Matrix completion with Singular Value Thresholding

Hon Ming Chun

Introduction

Matrix completion aims to fill in the missing entries of a given incomplete matrix from a small subset of its entries. Generally, matrix completion is referred to low rank matrix completion since high rank matrix completion is NP-hard. The matrix given need to be low rank and sparse.

One famous example of the usage of matrix completion would be the Netflix problem: Given a ratings matrix in which each entry (i,j) represents the rating of movie j by customer i, apply matrix completion the rating matrix in order to make recommendations to customers on what they would like to watch next. The rating matrix would be low-rank and sparse assume that the user preferences can often be described by a few factors, such as the movie genre and time of release and most users only watch a rate a few movies.

This report mainly discusses the matrix completion problem with an algorithm called Singular Value Thresholding algorithm (SVT) introduced by Cai, Candès and Shen [1].

The algorithm

Background

To solve matrix completion, the problem needs to be converted to a mathematical minimization problem:

$$\begin{aligned} \min_{X} \quad & \operatorname{rank}(X) \\ \text{subject to} \quad & X_{ij} = M_{ij} \ \ \forall i, j \in E \end{aligned}$$

Given an incomplete matrix M, find a matrix X such that the rank(X) is minimized subject to $X_{ij} = M_{ij}$, meaning use as few features as possible to construct a matrix that the original filled entries are unchanged. The above minimization, however, is NP-hard and required a convex relaxation to make it not NP-hard:

$$\begin{aligned} \text{minimize} \quad & \|\boldsymbol{X}\|_* \\ \text{subject to} \quad & X_{ij} = M_{ij}, \quad (i,j) \in \Omega, \end{aligned}$$

Since the rank of a matrix can be defined as the number of non-zero eigenvalues of the matrix and the nuclear norm is the sum of all eigenvalues, the rank(X) can be replaced by $\|\boldsymbol{X}\|_*$ and this is a convex minimization is not NP-hard. In Singular Value Thresholding (SVT), an orthogonal projection is then applied:

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

The projection operator P_Ω project onto the span of matrices vanishing outside of Ω so that the (i, j)th component of $P_\Omega(X)$ is equal to X_{ij} if (i, j) ∈ Ω and zero otherwise. Finally, the SVT use a similar minimization for matrix completion:

$$\begin{aligned} \text{minimize} \quad & \tau\|\boldsymbol{X}\|_* + \tfrac{1}{2}\|\boldsymbol{X}\|_F^2 \\ \text{subject to} \quad & \mathcal{P}_\Omega(\boldsymbol{X}) = \mathcal{P}_\Omega(\boldsymbol{M}). \end{aligned}$$

where τ is the parameter threshold and $\|\boldsymbol{X}\|_F^2$ is the square of Frobenius norm of X. It is obvious that when $\tau \to \infty$, this minimization converges to the previous minimization. Therefore, SVT is an algorithm to minimize the nuclear norm of a matrix and solve matrix completion using an iterative algorithm with a large τ.

Basic idea

SVT was inspired by the linearized Bregman iterations and iterative soft-thresholding algorithms which can be used in L1 minimization. It can be regarded as an extension of classic iterative soft-thresholding since it finds a sparse vector of singular values and the bases instead of finds sparse solutions.

SVT iterate two matrices X and Y until the escaping condition is reached where X and Y both are n x m which is the same size as M. Set Y = M initially, each iteration calculates the truncated SVD of Y using soft-thresholding and use only the dominant singular values and singular vectors to calculate X, then calculate the new Y using the new X and old Y, finally break the loop if the error is small or maximum iteration is reached and set Xopt = X where Xopt is the solution

How it is derived

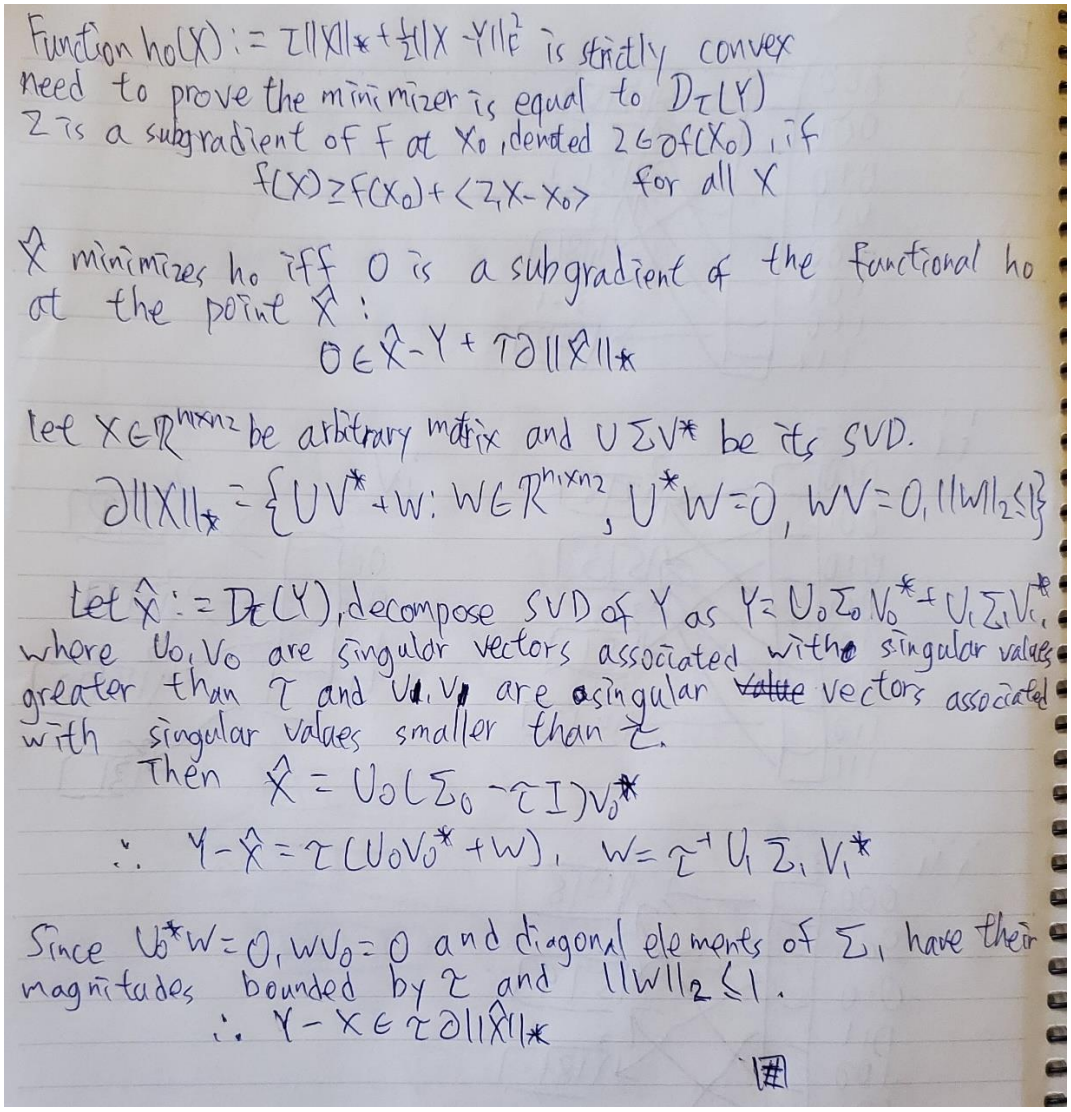
To understand SVT, one need to first understand the soft-thresholding operator Dτ:

$$\mathcal{D}_\tau(\boldsymbol{X}) := \boldsymbol{U}\mathcal{D}_\tau(\boldsymbol{\Sigma})\boldsymbol{V}^*, \quad \mathcal{D}_\tau(\boldsymbol{\Sigma}) = \operatorname{diag}(\{\sigma_i - \tau)_+\})$$

where $t_+ = \max(0, t)$, this operator applies a soft-thresholding to the singular values of X, and shrinking those singular values to zero. As mentioned above, this shrinkage operator can be seen as an extension of the soft-thresholding rule for scalars and vectors, like when the soft thresholding rule applies to vector leads to sparse outputs whenever some entries of the vector are below threshold, if many of the singular values of X are below the threshold τ, the rank of $\mathcal{D}_\tau(\boldsymbol{X})$ is than lower than that of X. Also note that the singular value shrinkage operator obeys:

$$\mathcal{D}_\tau(\boldsymbol{Y}) = \arg\min_{\boldsymbol{X}} \ \frac{1}{2}\|\boldsymbol{X} - \boldsymbol{Y}\|_F^2 + \tau\|\boldsymbol{X}\|_*$$

Proof:



The mentioned basic idea can then be formally defined as Fix τ > 0 and a sequence {δk} of positive step sizes. Starting with Y_0, inductively define for k = 1, 2,... ,

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

until the error is small or maximum iteration is reached.

The SVT can also be recast to a type of Lagrange multiplier algorithm because the SVT is essentially also a type of minimization that subject to a certain constrains. First set $f_\tau(\boldsymbol{X}) = \tau\|\boldsymbol{X}\|_* + \frac{1}{2}\|\boldsymbol{X}\|_F^2$ and then recall the problem is:

$$\begin{array}{ll} \text{minimize} & f_\tau(\boldsymbol{X}) \\ \text{subject to} & \mathcal{P}_\Omega(\boldsymbol{X}) = \mathcal{P}_\Omega(\boldsymbol{M}) \end{array}$$

The Lagrangian for this problem is then $\mathcal{L}(\boldsymbol{X}, \boldsymbol{Y}) = f_\tau(\boldsymbol{X}) + \langle \boldsymbol{Y}, \mathcal{P}_\Omega(\boldsymbol{M} - \boldsymbol{X}) \rangle$. By applying the Uzawa's algorithm which approaches the problem by finding a saddle point using iterative procedure, one can get:

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

where $\{\delta k\}_{k\geq 1}$ is a sequence of positive step sizes. The Uzawa's algorithm use a subgradient method and can observe that $\partial_{\boldsymbol{Y}} g_0(\boldsymbol{Y}) = \partial_{\boldsymbol{Y}} \mathcal{L}(\tilde{\boldsymbol{X}}, \boldsymbol{Y}) = \mathcal{P}_\Omega(\boldsymbol{M} - \tilde{\boldsymbol{X}})$

where $\tilde{\boldsymbol{X}}$ is the minimizer of the Lagrangian for the value of Y and thus a gradient descent update for Y is

$$\boldsymbol{Y}^k = \boldsymbol{Y}^{k-1} + \delta_k \partial_{\boldsymbol{Y}} g_0(\boldsymbol{Y}^{k-1}) = \boldsymbol{Y}^{k-1} + \delta_k \mathcal{P}_\Omega(\boldsymbol{M} - \boldsymbol{X}^k)$$

Still require to compute the minimizer of the Lagrangian and note that

$$\arg\min \ f_\tau(\boldsymbol{X}) + \langle \boldsymbol{Y}, \mathcal{P}_\Omega(\boldsymbol{M} - \boldsymbol{X}) \rangle = \arg\min \ \tau \|\boldsymbol{X}\|_* + \frac{1}{2} \|\boldsymbol{X} - \mathcal{P}_\Omega \boldsymbol{Y}\|_F^2$$

Since the minimizer is $\mathcal{D}_\tau(\mathcal{P}_\Omega(\boldsymbol{Y}))$ and $\boldsymbol{Y}^k = \mathcal{P}_\Omega(\boldsymbol{Y}^k)$ for all k>= 0,

Uzawa's algorithm takes the form:

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

which is exactly the SVT algorithm. This point of view makes proving the convergence of SVT easier since it can be seen as a type of Lagrange multiplier algorithm.

Remark

The solution of SVT is of low rank since most singular value (factors) of the original matrix is eliminated by the parameter threshold τ. Also, because of the project operator P_Ω, Y_k is always sparse. Finally, he SVD and the respective X_k can be computed quickly since each iteration only the truncated SVD is computed but not the full SVD.

Implementation

This section will mention about the implementation details and introduce different parameters in the SVT.

To calculate the truncated SVD using SVD packages, it is required that the number of singular values and the corresponding singular vectors to be calculated is given as a parameter. To do so, let $r_{k-1} = \text{rank}(X^{k-1})$ be the number of nonzero singular values of X^{k-1} at the previous iteration, then let $s_k = r_{k-1} + 1$ and calculate the first s_k singular values for Y^{k-1}. s_k is the right choice for the parameter if some of the computed singular values are smaller than the threshold parameter τ. Otherwise, increment s_k by an integer ℓ until some of the singular values are smaller than τ.

The step size δ define how much progress the algorithm would make in each iteration. The loop escaping tolerance ε define how small the loop escaping criterion is, the smaller ε is, the harder for the loop to escape and the error would be smaller. Finally, K_{max} define the maximum iteration count of the algorithm.

The implementation details are as follows:

1. Set up all the parameters
2. Set X^0 and Y^0 to zero matrix and r_0 to zero
3. For iteration k=1 to k = k_{max}, loop the following
4. Set $s_k = r_{k-1} + 1$
5. Repeat to compute the truncated SVD of the original matrix using only s_k singular value and set $s_k = s_k + \ell$ until the smallest singular value of that SVD is smaller than τ.
6. Set r_k to the maximum singular value of that SVD that is bigger than τ.
7. Set X^k to the truncated SVD using the U, Σ, V computed and using r_k
8. Break the loop if the error is smaller than ε
9. Set the entries of Y^k to $Y_{ij}^{k-1} + \delta(M_{ij} - X_{ij}^k$ if $(i, j) \in \Omega$ and set to zero if otherwise
10. End this iteration
11. End the loop
12. Let $X^{opt} = X^k$ where X^{opt} is the output matrix

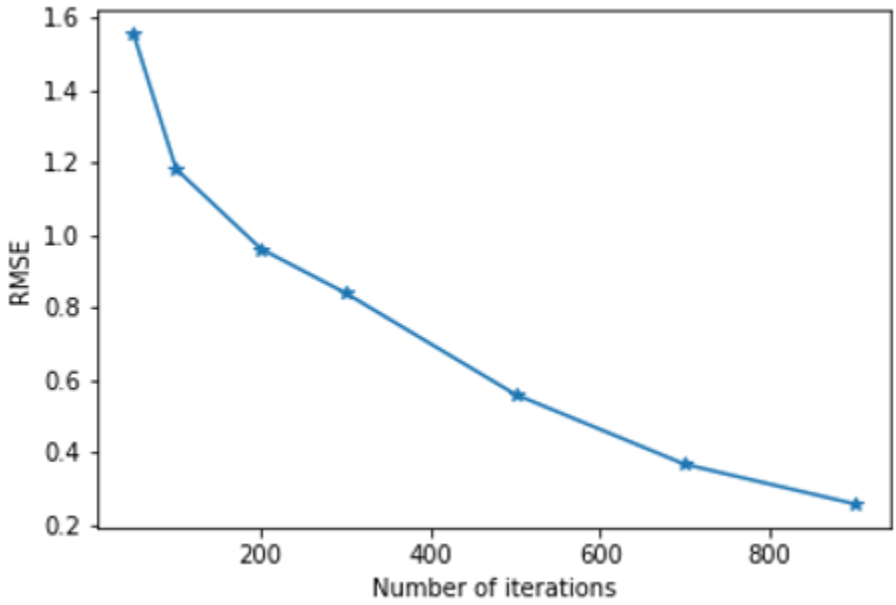
Note that the truncated SVD in step 5 can be easily computed using a library called sparsesvd in python

Testing with SVT

This section demonstrates and tests the performance of Singular Value Thresholding using the MovieLens 1M dataset [2]. The MovieLens 1M dataset contains 1,000,209 ratings with 6040 users and 3883 movies and it is about 95% sparse.

In the original paper [1], the error with generated data is measured using the relative error which is defined as $||X^{opt}-M||_F/||M||_F$ where M is the real unknown matrix. However, the real unknown matrix does not exist with real dataset and for replacement, the root-mean-square error (RMSE) between the non-zero entries of the output matrix and the original matrix is used for error measurement.

By using τ=20000, δ=2, $\varepsilon=0.001$ and ℓ=5 which are some standard values for this 1M dataset suggested by the paper, the result is:



Time:4610s RMSE: 0.257 Rank=153

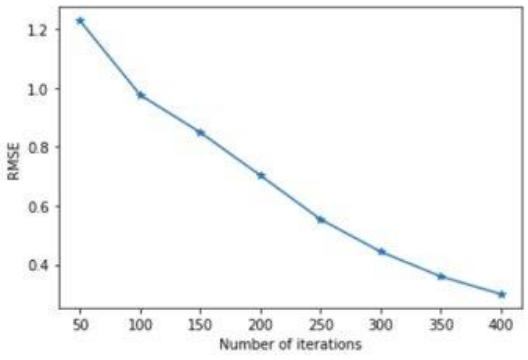
The computation time is not impressive and one possible reason might be the movieLens 1M dataset is not that low rank as imagined. There may be many other underlying factors such as the mood of the user, is the name of the movie appealing etc. beside those factors that are expected. Since SVT is created specifically for low rank matrix, the time performance may not be satisfying.

Another finding is that the computation become slower as the iteration number increases. This is because as the iteration number increases, the rank increases and more SVDs need to be computed.

Testing parameters

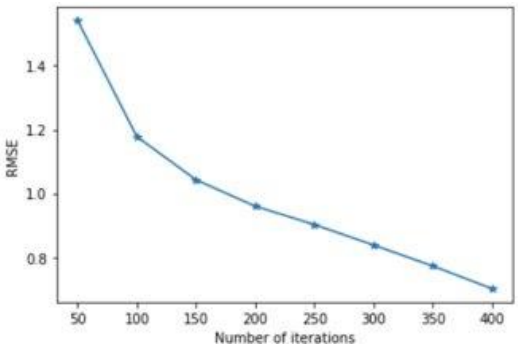
To test SVT with different parameter values, some comparisons are done and unless otherwise specified, the control parameters are τ=20000,δ=2,ε=0.001,ℓ=5,K_{max}=400

τ=10000



Time:2839s RMSE: 0.300 Rank=140

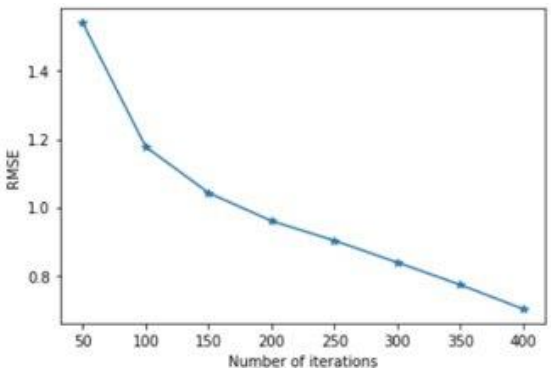
τ=20000



Time:735s RMSE: 0.701 Rank=35

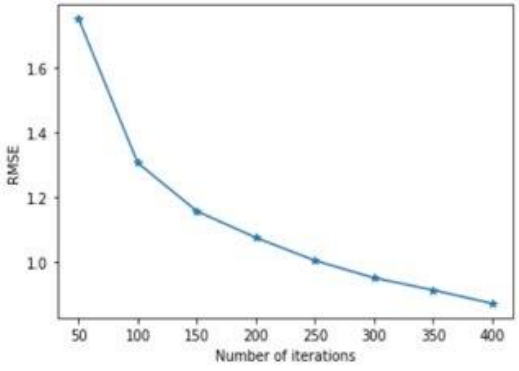
Note that with a lower τ, a smaller RMSE can be reached using fewer iteration, but this also require more SVD computations each iteration, thus resulting higher computation time

τ=20000



Time:735s RMSE: 0.701 Rank=35

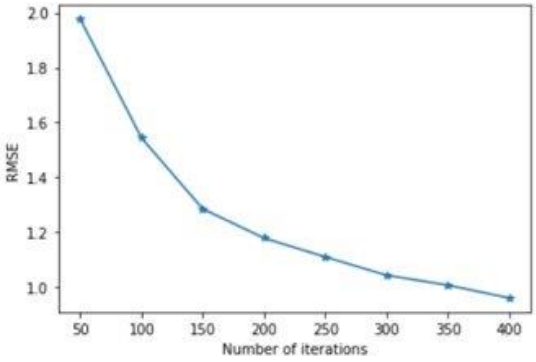
τ=30000



Time:98s RMSE: 0.872 Rank=10

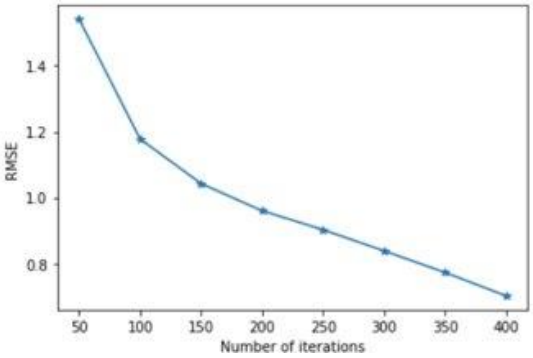
In contrast, using a higher τ require less SVD computations each iteration but this could be too conservative and make the algorithm slow to converge

δ =1



Time:94s RMSE: 0.96 Rank=6

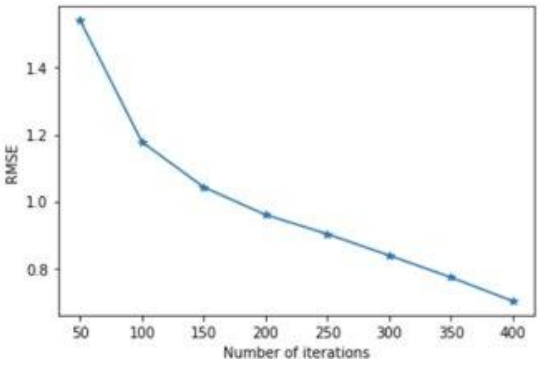
δ =2



Time:98s RMSE: 0.872 Rank=10

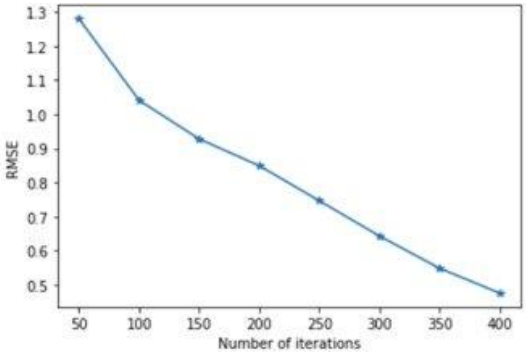
Note that with a lower δ (0-2), SVT is guaranteed to converge according to [1], however, the converge speed is slow and it takes more iterations to converge.

δ =2



Time:735s RMSE: 0.701 Rank=35

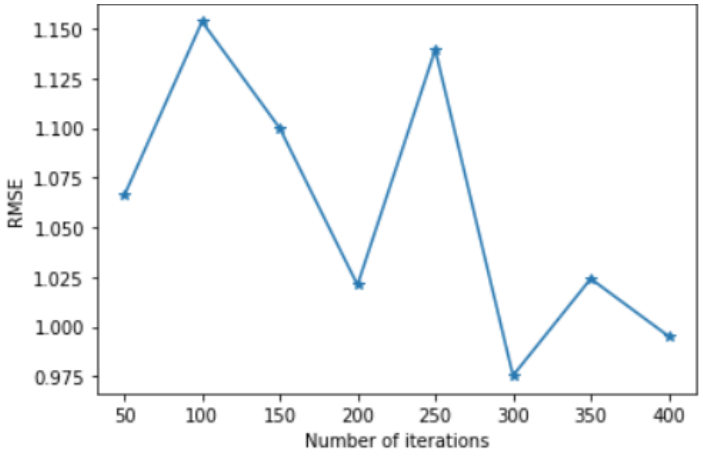
δ =3



Time:2169s RMSE: 0.476 Rank=95

In contrast, using a higher δ can potentially increase the converge speed but this may result the algorithm not converge if δ is set too high as shown in the graph below.

δ =6

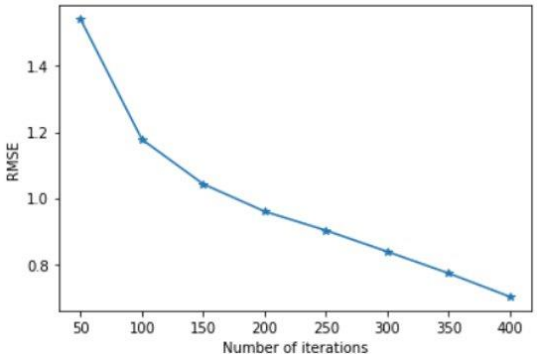


Comparing SVT with Stochastic Gradient Descent

As mentioned, the SVT can be seen as a projected gradient-descent algorithms. Therefore, it is natural that one may want to compare the performance of SVT and the performance of some other gradient descent matrix factorization approaches. In this report, the Stochastic Gradient Descent algorithm (SGD) is chosen for comparison. SGD is fundamentally the original gradient descent approach, but it only uses a mini batch of data instead of all the data for each step [3].

Compare two algorithms using standard parameters:

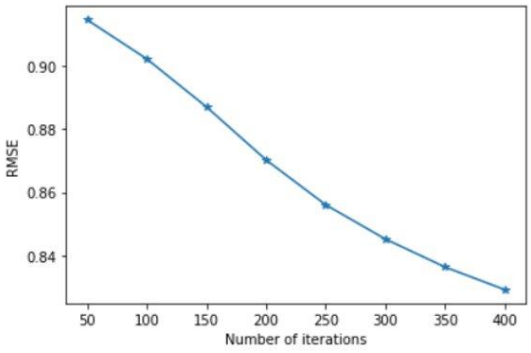
τ=20000, δ=2, ε=0.001, ℓ=5



SVT

Time:735s RMSE: 0.701

learn rate=0.005 batch size = 50000



SGD

Time:665s RMSE: 0.829

From the two graphs, one can observe that with similar amount of time, SVT can reach a much lower RMSE and thus perform better compared to the classic SGD.

Conclusion

The Singular Value Thresholding is created to solve matrix completion where the matrix should be sparse and low rank and is inspired by different iterative soft-thresholding algorithms. It is intuitive and thus easy to understand and implement. It also makes good use of the characteristics of sparsity and low rank of the matrix completion problem. However, this can also be its limit as it can only be used in low rank matrix and may performs poorly against matrix that is not sparse and low rank. Although the performance of SVT is not as impressive as some newer algorithms, it still performs better than the classic gradient descent approach and provides some inspiration to other convex relaxation-based algorithms after SVT.

References

[1] J.-F.Cai, E. J. Candès, Z. Shen "A SINGULAR VALUE THRESHOLDING ALGORITHM FOR MATRIX COMPLETION*," 2010. [Online]. Available: https://cpb-us-w2.wpmucdn.com/blog.nus.edu.sg/dist/d/11132/files/2019/01/SVT-112fnaa.pdf. [Accessed: 01-Dec-2022].

[2] "Movielens," *GroupLens*, 08-Dec-2021. [Online]. Available: https://grouplens.org/datasets/movielens/. [Accessed: 02-Dec-2022].

[3] A. V. Srinivasan, "Stochastic gradient descent - clearly explained !!," *towardsdatascience*, 07-Sep-2019. [Online]. Available: https://towardsdatascience.com/stochastic-gradient-descent-clearly-explained-53d239905d31. [Accessed: 02-Dec-2022].