Recommendation Systems

Yash Pattarkine
Department of Computer and Information Sciences
University of Florida
Gainesville

1 Team Introduction

1. **Yash Pattarkine - 28616005**

2 List of Papers

1. **Item-based Collaborative Filtering Recommendation Algorithms** by Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl
2. **Probabilistic Matrix Factorization** by Ruslan Salakhutdinov and Andriy Mnih

3 Introduction

In this project, i am planning on studying the working of different algorithms to implement a recommendation system. The 2 algorithms that i intend to work on, are Collaborative Filtering and Probabilistic Matrix Factorization.

Recommender systems are taking more and more place in our lives especially due to the rise of Youtube, Netflix, Amazon and other such web services. They can't be avoided in our daily journeys. The e-commerce sites, online advertisement sites and many such sites use them to recommend things to us.

Vaguely speaking, recommender systems are algorithms aimed at suggesting relevant items to users. Eg. movies to watch, books to read, products to purchase, etc. As a proof of the importance of recommender systems, we can mention that, a few years ago, Netflix organised a challenges (the "Netflix prize") where the goal was to produce a recommender system that performs better than its own algorithm with a prize of 1 million dollars to win.

Why they interest me the most is because they exactly know what the user wants to see (if we take the example of Youtube), even if the user himself does not. It seems quite astonishing that they almost always make the most of the user. For eg, if we visit an e-commerce site, they recommend us to buy something and even if we have not come looking for that object, we end up buying that. Or intending to see just one video on Youtube, we end up seeing multiple.

Through the study of these 2 papers, mentioned above, i can not only understand about recommendation systems but also learn ways to design my own recommendation systems. I intend to study 2 different algorithms to implement the recommendation system. The first paper talks about the most common algorithm used to implement the recommender system called the Collaborative Filtering and the second is the Probabilistic Matrix Factorization method which is suitable for large datasets. I intend to work on the movielens dataset which has all the attributes relating to a movie.

4 Item Based Collaborative Filtering

4.1 Introduction

Collaborative filtering algorithm works by building a database of users' preferences for different items. For example, a new user, A, is matched against the database to discover neighbors, which are essentially other users who have similar taste as A. Items that the neighbors like are then recommended to A, as it is a good chance that A might also like them. Although Collaborative filtering is very successful in both research and practice and also in E-commerce and information filtering applications, it is faced by some challenges.

The first challenge is the improvement of the scalability of the collaborative filtering algorithms. Although these algorithms are able to search tens of thousands of neighbors at a time, the demands of modern systems are to search tens of millions of neighbors which are essentially potential. Also, these algorithms have performance problems for individual users for whom the site holds large information.

The second challenge is to improve the quality of recommendations to the users.

4.2 Problem Statement

Try to improve the scalability and performance of the collaborative filtering algorithm by using the item-based approach instead of user based approach.

4.3 Algorithm

4.3.1 Item Similarity Computation

This is the critical step in item-based collaborative filtering algorithm. It computes the similarity between items and then selects the most similar items. The basic idea in similarity computation between two items i and j is to first isolate the users who have rated both of these items and then to apply a similarity computation technique to determine the similarity $s_{i,j}$.

In this figure, the rows represent users and columns represent items. There are different ways to compute similarity between items. Here, i am presenting 2 such methods. They are cosine based similarity and correlation based similarity.

Cosine based Similarity In this case, two items are thought of as two vectors in the m dimensional user-space. The similarity between them is measured by computing the cosine of the angle between these two vectors. Formally, in the m n ratings matrix in Figure 1, similarity between items i and j, denoted by sim(i, j) is given by

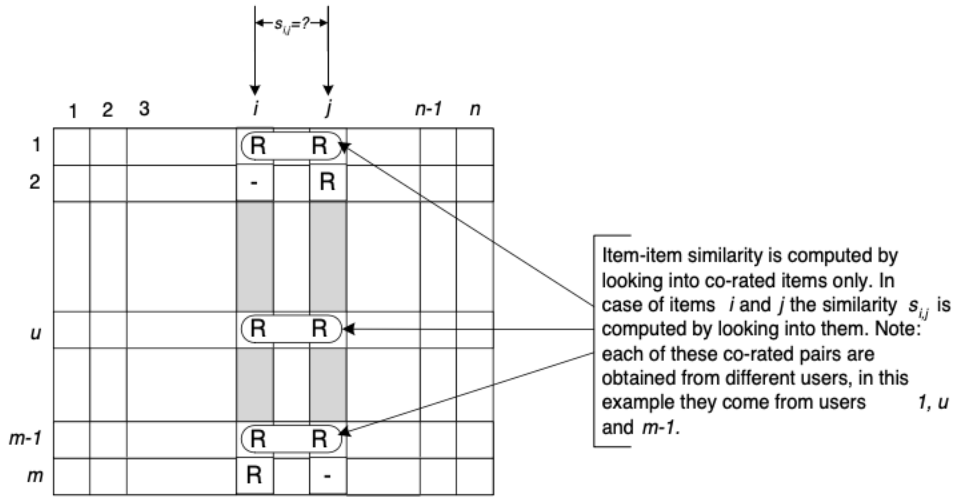


Figure 1: Isolation of co-rated items and similarity computation

$$sim(i,j) = cos(\vec{i},\vec{j}) = \frac{\vec{i}\cdot\vec{j}}{\|\vec{i}\|_2 * \|\vec{j}\|_2}$$

where "$\cdot$" denotes the dot product of 2 vectors.

Corelation based Similarity In this case, similarity between two items i and j is measured by computing the Pearson-r correlation $corr_{i,j}$. To make the correlation computation accurate we must first isolate the co-rated cases (i.e., cases where the users rated both i and j) as shown in Figure 1. Let the set of users who both rated i and j are denoted by U then the correlation similarity is given by

$$sim(i,j) = \frac{\sum_{u\in U}(R_{u,i} - \overline{R}_i)(R_{u,j} - \overline{R}_j)}{\sqrt{\sum_{u\in U}(R_{u,i} - \overline{R}_i)^2 (R_{u,j} - \overline{R}_j)^2}}$$

Here $R_{u,i}$ denotes the rating of user u on item i, R_i is the average rating of the i^{th} item.

4.3.2 Prediction Computation

The most important step in a collaborative filtering system is to generate the output interface in terms of prediction. Once we isolate the set of most similar items based on the similarity measures, the next step is to look into the target users ratings and use a technique to obtain predictions. Here we consider two such techniques.

Weighted Sum As the name implies, this method computes the prediction on an item i for a user u by computing the sum of the ratings given by the user on the items similar to i. Each ratings is weighted by the corresponding similarity $s_{i,j}$ between items i and j.

Regression This approach is similar to the weighted sum method but instead of directly using the ratings of similar items it uses an approximation of the ratings based on regression model. In practice, the similarities computed using cosine or correlation measures may be misleading in the sense that two rating vectors may be distant (in Euclidean sense) yet may have very high similarity.

4.4 Experiments

In this, different items are recommended for 5 different users and the precision, recall, coverage and popularity is also being shown.

```
(base) Yashs-Air:ItemBasedCollaborativeFiltering yhpatt10$ python main.py
********************************************************************
        This is Random model trained on ml-100k with test_size = 0.10
********************************************************************

RandomPredict start...

RandomPredict model has saved before.
Load model success...

recommend for userid = 1:
['1560', '634', '540', '408', '915', '989', '801', '1620', '646', '720']

recommend for userid = 100:
['1245', '124', '1322', '200', '538', '537', '1571', '127', '446', '1652']

recommend for userid = 233:
['1513', '247', '935', '1673', '1287', '246', '1230', '265', '1175', '1501']

recommend for userid = 666:
['781', '1445', '1318', '51', '982', '1471', '472', '453', '1179', '1364']

recommend for userid = 888:
['646', '823', '1154', '597', '999', '1669', '1383', '13', '228', '1464']

Test recommendation system start...
 steps(0), 0.00 seconds have spent..
Test recommendation system success.
total  step number is 943
total 0.02 seconds have spent

precision=0.0082        recall=0.0077   coverage=0.9964 popularity=3.0332

total Main Function step number is 0
total 0.08 seconds have spent
```

5 Probabilistic Matrix Factorization

5.1 Introduction

Many existing approaches to collaborative filtering can neither handle very large datasets nor easily deal with users who have very few ratings. One of the most popular approaches to collaborative filtering is based on low-dimensional factor models. The idea behind such models is that attitudes or preferences of a user are determined by a small number of unobserved factors. In a linear factor model, a user's preferences are modeled by linearly combining item factor vectors using user-specific coefficients.

5.2 Problem Statement

Try to improve the performance of the recommendation systems by applying Probabilistic Matrix Factorization algorithm.

5.3 Algorithm

Suppose we have M movies, N users, and integer rating values from 1 to K. Let R_{ij} represent the rating of user i for movie j, $U \in R^{DXN}$ and $V \in R^{DXM}$ be latent user and movie feature matrices, with column vectors U_i and V_j representing user-specific and movie-specific latent feature vectors respectively. Since model performance is measured by computing the root mean squared error (RMSE) on the test set we first adopt a probabilistic linear model with Gaussian observation noise. We define the conditional distribution over the observed ratings as

$$p(R|U,V,\sigma^2) = \prod_{i=1}^{N}\prod_{j=1}^{M}[N(R_{ij}|U_i^T V_j,\sigma^2)]^{I_{ij}} \tag{1}$$

where $N(x|\mu,\sigma^2)$ is the probability density function of the Gaussian distribution with mean and variance σ^2, and I_{ij} is the indicator function that is equal to 1 if user i rated movie j and equal to 0 otherwise. We also place zero-mean spherical Gaussian priors [1, 11] on user and movie feature vectors:

$$p(U|\sigma_U^2) = \prod_{i=1}^{N} N(U_i|0,\sigma_U^2\boldsymbol{I}), p(V|\sigma_V^2) = \prod_{j=1}^{M} N(V_j|0,\sigma_V^2\boldsymbol{I}) \tag{2}$$

The log of the posterior distribution over the user and movie features is given by

$$\ln p(U,V|R,\sigma^2,\sigma_V^2,\sigma_U^2) = -\frac{1}{2\sigma^2}\sum_{i=1}^{N}\sum_{j=1}^{M} I_{ij}(R_{ij}-U_i^T V_j)^2 - \frac{1}{2\sigma_U^2}\sum_{i=1}^{N} U_i^T U_i -$$
$$\frac{1}{2\sigma_V^2}\sum_{j=1}^{M} V_j^T V_j - \frac{1}{2}((\sum_{i=1}^{N}\sum_{j=1}^{M} I_{ij})\ln\sigma^2 + ND\ln\sigma_U^2 + MD\ln\sigma_V^2) + C \tag{3}$$

where C is a constant that does not depend on the parameters. Maximizing the log-posterior over movie and user features with hyperparameters (i.e. the observation noise

variance and prior variances) kept fixed is equivalent to minimizing the sum-of-squared-errors objective function with quadratic regularization terms:

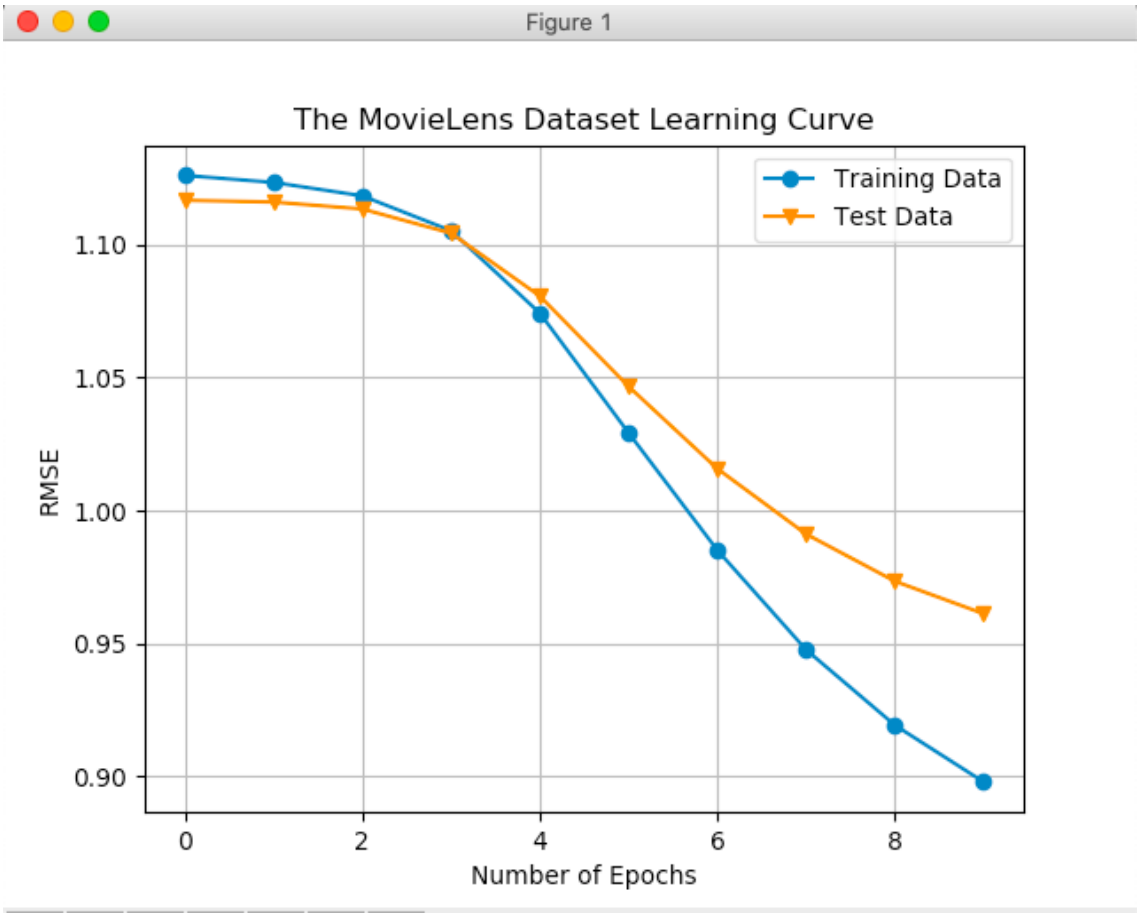
$$E = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M} I_{ij}(R_{ij} - U_i^T V_j)^2 + \frac{\lambda_U}{2}\sum_{i=1}^{N}\|U_i\|_{Fro}^2 + \frac{\lambda_V}{2}\sum_{j=1}^{M}\|V_j\|_{Fro}^2 \tag{4}$$

where $\lambda_U = \sigma^2/\sigma_U^2, \lambda_V = \sigma^2/\sigma_V^2$ and $\|\cdot\|_{Fro}^2$ denotes the Frobenius norm. A local minimum of the objective function given by Eq. 4 can be found by performing gradient descent in U and V . Note that this model can be viewed as a probabilistic extension of the SVD model, since if all ratings have been observed, the objective given by Eq. 4 reduces to the SVD objective in the limit of prior variances going to infinity

5.4 Experiments

In this algorithm, we are seeing the RMSE of the training and testing data and comparing them by plotting a graph.

I am also giving the recall and precision accuracy of the algorithm.



```
(base) Yashs-Air:ProbabilisticMatrixFactorization yhpatt10$ python RunExample.py
943 1682 10
Training RMSE: 1.125999, Test RMSE 1.116649
Training RMSE: 1.123357, Test RMSE 1.115978
Training RMSE: 1.118321, Test RMSE 1.113380
Training RMSE: 1.105210, Test RMSE 1.104277
Training RMSE: 1.074248, Test RMSE 1.080645
Training RMSE: 1.029209, Test RMSE 1.046655
Training RMSE: 0.985111, Test RMSE 1.015738
Training RMSE: 0.947658, Test RMSE 0.991141
Training RMSE: 0.919294, Test RMSE 0.973422
Training RMSE: 0.897956, Test RMSE 0.961197
precision_acc,recall_acc:(0.06255319148936203, 0.03368187884503869)
(base) Yashs-Air:ProbabilisticMatrixFactorization yhpatt10$
```

6 Conclusion

From the item based collaborative filtering algorithm, we saw that item based techniques are better to some extent than the common user based systems. They can scale to larger data sets and at the same time produce high quality recommendations. We also saw the Probabilistic Matrix Factorization algorithm. This can also be efficiently trained and successfully applied to a large dataset containing over 100 million movie ratings.

References

[1] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl, Department of Computer Science and Engineering University of Minnesota, Minneapolis

[2] Ruslan Salakhutdinov and Andriy Mnih, Department of Computer Science, University of Toronto