Movie Recommendation
Arighna Roy

I. Definition (approx. 1-2 pages)

Project Overview

A recommendation engine can be used for a lot of applications such as categorization of products and responding to users' individual interest. In this project, I have tried to recommend products (movies) to users (viewers) based on the viewer to movie rating data. The underlying technique to achieve the goal is collaborative filtering. I have used both user based and content based collaborative filtering and made a comparative study.

There are multiple techniques to build a recommendation engine such as item set mining, collaborative filtering, content based filtering and knowledge based filtering. Item set mining is used in filtering out the items which are frequently used together. Collaborative filtering can be used when users' ratings for different products are available. For Content based and knowledge based filtering, we need information related to products or some prior knowledge about the market.

Problem Statement

I have a set of users for which the ratings to different movies are available. Now, for a new user with his ratings for few of the existing (rated by existing users) movies is taken as an input. I aim to predict the rating of that user to a movie which he has not rated, but that movie is rated by other users. Let me explain the problem with an example. Suppose, there are 3 users a,b,c. a rated movie 1,2,3; b rated movie 2,3,4; c rated 3,4,5. Now a new user d rated movie 1,4,5; I aim to predict the rating of movie 2,3 for user d.

Step1: preprocess the user rating file. We removed the outliers from the rating dataset. Users who rated too low or too high number of movies are considered as outliers. Similarly, movies rated by too low or too high number of users are considered as outliers.

step2: generate a python dictionary for the user to movie ratings. Each key of the dictionary is a user id(suppose u). Values of each key(u) is another dictionary. The key of each dictionary is a movie id (suppose m) and the corresponding value is the rating of that movie(m) by the user (u).

step3: build a similarity matrix between users. Each key of the dictionary is a user id(suppose u1). Values of each key(u1) is another dictionary. The key of each dictionary is another user id (suppose u2) and the corresponding value is the similarity between the two users (u1 and u2).

step4: build a similarity matrix between movies. Each key of the dictionary is a movie id(suppose m1). Values of each key(m1) is another dictionary. The key of each dictionary is another movie id (suppose m2) and the corresponding value is the similarity between the two movies (m1 and m2).

step5: loop through each user-movie pair and find the predicted rating of the movie for that user using both user based and movie based collaborative filtering.

step6: Compare the predicted value and actual rating. Calculate the MSE.

Metrics

I use the same user-movie pairs from the training set and compare it with the existing rating. Suppose, user a has rated x(<10) to movie 1. I predict the ratings of movie 1 for the user a using my method. The rating of movie 1 for user a is ignored. And the performance of the prediction is measured based on the comparison with the value x (actual rating). I measure the **Mean Squared Error (MSE)** as a performance metric.

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2$$

$\hat{Y}_i$ is the actual rating rating of the movie by that user and Y_i is the predicted rating of that movie by the same user, n is the number of movie and user pair.

II. Analysis (approx. 2-4 pages)

Data Exploration

The data set contains a rating file. I have splitted the data into training and testing dataset. Below is a sample of 10 records from the rating dataset:

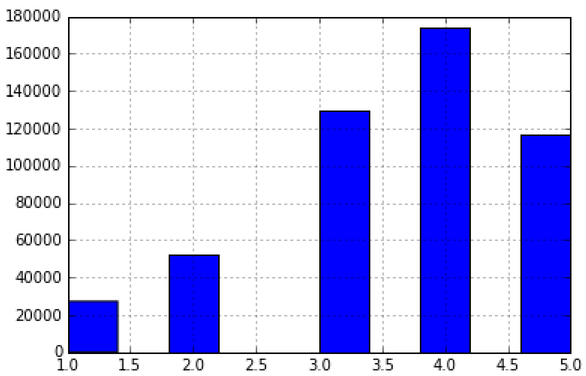
	user	movie	rating	id
0	2783	1253	5	2783_1253
1	2783	589	5	2783_589
2	2783	1270	4	2783_1270
3	2783	1274	4	2783_1274
4	2783	741	5	2783_741
5	2783	750	5	2783_750
6	2783	924	5	2783_924
7	2783	2407	4	2783_2407
8	2783	3070	3	2783_3070
9	2783	208	1	2783_208

Below is a short summary of the dataset:

Total number of ratings:	500100
Total number of users:	3255
Total number of movies:	3551

Below is a summary statistics and the histogram of the ratings:

count	500100
mean	3.60222
std	1.11469
min	1
25%	3
50%	4
75%	4
max	5



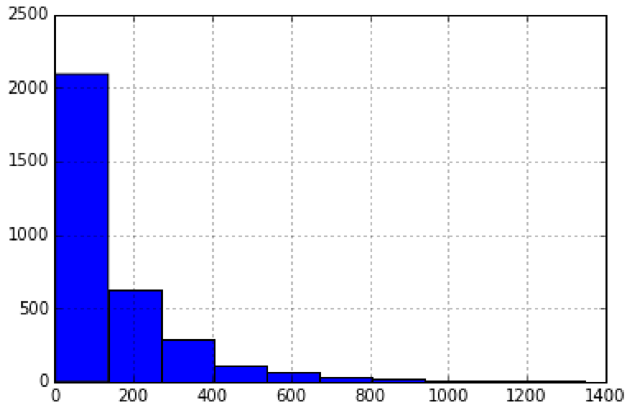
Exploratory visualization

Below are 5 users who rated the highest number of movies and the count of the movies they rated:

User_id	number of rated movies
3618	1344
5795	1272
4344	1271
4510	1240
4227	1222

Below is short summary of the count of the rated movies by each user and its histogram:

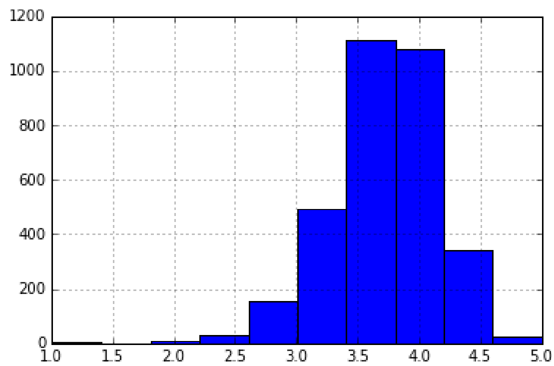
count	3255
mean	153.640553
std	173.90484
min	2
0.25	42
0.5	92
0.75	197
max	1344



We can clearly see that most of the users rated very few number of movies, where as very few rated more than 400 movies.

Below is short summary of the average ratings of each user and its histogram:

count	3255
mean	3.713232
std	0.431985
min	1.015385
25%	3.455723
50%	3.752475
75%	4
max	5



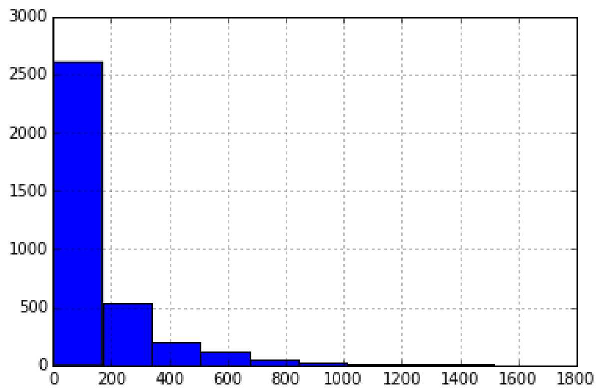
Below are 5 movies which are rated the highest number of time and the count of the usersrated those:

Movie_id	number of users rated
2858	1684
1196	1585
260	1573

1210	1539
127000%	1396

Below is short summary of the count of the users rated each movie and its histogram:

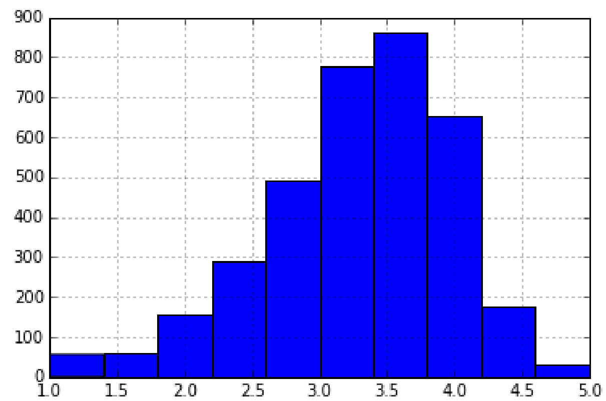
count	3551
mean	140.833568
std	200.241913
min	1
25%	18
50%	66
75%	181
max	1684



We can clearly see that most of the movies are rated by very few number of users, where as very few movies are rated by more than 400 users.

Below is short summary of the average ratings of each movie and its histogram:

count	3551
mean	3.283631
std	0.702784
min	1
25%	2.869126
50%	3.375427
75%	3.792778
max	5



Algorithms and Techniques

I have used the collaborative filtering (both user based and movie based) to find the nearest neighbors for predicting the rating. For each pair of user and movie in the testing dataset, I have omitted the rating and made a prediction based on the all other ratings. Then compared the prediction.

Benchmark

The primary benchmark here would be the mean squared error.

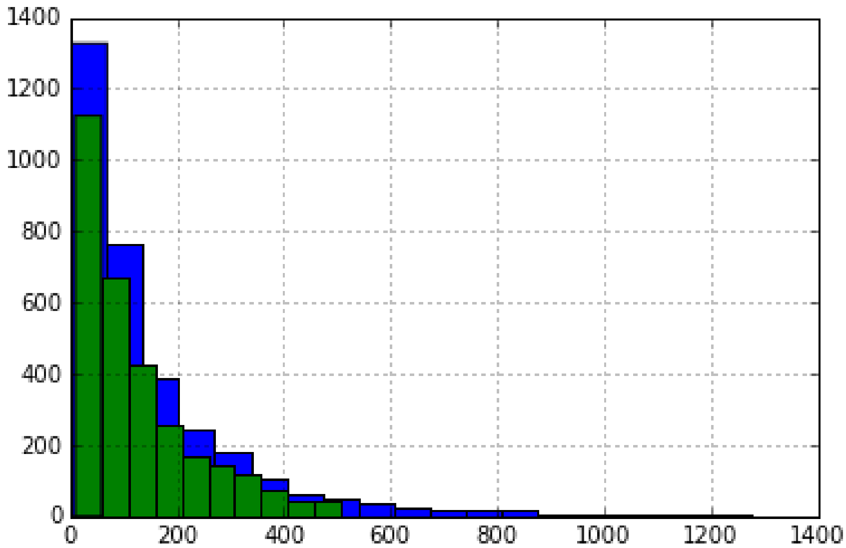
III. Methodology (approx. 3-5 pages)

Data Preprocessing

It's always a good idea to pre-process the data before starting the actual analysis or building the application. This dataset doesn't have any missing values or non-numeric values. I have done outlier detection and removal as part of pre-processing.

1% of the users rated less than 9 movies. 9 is a very small number in terms of movies rated. It's very difficult to judge someone's taste of movie based on the ratings of just 9 movies. So remove the users who rated less than 9 movies for my analysis. 5% of the users rated more than 507 movies. I consider them someone having variety of taste in the choice of movies. Those users should not be used for the analysis. The reason we choose 5% in the upper tail and 1% in the lower tail is because we have a right skewed distribution for the count of movies rated by each user.
I want to consider the mean and the standard deviation of the distribution while finding the outliers in the upper tail of the distribution. 77 users rated number of movies which are 3 standard deviation above the mean of the distribution. But I am dealing with the highly skewed distribution. I would rather prefer the quadrantile data than the variance. Also, I set these values after a careful consideration of the count of movies.

Below is the histogram of the count of movies rated by the each user before (blue) and after (green) the removal of the outliers.



Now, 5% of the movies have ratings from less than 3 users. I feel that movies having very low ratings cannot be removed because those could be movies of rare taste. Also movies with a lot of ratings can be removed because those are the most watched movies which are important to judge users' taste of movies.

Implementation

The implementation of my project is broken down into few ipython notebooks.
Visualization:
In this notebook, I have performed the statistical analysis of the data and further exploratory analysis.
I have used pandas and numpy to read the data and visualize the summary statistics of the data.
Pre-processing:
In this notebook I have detected the outliers and prepared the final data for analysis.
Reco:
In this notebook, I have implemented the prediction algorithm
I have implemented user based prediction, movie based prediction and the average of the previous 2 values to test the accuracy of the result.
Results:
Here, I have generated the plots based the prediction of movie rating.

Refinement

Several methods have been implemented to improve the performance of the algorithm.

- The value of k:
  I have used the nearest neighbour approach to predict the rating of a movie by a user. The number of nearest neighbours is to be optimized to get the best result. I have collected the results for each iteration for different values of 'k' (number of neighbours).

- Distance metric:
  Different distance metrics are applied to find the nearest neighbours.
  - Euclidean distance:
    This is the most widely used distance metric. Euclidean distance is pretty straight forward and easy to calculate.
    The Euclidean distance between two vectors X and Y is calculated as,

$$d(x,y) = \sqrt{\sum_{i}^{n} (x_i - y_i)^2}$$

- Normalized Euclidean Distance:

  It's always a good idea to normalize the rating for a user. For example, suppose one user tend to give a good rating to movies. So a rating of 6 out of 10 is very poor. And the other user might be quite reserved when it comes to rate movies. So 6 could be average for him. So it's always better to normalize the ratings so that scale comes to equal.

- Manhattan Distance:

  This is another distance metric when the absolute magnitude of the distance is desired.

  The Manhattan distance between two vectors X and Y is calculated as,

  $$d = \sum_{i=1}^{n} |x_i - y_i|$$

- Minkowski Distance:

  This the generalized version of both Manhattan and Euclidean distance. I have implemented this distance with the degree of 3.

  The Minkowski distance between two vectors X and Y is calculated as,

  $$d(\mathbf{x}, \mathbf{y}) = \left(\sum_{i=0}^{n-1} |x_i - y_i|^p \right)^{1/p}$$

- Pearson Correlation Distance:

  This is the most useful distance metric when it comes to collaborative filtering. It takes care of the variance of the ratings for each user.

  The Pearson correlation distance between two vectors X and Y is calculated as,

  $$r(X,Y) = \frac{\frac{1}{n}\sum_i x_i y_i - \mu_X \mu_Y}{\sigma_X \sigma_Y}$$

  where μ_X and μ_Y are the means of X and Y respectively, and σ_X and σ_Y are the standard deviations of X and Y. The numerator of the equation is called the covariance of X and Y,

- I have implemented both the user based filtering and movies filtering to predict the rating and checked the accuracy. Also, the average of the previous 2 ratings are tested for accuracy.

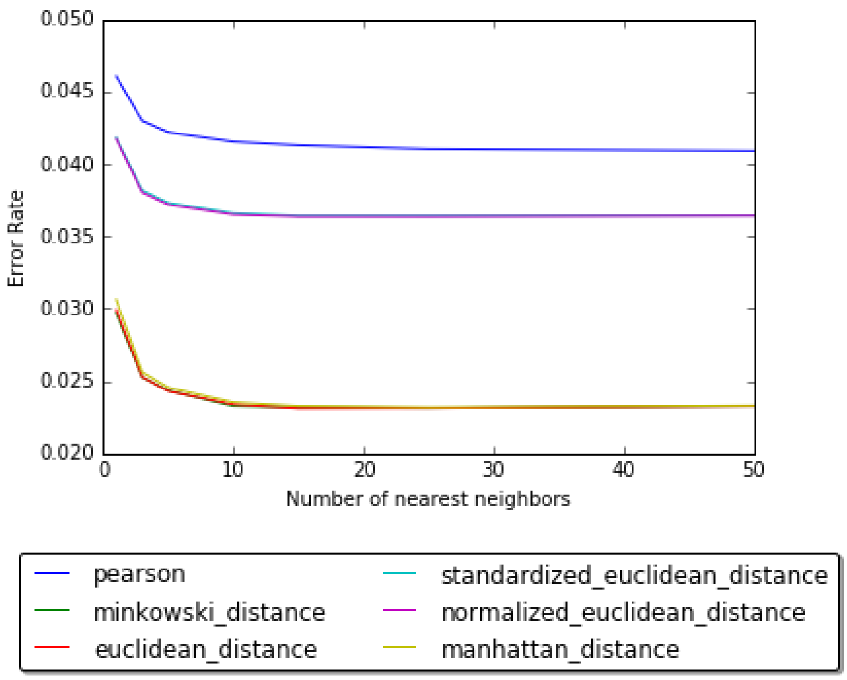
IV. Results *(approx. 2-3 pages)*

I have evaluated the recommendation engine with different parameters which are explained in the refinement section. Now let's check the comparative results.
To evaluate the performance of the algorithm, I have used **root-mean-square error** as the performance metric.

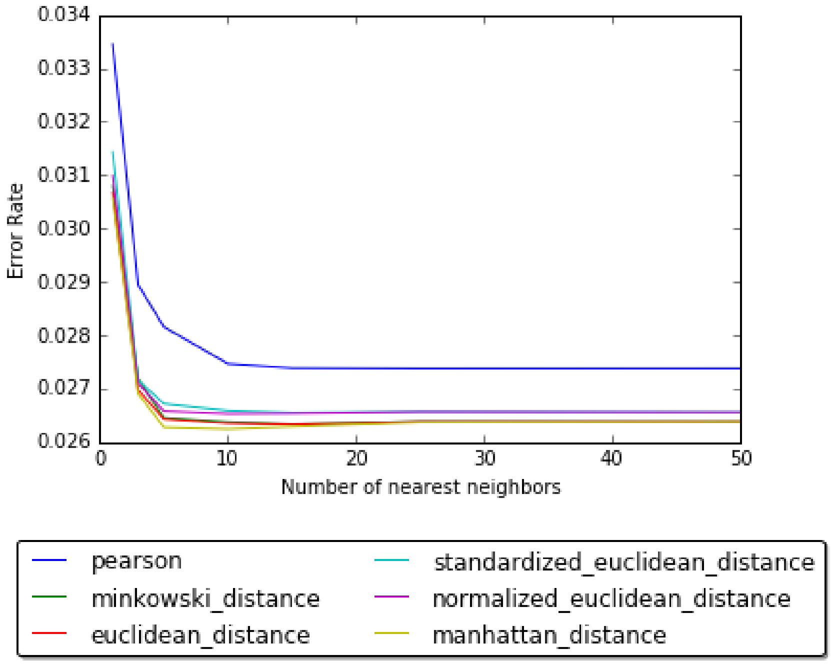
To construct the r.m.s. error, you first need to determine the residuals. Residuals are the difference between the actual values and the predicted values. I denoted them by $\hat{y}_i - y_i$,

where y_i is the observed value for the ith observation and $\hat{y}_i$ is the predicted value.

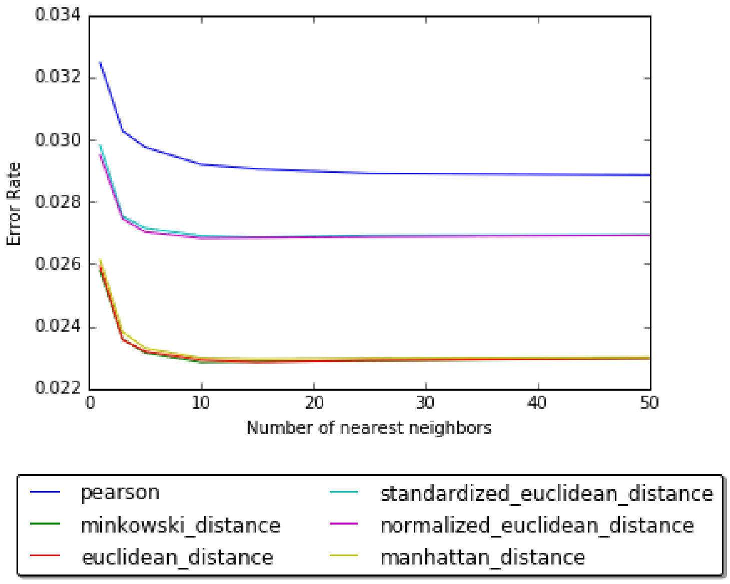
$$RMSErrors = \sqrt{\frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{n}}$$



Error Rate vs number of nearest neighbors for predicted ratings of user based collaborative filtering



Error Rate vs number of nearest neighbors for predicted ratings of movie based collaborative filtering



Error Rate vs number of nearest neighbors for average ratings of user based and movie based collaborative filtering

The performance of the recommendation engine is very good overall. Let us dig more into the parameters.

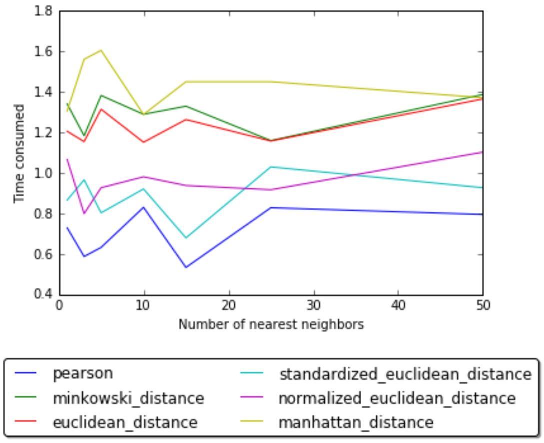
Number of nearest neighbours

All three figures show a sharp decline in the error rate from k=1 to k=3. It further slowly decrease up till k=10 and then freezes. So we can conclude that k=10 is the optimal value of the number of nearest neighbours.

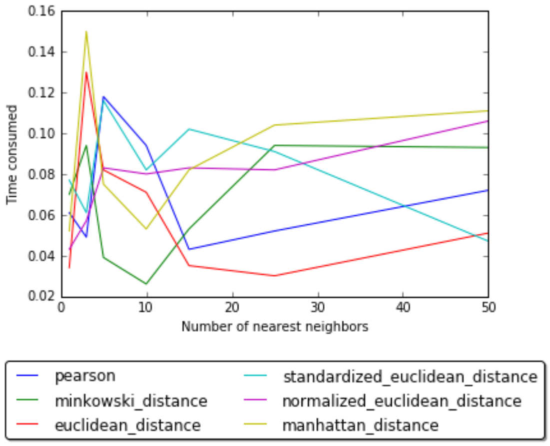
Distance metrics

The above 3 figure clearly shows that the engine performs best when the distance metric is either Manhattan or Euclidean. These show significant difference in performance when used for user based filtering or the average of used based and movie based.
Pearson Correlation metric performs the worst as a distance metric which surprises me because this is one of the most widely used distance metric when it comes to collaborative filtering.

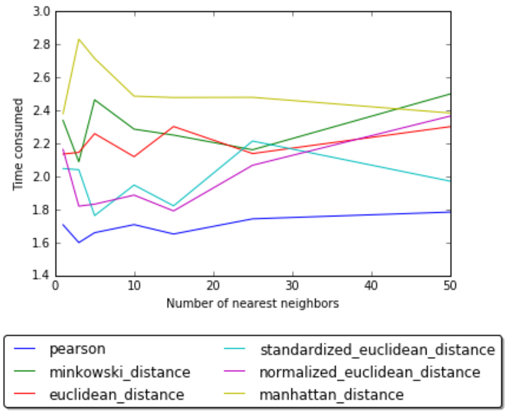
Now let's check the speed of the engine for different parameters. The below 3 figures clearly shows that the engines runs fastest when used with Pearson correlation distance. Manhattan, Euclidean and Minkowski make it slow. From the performance of the algorithm, we can say that there is trade off between speed and performance when Manhattan, Euclidean or Pearson correlation are used. But the engine is fast enough; so we can choose Manhattan or Euclidean for better performance.



Time consumed vs number of nearest neighbors
for predicted ratings of user based collaborative filtering



Time consumed vs number of nearest neighbors
for predicted ratings of movie based collaborative filtering



Time consumed vs number of nearest neighbors
for average ratings of user based and movie based collaborative filtering

V. Conclusion *(approx. 1-2 pages)*

From the results section, we can see that both the speed and accuracy of the engine is quite good. The engine performances the best when used with Manhattan or Euclidean distance as the distance metric and with the number of nearest neighbours to be considered as 10. Also, we can see that it's wise to use the average of the ratings calculated from the movie based filtering and user based filtering.

Improvement

I have used collaborative filtering to predict the rating of a movie by a user. There is another popular method used for the same purpose which is content based filtering. One of the major area of improvement for this engine is to implement the prediction based on content based filtering. Doing a comparative study between these two kinds of filtering for recommendation system would be a nice approach. The features for each movie for my dataset wasn't quite strong. It would be naïve to apply content based filtering for this dataset. But with enough information collected for the movies, we can perform a content based filtering for the same user and movie pairs.