A Comparative Study of Matrix Completion Methods

Sriram Ganesan[*]
sriramg@umich.edu

Nitin Garg
gargn@umich.edu

Jeeheh Oh
jeeheh@umich.edu

Jonathan Stroud
stroud@umich.edu

Abstract

Matrix completion, is the task of filling in the unknown entries of a sparse matrix. In the present paper, the baseline algorithms for matrix completion, such as eigentaste, nearest K-neighbor, and column mean are compared against the more sophisticated matrix completion algorithms, such as Spectral Matrix Completion (SMC) and Singular Value Thresholding (SVT). The comparison metric used in the present paper includes accuracy, convergence time, and the nature of the input matrix. All of the presented methods have their advantages and disadvantages based on the problem. In the present paper, the Normalized Mean Absolute Error (NMAE) are presented for synthetic data and the real-world data (Jester jokes dataset). While Eigentaste needs at least few fully filled columns to complete the sparse matrix, SMC and SVT does not have that limitation. Between SMC and SVT, SMC provides better accuracy but is much slower than the SVT.

1 Introduction

How can we predict what movies a person will enjoy? On what web-pages will they click? Which emails will they read? *Collaborative Filtering* problems like these are tricky. Consider predicting a person's movie preferences using Netflix's database. We can predict how highly a user will rate the movie *Rambo* using a regression model trained on the ratings of users that have rated *Rambo* before. We quickly run into problems when we realize that most users have not rated *Rambo*, and most users who have, have not rated the same ones as the user for whom we are making the prediction. This means that our model would have to ignore a majority of the data available, limiting performance. Furthermore a separate model would need to be trained for each movie in the system, leading to computational difficulties.

A better framework for these sorts of problem is *matrix completion*. We can view each user-item pair as an entry in a matrix that has only a few entries filled in. Our task is then to complete the matrix, assuming that it has some simple underlying structure. Low rank is a common structural assumption. Recovering a low rank matrix from a few entries is a problem with applications not only in collaborative filtering [15], but also in dimensionality reduction [11, 17] and multi-task learning [1, 12]. This problem turns out to be computationally hard. In fact, rank minimization is NP-hard, meaning that is at least as hard as all those that can be solved by a nondeterministic Turing machine in polynomial time, like the Hamiltonian Cycle and Traveling Salesman problems. In both theory and practice, all algorithms that perform this task have time complexity exponential in the size of the matrix.

Candes and Recht showed that most low-rank matrices could be recovered instead by minimizing the nuclear norm, defined as the sum of the singular values [3]. Theoretically, the nuclear norm is the tightest convex lower bound on the rank function for singular values no greater than 1. Intuitively,

[*]The names are printed in alphabetical order by last name.

while the rank function counts the number of nonzero singular values, the nuclear norm sums their amplitude, much like how the 1-norm is a useful substitute for counting the number of nonzero entries in a vector. Conveniently, the nuclear norm can be minimized efficiently subject to equality constraints via semidefinite programming.

Nuclear norm minimization has long been observed to produce fairly low-rank solutions [6, 7, 16], but only recently there was any theoretical basis for the instances where it produced the true minimum rank solution. One of the first papers to provide such foundation was [14], where Recht, Fazel, and Parrilo developed probabilistic techniques to study average case behavior. They showed that the nuclear norm heuristic could solve most instances of the rank minimization problem when the number of linear constraints were sufficiently large. This inspired interest in theoretical guarantees for rank minimization, and these results lay the foundation for [3]. Candes and Rechts bounds were subsequently improved by Candes and Tao [4] and Keshavan, Montanari, and Oh [10] to show that one could, in special cases, reconstruct a low-rank matrix by observing a set of entries of size at most a polylogarithmic factor larger than the intrinsic dimension of matrix.

To test the validity of these theoretical guarantees in practice, we explore two competing algorithms for matrix completion via nuclear norm-minimization and compare their performance on collaborative filtering baselines, using both synthetic and real-world data. The Singular Value Thresholding (SVT) algorithm was introduced in "A Singular Value Thresholding Algorithm for Matrix Completion" [2]. Spectral Matrix Completion (SMC) was introduced in "Matrix Completion from a Few Entries" [10].

In Section 2, we explain the methodology and parameters used for the SVT algorithm. Section 3 explains the SMC algorithm with detailed description of the steps involved. Section 4 explains the baseline algorithms used in the present paper, namely, Eigentaste, and K-nearest neighbor. In Section 5.1 and Section 5.2, we present a detailed comparison of the results for synthetic data and real-world data (Jester joke dataset) respectively. Section 5.3 contains the two-dimensional visualization of the Jester joke dataset. Section 6 presents the conclusions, with Section 7 giving a brief overview of the team memeber's efforts.

2 Singular Value Thresholding

Formally, Singular Value Thresholding (SVT) addresses the optimization problem

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

where $\|\cdot\|_*$ is the *nuclear norm*, or the sum of the singular values and $\mathcal{P}_\Omega(\cdot)$ is the projection operator, which makes zero all entries $(i, j) \notin \Omega$.

Singular Value Thresholding works in an iterative, alternating fashion reminiscent of the Alternating Direction Method of Multipliers (ADMM). The complete matrix X is constructed iteratively using a low-rank, low-singular value approximation to an auxiliary sparse matrix Y. Y is then adjusted to ensure the resulting approximation in the subsequent step has matching entries $X_{ij} = M_{ij}$. Each iteration consists of the inductive steps

$$\begin{cases} X^k = \text{shrink}(Y^{k-1}, \tau) \\ Y^k = Y^{k-1} + \delta_k \mathcal{P}_\Omega(M - X^k), \end{cases}$$

where $\text{shrink}(\cdot, \cdot)$ is the *singular value shrinkage operator*, which reduces each singular value by a given singular value decomposition $X = U\Sigma V^T$, $\Sigma = \text{diag}(\{\sigma_i\}_{1 \le i \le r})$, we can write this as

$$\text{shrink}(X, \tau) = U\Sigma_\tau V^T, \;\; \Sigma_\tau = \text{diag}(\{(\sigma_i - \tau)_+\}).$$

These two operations when repeated, approach a low-nuclear norm solution by repeatedly shrinking the singular values of X. This algorithm has shown success in recovering accurate low-rank solutions when the source of M is also low-rank, even though it does not optimize this objective directly. The original authors discuss its theoretical guarantees in detail, but we choose to omit them in this discussion.

SVT has hyperparameters that must be carefully tuned to guarantee convergence. The shrinkage value τ must be set high in order for the algorithm to converge, but not too high that it exceeds the true singular values. The stepsize δ_k are similarly sensitive. The stepsize can be set dynamically as well, though we choose to maintain a fixed stepsize throughout. Furthermore, we compute the decomposition of Y^K in batches, which introduces a new batch size parameter l that effects the speed of convergence. Also important is the initialization of Y, for which the authors provide helpful strategies. Finally, we use the relative error $\|\mathcal{P}_\Omega(X^k - M)\|_F / \|P_\Omega(M)\|$ as a stopping criterion. We terminate when this drops below a small ϵ.

We find from experimentation that SVT is frustratingly sensitive to the settings of these parameters. The suggested settings only lead to convergence for a limited range of matrix sizes and ranks, and choosing the correct settings often requires unrealistic knowledge of the underlying matrix structure.

3 Spectral Matrix Completion

Spectral Matrix Completion (SMC), presented in Keshavan et al. [10] minimizes the reconstruction error of a low-rank matrix. Let M be the complete matrix and M^E be the $m \times n$ matrix with known entries $(i, j) \in E$ filled in and zero entries otherwise. That is,

$$M^E_{i,j} = \begin{cases} M_{i,j} & \text{if } (i,j) \in E \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

As presented in [10], the SMC algorithm has three steps.

1. Trim M^E, giving $\widetilde{M}^E$.
2. Project $\widetilde{M}^E$ to $T_r(\widetilde{M}^E)$.
3. Clean, minimizing the reconstruction error $F(X, Y)$.

3.1 Trimming

In the trimming step, we throw out information in order to make the underlying true-rank structure more apparent. First, we zero all columns in M^E with degree larger than $2|E|/n$ and set to zero all rows with degree larger than $2|E|/m$, where $|E|$ is the number of non-zero entries in M. This step is crucially important when the number of revealed entries per row/column follows a heavy tail distribution, as is the case for most of the real data.

3.2 Projection

In the projection step, we build a low-rank reconstruction of M^E. We begin by computing its the singular value decomposition (SVD). Let the singular values be $\sigma_1 \geq \sigma_2 \geq ...\sigma_{\min(m,n)} \geq 0$. Therefore, the decomposition can be written as,

$$M^E = \sum_{i=1}^{\min(m,n)} \sigma_i x_i y_i^T. \tag{2}$$

The projection

$$T_r(M^E) = \frac{mn}{|E|} \sum_{i=1}^{r} \sigma_i x_i y_i^T \tag{3}$$

is obtained by zeroing all but the r largest singular values. Note that apart from the rescaling factor $(mn/|E|)$, $T_r(M^E)$ is the orthogonal projection of M^E onto the set of rank-r matrices. The rank of the matrix, r, should either be supplied by the user or it will be estimated using the change in consecutive singular values.

3.3 Cleaning

This is the step where actual optimization takes place in SMC algorithm. We "clean" the errors produced by the projection step by iteratively minimizing the reconstruction error. Given $X \in$

$R^{m\times r}$, $Y \in R^{n\times r}$ with $X^TX = mI$ and $Y^TY = nI$,

$$F(X,Y) = \min_{S\in R^{r\times r}} F(X,Y,S) \tag{4}$$

$$F(X,Y,S) = \frac{1}{2}\sum_{(i,j)\in E}(M_{ij} - (XSY^T)_{ij})^2 \tag{5}$$

We initiliaze $T_r(\widetilde{M^E}) = X_0S_0Y_0^T$ and minimize $F(X,Y)$ locally with initial condition $X = X_0$, $Y = Y_0$. Note that $F(X,Y)$ is easy to evaluate since it is defined by minimizing the quadratic function $S \mapsto F(X,Y,S)$ over the low-dimensional matrix S. Further, it depends on X and Y only through their column spaces. In geometric terms, F is a function defined over the cartesian product of two Grassmann manifolds. Optimization over Grassmann manifolds is a well understood topic [5] and efficient algorithms (in particular Newton and conjugate gradient) can be applied. In our implementation, we minimize $F(X,Y)$, using gradient descent with line search.

4 Baseline Algorithms

For comparison, we introduce two simple baseline algorithms for matrix completion.

4.1 Eigentaste

Eigentaste [9] is an another collaborative filtering algorithm specifically designed for the Jester Dataset. It relies on the assumption that a subset of the columns will be dense, corresponding to a few "seed" items which every user is required to review. Eigentaste applies principal component analysis to this dense subset of columns, reducing dimensionality and allowing rapid clustering. Recommendation is done by averaging ratings over clusters.

Mean rating of the jth item in the gauge set is given by

$$\mu_{ij} = \frac{1}{n}\sum_{i\epsilon U_j}\widetilde{r}_{ij}$$

$$\sigma_j^2 = \frac{1}{n-1}\sum_{i\epsilon U_j}(\widetilde{r}_{ij} - \mu_j)^2$$

In A, the normalized rating r_{ij} is set to $(\widetilde{r}_{ij} - \mu_j)/\sigma_j$. The global correlation matrix is given by

$$C = \frac{1}{n-1}A^TA = E^T\Lambda E$$

The data is projected along the first v eigenvectors $x = RE_v{}^T$

Recursive Rectangular Clustering:

1. Find the minimal rectangular cell that encloses all the points in the eigenplane.
2. Bisect along x and y axis to form 4 rectangular sub-cells.
3. Bisect the cells in step 2 with origin as a vertex to form sub-cells at next hierarchical level.

Online Computation of Recommendations

1. Collect ratings of all items in gauge set.
2. Use PCA to project this vector to eigenplane.
3. Find the representative cluster.
4. Look up appropriate recommendations, present them to the new user, and collect ratings.

4.2 K-Nearest Neighbors

In the K-Nearest Neighbors algorithm, we predicts ratings based on the mean ratings of each person's nearest neighbors.

$$p_{ij} = \overline{r_i} + \kappa \sum_{k=1}^{n} w(i,p)(\overline{r_{pj}} - \overline{r_p})$$

where $\overline{r_i}$ is the average joke rating for user i, and κ is a normalizing factor ensuring that the absolute value of the weights sum to 1. We used a function of Euclidean distance from user i to user p as the weight w(i, p), and $\kappa = \sum_{k=1}^{n} w(i,p)$. Specifically, if we are interested in q nearest neighbors, $w(i;p) = d(i, q+1) - d(i,p)$. This ensures that is closest neighbor has the largest weight. In the present paper, we are using $K = 50$ as the baseline algorithm to compare against more sophisticated algorithms.

5 Experimental Results

5.1 Synthetic Data

The performance of the SVT and SMC algorithms were compared on synthetically generated matrices of varying ranks, sizes and sparsity. Specifically, the algorithms were tested on all combinations of matrix size (1000, 5000), rank (5, 10 and 20), and sparsity (30%, 50% and 70%). As a baseline to SVT and SMC, a column mean matrix completion method was implemented. This method predicts the average column value for all missing entries. In the Tables 1, 2 the mean absolute error and the run time is averaged over the categories of size, rank, and sparsity. For example, out of the eighteen tests that were run, nine tests include the matrix of size 1000×1000. Therefore the mean absolute error value for all nine tests are averaged in order to represent the values in the size 1000 column of the table below. Mean absolute error is the absolute value of the average error on the unknown entries. Run time is measured in seconds and all simulations were run at University of Michigan's CAEN lab computers (8GB memory with one core i7 processors). The columns in Tables 1, 2 contain the averaged mean absolute error and the run time for all experiments.

While low rank is a desirable property and a suitable target for optimization, we are primarily concerned with accurately predicting unknown entries in a matrix.

Table 1: Mean Absolute Error of predictions of unknown entries. The tolerance in SMC and SVT algorithms were set to 10^{-6} as the stopping criterion.

	Rank			Sparsity			Size ($n \times n$)	
Algorithm	5	10	20	30%	50%	70%	1000	5000
Mean	5.65	9.90	13.98	9.10	10.21	10.21	9.43	10.26
SVT	2.05	7.2E-04	1.4E-03	2.05	8.4E-04	9.6E-04	1.37	7.6E-04
SMC	2.1E-04	4.3E-07	4.5E-07	2.1E-04	4.2E-07	3.8E-07	1.4E-04	3.7E-07

Table 1 shows that both SVT and SMC outperform the baseline column-mean method, which simply fills in all unknown entries in a column with the mean of its known entries. In addition, SMC consistently outperforms SVT in terms of mean absolute error, often by several orders of magnitude.

We are also concerned with the speed at which we arrive at accurate predictions. Table 2 shows that, while SMC significantly outperforms SVT in terms of accuracy, SVT significantly outperforms SMC in terms of time. It appears that the computational cost of SMC grows exponentially with the rank. Even at rank 5, the smallest tested rank value, SMC runs 9.6 times slower than SVT.

Since the baseline algorithms (Eigentaste) needs at least few fully filled columns, we did not use baseline algorithms to compare the synthetic data. However, we use the baseline algorithms to compare in the next section with Jester joke datatset.

Table 2: Run Time (in seconds) for different algorithms.

	Rank			Sparsity			Size ($n \times n$)	
Algorithm	5	10	20	30%	50%	70%	1000	5000
Mean	0.12	0.12	0.10	0.11	0.11	0.12	0.01	0.22
SVT	26	39	92	42	54	60	6	98
SMC	252	405	1623	1021	510	748	86	1433

5.2 Jester Dataset

The Jester Joke dataset contains 4.1 million ratings for 100 jokes from 73,421 users [8]. A set of 10 "seed" jokes were chosen to be presented to users before any others, and users that did not rate all of the seed jokes were discarded. This leaves us with 10 completely dense columns, allowing us to apply both supervised learning algorithms (Eigentaste, K-nearest neighbor) and matrix completion algorithms (SVT, SMC). We hypothesize that the matrix completion algorithms will outperform those that only take advantage of the dense columns because matrix completion algorithms are capable of utilizing all data during training.

For the purpose of evaluation we randomly select subsets of 100, 200, and 1000 users' ratings from the Jester Dataset. We choose two ratings at random from each user as test points and leave the remainder for training. We evaluate four algorithms, Eigentaste, k-nearest neighbor ($K = 50$), SMC, and SVT, using the Normalized Mean Absolute Error (NMAE) of the reconstruction on the test points, similar to [13].

The NMAE, a commonly used performance metric in collaborative filtering, is defined in terms of The Mean Absolute Error (MAE):

$$MAE = \frac{1}{|T|} \sum_{(u,i) \in T} |M_{ui} - \widetilde{M}_{ui}| \tag{6}$$

where M_{ui} is the original ratings, $\widetilde{M}_{ui}$ is the predicted rating for user u, item i, and T is the test set. The NMAE can be defined as:

$$NMAE = \frac{MAE}{M_{max} - M_{min}} \tag{7}$$

where M_{max} and M_{min} are the upper and lower bounds for the ratings. In the Jester joke dataset, the rating are in the range $[-10, 10]$.

Table 3 compares the NMAE for different number of users with all the four algorithms, namely, SMC, SVT, Eigentaste, and K-nearest neighbor. As noted from table 3, SMC performs the best for all the different size of users. Another point to note that, generally, NMAE keeps increasing with the matrix size. Thus, in Table 4, we present detailed numerical results on the Jester joke dataset with the SMC algorithm.

Table 3: NMAE comparison on the Jester joke dataset for SMC, SVT, Eigentaste, and K-nearest neighbour algorithms.

		NMAE			
# user	# jokes	SMC	SVT	Eigentaste	K-nearest neighbor ($K = 50$)
100	100	0.1573	0.1865	0.1873	0.2088
200	100	0.1603	0.1844	0.1909	0.2123
1000	100	0.1647	0.1708	0.2371	0.2134

To conclude, for the Jester joke dataset SMC algorithm performs best (in terms of NMAE) amongst the algorithms we compared against.

5.2.1 Visualization

Each of the algorithms compared previously rely on the assumption that the Jester Dataset can approximated using a low-rank matrix. To test this claim, we plot each of the 100 jester jokes in a

Table 4: Numerical results on the Jester joke dataset with SMC algorithm. Times reported are from a University of Michigan's CAEN lab computer (8GB memory with one core i7 processor)

# users	# jokes	samp. ratio	NMAE	Time (s)
100	100	5353	0.1573	25.31
200	100	10921	0.1603	17.44
1000	100	57578	0.1647	44.95

two-dimensional plane and attempt to explain the meaning of the directions. In axes according to the top two eigenvectors of the Singular Value Thresholding solution.

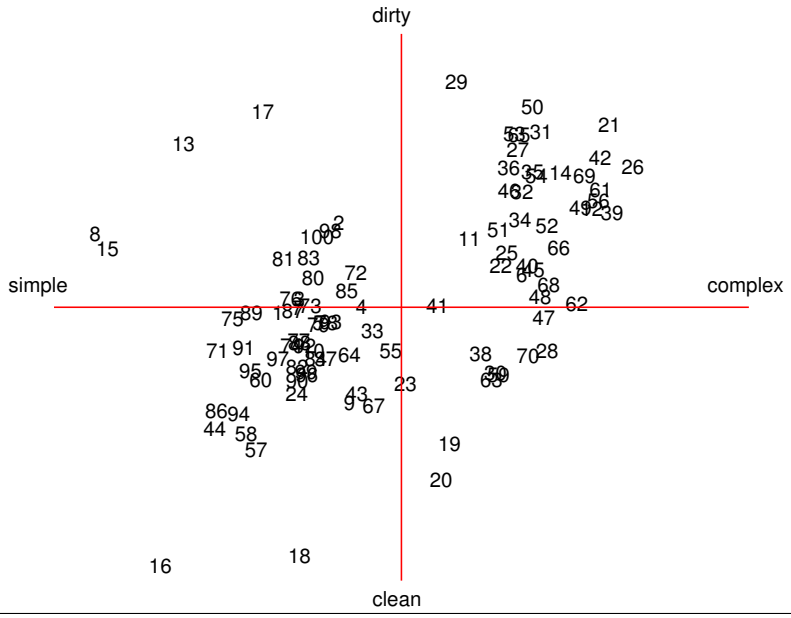
In figure 1 we find that the two principal components in the space of jokes correspond to short, simple puns (left) versus longer, complex jokes (right), and rowdy jokes (top) versus clean jokes (bottom). Only a few jokes are given here, but at closer inspection these observations seem to hold true in general.

6 Conclusions

In the present paper, we explained various algorithms to solve matrix completion problem. In particular, we focused on Spectral Matrix Completion (SMC) algorithm and Spectral Value Thresholding (SVT) algorithm. We compared these sophisticated algorithm with various baseline algorithms, namely, column mean, eigentaste, and K-nearest neighbor algorithm. We checked the performance of the algorithms with synthetic data, with varying size, rank, and sparsity of the input matrix. We concluded that both SMC and SVT performs better than the baseline column-mean algorithm. Between SMC and SVT, while the SMC attains better accuracy than SVT, SMC takes longer time to converge than SVT. For the Jester joke dataset, we compared NMAE (Normalized Mean Absolute Error) for different users (100, 200, 1000) on SMC, SVT, Eigentaste, and K-nearest neighbor (with $K = 50$). We concluded that the SMC performs the best (in terms for NMAE) for all the user sets selected in this paper. In terms of NMAE, SMC is followed by SVT, then Eigentaste, and lastly, K-nearest neighbor. This trend follows the intuition as well. Lastly, we present a two dimensional representation of the entire 100 jokes in the Jester joke datatset. We went through the jokes individually, and tried to guess the axes to the best of our abilities. It seems it follows the general trend, as explained in the paper.

7 Group Member's Accomplishments

All team members participated in all aspects of the project as well as the final report. However, each member was responsible for taking the lead on specific tasks. All team members contributed towards the literature review as well. Sriram implemented Eigentaste and K-nearest neighbor. He also had a leading role in the comparison of the methods on the Jester dataset. Nitin and Jeeheh lead the implementation of the SMC algorithm. Jeeheh took charge of the performance comparison on synthetic data. Nitin got the performance metric for different algorithms and also carried out integration of the final report. Jonathan was in command of implementing the SVT algorithm and the two dimensional visualizations of the Jester dataset. He also contributed in comparing various algorithms for the synthetic dataset.



Joke #	Joke Text
8	Q. Did you hear about the dyslexic devil worshiper? A. He sold his soul to Santa.
26	A guy walks into a bar and sits down next to an extremely gorgeous woman. The first thing he notices about her though, are her pants. They were skin-tight, high-waisted and had no obvious mechanism (zipper, buttons or velcro) for opening them. After several minutes of puzzling over how she got the pants up over her hips, he finally worked up the nerve to ask her. "Excuse me miss, but how do you get into your pants?" "Well," she replied, "you can start by buying me a drink."
29	An old Scotsmen is sitting with a younger Scottish gentleman and says the boy. "Ah, lad look out that window. You see that stone wall there, I built it with me own bare hands, placed every stone meself. But do they call me MacGregor the wall builder? No! He Takes a few sips of his beer then says, "Aye, and look out on that lake and eye that beautiful pier. I built it meself, laid every board and hammered each nail but do they call me MacGregor the pier builder? No! He continues..."And lad, you see that road? That too I build with me own bare hands. Laid every inch of pavement meself, but do they call MacGregor the road builder? No!" Again he returns to his beer for a few sips, then says, "Agh, but you screw one sheep..."
18	A dog walks into Western Union and asks the clerk to send a telegram. He fills out a form on which he writes down the telegram he wishes to send: "Bow wow wow, Bow wow wow." The clerk says, "You can add another 'Bow wow' for the same price." The dog responded, "Now wouldn't that sound a little silly?"

Figure 1: Two-dimensional projection of the Jester joke dataset. The horizontal and vertical axes correspond to the directions of the first and second eigenvectors, respectively, of the completed matrix found by the SVT algorithm on the full Jester dataset. Also given are the most extreme jokes in each direction, which characterize the axes.

References

[1] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. Convex multi-task feature learning. *Machine Learning*, 73(3):243–272, 2008.

[2] Jian-Feng Cai, Emmanuel J Candès, and Zuowei Shen. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, 20(4):1956–1982, 2010.

[3] Emmanuel J Candès and Benjamin Recht. Exact matrix completion via convex optimization.

Foundations of Computational mathematics, 9(6):717–772, 2009.

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

[5] Alan Edelman, Tomás A Arias, and Steven T Smith. The geometry of algorithms with orthogonality constraints. *SIAM journal on Matrix Analysis and Applications*, 20(2):303–353, 1998.

[6] Maryam Fazel. *Matrix rank minimization with applications*. PhD thesis, PhD thesis, Stanford University, 2002.

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

[8] Ken Goldberg. Anonymous ratings from the jester online joke recommender system, 2003.

[9] Ken Goldberg, Theresa Roeder, Dhruv Gupta, and Chris Perkins. Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval*, 4(2):133–151, 2001.

[10] Raghunandan H Keshavan, Andrea Montanari, and Sewoong Oh. Matrix completion from a few entries. *Information Theory, IEEE Transactions on*, 56(6):2980–2998, 2010.

[11] Nathan Linial, Eran London, and Yuri Rabinovich. The geometry of graphs and some of its algorithmic applications. *Combinatorica*, 15(2):215–245, 1995.

[12] Guillaume Obozinski, Ben Taskar, and Michael I Jordan. Joint covariate selection and joint subspace selection for multiple classification problems. *Statistics and Computing*, 20(2):231–252, 2010.

[13] Sewoong Oh. *Matrix completion: Fundamental limits and efficient algorithms*. PhD thesis, Stanford University, 2010.

[14] Benjamin Recht, Maryam Fazel, and Pablo A Parrilo. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM review*, 52(3):471–501, 2010.

[15] Jasson DM Rennie and Nathan Srebro. Fast maximum margin matrix factorization for collaborative prediction. In *Proceedings of the 22nd international conference on Machine learning*, pages 713–719. ACM, 2005.

[16] Nathan Srebro. *Learning with matrix factorizations*. PhD thesis, Citeseer, 2004.

[17] Kilian Q Weinberger and Lawrence K Saul. Unsupervised learning of image manifolds by semidefinite programming. *International Journal of Computer Vision*, 70(1):77–90, 2006.