Singular Value Thresholding for Matrix Completion

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. This problem is central in recommender systems and collaborative filtering, which provide individualized product suggestions based on ratings gathered from a large set of users. Matrix completion algorithms are used effectively today to provide helpful recommendations for users of services such as Netflix and Amazon.com. 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 include, 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 detailed results are presented for synthetic data and the Jester data. 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. Detailed comparison of the methods is presented in the later sections.

1 Introduction

Recovering a low rank matrix from a given subset of its entries is a well known problem in collaborative filtering [**?**] , dimensionality reduction[**?**, **?**] and multi-class learning [**?**, **?**], finding the lowest rank matrix satisfying equality constraints is NP-hard. The complexity class of decision problems that are intrinsically harder than those that can be solved by a nondeterministic Turing machine in polynomial time. Examples of NP-hard problems include the Hamiltonian cycle and traveling salesman problems. All known algorithms which can compute lowest rank solution for all instances require time atleast exponential in the dimensions of the matrix in both theory and practice.

Candes and Recht showed that most low rank matrices could be recovered from most sufficiently large sets of entries by computing the matrix of minimum nuclear norm that agreed with the provided entries[**?**] and the known set of entries could comprise a vanishing fraction of the entire matrix. The nuclear norm is equal to the sum of the singular values of a matrix and is the best convex lower bound of the rank function on the set of matrices whose singular values are all bounded by 1. The intuition behind this measure is that whereas the rank function counts the number of nonvanishing singular values, the nuclear norm sums their amplitude, much like how the 1- norm is a useful substitute for counting the number of nonzeros in a vector. Moreover, the nuclear norm can be minimized subject to equality constraints via semidefinite programming.

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

Nuclear norm minimization had long been observed to produce very low-rank solutions in practice (see, for example [**?**, **?**, **?**]), but only very recently was there any theoretical basis for when it produced the minimum rank solution. The first paper to provide such foundations was [**?**], where Recht, Fazel, and Parrilo developed probabilistic techniques to study average case behavior and showed that the nuclear norm heuristic could solve most instances of the rank minimization problem assuming the number of linear constraints was sufficiently large. The results in [**?**] inspired a groundswell of interest in theoretical guarantees for rank minimization, and these results lay the foundation for [**?**]. Candes and Rechts bounds were subsequently improved by Candes and Tao [**?**] and Keshavan, Montanari, and Oh [3] 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 the variety of rank r matrices.

Collaborative Filtering is the process of learning patterns by aggregating information over multiple sources of data, typically multiple users of the same system. These techniques have been applied effectively on many difficult problems, most notably on recommender systems such as those found on Netflix, Amazon, Spotify, and others. The goal of these systems is to produce targeted product recommendations for individual users given the products the user has previously viewed or rated. Users have typically viewed or rated only a small subset of the available products, and different users typically have not rated the same set of products. These problems do not easily lie into any supervised learning framework, so we instead employ the framework of *matrix completion*.

In matrix completion, we reconstruct a matrix M from a set of known entries $M_{ij},\ (i,j) \in \Omega$. Relying on the assumption that M has low rank, we can pose this as a constrained rank-minimization problem

$$\begin{array}{ll} \underset{X}{\text{minimize}} & \operatorname{rank}(X) \\ \text{subject to} & X_{ij} = M_{ij},\ \ \forall (i,j) \in \Omega. \end{array}$$

In this project, we explore several algorithms for matrix completion via rank-minimization and compare their performance on collaborative filtering baselines. We provide an implementation of the method described in “A Singular Value Thresholding Algorithm for Matrix Completion” [1] and reproduce their experiments. We also provide an implementation of a competing algorithm from “Matrix Completion from a Few Entries” [3] and compare their effectiveness.

2 Singular Value Thresholding

Singular Value Thresholding (SVT) [1] is an algorithm proposed for *nuclear norm minimization* of a matrix. Formally, SVT addresses the optimization problem

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

where $\|\cdot\|_*$ is the *nuclear norm*, or the sum of the singular values and $\mathcal{P}_\Omega(\cdot)$ makes zero all entries $(i,j) \notin \Omega$. This can be thought of as a convex relaxation to the rank minimization problem, and the two are formally equivalent under some conditions. The rank minimization problem is, however, highly non-convex and therefore not a suitable candidate for black-box optimization algorithms.

Singular Value Thresholding works by iteratively constructing X 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

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

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

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

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.

In practice, this system has a number of hyperparameters that must be carefully tuned to guarantee convergence. The shrinkage value τ must be set fairly high in order for the algorithm to converge quickly, but not too high that it dwarfs the true singular values. The stepsizes δ_k are similarly sensitive. These can be set dynamically as well, though we choose to maintain a fixed stepsize throughout. We compute the decomposition of Y^K in batches, which introduces a new batch size parameter l. Also important is the initialization of Y^0, 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.

3 Spectral Matrix Completion

In the present paper, the Spectral Matrix Completion (SMC) algorithm presented in Keshavan et al. [3] is implemented. Lets say, M is the matrix whose entry $(i, j) \in [m] \times [n]$ corresponding to the rating user i would assign to movie/joke j. M^E is the $m \times n$ matrix that contains the revealed entries of M, and is filled with 0's in the other positions

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

As presented in [3], the SMC algorithm has the following structure:

1. Trim the M^E, and let $\widetilde{M}^E$.
2. Project $\widetilde{M}^E$ to $T_r(\widetilde{M}^E)$.
3. Clean residual errors by minimizing the discrepancy F(X,Y).

3.0.1 Trimming

In the trimming step, 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. Trimming leads to 'throwing out information' which makes the underlying true-rank structure more apparent. This effect becomes even more important when the number of revealed entries per row/column follows a heavy tail distribution, as for real data.

3.0.2 Projection

In the projection step, compute the singular value decomposition (SVD) of M^E (with $\sigma_1 \geq \sigma_2 \geq ..... \geq 0$)

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

and, then the matrix $T_r(M^E)$ is $(mn/|E|) \sum_{i=1}^{r} \sigma_i x_i y_i^T$, obtained by setting to 0 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.

3.0.3 Cleaning

This is the step where all the magic happens in SMC algorithm. Given $X \in R^{m \times r}$, $Y \in R^{n \times r}$ with $X^T X = m1$ and $Y^T Y = n1$,

$$F(X,Y) = \min_{S \in R^{r \times r}} F(X,Y,S) \tag{3}$$

$$F(X,Y,S) = \frac{1}{2} \sum_{(i,j) \in E} (M_{ij} - (XSY^T)_{ij})^2 \tag{4}$$

The cleaning step consists in writing $T_r(\widetilde{M^E}) = X_0 S_0 Y_0^T$ and minimizing $F(X,Y)$ 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 [2] and efficient algorithms (in particular Newton and conjugate gradient) can be applied. In the present algorithm, gradient descent with line search is used to minimize $F(X,Y)$.

4 Baseline Algorithms

4.1 Eigentaste

Eigentaste is a collaborative filtering algorithm which applies principal component analysis to the dense subset of user ratings, thus facilitating dimensionality reduction for offline clusters and rapid computation of recommendations. 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

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

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

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 hierarchial 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 Neighbor

Nearest neighbor algotihm predicts the ratings based on mean user rating and variation of the current rating from the mean rating of its 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 implemented the weighted nearest neighbor algorithm. We used a function of Euclidean distance from user i to user p as the weight w(i, p), and κ = $\sum_{k=1}^{n} w(i,p)$. 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.

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 2, 3 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 1000x1000. 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 2, 3 contain the averaged mean absolute error and the run time for all experiments.

Table 1: Rank Recovery

		Gaussian Noise						
Algorithm	0.0	0.01	0.1	1.0	5.0	10.0	50.0	100.0
Recovered Rank (SVT)	6	98	26	39	92	42	54	60
Recovered Rank (SMC)	86	1433	252	405	1623	1021	510	748

Table 2: Mean Absolute Error

		Rank			Sparsity		Size	(Average)
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 3: Run Time (in seconds)

		Rank			Sparsity		Size	(Average)
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

Both SVT and SMC outperform the baseline column-mean method. In addition, SMC consistently outperforms SVT in terms of mean absolute error, by order of three. 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 rank size. Even at rank five, the smallest tested rank value, SMC runs 9.6 times slower than SVT.

5.2 Real Datasets

The Jester joke dataset contains 4.1 million rating for 100 jokes from 73,421 users [**?**].

For the purpose of comparison between different algorithms, namely SMC, SVT, and Eigentaste (as baseline), 100, 200, 1000 users are selected at random. Two ratings at random from each user is selected as a test set and the Normalized Mean Absolute Eroor (NMAE) is compared in this set (similar to presented in [**?**]. The NMAE is a commonly used performance metric in collaborative filtering. The Mean Absolute Error (MAE) is define as:

$$MAE = \frac{1}{|T|} \sum_{(u,i) \in T} |M_{ui} - \widetilde{M_{ui}}| \tag{5}$$

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. Then, the NMAE can be defined as:

$$NMAE = \frac{MAE}{M_{max} - M_{min}} \tag{6}$$

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

In the Table 4, numerical results on the Jester joke dataset with SMC algorithm is presented. The number of jokes are fixed at 100 and number of users were varied as 100, 200, 1000. The NMAE is calculated as shown earlier and CPU time is presented as the time (in seconds) on University of Michigan's CAEN lab computer(8GB memory with one core i7 processors).

Table 4: Numerical results on the Jester joke dataset with SMC algorithm

num. user	num. jokes	samp. ratio	NMAE	Time (in sec.)
100	100	5353	0.1573	25.31
200	100	10921	0.1603	17.44
1000	100	57578	0.1647	44.95

Table 5: NMAE comparison on the Jester joke dataset for SMC, SVT, and Eigentaste algorithm

			NMAE	
num. user	num. jokes	SMC	SVT	Eigentaste
100	100	0.1573	0.1865	0.187
200	100	0.1603	0.1843	0.190
1000	100	0.1647	0.1714	0.237

We have eigentaste, SMC and SVT on subsets of the Jester dataset. Compare accuracy and time. Mention that we used LMSVD in SMC for this version.

6 Conclusions

General Points: synthetic data conclusions: SMC and SVT are better than baseline mean method. SMC attains better accuracy than SVT but takes longer. One potential improvement to SMC is to use the LMSVD mentioned in the SVT paper. We used this upgrade in the Jester dataset tests and we found that

7 Group Member's Accomplishments

All team members participated in data processing early in the project. All team members contributed to the final report. Each member was responsible for the writeup regarding their individual experiments. Sriram spearheaded the eigentaste approach. He discovered ways of modifying these methods to supercharge performance and he also contributed a great deal towards the performance metric for different algorithms. Nitin and Jeeheh implemented the SMC algorithm. Jeeheh took charge in getting the performance comparison for the synthetic data. Nitin contributed towards the performance metric for different algorithms and also helped integrating the final report together. Jonathan was in command of the SVT algorithm. SVT algorithm lot of parameter tuning too. He also created those nice-looking visulaization plots in the report.

References

[1] 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.

[2] 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.

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