Recommender Systems

Girish Krishna Tokachichu

[gtokachi@gmu.edu](mailto:gtokachi@gmu.edu)

**Introduction**

Recommender systems are being employed in various businesses like e-commerce websites, search engines etc.. with the advent of capability to store and process huge amounts of data collected from the consumers. The purpose of the project is to get to know various problems in recommender systems and core methodologies that have been used in recommender systems and look at the effectiveness of these methods. One of the problems of recommender systems is the sparsity of data. This arises due to the fact that there will be thousands of items, users and most of the items will not be rated by all the users. This lack of data can lead to inaccurate predictions about the user behaviour. Different collaborative filtering models have been applied to movie-lens data set which is very sparse and the resulting performance has been analysed.

**Methodologies**

The research questions that have been investigated are the relative performances of the core collaborative techniques that have been implemented from scratch, whose performance has been tested on the movie lens data set, which can be considered very sparse. More emphasis has been put on trying to implement the models from scratch. There are two prominent set of methods in the field of recommender systems namely content based filtering and collaborative filtering methods. Content based filtering methods perform recommendations based on the features or descriptors of a product. These methods have the ability to capture the specific preferences of a user and recommend items that are not preferred by many other users. One of the disadvantages of these methods is that domain expertise is required to select features. One of the drawbacks of the content based filtering methods, which is, relying on manual selection of the features, can be overcome by using collaborative filtering methods. These methods use similarity between users or items to provide recommendations. Collaborative filtering methods recommend a user A the product based on the preference of another user B who is similar to the User A in user based collaborative filtering. The problem scenario in this case would consist of an user- item matrix with rows denoted by users U1…Ui and columns denoted by items I1…Ij and a cell in the matrix Rij denotes the rating the user i gave to the item j. The idea is to predict the rating the user would give to an item he has not rated previously and based on the predicted rating, decide whether the item should be recommended. Some of the prominent methods to perform collaborative filtering include neighbourhood methods or memory based methods and model based methods like latent matrix factorization methods whose procedures have been discussed in the description of methods being compared section.

**Experiments:**

The experiments have been conducted using the movie lens-100k data set. Different parameters like the similarity measure, neighbourhood size has been varied in the case of the Neighbourhood based methods or memory based methods, where as in case of latent factor matrix factorization method number of times the epochs performed is varied and the performance of the model has been measured in terms of the Root mean square error.

Description of methods being compared:

The problem is modelled as the prediction of missing values of the user-item matrix, where in the prediction of missing rating by a user tries to capture the tendency of the user to like a particular item in a user-item matrix. The following is the brief description of the models implemented .

Neighbourhood based methods:

These can be user-user based or item-item based, depending on how the similarity is calculated to predict the rating. Represent a row of the user-item matrix as a vector. In case of User based collaborative filtering, the vector would consist of ratings a user had given to items. The similarity between all the users or items, represented as vectors, would be calculated. Some of the similarity metrics are Euclidean distance, Cosine similarity and Pearson correlation. Pearson correlation has the advantage of taking the biases into consideration. Based on this, find similar K nearest neighbours to a user and predict the rating the user u would give to item i by,

Ru,i = Ru(mean) + y ∈ Suy (Ry,i – Ru(mean)) \* sim(u, y) / y∈Suy sim(u, y)

Where, Ru,i is rating of user x on item i and Ry,i is rating of user y on item i, Sxy indicates the items that user u and y both rated. The method for item-item based collaborative filtering is same but user vectors are replaced by item vectors. The item-item CF is implemented and similarity metric is varied to record the performance. One tweak I employed in my experiments not inspired from papers is, adjusting the bias. In adjusted cosine similarity, as per paper [1] the bias a user has, is subtracted from the rating user gave to an item, where as in my experiments I tried to assume the possibility of an item having bias, because it can be from a well-known company. The above tweak in model has been made and the results are discussed in the experiments section.

Latent factors based method:

This model has been implemented from scratch for milestone 2. To briefly explain the details of how this model works, it maps users and items to a latent space, of say, f dimensions such that user item interactions can be considered as inner products in that space. The user item matrix is factorized as follows.

R00 R01 R02 R03 R04 U00 U01

R10 R11 R12 R13 R14 U10 U11 I00 I01 I02 I03 I04

R20 R21 R22 R23 R24 = U20 U21 I10 I11 I12 I13 I14

R30 R31 R32 R33 R34 U30 U31

User-item-rating matrix user-latent matrix item-latent matrix

4\*5 matrix 4\*2 matrix 2\*5 matrix

Initially, the user-latent matrix and item latent matrix are filled with random values. The values are tweaked to minimize the following function.

min p\*,q\* ∑(i, j **∈** k) (rij - qjT pi )2 +regcons (|qj|2 +|pi|2 ).

Here pi represents the ith row of the user-latent matrix, qjT represents the jth column of the item-latent matrix, rij represents the value Rij. Training is performed using stochastic gradient descent as follows.

For every rating rij rating available in the user-item-rating matrix of test data, find

eij = rij - qjT pi

Update the respective row, i and column, j of the user-latent and item-latent matrices.

pi = pi + gamma(eij qj – lambda \* pi)

qj = qj + gamma(eij pi – lambda \* qj)

**Description of dataset, Evaluation metric and experiments:**

The models explained above have been implemented and they have been trained and tested using the movielens-100k data set. MovieLens data set was collected by the GroupLens Research Project at the University of Minnesota.

<https://grouplens.org/datasets/movielens/100k/>

The following are some of the features of the dataset used. This data set contains 100,000 ratings (1-5) from 943 users on 1682 movies. It has been given that each user has rated at least 20 movies. Also, the demographic information of the users like age, gender, occupation, zip has also been provided. u.data has the full u data set, 100000 ratings by 943 users on 1682 items in which every user has rated at least 20 movies. The data is of the form user id | item id | rating | timestamp in u.data .

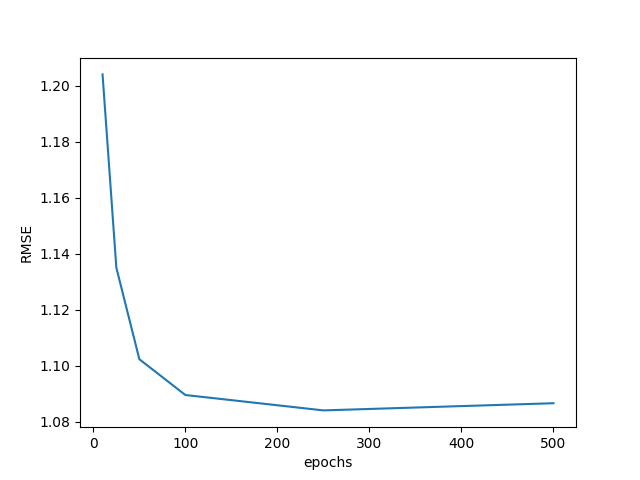
The data sets u1.base and u1.test …u5.base and u5.test are divided into test and train 80 percent to 20 percent ratio and these can be used for 5 fold cross validation.

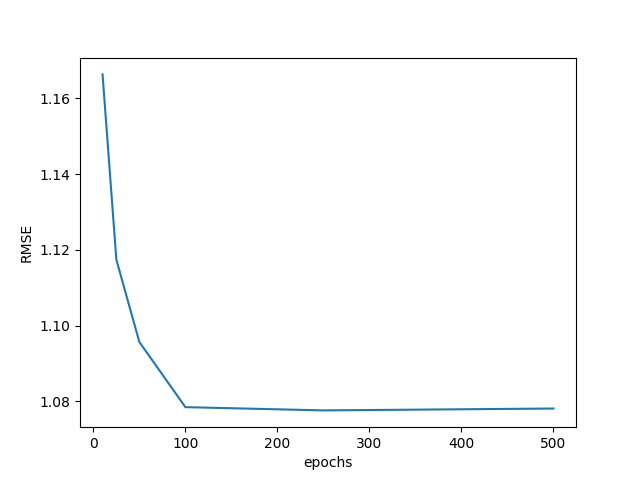
The training has been conducted on the u1.base .. u5.base and the testing has been performed on the respective u1.test .. u5.test and the average of the results has been taken.

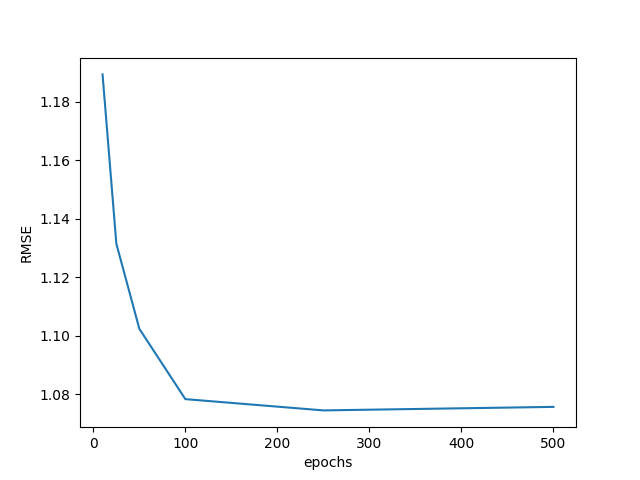
For the calculation of the performance the RMSE(Root mean square error) metric is used.

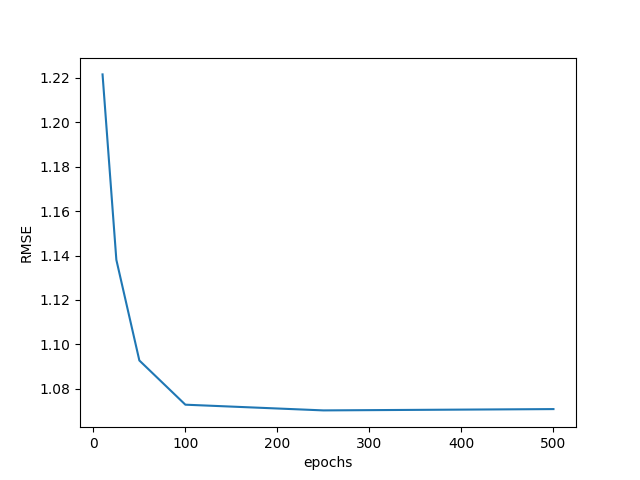
It is given by the formula sqrt( (predicted – actual)2 / number of data points). It measures how much on average the predicted value deviates from the actual value.

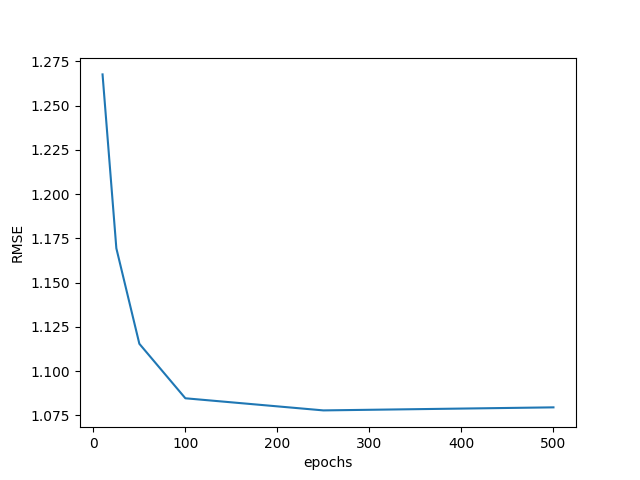
The following were some of the important results observed. The RMSE vs number of training epochs for the Matrix factorization is given as follows.

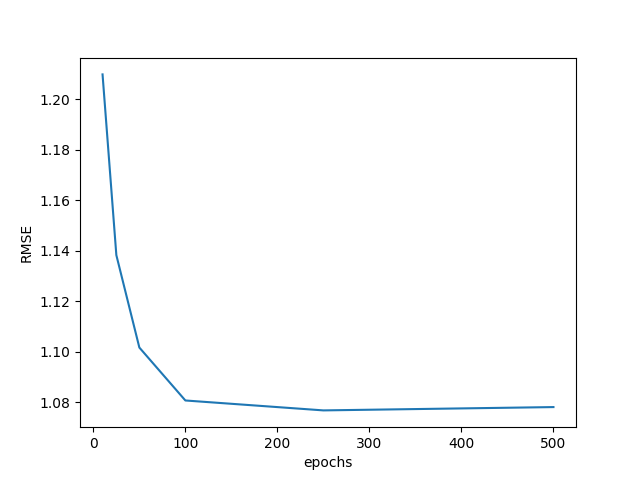
Test train split 1

Test-train split 2 

Test-train split 3

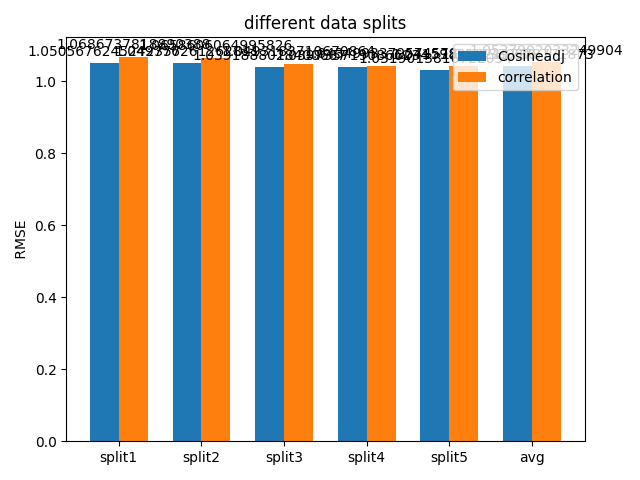
Test train split 4

Test train split 5

Average of all test train splits 

Based on the experiments conducted , the best number of stochastic gradient epochs would be 250 on an average, over all the splits. One draw back of my model is inability to scale to higher latent factors due to double scalar exceptions, since I implemented using lists. Paper claims to get an RMSE of 0.95 [2] for basic matrix factorization model and my results obtained were very close in the range of 1.08.

The following were the results obtained for the item based neighbour hood model for the for different similarity metrics.



The similarity metric change, I thought of, performed better than correlation, but not better than the best metric which is adjusted cosine in paper[1].

Also comparing basic matrix factorization with item item collaborative filtering, MF takes 1/3 times the time, it takes to train item item collaborative filtering. Hence MF is better than item-item CF interms of efficiency with RMSE being almost the same(a difference of 0.03) approximately.

**Source code:**

Language used:Python

Libraries used: numpy for item-item vectors to be passed for similarity metrics, pandas for reading data and converting into matrix and accession locations in matrix in milestone 3, scipy for similarity metrics.

**References:**

**[**1]Item-based Collaborative Filtering Recommendation Algorithms Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl

[2] Matrix factorization techniques for recommendation systems. Yehuda Koren, Yahoo Research Robert Bell and Chris Volinsky, AT&T Labs—Research

[3]User-based Collaborative-Filtering Recommendation Algorithms on Hadoop Zhi-Dan Zhao,Ming-Sheng Shang