COEN 281 Term Project

Predicting Movie Ratings of IMDB Users

By

Team #3

Jingqiu Zhou(Jenny)

Mingyuan Xiao(Amity)

Xiaoguang Mo(Michael)

In

Computer Science and Engineering

At

Santa Clara University

DECLARATION

We hereby declare that all the work done in this Project is of our independent effort. We also certify that we have never submitted the idea and product of this Project for academic or employment credits.

_____

Jingqiu Zhou, Mingyuan Xiao, Xiaoguang Mo

ACKNOWLEDGEMENT

We would like to express our great gratitude towards our supervisor, Dr. Ming-Hwa Wang who had given us invaluable advice to this project.

Table of Contents

Abstract

Data mining is playing a significant role in today's information world.

Nowadays data scientists make use of the infinite information to provide a better way that benefits the entire society. In this project, at least three most important algorithms of data mining will be investigated in details and compared to screen out the best approach of predicting the rating of movies of IMDB users.

1. Introduction

1.1 Objective

Our object is to build a system to predict IMDB users' rating about movies with different algorithms and compare their performance through a benchmark.

1.2 What is the problem

The problem can be rephrased as which kind of algorithm could provide the most accurate recommendation for IMDB users.

1.3 Why this is a project related to this class

We think this project is related to this class because recommendation system is one of the most typical approaches of data mining. To predict the user's possible preference for different movies, data scientists need to dig deep inside the relation between user and movie, cluster user with similar taste and movies with akin types, then create an algorithm that could beat most of other approaches in majority of situations.

1.4 Why other approach is no good

Since IMDB has been working on the recommendation system for many years, countless algorithms were invented to improve the accuracy of

prediction. However, it is lack of the comparison of the recent algorithms, because users' behaviors changes all the time, every little change of the webpage of IMDB or the movie industry might cause significant effect on users' rating habit according to the butterfly effect. So data scientists should track the performance of different algorithms and make sure the best one if chosen to provide the most effective prediction.

1.5 Why you think your approach is better

We think our approach is better because we chose the best algorithms after a comprehensive investigation of all popular algorithms.

1.6 Statement of the Problem

The statement of this problem can be concluded as predicting the movie ratings according the users' behavior and movie category.

1.7 Area or Scope of Investigation

The arenas of this project include data mining, machine learning and recommendation system.

2. Theoretical bases and literature review

2.1 Definition of the Problem

We would predicate how a user would rate a given movie with the data set, which contains (user, movie, rating) triplets where rating is discrete number (from 1 to 5). Specifically, suppose we have a u x m matrix R, which contains the actual ratings by the users, where u is the number of users and m is the number of movies. The matrix R is sparse, lots of elements is unknown as user will not rate every movie, our task is to predicate those

unknown elements.

2.2 Theoretical Background of the Problem

Movie rating prediction can be treated as recommendation problem in movie domain. Recommendation is concerned with learning from noisy observations (x, y) , where $f(x) = \hat{y}$ has to be determined such that $\Sigma(y - \hat{y})^2$or $\Sigma|y - \hat{y}|$is minimal.

A huge variety of different learning strategies have been applied trying to estimate $f(x)$, including:

- Non parametric neighborhood models (Collaborative filtering)
- MF models, SVMs, Neural Networks, Bayesian Networks

2.3 Related Research to Solve the Problem

Collaborative filtering is widely used approach in recommendation domain, it can be further divided into two categories:

- User-based nearest neighbor CF
- Item-based CF

Other than CF, there are also many other approaches to make recommendation

- Matrix factorization techniques (SVD, PCA)
- Associate rule mining
- Probabilistic models (PLSA) probabilistic Latent Semantic Analysis
- Slope one predictor

2.4 Our Solution to Solve This Problem

Combining multiple approaches to make better prediction.

We choose following methods:

1. Incremental SVD on rating matrix with regularization
2. Item-Based Collaborative Filtering Recommendation (IBCF)
3. User-based Collaborative Filtering (UBCF)

We still has one bonus method to solve this problem:

4. Combination of User-based Collaborative Filtering (UBCF) and Support Vector Machine (SVM)

2.5 Our Solution Different From Others

We use three popular and good performance methods to predict the movie rating and compare their performance.

2.6 Why Our Solution Is Better

Firstly, as we use three different methods to predict the movie ranting, then we could compare their performance. Secondly, we could combine the predictions to get better result.

3. Goals

Our goal is to study different recommendation approaches in the movie rating prediction domain and combine them together to achieve better prediction accuracy in real dataset in terms of mean squared error and mean absolute error.

4. Methodology

4.1 How to generate/collect input data

In this section, we would like to use the open data sets of IMDB from *Grouplens* (http://grouplens.org/datasets/movielens/). We will classify this data set into two parts: the training data, and the testing data. We will also use some data as sample submission. Specifically, we will use MovieLens 100k as the dataset. MovieLens 100k data has 100k ratings from 943 users on 1682 movies. It provides 7 datasets, which divided the dataset into training set and test set. u1-u5 data sets split the data in 80/20 fashion to generate training and test data.While ua and ub split the test data with user exactly 10 ratings in it.

4.2 How to solve the problem

Three procedures are provided to implement the movie rating prediction system in this project. The first one is to clean the data to make the input data become the output format that can be processed by the program. Secondly, at least three algorithm methods will be used and implemented in this project. Thirdly, besides outputting the prediction of the rating that the user will mark for the movie, the RMSE (Root Mean Square Error), which evaluates the performance of different algorithms, will be used.

4.2.1 Algorithm Design

Every member in our project will implement at least one algorithm to predict the movie ratings. Three or four algorithm methods will be implemented. They are respectively incremental SVD on rating matrix with regularization, support vector machine, PLSA (Probabilistic Latent Semantic Analysis), and item-based collaborative filtering recommendation.

a). SVD

As for the SVD method, it states that every m x n matrix A can be written as

A = USVT where: U is an mxm orthogonal matrix, S is an m x n diagonal matrix with singular values of A along the diagonal, and V is an m x n orthogonal matrix. The SVD theorem is as shown as follows:

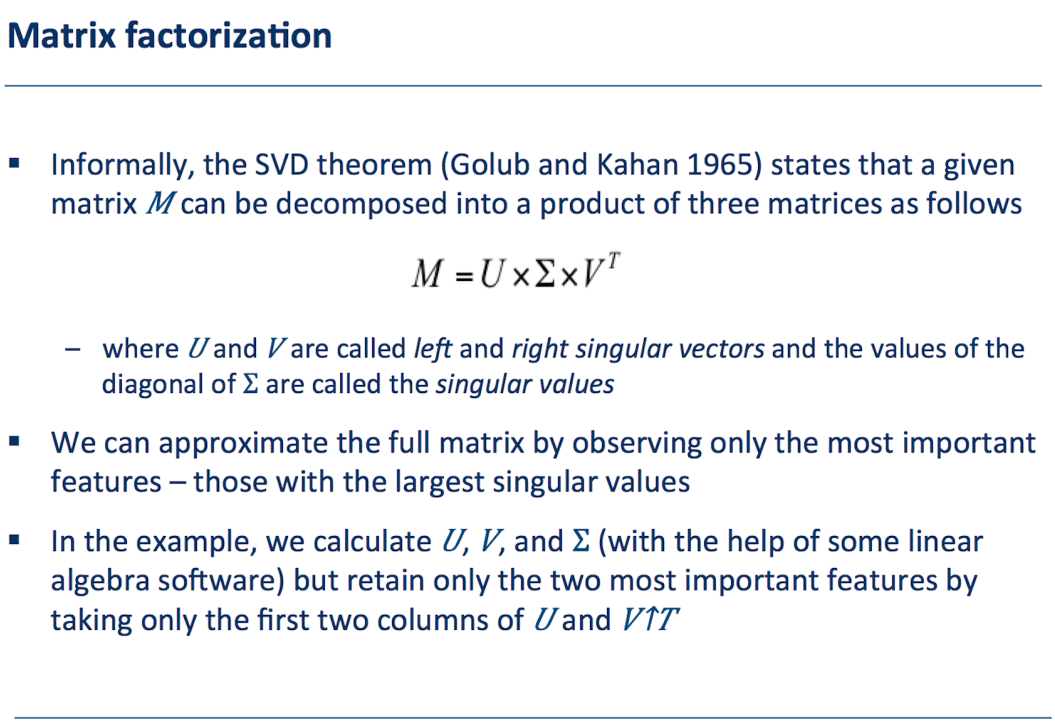


Figure 1: SVD theorem

The application is that instead of using all the singular values of S, use only the most significant r. Compute a rank-r approximation A' to A such that A' = U'S'V'^(T) where U' is m x r, S' is r x r, and V' is m x r. This approximation minimizes the Frobenius form: ||A-A'||F = sqrt(∑(aij-a'ij)^2). Given a matrix of ratings R, we want to compute an approximate matrix Rapp such that RMSE is minimized. (RMSE = ||R-Rapp||F). Then, Rapp can be regarded as the predicted rating matrix.

b). Item-Based Collaborative Filtering (IBCF)

The second algorithm is Item-based collaborative filtering recommendation. Two main steps are needed in this Item-based collaborative filtering recommendation. The first one is item similarity computation, and another

is prediction computation. In this method, cosine-based similarity and weighted sum would be used.

c). User-Based Collaborative Filtering (UBCF)

The third method we used is **User-based Collaborative Filtering. In this algorithm, the vital task is to calculate the correlation below:**

Pearson Correlation

$$corr_{ij} = \frac{\sum_{k=1}^{l}(r_{ik} - \overline{r_i})\ (r_{jk} - \overline{r_j})}{\sqrt{\sum_{k=1}^{l}(r_{ik} - \overline{r_i})^2\ \sum_{k=1}^{l}(r_{jk} - \overline{r_j})^2}}$$

More importantly, we also implement a regularization parameter β to optimize the result. It is because using the parameter of β, the problem of overfit will be solved. The detailed designs of this method will be discussed in Chapter 5.

d). Combination of User-Based Collaborative Filtering and Support Vector Machine

As for the algorithm of support vector machine, it is that given a set of training examples, each marked for belonging to one of two categories, an SVM training algorithm builds a model that assigns new examples into one category or the other, making it a non-probabilistic binary linear classifier. What we use in this project is a branch of SVM, which is called support vector regression. In the bonus method, we will use both support vector machine and user-based collaborative filtering recommendation to predict the movie ratings. The detailed designs of this method will also be discussed in Chapter 5.

4.2.2 Language Used

In this project, the language of Java would be used to develop the movie rating prediction systems.

4.2.3 Tools Used

On the purpose of better test the performance of different algorithms, some machine learning tools such as Libsvm and scikit-learn will be used when implementing some algorithm methods.

4.3 How to generate output

We would like to process the training data to make it acceptable by different algorithm methods. Run these programs on the processed data to get the model. Then we would run the model on test data to generate the output. Finally, we would compute RMSE from the prediction output.

4.4 How to test against hypothesis

Our goal is to provide more experimental samples to the field of movie rating algorithms, so that we can make a little contribution to the academia. On this purpose, we would not only successfully implement at least three algorithm methods, but also use RMSE to measure the accuracy of prediction.

4.5 How to proof correctness (required by dissertation only)

After implementing the algorithm methods, we would check RMSE (Root Mean Square Error) to measure the accuracy of prediction. Bigger RMSE means bad accuracy, and smaller RMSE means better accuracy. In this way we can well proof the correctness.

5. Implementation

5.1 Code (refer programming requirements)

All the code is written in the submit file where three algorithms are successfully implemented.

5.2 Design Document and Flowchart

In order to better evaluate the performance of our methods, we use User average as the baseline. Then we implemented three methods and one more combination algorithm.

5.2.1 SVD

Incremental SVD is a gradient descent algorithm that takes the derivative of the approximation error for known data to overcome the missing value problem, while traditional SVD does not work for sparse matrices.

The algorithm can be described in following pseudo code:

1. Training features one by one, from most significant one to least significant one.
2. In order to train each feature, iterate continuously until converge, in each iteration
    a. Goes through all training samples
        i. Compute error based on predict rating and actual rating
        ii. Update user/movie feature vector accordingly

Algorithm 1 Incremental SVD

procedure TRAINING($trainingData$)
  for each feature f **do**
    $rmse = 0$
    while not enough $round$ And $rmse$ did improve **do**
      $sq = 0$
      for each (u, m, r) in $trainingData$ **do**
        $pRating \leftarrow PREDICT(u, m, r)$
        $error \leftarrow r - pRating$
        $sq \leftarrow sq + error * error$
        $uFeat_{f,u} \leftarrow uFeat_{f,u} + \alpha * (error * mFeat_{f,m} - \lambda * uf)$
        $mFeat_{f,m} \leftarrow mFeat_{f,m} + \alpha * (error * uFeat_{f,u} - \lambda * mf)$
      end for
      $rmse \leftarrow \sqrt{\frac{sq}{trainingData.size}}$
      $round \leftarrow round + 1$
    end while
  end for
end procedure

Despite the algorithm is easy to implement, there are several implementation details:

- How to determine whether converge or not
  - We use two metrics to determine convergence, one is the training round; and the other is difference of RMSE between two rounds.
- Cache the computation result of previou trained features for training new feature to improve efficiency.
- Rating normalization, we normalize the rating to be the range of [1,5].

5.2.2 Item-based Collaborative Filtering

a) Generate a user-item matrix to see every user's rating to every movie.
b) Get an array of every user's average rating in order to eliminate the influence of user's rating habit. (Some users like to give high rating, some users like to give low rating)
c) Compute the similarity of different items with the formula

$$sim(i,j) = \frac{\sum_{u \in U}(R_{u,i} - \bar{R_u})(R_{u,j} - \bar{R_u})}{\sqrt{\sum_{u \in U}(R_{u,i} - R_u)^2}\sqrt{\sum_{u \in U}(R_{u,j} - R_u)^2}}$$

d) Predict the ratings of testing datasets $P_{u,i} = \frac{\sum_{\text{all similar items, N}}(s_{i,N} * R_{u,N})}{\sum_{\text{all similar items, N}}(|s_{i,N}|)}$

5.2.3 User-Based Collaborative Filtering (UBCF)

The third method we used is User-based Collaborative Filtering. In this algorithm, the vital task is to calculate the correlation below:

Pearson Correlation

$$corr_{ij} = \frac{\sum_{k=1}^{l}(r_{ik} - \overline{r_i})\ (r_{jk} - \overline{r_j})}{\sqrt{\sum_{k=1}^{l}(r_{ik} - \overline{r_i})^2\ \sum_{k=1}^{l}(r_{jk} - \overline{r_j})^2}}$$

More importantly, we also implement a regularization parameter β to optimize the result.

$$corr_{ij} = \frac{\sum_{k=1}^{l}(r_{ik} - \overline{r_i})\ (r_{jk} - \overline{r_j})}{\sqrt{\sum_{k=1}^{l}(r_{ik} - \overline{r_i})^2\ \sum_{k=1}^{l}(r_{jk} - \overline{r_j})^2}} + \beta$$

η is imported because due to the small size of the dataset, when a small number of users rate the movies, the correlation may overfit without β. When the parameter of η is added, the problem of overfit will be solved, and those of large numbers will almost not be influenced.

5.2.4 Combination of User-Based Collaborative Filtering and Support Vector Machine (UBCF and SVM)

The forth method we implemented is a combination of User-based Collaborative Filtering and Support Vector Machine. There are two main

procedures in this algorithm. The procedures are as follows.

1st Step: User-based Collaborative Filtering

Firstly, we need to implement User-based Collaborative Filtering so that we can represent the features of the relationships between users and movies. This method has been successfully implemented.

2nd Step: Support Vector Machine

a) **Data Preprocessing**

Use the User-based Collaborative filtering method in the to predict the ratings for training data. Consider user and movie meta information such as user_age, user_gender, user_occupation, movie_genre. We combine all these information and generate the processed training data and test data in below format:

<rating> 1:<u_p_score> 2:<u_age> 3:<u_gender> 4:<u_occupation> 5:<is_unknown_genre> 6:<is_action_genre> 7:<is_adventure_genre> 8:<is_animation_genre> 9:<is_children's_genre> 10:<is_comedy_genre> 11:<is_crime_genre> 12:<is_documentary_genre> 13:<is_drama_genre> 14:<is_fantasy_genre> 15:<is_film-Noir_genre> 16:<is_Horror_genre> 17:<is_ Musical_genre> 18:<is_Mystery_genre> 19:<is_romance _genre> 20:<is_sci-fi_genre> 21:<is_thriller_genre> 22:<is_war_genre> 23:<is_western_genre>

From feature 5 to the feature 23, the output is Boolean value, which is represented by 0 or1, where 0 represents the movie is not in a genre, while 1 means movie is in that genre.

b) **Train SVR**

Based on this new dataset, we use libsvm to train a SVR model

c) **Predict**

Use the SVR model to predict the result.

In this way, we successfully predict the movie ratings.

6. Data Analysis And Discussion

6.1 Output Generation

a). The outcome of baseline is as follows:

Baseline:

Dataset	RMSD(Float)	Base Case RMSD(Float)
u1	1.0957456	1.1036304
u2	1.0643291	1.0872902
u3	1.0467231	1.0728933
u4	1.0617895	1.0753604
u5	0.914376	1.0797685
ua	1.0812217	1.0797962
ub	1.0863218	1.0990211
average	1.0500724	1.0853943

b). Singular Value Decomposition (SVD)

MovieLens 100k data has 100k ratings from 943 users on 1682 movies. It provides 7 datasets, which divided the dataset into training set and test set. u1-u5 data sets split the data in 80/20 fashion to generate training and test data.While ua and ub split the test data with user exactly 10 ratings in it.

We studied the effect of five parameters in the algorithm, namely regularization parameter K, learning rate alpha, number of features, initial value for feature vector and round. The output can be summarized in following tables.

Table 6.1

regK	0		0.001		0.015		0.02		0.05		0.1	
	training	test	training	test	training	test	training	test	training	test	training	test
u1	0.3037	1.1351	0.3037	1.1331	0.3284	1.0730	0.4245	1.0125	0.5825	0.9514	0.6877	0.9352
u2	0.3033	1.1358	0.3047	1.1235	0.3270	1.0727	0.4260	1.0092	0.5778	0.9447	0.6860	0.9260
u3	0.3041	1.1245	0.3075	1.1171	0.3282	1.0639	0.4258	1.0032	0.5778	0.9341	0.6873	0.9182
u4	0.3042	1.1198	0.3064	1.1136	0.3278	1.0674	0.4260	1.0004	0.5818	0.9399	0.6895	0.9239
u5	0.3055	1.1297	0.3065	1.1165	0.3289	1.0709	0.4242	1.0066	0.5759	0.9420	0.6865	0.9248
ua	0.3404	1.1251	0.3412	1.1233	0.3619	1.0815	0.4361	1.0068	0.5796	0.9573	0.7098	0.9454
ub	0.3397	1.1477	0.3405	1.1473	0.3604	1.1058	0.4342	1.0284	0.5808	0.9782	0.7100	0.9642

Table 6.2

learningRate	0.0005		0.001		0.0015		0.002	
	training	test	training	test	training	test	training	test
u1	0.9143	0.9618	0.7856	0.9316	0.5905	0.9618	0.4695	0.9988
u2	0.9158	0.9524	0.7870	0.9224	0.5936	0.9537	0.4712	0.9982
u3	0.9165	0.9471	0.7901	0.9169	0.5922	0.9504	0.4708	0.9917
u4	0.9157	0.9475	0.7849	0.9207	0.5934	0.9490	0.4716	0.9913
u5	0.9165	0.9481	0.7821	0.9188	0.5883	0.9474	0.4688	0.9875
ua	0.9045	0.9680	0.7511	0.9412	0.5665	0.9603	0.4652	0.9948
ub	0.9031	0.9841	0.7516	0.9587	0.5637	0.9800	0.4665	1.0167

Table 6.3

#feature	10		20		30		40		50	
	training	test	training	test	training	test	training	test	training	test
u1	0.8239	0.9357	0.8022	0.9332	0.7935	0.9320	0.7892	0.9313	0.7875	0.9308
u2	0.8263	0.9290	0.8027	0.9247	0.7940	0.9228	0.7898	0.9217	0.7887	0.9212
u3	0.8254	0.9224	0.8013	0.9184	0.7968	0.9177	0.7935	0.9168	0.7914	0.9165
u4	0.8200	0.9257	0.7973	0.9227	0.7906	0.9211	0.7872	0.9203	0.7851	0.9197
u5	0.8187	0.9262	0.7961	0.9232	0.7891	0.9219	0.7849	0.9213	0.7824	0.9201
ua	0.8067	0.9445	0.7778	0.9418	0.7657	0.9422	0.7574	0.9410	0.7541	0.9406
ub	0.8105	0.9632	0.7802	0.9608	0.7656	0.9592	0.7569	0.9582	0.7525	0.9582

Table 6.4

initiValue	0.001	0.01	0.05	0.1	0.2

	training	test	training	test	training	test	training	test	training	test
u1	0.9156	0.9628	0.9154	0.9626	0.8664	0.9422	0.7856	0.9316	1.1365	1.2741
u2	0.9173	0.9537	0.9171	0.9534	0.8682	0.9344	0.7870	0.9224	1.1282	1.2752
u3	0.9184	0.9498	0.9182	0.9495	0.8604	0.9276	0.7901	0.9169	1.1311	1.2969
u4	0.9184	0.9494	0.9182	0.9492	0.8573	0.9294	0.7849	0.9207	1.1326	1.2893
u5	0.9191	0.9513	0.9189	0.9510	0.8568	0.9309	0.7821	0.9188	1.1358	1.2583
ua	0.9173	0.9694	0.9172	0.9693	0.8415	0.9484	0.7511	0.9412	1.1665	1.2367
ub	0.9157	0.9882	0.9156	0.9880	0.8400	0.9658	0.7516	0.9587	1.1636	1.2405

Table 6.5

	100		200		300		400		500	
Round	training	test	training	test	training	test	training	test	training	test
u1	0.9155	0.9623	0.7856	0.9316	0.5875	0.9579	0.4679	1.0013	0.4060	1.0294
u2	0.9168	0.9530	0.7870	0.9224	0.5904	0.9549	0.4711	0.9973	0.4061	1.0242
u3	0.9178	0.9478	0.7901	0.9169	0.5928	0.9471	0.4710	0.9852	0.4067	1.0156
u4	0.9174	0.9481	0.7849	0.9207	0.5946	0.9497	0.4706	0.9892	0.4056	1.0201
u5	0.9180	0.9487	0.7821	0.9188	0.5891	0.9499	0.4683	0.9915	0.4044	1.0211
ua	0.9058	0.9683	0.7511	0.9412	0.5647	0.9618	0.4643	0.9988	0.4182	1.0249
ub	0.9046	0.9845	0.7516	0.9587	0.5637	0.9798	0.4653	1.0135	0.4180	1.0429
avg	0.9137	0.9590	0.7760	0.9300	0.5833	0.9573	0.4684	0.9967	0.4093	1.0254

c). Item-Based Collaborative Filtering

The output is as follows.

Table6.6

Dataset	Accuracy	RMSD	Accuracy(Float)	RMSD(Float)
u1	0.82	1.22448877	0.65025	1.0951171
u2	0.83215	1.2140634	0.6512	1.0616156
u3	0.84355	1.1927909	0.6592	1.0450088
u4	0.8308	1.2225384	0.6516	1.0591254
u5	0.6102	1.0359054	0.47905	0.91180253
ub	0.8120891	1.3063188	0.6465536	1.0863218

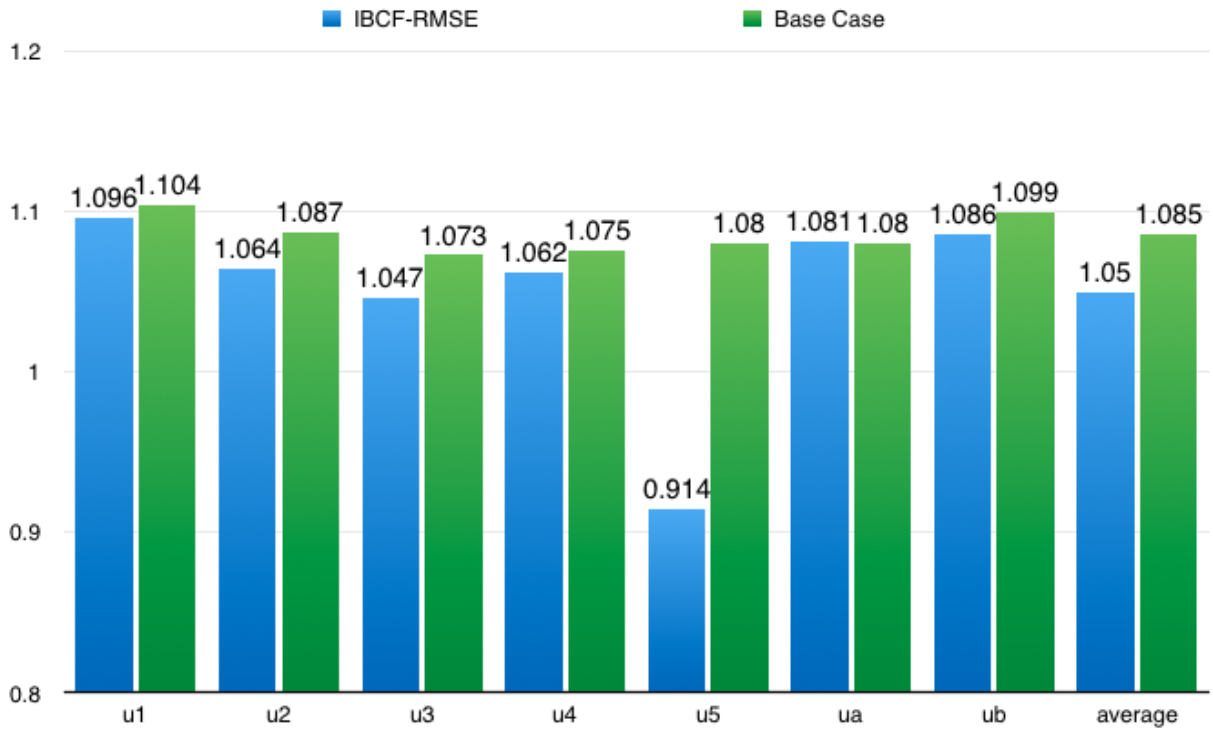


Diagram 6.1

d). User-based Collaborative Filtering

The following is the table of RMSE which is changed by the parameter β .

Table 6.7

BETA	0	0.01	0.02	0.03	0.04	0.05
u1	0.996193	0.992321	0.993705	0.996845	0.99905	1.000975
u2	0.988079	0.985901	0.985571	0.986636	0.988205	0.990353
u3	0.981402	0.982369	0.983946	0.984657	0.987649	0.989747
u4	0.980612	0.980867	0.98023	0.981147	0.983997	0.985951
u5	0.982242	0.979388	0.97788	0.977113	0.978698	0.980331
ua	1.002224	0.996335	0.99931	1.001431	1.002859	1.004074
ub	1.019273	1.017034	1.015312	1.018961	1.020365	1.02296
Average	0.992860714	0.990602143	0.990850571	0.992398571	0.994403286	0.996341571

e). Combination of User-Based Collaborative Filtering and Support Vector Machine (Combination of UBCF and SVM)

Too much time is spent on training the data. However, results of three datasets have been generated by SVM machine.

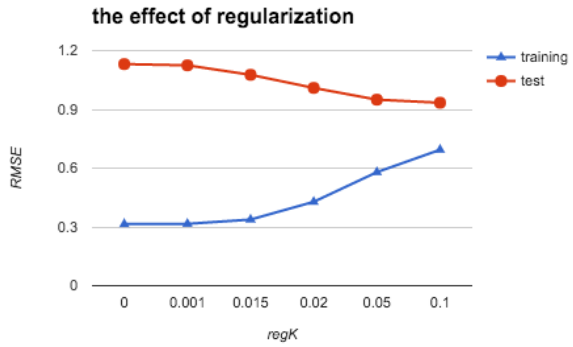
Table 6.8

Dataset	SVM
u1	0.998323595
u2	0.988281
u3	0.984733

6.2 Output Analysis

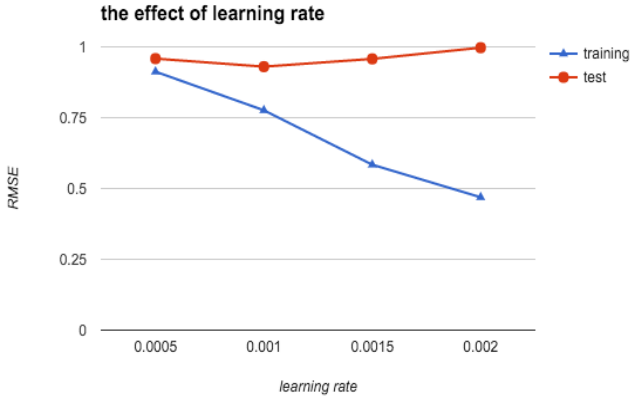
a). Singular Value Decomposition (SVD)

In order to better visualize the effect of those parameters in the SVD algorithm, we take the average of RMSE of 7 datasets and draw following diagrams.



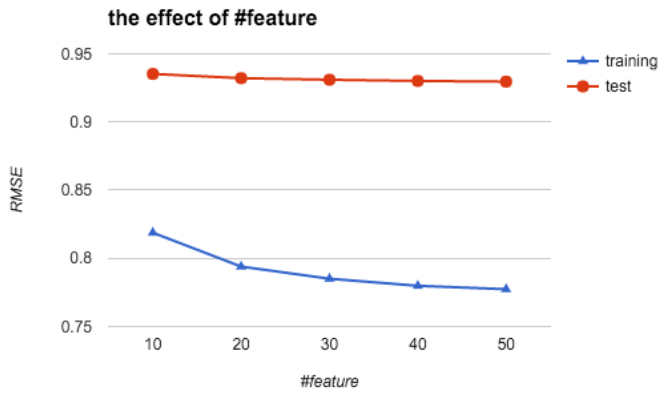
Graph 6.1

From above diagram we can see that without regularization, the overfitting problem is very serious. As increasing the regularization parameter k, the RMSE of training data is also increasing, while RMSE of test data is decreasing.



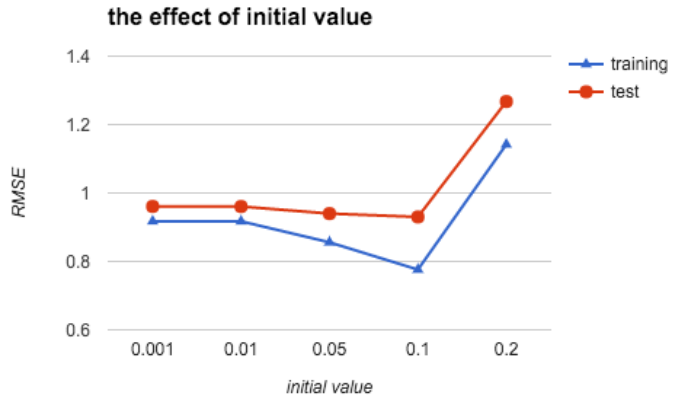
Graph 6.2

From above diagram, we can see that as learning rate increase, the RMSE of training data decrease, however, when learning rate is larger than 0.0015, the RMSE of test data increases as well, thus it is not the larger learning rate result in better performance. The other thing is when learning rate is too small, say 0.0005, the converge speed is very slow, which is expected.



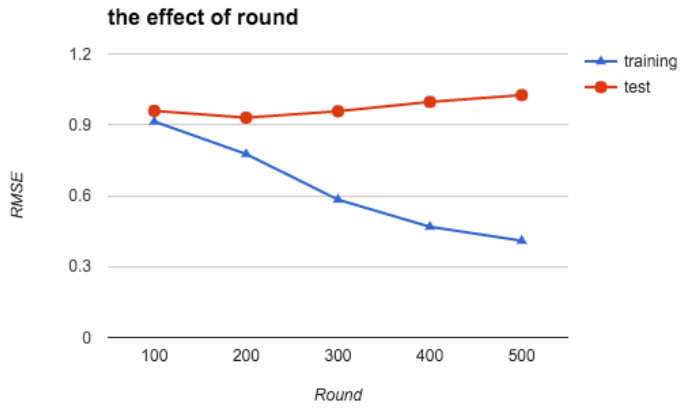
Graph 6.3

The above diagram shows that the number of features does not impact the accuracy significantly. When number of features is greater than 40, the RMSE of test data keeps almost flat.



Graph 6.4

In our test, proper chosen initial value will yield good prediction performance in terms of RMSE. However, when initial value is very large (greater than 0.2 under our other parameter chosen), RMSE for both training set and test set has increased, it may due to the training did not converge.



Graph 6.5

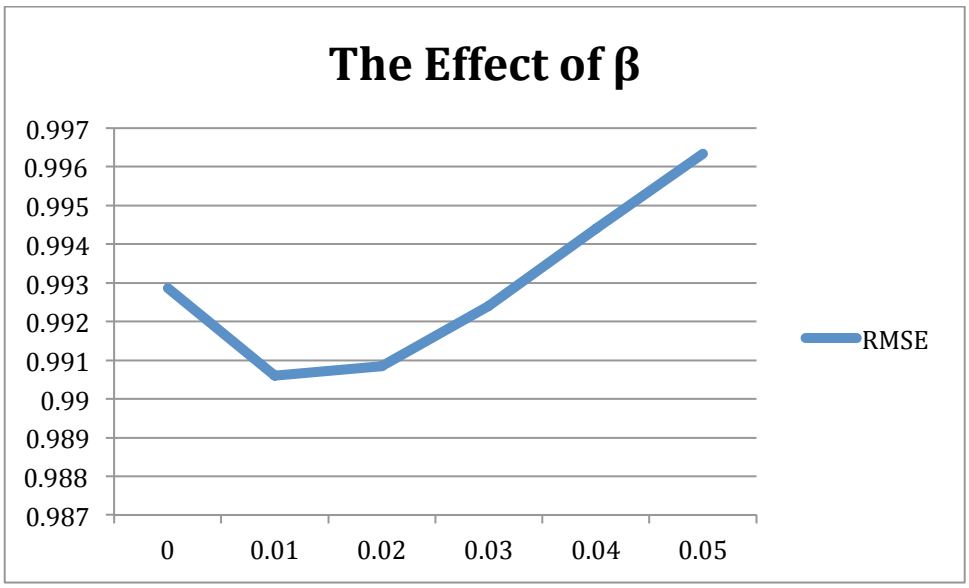
In our test, we also observed that how to determine convergence will have effect of prediction accuracy. Besides considering converge when RMSE did not improve, we also limit the number of round of training, in our 200 round of training for each feature achieves good performance, while increasing the round number will encounter overfit problem.

b). Item-Based Collaborative Filtering (IBCF)

There is no parameter in this algorithm. The output is shown in 6.1.

c). User-Based Collaborative Filtering (UBCF)

As can be seen from the above output, the RMSE of User-Based Collaborative Filtering (UBCF) algorithm is 0.9906, which is smaller than the baseline. the value of parameter β is also changed to check the changes of RMSE.



Graph 6.6

From this chart it can be seen, in this dataset, the method has better performance when the value of β is around 0.01.

d). Combination of User-Based Collaborative Filtering and Support Vector Machine (Combination of UBCF and SVM)

After finishing all of the three algorithms, we also tried to implement the fourth algorithm, the combination of User-Based Collaborative Filtering (UBCF) and Support Vector Machine (SVM). We once expect the results of this method would be the best. However, in reality, the results are not as good as what we expected. We guess the reason is that we use the default parameters of libsvm, which make the results not so good. In the future, we may adjust the parameters below in order to generate better results.

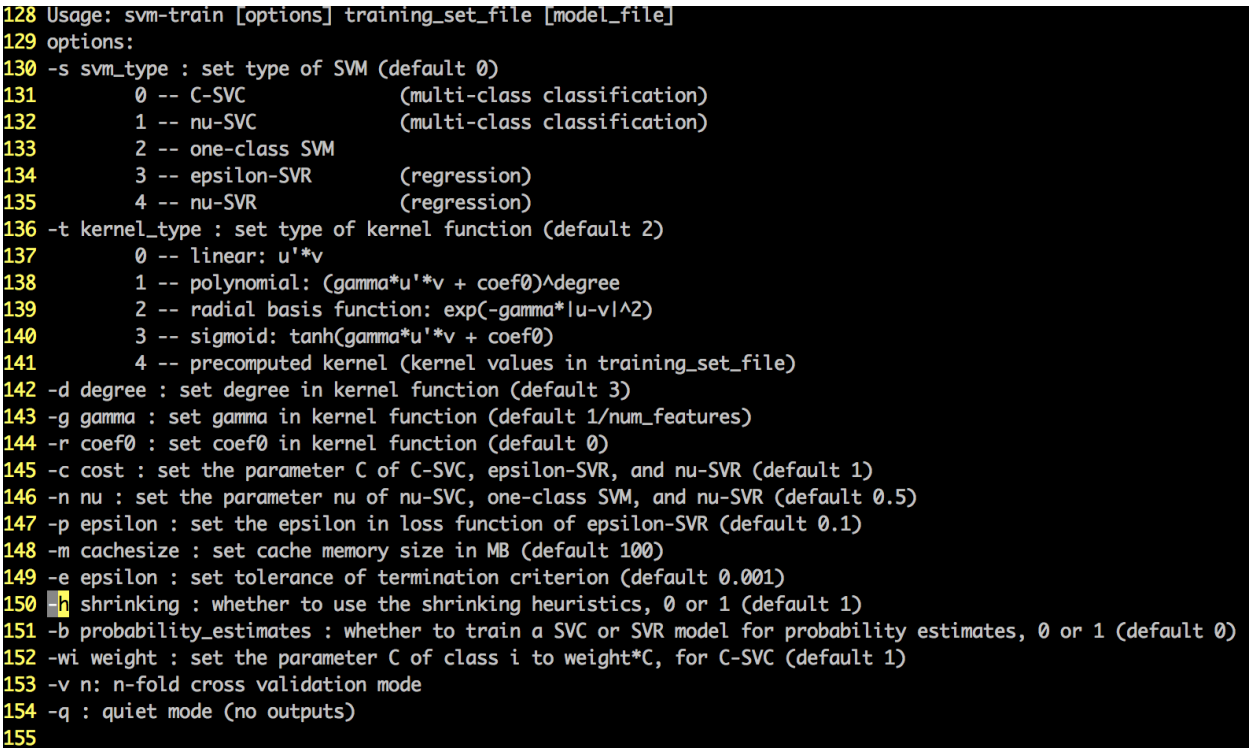


Image 6.1

6.4 Abnormal Case Explanation (the most important task)

The abnormal case is obviously u5. Movielens' datasets are divided into training datasets and testing datasets, the u5 dataset is the only one which includes users and movies that never appear in training datasets. Movielens uses these corner cases to test systems' ability of dealing with users and movies from nowhere.

7. Conclusions And Recommendations

7.1 Summary and Conclusions

Incremental SVD can achieve RMSE 0.93 for MovieLens 100k data set with proper chosen initial value and learning rate. It does not need much features, our study shows 40-50 features is enough. Overfitting is the

biggest problem, adding regularization and limiting the number of training helps. For the User-based Collaborative Filtering (IBCF), it also better performed than the baseline, which we can see in the diagram in Chapter 6. User-based Collaborative Filtering (UBCF) can achieve an average RMSE of 0.9906. It calculates the correlation between users and measure how close they are. Adding a regularization parameter will improve the accuracy to some extent, and we also discussed the different values this parameter to choose the right regularization parameter.

In summary, we implemented three collaborative filtering techniques and research on a combination algorithm; they are Incremental SVD, Item-based Collaborative Filtering (IBCF), User-based Collaborative Filtering (UBCF), and User-Based Collaborative Filtering (UBCF). The result shows that comparing with the base algorithm, all the three methods we successfully implemented can better describe the data.

7.2 Recommendations for Future Studies

The purpose of rating movies is to recommend movies to the users. Normally we use a threshold such as 3.5 to recommend movies to the users. However, it will still cause some false negatives and false positives. Therefore, although these false predictions cannot be avoided, we still can optimize the methods to predict the users' ratings closer to the actual ratings.

In the future work, several aspects can be improved. As for IBCF, for some special corner cases like predicting of the user who never appear in training datasets or movie that never exists in training datasets, IBCF could use some special technology to give a rough prediction only based on the similarity of different movies. When recommending the movies to the users, the reference information of some of the hot movies is not so useful,

because the users may rate those movies according to the Media's preference instead of themselves. Therefore, in the future work, without considering the hot movies, we can test the program to see if the result can better predict the users actual ratings.

8. Bibliography

1. Joseph Konstan Badrul Sarwar, George Karypis and John Riedl. Incremental singular value decomposition algorithms for highly scalable recommender systems. In Proceedings of the 5th International Conference on Computer and Information Technology (ICCIT), 2002.

2. D. Lemire and A. Maclachlan. Slope One predictors for online rating-based collaborative filtering. In Proceedings of the SIAM International Conference on Data Mining (SDM'05), 2005.

3. Joseph Konstan Badrul Sarwar, George Karypis and John Reidl. Item-based collaborative filtering recommendation algorithms. In WWW '01: Proceedings of the 10th international conference on World Wide Web, pages 285–295, New York, NY, USA, 2001. ACM Press.

4. Thomas Hofmann. Latent semantic models for collaborative filtering. ACM Transactions on Information Systems, 22(1):89–115, 2004.

5. Huang, Zan, Hsinchun Chen, and Daniel Zeng. "Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering." ACM Transactions on Information Systems (TOIS) 22.1 (2004): 116-142.

6. Koren, Yehuda, Robert Bell, and Chris Volinsky. "Matrix factorization techniques for recommender systems." Computer 8 (2009): 30-37.

7. Adomavicius, Gediminas, and Alexander Tuzhilin. "Toward the next

generation of recommender systems: A survey of the state-of-the-art and possible extensions." Knowledge and Data Engineering, IEEE Transactions on 17.6 (2005): 734-749.

9. Appendices

9.1 Sample code

Since the code is too long, so we will not show here. Please see the code from Submit.