OptSpace : A Gradient Descent Algorithm on the Grassman Manifold for Matrix Completion

Raghunandan H. Keshavan and Sewoong Oh

November 3, 2009

Abstract

We consider the problem of reconstructing a low rank matrix from a small subset of its entries. In this paper, we describe the implementation of an efficient algorithm proposed in [19], based on singular value decomposition followed by local manifold optimization, for solving the low-rank matrix completion problem. It has been shown that if the number of revealed entries is large enough, the output of singular value decomposition gives a good estimate for the original matrix, so that local optimization reconstructs the correct matrix with high probability. We present numerical results which show that this algorithm can reconstruct the low rank matrix exactly from a very small subset of its entries. We further study the robustness of the algorithm with respect to noise, and its performance on actual collaborative filtering datasets.

1 Introduction

In this paper we consider the problem of reconstructing an $m \times n$ low rank matrix M from a small set of observed entries. This problem is of considerable practical interest and has many applications. One example is collaborative filtering, where users submit rankings for small subsets of, say, movies, and the goal is to infer the preference of unrated movies for a recommendation system [3]. It is believed that the movie-rating matrix is approximately low-rank, since only a few factors contribute to a users preferences. Other examples of matrix completion include the problem of inferring 3-dimensional structure from motion [12] and triangulation from incomplete data of distances between wireless sensors, also known as the sensor localization problem [28], [26].

1.1 Prior and related work

On the theoretical side, most recent work focuses on algorithms for exactly recovering the unknown low-rank matrix. They provide an upper bound on the number of observed entries that guarantee successful recovery with high probability. The main assumptions of this *exact matrix completion* problem are that the matrix M to be recovered has rank $r \ll m, n$ and that the observed entries are known exactly. The problem is equivalent to finding a minimum rank matrix matching the observed entries . This problem is NP-hard. Adapting techniques from compressed sensing, Candès and Recht introduced a convex relaxation of this problem [9]. They introduced the concept of incoherence and proved that for a matrix M of rank r which has the incoherence property, solving the convex relaxation correctly recovers the unknown matrix, with high probability, if the number of observed entries $|E|$ satisfies, $|E| \geq C(\alpha) r n^{1.2} \log n$.

Recently [19] improved the bound to $|E| \geq C(\alpha) r n \max\{\log n, r\}$ for matrices with bounded condition number and provided an efficient algorithm called OptSpace , based on spectral methods followed by local manifold optimization. For a bounded rank r, it is order optimal in the sense that

an $m \times n$ rank-r matrix M has $r(m+n-r)$ degrees of freedom and without the same number of observations it is impossible to fix them. The extra $\log n$ factor is due to a coupon-collector effect: it is necessary that E contains at least one entry per row and one per column, which happens only for $|E| \geq Cn \log n$ [9, 18]. Candès and Tao proved a similar bound $|E| \geq C(\alpha) nr(\log n)^6$ with a stronger assumption on the original matrix M, known as the *strong incoherence condition* [10] but without any assumption on the condition number of M. For any value of r, it is only suboptimal by a poly-logarithmic factor[1].

When the pattern of observed entries is non-random, it is interesting to ask if the solution of the rank r matrix completion problem is unique [29]. Building on the ideas from rigidity theory, Singer and Cucuringu introduce a randomized algorithm to determine whether it is possible to uniquely complete a partially observed matrix to a matrix of specific rank r. Furthermore, by applying their algorithm to random patterns of observed entries, one can get a lower bound on the minimum number of observed entries necessary to correctly recover the matrix M.

While most theoretical work focuses on proving bounds for the exact matrix completion problem, a more interesting and practical problem is when the matrix M is only approximately low rank or when the observation is corrupted by noise. The main focus of this *approximate matrix completion* problem is to design an algorithm to find an $m \times n$ low-rank matrix $\widehat{M}$ that best approximates the original matrix M and provide a bound on the root mean squared error (RMSE) given by, $\text{RMSE} = \frac{1}{\sqrt{mn}} ||M - \widehat{M}||_F$. Candès and Plan introduced a generalization of the convex relaxation from [9] to the approximate case, and provided a bound on the RMSE [8]. More recently, a bound on the RMSE achieved by the OptSpace algorithm with noisy observations was obtained in [20]. This bound is order optimal in a number of situations and improves over the analogous result in [8].

On the practical side, directly solving the convex relaxation introduced in [9] requires solving a Semidefinite Program (SDP), the complexity of which grows proportional to n^3. In the last year, many authors have proposed efficient algorithms for solving the low-rank matrix completion problem. These include Singular Value Thresholding (SVT) [7], Accelerated Proximal Gradient (APG) algorithm [30], Fixed Point Continuation with Approximate SVD (FPCA) [22], Atomic Decomposition for Minimum Rank Approximation (ADMiRA) [21], Soft-Impute [23], Subspace Evolution and Transfer (SET) [13], Singular Value Projection (SVP) [24] and OptSpace [19].

The problems each of these algorithms are trying to solve are described in Section 2. SVT is an iterative algorithm for solving the convex relaxation of the *exact matrix completion* problem, which minimizes the nuclear norm (a sum of the singular values) under the constraints of matching the observed entries as in (5). APG, FPCA and Soft-Impute are efficient algorithms for solving the convex relaxation of the *approximate matrix completion* problem, which is a nuclear norm regularized least squares problem as in (7). ADMiRA is an extension of Compressive Sampling Matching Pursuit (CoSaMP) [25] , which is an iterative method for solving a least squares problem with bounded rank r as described in (8). SVP is another approach to solve (8), which is a generalization of Iterative Hard Thresholding (IHT) [6].

1.2 Contributions and outline

The main contribution of this paper is to develop and implement efficient procedures, based on the OptSpace algorithm introduced in [19], for solving the exact and approximate matrix completion problems and add novel modifications, namely Rank Estimation and Incremental OptSpace,

[1] In [17, 27, 16], which appeared while we were preparing this manuscript, improved guarantees were proved for the convex relaxation algorithm. Namely, assuming only the incoherence property on the original matrix M, the convex relaxation correctly recovers the matrix M if $|E| \geq C(\alpha) nr(\log n)^2$.

that allow for a broader application and a better performance.

The algorithm described in [19] requires a knowledge of the rank of the original matrix. In the following, we introduce a procedure, RANK ESTIMATION, that is guaranteed to correctly estimate the rank of the original matrix from a partially revealed matrix under some conditions, which in turn allows us to use the algorithm in a broader set of applications.

Next, we introduce INCREMENTAL OPTSPACE, a novel modification to OPTSPACE. We show that, empirically, INCREMENTAL OPTSPACE has substantially better performance than OPTSPACE when the underlying matrix is ill-conditioned.

Further, we carry out an extensive empirical comparison of various reconstruction algorithms. This is particularly important because performance guarantees are only "up to constants" and therefore they have limited use in comparing different algorithms. Finally, we apply our algorithm to real-world data and demonstrate that it is readily applicable to the real data.

The organization of the paper is as follows. In Section 2 we describe the low rank matrix completion problem and convex relaxations to the basic NP-hard approach, mostly to set our notation for later use. Section 3 introduces an efficient implementation of the OPTSPACE algorithm with novel modifications. In Section 4 we discuss the results of numerical simulations with respect to speed and accuracy and compare the performance of OPTSPACE with that of the other algorithms.

2 The model definition

In the case of *exact matrix completion*, we assume that the matrix M has exact low rank $r \ll \min\{m, n\}$ and that the observed entries are known exactly. More precisely, we assume that there exist matrices U of dimensions $m \times r$, V of dimensions $n \times r$, and a diagonal matrix Σ of dimensions $r \times r$, such that

$$M = U\Sigma V^T \,. \tag{1}$$

Notice that for a given matrix M, the factors (U, V, Σ) are not unique.

Out of the $m \times n$ entries of M, a subset $E \subseteq [m] \times [n]$ is observed. Let M^E be the $m \times n$ observed matrix that stores all the observed values, such that

$$M^E_{i,j} = \begin{cases} M_{ij} & \text{if } (i,j) \in E \,, \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

Our goal is to find a low rank estimation $\widehat{M}(M^E, E)$ of the original matrix M from the observed matrix M^E and the set of observed indices E.

If the number of observed entries $|E|$ is large enough, there is a unique rank r matrix which matches the observed entries. In this case, solving the following optimization problem will recover the original matrix correctly.

$$\begin{aligned} \text{minimize} \quad & \operatorname{rank}(X) \\ \text{subject to} \quad & \mathcal{P}_E(X) = \mathcal{P}_E(M) \,, \end{aligned} \tag{3}$$

where $X \in \mathbb{R}^{m \times n}$ is the variable matrix, $\operatorname{rank}(X)$ is the rank of matrix X, and $\mathcal{P}_E(\cdot)$ is the projector operator defined as

$$\mathcal{P}_E(M)_{ij} = \begin{cases} M_{ij} & \text{if } (i,j) \in E, \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

The solution of this problem is the lowest rank matrix that matches the observed entries. Notice that this is optimal in the sense that if this problem does not recover the correct matrix M then there exists at least one other rank-r matrix that matches all the observations, and no other algorithm can do better. However, this optimization problem is NP-hard and all known algorithms require doubly exponential time in n [9]. This is especially inadequate since we are interested in cases where the dimension of the matrix M is large (eg. such as $5 \cdot 10^5 \times 2 \cdot 10^4$ for [3]).

In compressed sensing problems, minimizing the ℓ_1 norm of a vector is used as a convex relaxation of minimizing the ℓ_0 norm, or equivalently minimizing the number of non-zero entries, for sparse signal recovery. We can adopt this idea to matrix completion, where rank $(\cdot)$ of a matrix corresponds to ℓ_0 norm of a vector, and nuclear norm to ℓ_1 norm [9],

$$\begin{aligned} \text{minimize} \quad & ||X||_* \\ \text{subject to} \quad & \mathcal{P}_E(X) = \mathcal{P}_E(M) \ , \end{aligned} \tag{5}$$

where $||X||_*$ denotes the nuclear norm of X, i.e the sum of its singular values.

In the case of *approximate matrix completion* problem, where the observations are contaminated by noise or the original matrix to be reconstructed is only approximately low rank, the constraint $\mathcal{P}_E(X) = \mathcal{P}_E(M)$ must be relaxed. This results in either the problem [8, 22, 30, 23]

$$\begin{aligned} \text{minimize} \quad & ||X||_* \\ \text{subject to} \quad & ||\mathcal{P}_E(X) - \mathcal{P}_E(M)||_F \leq \Theta \ , \end{aligned} \tag{6}$$

or its Lagrangian version

$$\text{minimize} \quad \mu||X||_* + \frac{1}{2}||\mathcal{P}_E(X) - \mathcal{P}_E(M)||_F^2 \ . \tag{7}$$

In [21, 24], problem (3) is recast into the rank-r matrix approximation problem of

$$\begin{aligned} \text{minimize} \quad & ||\mathcal{P}_E(X) - \mathcal{P}_E(M)||_F \\ \text{subject to} \quad & \text{rank}\,(X) \leq r \ . \end{aligned} \tag{8}$$

In the following, we present an efficient algorithm, namely OptSpace, to solve the low-rank matrix completion problem which is closely related to (8), and numerically compare its performance with those of the competing algorithms in the case of exact as well as approximate matrix completion problems.

3 Algorithm

Algorithm 1 describes the overview of OptSpace . Each step is explained in detail in the following sections. The basic idea is to minimize the cost function $F : \mathbb{R}^{m\times r} \times \mathbb{R}^{n\times r} \to \mathbb{R}$, defined as

$$F(X,Y) \equiv \min_{S\in\mathbb{R}^{r\times r}} \mathcal{F}(X,Y,S)\,, \tag{9}$$

$$\mathcal{F}(X,Y,S) \equiv \sum_{(i,j)\in E} f\Big(M_{ij}, (XSY^T)_{ij}\Big)\,. \tag{10}$$

Here $X \in \mathbb{R}^{n\times r}$, $Y \in \mathbb{R}^{m\times r}$ are orthogonal matrices, normalized as $X^TX = m\mathbb{I}$, $Y^TY = n\mathbb{I}$ where $\mathbb{I}$ denotes the identity matrix, and $f : \mathbb{R}\times\mathbb{R} \to \mathbb{R}$ is a element-wise cost function. A common example that is useful in practice is the squared distance $f(x,y) = \frac{1}{2}(x-y)^2$.

Minimizing $F(X, Y)$ is an *a priori* difficult task, since F is a non-convex function. The basic idea is that the singular value decomposition (SVD) of M^E provides an excellent initial guess, and that the minimum can be found with high probability by standard gradient descent after this initialization. Two caveats must be added to this description: (1) In general the matrix M^E must be 'trimmed' to eliminate over-represented rows and columns; (2) We need to estimate the target rank r.

Algorithm 1 : OptSpace

Input: observation matrix M^E, observed set E
Output: estimated matrix $\widehat{M}$

1: Trim M^E, and let $\widetilde{M}^E$ be the output;
2: Estimate the rank of M, and let $\hat{r}$ be the estimation;
3: Compute the rank-$\hat{r}$ projection of $\widetilde{M}^E$, $\mathcal{P}_{\hat{r}}(\widetilde{M}^E) = X_0 S_0 Y_0^T$;
4: Minimize $F(X, Y)$ through gradient descent, with initial condition (X_0, Y_0), and return $\widehat{M} = XSY^T$.

3.1 Trimming

We say that a row is over-represented if its degree is more than $2|E|/m$ (twice the average degree), where degree of a row is defined as the number of observed entries in that row. Analogously, a column is over-represented if its degree is more than $2|E|/n$. Trimming is a procedure that takes M^E and E as input and outputs $\widetilde{M}^E$ by setting to 0 all of the entries in over-represented rows and columns. Let $d_l(i)$ and $d_r(j)$ be the degree of i^{th} row and j^{th} column of M respectively. Then the trimmed matrix $\widetilde{M}^E$ is defined as

$$\widetilde{M}^E_{ij} = \begin{cases} 0 & \text{if } d_l(i) > 2|E|/m \text{ or } d_r(j) > 2|E|/n \ , \\ M^E_{ij} & \text{otherwise.} \end{cases} \tag{11}$$

The trimming step is essential when $|E| = \Theta(n)$, in which case there exists over-represented columns and rows of degrees $\Theta(\log n / \log\log n)$, corresponding to singular values of the order $\Theta(\sqrt{\log n / \log\log n})$. As n grows large, while these spurious singular values dominate the principal components in step 3 of the Algorithm 1, the corresponding singular vectors are highly concentrated on the over-represented rows and columns (respectively for left and right singular vectors) and do not provide any useful information about the unobserved entries of M.

3.2 Estimating the rank

Define $\epsilon \equiv |E|/\sqrt{mn}$. In the case of a square matrix M, ϵ corresponds to the average degree per row or per column. Throughout this paper, the parameter ϵ will be frequently used as the model parameter indicating how difficult the problem instance is.

By singular value decomposition of the trimmed matrix, let

$$\widetilde{M}^E = \sum_{i=1}^{\min(m,n)} \sigma_i x_i y_i^T \ , \tag{12}$$

where x_i and y_i are the left and right singular vectors corresponding to ith singular value σ_i. Then,

the following cost function is defined in terms of the singular values.

$$R(i) = \frac{\sigma_{i+1} + \sigma_1 \sqrt{\frac{i}{\epsilon}}}{\sigma_i} \,. \tag{13}$$

Based on the above definition, RANK ESTIMATION consists of two steps:

RANK ESTIMATION

Input: trimmed observation matrix $\widetilde{M}^E$
Output: estimated rank $\hat{r}$
1: Compute singular values $\{\sigma_i\}$ of $\widetilde{M}^E$;
2: Find the index i that minimizes $R(i)$, and let $\hat{r}$ be the minimizer.

The idea behind this algorithm is that, if enough entries of M are revealed then there is a clear separation between the first r singular values, which reveal the structure of the matrix M to be reconstructed, and the spurious ones [19]. As described in the following proposition, we can show that this simple procedure is guaranteed to reconstruct the correct rank r, with high probability, for $|E|$ large enough. For the proof of this proposition, we refer to Appendix A.

Proposition 3.1. *Assume M to be a rank r $m \times n$ matrix with bounded condition number κ. Then there exists a constant $C(\kappa)$ such that, if $\epsilon > C(\kappa)r$, then* RANK ESTIMATION *correctly estimates the rank r, with high probability.*

3.3 Rank-ρ projection

Rank-ρ projection consists of performing a sparse SVD on $\widetilde{M}^E$ and rescaling the singular values and singular vectors appropriately. From the RANK ESTIMATION step we have the SVD of $\widetilde{M}^E$ in Eq. (12), namely $\widetilde{M}^E = \sum_{i=1}^{\min(m,n)} \sigma_i x_i y_i^T$. Define the projection :

$$\mathcal{P}_\rho(\widetilde{M}^E) = X_0 S_0 Y_0^T \,, \tag{14}$$

for normalized orthogonal matrices $X_0 \in \mathbb{R}^{m\times\rho}$ and $Y_0 \in \mathbb{R}^{n\times\rho}$, and a $\rho \times \rho$ diagonal matrix S_0, defined in terms of the singular values and singular vectors in Eq. (12) as $X_0 = \sqrt{m}[x_1, \ldots, x_\rho]$, $Y_0 = \sqrt{n}[y_1, \ldots, y_\rho]$, and $S_0 = (1/\epsilon)\mathrm{diag}(\sigma_1, \ldots, \sigma_\rho)$. Notice that we do not need to compute the scaled singular values S_0, since we only require X_0 and Y_0 for the following local optimization step. There are a number of low complexity algorithms available for forming a sparse SVD, as well as a number of open source implementations of these algorithms.

3.4 Gradient descent on the Grassman manifold

The MANIFOLD OPTIMIZATION step involves gradient descent with variables $X \in \mathbb{R}^{m\times r}$ and $Y \in \mathbb{R}^{n\times r}$ using the cost function $F(X,Y)$ defined below. In this section, we use r and $\hat{r}$ interchangeably to denote the estimated rank of matrix M.

$$F(X,Y) \;\equiv\; \min_{S\in\mathbb{R}^{r\times r}} \mathcal{F}(X,Y,S)\,, \tag{15}$$

$$\mathcal{F}(X,Y,S) \;\equiv\; \sum_{(i,j)\in E} f\Big(M_{ij}, (XSY^T)_{ij}\Big) + \lambda \sum_{(i,j)\notin E} \frac{1}{2}\big(XSY^T\big)_{ij}^2\,, \tag{16}$$

where $f : \mathbb{R}\times\mathbb{R} \to \mathbb{R}$ is an element-wise cost function. Note that compared to Eq. (10), we have additional term in Eq. (16), which is a regularization term with a regularization coefficient $\lambda \in [0,1]$.

The above general formulation allows for a freedom in choosing a suitable cost function f for different applications. However, a common example of the cost function $f(x,y) = \frac{1}{2}(x-y)^2$ works very well in practice as well as in proving performance bounds [19]. Hence, throughout this paper, we use the squared difference as the cost function, resulting in

$$\mathcal{F}(X,Y,S) \equiv \frac{1}{2}\Big|\Big|\mathcal{P}_E(M - XSY^T)\Big|\Big|_F^2 + \lambda\frac{1}{2}\Big|\Big|\mathcal{P}_{E^\perp}(XSY^T)\Big|\Big|_F^2 ,$$

where the projector operator $\mathcal{P}_E$ for a given E is defined in Eq. (4), and $E^\perp$ is the complementary set of E.

For the results in this paper, we choose $\lambda = 0$ but we observe that using a positive λ helps when the matrix entries are corrupted by noise. For $\lambda = 0$, the gradient of $F(X,Y)$ can be written explicitly as

$$\begin{aligned} \operatorname{grad} F(\mathbf{x})_X &= \mathcal{P}_E(XSY^T - M)YS^T , \\ \operatorname{grad} F(\mathbf{x})_Y &= \mathcal{P}_E(XSY^T - M)^T XS , \end{aligned}$$

where S is the $r \times r$ matrix that achieves the minimum in the definition of $F(X,Y)$, Eq. (15).

One important feature of OptSpace is that $F(X,Y)$ is regarded as a function of the r-dimensional subspaces of $\mathbb{R}^m$ and $\mathbb{R}^n$ generated (respectively) by the columns of X and Y. This interpretation is justified by the fact that $F(X,Y) = F(XA, YB)$ for any two orthogonal matrices A, $B \in \mathbb{R}^{r\times r}$. The set of r dimensional subspaces of $\mathbb{R}^m$ is a differentiable Riemannian manifold $\mathsf{G}(m,r)$ (the Grassman manifold) [19]. The gradient descent algorithm is applied to the function $F : \mathsf{G}(m,r) \times \mathsf{G}(n,r) \to \mathbb{R}$ For further details on optimization by gradient descent on matrix manifolds we refer to [14, 4].

In the following, we use a compact representation $\mathbf{x}$ for a pair (X,Y), with X an $n\times r$ matrix and Y an $m\times r$ matrix. Similarly, the gradient is represented by : $\operatorname{grad} F(\mathbf{x}_k) = (\operatorname{grad} F(\mathbf{x}_k)_X, \operatorname{grad} F(\mathbf{x}_k)_Y)$. Let $\mathbf{x}_0 = (X_0, Y_0)$, where X_0 and Y_0 are the normalized left and right singular matrices from rank-r projection. The Manifold Optimization algorithm starting at $\mathbf{x}_0$ is described below. We refer to [19] for justifications and performance bounds of the algorithm.

For any scalar τ, it is shown in [5] that this algorithm converges to the local minimum. However, numerical experiments suggest $\tau = 10^{-3}$ is a good choice. The algorithm stops when the fit error $||\mathcal{P}_E(M - \widehat{M})||_F / ||\mathcal{P}_E(M)||_F$ goes below some threshold δ_{tol}, e.g. 10^{-6}. The basic idea is that this is a good indicator of the relative error on the whole set, $||M - \widehat{M}||_F / ||M||_F$. This stopping criterion is also used in other algorithms such as SVT in [7] where the authors provide a convincing argument for its use.

MANIFOLD OPTIMIZATION

Input: observed matrix M^E, estimated rank $\hat{r}$, initial factors $\mathbf{x}_0 = (X_0, Y_0)$, tolerance δ_{tol}, maximum iteration count k_{max}, step size τ

Output: reconstructed matrix $\widehat{M}$

1: **For** $k = 0, 1, \ldots, k_{max}$ **do**:
2: Compute $S_k = \arg\min_S \{\mathcal{F}(X_k, Y_k, S)\}$
3: Compute $\mathbf{w}_k = \text{grad}\, F(\mathbf{x}_k)$
4: Set $t_k = \tau$
5: **While** $F(\mathbf{x}_k - t_k \mathbf{w}_k) - F(\mathbf{x}_k) > \frac{1}{2} t_k ||\mathbf{w}_k||^2$, **do**
6: $t_k \leftarrow t_k/2$
7: Set $\mathbf{x}_{k+1} = \mathbf{x}_k - t_k \mathbf{w}_k$
8: **If** $||\mathcal{P}_E(M - \widehat{M})||_F / ||\mathcal{P}_E(M)||_F < \delta_{\text{tol}}$ **then break**
9: **End for**
10: Set $\widehat{M} = X_k S_k Y_k^T$

3.5 A novel modification to OPTSPACE for ill-conditioned matrices

In this section, we describe a novel modification to the OPTSPACE algorithm, which has substantially better performance in the case when the matrix M to be reconstructed is ill-conditioned. When the condition number $\kappa(M)$ is high, the initial guess in step 3 of OPTSPACE for (u_r, v_r), the singular vectors which correspond to the smallest singular value, are often far from the correct ones. To compensate for this discrepancy, we start by first finding (u_1, v_1), the singular vectors corresponding to the first singular value, and incrementally search for the next ones.

Algorithm 2 : INCREMENTAL OPTSPACE

Input: observation matrix M^E, observed set E, tolerance δ_{tol}, maximum rank count ρ_{max}

Output: estimation $\widehat{M}$

1: Trim M^E, and let $\widetilde{M}^E$ be the output
2: Set $\widehat{M}^{(0)} = 0$
3: **For** $\rho = 0, 1, \ldots, \rho_{\max}$ **do**:
4: Compute the rank-1 projection of $\widetilde{M}^E - \widehat{M}^{(\rho)}$, $\mathcal{P}_{\hat{r}}(\widetilde{M}^E - \widehat{M}^{(\rho)}) = X_0^{(\rho)} S_0^{(\rho)} Y_0^{(\rho)T}$
5: Set $X_0^{(\rho)} = [X^{(\rho-1)}; X_0^{(\rho)};]$ and $Y_0^{(\rho)} = [Y^{(\rho-1)}; Y_0^{(\rho)};]$
6: Minimize $F(X, Y)$ through MANIFOLD OPTIMIZATION with ρ replacing $\hat{r}$, with initial condition $(X_0^{(\rho)}, Y_0^{(\rho)})$ and stopping criterion of $|F(\mathbf{x}_{k+1}) - F(\mathbf{x}_k)| \le \delta_{\text{tol}} F(\mathbf{x}_k)$, and let $\widehat{M}^{(\rho)} = X^{(\rho)} S^{(\rho)} Y^{(\rho)T}$ be the output
7: **If** $||\mathcal{P}_E(M - \widehat{M}^{(\rho)})||_F / ||\mathcal{P}_E(M)||_F < \delta_{\text{tol}}$ **then break**
9: **End for**
10: Return $\widehat{M}^{(\rho)}$.

In the following numerical simulations, we demonstrate that INCREMENTAL OPTSPACE brings significant performance gains when applied to ill-conditioned matrices, cf. Section 4.

4 Numerical results with randomly generated matrices

The OptSpace algorithm described above was implemented in C[2] and tested on a 3.4 GHz Desktop computer with 4 GB RAM. For efficient singular value decomposition of sparse matrices, we used (a modification of) SVDLIBC[3] which is based on SVDPACKC. In this section, we compare the performance of OptSpace with other algorithms by numerical simulations. In Section 4.1, the algorithms are tested on randomly generated matrices with noiseless observations, and in Section 4.2 we compare the algorithms when we have noisy observations under different scenarios.

For exact matrix completion experiments, we use $n \times n$ test matrices M of rank r generated as $M = UV^T$. Here, U and V are $n \times r$ matrices with each entry being sampled independently from a standard Gaussian distribution $\mathcal{N}(0, 1)$, unless specified otherwise. Then, each entry is revealed independently with probability ϵ/n, so that on an average $n\epsilon$ entries are revealed. Numerical results show that there is no notable difference if we choose the revealed set of entries E uniformly at random over all the subsets of the same size $|E| = n\epsilon$. We use $\delta_{\rm tol} = 10^{-5}$ and $k_{max} = 1000$ as the stopping criteria.

For approximate matrix completion, the matrices are generated as above and corrupted by additive noise Z_{ij}. First, in the standard scenario, Z_{ij}'s are independently and identically distributed according to a Gaussian distribution. For comparison, we also present numerical simulation results with different types of noise in the following subsections. Again, each entry is revealed independently with a probability ϵ/n. We use $||\mathcal{P}_E(\widehat{M} - (M + Z))||_F^2 \leq (1 + \epsilon)|E|\sigma_n^2$ [7] (where σ_n^2 is the noise variance per entry) as the stopping criterion.

4.1 Exact matrix completion

We first illustrate the rate of convergence of OptSpace . In Figure 1, we plot the fit error, $||\mathcal{P}_E(\widehat{M} - M)||_F/n$ and the prediction error $||\widehat{M} - M||_F/n$, with respect to the number of iterations of the Manifold Optimization step. These plots are obtained for matrices with $n = 1000$ and $r = 10$ and averaged over 10 instances. The results are shown for two values of ϵ: 100 and 200. We can see that the prediction error decays exponentially with the number of iterations in both cases. Also, the prediction error is very close to the fit error, thus lending support to the validity of the chosen stopping criterion.

We next study the *reconstruction rate* of the algorithm. We declare a matrix to be reconstructed if $||M - \widehat{M}||_F/||M||_F \leq 10^{-4}$. The reconstruction rate is the fraction of instances for which the matrix was reconstructed.

In Figure 2, we plot the reconstruction rate as function of $|E|/n$ for OptSpace on randomly generated rank-4 matrices for different matrix sizes n. As predicted by Theorem 1.2 of [19], threshold of the reconstruction rate of OptSpace is upper bounded by $|E| = Cn(\log n)^2$, for fixed rank $r = 4$. Here, an extra factor of $\log n$ comes from the fact that if we generate random factors U and V from a Gaussian distribution, then the incoherence parameter μ_0 scales like $\log n$. However, the location of the threshold is surprisingly close to the lower bound proved in [29] which scales as $|E| = Cn \log n$. The lower bound provides a threshold below which the problem admits more than one solution. Note that the lower bound is displayed only for the case when $n = 1000$.

In Figure 3, we plot the reconstruction rate for randomly generated matrices with dimensions $m = n = 500$ using OptSpace . The resulting reconstruction rate is plotted for different ranks r as a function of $|E|/n$. As rank increases and for fixed n, the reconstruction rate has a sharp threshold

[2]The code is available at http://www.stanford.edu/∼raghuram/optspace.html

[3]Available at http://tedlab.mit.edu/∼dr/svdlibc/

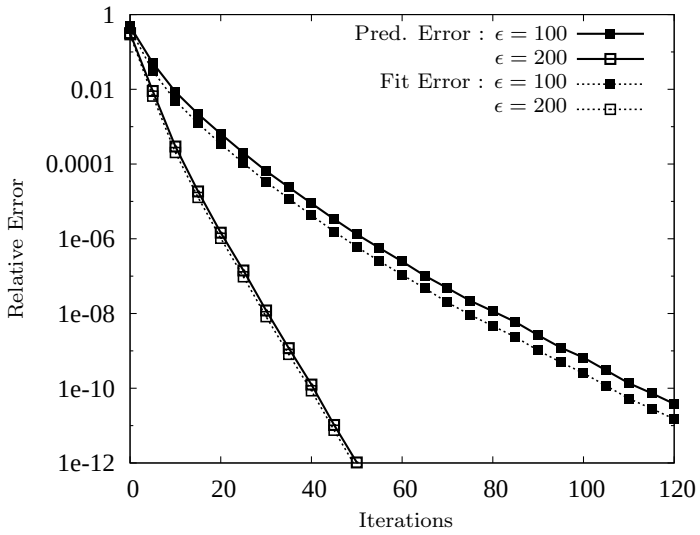

Figure 1: Prediction and fit errors versus the number of iterations of the MANIFOLD OPTIMIZATION step for rank 10 matrices of dimension $n \times n$ with $n = 1000$. Each entry is sampled with probability ϵ/n for two different values of ϵ: 100 and 200.

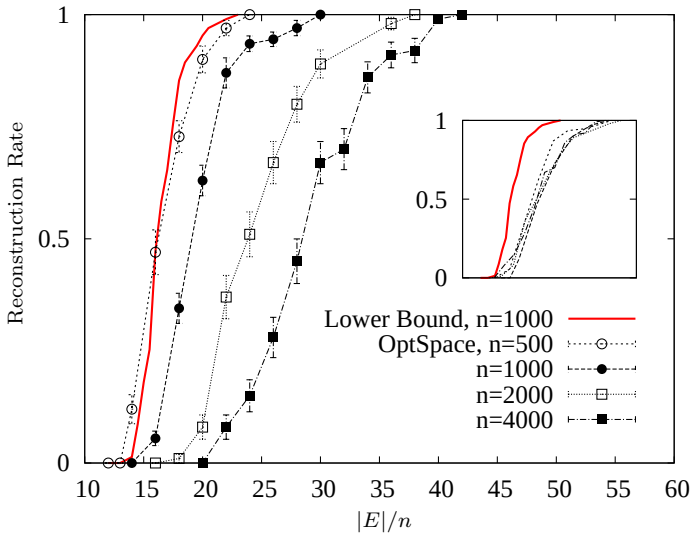

Figure 2: Reconstruction rates for rank 4 matrices using OPTSPACE for different matrix sizes. The solid curve is bound proved in [29]. In the inset, the same data are plotted vs. $|E|/(n(\log n)^2)$

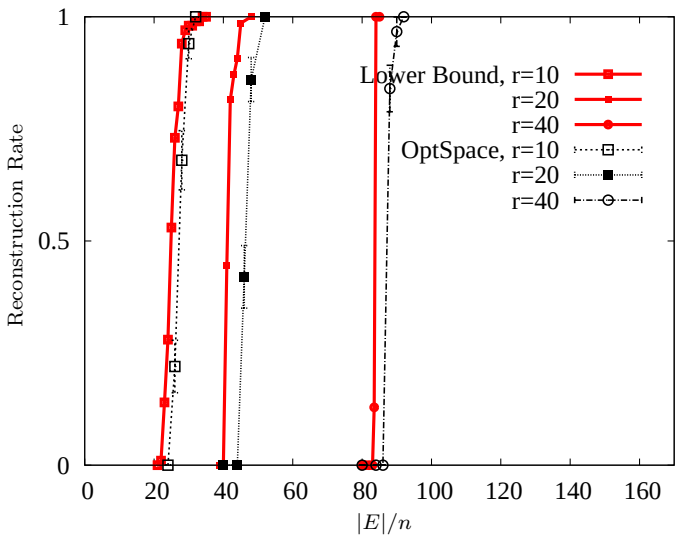

Figure 3: Reconstruction rates for matrices with dimension $m = n = 500$ using OptSpace for different ranks. The solid curves are the bounds proved in [29].

at $|E| = Crn \log n$. This indicates that in practice the dependence of the threshold on the rank scales like r rather than r^2 as predicted by Theorem 1.2 of [19]. Also, for all values of rank, the location of the threshold is surprisingly close to the lower bound proved in [29], below which the problem admits more than one solution.

In Figure 4, we plot the reconstruction rate of OptSpace as a function of $|E|/n$ for rank 10 matrices of dimension $m = n = 1000$. Also plotted are the reconstruction rates obtained for the convex relaxation approach of [10] solved using the Singular Value Thresholding algorithm [7], the FPCA algorithm from [22] and ADMiRA [21]. We compare these with a theoretical lower bound on the reconstruction rate described in [29]. Various algorithms exhibit threshold at different values of $|E|/n$, and the threshold depends on the problem size n and the rank r. This figure clearly illustrates that OptSpace outperforms the other algorithms on random data, and this was consistent for various values of n and r.

In the following Tables 1 and 2, we present numerical results obtained using these algorithms for different values of n and r. Table 1 presents results for smaller values of ϵ and hence for *hard* problems, whereas Table 2 presents results for larger values of ϵ which are relatively *easy* problems. Note that the values of ϵ used in Table 1 all correspond to $|E| \leq 2.6d(n, r)$ where $d(n, r) = 2nr - r^2$ is the number of degrees of freedom. We ran into *Out of Memory* problems for the FPCA algorithm for $n \geq 20000$ and hence we omit these problems from the table. All the results presented in tables are averaged over 5 instances. On the easy problems, all the algorithms achieved similar performances, whereas on the hard problems, OptSpace outperforms other algorithms on most of instances.

To add robustness in the case when the condition number of the matrix M is high, we introduced a novel modification to OptSpace in Section 3.5. To illustrate the robustness of this Incremental OptSpace, in Table 3, we compare the results of exact matrix completion for different values of κ. Here, κ denotes the condition number of the randomly generated matrix M used in the simulation. For this simulation with ill-conditioned matrices, we use $n \times n$ random matrices generated as follows. For fixed $n = 1000$, let $\widetilde{U} \in \mathbb{R}^{n \times r}$ and $\widetilde{V} \in \mathbb{R}^{n \times r}$ be the orthonormal basis for the space spanned

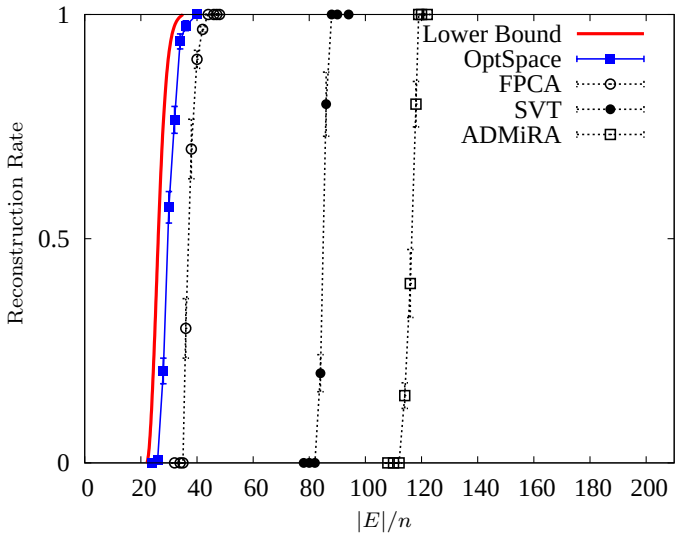

Figure 4: Comparison of reconstruction rates for randomly generated rank 10 matrix of dimension $m = n = 1000$ for the OptSpace and competing algorithms : FPCA, SVT and ADMiRA [7, 22, 21]. The leftmost solid curve is a lower bound proved in [29].

n	r	ϵ	OptSpace		SVT		FPCA		ADMiRA	
			rel. error	time(s)	rel. err.	time(s)	rel. err.	time(s)	rel. err.	time(s)
1000	10	50	1.95×10^{-5}	33	3.42×10^{-1}	734	6.04×10^{-4}	65	4.41×10^{-1}	8
	50	200	1.28×10^{-5}	235	2.54×10^{-1}	11769	1.07×10^{-5}	83	3.54×10^{-1}	141
	100	400	9.22×10^{-6}	837	7.99×10^{-2}	27276	3.86×10^{-6}	165	1.28×10^{-1}	1767
5000	10	50	7.27×10^{-5}	338	5.34×10^{-1}	476	9.99×10^{-1}	1776	5.13×10^{-1}	77
	50	200	1.47×10^{-5}	1930	4.87×10^{-1}	36022	2.17×10^{-2}	2757	5.36×10^{-1}	358
	100	400	1.38×10^{-5}	6794	4.12×10^{-1}	249330	2.49×10^{-5}	3942	4.84×10^{-1}	36266
10000	10	50	1.91×10^{-5}	725	6.33×10^{-1}	647	9.99×10^{-1}	9947	6.19×10^{-1}	129
	50	200	5.02×10^{-6}	3032	5.50×10^{-1}	18558	9.97×10^{-1}	14048	5.79×10^{-1}	11278
	100	400	1.33×10^{-5}	18928	4.84×10^{-1}	169578	8.59×10^{-3}	18448	5.30×10^{-1}	67880
20000	10	50	1.95×10^{-2}	2589	7.30×10^{-1}	1475	–	–	7.20×10^{-1}	286
	50	200	1.49×10^{-5}	10364	6.30×10^{-1}	14588	–	–	6.04×10^{-1}	29323
30000	10	50	1.62×10^{-2}	5767	7.74×10^{-1}	2437	–	–	7.43×10^{-1}	308

Table 1: Numerical results for OptSpace , SVT, FPCA and ADMiRA for *hard* problems. $\epsilon = |E|/n$ is the number of observed entries per row/column.

n	r	ϵ	OptSpace		SVT		FPCA		ADMiRA	
			rel. error	time(s)	rel. err.	time(s)	rel. err.	time(s)	rel. err.	time(s)
1000	10	120	1.18×10^{-5}	28	1.68×10^{-5}	40	5.20×10^{-5}	18	9.09×10^{-4}	52
	50	390	9.26×10^{-6}	212	1.62×10^{-5}	247	3.53×10^{-6}	106	3.62×10^{-5}	701
	100	570	1.49×10^{-5}	723	1.71×10^{-5}	694	1.92×10^{-6}	160	1.88×10^{-5}	2319
5000	10	120	1.51×10^{-5}	252	1.76×10^{-5}	112	1.69×10^{-4}	1083	4.68×10^{-2}	198
	50	500	1.16×10^{-5}	850	1.62×10^{-5}	1312	5.99×10^{-5}	1005	7.42×10^{-3}	92751
	100	800	8.39×10^{-6}	3714	1.73×10^{-5}	5432	3.32×10^{-5}	1953	4.42×10^{-2}	634028
10000	10	120	7.64×10^{-6}	632	1.75×10^{-5}	221	9.95×10^{-1}	13288	1.22×10^{-1}	442
	50	500	1.19×10^{-5}	2585	1.63×10^{-5}	2872	9.51×10^{-5}	7337	2.58×10^{-2}	186591
	100	800	1.46×10^{-5}	8514	1.76×10^{-5}	10962	6.90×10^{-5}	9426	9.66×10^{-2}	755082
20000	10	120	1.59×10^{-5}	1121	1.76×10^{-5}	461	–	–	3.04×10^{-1}	181
	50	500	9.77×10^{-6}	4473	1.64×10^{-5}	6014	–	–	4.33×10^{-2}	346651
30000	10	120	1.56×10^{-5}	1925	1.80×10^{-5}	838	–	–	4.19×10^{-1}	71

Table 2: Numerical results for OptSpace , SVT, FPCA and ADMiRA for *easy* problems. $\epsilon = |E|/n$ is the number of observed entries per row/column.

κ	r	OptSpace		Inc. OptSpace		SVT		FPCA		ADMiRA	
		rel. error	time	rel. error	time(s)	rel. err.	time(s)	rel. err.	time(s)	rel. err.	time(s)
1	10	8.56×10^{-6}	20	8.66×10^{-6}	19	1.70×10^{-5}	55	5.32×10^{-5}	22	1.57×10^{-5}	242
	50	1.16×10^{-5}	78	1.09×10^{-5}	832	1.64×10^{-5}	628	2.97×10^{-6}	115	1.60×10^{-5}	1252
	100	7.05×10^{-6}	401	7.37×10^{-6}	4605	1.78×10^{-5}	2574	1.88×10^{-6}	174	1.67×10^{-5}	3454
5	10	1.08×10^{-1}	124	1.53×10^{-5}	70	1.53×10^{-5}	72	5.53×10^{-5}	21	1.56×10^{-5}	234
	50	1.10×10^{-1}	1591	1.30×10^{-5}	921	1.46×10^{-5}	639	1.08×10^{-5}	145	1.61×10^{-5}	1221
	100	1.24×10^{-1}	5004	1.41×10^{-5}	5863	1.54×10^{-5}	1541	4.38×10^{-6}	664	1.68×10^{-5}	3450
10	10	1.09×10^{-1}	112	2.00×10^{-1}	238	1.47×10^{-5}	127	5.22×10^{-5}	21	1.55×10^{-5}	243
	50	1.04×10^{-1}	1410	1.32×10^{-5}	1593	1.36×10^{-5}	1018	1.42×10^{-5}	270	1.61×10^{-5}	1206
	100	1.10×10^{-1}	4569	1.36×10^{-5}	9550	1.41×10^{-5}	2473	4.54×10^{-6}	996	1.65×10^{-5}	3426

Table 3: Numerical results for OptSpace, Incremental OptSpace, SVT, FPCA and ADMiRA for different condition numbers κ. $\epsilon = |E|/n$ depends only on r and is the same as used in Table 4.

by the columns of U and V respectively. Also, let D be an $r \times r$ diagonal matrix with its diagonal entries linearly spaces between n and n/κ. Then the matrix M is formed as $M = \widetilde{U} D \widetilde{V}^T$. We use $\delta_{\text{tol}} = 10^{-5}$ as the stopping criterion. Table 3 shows that Incremental OptSpace improves significantly over OptSpace and achieves results comparable to the other algorithms.

4.2 Approximate matrix completion

In this section we compare the performance of different algorithms for matrix completion with noisy observations. As a metric, we use the relative root mean squared error defined as

$$\text{RMSE} = \frac{1}{\sqrt{mn}} ||M - \widehat{M}||_F \ . \tag{17}$$

4.2.1 Standard scenario

For direct comparison we start with an example taken from [8]. In this example, M is a square matrix of dimensions $n \times n$ and rank r generated as $M = UV^T$ with fixed $n = 600$. U and V are $n \times r$ matrices with each entry being sampled independently from a standard Gaussian distribution $\mathcal{N}(0, \sigma_s^2 = 20/\sqrt{n})$. As before, each entry is revealed independently with probability ϵ/n. Each entry

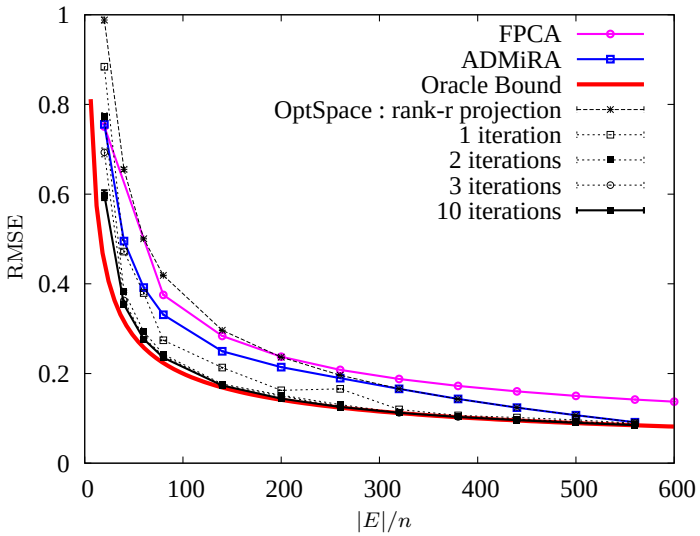

Figure 5: Root mean squared error achieved by OptSpace as a function of the observed entries $|E|$ and of the number of iterations in the Manifold Optimization step. M is a rank-2 matrix with dimensions $m = n = 600$. The performances of the convex relaxation approach and ADMiRA, and an oracle lower bound are shown for comparison.

is corrupted by added noise matrix Z, so that the observation for the index (i, j) is $M_{ij}+Z_{ij}$. Further, Z has each entries drawn from i.i.d. standard Gaussian distribution $\mathcal{N}(0, 1)$. In the following we refer to this noise model as the standard scenario. We also refer to [8] for the data for the convex relaxation approach and the information theoretic lower bound.

Figure 5 compares the average root mean squared error achieved by the different algorithms for a fixed rank $r = 2$ as a function of $|E|/n$. After one iteration, for most values of ϵ, OptSpace has a smaller root mean square error than the convex relaxation approach and in about 10 iterations, it becomes indistinguishable from the information theoretic lower bound. In Figure 6, we compare the average root mean squared error obtained for a fixed sample size $\epsilon = 120$ as a function of the rank. Again, for most values of r, after one iteration OptSpace has a smaller root mean square error than the convex relaxation based algorithm.

Table 4 illustrate how the performance changes with different noise power for fixed $n = 1000$. We present the results of our experiments with different ranks and *noise ratios* defined as

$$N \equiv ||\mathcal{P}_E(Z)||_F / ||\mathcal{P}_E(M)||_F \ , \tag{18}$$

as in [7].

Next, in the following series of examples, we illustrate how the performances change under different noise models. In the following, M is a square matrix generated as UV^T like above, but U and V now have each entry sampled independently from a standard Gaussian distribution $\mathcal{N}(0, 1)$, unless specified otherwise. As before, each entry is revealed independently with probability ϵ/n and the observation is corrupted by added noise matrix Z. We compare the resulting RMSE of FPCA, ADMiRA and OptSpace on this randomly generated matrices with noisy observations and missing entries. Since ADMiRA requires a target rank, we use the rank estimated using Rank Estimation described in Section 3.2. For FPCA we choose $\mu = \sqrt{2np}\sigma$, where $p = |E|/n^2$ and σ^2 is the variance

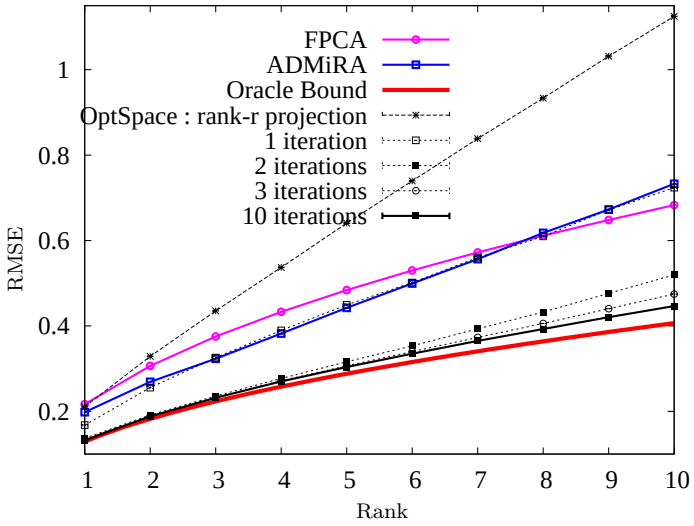

Figure 6: The root mean square error of OptSpace as a function of the rank r, and of the number of iterations in the Manifold Optimization step. M has dimensions $m = n = 600$ and the number of observations is $|E| = 72000$. The performances of the convex relaxation approach and ADMiRA, and an oracle lower bound are shown for comparison.

N	r	ϵ	OptSpace		SVT		FPCA		ADMiRA	
			rel. error	time(s)	rel. err.	time(s)	rel. err.	time(s)	rel. err.	time(s)
	10	120	4.47×10^{-3}	24	7.8×10^{-3}	11	5.48×10^{-3}	99	2.01×10^{-2}	35
10^{-2}	50	390	5.49×10^{-3}	149	9.5×10^{-3}	88	7.18×10^{-3}	805	1.83×10^{-2}	391
	100	570	6.39×10^{-3}	489	1.13×10^{-2}	216	1.08×10^{-2}	1111	1.63×10^{-2}	1424
	10	120	4.50×10^{-2}	23	0.72×10^{-1}	4	6.04×10^{-2}	140	1.18×10^{-1}	12
10^{-1}	50	390	5.52×10^{-2}	147	0.89×10^{-1}	33	7.77×10^{-1}	827	1.20×10^{-1}	139
	100	570	6.38×10^{-2}	484	1.01×10^{-1}	85	1.13×10^{-1}	1140	1.15×10^{-1}	572
	10	120	4.86×10^{-1}	31	0.52	1	5.96×10^{-1}	141	5.38×10^{-1}	3
1	50	390	6.33×10^{-1}	153	0.63	8	9.54×10^{-1}	1088	5.92×10^{-1}	47
	100	570	1.68	107	0.69	35	1.19	1582	6.69×10^{-1}	181

Table 4: Numerical results for OptSpace , SVT, FPCA and ADMiRA when the observations are corrupted by additive Gaussian noise with noise ratio N. $\epsilon = |E|/n$ is the number of observed entries per row/column.

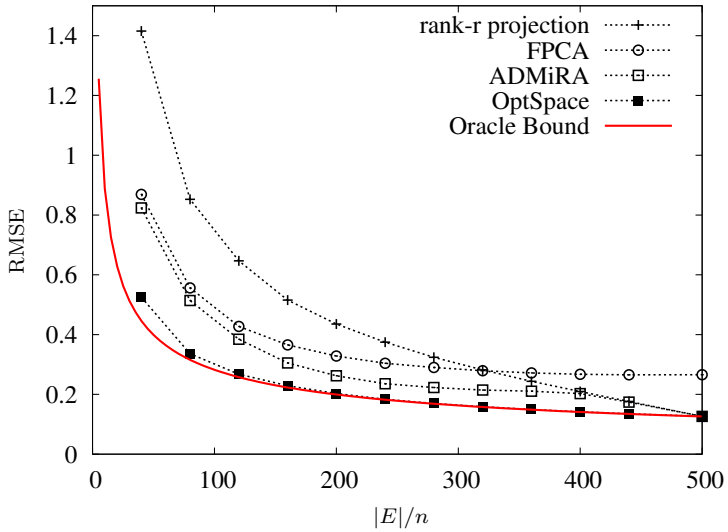

Figure 7: The root mean squared error as a function of the average number of observed entries per row $|E|/n$ for fixed noise ratio $N = 1/2$ under the standard scenario.

of each entry in Z. A convincing argument for this choice of μ is given in [8].

In the standard scenario, we typically make the following three assumptions on the noise matrix Z. (1) The noise Z_{ij} does not depend on the value of the matrix M_{ij}. (2) The entries of Z, $\{Z_{ij}\}$, are independent. (3) The distribution of each entries of Z is Gaussian. The matrix completion algorithms described in Section 2 are expected to be especially effective under this standard scenario for the following two reasons. First, the squared error objective function that the algorithms minimize is well suited for the Gaussian noise. Second, the independence of Z_{ij}'s ensure that the noise matrix is almost full rank and the singular values are evenly distributed. This implies that for a given noise power $||Z||_F$, the spectral norm $||Z||_2$ is much smaller than $||Z||_F$. In the following, we fix $m = n = 500$ and $r = 4$, and study how the performance changes with different noise. Each of the simulation results is averaged over 10 instances and is shown with respect to two basic parameters, the average number of revealed entries per row $|E|/n$ and the noise ratios N, defined as Eq. (18).

In this standard scenario, the noise Z_{ij}'s are distributed as i.i.d. Gaussian N(0,σ^2). Note that the noise ratio is equal to $N = \sigma/2$. The accuracy of the estimation is measured using RMSE. We compare the resulting RMSE of FPCA, ADMiRA and OptSpace to the RMSE of the oracle estimate, which is $\sigma\sqrt{(2nr - r^2)/|E|}$ [8].

Figure 7 shows the performance for each of the algorithms with respect to $|E|/n$ under the standard scenario for fixed $N = 1/2$. For most values of $|E|$, the simple rank-r projection has the worst performance. However, when all the entries are revealed and the noise is i.i.d. Gaussian, the simple rank-r projection coincides with the oracle bound, which in this simulation corresponds to the value $|E|/n = 500$. Note that the behavior of the performance curves of FPCA, ADMiRA, and OptSpace with respect to $|E|$ is similar to the oracle bound, which is proportional to $1/\sqrt{|E|}$.

Among the three algorithms, FPCA has the largest RMSE, and OptSpace is very close to the oracle bound for all values of $|E|$. Note that when all the values are revealed, ADMiRA is an efficient way of implementing rank-r projection, and the performances are expected to be similar. This is

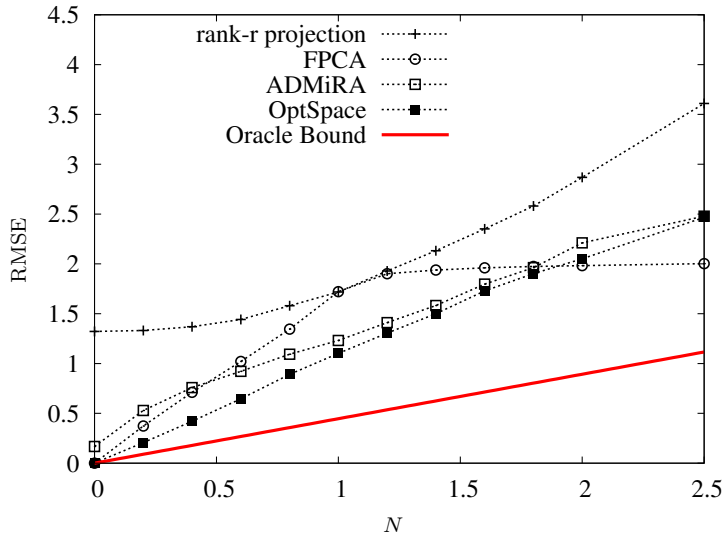

Figure 8: The root mean squared error as a function of noise ratio N for fixed $|E|/n = 40$ under the standard scenario.

confirmed by the observation that for $|E|/n \geq 400$ the two curves are almost identical. One of the reasons why the RMSE of FPCA does not decrease with $|E|$ for large values of $|E|$ is that FPCA overestimates the rank and returns estimated matrices with rank much higher than r, whereas the rank estimation algorithm used for ADMiRA and OptSpace always returned the correct rank r for $|E|/n \geq 80$.

Figure 8 show the performance for each of the algorithms against the noise ratio N within the standard scenario for fixed $|E|/n = 40$. The behavior of the performance curves of ADMiRA and OptSpace are similar to the oracle bound which is linear in σ which, in the standard scenario, is equal to $2N$. The performance of the rank-r projection algorithm is determined by two factors. One is the added noise which is linear in N and the other is the error caused by the erased entries which is constant independent of N. These two factors add up, whence the performance curve of the rank-r projection follows. The reason the RMSE of FPCA does not decrease with SNR for values of SNR less than 1 is not that the estimates are good but rather the estimated entries gets very small and the resulting RMSE is close to $\sqrt{\mathbb{E}[||M||_F^2/n^2]}$, which is 2 in this simulation, regardless of the noise power. When there is no noise, which corresponds to the value $N = 0$, FPCA and OptSpace both recover the original matrix correctly for this chosen value of $|E|/n = 40$.

4.2.2 Multiplicative Gaussian noise

In sensor network localization [31], where the entries of the matrix corresponds to the pair-wise distances between the sensors, the observation noise is oftentimes assumed to be multiplicative. In formulae, $Z_{ij} = \xi_{ij} M_{ij}$, where ξ_{ij}'s are distributed as i.i.d. Gaussian with zero mean. The variance of ξ_{ij}'s are chosen to be $1/r$ so that the resulting noise ratio is $N = 1/2$. Note that in this case, Z_{ij}'s are mutually dependent through M_{ij}'s and the values of the noise also depend on the value of the matrix entry M_{ij}.

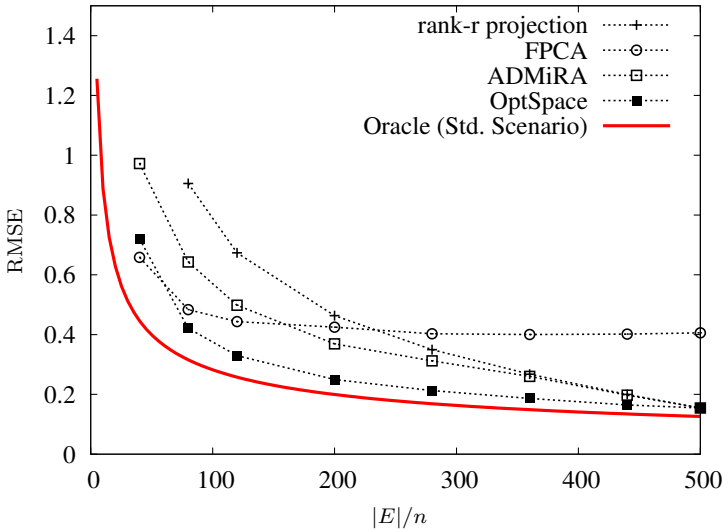

Figure 9: The root mean squared error as a function of the average number of observed entries per row $|E|/n$ for fixed noise ratio $N = 1/2$ within the multiplicative noise model.

Figure 9 shows the RMSE with respect to $|E|/n$ under multiplicative Gaussian noise. The RMSE of the rank-r projection for $|E|/n = 40$ is larger than 1.5 and is omitted in the figure. The bottommost line corresponds to the oracle performance under standard scenario, and is displayed here, and all of the following figures, to serve as a reference for comparison. The main difference with respect to Figure 7 is that most of the performance curves are larger under multiplicative noise. For the same value of the noise ratio N, it is more difficult to distinguish the noise from the original matrix, since the noise is now correlated with the matrix M.

4.2.3 Outliers

In structure from motion [12], the entries of the matrix corresponds to the position of points of interest in 2-dimensional images captured by cameras in different angles and locations. However, due to failures in the feature extraction algorithm, some of the observed positions are corrupted by large noise where as most of the observations are noise free. To account for such outliers, we use the following model.

$$Z_{ij} = \begin{cases} a & \text{with probability } 1/200 \ , \\ -a & \text{w.p. } 1/200 \ , \\ 0 & \text{w.p. } 99/100 \, . \end{cases}$$

The value of a is chosen according to the target noise ratio $N = a/20$. The noise is independent of the matrix entries and Z_{ij}'s are mutually independent, but the distribution is now non-Gaussian.

Figure 10 shows the performance of the algorithms with respect to $|E|/n$ and the noise ratio N with outliers. Comparing the first figure to Figure 7, we can see that the performance for large value of $|E|$ is less affected by outliers compared to the small values of $|E|$. The second figure clearly shows how the performance degrades for non-Gaussian noise when the number of samples is small. The algorithms minimize the squared error $||\mathcal{P}_E(X) - \mathcal{P}_E(N)||_F^2$ as in (7) and (8). For outliers, a suitable algorithm would be to minimize the l_1-norm of the errors instead of the l_2-norm [11, 32]. Hence, for

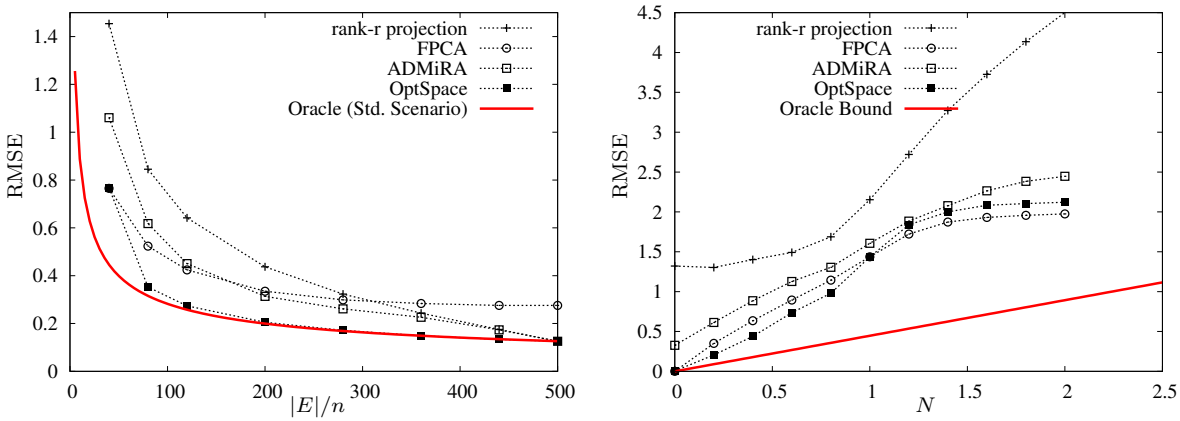

Figure 10: The root mean squared error as a function of the average number of observed entries per row $|E|/n$ for fixed noise ratio $N = 1/2$ with outliers (left) and the RMSE as a function of the noise ratio N for fixed $|E|/n = 40$ with outliers (right).

this simulation with outliers, we can see that the performance of the rank-r projection, ADMiRA and OptSpace is worse than the Gaussian noise case. However, the performance of FPCA is almost the same as in the standard scenario.

4.2.4 Quantization noise

One common model for noise is the quantization noise. For a regular quantization, we choose a parameter a and quantize the matrix entries to the nearest value in $\{\ldots, -a/2, a/2, 3a/2, 5a/2, \ldots\}$. The parameter a is chosen carefully such that the resulting noise ratio is $1/2$. The performance for this quantization is expected to be worse than the multiplicative noise case. The reason is that now the noise is deterministic and completely depends on the matrix entries M_{ij}, whereas in the multiplicative noise model it was random.

Figure 11 shows the performance against $|E|/n$ within quantization noise. The overall behavior of the performance curves is similar to Figure 7, but most of the curves are shifted up. Note that the bottommost line is the oracle performance in the standard scenario which is the same in all the figures. Compared to Figure 9, for the same value of $N = 1/2$, quantization is much more detrimental than the multiplicative noise as expected.

4.2.5 Ill conditioned matrices

In this simulation, we look at how the performance degrades under the standard scenario if the matrix M is ill-conditioned. M is generated as $M = \sqrt{4/166}\, U\, \mathrm{diag}([1, 4, 7, 10])\, V^T$, where U and V are generated as in the standard scenario. The resulting matrix has a condition number close to 10 and the normalization constant $\sqrt{4/166}$ is chosen such that $\mathbb{E}[||M||_F]$ is the same as in the standard case. Figure 12 shows the performance as a function of $|E|/n$ with ill-conditioned matrix M. The performance of OptSpace is similar to that of ADMiRA for many values of $|E|$. However, similar to the noiseless simulation results, Incremental OptSpace achieves a better performance in this case of ill-conditioned matrix.

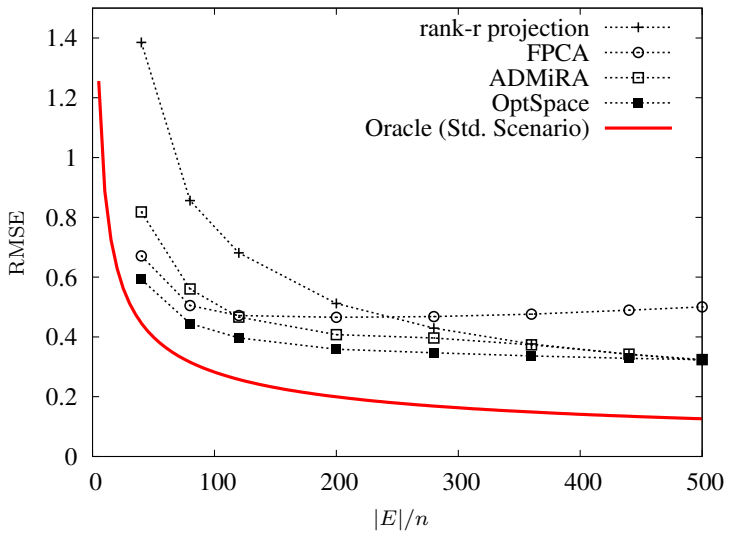

Figure 11: The root mean squared error as a function of the average number of observed entries per row $|E|/n$ for fixed noise ratio $N = 1/2$ with quantization.

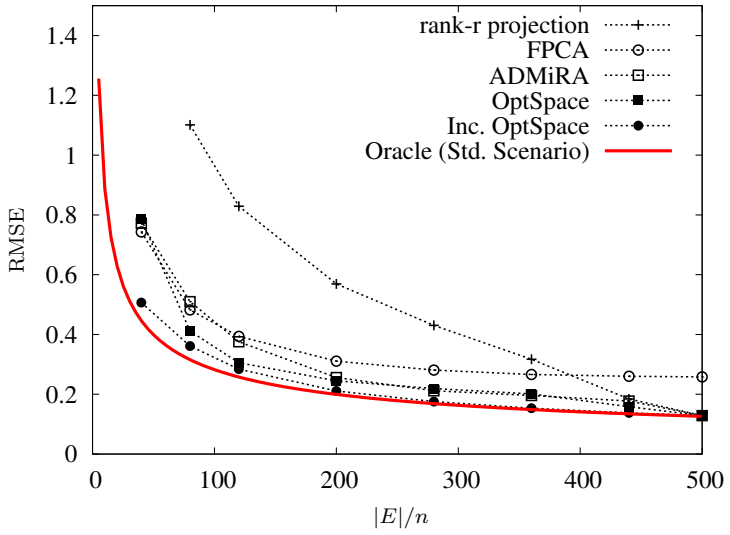

Figure 12: The root mean squared error as a function of the average number of observed entries per row $|E|/n$ for fixed $N = 1/2$ with ill-conditioned matrices.

n_u	m	samples	rank	Incremental OptSpace		FPCA		ADMiRA	
				NMAE	time(s)	NMAE	time(s)	NMAE	time(s)
100	100	7484	2	0.17674	0.1	0.20386	25	0.18194	0.3
1000	100	73626	9	0.15835	11	0.16114	111	0.16194	0.5
2000	100	146700	9	0.15747	26	0.16101	243	0.16286	0.9
4000	100	290473	9	0.15918	56	0.16291	512	0.16317	2
943	1682	80000	10	0.18638	213	0.19018	753	0.24276	5

Table 5: Numerical results for the Jester joke data set, where the number of jokes m is fixed at 100 (top four rows), and for the Movielens data set with 943 users and 1682 movies (last row).

5 Numerical results with real data matrices

In this section, we consider low-rank matrix completion problems in the context of recommender systems, based on two real data sets: the Jester joke data set [1] and the Movielens data set [2]. The Jester joke data set contains 4.1×10^6 ratings for 100 jokes from 73,421 users. [4] Since the number of users is large compared to the number of jokes, we randomly select $n_u \in \{100, 1000, 2000, 4000\}$ users for comparison purposes. As in [22], we randomly choose two ratings for each user as a test set, and this test set, which we denote by T, is used in computing the prediction error in Normalized Mean Absolute Error (NMAE). The Mean Absolute Error (MAE) is defined as in [22, 15].

$$MAE = \frac{1}{|T|} \sum_{(i,j)\in T} |M_{ij} - \widehat{M}_{ij}| \,,$$

where M_{ij} is the original rating in the data set and $\widehat{M}_{ij}$ is the predicted rating for user i and item j. The Normalized Mean Absolute Error (NMAE) is defined as

$$NMAE = \frac{MAE}{M_{\max} - M_{\min}} \,,$$

where $M_{\max}$ and $M_{\min}$ are upper and lower bounds for the ratings. In the case of Jester joke, all the ratings are in $[-10, 10]$ which implies that $M_{\max} = 10$ and $M_{\min} = -10$.

The numerical results for Jester joke data set using Incremental OptSpace, FPCA and ADMiRA are presented in the first four columns of Table 5. In the table, *rank* indicates the rank used to estimate the matrix and *time* is the running time of each matrix completion algorithm. To get an idea of how good the predictions are, consider the case where each missing entries is predicted with a random number drawn uniformly at random in $[-10, 10]$ and the actual rating is also a random number with same distribution. After a simple computation, we can see that the resulting NMAE of the random prediction is 0.333. As another comparison, for the same data set with $n_u = 18000$, simple nearest neighbor algorithm and Eigentaste both yield NMAE of 0.187 [15]. The NMAE of Incremental OptSpace is lower than these simple algorithms even for $n_u = 100$ and tends to decrease with n_u.

Numerical simulation results on the Movielens data set is also shown in the last row of the above table. The data set contains $100,000$ ratings for $1,682$ movies from 942 users.[5] We use $80,000$ randomly chosen ratings to estimate the $20,000$ ratings in the test set, which is called $u1.base$ and

[4] The dataset is available at http://www.ieor.berkeley.edu/∼goldberg/jester-data/

[5] The dataset is available at http://www.grouplens.org/node/73

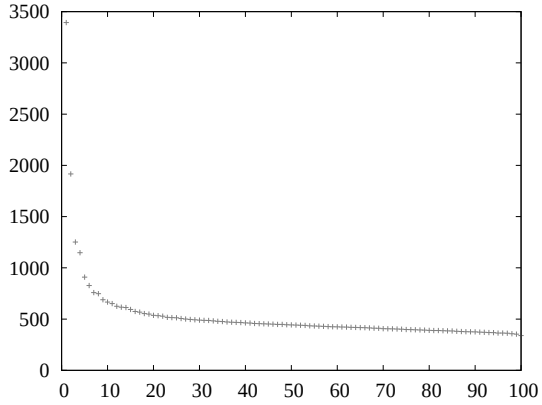

Figure 13: Distribution of the singular values of the complete sub matrix in the Jester joke data set.

$u1.test$, respectively, in the movielens data set. In the last column of Table 5, we compare the resulting NMAE using INCREMENTAL OPTSPACE , FPCA and ADMIRA.

Next, to get some insight on the structure of real data, we look at a complete sub matrix where all the entries are known. With Jester joke data set, we deleted all users containing missing entries, and generated a complete matrix M with $14,116$ users and 100 jokes. The distribution of the singular values of M is shown in Figure 13. We must point out that this rating matrix is not low-rank or even approximately low-rank, although it is common to make such assumptions. This is one of the difficulties in dealing with real data. The other aspect is that the samples are not drawn uniformly at random as commonly assumed in [10, 19].

Finally we test the incoherence assumption for the Netflix dataset [3] in Figure 14 and Figure 15. For a $m \times n$ matrix whose singular value decomposition is given by $M = U\Sigma V^T$, M is said to be (μ_0, μ_1)-incoherent if it satisfies the following properties [9]:

A1. There exists a constant $\mu_0 > 0$ such that for all $i \in [m]$, $j \in [n]$ we have $\sum_{k=1}^{r} U_{i,k}^2 \leq \mu_0 r/m$, $\sum_{k=1}^{r} V_{i,k}^2 \leq \mu_0 r/n$.

A2. There exists μ_1 such that $|\sum_{k=1}^{r} U_{i,k}V_{j,k}| \leq \mu_1 \sqrt{r/mn}$.

To check if **A1** holds for the Netflix movie ratings matrix, we run OptSpace on the Netflix dataset and plot cumulative sum of the sorted row norms of the left and right factors defined as follows. Let the output of OptSpace be $X \in \mathbb{R}^{m\times r}$, $Y \in \mathbb{R}^{n\times r}$ and $S \in \mathbb{R}^{r\times r}$. Here $m = 480,189$ is the number of users and $n = 17,770$ is the number of movies. For the target rank we used $r = 5$. Let $x_i = \frac{m}{r}||X^{(i)}||^2$ and $y_i = \frac{n}{r}||Y^{(i)}||^2$ where $X^{(i)}$ and $Y^{(i)}$ denote the ith row of the left factor X and the right factor Y respectively. Define a permutation $\pi_l : [m] \to [m]$ which sorts x_i's in a non-decreasing order such that $x_{\pi_l(1)} \leq x_{\pi_l(2)} \leq \ldots \leq x_{\pi_l(m)}$. Here, we used the standard combinatorics notation $[k] = \{1, 2, \ldots, k\}$ for an integer k. Similarly, we can define $\pi_r : [n] \to [n]$ for y_i's.

In Figure 14, we plot $\sum_{i=1}^{k} x_i$ vs. k. For comparison, we also plot the corresponding results for a randomly generated matrix X_G. Generate $U \in \mathbb{R}^{m\times r}$ by sampling its entries U_{ij} independently and distributed as $\mathcal{N}(0, 1)$ and let X_G be the left singular vectors of U. Since x_i's are scaled by m/r, when $k = m$ we have $\sum_{i=1}^{m} x_i = m$. This is also true for the random matrix X_G. Figure 15 shows the corresponding plots for Y. For a given matrix, if **A1** holds with a small μ_0 then the corresponding

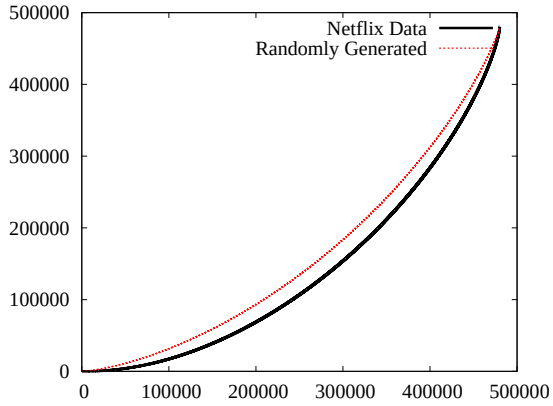

Figure 14: Cumulative sum of the sorted row norms of the factor corresponding to users.

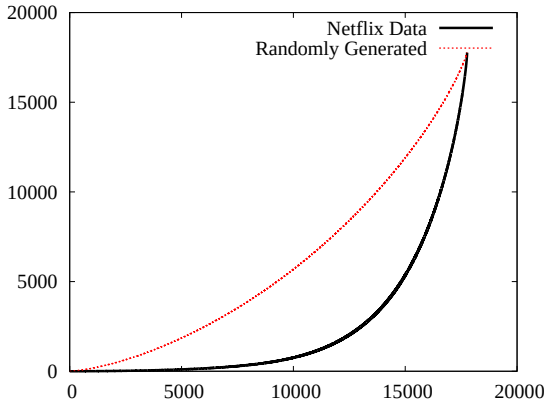

Figure 15: Cumulative sum of the sorted row norms of the factor corresponding to movies.

curve would be close to a straight line. The curvature in the plots is indicative of the disparity among the row weigths of the factors. We can see that a randomly generated matrix would satisfy **A1** with a smaller value of μ_0 compared to the movie ratings matrix, hence can be said to be more incoherent. The factor corresponding to movies has a larger disparity than the factor corresponding to users, and hence challenges the incoherence assumption.

A Proof of Proposition 3.1

The matrix M to be reconstructed is factorized as Eq. (1), where $\Sigma = \mathrm{diag}(\Sigma_1, \ldots, \Sigma_r)$ is a diagonal matrix of the singular values. We start from following key lemma.

Lemma A.1. *There exists a numerical constant $C > 0$ such that, with high probability*

$$\left| \frac{\sigma_q}{\epsilon} - \Sigma_q \right| \leq C M_{\max} \left(\frac{\alpha}{\epsilon} \right)^{1/2} , \tag{19}$$

where it is understood that $\Sigma_q = 0$ *for* $q > r$*, and* $M_{\max} = \max\{M_{ij}\}$.

The proof of this lemma is provided in [19]. Applying this result to the cost function $R(i) = \frac{\sigma_{i+1}+\sigma_1\sqrt{i/\epsilon}}{\sigma_i}$, we get the following bounds :

$$\begin{aligned} R(r) &\leq \frac{CM_{\max}\sqrt{\alpha\epsilon} + (\Sigma_1 + CM_{\max}\sqrt{\alpha/\epsilon})\sqrt{r\epsilon}}{\epsilon\Sigma_r - CM_{\max}\sqrt{\alpha\epsilon}} \ , \\ R(i) &\geq \frac{\epsilon\Sigma_{i+1} - CM_{\max}\sqrt{\alpha\epsilon}}{\epsilon\Sigma_i + CM_{\max}\sqrt{\alpha\epsilon}} \quad , \ \forall i < r \\ R(i) &\geq \frac{(\Sigma_1 - CM_{\max}\sqrt{\alpha/\epsilon})\sqrt{r\epsilon}}{CM_{\max}\sqrt{\alpha\epsilon}} \quad , \ \forall i > r \end{aligned}$$

Let, $\beta = \Sigma_1/\Sigma_r$ and $\xi = (M_{\max}\sqrt{\alpha})/(\Sigma_1\sqrt{r})$. After some calculus, we establish that for

$$\epsilon > C\, r\, \max\left\{ \ \xi^2 \ , \ \xi^4\beta^2 \ , \ \beta^4 \ \right\} \ ,$$

we have the desired inequality $R(r) < R(i)$ for all $i \neq r$. This proves the remark.

Acknowledgment

We thank Andrea Montanari for stimulating discussions and helpful comments on the subject of this paper. This work was partially supported by a Terman fellowship, the NSF CAREER award CCF-0743978 and the NSF grant DMS-0806211.

References

[1] Jester jokes. http://eigentaste.berkeley.edu/user/index.php.

[2] Movielens. http://www.movielens.org.

[3] Netflix prize. http://www.netflixprize.com/.

[4] P.-A. Absil, R. Mahony, and R. Sepulchrer. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, 2008.

[5] L. Armijo. Minimization of functions having lipschitz continuous first partial derivatives. *Pacific J. Math.*, 16(1):1–3, 1966.

[6] T. Blumensath and M. E. Davies. Iterative hard thresholding for compressed sensing. *Applied and Computational Harmonic Analysis*, 27(3):265–274, 2009. `arXiv:0805.0510`.

[7] J-F Cai, E. J. Candès, and Z. Shen. A singular value thresholding algorithm for matrix completion. `arXiv:0810.3286`, 2008.

[8] E. J. Candès and Y. Plan. Matrix completion with noise. `arXiv:0903.3131`, 2009.

[9] E. J. Candès and B. Recht. Exact matrix completion via convex optimization. `arxiv:0805.4471`, 2008.

[10] E. J. Candès and T. Tao. The power of convex relaxation: Near-optimal matrix completion. `arXiv:0903.1476`, 2009.

[11] V. Chandrasekaran, S. Sanghavi, P. A. Parrilo, and A. S. Willsky. Rank-sparsity incoherence for matrix decomposition. arXiv:0906.2220, 2009.

[12] P. Chen and D. Suter. Recovering the missing components in a large noisy low-rank matrix: application to sfm. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(8):1051–1063, Aug. 2004.

[13] W. Dai and O. Milenkovic. Set: an algorithm for consistent matrix completion. arXiv:0909.2705, 2009.

[14] A. Edelman, T. A. Arias, and S. T. Smith. The geometry of algorithms with orthogonality constraints. *SIAM J. Matr. Anal. Appl.*, 20:303–353, 1999.

[15] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins. Eigentaste: A constant time collaborative filtering algorithm, July 2001.

[16] D. Gross. Recovering low-rank matrices from few coefficients in any basis. arXiv:0910.1879v2, 2009.

[17] D. Gross, Y. Liu, S. T. Flammia, S. Becker, and J. Eisert. Quantum state tomography via compressed sensing. arXiv:0909.3304v2, 2009.

[18] R. H. Keshavan, A. Montanari, and S. Oh. Learning low rank matrices from $O(n)$ entries. In *Proc. of the Allerton Conf. on Commun., Control and Computing*, September 2008.

[19] R. H. Keshavan, A. Montanari, and S. Oh. Matrix completion from a few entries. arXiv:0901.3150, January 2009.

[20] R. H. Keshavan, A. Montanari, and S. Oh. Matrix completion from noisy entries. arXiv:0906.2027, June 2009.

[21] K. Lee and Y. Bresler. Admira: Atomic decomposition for minimum rank approximation. arXiv:0905.0044, 2009.

[22] S. Ma, D. Goldfarb, and L. Chen. Fixed point and Bregman iterative methods for matrix rank minimization. arXiv:0905.1643, 2009.

[23] R. Mazumder, T. Hastie, and R. Tibshirani. Spectral regularization algorithms for learning large incomplete matrices. http://www-stat.stanford.edu/∼hastie/Papers/SVD_JMLR.pdf , 2009.

[24] R. Meka, P. Jain, and I. S. Dhillon. Guaranteed rank minimization via singular value projection. arXiv:0909.5457, 2009.

[25] D. Needell and J. A. Tropp. Cosamp: Iterative signal recovery from incomplete and inaccurate samples. *Applied and Computational Harmonic Analysis*, 26(3):301–321, Apr 2008.

[26] S. Oh, , A. Karbasi, and A. Montanari. Sensor network localization from local connectivity : performance analysis for the MDS-MAP algorithm. http://infoscience.epfl.ch/record/140635, 2009.

[27] B. Recht. A simpler approach to matrix completion. arXiv:0910.0651v2, 2009.

[28] A. Singer. A remark on global positioning from local distances. *Proceedings of the National Academy of Sciences*, 105(28):9507–9511, 2008.

[29] A. Singer and M. Cucuringu. Uniqueness of low-rank matrix completion by rigidity theory. `arXiv:0902.3846`, January 2009.

[30] K. Toh and S. Yun. An accelerated proximal gradient algorithm for nuclear norm regularized least squares problems. http://www.math.nus.edu.sg/∼matys, 2009.

[31] Z. Wang, S. Zheng, S. Boyd, and Y. Ye. Further relaxations of the sdp approach to sensor network localization. Technical report, 2006.

[32] J. Wright, A. Ganesh, S. Rao, and Y. Ma. Robust principal component analysis: Exact recovery of corrupted low-rank matrices. `arXiv:0905.0233`, 2009.