Exploring Matrix Completions Methods for Recommender Systems

Anthony Hakim **Yifu Hou** **Victor Perez Martin**

Abstract

Recommender systems have been proven to be commercially useful in countless domains. In this project, we explore three different methods to implement a books recommender system via matrix completion and evaluate their performance. We first applied Singular Value Thresholding as the baseline model, and see if improvements are possible using Alternating Least Squares and Probabilistic Matrix Factorization. We found that Probabilistic Matrix Factorization performs the best, but with clear trade offs.

1 Data

We used the *Book-Crossing Dataset* gathered by Cai-Nicolas Zieger, from the Department of Computer Science, the University of Freiburg. The data set contains the relation table *ratings.csv*, a database mapping user id with ISBN and rating (an integer ranging from 0 to 10).

With ratings.csv, we built book-rating matrix X that has $105,281$ unique user IDs (rows) and $340,551$ unique book ISBN numbers (columns). In the rating matrix X, an entry x_{ij} is given by:

$$x_{ij} = \begin{cases} r \in [1, 10], & \text{if the } j^{th} \text{ book is rated by the } i^{th} \text{ user} \\ 0, & \text{if the } j^{th} \text{ book is not rated} \end{cases}$$

In order to carry pilot tests with different matrix completion methods (Singular Value Thresholding and Alternating Least Squares), we introduced other 2 datasets.

- *jesterdata.mat*: a complete database of size (100, 7200) for joke rating scores. Each entry varies from [-10, 10].
- *movies.csv*: a truncated version of the classic MovieLens dataset with 100,000 rows of users, and 4 columns of user id, movie id, rating, and timestamp.

2 Models

2.1 Singular Value Thresholding

We applied Singular Value Thresholding as the baseline model for the matrix completion problem. Our goal is to find the best low-rank approximation to a given $M \times N$ matrix X_0. The approach to evaluate the proximity of our estimation is to compare the squared distance our estimation matrix and our target matrix.

To find:

$$\arg\min_X = ||X - X_0||_F^2 \text{ , subject to } rank(X) = R,$$

Preprint. Under review.

$$where||X - X_0||_F^2 = \sum_{m,n}(X[m,n] - X_0[m,n])^2$$

We first compute the SVD for the incomplete matrix X_0

as $X_0 = U\Sigma V^T = \sum_{k=1}^{k} \sigma_k u_k v_k^T$, where $K = min(M, N)$

With singular value vector, we kill off the smaller singular values and keep the large ones by using a hard threshold γ:

$$\sigma'_k = \begin{cases} \sigma'_k, & \sigma'_k \geq \gamma \\ 0, & \text{otherwise} \end{cases}$$

And we repeat the process until the matrix distance is smaller than our tolerance ϵ.

On a small matrix of size (25, 20), the matrix completion error is reduced to 0.05 at a similar speed regardless of sparsity. The small-sized matrix completion efficiency does not seem to be affected greatly matrix sparsity Figure 1 shows the reduction curve of error size (squared distance) for small matrices with different sparsity ranging from 0.1 to 0.9. In general, the error is effectively reduced within 20 iterations (See Figure 1).

One special case is when the sparsity = 0.1, the error remains at a stable level after the first few iterations. This is because the singular values tend to reduce after each iteration, and eventually become generally smaller than the threshold. Since almost all singular values are filtered out, the model fails to retrieve more information, and the error remains at a fixed level.

On large dataset, SVT suffers more from the problematic threshold. Because the size of the dataset, the singular values change drastically. In our test on the big sample of joke matrix with size (100, 720), the largest singular value from each iteration changed from 680.89844187 to 90.66740445. In this case, a hard threshold can not satisfy the matrix completion problem. It usually falls into the predicament of either missing too much information (with errors remain high after a few iterations) or quickly overfit the model and kill the simulation (with errors dropping to 0 after the first few iterations) (see Figure 3).

Thus, due to the huge expense of computing SVD iteratively and the flaw of hard threshold, Singular Value Thresholding is not the optimal solution to recover enormous matrix such as book rating data. To carry on the task, we introduce Alternating Least Squares and Probabilistic Matrix Factorization.

2.2 Alternating Least Squares

The Alternating Least Squares (ALS) method is used for matrix factorization such that you find the user and item matrices U and V that constructs a non-sparse ratings matrix M. However, we initially do not know the complete matrix M and instead begin with a sparse matrix M_{sparse} which we are to fill using the dot product of the elements of U and V. These elements l_u and l_v are called the **latent features** of their respective user and item matrices, and are learned by ALS method in our case. Latent features compose the item and user matrices, and are characteristic of the ratings matrix M that it hopes to predict.

If $M \in \mathbb{R}^{m \times n}$, then $U^{m \times len(l_u)}$ and $V^{n \times len(l_v)}$, where l_u and l_v are latent features.

The predicted rating $\hat{r} \in M$ is derived by $\hat{r}_{ij} = u_i v_j^T$, where u_i is a vector of U and v_j is a vector of V.

The ALS method minimizes the following loss function with an $l2$ regularization term:

$$Loss = \sum_{i,j \in D} (r_{ij} - \hat{r}_{ij})^2 + \lambda(\sum_i ||u_i||^2 + \sum_j ||v_j||^2)$$

The uniqueness of the ALS method is that to find the appropriate u_i and v_j vectors to construct our user and item matrices U and V, we hold either U or V constant, and minimize the loss function using gradient descent with respect to the non-constant term, and alternate until we reach convergence. In each iteration we solve for user and item vectors:

$$u_i = r_i V (V^T V + \lambda I)^{-1}$$

$$v_j = r_j U(U^T U + \lambda I)^{-1}$$

While we initially instantiate U and V randomly, we want to compute the elements of U and V such that their dot product accurately construct $M = UV^T$, by iteratively computing the vectors u_i and v_j until convergence in the way presented above.

The performance of the algorithms is calculated by masking some of the items in our original matrix M_{sparse}, placing those masked values in a test matrix called M_{test} and then training on this masked matrix to compute $\hat{M}$. Finally we compare the values of $\hat{M}$and M_{test}.

To implement the ALS algorithm, we pre process the books rating dataset to include only non-zero rows and columns. Then we split the dataset into training and testing sets, where we mask a certain number of values in the training set and leave them in the testing set. Afterwards, we intitalize our User and Items vectors randomly based on the shape of our training data. We attempt to calculate the proper Item and User matrices by iterating over the vectors of these matrices, and performing ALS until convergence to find our best estimates of U and V. Last but not least, we calculate and same the mean squared error for each iteration. We optimize for the best hyperparameters using grid search and find that when the number of iterations is 100, the number of latent factors to consider is 20, and the regularization term is 0.01, we arive to the best result.

While the implementation of ALS on the classic movie dataset, the algorithm performs well with a testing error of 8 percent. However, as seen in figure 4 in the appendix, the error rate converged to above 60 percent. This is due to the sparsity of the Books dataset. When we mask the values to be included in the testing set, the training matrix becomes so sparse that there is not enough valuable information in the matrix to train the algorithm well enough to generalize it.

2.3 Probabilistic Matrix Factorization (PMF)

Probabilistic Matrix Factorization considers the rating matrix $X \in \mathbb{R}^{M \times N}$ as a product of two lower rank matrices, $U \in \mathbb{R}^{D \times N}$ and $V \in \mathbb{R}^{D \times M}$, that capture users and book D distinct latent features. Columns U_i and V_j represent the i^{th} user latent features and the j^{th} book latent features, respectively. PMF defines conditional distributions over the observed ratings and a set of priors on users and books feature columns to maximize the log of the posterior over users and books features with respect to columns U_i and V_j. This optimization problem is solved using Gradient Descent or a maximum a posteriori probability (MAP) estimation to find the optimal columns $\hat{U}_i$ and $\hat{V}_j$, and obtain the optimal lower rank matrices $\hat{U}$ and $\hat{V}$. Then we can use data provided by readers with similar preferences to offer recommendations to a specific reader.

Formally, define the conditional distributions over the observed ratings as:

$$p(X \mid U, V, \sigma^2) = \prod_{i=1}^{N} \prod_{j=1}^{M} [\mathcal{N}(x_{ij} \mid U_i^T V_j, \sigma^2)]^{\mathbb{1}_{ij}}$$

where $\mathcal{N}(X \mid \mu, \sigma^2)$ is the probability density function of the Normal distribution with mean μ and variance σ^2, and $\mathbb{1}_{ij}$ is an indicator function that is equal to 1 if user i rated book j and 0 otherwise.

The priors on users and books latent feature U and V, as:

$$p(U \mid \sigma_U^2) = \prod_{i=1}^{N} \mathcal{N}(U_i \mid 0, \sigma_U^2)$$
$$p(V \mid \sigma_V^2) = \prod_{j=1}^{M} \mathcal{N}(V_j \mid 0, \sigma_V^2)$$

Maximizing the log of the posterior is equivalent to minimizing the sum-of-squared-errors objective function with quadratic regularization terms (Mnih and Salakhutdinov, 2017):

$$L = -\frac{1}{2}\left(\sum_{i=1}^{N}\sum_{j=1}^{M}(R_{ij} - U_i^T V_j)^2 + \lambda_U \sum_{i=1}^{N}\|U_i\|_F^2 + \lambda_V \sum_{j=1}^{M}\|V_j\|_F^2\right)$$

where $\lambda_U = \frac{\sigma^2}{\sigma_U^2}, \lambda_V = \frac{\sigma^2}{\sigma_V^2},$ and $\|\cdot\|_F^2$ denotes the Frobenius norm.

We implemented PMF method in the Books ratings dataset, we selected parameters $D = 10$ for the number of latent features, $\lambda_V = 0.30$ and $\lambda_U = 0.30$, number of epochs equal to 150. We split the data into a training sample (80%) and testing sample (20%). We attempted to train the model on different random subsamples of the Books ratings data on a MacBook Air M1 2020 with 16GB of RAM. The time to run the model on a random subsample that gave us a matrix X of approximately 30,000 users and 70,000 books was around 70 minutes. We attempted to run the model on another random subsample that resulted on a matrix X of approximately 60,000 users and 140,000 books. It took our implementation of the model around 450 minutes to run with 150 epochs. We were not able to train the model on random subsamples that resulted on larger raiting matrices due to technical constraints.

The root-mean-square error (RMSE) of the training model in the larger matrix was 3.6906 and the testing error was 3.8964. On the smaller matrix the training RMSE was 3.9065 and the testing one was 4.2286. Figure 5 in the Graphical Appendix shows the performance of the training and testing model, we see that after 40 epochs the RMSE associated to the training model remains constant.

3 Conclusion

In this paper we presented a survey of three different matrix completion approaches for recommender systems: Singular Value Thresholding, Alternating Least Squares and Probabilistic Matrix Factorization. As one of the traditional matrix completion methods, Singular Value Thresholding is able to be operated on small matrix and reduce the error effectively. However, it is not adequate to recover huge matrices due to lack of flexibility. Moreover, computing SVD for large matrix can be extremely expensive. We then implement Alternating Least Squares. We find that although computation is possible and less computationally expensive than SVT, due to the sparsity of the matrix, and the condition determining the successful computation of the ALS method is a minimum value of features per row and column of the sparse matrix, the test error rate is too high for the ALS model to be used as a sufficient book recommender system. Finally, Probabilistic Matrix Factorization (PMF) performed well on relatively small random subsamples of the book raiting matrix, however, it was computationally expensive which prevent us from training the model on larger random subsamples of the data.

References

[1] Contreras Carrasco, Oscar. "PMF for Recommender Systems. Probabilistic Matrix Factorization and Collaborative Filtering". Towards Data Science (2020).

[2] Davenport, Mark A., and Justin Romberg. "An overview of low-rank matrix recovery from incomplete observations." IEEE Journal of Selected Topics in Signal Processing 10, no. 4 (2016): 608-622.

[3] Hastie, Trevor, Rahul Mazumder, Jason D. Lee, and Reza Zadeh. "Matrix completion and low-rank SVD via fast alternating least squares." The Journal of Machine Learning Research 16, no. 1 (2015): 3367-3402.

[4] Liu, Chenghao, Tao Jin, Steven CH Hoi, Peilin Zhao, and Jianling Sun. "Collaborative topic regression for online recommender systems: an online and Bayesian approach." Machine Learning 106, no. 5 (2017): 651-670.

[5] Mnih, Andriy, and Russ R. Salakhutdinov. "Probabilistic matrix factorization." Advances in neural information processing systems 20 (2007).

[6] Ramlatchan, Andy, Mengyun Yang, Quan Liu, Min Li, Jianxin Wang, and Yaohang Li. "A survey of matrix completion methods for recommendation systems." Big Data Mining and Analytics 1, no. 4 (2018): 308-323.

[7] Ziegler, Cai-Nicolas, Sean M. McNee, Joseph A. Konstan, and Georg Lausen. "Improving recommendation lists through topic diversification." In Proceedings of the 14th international conference on World Wide Web, pp. 22-32. 2005.

Graphical Appendix

A.1 Singular Value Thresholding

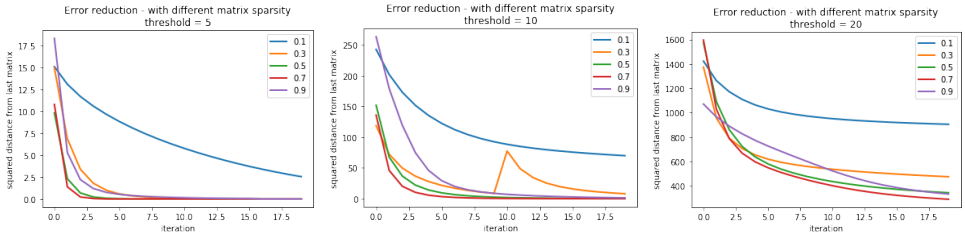

Figure 1: SVT with different matrix sparsity (small dataset)

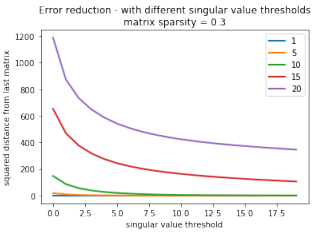

Figure 2: SVT with different thresholds (small dataset, sparsity = 0.3)

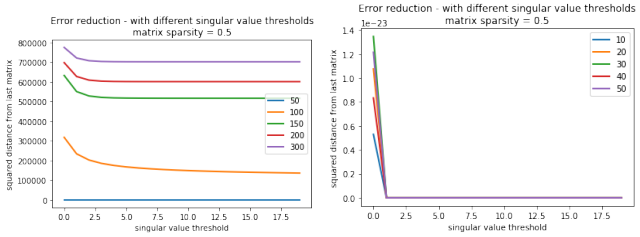

Figure 3: SVT on large dataset

A.2 Alternating Least Squares

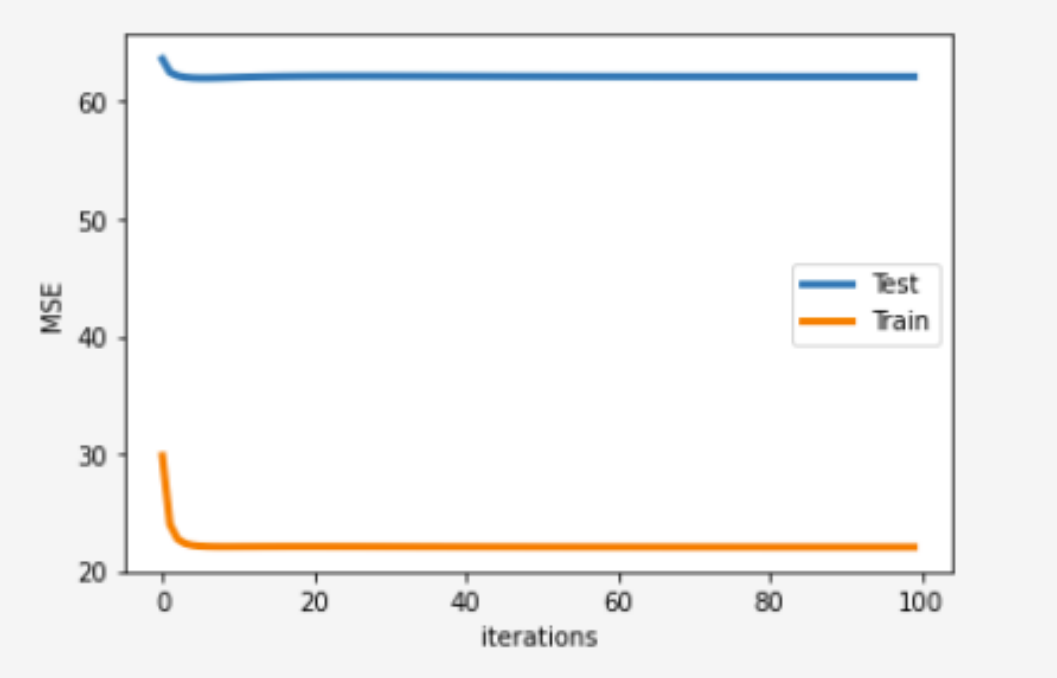

Figure 4: Testing and Training accuracy on Books Ratings Dataset

A.3 PMF

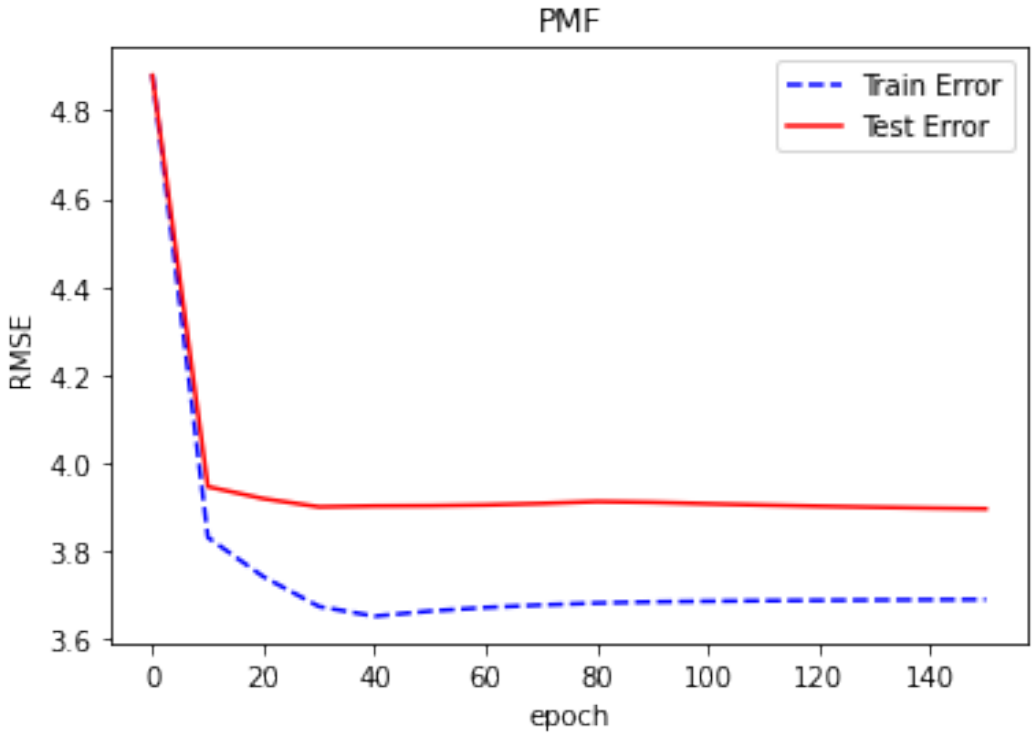

Figure 5: PMF Performance