UNSW
THE UNIVERSITY OF NEW SOUTH WALES

COMP9417
Machine Learning Project

Recommender System Using Collaborative Filtering

Tingfeng Lin z5147976
Tianwei Zhu z5184819
Yi Xiao z5125251
Weichen Luo z5171167

Introduction

With the development of wireless network and surge of smart devices, all of us are surrounded by tons of information every day. Consumers have a huge number of choices whenever they want to do something, such as dinning out or watching a movie. This makes the prominence of recommender system more and more obvious for modern business.

Collaborative filtering is probably the most common technology for recommender system. The main idea of collaborative filtering is to use the wisdom of population, which assumes that customers who have similar tastes in the past will have similar tastes in the future. The profile of user for the collaborative filtering is a vector of items and the corresponding ratings the user gave.

In this project, we built a movie recommender system using four different approaches, including a pure SVD latent factor model, SVD with collaborative filtering model, SVD++ and neural network. The results of these models were obtained from a 5-fold cross-validation experiment and compared. The data set used in this project is the 100k MovieLens data set.

Related work

Traditional collaborative filtering method is based on user profiles and performs prediction based on user similarity with the assistance of Pearson correlation, which means the final prediction will derive from nearest neighbours who potentially have same tastes with the aim user.

$$sim(\overrightarrow{a}, \overrightarrow{b}) = \frac{\sum_{u \in U}(r_{u,a} - \overline{r}_u)(r_{u,b} - \overline{r}_u)}{\sqrt{\sum_{u \in U}(r_{u,a} - \overline{r}_u)^2}\sqrt{\sum_{u \in U}(r_{u,b} - \overline{r}_u)^2}}$$

If item based collaborative filtering is implemented, the similarity calculated in row-based will be turned to be column-based. The similarity is measured in cosine similarity of two item vectors.

$$sim(\overrightarrow{a}, \overrightarrow{b}) = \frac{\sum_{p \in P}(r_{a,p} - \overline{r}_a)(r_{b,p} - \overline{r}_b)}{\sqrt{\sum_{p \in P}(r_{a,p} - \overline{r}_a)^2}\sqrt{\sum_{p \in P}(r_{b,p} - \overline{r}_b)^2}}$$

However, both methods above are suffered from matrix sparsity due to the lack of explicit feedback collected from users. Many methods are figured out to solve this problem. Basically, we can infer some implicit information from user's length of browsing time or searching records or do imputation with the mean of values.

The imputation as well as the inferring achieve some improvement sometimes, but the best method is still trying to make use of what we actually have, instead of getting too much simulated data involved. As a result, SVD-based decomposition and hidden embedding layer learned by neural network to get more latent correlations become more and more popular.

For this project, we have done many explorations. First, we implemented a basic SVD latent factor model only based on the user-rating table and used the prediction result as a baseline of our following work. After that, more features were involved to enhance the SVD based method. Finally, we built a neural network based on almost all features we have to learn the hidden relationship between users and movies.

Implementation

The dataset is ml-100k collected by University of Minnesota from the MovieLens web site. It contains 100,000 ratings (1-5) from 943 users on 1,682 movies, simple demographic info for the users (age, gender, occupation, zip code). Each user has rated at least 20 movies.
In this project, we have implemented the following four models.

1. Pure SVD Based on User-Rating Table

In this model, u.data in the ml-100k was mainly used and u.item was used to be assistance to get movie names for recommendation phrase.
The u.data set contains 100,000 ratings by 943 users on 1,682 movies. We first pivoted this table to a user-movie matrix with the ratings as value, and we obtained a matrix with 943 rows and 1,682 columns. Because the majority of movies is never rated by an individual, the matrix we got is quite sparse. Singular Value Decomposition is introduced to decompose the sparse matrix into three dense matrices.

$$M = U \times \Sigma \times V^T$$

We only picked the 7 most important components from the result we got with the assistance of Scikit-learn, which means matrix U, Σ, V will be 942×7, 7×7 and 1,643×7 respectively. Our rating prediction for one specified user and one movie is calculated by:

$$\hat{r}(user, movie) = \bar{r}_{user} + U_{user} \times \Sigma \times {V^T}_{movie}$$

2. SVD++

Although pure SVD is one possible solution to deal with sparse matrix and easy to implement, but the prediction result is based on unrenewable factors. SVD++ is one method that makes parameters of model renewable, which are learnt from the loss between prediction and given ratings.

We predict the rating based on the following equation, μ denotes the average rating of all given movies, b_u and b_i denote the bias of user and of movie, q_i,p_u are explicit factors of

movie and user that can be learned through gradient descent. y_j is one implicit factor that describes the rating preference of one user.

$$\hat{r}_(u,i) = \mu + b_u + b_i + q_i^T(p_u + \left|N_{(u)}^{-\frac{1}{2}}\right| \sum_{j \in N_{(u)}} y_j)$$

To minimize the loss function

$$loss = \sum_{u,i \in K} (r_{u,i} - \hat{r}_{u,i})^2$$

the gradient descent rule is as following:

$$b_u \leftarrow b_u + \gamma(e_{ui} - \lambda b_u)$$

$$b_i \leftarrow b_i + \gamma(e_{ui} - \lambda b_i)$$

$$q_i \leftarrow q_i + \gamma(e_{ui}(p_u + |N(u)|^{-\frac{1}{2}} \sum_{j \in N_{(u)}} y_j) - \lambda q_i)$$

$$p_u \leftarrow p_u + \gamma(e_{ui} q_i - \lambda p_u)$$

$$\forall j \in N_{(u)} : y_j \leftarrow y_j + \gamma(e_{(ui)} \left|N_{(u)}\right|^{-\frac{1}{2}} q_i - \lambda y_i)$$

3. SVD with collaborative filtering

3.1 Based on user features

SVD is a popular choice to simplify data, mitigate noise and improve algorithm result. Original datasets are represented by much smaller datasets with SVD, and those smaller datasets can be regard as related features or latent factors.

In the past several years, collaborative filtering has played a role in recommender system as a common and useful method. Using collaborative filtering, recommender system calculates similarity between a pair of users or items. High similarity means high possibility that two users have similar preference, or two items have similar features. For example, if user A and user B have high similarity and user B prefers item C, then there is much possibility that user A also prefers item C.

In this Movielens Dataset, the number of movies is much larger than that of user, so we purpose a method combining SVD and collaborative filtering based on user features.

Process

We pivoted original ratings dataframe and denoted it as dataMat and constructed a score matrix. We used built-in SVD function in numpy Linalg package to decompose dataMat into

U, Σ, V^T, then kept 90% of information in Σ and discarded the rest being noise or redundant features. We multiplied sliced $\Sigma_{k\times k}, V^T_{k\times k}$ and data matrix by following equation:

$$dataMat^T * V^T_{k\times k} * \Sigma^I_{k\times k}$$

and got the transformed data matrix with enhanced user features, which will be used as the comparison source. We then compared target user with other users who has rated target items by Pearson correlation. The sum of multiplication of every pair of similarities and ratings will be the prediction rating of target user to target item, in other words, we can regard each similarity as weight of each corresponding users to target user (rescale to 1-5 if necessary). MSE between prediction and true rating in test dataset was used to measure performance.

3.2 Based on both user features and movie features

Compared with the above, the only difference is that besides the comparison of users, this algorithm also considers the similarities between target item with other items, and calculates another prediction rating with the same method as above. The mean result of two rating prediction is used as the final rating.

4. Neural Network

In this model, u.data, u.genre, u.user, u.occupation, u.item were included. After the pre-processing, u.data was turned into a dataframe with userId, moiveId, rating as features, timestamp was dropped.

u.genre was turned into a dictionary with genre as key and id as value.
For the u.item, we first stripped the year of the movie title, and made all title's word into a bag of words, then mapped each word in every title to a list of indices in bag of words, and added padding to length 15. Since one movie may belong to many genres, we mapped every movie's genres to a list with the help of dictionary we got form u.genre, and added padding to length 18. Gender and occupation were converted to numeric values. Equal width binning was applied to age, which divides all users into 10 different age ranges.

Network model

The graph below is the network model that we have built. It contains multiple embedding layers, convolutional layers, fully connected layers and dropout layers.

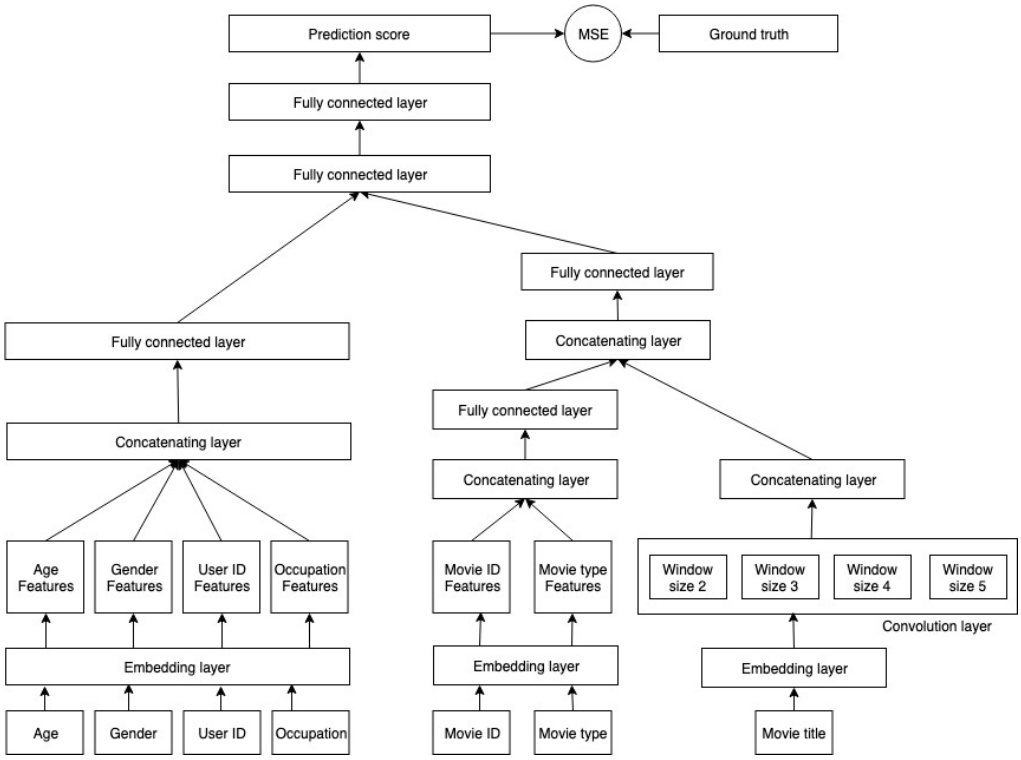

Figure 1. Neural Network

Model analysis

User features and Movie ID:

By studying the field types in the dataset, we found that some are categorical fields. The usual pre-processing is to encode these fields using one-hot encoding, but fields like user ID and movie ID will become highly sparse, and the input dimensions will be a sharp expansion. This is what we do not want to see.

Considering this problem, we converted these fields into numbers during pre-processing. We used this number as the index of the embedded matrix. The first layer of the network uses the embedding layer with dimension (N,32). The idea is similar to word2vec technique. We assigned a feature dimension space to each attribute of the user and movie. This is similar to specifying a feature dimension space for each word in natural language processing (NLP).

Movie genre:

Sometimes one movie might have multiple genres. Hence, the index from the embedding matrix is an (n,32) matrix. We summed up this matrix into the vector (1,32).

Movie title:

For the movie title, in the first layer, we used a word embedding layer. The next layer uses a number of convolution kernels of different size to convolve on the embedding matrix. This is

not the same as convolving the image. The convolution of the image is usually 2x2, 3x3, 5x5 and the text convolution is to convert the embedding vector of the entire word, so the size is (number of words, vector dimension). The third layer is a max pooling layer to extract key factor in the vector. Finally, dropout layer was added for regularization. After these processes, we can get the characteristics of the movie title.

After indexing the features from the embedding layers, each feature is passed to the fully connected layer then we can get a (1,200) feature vector combined with user features and movie features.
This network learns user's features and movie's features by minimising the MSE between prediction result and ground truth.

Results

For this project, we decided to use Mean Square Error (MSE) to measure the different outcomes generated by different models. We conducted a 5-fold cross-validation experiment to compare the results from four different models.

1. Pure SVD Based on User-Rating Table

Since this model does not take any effect that may cause overfitting or bad prediction into consideration, the performance of this model is pretty poor.
The MSE loss is around 1.47, which means the average deviation between our prediction and true rating is more than 1.20. The next two approaches will focus on how to improve the accuracy.

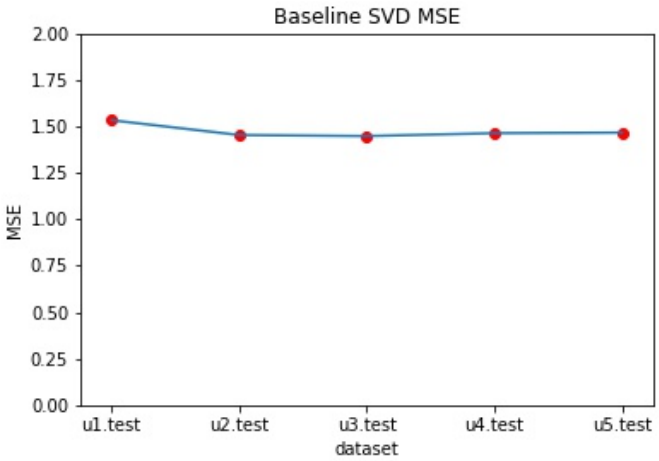

Figure 1.1 MSE of Pure SVD++

2. SVD++

From the MSE figure below, we can see the outcome of SVD++ improves a lot compared with the pure SVD's. The average MSE is around 0.83, which means the average deviation is less than 0.92.

We generated the above result with following hyper parameters: iteration = 30, gamma = 0.04 and lambda = 0.15, since the complexity of the algorithm we have implemented is quite high, we do not have enough time to fine-tune the model to the best state.

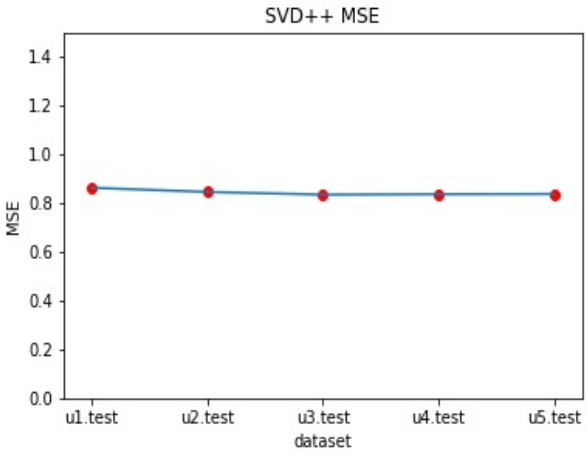

Figure 2.1 MSE of Pure SVD++

3. SVD with collaborative filtering

As we can find in above two graphs, both curves are similar, but the algorithm has slightly lower MSE among 5-fold cross-validation test when it focuses on both user features and movie features. Thus, it can be proved that, in these experiments, both features help predict ratings more accurately.

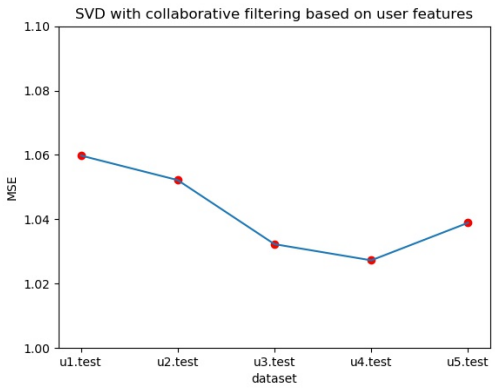

Figure 3.1 Based on user features

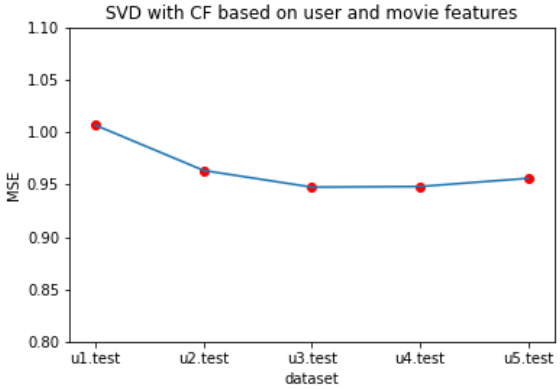

Figure 3.2 Based on user and movie features

4. Neural network

As we can see from the graphs below, the training accuracy and validation accuracy reached 0.4 and training loss decreased from 1.4 to nearly 1.0. However, the validation loss declined from 1.4 to 1.0 then constantly fluctuated. After our discussion, following are some factors that might cause this phenomenon:

- Features like user ID, user gender, user job and movie ID are not closely related to user ratings. These factors might affect the accuracy of the ratings. Compared with the SVD methods, although using neural network is not the worst, SVD only uses the rating table

to make predictions. The rating data can intuitively reflect the user's preference for the movie, and the results are more realistic.

- This model considers movie title feature. Looking at all the movie titles, we can find that many movies have titles of 4 to 5 words. However, when we set the features of the movie title, we used the title embedding of length 15, which highly sparsified this feature makes a lot of padded zero in the title. This might affect the results.
- Through cross-validation, we found that this model is easy to overfit. We made improvements on the original model by adding some dropout layers and a fully connected layer at the end of the model with L2 regularization to prevent overfitting. It has performed better than before but still overfits sometimes.

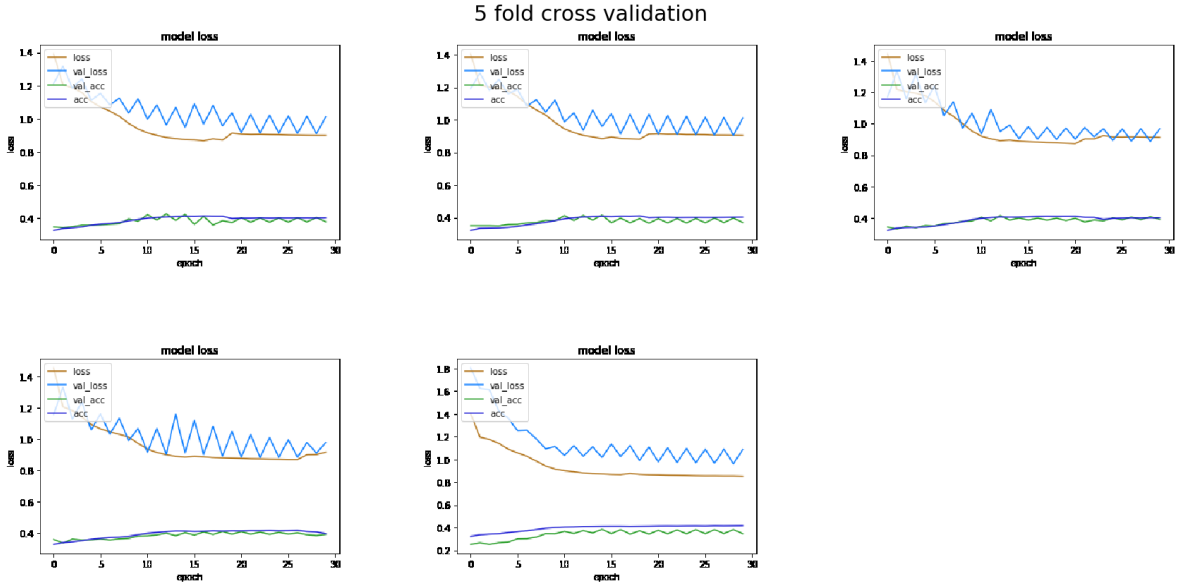

Figure 4.1 Neural network 5- fold cross-validation results

The average MSE for neural network model goes to 1.06, which is better than the baseline model but worse than the SVD and SVD++ models.

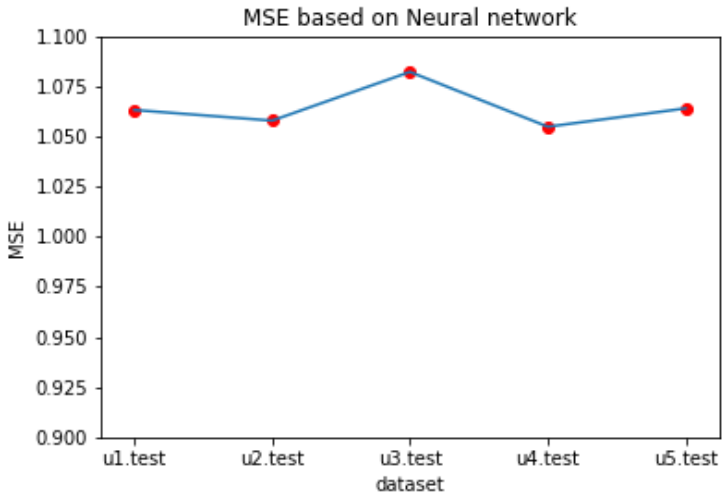

Figure 4.2 Neural network MSE on each test set

Conclusion

In this project, we implemented four different models developed from the idea of collaborative filtering to build a movie recommender system, namely 1: latent factor model using purely SVD of the rating table, as our baseline model; 2: SVD++, which is developed based on the baseline SVD model, taking the implicit feedback into account, i.e., whether the user has given rating to the item; 3: SVD with collaborative filtering model, which aims to enhance user's features when calculating similarity among users or items; 4: Neural network model, which builds embeddings of user's features such as age and occupation, and movie's features such as genre and title, and uses these embedding as the input of a fully connected neural network model to learn the latent factors of users and movies. Note that convolutional layer is also utilized to handle movie titles. After conducting a 5-fold cross-validation experiment, the results of each model are listed below:

Table 1. Average MSE and deviation from ground truth obtained from 5-fold cross-validation

Model	5-fold Average MSE	Deviation from ground truth
Pure SVD (baseline)	1.47	1.21
SVD++	0.84	0.92
SVD with CF based on user features	1.04	1.01
SVD with CF based on user and movie features	0.98	0.99
RS based on Neural network	1.06	1.02

It can be seen from above that SVD++ performed the best under our experiment framework and dataset. All models outperformed the baseline SVD model. It can be concluded that the recommender system using collaborative filtering performs a fairly good result, and it can be improved by taking into account different latent factors and implicit feedback from users.

Future work

To utilize the full potential of the MovieLens dataset, the complete 1M dataset can be used. Particularly, the movie year and rating timestamp can be taken into account by turning the parameters in SVD++ into a function of time, so that the time variant user and movie features can be captured. Tags of movies given by users can also be taken into account in the neural network model, which can better describe the movie features and complement the limit of movie title and genre. In our implementation of neural network, the embeddings of movie genre are summed up. Instead, other aggregation methods can be experimented. Lastly, ensemble learning technique can be used such as combining the result of SVD++ and neural network.

References

1. Yehuda koren, `Factorization Meets the Neighborhood: A Multifaceted Collaborative Filtering Model`, August 24 - 27, 2008
ACM Pages 426-434

2. F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4, Article 19 (December 2015), 19 pages.

3. Y. Kim. Convolutional neural networks for sentence classification. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1746–1751, Doha, Qatar, October 2014. Association for Computational Linguistics.

4. B. Marlin. Collaborative filtering: A machine learning perspective. Master's thesis, University of Toronto, 2004.

5. Nathan Srebro. Learning with Matrix Factorization. PhD thesis, Massachusetts Institute of Technology, 2004.

6. Benjamin Marlin. Modeling user rating profiles for collaborative filtering. In Advances in Neural Information Processing Systems, volume 16, 2004.

Appendix:
Network Architecture