UNIVERSITY OF TECHNOLOGY SYDNEY
SOCIAL AND INFORMATION NETWORK ANALYSIS - AUTUMN 2020

User Attribute Enhanced Movie Recommendation

Yang Liu (13174420) Yuhao Shi (13338239)

In this project, we investigate the influence of user's attributes on movie recommendation. We use two methods, dataset division and deep learning based embedding, to take the user's gender and age into consideration. We conduct experiments with both traditional and deep learning based algorithms on two datasets with different sizes. The quantitative results show that deep learning based embedding can effectively aggregate the multiple attributes, yielding more accurate recommendation. The qualitative experiments show how the gender and age are considered during recommendation.

1 Introduction

The development of the internet has brought a huge number of information for users, however, they cannot quickly obtain the information. Compared with search engines, the recommendation system can meet the personalized demand of users. The recommendation system can be an information filtering system which predicts the 'rating' and 'preference' to recommend for users [21]. An effective recommendation system cannot only provide personalized services for users but also establish a close connect with the users. Over the past decades, a significant increasing number of recommendation systems were implemented in different types of fields and industries, such as music recommendation, movie recommendation, online shopping system and social networking platform [17].

With the increasing number of movies, the recommendation system performs an important role in the movie industry. By modeling the user's preference or watching history, the recommendation system can help the user to find movies they may like without being struggled by

a huge movie set. In the area of movie recommendation, the MovieLens is one of the most important benchmark dataset [13]. This dataset consists of a large number of users' ratings (ranging from 1 to 5) upon movies. It has been used for evaluating not only traditional algorithms, but also recently proposed deep learning based methods [23]. For most of the existing studies, in order to building a general algorithm, minimal domain specific information is introduced. In the MovieLens dataset, while both the movies and users are with rich attributes such as the user's gender, age, and the movie's type, only the users' and movies' IDs are used for recommendation.

In this project, we will study the movie recommendation system based on the MovieLens dataset. Different from previous works where only the IDs are served as inputs, we would like to further investigate the influence of attributes on recommendation. Specifically, we argue that the users with different genders and ages will have different movie preference, therefore taking such attributes into consideration will be beneficial to building an accurate movie recommendation system. To this end, we design two ways to introduce gender and age awareness for traditional recommendation methods and deep learning based methods: 1) dividing the dataset based on attributes, and 2) aggregating these additional attributes with user & movie IDs. We conduct the both quantitative and qualitative experiments on two datasets, MovieLens-100k and MovieLens-1m. The experimental results show that the second method is helpful for performance improvement. In addition, the qualitative results show how the gender and age are considered during recommendation.

2 Related work

The main methods of recommendation systems include but are not limited to collaborative filtering, content-based filtering, hybrid recommendation systems [21, 22]. From the perspective of domain and data type, they can be further divided as multi-criteria recommendation systems, risk-aware recommendation, mobile recommendation, session-based recommendation, etc. Content-based filtering is based on the items which users like or followed to recommend the similar content. For example, if the user watched Harry Potter I, the content-based recommendation algorithm can find Harry Potter II-VI for users. According to the highly relevant content or many keywords to recommend information. The content-based recommendation algorithm can analyze the relationship between the items and then implement the recommendation. The disadvantage is that the recommended items may be repeated.

The principle of Collaborative filtering algorithm is that users like the products which users with similar interests have liked. For instance, if a user's friend like Harry Potter I, then the systems will recommend it for the user. This is the simplest user-based collaborative filtering algorithm (user-based collaborative filtering). Another collaborative filtering algorithm is item-based collaborative filtering. These two methods are reading the users' data into the memory for mathematical operation, then it becomes Memory-based Collaborative Filtering. The collaborative filtering can also be integrated with machine learning techniques to form model-based collaborative filtering, including Aspect Model, pLSA, LDA, clustering, SVD, Matrix Factorization, etc.

In recent years, with the progress of deep learning, it becomes popular to combine recom-

mendation system with neural network approximation [23]. In this way, the id-based inputs can be encoded as higher dimensional latent representations to contain richer information for predicting the ratings. Besides, the DL-based embedding enables the aggregation of different attributes. The deep learning based recommendation system can be divided as two branches: 1) those with neural building blocks, such as multilayer perceptron (MLP) [1, 3, 6], recurrent neural network (RNN) [2, 8, 9], convolutional neural networks (CNN) [7, 14, 15] and attention mechanism (AM) [4, 12], 2) those with deep hybrid models, such as CNN+RNN and AE+CNN [5, 10].

3 Data analysis

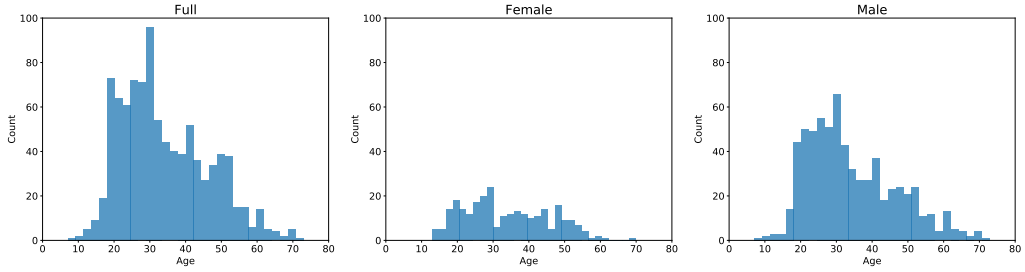

Figure 3.1: The age distribution of all users (“Full”), female users (“Female”) and male users (“Male”) in ml-100k.

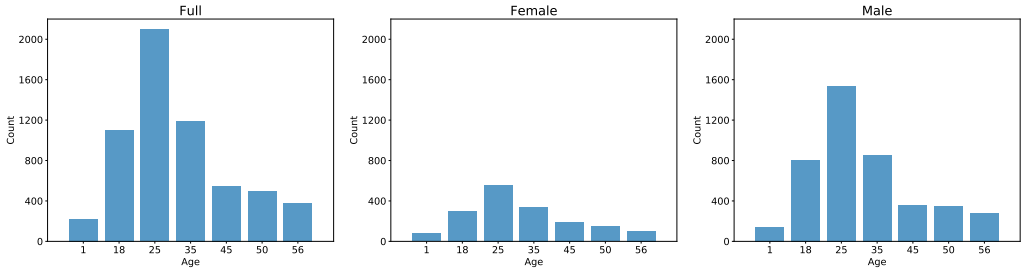

Figure 3.2: The age distribution of all users (“Full”), female users (“Female”) and male users (“Male”) in ml-1m.

In this section we perform data analysis to investigate our first argument: users with different genders and ages will have different movie preference. In this project we use two MovieLens datasets: 1) MovieLens-100k (ml-100k), which contains 100,000 ratings from 943 users over 1682 movies, and 2) MovieLens-1m (ml-1m), which is a much larger dataset containing 1,000,209 ratings from 6040 users over 3883 movies. The user has five attributes: user id, gender, age, occupation and “zip code”. The movie contains its id, the title, the release date, the imdb url and its genres (in “ml-100k” it is represented as multiple one-hot attributes, in “ml-1m” it is represented as concatenated text). The rating has four attributes: user’s id,

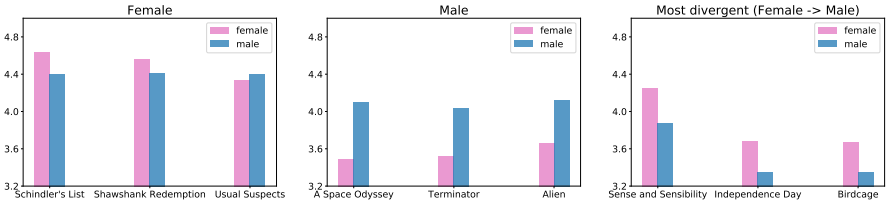

Figure 3.3: The three movies most favored by the female (“Female”), the three movies most favored by the male (“Male”), and the three movies with most significant “female → male” preference difference (more favored by the female but unfavored by the male) in ml-100k dataset.

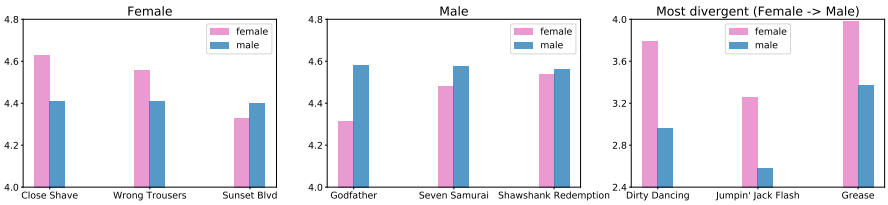

Figure 3.4: The three movies most favored by the female (“Female”), the three movies most favored by the male (“Male”), and the three movies with most significant “female → male” preference difference in ml-1m dataset.

movie’s id, rating and timestamp. Most existing work only consider the user’s id and the movie’s id to predict the rating. Instead, we will additionally take the user’s gender and age into consideration. Other attributes, such as the user’s occupation and the movie’s genres may also be helpful. We would like to leave them as a part of future work.

3.1 Age distribution among genders

We first conduct an overview of the age and gender distribution. Fig. 3.1 shows the age distribution of all users, female users and male users in the ml-100k. The number of male users is much larger than female users. The major age group of participants is 20-50 years old, which also indicates that people around 20-50 year-old tend to go to movies more than others. Similar distribution can be observed in ml-1m, as shown in Fig. 3.2. Note that the ages in this dataset have been classified as 7 age groups.

3.2 Gender difference

We next analysis whether the female users and male users will have significant preference difference on some particular movies. Fig. 3.3 shows the three movies most favored by the female (“Female”), the three movies most favored by the male (“Male”) in ml-100k dataset. It seems that those most favored by the female are also popular for the male users. In contrast,

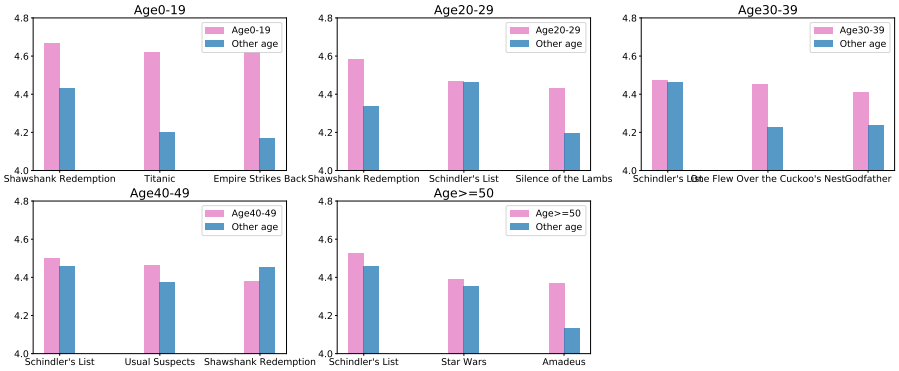

Figure 3.5: ml-100k: The three movies favored by each age group v.s. their average ratings of all other age groups.

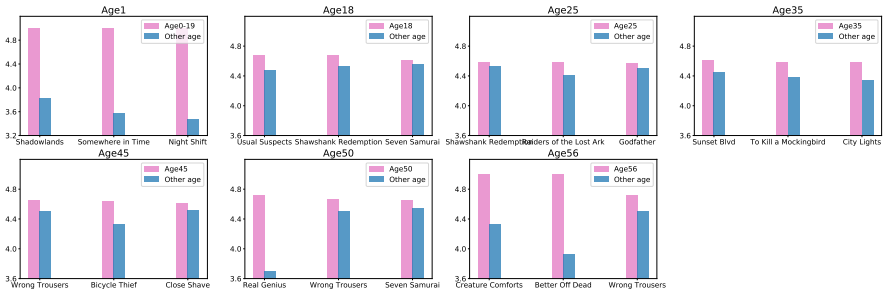

Figure 3.6: ml-1m: The three movies favored by each age group v.s. their average ratings of all other age groups.

those most favored by the male such as “A Space Odyssey”, are with much lower ratings from the female users. We further investigate what kinds of movies are with the most significant preference between the genders. For simplicity we only select those with higher average ratings from the female than from the male (e.g., “A Space Odyssey” is excluded since its rating from the male is higher than female). As shown in Fig. 3.3 “Most divergent”, three movies favored by the female may receive much lower ratings from the male. Fig. 3.4 shows the comparisons in ml-1m dataset. It can be observed that there is significant preference difference on movie “Close Shave”, which is most favored by the female, and “GodFather”, which is most favored by the male. Another movie “Shawshank Redemption” seems to be more general that there’s nearly no difference among the genders.

3.3 Age difference

In this section, we analysis whether the users within different age groups have different preference. For ml-100k, we divide the age as five groups: $\{<20, 20-39, 30-39, 40-49, \geq 50\}$. For each age group, we find the top three movies they like and compare their ratings with the average ratings from all other age groups. Fig. 3.5 shows the comparison results. Similar to the analysis on gender, we found that the users with different age group may also have significantly different preference. For example, the users from 0-19 are crazy about "Titanic" and "Empire Strike Back", while according to the other users, these movies seem not so attractive. Another explanation is that the users in this age group tend to give extreme ratings (i.e. too high or too low), while other users may rate a movie with a more moderate score. For ml-1m, the age has already grouped as 7 groups. As shown in Fig. 3.6, the preference difference caused by age can also be observed.

Based on the initial analysis on the ml-100k and ml-1m dataset, we conclude that our first argument is valid, such that the difference in gender and age group will lead to significant movie preference.

4 Algorithms

10 classic recommending algorithms supported by Surprise[1] are evaluated in this project. Besides, we design a deep learning based architecture with four variants.

4.1 BaselineOnly

This method estimates the baseline for given user and item, then treats the baseline as predicted rating:

$$\hat{r}^{ui} = b_{ui} = \mu + b_u + b_i \tag{4.1}$$

If the user u is a new user, the bias b_u is assumed to be 0. Similar mechanism is applied for item i and its bias b_i.

4.2 SVD

The singular vector decomposition (SVD) [21] is a classic method for recommendation. In linear algebra, SVD is a method of factorization of a real or complex matrix. Based on following equation, a matrix will be decomposed into three other matrices:

$$R_{n\times m} = U_{n\times n} S_{n\times m} M^T_{m\times m} \tag{4.2}$$

By blending the S matrix with U, the remaining two matrices $U_{n\times m}$ and $M^T_{m\times m}$ are learnable. In terms of movie recommendation, these two matrices can be treated as the user matrix with

[1] http://surpriselib.com/

n users and movie matrix with m movies, and their multiplication R denotes the rating matrix. The U and M can be learnt by minimizing the loss:

$$L = \frac{1}{2}\sum_{i=1}^{n}\sum j = 1^{m} I_{ij}(R_{ij} - \hat{r}(U_i, M_j))^2 + \frac{k_u}{2}\sum_{i=1}^{n}||U_i||^2 + \frac{k_m}{2}\sum_{j=1}^{m}||M_j||^2 \tag{4.3}$$

The $\hat{r}(U_i, M_j)$ denotes the predicted rating of movie j made by user i

$$\hat{r}_{um} = \mu + b_u + b_m \tag{4.4}$$

where μ denotes the global mean value of ratings in the training dataset, b_u and b_m denote user's and movie's bias. For more general use, in next sections we redefine movie's bias b_m as item's bias b_i.

4.3 SVD++

The SVD++ algorithm [18] is an extension version of SVD. It is considered to have implicit ratings. The prediction $\hat{r}_{ui}$ is set as:

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T\left(p_u + |I_u|^{-\frac{1}{2}}\sum_{j\in I_u} y_j\right) \tag{4.5}$$

where y_j are a new set of item factors that catch implicit ratings. An implicit rating depicts the fact that a user u rates an item j in spite of the rating value. If u is unknown, the bias b_u and the factors p_u are assumed to be 0. Similar mechanism is applied for item i with b_i, q_i and y_i.

4.4 NMF

The non-negative matrix factorization (NMF) [19] predicts the ratings via:

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T p_u \tag{4.6}$$

The user u's factor p_{uf} and the item i's factor q_{if} are updated via SGD based on follow equations:

$$p_{uf} \leftarrow p_{uf} \cdot \frac{\sum_{i\in I_u} q_{if}\cdot r_{ui}}{\sum_{i\in I_u} q_{if}\cdot \hat{r}_{ui} + \lambda_u |I_u| p_{uf}} \tag{4.7}$$

$$q_{if} \leftarrow q_{if} \cdot \frac{\sum_{u\in U_i} p_{uf}\cdot r_{ui}}{\sum_{u\in U_i} p_{uf}\cdot \hat{r}_{ui} + \lambda_i |U_i| q_{if}} \tag{4.8}$$

4.5 SlopeOne

The SlopeOne [20] is a simple but effective collaborate filtering algorithm totally based on statistics. The main idea of the algorithm is to estimate the current user's score on item i by

the sum of the deviations of all items scored by the current user and the current item i under the same user:

$$\hat{r}_{ui} = \mu_u + \frac{1}{|R_i(u)|} \sum_{j \in R_i(u)} \text{dev}(i, j) \tag{4.9}$$

where $R_i(u)$ denotes the set of items rated by user u, U_{ij} denotes the set of users that rate both item i and item j. $\text{dev}(i, j)$ is the average differences between two items' ratings:

$$\text{dev}(i, j) = \frac{1}{|U_{ij}|} \sum_{u \in U_{ij}} (r_{ui} - r_{uj}) \tag{4.10}$$

4.6 KNN-based methods

The principle of k-nearest neighbour (KNN) is to evaluate an item based on k items most similar to it.

KNNBasic The collaborate filtering can be either user-based or item-based. The score for user u and item i $\hat{r}_{ui}$ is predicted based on the other users' ratings on item i, or user j's ratings on other items:

$$\hat{r}_{ui} = \frac{\sum_{v \in N_i^k(u)} sim(u, v) \cdot r_{vi}}{\sum_{v \in N_i^k(u)} sim(u, v)} \tag{4.11}$$

or

$$\hat{r}_{ui} = \frac{\sum_{j \in N_u^k(i)} sim(i, j) \cdot r_{uj}}{\sum_{j \in N_u^k(i)} sim(i, j)} \tag{4.12}$$

KNNBaseline The KNNBaseline is an extension of KNNBasic with a baseline rating:

$$\hat{r}_{ui} = b_{ui} + \frac{\sum_{v \in N_i^k(u)} sim(u, v) \cdot (r_{vi} - b_{vi})}{\sum_{v \in N_i^k(u)} sim(u, v)} \tag{4.13}$$

or

$$\hat{r}_{ui} = b_{ui} + \frac{\sum_{j \in N_u^k(i)} sim(i, j) \cdot (r_{uj} - b_{uj})}{\sum_{j \in N_u^k(i)} sim(i, j)} \tag{4.14}$$

KNNWithMeans The KNNWithMeans is an special case of KNNBaseline, where the baseline rating is the mean rating. The aim is to reduce the impact of overall high and low reference score.

$$\hat{r}_{ui} = \mu_u + \frac{\sum_{v \in N_i^k(u)} sim(u, v) \cdot (r_{vi} - \mu_v)}{\sum_{v \in N_i^k(u)} sim(u, v)} \tag{4.15}$$

or

$$\hat{r}_{ui} = \mu_i + \frac{\sum_{j \in N_u^k(i)} sim(i, j) \cdot (r_{uj} - \mu_j)}{\sum_{j \in N_u^k(i)} sim(i, j)} \tag{4.16}$$

KNNWithZScore The KNNWithZscore is an extension of KNNWithMeans, where the rating is normalized by z-score:

$$\hat{r}_{ui} = \mu_u + \sigma_u \frac{\sum_{v \in N_i^k(u)} sim(u, v) \cdot (r_{vi} - \mu_v)/\sigma_v}{\sum_{v \in N_i^k(u)} sim(u, v)} \tag{4.17}$$

or

$$\hat{r}_{ui} = \mu_i + \sigma_i \frac{\sum_{j \in N_u^k(i)} sim(i,j) \cdot (r_{uj} - \mu_j)/\sigma_j}{\sum_{j \in N_u^k(i)} sim(i,j)} \tag{4.18}$$

4.7 CoClustering

The main idea of CoClustering [11] is to classify users and items separately, and then perform the predictions based on the cluster center:

$$\hat{r}_{ui} = \overline{C_{ui}} + (\mu_u - \overline{C_u}) + (\mu_i - \overline{C_i}) \tag{4.19}$$

where $\overline{C_{ui}}$ is the average score of co-cluster C_{ui}. $\overline{C_u}$ is the average score of the user cluster, and $\overline{C_i}$ is the average score of item cluster. The clusters can be assigned using KNN.

4.8 DeepLearning

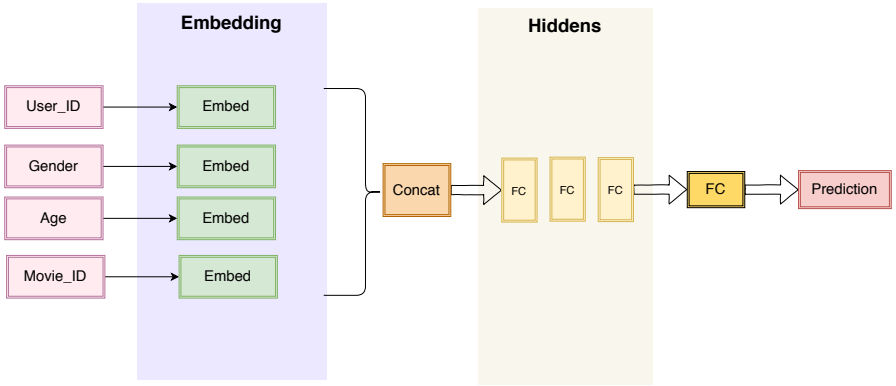

Figure 4.1: The architecture of DL model (“with G, with A”)

The progress of deep learning (DL) techniques enables us to combine multiple attributes for recommendation. Based on recent work such as NCF [16], we design a simple DL model to be compatible with multiple attributes. Fig. 4.1 shows the architecture: different attributes will first be embedded, then concatenated as a combined representation vector. This vector will then be processed by multiple fully-connected (FC) layers. The finally output is the predicted rate ranges from 1 to 5. We denote the model in Fig. 4.1 as “with G, with A” since both the gender (G) and age (A) are considers. Besides this model, we design another three variants:

- “w/o G, w/o A”: only consider user id and item id.
- “with G, w/o A”: consider gender. Age is not considered.
- “w/o G, with A”: consider age. Gender is not considered.

5 Experiments

5.1 Data pre-processing

For ml-100k dataset, we use the provided five training and testing splitation: each training dataset contains 80,000 ratings and each testing dataset contains 20,000 ratings. For ml-1m dataset, we similarly split the dataset as 80% training and 20% testing, and fix these two datasets for all algorithms. The age attribute of ml-100k dataset is grouped into 5 groups: $\{<20, 20-29, 30-39, 40-49, \geq 50\}$. In ml-1m dataset, the age has been grouped as 7 groups.

For traditional algorithms such as SVD, it's hard to directly inject additional attributes, which will result in a high-dimensional matrix. In this project we apply a simple method: we divide both the training and testing dataset by a single attribute, then perform training on each of them. For example, we divide the dataset by gender, then train a "female model" for female dataset, and a "male model" for male dataset. The testing results of multiple datasets will be aggregated for comparison with those trained on the full dataset. For deep learning based algorithms, we just use the full dataset.

5.2 Experiment details

We utilise the Surprise, which is a Python toolkit for building and analysing the recommendation system, to implement the traditional algorithms. We implement the DL-based algorithms via PyTorch, and use following hyper-parameter settings: "n" denotes the embedding dimension for user id and item id. "g" denotes the embedding dimension for gender. "a" denotes the embedding dimension for age. "hidden" denotes the hidden dimension of FC layers in hidden part. The settings shown in the Table 5.1 are for variants with best performance, we also conduct hyper-parameter tuning to compare the their influence.

Table 5.1: The best hyper-parameters for DL-based algorithms

Algorithm	**n**	**g**	**a**	**hidden**
w/o G, w/o A	200	-	-	1000
with G, w/o A	200	25	-	1000
w/o G, with A	150	-	15	1000
with G, with A	200	10	10	1000

We follow the instruction of Surprise toolkit to train the traditional algorithms. The traditional algorithms are trained and evaluated on both ml-100k and ml-1m datasets except SVD++. Due to large time consumption, the experiments of SVD++ are only conducted on ml-100k dataset.

The DL-based algorithms are trained and evaluated on ml-1m dataset. The batch size is set as 2000. We use the sum of mean square error as the loss, and ADAM as optimizer. The learning rate is initialized as $1e-3$ with decay factor $1e-5$. We test the algorithms after each training epoch, if the performance does not keep improving for 10 epochs, we terminate the training process.

5.3 Evaluation metrics

We use root mean square deviation error (RMSE) and recall as evaluation metrics. The RMSE for a dataset X can be defined as:

$$\text{RMSE}(X) = \sqrt{\frac{\sum_{t=1}^{n}(\hat{y}_t - y_t)^2}{n}} \tag{5.1}$$

where n is the number of samples in X The recall can be defined as the number of true positives (TP) over the number of true positives plus the number of false negatives (FN):

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{5.2}$$

In this project, we predict the top 10 movies for each user, and set those rated with >= 3.5 as positive samples. For a dataset X, the recall value is averaged by the total number of samples n.

For traditional methods, as we split the dataset by an attribute (gender or age), we need to combine the evaluation result. Suppose we have K datasets $\{X_1, X_2, ..., X_K\}$, the combined evaluation metrics will be computed as:

$$\text{RMSE}(X_1, X_2, ..., X_K) = \sqrt{\frac{\sum_{i=1}^{K}\text{RMSE}(X_i)^2 \times n_i}{\sum_{i=1}^{K} n_i}} \tag{5.3}$$

$$\text{Recall}(X_1, X_2, ..., X_K) = \frac{\sum_{i=1}^{K}\text{Recall}(X_i) \times n_i}{\sum_{i=1}^{K} n_i} \tag{5.4}$$

where n_i denotes the number of samples in dataset X_i. Lower RMSE and higher recall denote better performance.

For ml-100k, we report the average result across 5 testing datasets (similar to 5-fold cross validation). For ml-1m, we fix the training dataset and testing dataset, and report the performance on the testing dataset.

6 Results and Discussions

6.1 Traditional algorithms

Fig. 6.1 shows the RMSE of traditional algorithms on ml-100k. Among all algorithms, SVD++ achieves the lowest RMSE, while it requires the longest training time. It can be observed that the dataset division leads to decreased performance. The ml-100k dataset is a relatively small dataset that it only contains hundreds of users. While the training dataset becomes smaller, the user-item matrix would be too sparse to complete accurately. From the perspective of machine learning, too few training samples will cause under fitting. The performance of dataset division seems better in the larger dataset: ml-1m. As shown in Fig. 6.2, when considering female and male separately ("Gender"), the resulted performance is similar to training on the full

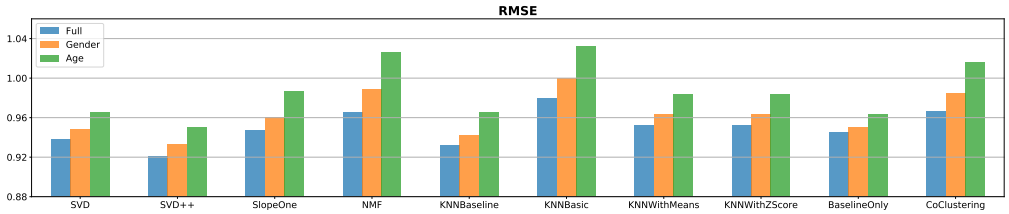

Figure 6.1: The performance of traditional algorithms on ml-100k.

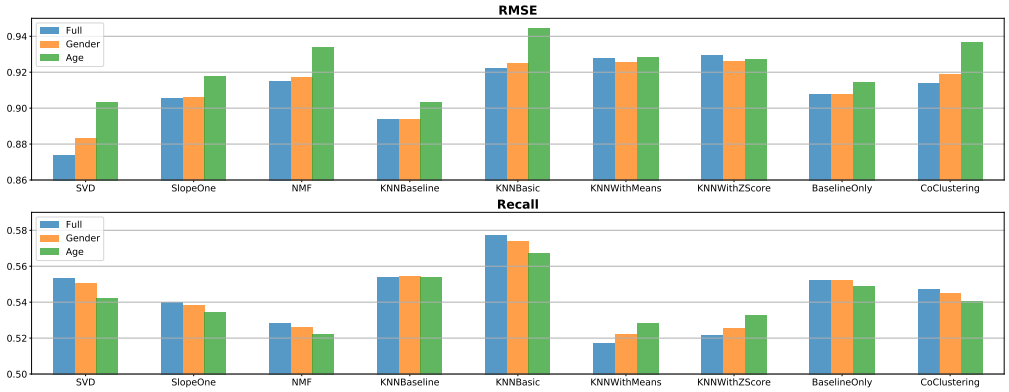

Figure 6.2: The performance of traditional algorithms on ml-1m.

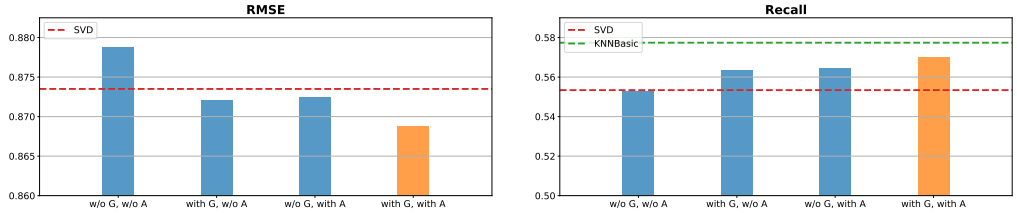

Figure 6.3: The performance of DL-based algorithms on ml-1m. The dashed lines show the result of best-performed traditional algorithms.

dataset ("Full") in most of the algorithms. For algorithm "KNNWithMeans", the performance of "Gender" is slightly better than "Full". However, we conclude that the dataset division is not a suitable method to introduce additional attributes. It reduces the dataset size, which may lead to under-fitting and sparse matrix, making its performance worse than training on the full dataset. Furthermore, this method lacks adaptability, especially when there are multiple attributes to be considered. As a result, we have to either consider each attribute solely, or perform a tree-like division and lead to a set of much smaller subsets.

6.2 DL-based algorithms

We next investigate whether neural network approximation can result in better aggregation of multiple attributes. Fig. 6.3 shows the performance of 4 DL-based variants on ml-1m. The introduce of additional attributes lead to performance improvement, and the variant who considers both gender and age achieves the best performance ("with G, with A", in orange color). We also compare the DL-based algorithms with the performance of traditional algorithms analyzed in the last section. When neither gender nor age is considered, the DL-based algorithm ("w/o G, w/o A") is worse than the best performed SVD algorithm (red dashed line), while other three variants with additional attributes achieve lower RMSE than SVD. Similar phenomenon can be observed in terms of recall, "with G, w/o A", "w/o G, with A", "with G, with A" are with higher recall than SVD. The highest recall is achieved by KNNBasic (without dataset division). However, according to Fig. 6.2 its RMSE is higher than 0.92, which is much worse than DL-based algorithms.

For DL-based algorithms, the hyper-parameter tuning plays an important role for performance improving. We perform rough parameter tuning process that we change the dimension of four components: the user & movie embeddings ("n"), the gender embeddings ("g"), the age embeddings ("a") and the hidden layers ("hidden"). To investigate their influence, we compare the DL-algorithms with different hyper-parameter settings. Fig. 6.4 shows the result. Increase the hidden dimension ("hidden") is helpful to improve the performance that those with 1000 hidden dimension tend to have lower RMSE. In contrast, 200 seems to be sufficient to characterize the users and movies. As there are only two groups of genders and seven groups of ages, their the embedding dimension should be much lower.

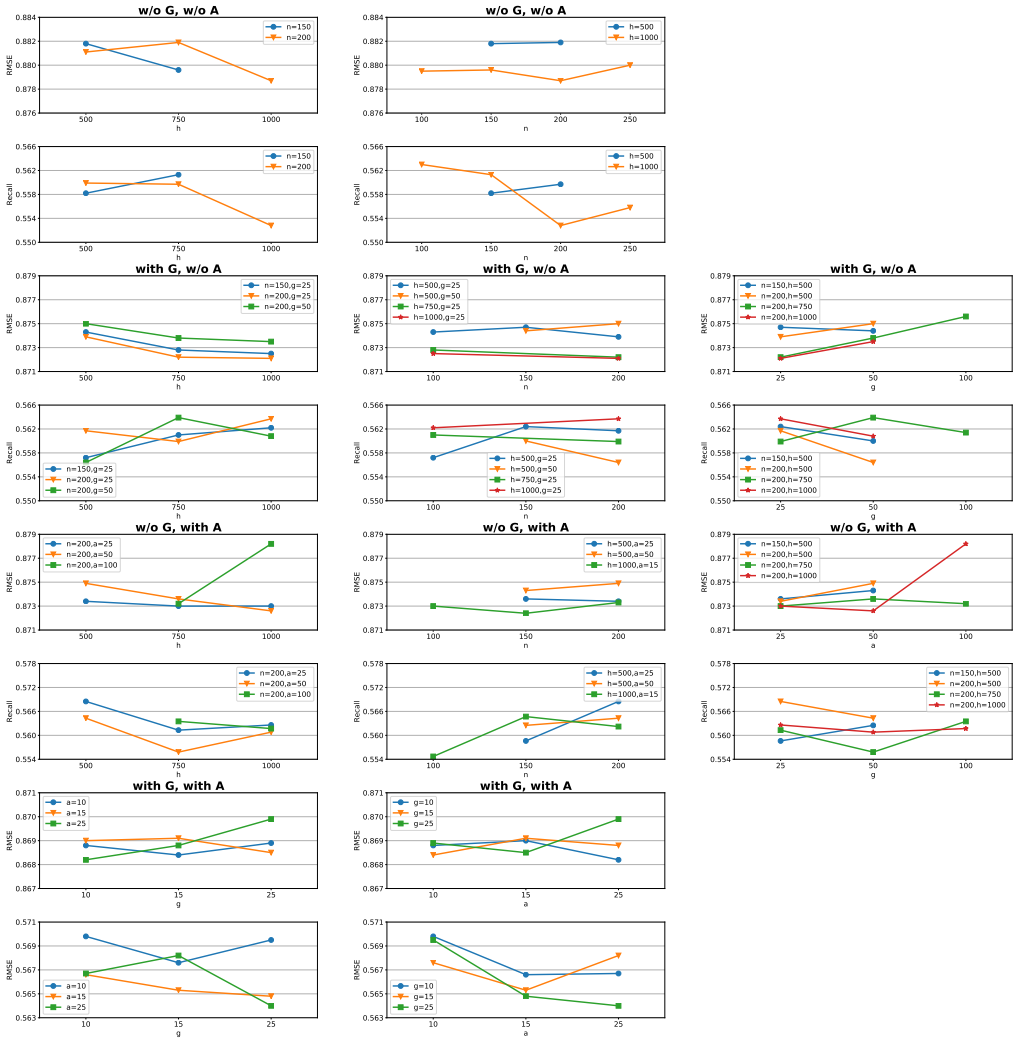

Figure 6.4: The performance of DL-based algorithms with different hyper-parameter settings.

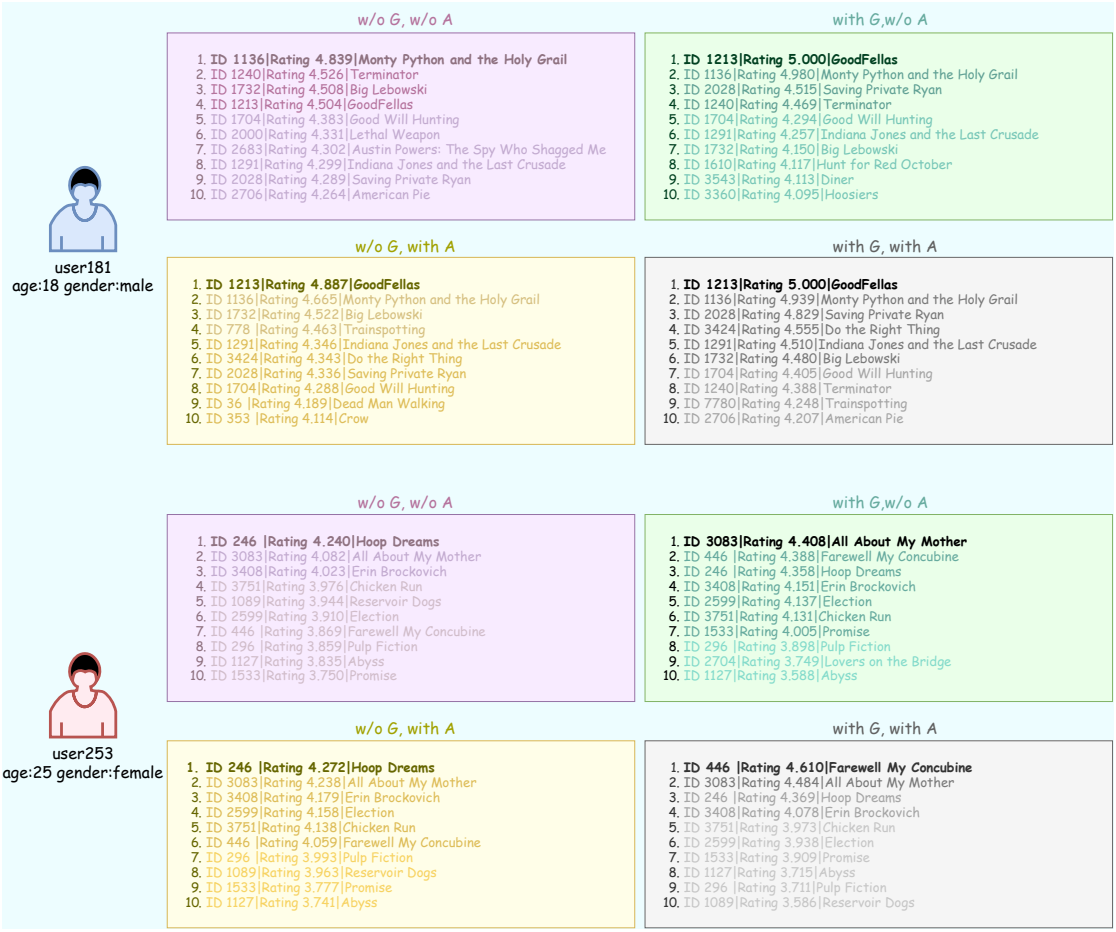

Figure 6.5: The qualitative analysis.

6.3 Qualitative analysis

Besides the quantitative comparisons, we perform qualitative experiments on DL-based algorithms to analysis their difference in recommendation. We select two users: user 181, who is a 18-age-group male, and user 253, who is a 25-age-group female. Fig. 6.5 visualizes the top 10 recommendation made by four DL-based variants. In terms of user 181, we found that when taking gender and/or age into consideration, the movie "GoodFellas" is with the highest recommendation. In "w/o G, w/o A" this movie is at the fourth position. The reason may be that this movie is favored by male users or younger users. As a result, these with additional attributes have more accurate recommendation. Another movie "American Pie", appears in "w/o G, w/o A" and "with G, with A". The reason may be that this movie may be less popular for female in age group 18 or male in other age groups, therefore considering single attribute will make this movie not be recommended.

In terms of user 253, we found that the rating movie "Farewell My Concubine" given by gender-related algorithms ("with G, w/o A" and "with G, with A") and gender-unrelated algo-

rithms (“w/o G, w/o A” and “w/o G, with A”) are rather different. It seems that this movie As illustrated in the Section 3.2, this movie may be with significant gender difference that it is favored by female users but unfavored by male users. Another example is the movie “Reservoir Dogs”, which is the 5th recommendation in algorithm “w/o G, w/o A”. It is less favored by female users (“with G, w/o A”, “with G, with A”). Regarding the age, it seems this movie is favored by users in this age group, but not popular. To conclude, this movie is more likely to be recommended to male users in this age group.

7 Conclusion

In this project, we evaluated the MovieLens dataset and tried to predict what kind of movie a person most like. We argued that people with different genders and ages might have different preference, therefore taking these attributes into consideration would be helpful for accurately recommending. We proposed two methods, dataset division for traditional algorithms and deep learning based embedding for DL-based algorithms. The experimental results shown that dataset division leads to performance decrease, while for DL-based algorithms, introducing additional attributes results in better performance.

With respect to the future works, we are not able to show the performance of SVD++ in the ml-1m dataset. In the future work we would like to compare it with DL-based algorithms. Another direction is to consider more attributes, such as user’s occupation, and movie’s genres. For DL-based algorithms, more parameter tuning steps can be performed to obtain better performance.

References

[1] Taleb Alashkar, Songyao Jiang, Shuyang Wang, and Yun Fu. Examples-rules guided deep neural network for makeup recommendation. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[2] Trapit Bansal, David Belanger, and Andrew McCallum. Ask the gru: Multi-task learning for deep text recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 107–114, 2016.

[3] Cen Chen, Peilin Zhao, Longfei Li, Jun Zhou, Xiaolong Li, and Minghui Qiu. Locally connected deep learning framework for industrial-scale recommender systems. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 769–770, 2017.

[4] Jingyuan Chen, Hanwang Zhang, Xiangnan He, Liqiang Nie, Wei Liu, and Tat-Seng Chua. Attentive collaborative filtering: Multimedia recommendation with item-and component-level attention. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 335–344, 2017.

[5] Xu Chen, Yongfeng Zhang, Qingyao Ai, Hongteng Xu, Junchi Yan, and Zheng Qin. Personalized key frame recommendation. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 315–324, 2017.

[6] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*, pages 7–10, 2016.

[7] Wei-Ta Chu and Ya-Lun Tsai. A hybrid recommendation system considering visual information for predicting favorite restaurants. *World Wide Web*, 20(6):1313–1331, 2017.

[8] Hanjun Dai, Yichen Wang, Rakshit Trivedi, and Le Song. Deep coevolutionary network: Embedding user and item features for recommendation. *arXiv preprint arXiv:1609.03675*, 2016.

[9] Tim Donkers, Benedikt Loepp, and Jürgen Ziegler. Sequential user-based recurrent neural network recommendations. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*, pages 152–160, 2017.

[10] Travis Ebesu and Yi Fang. Neural citation network for context-aware citation recommendation. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 1093–1096, 2017.

[11] Thomas George and Srujana Merugu. A scalable collaborative filtering framework based on co-clustering. In *Fifth IEEE International Conference on Data Mining (ICDM'05)*, pages 4–pp. IEEE, 2005.

[12] Yuyun Gong and Qi Zhang. Hashtag recommendation using attention-based convolutional neural network. In *IJCAI*, pages 2782–2788, 2016.

[13] F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4):1–19, 2015.

[14] Ruining He and Julian McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceedings of the 25th international conference on world wide web*, pages 507–517, 2016.

[15] Ruining He and Julian McAuley. Vbpr: visual bayesian personalized ranking from implicit feedback. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[16] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*, pages 173–182, 2017.

[17] Kartik Narendra Jain, Vikrant Kumar, Praveen Kumar, and Tanupriya Choudhury. Movie recommendation system: hybrid information filtering system. In *Intelligent Computing and Information and Communication*, pages 677–686. Springer, 2018.

[18] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434, 2008.

[19] Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

[20] Daniel Lemire and Anna Maclachlan. Slope one predictors for online rating-based collaborative filtering. In *Proceedings of the 2005 SIAM International Conference on Data Mining*, pages 471–475. SIAM, 2005.

[21] Francesco Ricci, Lior Rokach, and Bracha Shapira. Introduction to recommender systems handbook. In *Recommender systems handbook*, pages 1–35. Springer, 2011.

[22] Lalita Sharma and Anju Gera. A survey of recommendation system: Research challenges. *International Journal of Engineering Trends and Technology (IJETT)*, 4(5):1989–1992, 2013.

[23] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys (CSUR)*, 52(1):1–38, 2019.