Recommender System Using Collaborative Filtering

Chenhua Shi
University of California, San Diego
c8shi@eng.ucsd.edu

Abstract

Recommender systems are widely used in our daily life. Generally speaking, recommender systems are based on two strategies that content filtering approach and collaborative filtering. In this report, I focus on the exploration of collaborative filtering, which is based on latent factor model and K-Nearest-Neighborhood (KNN) method. I built models to make rating predictions and Top-N recommendations under different approaches and then compared their metrics. I first applied singular value decomposition (SVD) for collaborative filtering since SVD can effectively reduce dimensionality of recommender system databases and solve the sparsity problem by Latent Semantic Indexing (LSI). In addition, I made user-item rating matrix based on user/item similarities. I used cosine similarity to create a similarity matrix for pairs of users/items. After that, I used KNN to find the k-closest users/items to a target user/item. Lastly, I made rating predictions according to the K nearest neighbors.

This report presents three different experiments where I have explored one technology called collaborative filtering. The first experiment compares the quality of a recommender system using Baseline Model, Pure SVD, Stochastic Gradient Descend (SGD), Alternating Least Squares (ALS), Item-based Similarity, and User-based Similarity. I compared the effectiveness of recommender systems at predicting ratings in the both training sets and testing sets. The evaluation metric of the rating prediction is Mean Absolute Error (MAE) because it is most commonly used and easiest to interpret. The second experiment compares the quality of different recommender systems at producing Top-N items. The evaluation of judging Top-N items is using standard F1 metric, which is based on computing recall and precision. The first part of comparing the effectiveness of Top-N items is using Pure SVD in high dimension and low dimension under binary feature recommender generation. The second part is to compare the effectiveness under Pure SVD, Stochastic Gradient Descend (SGD), Alternating Least Squares (ALS), Item-based Similarity, and User-based Similarity under rating prediction in recommender generation. The third experiment is to evaluate the performance of on-line system and off-line version. The Incremental SVD algorithm is applied to update SVD because the folding-in technique just relies on the existing latent semantic structure. Meanwhile, the evaluation metric of the performance is still Mean Absolute Error (MAE), which is based on different training and test ratio and folding-in model size. My experiments suggest that SVD has the potential to meet different challenges of recommender system at making both rating predictions and recommending Top-N items. SVD based models can also make the recommender system highly scalable and less cost in some degree.

1. Project Note

Here, I describe which sections correspond to the five tasks of the assignment. In the Introduction section, I describe the dataset. Then, I introduce approaches for recommender system and evaluation metrics in experiments. In the Related Works section, I surveyed literatures studying the same problem. This is the section that completes task 1. In the Collaborative Algorithm for Rating Prediction and Top-N Recommendations, I study algorithms for Pure SVD, Stochastic Gradient Descend (SGD), and Alternating Least Squares (ALS) under latent factor model and algorithms for Item-based Similarity and User-based Similarity under KNN method. Rating predictions and Top-N recommendations are based on these approaches. In addition, I study Incremental SVD to explore the performance of on-line system and off-line version. These are the main section that completes task2, task3, task4, and task5. In the Experiments section, I present performances of different approaches in computing metrics of MAE, recall, precision, and F1 and analyze the results. This is the section that also completes task2, task3, task4, and task5.

2. Introduction

In this project, I used Anonymous Ratings from the Jester Online Joke Recommender System. The dataset I use includes 24,983 users who have rated 36 or more jokes. This dataset is available online at http://eigentaste.berkeley.edu/dataset/. It has the following format:

1. Ratings are real values ranging from -10.00 to +10.00 (the value "99" corresponds to "null" = "not rated").
2. One row per user
3. The first column gives the number of jokes rated by that user. The next 100 columns give the ratings for jokes 01 - 100.
4. The sub-matrix including only columns {5, 7, 8, 13, 15, 16, 17, 18, 19, 20} is dense.

I split the data into train/test with 0.8/0.2 ratio. Basically, I randomly draw 20 rated items from each user to form tested user-item rating matrix.

Here's a summary of counts of subsets of all data:

Table 1: Dataset Overall

Set	Count
All Data	2,498,300
Training Data	1,998,640
Testing Data	499,660
User	24,983
Item	100
Rated Item	1,818,986
Not rated Item	679,314

In this section, I also briefly introduce approaches and evaluation metrics I use in experiments. The recommender system, in this case a collaborative filtering system, uses known user-item rating in the dataset to form neighbors to make prediction for unknown ratings to recommend Top-N items. However, collaborative filtering suffers problems of cold start that the system can not make recommender system for unseen users/items due to inability to address the system's new users and products. Although collaborative filtering has cold start problems, it is still effectively and accurately in making predictions and generating recommendations compared with content-based techniques.

The report first introduces Pure SVD to generate rating predictions and make recommendations since SVD is crucial and effective in reducing dimensionality in recommender system. Although SVD will undergo expensive matrix factorization steps, the Incremental SVD algorithm can solve this problem and make the recommender system highly scalable due to SVD's properties. Pure SVD also faces the defect of sparsity. Unknown ratings need to be filled zero or average ratings so that the user-rating matrix can be achieved decomposition. Thus, Stochastic Gradient Descend (SGD) and Alternating Least Squares (ALS) can solve this sparsity problem by applying latent vectors. The system learns the model by fitting previously observed ratings and use λ to regularize learned parameters. On the other hand, I also study KNN method to make rating predictions and recommendations according to either users similarity or items similarity. The neighborhood formation usually uses Pearson Correlation or Cosine Similarity to form a similarity matrix for either users or products. Then, K nearest neighbors are found in this similarity matrix and generate recommendations based on K nearest neighbors because they have most similar features or tastes. The rest part of the report presents the performance of different algorithms. Basically, I use MAE to measure the difference between original rating and predicted rating. Meanwhile, informational retrieval (IR) community such as recall and precision are used to evaluate the metric of Top-N recommendations. I also use standard F1 metric to see the quality of the judgment.

3. Related Works

Recommender systems have become more and more popular in recent years. It is a subclass of information filtering system that could generate rating prediction that a user rates the item. It is widely used in many areas such as movies, music, books, and so on. [3] points out that existing most collaborative filtering based recommender systems that use KNN method to generate predictions and make recommendations have limitations. It faces the problem of sparse rating of neighbors, serious scalability, and synonymy due to either sparse ratings of neighbors or long computation times with large number of users/items. Thus, the accuracy of the recommender system may be poor and time-consuming. Instead, it suggests to apply SVD for collaborative filtering. LSI can not only solve the sparsity but also reduce the dimensionality of the user-item rating matrix. Hence, the SVD-based recommendation system in generating rating predictions and Top-N recommendation is efficient and reliable because the lower dimensional representation of the original user-item space can have almost the same accuracy as in higher dimensional space. We need to try multiple times to find best value of dimensional size k in the experiment. [1] has the same idea as [3] that Pure SVD is consistently the top performer. Although neighborhood models could perform better in RMSE in some degree, they sometimes discard the users/items poorly correlated to the target user/item, which decreases noise for improving the quality of recommendations. [1] argues that accuracy metrics such as recall and precision are better in evaluating Top-N recommendation tasks than common methodologies based on error metrics such as RMSE. Improvements in RMSE can not indeed help accuracy improvement because they do not have any trivial relationship between each other. Thus, accuracy measures are better since they directly access the quality of Top-N recommendations, which is also used in [3]. But, there exits subtle difference in defining recall and precision since [1][3] use different approaches to make Top-N items recommendations.

Stochastic Gradient Descend (SGD) and Alternating Least Squares (ALS) algorithms are demonstrated in [2]. Pure SVD faces the problem of sparsity so that missing ratings need to be filled in to make the user-item rating matrix be dense. However, imputation can be expensive in increasing amounts of data and increase errors in distorting original ratings. Thus, SGD and ALS are better approaches since the algorithm learns the model by fitting previously observed ratings and use λ to regularize learned parameters through using latent vectors. In general, SGD is easier and faster than ALS. But, they are both beneficial and practical in making predictions and recommendations with better results compared to Pure SVD. SVD based recommender system can incorporate addition of new users/items through applying Incremental SVD, which is shown in [4]. Incremental SVD is similarly to Pure SVD to do prediction generation as in [1][3]. But, it also needs to do some proper mathematical justification such as projection and basis based on folding-in technique. The computation of Incremental SVD is efficient because the time complexity is just O (1) that involves dot product only. The evaluation metric is the same as in [1][3] to use MAE to evaluate the error between predicted rating and original rating. In [5], it explains the algorithm and the reasoning about Latent Semantic Indexing (LSI) in detail. In order to update SVD, it uses folding technique to make the project incrementally build on the original model. In the folding-in procedure, we need to compute a projection P that projects N_u onto the basis U_k. Hence, new users or items do not affect the existing user-item matrix under this method. It is not necessary to re-compute the low-dimension matrix when the user-item matrix involves new users or items. The application of folding-in technique can effectively solve time-consuming problem and make the model more scalable. Overall, [1][2][3][4][5] introduces different techniques for designing a recommender system. I apply Baseline Model, Pure SVD, Stochastic Gradient Descend (SGD), Alternating Least Squares (ALS), Item-based Similarity, and User-based Similarity in experiments to compare their MAE and accuracy metrics in generating rating prediction and making Top-N recommendations. Meanwhile, Incremental SVD is used for evaluating the performance of on-line system and the off-line version with the incorporation of new users/items. Hence, techniques and algorithms behind recommender system are worthy of exploration because we want to make the system more effective and more precise.

4. Collaborative Algorithm for Rating Prediction and Top-N Recommendations

4.1 Rating Predication Generation

Rating predication generation is based on collaborative filtering. The mainly two techniques are latent factor model and K-Nearest-Neighborhood (KNN) method. In the experiment, I first generate rating predication according to the Baseline model, which will be beaten by above two techniques. After that, I explore latent factor model by applying Pure SVD, Stochastic Gradient Descend (SGD), and Alternating Least Squares (ALS). Later, I explore KNN method by using Item-based Similarity and User-based Similarity.

4.1.1 Basline Model

The Baseline Model is the most straightforward to compute. I only compute the average rating on all known items for each user and then use these average ratings to form the rating prediction matrix. The error will be the largest because it does not consider the relationship between users and items.

4.1.2 Singular Value Decomposition (SVD)

SVD is a well-known matrix factorization technique. The singular value decomposition of an $m \times n$ matrix R is a factorization of the form USV^H. Here, if R is a real matrix, then U is an $m \times r$ real unitary matrix. S is a $r \times r$ diagonal matrix with non-negative real numbers on the diagonal. V is an $r \times n$ real unitary matrix. Since r is the rank of the matrix R, all entries of the matrix S are positive and stored in decreasing order of their magnitude on the diagonal. SVD is useful in application because SVD provides the best lower rank approximations of the original matrix R in lower dimension. It can be reduced to have k largest singular values to compose new diagonal matrix S_k with the dimension of $k \times k$. Thus, the original matrix U and V will be reduced as well to be the new form U_k and V_k with the dimensionality of $m \times k$ and $k \times n$ respectively. Thus, the reconstructed matrix R_k can be expressed as the form $U_k S_k\ V_k^T$. It can effectively minimize the Frobenius norm$||R - R_k\ ||$ over all rank-k matrices. Since the given user-item rating matrix is sparse because so many unknown ratings are in the dataset, I need to pre-process this user-item matrix so it can do SVD. First of all, the user-item rating matrix is removed sparsity by using item average to fill ninety-nine and then doing subtraction of user average from each rating. After that, the normalized user-item rating matrix can be factorized into U, S, and V. Then, the diagonal matrix S can be reduced to S_k through choosing first k singular values that k is smaller than the rank of S. Last, the rating prediction can be computed through the following formula:

$$C_{P_{pred}} = \bar{C} + U_k\sqrt{S_k}'(c) * \sqrt{S_k}V_k'(p)$$

Finding best value of k can be achieved through running different values of k in the experiment. I need to compare MAE in both training and test results to find the most optimum value k to avoid overfitting.

4.1.3 Stochastic Gradient Descend (SGD)

SGD is a stochastic approximation of the gradient descent optimization that minimizes the objective function iteratively. The system learns the model by fitting previously observed ratings and use λ to regularize learned parameters. It has the following expression:

$$L = \sum_{i,j}(u_i v_j - x_{ij})^2 + \frac{\lambda}{2}\sum_i ||u_i|| + \frac{\lambda}{2}\sum_j ||v_j||$$

In the algorithm, the system predicts a rating and then computes associated prediction error. I need to take gradient on u_i and v_j for each i and j. Thus, gradient descent formulas are shown below:

$$\frac{\partial L_{ij}}{\partial u_i} = 2\left(u_i v_j - x_{ij}\right)v_j + \lambda u_i$$

$$\frac{\partial L_{ij}}{\partial uv_i} = 2\left(u_i v_j - x_{ij}\right)u_i + \lambda v_j$$

SGD needs us to set suitable learning rate, step size and λ to make the algorithm converge and to regularize learned parameters. This procedure needs to be run multiple times to find best parameters so as to generate optimal rating prediction.

4.1.4 Alternating Least Squares (ALS)

In ALS, I assume the predictor has the following form:

$$\text{rating(user, item)} = \alpha + \beta_u + \beta_i + \gamma_u * \gamma_i$$

The system needs to compute each γ_u independently of the other item factor and compute each γ_i independently of the other user factor. Thus, the optimization problem becomes the following form:

$$argmin_{\alpha,\beta,\gamma}\sum_{u,i}\left(\alpha + \beta_u + \beta_i + \gamma_u * \gamma_i - R_{u,i}\right)^2 + \lambda[\sum_u \beta^2{}_u + \sum_i \beta^2{}_i + \sum_i ||\gamma^2{}_i|| + \sum_u ||\gamma^2{}_u||\,]$$

Since the optimization equation is not convex, I have to fix γ_u to solve $argmin_{\alpha,\beta,\gamma_i} objective(\alpha,\beta,\gamma)$ and then fix γ_i to solve $argmin_{\alpha,\beta,\gamma_u} objective(\alpha,\beta,\gamma)$. Hence, I need to use ALS and closed form to solve the problem. The five derivative equations are shown below:

$$\gamma_i(t+1) = \frac{\sum_{u \in U_i}(R(u,i) - \alpha - \beta_u - \beta_i) * \gamma_u}{\lambda + \sum_{u \in U_i} {\gamma^2}_u}$$

$$\gamma_u(t+1) = \frac{\sum_{i \in I}(R(u,i) - \alpha - \beta_u - \beta_i) * \gamma_i}{\lambda + \sum_{i \in I_u} {\gamma^2}_i}$$

$$\alpha(t+1) = \frac{\sum_{u,i \in train} R(u,i) - \beta_u - \beta_i - \gamma_u * \gamma_i}{N_{train}}$$

$$\beta_i(t+1) = \frac{\sum_{u \in U_i} R(u,i) - \alpha - \beta_u - \gamma_u * \gamma_i}{\lambda + |U_i|}$$

$$\beta_u(t+1) = \frac{\sum_{i \in I_u} R(u,i) - \alpha - \beta_i - \gamma_u * \gamma_i}{\lambda + |I_u|}$$

Since γ_u and γ_i are both randomly initialized, I need to run the algorithm multiple times to find the optimal regularize λ in the experiment. Thus, I can gain the best prediction result without overfitting or underfitting.

4.1.5 Item-Based Similarity

Item-based Similarity is computed based on the common raters. The similarity between items is measured by Cosine Similarity or Pearson Correlation Coefficient. Formulas of similarity between item i and item j are shown below:

$$\mathrm{Sim}(i,j) = \frac{\sum_{u \in U_i \cap U_j}(R_{u,i} - \overline{R_u})(R_{v,j} - \overline{R_u})}{\sqrt{\sum_{u \in U_i \cap U_j}(R_{u,i} - \overline{R_u})^2 \sum_{u \in U_i \cap U_j}(R_{u,j} - \overline{R_u})^2}}$$

$$\mathrm{Sim}(i,j) = \frac{\sum_{u \in U_i \cap U_j} R_{u,i} R_{v,i}}{\sqrt{\sum_{u \in U_i \cap U_j}(R_{u,i})^2 \sum_{u \in U_i \cap U_j}(R_{u,j})^2}}$$

Then, KNN approach can be applied to find Top-K items rated by user u that are most similar to item i in predicting rating $R_{u,i}$. Last, the rating prediction can be achieved through computing the following equation (knn(i) represents the set of KNN to i):

$$P_{u,i} = \frac{\sum_{j \in knn(i)} \mathrm{sim}(i,j)(r_{u,i})}{\sum_{j \in knn(i)} |\mathrm{sim}(i,j)|}$$

The above formula shows the weighted average of the ratings of i's K nearest neighbors. The algorithm also needs to be run multiple times since I need to find optimal k value for target items in KNN approach that generates best rating prediction in the experiment.

4.1.6 User-Based Similarity

User-based Similarity is computed based on the common items. The algorithm and logic are almost the same as Item-based Similarity. The similarity between users is measured by Cosine Similarity or Pearson Correlation Coefficient. Formulas of similarity between user u and user v are shown below:

$$\mathrm{Sim}(u,v) = \frac{\sum_{i \in I_u \cap I_v}(R_{u,i} - \overline{R_u})(R_{v,i} - \overline{R_v})}{\sqrt{\sum_{i \in I_u \cap I_v}(R_{u,i} - \overline{R_u})^2 \sum_{i \in I_u \cap I_v}(R_{v,i} - \overline{R_v})^2}}$$

$$\mathrm{Sim}(u,v) = \frac{\sum_{i \in I_u \cap I_v} R_{u,i} R_{v,i}}{\sqrt{\sum_{i \in I_u \cap I_v}(R_{u,i})^2 \sum_{i \in I_u \cap I_v}(R_{v,i})^2}}$$

Then, KNN approach can be applied to find Top-K users that are most similar to user u in predicting rating $R_{u,i}$. Last, the rating prediction can be achieved through computing the following equation (knn(u) represents the set of KNN to u):

$$P_{u,i} = \bar{r_u} + \frac{\sum_{v \in knn(u)} sim(u,v)(r_{v,i} - \bar{r_v})}{\sum_{v \in knn(u)} |sim(u,v)|}$$

The above formula expresses the average of user u plus the weighted average of the ratings of u's K nearest neighbors. The algorithm also needs to be run multiple times since I need to find optimal k value for target users in KNN approach that generates best rating prediction in the experiment.

4.2 Recommendation Generation

Recommender generation is executed in two different ways to make Top-N items recommendation. In the first approach, the user-item rating matrix needs to be converted to binary feature. After that, the user-item rating matrix can be factorized into lower-dimension and application of Cosine Similarity and KNN to find most frequent Top-N items are recommended through scanning the given user's neighborhood. In the second method, the rating prediction is used for recommending Top-N items. I first need to generate test set that contains Top T ratings from items for each user. Then, I randomly select X additional items unrated by user u. I need to rank them to see whether the rank of the test item is smaller than N top ranked items in the list.

4.2.1 Binary Feature in Generating Recommendation

The Top-N recommendation is based on the neighbors' opinions on items they have purchased. This means I need to compute user similarity and find K nearest neighbors to make Top-N items recommendation. First, the user-item rating matrix needs to be converted into binary. Basically, each non-zero entry will be regarded as one and ninety-nine entry will be regarded as zero because we only care about whether the user has rated items but not how much he/she likes the item. I need to compare the difference between high dimensional neighborhoods and low dimensional neighborhoods through computing standard F1 metric.

4.2.2 Rating Prediction in Generating Recommendation

The test set is generated and contains Top 30 ratings from items for each user. Then, I randomly select 20 additional items unrated by user u. I first need to come up with the rating prediction and then rank them to see whether the rank of the test item is smaller than N top ranked items in the list. Let p denote the rank of the test item i within the list. I have to compare whether p $\leq N$ and have a hit if so. This means the item i will be in the Top-N list that recommends to users. Meanwhile, the chance of hit will increase with the increasing of N.

4.3 Incorporate Addition of New Users/Items

Although SVD-based recommendation can provide high quality recommendations, it takes a long time to experience matrix factorization, especially for considering new users. Thus, Incremental SVD algorithm can effectively make the recommender system highly scalable and accuracy.

4.3.1 Incremental SVD

Usually, the recommender system involves both off-line and on-line steps. In order to make the off-line model of building SVD more scalable, Incremental SVD algorithm is applied because SVD can allow the model to be incrementally computed. It uses folding-in technique to make the project incrementally build on the original model. In the folding-in procedure, I need to compute a projection P that projects N_u onto the basis U_k. Thus, the formula can be expressed as below:

$$\mathrm{P} = \mathrm{U_k} * \mathrm{U_k^T} * \mathrm{N_u}$$

More specifically, let us consider Latent Semantic Indexing (LSI) and the folding-in process in updating SVD. Since folding-in just relies on the existing latent semantic structure, A_k, new terms and documents will not influence existing terms and documents. After new documents or terms have been computed, they have to be appended to the existing document vectors or columns of V_k or the existing term vectors or columns of U_k. To fold-in a new $m \times 1$ document vector d (new item) or a new $1 \times \mathrm{n}$ term vector t (new user) to existing LSI model can be expressed as following:

$$\hat{\mathrm{d}} = \mathrm{d^T U_k S_k^{-1}}$$
$$\hat{\mathrm{t}} = \mathrm{t V_k S_k^{-1}}$$

Hence, for adding new documents (new items), the LSI structure becomes $A_k = U_k S_k V_k^T$ with the dimension size of
$m \times (n + p), m \times k, k \times k$, and $k \times (n + p)$ respectively. Here, V_k^T is the concatenation of the original online part with new items that project onto the current product vectors $\hat{\mathrm{d}}$. Similarly, for adding new terms (new users), the LSI structure becomes $A_k = U_k S_k V_k^T$ with the dimension size of $(m + q) \times \mathrm{n}$, $(m + q) \times k$, $k \times k$, and $k \times n$ respectively. Here, U_k is the concatenation of the original online part with new users that project onto the current product vectors$\hat{\mathrm{t}}$. As we can see, new users or items do not affect the existing user-item matrix under this method. It is not necessary to re-compute the low-dimension matrix when the user-item matrix involves new users or items.

5. Experiment

Recommender systems are usually different types of measures for evaluating the quality of rating predication and Top-N items recommendation. Here, I use MAE to measure the rating prediction and use accuracy metric to evaluate the performance of recommendations.

5.1 Prediction Evaluation Metrics

Generally speaking, Mean Absolute Error (MAE), Root Mean Squared Error (RMSE) and Correlation between ratings and predictions are widely used metrics to measure the accuracy in statistical. It can present the numerical score by comparing the predicted rating scores against the actual scores in the dataset. In the first experiment, I use MAE to compare different collaborative filtering methods in both training sets and test sets. MAE directly shows the different between prediction and actual rating, which clearly tells us which approach is better in this dataset. Usually, the lower the MAE, the more accurate the system in making rating prediction. Assume I have rating-prediction pairs $<p_i, q_i>$, the formula of MAE can be expressed as below:

$$\mathrm{MAE} = \frac{|\sum_{i=1}^{N} p_i - q_i|}{N}$$

5.2 Top-N Recommendation Evaluation Metrics

Accuracy metrics that recall and precise are used to measure the performance of Top-N recommendations. Error metrics such as RMSE is not used here because it can not really help improve accuracy. In the binary feature in generating recommendation, I first define hit set be the intersection set of test set and Top-N set. Then, recall and precision are defined as in the following:

$$\mathrm{Recall} = \frac{\text{size of hit set}}{\text{size of test set}} = \frac{|\mathrm{test} \cap \mathrm{Top\ N}|}{|\mathrm{test}|}$$

$$\mathrm{Precision} = \frac{\text{size of hit set}}{\text{size of Top N set}} = \frac{|\mathrm{test} \cap \mathrm{Top\ N}|}{N}$$

Meanwhile, standard F1 metric is also used here since it can fairly judge the quality of the combination of recall and precision. F1 is calculated as following though considering equal weights in both recall and precision.

$$\mathrm{F1} = \frac{2 * \mathrm{Recall} * \mathrm{Precision}}{(\mathrm{Recall} + \mathrm{Precision})}$$

However, In the rating prediction in generating recommendation, the recall and precision are defined a bit different. They have following expressions:

$$\mathrm{Recall}\ (N) = \frac{\#\ \mathrm{hits}}{|\mathrm{test}|}$$

$$\mathrm{Precision} = \frac{\#\ \mathrm{hits}}{N\,|\mathrm{test}|} = \frac{\mathrm{Recall}\ (N)}{N}$$

5.3 Rating Predication Results

Figure 1 shows the relationship between MAE in both training and test set and dimension k under Pure SVD approach. The goal is to find the optimal value k in lower dimensionality since it can effectively reduce the reconstructed matrix error. Through observing the result, I see that MAE gradually decreases with the increasing size of dimensionality in the training set. This phenomenon satisfies my expectation because the higher dimension size k can better reconstruct the original rating matrix due to larger number of non-zero singular values. MAE first decreases and then increases with the increasing size of dimensionality in the test set. It shows that there exists a turning point in the dimensionality of zero to twenty. Hence, it is easier for us to choose the optimal value k in this range and we can avoid overfitting through analyzing this plot. Figure 2 is the zoom-in plot that clearly shows the dimensionality between zero to fourteen, which helps us to choose the optimal value k better.

Figure 1. Determination of optimal value k

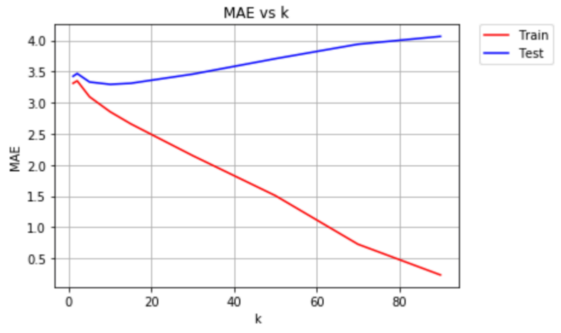
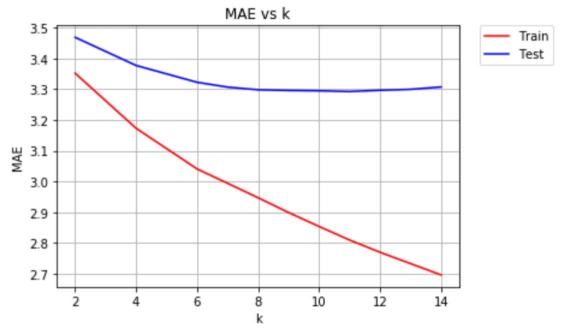
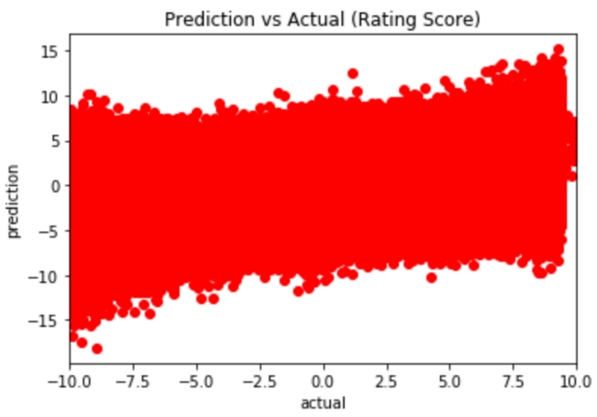


Figure 2. Zoom-in for choosing k



Thus, I find 10 is the best value for k since it has the lowest MAE in test set and avoids underfitting and overfitting. Figure 3 displays the relationship between prediction and actual rating score. I observe that the error range is pretty big due to 27.19% unknown ratings in the dataset.

Figure 3. Prediction vs Actual Score



After that, I run different models under methods I describe in 4.1 Rating Predication Generation. For the Pure SVD approach, the optimal dimensional value is 10 because of the best performance in the lower dimensional space. For the KNN approach, I first use Cosine Similarity to gain the similarity matrix for both items and users. Then, I try different K values in K nearest neighbors and different similarity values to find best performance in both ItemBased Similarity and UserBased Similarity models. Last, I apply ALS and SGD models. For the ALS, I choose K=1 for the latent factor model so that γ_u will be an $2{,}498{,}300 \times 1$ random initialized matrix and γ_i will be a 100×1 random initialized matrix. For the SGD, I fix alpha be 0.0001 and regularizer λ be 0.02. I adjust step size to make the algorithm converges so that it can give me the best performance result. Seventeen sets of data results are shown in the table below:

Table 2: Model Performance

Model	Training MAE	Test MAE
Baseline	3.6528	3.7325
Pure SVD (dimK = 10)	2.8561	3.2914
ItemBased (Without KNN)	3.6023	3.6204
ItemBased (KNN = 5)	3.8526	3.6184
ItemBased (KNN = 20)	3.4426	3.4641
ItemBased (KNN = 30)	3.4202	3.4381
ItemBased (Similarity > 0.3)	3.6838	3.7052
UserBased (Without KNN)	3.4284	3.5044
UserBased (KNN = 5)	3.4202	3.4381
UserBased (KNN = 20)	3.4237	3.5401
UserBased (KNN = 1000)	3.3794	3.4768
UserBased (Similarity > 0.3)	3.4636	3.5173
ALS (k = 1)	2.8127	3.2869
SGD (Step = 10)	3.9231	3.9673
SGD (Step = 20)	3.3185	3.413
SGD (Step = 50)	3.0241	**3.1764**
SGD (Step = 100)	**2.7006**	3.2869

From Table 2, I observe that latent factor model has better performance than KNN method. SGD with step size 100 has the smallest MAE in both training and test set. Pure SVD and ALS have similar performance as SGD in the rating prediction. SGD and ALS have good better predicted results since both of the system learn the model by fitting the previously observed ratings and then make prediction for future unknown ratings. On the contrast, KNN approaches based on either ItemBased Similarity or UserBased Similarity do not have expected performance. 27.19% unknown ratings in the dataset may result in the difficulty in finding similarity between users/items. On the other hand, KNN approach discards the users/items poorly correlated to the target user/item, thus decreasing noise for improving the quality of recommendations. As we can see, results from Table 2 encourage the use of latent factor models such as SVD, ALS, and SGD in collaborative filtering for rating prediction generation. SVD algorithm not only fits well with the collaborative filtering data but, it also saves time complexity in computing huge amounts of datasets. ALS and SGD will potentially provide better predicted results but they need longer time to compute to make their algorithms converge. In a conclusion, latent factor models perform better than KNN approach in the experiment.

5.4 Recommendation Generation Results

The first part of the experiment is based on binary feature in generating recommendation, which I have described the procedure in 4.2.1. Here, I use accuracy metrics to compare different models in generating Top-N recommendations. Basically, I either fix the dimension size k or KNN to observe the different performances among these models. The accuracy metrics are shown in the table below:

Table 3: SVD Accuracy Metrics

Model	Recall	Precision	F1
SVD (dimK=10, KNN=5)	0.888	0.2989	**0.4494**
SVD (dimK=20, KNN=5)	0.8936	**0.299**	0.4222
SVD (dimK=10, KNN=50)	0.7832	0.2963	0.3599
SVD (High Dim, KNN=5)	**0.8993**	0.2972	0.4222
SVD (High Dim, KNN=50)	0.8193	0.294	0.4

From the table above, I can see that SVD in high dimensional neighborhood (original user-item rating matrix) with smaller KNN neighbors has better recall. This demonstrates that the recommender system returns most of relevant results. SVD in lower dimensional neighborhood (reduced space by SVD) with smaller KNN neighbors has better precision. This shows that the system returns substantially more relevant results than irrelevant ones. Last, I use F1 to explore the relation in weighted average of the recall and precision. I observe that SVD in lower dimensional neighborhood with smaller KNN neighbors has the best performance in Top-N recommendation task due to the largest F1 score. Theoretically, F1 score reaches its best value at 1 (perfect precision and recall) and worst at 0. Thus, the larger F1 score reaches, the better accuracy the system achieves. As we can see, the smaller k is sufficient to provide a useful approximation for the original space in making Top-N recommendations. Meanwhile, I need to choose smaller K nearest neighbors in the experiment due to the sparsity. Thus, I can conclude that SVD plays a good performance in recommender system generation.

The second part of the experiment is rating prediction in generating recommendation that I have talked about the method in 4.2.2. Here, I use the same accuracy metrics as in the first part to evaluate the performance of SVD, ItemBased Similarity, UserBased Similarity, SGD, and ALS. Accuracy metric results are shown in the following:

Figure 4. Recall at N

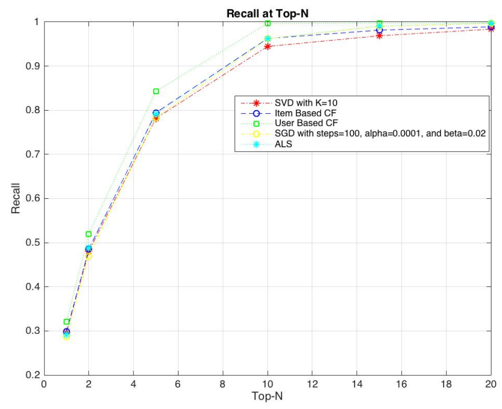


Figure 5. Precision at N

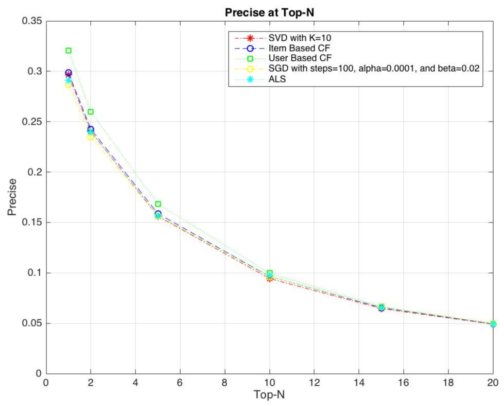
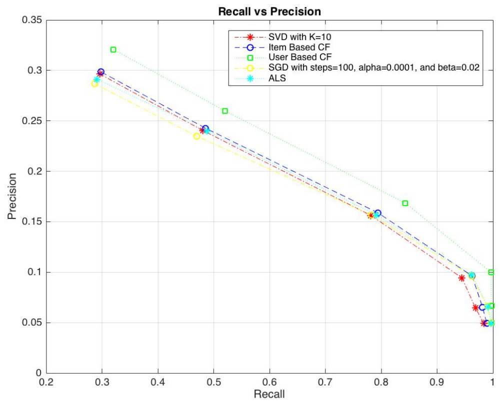


Figure 6. Recall vs Precision



From these plots, I can see that UserBased Similarity has the best performance in generating Top-N recommendations in terms of recall and precision. The rest four models have similar performances. Through observing curves in Recall vs Precision, I find that high precision is at the cost of lower recall and high recall is at the cost of lower precision. UserBased CF curve is always higher than other four curves, which shows its better accuracy in recommending Top-N items. For instance, the recall of UserBased CF at N=5 is 0.84. This means that the model has a probability of 84% to place an appealing movie in the Top 5. The result meets my expectation because the dataset includes 24,983 users that are much larger than the number of items. Huge amount number of users can provide more features in finding similarity. Although other four models' performance are not as good as UserBased Similarity, their performances are still acceptable because their accuracy metrics are close each other. This point is similar to the result that SVD++ has a recall close to that of NNCosNgbr in [1]. On the other hand, larger values of N can be ignored because there is not much obvious difference between 15 and 20 in the experiment. Overall, KNN approach performs well in Top-N recommendations generation. The UserBased CF or ItemBased CF can be applied here, depending on the larger number of users/ items in the dataset.

5.5 On-line System vs the Off-line Version

In the experiment, I still use MAE to measure the deviation of prediction from the original ratings. The dimension size I choose is k = 10 since it outperformances k in other dimension sizes. Then, I need to find the optimal basis size through projecting the rest of (total-basis) users/items to the SVD model by using the folding-in technique. In both the addition of new users/items, I divide my training and test set into 8:2 ratio and 5:5 ratio. For the incorporation of new items, I start with a model size of 20 and go up to 95 with an increment of 20 at each step. While, for the incorporation of new users, I start with a model size of 100 and go up to 5,000 with unfixed steps and then go up to 20,000 with an increment of 5,000 at each step. Plots of MAE vs Folding-in Sizes are shown in the following:

Figure 7. Addition of new Items at ratio 8:2

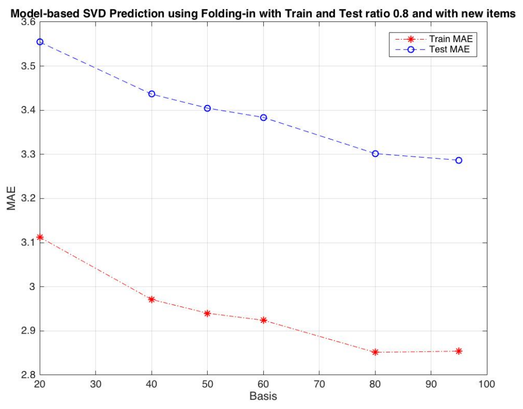


Figure 8. Addition of new Items at ratio 5:5

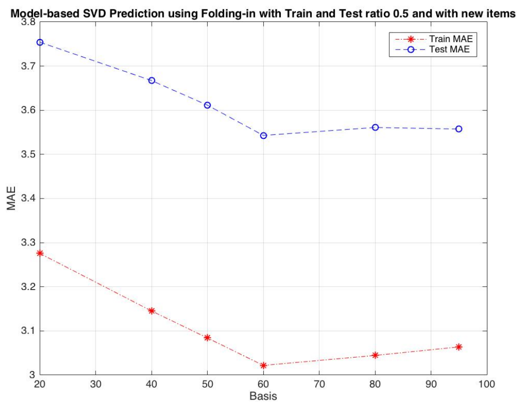


Figure 9. Addition of new Users at ratio 8:2

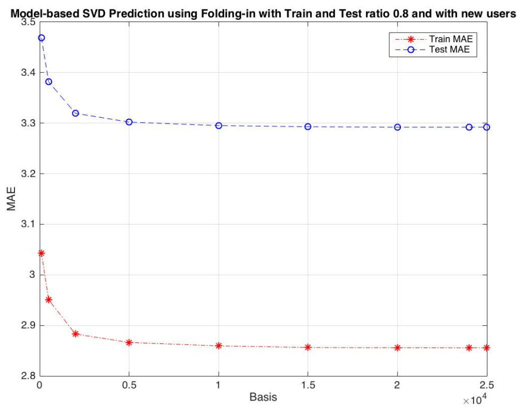
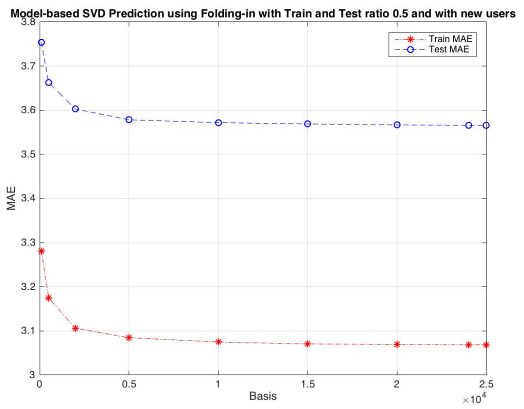


Figure 10. Addition of new Users at ratio 5:5



From above plots, I observe that the optimal basis size will be 70 in the addition of new items and will be 10,000 in the addition of new users. Although I start with a small basis size, I can still gain a pretty good predicted result. Take the addition of new items at ratio 8:2 for example, MAE is around 3.28 at full model size and 3.33 for a model size of 70 in test. This result means only 1.502% quality drop. Meanwhile, let us have a look for the addition of new users at ratio 8:2. MAE is around 3.288 at full model size and 3.293 for a model size of 10,000. This result means only 0.121% quality drop. From these results, we can see that the inexpensive folding-in technique is useful and efficient in achieving a good quality of rating prediction with small basis size. The results are almost the same as in [4]. On the other hand, when the basis size exceeds the optimal basis size, the error is almost negligible because the system is scalable. Thus, the performance of on-line system is as good and scalable as the off-line version. When I incorporate the addition of new items/users, the system still can make acceptable and good rating prediction as the original one. In conclusion, the quality of the system does not change dramatically with varying model size. The folding-in technique works well in achieving high scalability and good predictive accuracy in recommender system. In this way, the model building that requires time-consuming can be done offline and infrequently. The actual execution to do prediction generation can be accomplished online. Hence, Incremental SVD can be especially useful and powerful in combining the off-line system and on-line system.

6. Conclusion

Recommender system is powerful in recommending items for user in current modern life. This report shows that collaborative filtering is a useful technique in recommender system, especially using latent factor model such as SVD. It is easier to implement SVD to generate good predicted ratings and Top-N recommendations in reducing the dimension of the user-item rating matrix. Usually, SVD will perform better than KNN method in a denser dataset. This point has been shown in the experiment. Although computing SVD is expensive, this procedure can be done off-line. Incremental SVD algorithm can stimulate the performance of incorporation of new items/users that not only saves operation cost but also achieve a good predicted rating. Even if the quality may be poorer due to the non-orthogonality of the folding-in technique, it is still widely used in huge amounts of dataset since it requires less computing time and storage space. This strengthen can not be achieved by other collaborative filtering methods. Although ALS, and SGD have better performance in the experiment, they require more computing time to converge. KNN approach will have better quality in rating prediction generation and Top-N recommendations under quite sparse detests. Overall, SVD is highly recommended to use in the recommender system due to good predictive quality, scalability, and efficiency. Models in the experiments still exist weaknesses. For instance, when I evaluate the performance of Top-N recommender systems, we can take long-tail distributions into consideration. This probably will be helpful for us recommend non-trivial items that appeal users. Another weakness is I do not evaluate the throughput (predictions/seconds) based on different basis in the exploration of Incremental SVD algorithm. The improvements on these two aspects will make models richer, helping me gain more information in analyzing results and exploring recommender systems.

Further study is needed to explore the combination of computing SVD in lower dimensionality and choosing K nearest neighbors. Meanwhile, for the large dataset, we need to figure out how often we need to update the off-line version to achieve the higher quality prediction. Lastly, we can explore how to fill unknown ratings in the user-item rating matrix so as to make SVD more reliable and powerful in recommender system.

Reference:

[1] P. Cremonesi, Y. Koren, and R. Turrin. \Performance of recommender algorithms on top-n recommendation tasks." In Proceedings of the fourth ACM conference on Recommender systems, pp. 39-46. ACM, 2010.

[2] Y. Koren, R. Bell, and C. Volinsky. \Matrix factorization techniques for recommender systems." Computer vol. 42, no. 8 (2009).

[3] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. \Application of dimensionality reduction in recommender system-a case study", innesota Univ Minneapolis Dept of Computer Science 2000.

[4] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. \Incremental singular value decomposition algo-rithms for highly scalable recommender systems." In Fifth International Conference on Computer and Information Science, pp. 27-28. Citeseer, 2002.

[5] M. W. Berry, S. T. Dumais, and G. W. O'Brien. \Using linear algebra for intelligent information retrieval." SIAM review, vol.37, no. 4, pp. 573-595, 1995.