CS-504 - Final Project – Option 2 – Coding

(Due Dec 12th, 2016 8:00 AM)

Student: Rongxin Zheng
Instructor: Kevin Molloy

Date: Dec 7th, 2016

(My answer starts in page 5.)

In this project, you will explore the MovieLens datasets and utilize analytical software to develop models to analyze the data. This project should be programmed in Python and can optionally use SciKit-Learn, however, if you feel more comfortable using MATLAB and/or WEKA, **you MUST RECEIVE permission from me before November 24th**.

The goal of this work will be to utilize 2 techniques: collaborating filtering (CF) and k-nearest neighbors (KNN) to build a recommender system for movies using the MovieLens dataset ml-latest-small (http://grouplens.org/datasets/movielens/) that contains 100,000 ratings for 9,000 different movies. The project will involve perform some minor data manipulation to format the data for each technique (constructing the ratings matrix R for example). Two techniques will be employed to predict movie ratings:

Problem

1. Use k-nearest-neighbors (KNN) to predict movie ratings. The KNN problem in this setting is:

   For a given user u, find it k nearest neighbors (that have also rated movie m) for some distance metric. Your predicted rating will be the weighted average of its nearest neighbor ratings:

   $$P_{u,i} = \frac{\sum_{k=1}^{K} r_{k,i}}{K}$$

   Such that $r_{u,i}$ is the rating for user u on movie i. K represents the number of neighbors used (all of which have rated movie m). You should evaluate at least 3 different values for k. For the final SSE plot requested, you k = 3. Show your predictions for ALL entries in the ratings_test_10.csv file using the file format discussed in the collaborate filtering section (which is next).

2. Use the collaborative filtering (CF) item-item recommendation system discussed in class to predict movie rating. The similarity between two items (i , j) is defined as follows:

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

To calculate a prediction, use the weighted sum technique discussed in the slides:

$$P_{u,i} = \frac{\sum_{n \in N}(s_{i,n} * R_{u,n})}{\sum_{n \in N} s_{i,n}}$$

For each training/test set provided (80, 85, 90, and 95 percent), show the results/predictions for EVERY test case for each pair of training/test sets. Name the result files: ratings_predictions_20.csv (formed from evaluated the ratings_test_20.csv file after training with the ratings_train_80.csv file), ratings_predictions_15.csv (formed from evaluating ratings_test_15.csv after training with ratings_train_85.csv), ratings_predictions_10.csv (formed from evaluating ratings_test_10.csv after training with ratings_train_90.csv), and finally ratings_predictions_05.csv (formed from evaluating ratings_test_05.csv after training with ratings_train_95.csv). The ratings_predictions_??.csv file should be formatted as follows (with each field separated by a comma): userID, movieID, realRating, predictedRating. The order should match the ratings_test_??.csv file to allow for easier result validation.

The test/training sets are in TAR ball called CS504FinalProject_ratings_test.tgz and is available on the class website.

You are to work on this project **alone and not in groups**. All projects will be checked for plagiarism against each other and against the Internet using software. Suspicious projects will be sent to the honor committee for further review.

Deliverable:

Turn in a tarball (or zip file) with the following information:

1. For each method (KNN and collaborate filtering) submit all source code. Organize the source code used for each step in its own directory (cs504final/step1, cs504final/step2, etc).
2. A professionally organized single PDF document. **Any document not in PDF format will automatically lose 20% of the potential grade**. This document should include:
    a. Brief introduction to the problem
    b. A METHODS section that outlines the technique and any observations noted for the KNN and CF methods. Each method should be discussed in a separate paragraph.

   c. A result section, with a paragraph for KNN and CF. You should show the accuracy of the method and compare how changing the values in your model (k for KNN and the training/testing set sizes for CF) impacted your results. Again, I expect explanations in words accompanied by PLOTS of your results (you need BOTH plots and words).
   d. For each training test set, show a plot and a table of the SSE. Comment on how the different amounts of training/test data impact the results. For KNN, use the ratings_test_10.csv file and plot the SSE.
   e. A small conclusion paragraph with any final thoughts on your findings.
3. Include all 5 files of predictions as discussed. These files should be named:
   a. ratings_predictions_knn_k_3_10.csv
   b. ratings_predictions_cf_10.csv
   c. ratings_predictions_cf_15.csv
   d. ratings_predictions_cf_20.csv

And files should be formatted as requested with ALL PREDICTIONS ROUNDED TO ONE DECIMAL POINT.

Examples of KNN:

Using the simple distance function from above, here is a small table of results from my program:

UserID	MovieID	K	Real Rating	Predicted Rating	UserID for nearest neighbors and distance (only for k=3)
1	31	3	2.5	3.33	325 (13.67) 310 (16.30) 485(18.91)
		5		3.5	
		10		3.45	
614	1370	3	1.0	2.6	210 (41.35) 204 (41.96) 53(43.92)
		5		2.7	
		10		2.8	
608	247	3	4.0	4.33	647(69.58),35 (69.78), 403(69.83)
		5		3.8	
		10		3.7	

SSE calculation for N test case:

$$SSE = \sum_{i=1}^{N} (R_i - P_i)^2$$

Where R_i is the real value for test case i and P_i is the predicted value for test case i.

RMSE (if you want to use it instead of SSE):

$$RMSE = \frac{\sqrt{\sum_{i=1}^{N} (R_i - P_i)^2}}{N}$$

Answer:

For question 1 and 2, I submit all the output files.

For Q1, I submit six output files. (K=3, K=5, K=10)
Using v2 file as testing data:
ratings_predictions_knn_k_3_10.csv
ratings_predictions_knn_k_3_10.csv
ratings_predictions_knn_k_3_10.csv
Using v3 file as testing data:
ratings_predictions_knn_k_3_10_V3.csv
ratings_predictions_knn_k_3_10_V3.csv
ratings_predictions_knn_k_3_10_V3.csv

For Q2, I submit eight output files. (05, 10, 15, 20)
Using v2 file as testing data:
ratings_predictions_cf_05.csv
ratings_predictions_cf_10.csv
ratings_predictions_cf_15.csv
ratings_predictions_cf_20.csv
Using v3 file as testing data:
ratings_predictions_cf_05_V3.csv
ratings_predictions_cf_10_V3.csv
ratings_predictions_cf_15_V3.csv
ratings_predictions_cf_20_V3.csv

Q1.

1. Pruning method used;
2. Improvement of this question and how I debug it;
3. Code Part;
4. Verify my code, to compare answers with samples; (All the same)
   (1. The first 10 output you gave through email; 2. The sample answers you gave in the paper, which using the whole data set as training data.)
5. Calculations of SSE and RMSE, and conclusion; (including bar plots)

The steps to finish this question are as follows: (all the steps are kept the same as the annotations in my code.)

1. Pruning method used:

In my model, to prune the complex, I didn't build the whole distance matrix. I only calculated the cell which we are going to use, which will greatly reduce the running time. My program can get the output in only 5 to 10 min, which is much shorter than the time to build up the distance matrix.

Pruning Code:

```
for x in range(n_users):
    if train_data_matrix[x][movie]!=0 :
        if (x!=user) and (D_matrix[user][x]==0) :
            D_matrix[user][x]=distance(train_data_matrix[user],train_data_matrix[x],n_movies)
            D_matrix[x][user]=D_matrix[user][x]
dist = D_matrix[user]
```

Besides, my code, no matter Q1 or Q2, will conduct a small section to test the model, comparing the answer you gave us and my own answers.

2. Improvement of this question and how I debug it:

There is one problem. I found that for the item whose movieID is 1484 there are only three people who have rated it, whose userID are 15, 451 and 546. You know, 3 is not larger than k. Therefore, for example, when we calculated for the record of (movieID = "1484", userID = "15"), we can only get two nearest neighbors. In this case, I still set the denominator be length(KNN). (for this case, length(KNN)=2)

$$P_{u,i} = \frac{\sum_{k=1}^{length(KNN)} r_{k,i}}{length(KNN)}$$

KNN is the user set, in which all the users also rated movie i.
(For here, length(KNN) < K)

3. Code Part:

3.0. Import all the module I will use in this project: (Line 1 – Line 6)

```
# 0. import modula
import numpy as np
import pandas as pd
import math
import pdb
import csv
```

"numpy", "pandas", "csv" and "math" are for the basic function;
"pdb" is used to debug my code;

3.1. Prepare the data:

For this problem, I will use the data which you give us on the class website. (for output file of K=3, I used the 90_v2 file, as you said.)
(Training set of data:
ratings.csv
-- downloaded from http://grouplens.org/datasets/movielens/
Testing set of data:
ratings_test_10_v2.csv
)

3.2. Load the data: (Line 8 – Line 17)

```
# 2. Load the data
header = ['user_id', 'movie_id', 'rating', 'timestamp']
df = pd.read_csv('ratings.csv', sep=',', names=header)
n_users = df.user_id.unique().shape[0]
n_movies = df.movie_id.unique().shape[0]
print 'Number of users = ' , n_users , ' | Number of movies = ' , n_movies

train_data = df
test_data = pd.read_csv('testing-input.csv', sep=',', names=header)
#test_data = pd.read_csv('ratings_test_10_v2.csv', sep=',', names=header)
```

The last row of code is used to test the answer, and I proved that I am correct, which will be shown in step 4.

3.3. Predicting using KNN:

3.3.1. Build the Hash Table for userID and MovieID: (Line 19 – Line 34)

```
# 3.1. Create the hash table
userid_dict = {}
matrix_idx = 0
movieid_dict = {}
matrix_idy = 0

for line in train_data.itertuples():
    if str(line[1]) not in userid_dict.keys():
        userid_dict[str(line[1])] = matrix_idx
        matrix_idx += 1
    if str(line[2]) not in movieid_dict.keys():
        movieid_dict[str(line[2])] = matrix_idy
        matrix_idy += 1

print "n_users = " + str(n_users) + ' , ' + "matrix_idx = " + str(matrix_idx)
print "n_movies = " + str(n_movies) + ' , ' + "matrix_idy = " + str(matrix_idy)
```

The last two rows are used to verify.
Even when you import the v3 file, the result won't be different, because the order will keep the same number.

3.3.2. Create the rating matrix: (Line 36 – Line 41)

```
# 3.2. Create the matrix for this project
train_data_matrix = np.zeros((n_users, n_movies))
for line in train_data.itertuples():
    matrix_user = userid_dict[str(line[1])]
    matrix_movie = movieid_dict[str(line[2])]
    train_data_matrix[matrix_user][matrix_movie] = line[3]
```

line[0] is Index number.;
line[1]: user id;
line[2]: movie id;
line[3]: ratings;
line[4]: timestamp.

3.3.3. Build the distance function: (Line 43 – Line 48)

```
# 3.3. build the distance function
def distance(user1, user2, length):
    d = 0
    for x in range(length):
        d += pow((user1[x] - user2[x]), 2)
    return math.sqrt(d)
```

"length" is the number of movie, which also is the number of matrix column here.

3.3.4. Get the K nearest neighbors and output the results into CSV file:

(Line 50– Line 123)

```
def GetNeighbors(DD, k):
    Order = sorted(enumerate(DD), key=lambda z:z[1])

    neighbors = []
    x=0
    y=0
    l=len(Order)
    while (y<k) and (x<l):
        if (Order[x][1]!=0) and (train_data_matrix[Order[x][0]][movie]!=0):
                neighbors.append(Order[x])
                y+=1
        x+=1

    return neighbors
```

This section contains much code, so I only paste the function of KNN here. Other code is in Line50 to Line123 in my code.

4. Verify my code, to compare answers with samples:

4.1. The first 10 lines, using 90 percent file as training data set:

1	userID	movieID	realRating	predictedRating
2	1	1371	2.5	2.5
3	1	1953	4	3.7
4	2	253	4	3.3
5	2	319	1	4
6	2	356	3	3.7
7	2	454	4	3.7
8	2	497	3	3.7
9	2	587	3	4
10	3	247	3.5	3.7
11	3	1884	4	3.8

It is totally the same as what you gave us through Email.

4.2. Use the whole data set as training data, and compare with the answer you gave in the paper:

(the table below is conducted by my code. And I will submit another three

csv files with the form you required. K=3, K=5, K=10)

UserID	MovieID	k	Real Ratin	Predicting	Nearest Neighbors
1	31	3	2.5	3.333333	325(13.6656503687) 310(16.3018403869) 485(18.9142803194)
		5		3.5	
		10		3.45	
614	1370	3	1	2.666667	210(41.3007263859) 204(41.9136016109) 53(43.8634243989)
		5		2.7	
		10		2.8	
608	247	3	4	4.333333	647(69.5844810285) 35(69.7782200977) 403(69.8283610004)
		5		3.8	
		10		3.7	

To compare this result with the table in the bottom of Page 3 of this paper, I got all the exactly right answers alone with each column. (including userID and distances)

5. Calculations of SSE and RMSE, and conclusion (including bar plots)

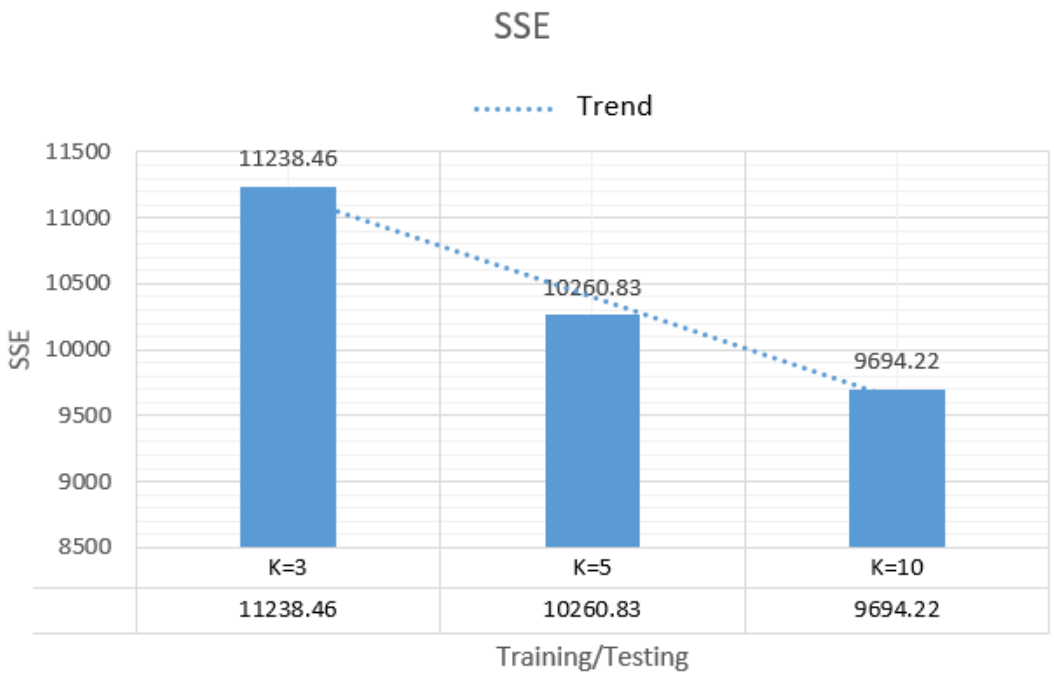
I collected the data of SSE and RMSE, and plot them out as follows:
(I used the whole data set as training data, rather than 90 percent V2 file)
The table of SSE:

KNN	K=3	K=5	K=10
SSE	11238.46	10260.83	9694.22

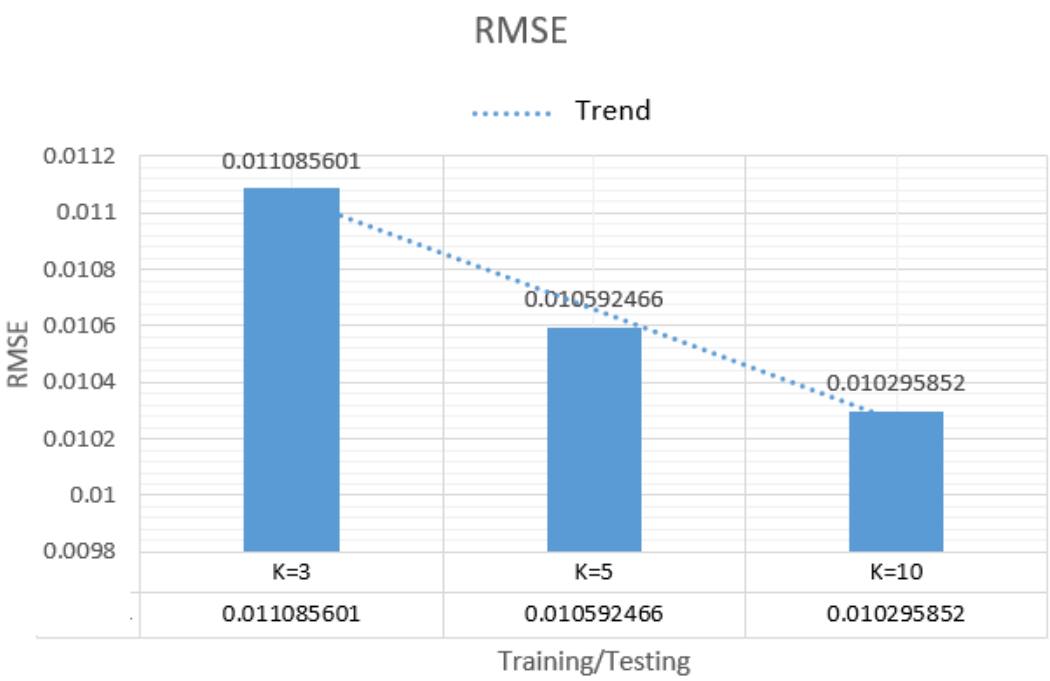
The table of RMSE:

KNN	K=3	K=5	K=10
RMSE	0.011086	0.010592	0.010296

The bar plot of SSE:



The bar plot of RMSE:



Conclusion:

All of these results showed a very good reflection on my prediction. As the number of the nearest neighbors grows, the SSE and RMSE become smaller and smaller. Of course, it should be that because if you use more similar data to predict, the prediction of course will be more accurate.

Q2:

For this problem, I will use the data which you gave us on the class website.
(Training set of data:

ratings_train_80.csv
ratings_train_85.csv
ratings_train_90.csv
ratings_train_95.csv

Testing set of data: (respectively)

ratings_test_20_v2.csv
ratings_test_20_v3.csv
ratings_test_15_v2.csv
ratings_test_15_v3.csv
ratings_test_10_v2.csv
ratings_test_10_v3.csv
ratings_test_05_v2.csv
ratings_test_05_v3.csv

)

I submit eight output files. (05, 10, 15, 20)
Using v2 file as testing data:
ratings_predictions_cf_05.csv
ratings_predictions_cf_10.csv
ratings_predictions_cf_15.csv
ratings_predictions_cf_20.csv
Using v3 file as testing data:
ratings_predictions_cf_05_V3.csv
ratings_predictions_cf_10_V3.csv
ratings_predictions_cf_15_V3.csv
ratings_predictions_cf_20_V3.csv

1. Pruning method used for this question;
2. Improvement of this question and how I debug it;
3. Code Part;
4. Verify my answers; (All the same)
5. Bar plot for SSE and RMSE, and conclude;

To finish this problem, I make it in several steps as follows:

1. Pruning method used for this question:

In my model, to prune the complex, I didn't build the whole similarity matrix. I only calculated the cell which we are going to use, which will greatly reduce the running time. The running time of my program is much shorter than the time to build up the whole similarity matrix.

Pruning Code:

```
    for m in range(n_movies):
        if (m!=movie) and (S_matrix[movie][m] ==0)and (train_data_matrix[user][m] != 0):
            US=[]
            for u in range(n_users):
                if (train_data_matrix[u][movie]!=0) and (train_data_matrix[u][m]!=0):
                    US.append(u)
            if (len(US)!=0):
                S_matrix[movie][m] = sim(m, movie, US)
                S_matrix[m][movie] = S_matrix[movie][m]

    s1 = 0.0
    s2 = 0.0
    for m in range(n_movies):
        if (m!=movie) and (train_data_matrix[user][m] != 0) and (S_matrix[m][movie]>0):
            s1 += S_matrix[movie][m]*train_data_matrix[user][m]
            s2 += S_matrix[movie][m]
```

Note: in order to save the time to set -2 as the initial value, I just set 0 for all of them. Of course, the value of sim(i,j) is in the range of -1 to 1, which could reach 0. However, even it is calculated, when we recalculate, it is still 0. So, it doesn't matter to set 0 as initial value. In the another hand, to do it could save time. If you reset -2 as the initial value for the similarity matrix, your initial part need to run 9000X9000 times. However, using my way, you could calculate

only a few more times for those value is 0.

2. Improvement of this question and how I debug it:

2.1. About Sim matrix:

For example, for movie i and j, there is only one user u who rated both of them, and user_bar[u]=4.0, so there could be three different situations:

a. r[u][i]=4.0, r[u][j]=3.5;
b. r[u][i]=3.5, r[u][j]=4.0;
c. r[u][i]=4.0, r[u][j]=4.0;

You know, in these cases, both of numerator and denominator are equal to 0. I actually found many data, facing this divided-by-0 case. For these case, I set sim[i][j]=1.

Here is a small part of them: (the ill data. The number below is the order rather than the movieID. You can use Hash table to get ID)

```
movie1: 5489
movie2: 126
[280]
movie1: 6479
movie2: 63
[538]
movie1: 2658
movie2: 129
[280]
movie1: 5491
movie2: 129
[280]
movie1: 5844
movie2: 129
[375]
movie1: 2658
movie2: 114
[280]
movie1: 5491
movie2: 114
[280]
```

2.2. About predicted rate:

$$P_{u,i} = \frac{\sum_{n \in N}(s_{i,n} * R_{u,n})}{\sum_{n \in N} s_{i,n}}$$

When N is an empty set.

For (u,i), all the sim[i][j] <= 0, j=0,1, .. , n_movie. In other words, no movie is similar to movie i.(Sim(i,n)<=0, for all n in range(n_movie)) Therefore, in this case, I let P[u][i]=user_bar[u].

For example, for user u, and movie i, r[u][i]=4.0, bar_user[u]=3.5. However, for this movie, all the other users v, r[v][i]-bar_user[v]<=0, which will lead to the

case sum(sim(i,n))=0, which is denominator in the formula of prediction, because userset is empty.

I actually found some movies are in this case as below:

```
s2=0, user: 16 ,movie: 7914
s2=0, user: 28 ,movie: 6326
s2=0, user: 76 ,movie: 7922
s2=0, user: 158 ,movie: 3012
s2=0, user: 168 ,movie: 6395
s2=0, user: 298 ,movie: 7914
s2=0, user: 396 ,movie: 4550
s2=0, user: 421 ,movie: 6639
s2=0, user: 438 ,movie: 6293
s2=0, user: 443 ,movie: 4270
s2=0, user: 481 ,movie: 5263
s2=0, user: 492 ,movie: 6785
```

3. Code Part:

3.0. Import all the formula

```
# 0. import modula
import numpy as np
import pandas as pd
import math
import pdb
import csv
import sys
```

"numpy", "pandas", "csv", "sys" and "math" are for the basic function;
"pdb" is used to debug my code;

3.1. Load the data

```
# 1. Load the data
header = ['user_id', 'movie_id', 'rating', 'timestamp']
train_data = pd.read_csv('ratings_train_80.csv', sep=',', names=header)
test_data = pd.read_csv('ratings_test_20_v2.csv', sep=',', names=header)

n_users = train_data.user_id.unique().shape[0]
n_movies = train_data.movie_id.unique().shape[0]
print 'Number of users = ' , n_users , ' | Number of movies = ' , n_movies
```

3.2. Model

3.2.1. Create the HASH table

```
# 2.1. Create the hash table
userid_dict = {}
matrix_idx = 0
movieid_dict = {}
matrix_idy = 0

for line in train_data.itertuples():
    if str(line[1]) not in userid_dict.keys():
        userid_dict[str(line[1])] = matrix_idx
        matrix_idx += 1
    if str(line[2]) not in movieid_dict.keys():
        movieid_dict[str(line[2])] = matrix_idy
        matrix_idy += 1

print "n_users = " + str(n_users) + ' , ' + "matrix_idx = " + str(matrix_idx)
print "n_movies = " + str(n_movies) + ' , ' + "matrix_idy = " + str(matrix_idy)
```

Even when you import the v3 file, the result won't be different, because the order will keep the same number.

3.2.2. Transfer the data frame into matrix and $\overline{R_u}$

```
# 2.2. Create the matrix for this project and calculate the average
train_data_matrix = np.zeros((n_users, n_movies))
n_user_rate = np.zeros(n_users)
s_user_rate = np.zeros(n_users)
bar_user = np.zeros(n_users)

for line in train_data.itertuples():
    matrix_user = userid_dict[str(line[1])]
    matrix_movie = movieid_dict[str(line[2])]
    train_data_matrix[matrix_user][matrix_movie] = line[3]
    n_user_rate[matrix_user] += 1
    s_user_rate[matrix_user] += line[3]

for i in range(n_users):
    bar_user[i]=s_user_rate[i]/n_user_rate[i]
```

3.2.3. Build the similarity function

```
# 2.3. Build the similarity function
def sim(movie1, movie2, user_set):
    s1 = 0.0
    s2 = 0.0
    s3 = 0.0
    for i in range(len(user_set)):
        diff1=train_data_matrix[user_set[i]][movie1]-bar_user[user_set[i]]
        diff2=train_data_matrix[user_set[i]][movie2]-bar_user[user_set[i]]
        s1 += diff1 * diff2
        s2 += diff1 * diff1
        s3 += diff2 * diff2
    if (s2==0) or (s3==0):
        return 1
        print 'movie1:',movie1
        print 'movie2:',movie2
        print user_set

    return (s1/ ( math.sqrt(s2) * math.sqrt(s3) ) )
```

3.2.4. Get the user set who rated both items

```
            for m in range(n_movies):
                if (m!=movie) and (S_matrix[movie][m] ==0)and (train_data_matrix[user][m] != 0):
                    US=[]
                    for u in range(n_users):
                        if (train_data_matrix[u][movie]!=0) and (train_data_matrix[u][m]!=0):
                            US.append(u)
                    if (len(US)!=0):
                        S_matrix[movie][m] = sim(m, movie, US)
                        S_matrix[m][movie] = S_matrix[movie][m]
```

3.3. Calculate SSE and RMSE

```
        answer.append("%.11f"%(pred))
        SSE += pow((pred - line[3]), 2)
        writer.writerow(answer)

print 'SSE: ',"%.11f"%SSE

n = len(test_data)
RMSE = math.sqrt(SSE)/n
print 'RMSE: ', "%.11f"%RMSE
```

4. Verify my answers:

userID	movieID	realRating	predictedRating
1	1061	3	2.0
1	1129	2	2.5
1	1293	2	2.4
1	2193	2	2.6
2	47	4	3.6
2	208	3	3.3
2	248	3	3.5
2	253	4	3.3
2	272	3	3.4
2	349	4	3.6
2	367	3	3.5
2	377	3	3.6
2	382	3	3.5
2	454	4	3.4
2	474	2	3.5
2	485	3	3.3
2	539	3	3.7
2	550	3	3.5
2	588	3	3.6
2	590	5	3.6
2	720	4	3.7
3	60	3	3.5
3	296	4.5	3.8
3	318	5	3.9

Compared with the sample output you gave us through Email, I got the exactly right answers for part of them.

5. Bar plot for SSE and RMSE, and conclude

I collected the data of SSE and RMSE, and plot them out as follows:

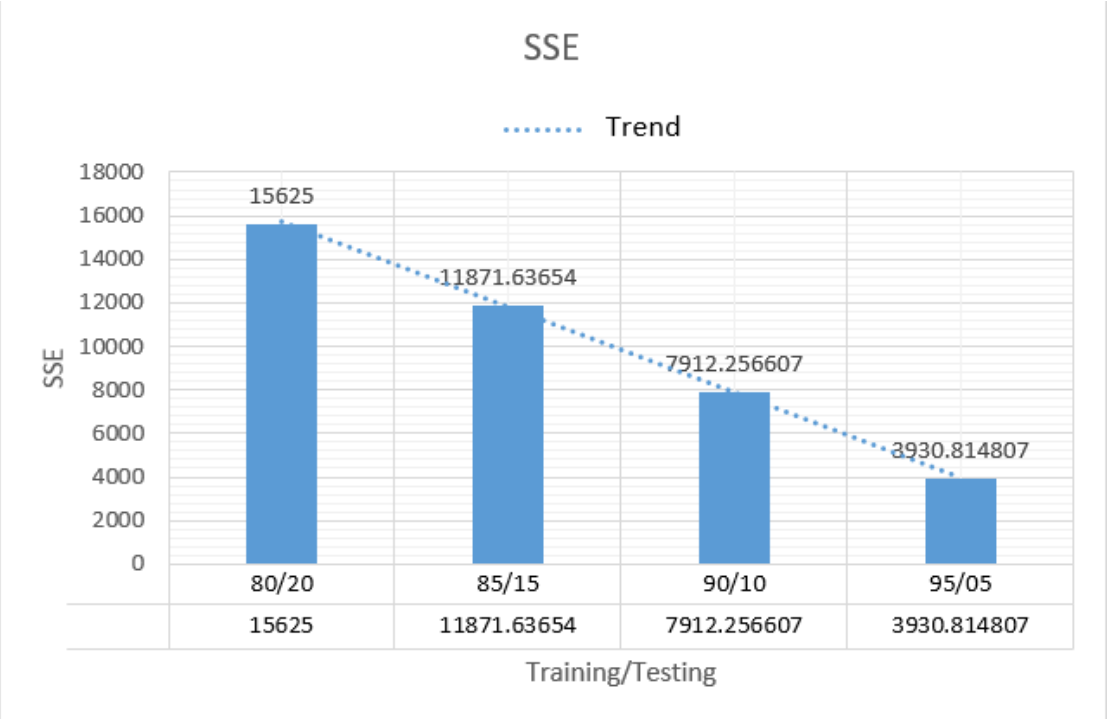
The table of SSE:

DATA	80/20	85/15	90/10	95/05
SSE	15625	11871.64	7912.257	3930.815

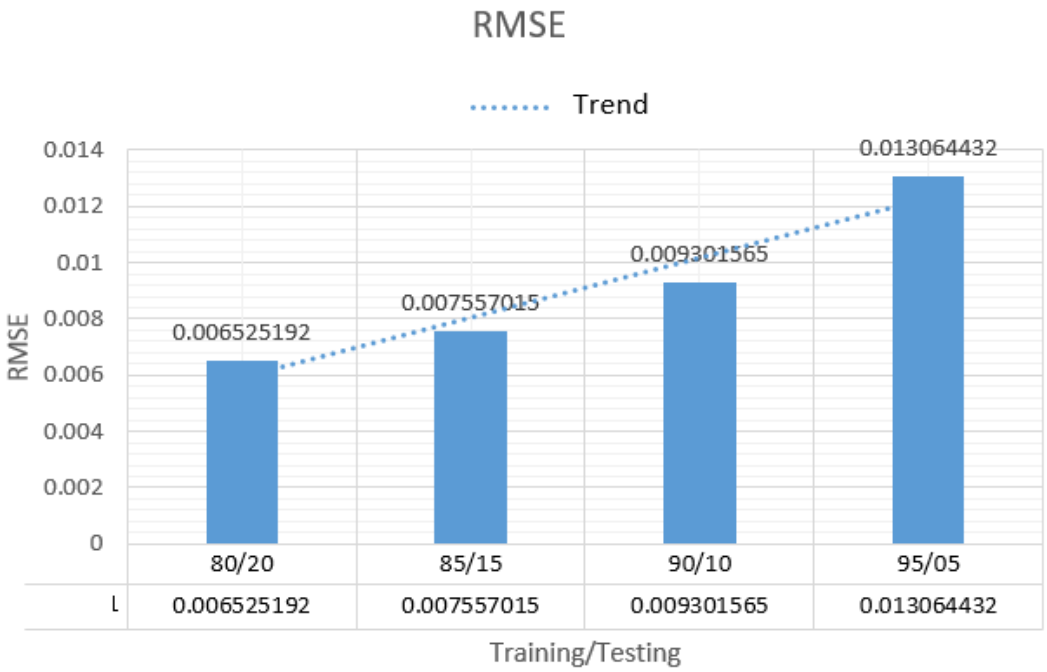
The table of RMSE:

DATA	80/20	85/15	90/10	95/05
RMSE	0.006525	0.007557	0.009302	0.013064

The bar plot of SSE:



The bar plot of RMSE:



Conclusion:

All of these results shown a very good reflection on my prediction. General speaking, more data you use as training data set to predict the future data will lead to a more accurate result, and the SSE and RMSE should be smaller and smaller.