**CS-504 - Final Project – Option 2 – Coding**

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

Student: Rongxin Zheng

Instructor:  Kevin Molloy

Date: Dec 7th, 2016

**Content**

1. **Introduction**
2. **Methods of KNN and CF for this project**
3. **Improvement on model and how I make up the ill data**
4. **Some codes and some brief explanation**
5. **Displaying the results**
6. **Comparison between KNN and CF**
7. **Conclusion**
8. **Introduction**

This course is CS-504, which is arranged for student to teach them how to mine data. As it always is, data mining is a very big branch in computer science, and it always takes an important role in technical companies no matter how big this company is. In data mining, in my opinion, there are normally two different parts. One is to search, collect, preprocess, get, use and show the data from any possible resource because as you know, data is everywhere and it could be anything you can image, touch and see. Another one part is to get the outside data from training data itself. In other words, in this sense, data mining seems like data-creating. Sometimes it is easy for people to get tons of data, but the real problem is how to use data. For example, prediction is always a nice dream of human beings. In the degree of digit, people can translate data to quantify them and predict the trend. As it shows in mathematical models, for any model to predict, all of them should be applied with a really large amount of data. Only big data can perfectly make some statistical value infinitely close to the theoretical values. In this case, it is impossible for people to predict the theoretical value from big data. That’s why, in my opinion, it is right that the prediction should belong to mathematics, but it is kind of data mining.

The goal of my paper is to build a recommender system to predict the rates on some specified movies, which will be given by some special customers, using a large amount of dataset from the webpage of grouplens. The two main techniques I will use are collaborating filtering (CF) and k-nearest neighbors (KNN). I will split the whole data set into two different parts. One will be used as training data, and another one as testing data to tell how good our models are. For each of them, I will calculate SSE, MSRE to compare the results between different models and different parameters.

In the coding part of this paper, I will give my pruning methods which will greatly decrease the complexity of the model. In the same part, because the data set identifies the users and the movies by IDs which are string and cannot be used to reach the data in computer, I used two Hash tables to translate these strings.

1. **Methods of KNN and CF for this project**
   1. **KNN**

KNN is an abbreviation of the model of “K nearest neighbors”, which is usually applied to classify data. The core method in this algorithm is to search the number of nearest neighbor points and use their values to predict and classify the current point. Normally speaking, when we get the K-scale point set of nearest neighbors, we will make prediction through calculating the arithmetical average or geometrical average of this small set. (In this paper, I will use the simple arithmetical average.)

The formula is used to predict:

K is a user set. For all the users in this set should have also rated the movie i.

* 1. **CF**

Collaborative Filtering(CF) is a pretty new technique in the field of information system. Usually, CF is used by people to predict the preference of customer. General speaking, there are two different methods we can pick for this model. One is item-item model, and another one is user-user model. In this paper, I will use item-item method because I want to predict the rate by user on item. For whichever model we pick, the core part of algorithm is the same. The first step is to build an item-item similarity matrix. And then, for the user u and item I, we need to enumerate the whole item set to find those items which are also rated by user u. For those item we got in last step, there is another constrain we should consider: these movies should be similar to movie i, which should satisfy Sim(i, j)>0.

The formula to build the similarity matrix:

U is a user set. All the users u in this set should have rated both of movie i and movie j.

The formula to predict the rate:

N is an item set. For all the items in this set should be also rated by this user u, and they have to be similar to item i which means Sim(i, n)>0.

1. **Improvement on model and how I make up the ill data**
   1. **Heap**

In the model of KNN, every time when we predict the rate r(u,i), which means the rate on movie I and given by user u, we have to search all the users who also rated movie i. Then we sort these users by Euclidean distance. You know, we only need the top k users rather than the all users. We don’t need sort all of them every time. Therefore, if we use the Heap algorithm here, we can get only several top users who are nearest to the current user. It could save much time in this small sorting part.

* 1. **Hash table to get the users who also rated**

In the model of KNN and CF, using easiest way to find all the users who also rated movie i, or all the movies which are also rated by user u, we have to search the whole user set or the whole movie set every single time. It surely wastes much useless time. Therefore, the smarter way to save some time from these frequent processes is to build a Boolean matrix, which is known as Hash table, to record whether or not the user u has rated movie i. The initial value of this matrix should be False before reading the training data.

* 1. **K value in KNN**

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)

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

* 1. **Similarity 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:

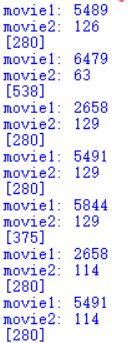
a. r[u][i]=4.0, r[u][j]=3.5;

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)

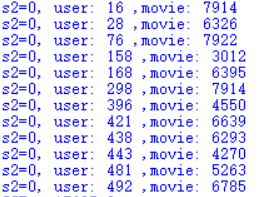


* 1. **N is an empty set when predicting**

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.

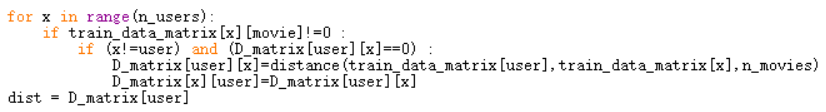
I actually found some movies are in this case as below:



1. **Some my code and some brief explanation**
   1. **KNN**
      1. Pruning method used:

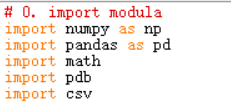
In my model, to prune the complex, I didn’t build the whole distance matrix. I only calculated the cell which we are going to use, which will greatly reduce the running time. My program can get the output in only 5 to 10 min, which is much shorter than the time to build up the distance matrix.

Pruning Code:



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.

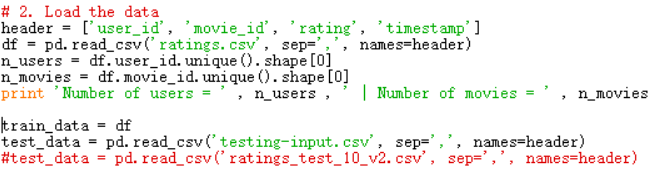
* + 1. Import all the module I will use in this project: (Line 1 – Line 6)



“numpy”, “pandas”, “csv” and “math” are for the basic function;

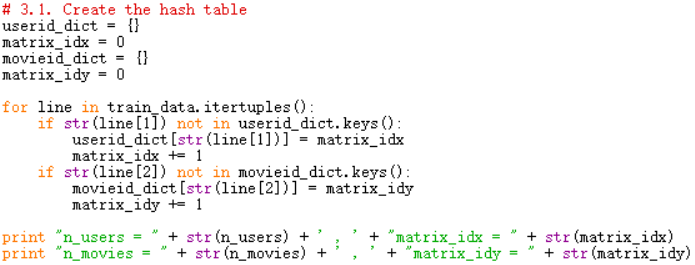
“pdb” is used to debug my code;

* + 1. Load the data: (Line 8 – Line 17)



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.

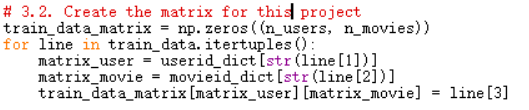
* + 1. Build the Hash Table for userID and MovieID:



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.

* + 1. Create the rating matrix:



line[0] is Index number.;

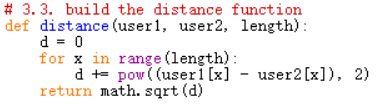
line[1]: user id;

line[2]: movie id;

line[3]: ratings;

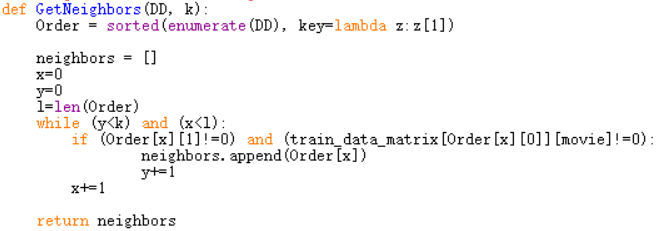
line[4]: timestamp.

* + 1. Build the distance function:



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

* + 1. Get the K nearest neighbors and output the results into CSV file:

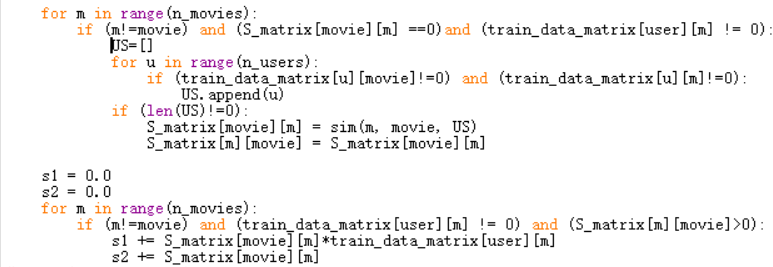


This section contains much code, so I only paste the function of KNN here. Other code is in Line50 to Line123 in my code.

* 1. **CF**
     1. Pruning method used for this question:

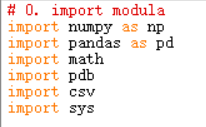
In my model, to prune the complex, I didn’t build the whole similarity matrix. I only calculated the cell which we are going to use, which will greatly reduce the running time. The running time of my program is much shorter than the time to build up the whole similarity matrix.

Pruning Code:



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.

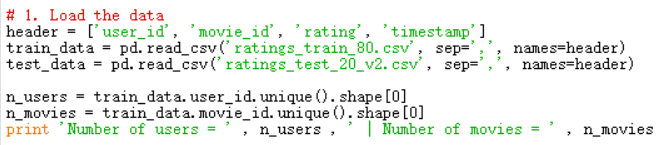
* + 1. Import all the formula



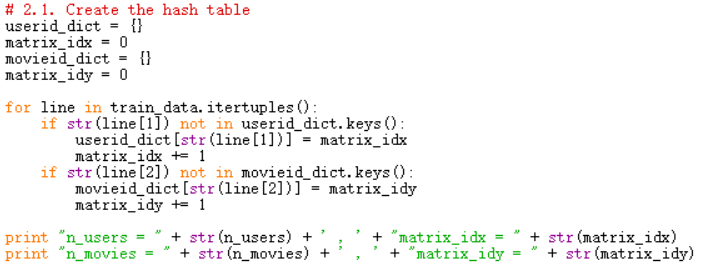
“numpy”, “pandas”, “csv”, “sys” and “math” are for the basic function;

“pdb” is used to debug my code;

* + 1. Load the data

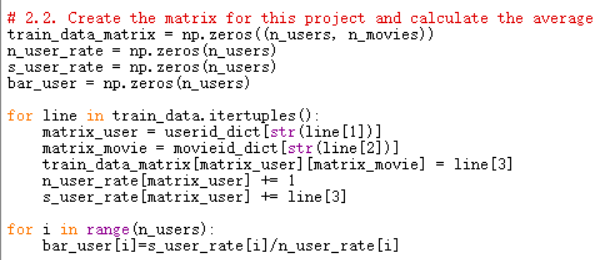


* + 1. Create the HASH table

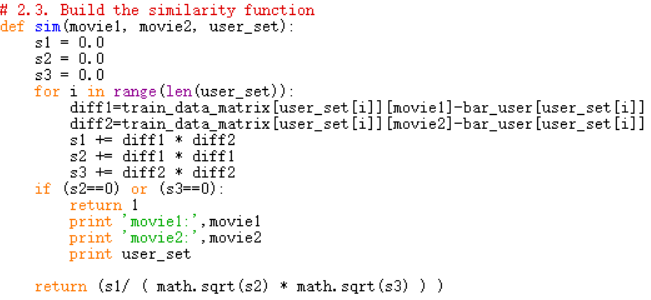


Even when you import the v3 file, the result won’t be different, because the order will keep the same number.

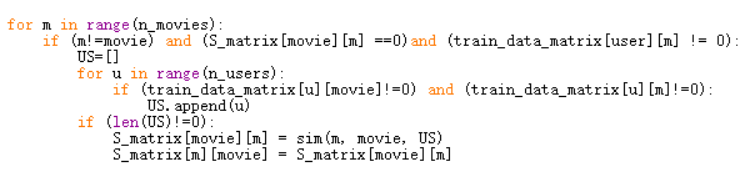
* + 1. Transfer the data frame into matrix and



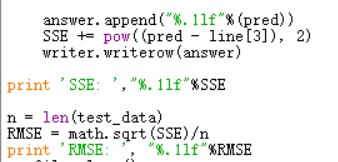
* + 1. Build the similarity function



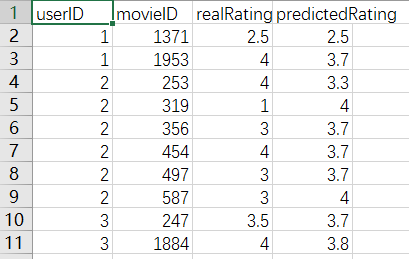
* + 1. Get the user set who rated both items



* + 1. Calculate SSE and RMSE



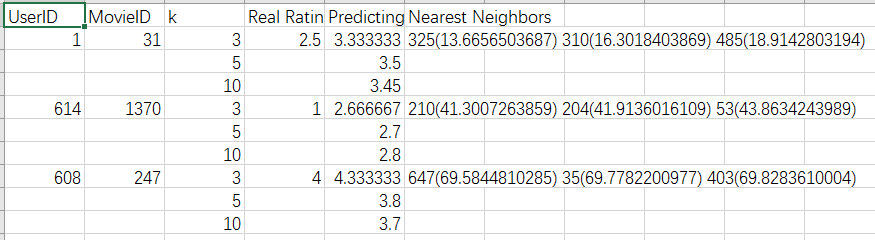
1. **Displaying the results**
   1. **KNN**
      1. The first 10 lines, using 90 percent file as training data set:



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

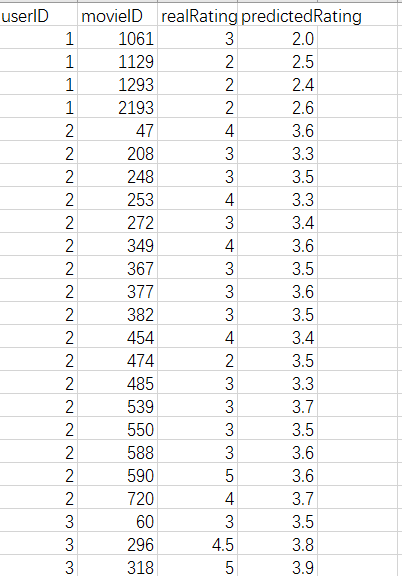
* + 1. Use the whole data set as training data, and compare with the answer you gave in the paper:

(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)



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)

* 1. **CF**



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

1. **Comparison between KNN and CF**
   1. KNN

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)

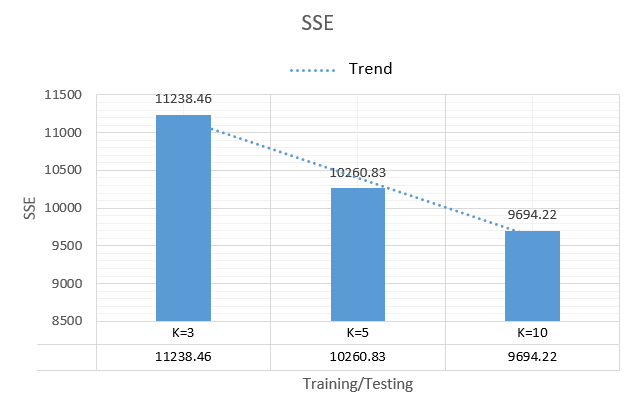
The table of SSE:

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

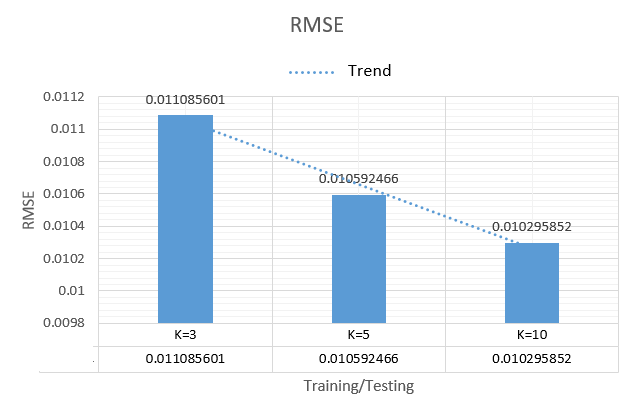
The table of RMSE:

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

The bar plot of SSE:



The bar plot of RMSE:



* 1. CF

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

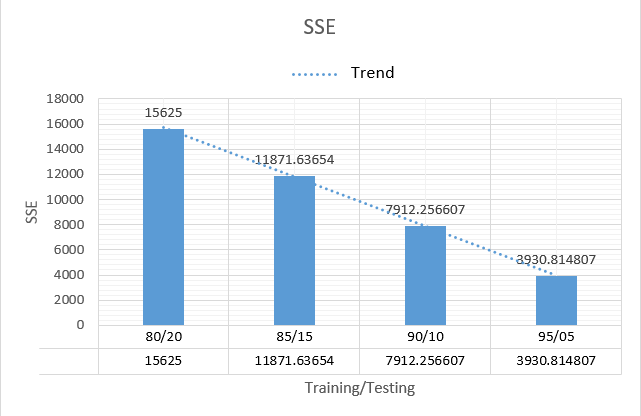
The table of SSE:

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

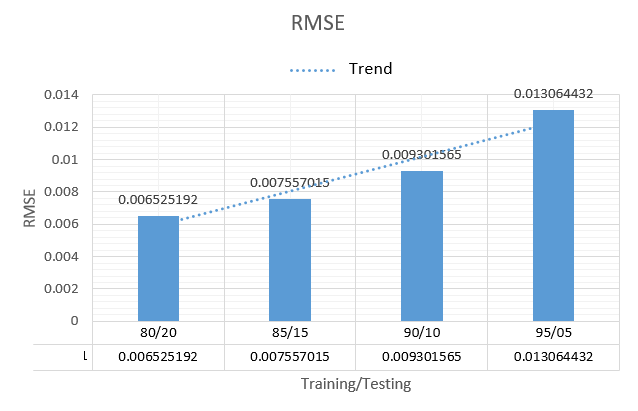
The table of RMSE:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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:



1. **Conclusion**
   1. Comparison among different K values inside the method of KNN:

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 training data to predict, the prediction, of course, will be more accurate. Therefore, the trend of both SSE and RMSE is downtrend. It makes sense.

* 1. Comparison among different size of training data inside the method of CF:

All of these results showed 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. However, based on the plot of SSE, people are wrongly led to believe that more data used in CF will make a better decision. It is incorrect. I also drew the bar plot for RMSE above, and it gave an increasing trend. Obviously, it is wrong. How? Why? You read the improvement part of my code in the section 3 of this paper. I set the all similarity values between items i and j into 1 in the case of divided-by-0. It is not good enough of course. Therefore, in the future work, to make up this problem, we can give three different rules to value the three different situations I gave above.

Besides, the results also show that the scholars should analyze the data in many different aspects before they make a conclusion. As it shows in this case, only SSE sometimes is far from enough.

* 1. Comparison between KNN and CF in this case:

The data above shows the best RMSE on the method of KNN for this case is equal to 0.010296. Although it is good enough for a prediction problem, the best RMSE result of CF method, which is 0.006525, is obviously better.

Therefore, for this movie-rating case, CF better fits the problem than KNN.