\epsilon)$.*

Proof First it is easy to see that

$$\min_{\boldsymbol{R}^T\boldsymbol{R}=\boldsymbol{I}} \|\boldsymbol{S}^* - \boldsymbol{S}\boldsymbol{R}\| = \|\boldsymbol{P}_{\boldsymbol{S}}\boldsymbol{S}^*\| \geq \eta. \tag{A7}$$

Now, let us consider the first derivative bound from (A4):

$$\left\| \frac{1}{nm} \sum_{i=1}^{n} \boldsymbol{a}_i \boldsymbol{S}(\boldsymbol{S}^T\boldsymbol{S})^{-1}\boldsymbol{M}^T\boldsymbol{P}_{\boldsymbol{S}}\boldsymbol{a}_i^T \right\| \leq 2r\epsilon\|\boldsymbol{M}\|.$$

We prove that given this, there exists an eigenvector of $\boldsymbol{S}^{*T}\boldsymbol{S}(\boldsymbol{S}^T\boldsymbol{S})^{-1}$ that has an eigenvalue of size $O(\epsilon)$. Assume that there is not. By definition $\boldsymbol{a}_i = \boldsymbol{u}_i^*\boldsymbol{S}^{*T}$, so we can write the bound as:

$$\left\| \frac{1}{nm} \sum_{i=1}^{n} \boldsymbol{u}_i^*\boldsymbol{S}^{*T}\boldsymbol{S}(\boldsymbol{S}^T\boldsymbol{S})^{-1}\boldsymbol{M}^T\boldsymbol{P}_{\boldsymbol{S}}\boldsymbol{S}^*\boldsymbol{u}_i^{*T} \right\| \leq 2r\epsilon\|\boldsymbol{M}\|. \tag{A8}$$

Then since there is not an eigenvalue of magnitude $O(\epsilon)$, let the smallest eigenvalue of $\boldsymbol{S}^{*T}\boldsymbol{S}(\boldsymbol{S}^T\boldsymbol{S})^{-1}$ have a magnitude of $O(\gamma) \gg O(\epsilon)$. Let $\boldsymbol{x}^T \in \mathbb{R}^{k\times 1}$ be the eigenvector associated with such eigenvalue. Then $\boldsymbol{S}^{*T}\boldsymbol{S}(\boldsymbol{S}^T\boldsymbol{S})^{-1}\boldsymbol{x}^T = O(\gamma)\boldsymbol{x}^T$. From Equation (A7) we know that $\|\boldsymbol{P}_{\boldsymbol{S}}\boldsymbol{S}^*\| \geq \eta$, so let $\boldsymbol{y} \in \mathbb{R}^{1\times m}$ be such that $\boldsymbol{y}\boldsymbol{P}_{\boldsymbol{S}}\boldsymbol{S}^* = O(\eta)\boldsymbol{v}^T$ where $\boldsymbol{v} \in \mathbb{R}^{1\times k}$, and rescaled such that for $\boldsymbol{M} = \boldsymbol{y}^T\boldsymbol{x}$, we have $\|\boldsymbol{M}\| = O(1)$. Then we have

$$\left\| \frac{1}{nm} \sum_{i=1}^{n} \boldsymbol{u}_i^*\boldsymbol{S}^{*T}\boldsymbol{S}(\boldsymbol{S}^T\boldsymbol{S})^{-1}\boldsymbol{M}^T\boldsymbol{P}_{\boldsymbol{S}}\boldsymbol{S}^*\boldsymbol{u}_i^{*T} \right\| = \left\| \frac{O(\eta\gamma)}{nm} \sum_{i=1}^{n} \boldsymbol{u}_i^*\boldsymbol{x}\boldsymbol{v}^T\boldsymbol{u}_i^{*T} \right\| = O\left(\frac{\eta\gamma}{m}\right) \leq O(\epsilon),$$

where the last equality follows from the submatrix full-rank property of $\boldsymbol{U}^*$ (note that $\boldsymbol{u}_i^* \in \mathbb{R}^{1\times k}$ are row vectors here). Since η is independent from ϵ (η only depends on how far $\boldsymbol{S}$ is from $\boldsymbol{S}^*$) and $\gamma \gg \epsilon$ by setup, we obtain a contradiction for sufficiently small ϵ.

Therefore, there exists ϵ sufficiently small such that, there is an eigenvector of $\boldsymbol{S}^{*T}\boldsymbol{S}(\boldsymbol{S}^T\boldsymbol{S})^{-1}$, $\boldsymbol{x}^T$, that has an eigenvalue of size $O(\epsilon)$. ■

Having proved that the desired matrix $\boldsymbol{M}$ does indeed exist, we show that indeed with such $\boldsymbol{M}$, (T1) is small, and (T3) is large and positive. We first bound (T1).

Lemma 8 *Let $\boldsymbol{S}$ satisfy Assumptions 1-2, and be such that $\|\nabla c(\boldsymbol{S})\| \le \epsilon$ for some ϵ sufficiently small. Assume that $\min_{\boldsymbol{R}^T\boldsymbol{R}=\boldsymbol{I}} \|\boldsymbol{S} - \boldsymbol{S}^*\boldsymbol{R}\| \ge \eta$. Then for $\boldsymbol{M} = \boldsymbol{z}^T\boldsymbol{x}$ where $\boldsymbol{z}$, $\boldsymbol{x}$ defined above, with probability at least $1 - O(\frac{1}{n})$:*

$$\left\| \frac{1}{nm}\sum_{i=1}^n \boldsymbol{a}_i\boldsymbol{S}_i(\boldsymbol{S}_i^T\boldsymbol{S}_i)^{-1}\boldsymbol{M}^T\boldsymbol{P}_{\boldsymbol{S}_i}\boldsymbol{M}(\boldsymbol{S}_i^T\boldsymbol{S}_i)^{-1}\boldsymbol{S}_i^T\boldsymbol{a}_i^T \right\| \le O(r\epsilon),$$

Proof From Lemma 7, we know that there is an eigenvector of $\boldsymbol{S}^{*T}\boldsymbol{S}(\boldsymbol{S}^T\boldsymbol{S})^{-1}$, $\boldsymbol{x}^T$, that has an eigenvalue of size $O(\epsilon)$. Similar to the proof for Lemma 6, we can bound term (T1) with the non-stochastic version with at least probability $1 - \frac{1}{n}$:

$$\begin{aligned}&\Bigg\| \frac{1}{nm}\sum_{i=1}^n \boldsymbol{a}_i\boldsymbol{S}_i(\boldsymbol{S}_i^T\boldsymbol{S}_i)^{-1}\boldsymbol{M}^T\boldsymbol{P}_{\boldsymbol{S}_i}\boldsymbol{M}(\boldsymbol{S}_i^T\boldsymbol{S}_i)^{-1}\boldsymbol{S}_i^T\boldsymbol{a}_i^T \\ &\underbrace{- \frac{r}{n}\sum_{i=1}^n \boldsymbol{a}_i\boldsymbol{S}(\boldsymbol{S}^T\boldsymbol{S})^{-1}\boldsymbol{M}^T\boldsymbol{P}_{\boldsymbol{S}}\boldsymbol{M}(\boldsymbol{S}^T\boldsymbol{S})^{-1}\boldsymbol{S}^T\boldsymbol{a}_i^T}_{\text{(S2)}} \Bigg\| \le r\epsilon\|\boldsymbol{M}\|^2. \end{aligned} \tag{A9}$$

Let us look at each term in the sum (S2). Taking $\boldsymbol{M} = \boldsymbol{z}^T\boldsymbol{x}$. We have

$$\begin{aligned}&\|\boldsymbol{a}_i\boldsymbol{S}(\boldsymbol{S}^T\boldsymbol{S})^{-1}\boldsymbol{M}^T\boldsymbol{P}_{\boldsymbol{S}}\boldsymbol{M}(\boldsymbol{S}^T\boldsymbol{S})^{-1}\boldsymbol{S}^T\boldsymbol{a}_i^T\| \\ &= \|\boldsymbol{P}_{\boldsymbol{S}}\boldsymbol{M}(\boldsymbol{S}^T\boldsymbol{S})^{-1}\boldsymbol{S}^T\boldsymbol{a}_i^T\|^2 \\ &= \|\boldsymbol{P}_{\boldsymbol{S}}\boldsymbol{M}(\boldsymbol{S}^T\boldsymbol{S})^{-1}\boldsymbol{S}^T\boldsymbol{S}^*\boldsymbol{u}_i^{*T}\|^2 \\ &= \|\boldsymbol{P}_{\boldsymbol{S}}\boldsymbol{z}^T\boldsymbol{x}(\boldsymbol{S}^T\boldsymbol{S})^{-1}\boldsymbol{S}^T\boldsymbol{S}^*\boldsymbol{u}_i^{*T}\|^2 \\ &= O(\epsilon^2)\|\boldsymbol{z}^T\boldsymbol{x}\boldsymbol{u}_i^{*T}\|^2 \\ &= O(\epsilon^2),\end{aligned}$$

where in the last equality we used the identity that $\boldsymbol{P_S z}^T = \boldsymbol{z}^T$. Then, we can bound the sum (S2) as

$$\begin{aligned}
&\left\| \frac{r}{n} \sum_{i=1}^{n} \boldsymbol{a}_i \boldsymbol{S}(\boldsymbol{S}^T\boldsymbol{S})^{-1}\boldsymbol{M}^T\boldsymbol{P_S}\boldsymbol{M}(\boldsymbol{S}^T\boldsymbol{S})^{-1}\boldsymbol{S}^T\boldsymbol{a}_i^T \right\| \\
&\leq \frac{r}{n} \sum_{i=1}^{n} \|\boldsymbol{a}_i \boldsymbol{S}(\boldsymbol{S}^T\boldsymbol{S})^{-1}\boldsymbol{M}^T\boldsymbol{P_S}\boldsymbol{M}(\boldsymbol{S}^T\boldsymbol{S})^{-1}\boldsymbol{S}^T\boldsymbol{a}_i^T\| \\
&= O(r\epsilon^2).
\end{aligned}$$

Substituting such bound on equation (S2) into (A9), we have that

$$\left\| \frac{1}{nm} \sum_{i=1}^{n} \boldsymbol{a}_i \boldsymbol{S}_i(\boldsymbol{S}_i^T\boldsymbol{S}_i)^{-1}\boldsymbol{M}^T\boldsymbol{P}_{\boldsymbol{S}_i}\boldsymbol{M}(\boldsymbol{S}_i^T\boldsymbol{S}_i)^{-1}\boldsymbol{S}_i^T\boldsymbol{a}_i^T \right\| \leq O(r\epsilon).$$

As here we have $\|\boldsymbol{M}\| = O(1)$. ■

Now, finally, we prove that (T3) is large under such $\boldsymbol{M}$.

Lemma 9 *Let $\boldsymbol{S}$ satisfy Assumptions 1-2. Further assume that $\min_{\boldsymbol{R}^T\boldsymbol{R}=\boldsymbol{I}} \|\boldsymbol{S}^* - \boldsymbol{SR}\| \geq \eta$ and $\|\nabla c(\boldsymbol{S})\| \leq \epsilon$ for some ϵ sufficiently small. Then for $\boldsymbol{M} = \boldsymbol{z}^T\boldsymbol{x}$ where $\boldsymbol{z}$, $\boldsymbol{x}$ defined above, with probability at least $1 - O(\frac{1}{n})$:*

$$\frac{1}{nm} \sum_{i=1}^{n} \boldsymbol{a}_i \boldsymbol{M}(\boldsymbol{S}_i^T\boldsymbol{S}_i)^{-1}\boldsymbol{M}^T\boldsymbol{P}_{\boldsymbol{S}_i}\boldsymbol{a}_i^T \geq O\left(\frac{r\eta^2}{k}\right),$$

Proof First it is easy to see that

$$\min_{\boldsymbol{R}^T\boldsymbol{R}=\boldsymbol{I}} \|\boldsymbol{S}^* - \boldsymbol{SR}\| = \|\boldsymbol{P_S}\boldsymbol{S}^*\| \geq \eta. \tag{A10}$$

We consider the term $\frac{1}{nm}\sum_{i=1}^{n} \boldsymbol{a}_i\boldsymbol{M}(\boldsymbol{S}^T\boldsymbol{S})^{-1}\boldsymbol{M}^T\boldsymbol{P_S}\boldsymbol{a}_i^T$. We have that

$$\begin{aligned}
&\frac{l}{nm} \sum_{i=1}^{n} \boldsymbol{a}_i \boldsymbol{M}(\boldsymbol{S}^T\boldsymbol{S})^{-1}\boldsymbol{M}^T\boldsymbol{P_S}\boldsymbol{a}_i^T \\
&= \frac{l}{nm} \sum_{i=1}^{n} \boldsymbol{a}_i \boldsymbol{z}^T\boldsymbol{x}(\boldsymbol{S}^T\boldsymbol{S})^{-1}\boldsymbol{x}^T\boldsymbol{z}\boldsymbol{a}_i^T,
\end{aligned}$$

where we utilized $\boldsymbol{zP_S} = \boldsymbol{z}$. By definition $\boldsymbol{a}_i^T = \boldsymbol{S}^*\boldsymbol{u}_i^{*T}$, so we have

$$\frac{l}{nm} \sum_{i=1}^{n} \boldsymbol{a}_i \boldsymbol{z}^T\boldsymbol{x}(\boldsymbol{S}^T\boldsymbol{S})^{-1}\boldsymbol{x}^T\boldsymbol{z}\boldsymbol{a}_i^T = \frac{r}{n} \sum_{i=1}^{n} \boldsymbol{u}_i^*\boldsymbol{S}^{*T}\boldsymbol{z}^T\boldsymbol{x}(\boldsymbol{S}^T\boldsymbol{S})^{-1}\boldsymbol{x}^T\boldsymbol{z}\boldsymbol{S}^*\boldsymbol{u}_i^{*T}.$$

Since $(\boldsymbol{S}^T\boldsymbol{S})^{-1}$ is positive definite (PD), we have a sum of quadratic forms over a PD matrix, which is positive as long as $\boldsymbol{x}^T\boldsymbol{z}\boldsymbol{S}^*\boldsymbol{u}_i^{*T}$ is non-zero. Since we have that $\boldsymbol{z}$ is a non-zero column vector of $\boldsymbol{P_S S^*}$, and we know that $\|(\boldsymbol{P_S S^*})^T \boldsymbol{S}^*\| = \|\boldsymbol{S}^{*T} \boldsymbol{P_S S}^*\|^2 \geq O(\eta^2)$, so in particular, we have that $\|\boldsymbol{x}^T \boldsymbol{z S}^*\| \geq O(\eta^2)$. Thus, we only need $\boldsymbol{u}_i^{*T}$ to not always be in the null space of $\boldsymbol{x}^T \boldsymbol{z S}^*$. Now by the sub-matrix full rank condition of $\boldsymbol{A}$, we know that $\boldsymbol{u}_i^*$ is not in the null space of $\boldsymbol{x}^T \boldsymbol{z S}^*$ for $O(n)$ indices. Therefore, we have that $\|\boldsymbol{x}^T \boldsymbol{z S}^* \boldsymbol{u}_i^{*T}\| = O(\eta^2)$. Then, we have

$$\frac{r}{n} \sum_{i=1}^{n} \boldsymbol{u}_i^* \boldsymbol{S}^{*T} \boldsymbol{z}^T \boldsymbol{x} (\boldsymbol{S}^T \boldsymbol{S})^{-1} \boldsymbol{x}^T \boldsymbol{z S}^* \boldsymbol{u}_i^{*T} \geq \frac{r}{\sigma_{max}(\boldsymbol{S})} C\eta^2,$$

for some constant C and where $\sigma_{max}(\boldsymbol{S})$ is the largest singular value of $\boldsymbol{S}$. By the nuclear-Frobenius norm inequality, we have

$$\|\boldsymbol{S}^T \boldsymbol{S}\|_* \leq \text{Rank}(\boldsymbol{S}^T \boldsymbol{S}) \|\boldsymbol{S}\|^2 = k.$$

Therefore, the maximum singular value of $\boldsymbol{S}$ is at most k, resulting in the bound

$$\frac{r}{\sigma_{max}(\boldsymbol{S})} C\eta^2 \geq \frac{Cr\eta^2}{k}. \tag{A11}$$

By a similar proof to what is done in Lemma 6, we can show that with probability greater than $1 - O(\frac{1}{n})$ we have

$$\left\| \frac{1}{nm} \sum_{i=1}^{n} \boldsymbol{a}_i \boldsymbol{P_S S}^* (\boldsymbol{S}^T \boldsymbol{S})^{-1} \boldsymbol{S}^{*T} \boldsymbol{P_S a}_i^T - \frac{l}{nm} \sum_{i=1}^{n} \overline{\boldsymbol{a}}_i \boldsymbol{P_S S}^* (\boldsymbol{S}^T \boldsymbol{S})^{-1} \boldsymbol{S}^{*T} \boldsymbol{P_S P}_{\boldsymbol{S}_i} \boldsymbol{a}_i^T \right\| \leq \frac{Cr\eta^2}{2k}. \tag{A12}$$

Combining (A12) with (A11) proves the lemma. ■

Finally, we combine the results of Lemmas 6, 8 and 9 to prove that for $\boldsymbol{M} = \boldsymbol{z}^T \boldsymbol{x}$, $\|\boldsymbol{M}\| = 1$ defined previously, we have

$$\begin{aligned}
\boldsymbol{M} : \nabla^2 c(\boldsymbol{S}) : \boldsymbol{M} = \frac{2}{nm} \sum_{i=1}^{n} \Bigg(& \boldsymbol{a}_i \boldsymbol{S}_i (\boldsymbol{S}_i^T \boldsymbol{S}_i)^{-1} \boldsymbol{M}^T \boldsymbol{P}_{\boldsymbol{S}_i} \boldsymbol{M} (\boldsymbol{S}_i^T \boldsymbol{S}_i)^{-1} \boldsymbol{a}_i^T \\
& + 2\boldsymbol{a}_i \boldsymbol{S}_i (\boldsymbol{S}_i^T \boldsymbol{S}_i)^{-1} \boldsymbol{M}^T \boldsymbol{S}_i (\boldsymbol{S}_i^T \boldsymbol{S}_i)^{-1} \boldsymbol{M}^T \boldsymbol{P}_{\boldsymbol{S}_i} \boldsymbol{a}_i^T \\
& - \overline{\boldsymbol{a}}_i \boldsymbol{M} (\boldsymbol{S}_i^T \boldsymbol{S}_i)^{-1} \boldsymbol{M}^T \boldsymbol{P}_{\boldsymbol{S}_i} \boldsymbol{a}_i^T \\
& + \boldsymbol{a}_i \boldsymbol{S}_i (\boldsymbol{S}_i^T \boldsymbol{S}_i)^{-1} \boldsymbol{S}_i^T \boldsymbol{M} (\boldsymbol{S}_i^T \boldsymbol{S}_i)^{-1} \boldsymbol{M}^T \boldsymbol{P}_{\boldsymbol{S}_i} \boldsymbol{a}_i^T \Bigg) \\
& \leq O(r\epsilon) + O(r\epsilon) - O\left(\frac{r\eta^2}{k}\right) + O(r\epsilon) \\
& = -O\left(\frac{r\eta^2}{k}\right),
\end{aligned}$$

as long as $\epsilon \leq O(\frac{\eta^2}{k})$. Therefore, for $\|\nabla c(\boldsymbol{S})\| \leq \epsilon$ and $\boldsymbol{S}$ at least $\eta = O(\sqrt{\epsilon})$ away from a global minimum, we have $\lambda_{min}(\nabla^2 c(\boldsymbol{S})) \leq -O(\frac{r\epsilon}{k})$. Thus, the desired statement

"For all $\boldsymbol{S}$ such that $\|\nabla c(\boldsymbol{S})\| \leq \epsilon$ and $\boldsymbol{S}$ more than η-away from a *global* minimum, we have that $\lambda_{\min}(\nabla^2 f(\boldsymbol{x})) \leq -\zeta$ for some suitable ζ."

is true for $\eta = O(\sqrt{\epsilon})$ and $\zeta = O(\frac{r\epsilon}{k})$, with probability $1 - O(\frac{1}{n})$.

Thus, all local minima are global minima, and in particular the function $c(\boldsymbol{S})$ satisfies a $(\epsilon, O(\frac{r\epsilon}{k}), O(\sqrt{\epsilon}))$-strict saddle for sufficiently small ϵ.

Appendix B. Proof of Technical Lemmas

Lemma 10 *Let $X_1, \cdots, X_n$ be independent (but not necessarily identically distributed) random variables which satisfy the following:*

$$\mathbb{P}\left(|X_i - a_i| \geq \sigma_i \sqrt{\log(\frac{1}{\epsilon})}\right) \leq \epsilon.$$

Then we have

$$\mathbb{P}\left(\left|\frac{\sum_{i=1}^n X_i - a_i}{n}\right| \geq \frac{2\sqrt{\sum_{i=1}^n \sigma_i^2}}{n}\sqrt{\log(\frac{1}{4\epsilon})}\right) \leq \epsilon. \tag{A13}$$

Proof See Bertsimas and Li (2018). ■

Lemma 11 *Assume that $\boldsymbol{S}$ satisfies Assumption 2. Then we have*

$$\mathbb{P}\left(\left\|\left(\frac{\boldsymbol{S}^T\boldsymbol{S}}{m}\right)^{-1} - \left(\frac{\boldsymbol{S}_i^T\boldsymbol{S}_i}{l}\right)^{-1}\right\| \leq \sqrt{\frac{D\log(\frac{1}{\epsilon})}{l}}\right) \geq 1 - \epsilon. \tag{A14}$$

Proof This follows immediately from Lemma 13 and the following lemma from matrix perturbation theory (for proof, see e.g. Stewart (1990)):

Lemma 12 *Let $\boldsymbol{A}, \boldsymbol{B}$ be invertible matrices and let $\boldsymbol{B} = \boldsymbol{A} + \boldsymbol{\Delta}$. Then, we have the following bound:*

$$\|\boldsymbol{A}^{-1} - \boldsymbol{B}^{-1}\| \leq \|\boldsymbol{A}^{-1}\|\|\boldsymbol{B}^{-1}\|\|\boldsymbol{\Delta}\|. \tag{A15}$$

■

Lemma 13

$$\mathbb{P}\left(\left\|\frac{\boldsymbol{S}^T\boldsymbol{S}}{m} - \frac{\boldsymbol{S}_i^T\boldsymbol{S}_i}{l}\right\| \leq \sqrt{\frac{C\log(\frac{1}{\epsilon})}{l}}\right) \geq 1 - \epsilon. \tag{A16}$$

To prove this, we first introduce a matrix analog of the well-known Chernoff bound, the proof of which can be found in Tropp (2012):

Lemma 14 *Let $\mathcal{X} \in \mathbb{R}^{k\times k}$ be a finite set of positive-semidefinite matrices, and suppose that*

$$\max_{\boldsymbol{X}\in\mathcal{X}} \lambda_{\max}(\boldsymbol{X}) \leq D,$$

where $\lambda_{\min}/\lambda_{\max}$ is the minimum/maximum eigenvalue function. Sample $\{\boldsymbol{X}_1, \cdots, \boldsymbol{X}_\ell\}$ uniformly at random without replacement. Compute:

$$\mu_{\min} := \ell \cdot \lambda_{\min}(\mathbb{E}\boldsymbol{X}_1) \qquad \mu_{\max} := \ell \cdot \lambda_{\max}(\mathbb{E}\boldsymbol{X}_1).$$

Then:

$$\mathbb{P}\left\{\lambda_{\min}\left(\sum_j \boldsymbol{X}_j\right) \leq (1-\delta)\mu_{\min}\right\} \leq k\cdot\exp\left(\frac{-\delta^2\mu_{\min}}{4D}\right) \quad \textit{for } \delta\in[0,1),$$
$$\mathbb{P}\left\{\lambda_{\max}\left(\sum_j \boldsymbol{X}_j\right) \leq (1+\delta)\mu_{\max}\right\} \leq k\cdot\exp\left(\frac{-\delta^2\mu_{\max}}{4D}\right) \quad \textit{for } \delta\geq 0.$$

Now we proceed with the proof.

Proof (Lemma 13) First, let us write L as the set of indices j such that $(\boldsymbol{W}_i)_{jj} = 1$. Then, we decompose $\boldsymbol{S} = \boldsymbol{QR}$ in a reduced $\boldsymbol{QR}$ factorization, where $\boldsymbol{Q}^T\boldsymbol{Q} = m\boldsymbol{I}$ (so that $\|\boldsymbol{R}\| = O(1)$). Then define $\boldsymbol{Q}_L$ as the $|L|\times k$ submatrix of $\boldsymbol{Q}$ formed with the rows in L. Then we can see that

$$\boldsymbol{S}^T\boldsymbol{S} = \boldsymbol{R}^T\boldsymbol{Q}^T\boldsymbol{QR}, \qquad \boldsymbol{S}_i^T\boldsymbol{S}_i = \boldsymbol{R}^T\boldsymbol{Q}_L^T\boldsymbol{Q}_L\boldsymbol{R}.$$

Now let us decompose the inner parts, which can be written using the rows of $\boldsymbol{Q}$, $\boldsymbol{q}_i \in \mathbb{R}^{1\times k}$:

$$\boldsymbol{Q}^T\boldsymbol{Q} = \sum_{i=1}^m \boldsymbol{q}_i^T\boldsymbol{q}_i,$$
$$\boldsymbol{Q}_L^T\boldsymbol{Q}_L = \sum_{i\in L} \boldsymbol{q}_i^T\boldsymbol{q}_i,$$

where $\boldsymbol{q}_i^T\boldsymbol{q}_i \in \mathbb{R}^{k\times k}$ rank-one positive semi-definite matrices. Therefore, we can take $\boldsymbol{Q}_L^T\boldsymbol{Q}_L$ as a random sample of size l from the set $\mathcal{X} = \{\boldsymbol{q}_i^T\boldsymbol{q}_i\}_{i=1,\cdots,m}$, which satisfies the conditions in Lemma 14 with $D = O(1)$. Furthermore, with $\mathcal{X}$, we observe that we have $\mathbb{E}\boldsymbol{X}_1 = \frac{\boldsymbol{Q}^T\boldsymbol{Q}}{m} = \boldsymbol{I}_k$, so we have

$$\lambda_{\min}(\mathbb{E}\boldsymbol{X}_1) = \lambda_{\max}(\mathbb{E}\boldsymbol{X}_1) = 1.$$

Therefore, we apply Lemma 14 to $\boldsymbol{Q}_L^T\boldsymbol{Q}_L$ and have that

$$\mathbb{P}\left\{\lambda_{\min}\left(\boldsymbol{Q}_L^T\boldsymbol{Q}_L\right) \leq (1-\delta)l\right\} \leq \exp\left(\frac{-\delta^2 l}{D'}\right),$$
$$\mathbb{P}\left\{\lambda_{\max}\left(\boldsymbol{Q}_L^T\boldsymbol{Q}_L\right) \geq (1+\delta)l\right\} \leq \exp\left(\frac{-\delta^2 l}{D'}\right),$$

where we set $D = \frac{D'}{4}$ with $D' = O(1)$. Some rearrangement gives:

$$\mathbb{P}\left\{\lambda_{\min}\left(\frac{\boldsymbol{Q}_L^T\boldsymbol{Q}_L}{l}\right) \geq 1 - \sqrt{\frac{D'\log\left(\frac{2}{\epsilon}\right)}{l}} \text{ and } \lambda_{\max}\left(\frac{\boldsymbol{Q}_L^T\boldsymbol{Q}_L}{l}\right) \leq 1 + \sqrt{\frac{D'\log\left(\frac{2}{\epsilon}\right)}{l}}\right\} \geq 1 - \epsilon, \tag{A17}$$

Now since $\frac{\boldsymbol{Q}^T\boldsymbol{Q}}{m} = \boldsymbol{I}_k$, we have

$$\lambda_{\min}\left(\frac{\boldsymbol{Q}^T\boldsymbol{Q}}{m}\right) = \lambda_{\max}\left(\frac{\boldsymbol{Q}^T\boldsymbol{Q}}{m}\right) = 1. \tag{A18}$$

Combining equation (A18) and (A17) gives:

$$\mathbb{P}\left\{\left\|\frac{\boldsymbol{Q}_L^T\boldsymbol{Q}_L}{l} - \frac{\boldsymbol{Q}^T\boldsymbol{Q}}{m}\right\| \leq \sqrt{\frac{D'\log\left(\frac{2}{\epsilon}\right)}{l}}\right\} \geq 1 - \epsilon. \tag{A19}$$

Then, we have

$$\mathbb{P}\left\{\left\|\frac{\boldsymbol{R}^T\boldsymbol{Q}_L^T\boldsymbol{Q}_L\boldsymbol{R}}{l} - \frac{\boldsymbol{R}^T\boldsymbol{Q}^T\boldsymbol{Q}\boldsymbol{R}}{m}\right\| \leq \|\boldsymbol{R}\|^2\sqrt{\frac{D'\log\left(\frac{2}{\epsilon}\right)}{l}}\right\} \geq 1 - \epsilon. \tag{A20}$$

Taking $C = D'\|\boldsymbol{R}\|^4\log(2)$ gives the required result. ■

Appendix C. List of Features for Netflix Data

- 24 Indicator Variables for Genres: Action, Adventure, Animation, Biography, Comedy, Crime, Documentary, Drama, Family, Fantasy, Film Noir, History, Horror, Music, Musical, Mystery, Romance, Sci-Fi, Short, Sport, Superhero, Thriller, War, Western
- 5 Indicator Variables for Release Date: Within last 10 years, Between 10-20 years, Between 20-30 years, Between 30-40 years, Between 40-50 Years
- 6 Indicator Variables for Top Actors/Actresses defined by their Influence Score at time of release: Top 100 Actors, Top 100 Actresses, Top 250 Actors, Top 250 Actresses, Top 1000 Actors, Top 1000 Actresses
- IMDB Rating
- Number of Reviews
- Total Production Budget
- Total Runtime
- Total Box Office Revenue

- Indicator Variable for whether it is US produced
- 11 Indicator Variables for Month of Year Released (January removed to prevent multicollinearity)
- Number of Original Music Score
- Number of Male Actors
- Number of Female Factors
- 3 Indicator Variables for Film Language: English, French, Japanese
- Constant