A Hybrid Movie Recommendation System Design Based on Neural Collaborative Filtering and Traditional Methods

Ruokun He
School of Electronic Information and Electrical Engineering
Email: hh15013075@sjtu.edu.cn

Donglin Yang
School of Electronic Information and Electrical Engineering
Email: ydljaccount@sjtu.edu.cn

Zhilin Zeng
School of Electronic Information and Electrical Engineering
Email: bernardeschi@sjtu.edu.cn

***Abstract*—We design a hybrid recommending system of movies in this project. We employ our work on MovieLens dataset and extend it with information from internet. Main method used in the system is the Neural Collaborative Filtering [1], while traditional collaborative filtering method based on user similarity is used to construct a parallel hybrid recommendation system. Some other methods are also implemented for comparison.**

1. Introduction

For all kinds of vedio sites, effectively recommending movies and vedios that meet interests of users, can increase website traffic and user stickiness. Traditional recommending method based on clustering, matrix factorization and nearest neighbor ideas, have achieved good result on many recommending tasks [2]. However, since the great success of deep learning in computer vision, natural language processing and other fields, recommending system also benefit from deep learning methods. Actually, today's state-of-the-art recommender systems, such as those from Youtube and Amazon, are driven by sophisticated deep learning systems rather than traditional methods [3].

But traditional method still has unique advantages, such as fast algorithm, stable performance, needless of large-scale data for training, etc. Besides, there are some differences between traditional methods and deep learning methods in terms of recommendation. Generally speaking, traditional methods can draw a global shape of data while deep learning method can extract more fine-grained information. This gives a chance to hybrid recommending system combining both traditional and deep learning method to construct a better recommending system.

MovieLens is a wildly used dataset for movie recommending tasks with detailed rating data from the MovieLens web site. We will employ our work on the latest small-scale version dataset.

2. Methods

We implement three traditional recommending methods including User Based CF(collaborative filtering), Item Based CF, Matrix Factorization and one deep learning recommending method, NCF with MLP layers. Then, to utilize both strength of traditional and deep learning methods, we construct a hybrid system that combines the results of two methods by weighted combination.

2.1. User Based CF

We consider the idea of collaborative filtering based on user information, that users with similar interests on movies they have seen should have similar interests on unseen movies. First we evaluate user similar by cosine similarity of their rating vectors, we represent user as u, v, there rating vector of all movies as R_u, R_v

$$sim(u.v) = \frac{R_u \cdot R_v}{||R_u|| \cdot ||R_v||}$$

users who have seen the same movies and rating similar scores will have higher similarity. Then to recommend movies to a specific user u, we predict his rating on unseen movies by the ratings from similar users of u. Consider top k similar user of u: $v_1, v_2, ;v_k$, the rating of u on his unseen movie m is:

$$r(u,m) = \frac{\sum_{i=1}^{k} sim(u, v_i) \cdot r(v_i, m) \cdot \mathbb{1}[r(v_i, m) > 0]}{\sum_{i=1}^{k} sim(u, v_i) \cdot \mathbb{1}[r(v_i, m) > 0]}$$

Here the indicator function $\mathbb{1}[r(v_i, m)]$ exclude users who have not seen the target movie, otherwise there will be bias as popularity of the movie is associated. The movie with highest rating is recommended.

2.2. Item Based CF

Item Based CF do recommendation on similarity between items, which is based on the idea that, user should like movies similar to those he has seen. We think the similarity between movie is mainly defined by their types, and actor, director, editor information should be an additional points. Assume movies m, n and their type vector as G_m, G_n,

TABLE 1. **PERFORMANCE COMPARISON**

Model	HR@5	HR@10	NDCG@5	NDCG@10
User Based CF	0.4033	0.5639	0.2643	0.3164
Item Based CF	0.1311	0.2049	0.08	0.1080
FunkSVD	0.2475	0.3704	0.1594	0.2073
NCF	0.4492	0.6082	0.3093	0.3589
NCF+User Based CF	0.5377	0.6541	0.3890	0.4317
NCF+type feature+User Based CF	**0.5443**	**0.6754**	**0.4076**	**0.4454**

which is a indicator vector of all types where the corresponding member is 1 if the movie has such type otherwise 0. The similarity is defined using cosine similarity:

$$sim(m,n) = \frac{G_m \cdot G_n}{||G_m|| \cdot ||G_n||}$$

Besides, for each of directors, editors, actors, if two movie has intersection on one , the similarity is plus 0.05.

Then, for specific user u, assume the movie he has seen is $m_1, m_2, \ldots, m_k$, then for movie n, we calculate the mean similarity of n between m_1 to m_k as the score of u on movie n:

$$score(u,n) = \frac{1}{k}\sum_{i=1}^{k} sim(m_i, n)$$

The movie with highest score is recommended.

2.3. FunkSVD

FunkSVD is first proposed in the Netflix Prize competition where it outperforms many traditional methods. The difference of FunkSVD between original SVD is that it abandoned matrix factorization based on eigenvalue, instead it uses two matrix which project rating matrix to low-dimension space, and fit the full rating matrix by gradient descent. Though the idea is simple, the performance is excellent.

We organize the dataset as a user-item matrix where the member is the rating, whose shape is $N \times N$. Then we define two project matrix $P_{N\times K}, Q_{K\times M}$, where the low-dimension size K need to be specified. To fit the dataset, we compute $R = PQ$ as the rating matrix, and for all user-item pair (u_i, m_j) in the dataset, the predicted rating is $R_{i,j}$ and the ground truth rating is $r(u_i, m_j)$, compute the MSE loss:

$$loss = \frac{1}{N}\sum_{(i,j)} (r(u_i, m_i) - R_{ij})^2$$

where N is the number of user-item pairs. The factorization is optimized by gradient descent method. Then we get the predicted rating of unseen movies for each user, and the recommending is make based on the rating matrix.

2.4. Neural Collaborative Factorization

A unique operation of NCF is the implicit representation of the dataset. That is, instead of using rating of user-item pair, NCF only consider whether user u and item i has interaction, that is:

$$y_{ui} = \begin{cases} 1, \text{if (u,i) is in the dataset} \\ 0, \text{otherwise} \end{cases} \Gamma$$

The dataset if transformed into a binary matrix $\mathcal{Y} \in U \times I$ where U, I are the user and item set. The task is then predicting the probability of interaction in user-item pair (u, i). The author of NCF combine GMF and MLP on the NCF framework, but here we use MLP layers only. The network takes user and item ont-hot vector as input. Then a embedding will be done on user and item vector, the output is then concatenated as a user-item vector. Then it is put into a MLP net, whose output will finally pass a sigmoid layer to produce the probability of the interaction between the user and item. Fig.1 shows the structure of NCF.

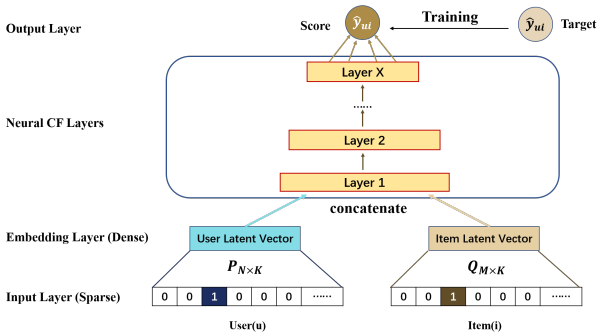

Figure 1. Framework of Neural Collaborative Filtering

The task is formulated as

$$\hat{y_{ui}} = f(P^T v_u^U, Q^T v_i^I | P, Q, \Theta_f)$$

However, not all user-item pairs that does not appear in the dataset should not be seen. So a negatie sample rate $r_negative$ should be specified, then for each (u, i) in positive sample set $\mathcal{Y}$, we randomly choose $r_negative$ negative user-item pair samples that does not appear in the positive sample set to add to the negative sample set $\overline{\mathcal{Y}}$ where $P \in \mathbb{R}^{N\times K}$ and $Q \in \mathbb{R}^{M\times K}$, denoting the embedding matrix for users and items respectively, and Θ_f denote the MLP layers parameters, represented by f. To learn model parameters, we perform optimization with binary cross entropy loss:

$$L_{bce} = \sum_{(u,i)\in\mathcal{Y}\cup\overline{\mathcal{Y}}} -(y_{ui}\log\hat{y}_{ui} + (1-y_{ui})\log(1-\hat{y}_{ui}))$$

2.5. Hybrid Neural Collaborative Filtering

A. Intuition

From implementing results of related methods, we some difference between traditional and deep learning methods. During traditional methods, User Based CF preforms the best(shown later); and for those correctly predicted user-item pairs, we find the target item is always a popular movie, that is, more than 10% user in dataset have ever seen it. For unpopular movies, always it cannot give right prediction. So we think User Based CF implicitly group users according to their common interests on popular movies. Instead, NCF performs better on unpopular movie recommending, which means NCF extract user-specific features by embedding of user and items. A natural idea is combining both of them to give better recommending.

B. Framework

Similar to Ensemble Learning in machine learning, hybrid recommending system has been widely used to improve performance both on traditional method [4] and deep learning models [5]. We construct our hybrid recommending system by a weighted score on User Based CF and NCF output. For User Based NCF, we normalize ratings to $0 \sim 1$ to fit NCF output format. Furthermore, taking into consider that movie type is a typical feature for movies, we try combining it into user embedding layers to gain a multi-embedding layer. Fig.2 shows the system framework.

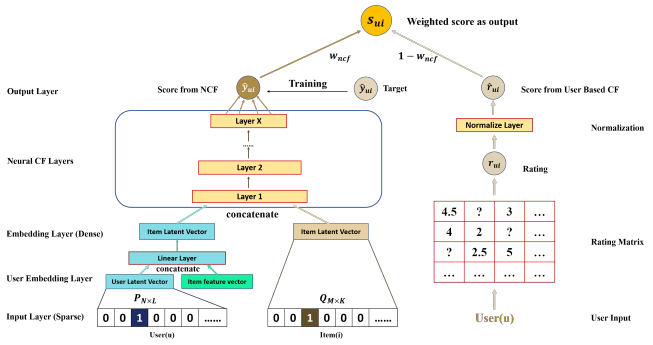

Figure 2. Framework of Hybrid Neural Collaborative Filtering

C. Task Formulation

The task is formulated as:

$$s_{ui} = w * f(\phi([P^T v_u^U, G_i]), Q^T v_i^I | P, Q, \Theta_f) + (1-w) * h(r_{ui})$$

For new symbols in the formulation, s_{ui} represents the final score for (u, i) pair, w denotes weight of NCF weight, G_i is the type vector for item i, r_{ui} is the rating of User Based CF on (u, i), h is the normalize function which normalize rating of (u, i) by dividing the max rating of user u on all unseen movies. The training method for NCF is similar to the former part.

3. Experiments

In this section, we conduct experiments to make comparison of performance of each model and present a evaluation of our hybrid NCF model.

3.1. Evaluation Setting

Metrics

To evaluate the performance of our recommending system, we adopt the leave-one-out evaluate strategy. That is, for each user, we use the latest interaction information as a evaluation, and others as training dataest. Besides, we follow the common strategy that randomly sample 100 items that have not interacted with the user, make evluation on the 100 item sets. The performance is evaluated using **Hit Ratio(HR)** and **Normalized Discounted Cumulative Gain(NDCG)**. HR@K measures whether the test item is present on the top-k list, and the NDCG accounts for the position of the hit by assigning higher score to hit at top ranks. We calculate both metric each user and report the mean value.

Dataset

We experiment with a widely used dataset: MovieLens. This movie rating dataset has been widely used to evaluate collaborative algorithms. We used the version containing 100,000 ratings.

TABLE 2. STATISTICS OF THE EVALUATEION DATASET

Dataset	Interaction#	Item#	User#	Sparsity
MovieLens	100,836	9,724	610	98.30%

3.2. Performance Comparison

We set low-dimension size 5 in funkSVD, similar users 20 in User Based CF, embedding size 16 and negative sample ratio 8 in NCF and hybrid NCF for performance comparison. Table.2 shows the performance of HR@5, HR@10, NDCG@5, NDCG@10 on all models. Among all traditional methods, User Based CF performs the best.

Item Based CF works the worse, which is not surprising, as only rating information is within the dataset, as for movie information, we only crawl type, actor, director and editor information of the movie. Among all these features only movie types is typical, but still not representative enough for a movie, so the recommending system based on it is not good, but type feature can be merged into other recommending system to be a complementary of movie feature. For User Based CF, we find that the correct recommending is always popular movies, which means User Based CF actually catch user interest on popular movies and group them by it. As for FunkSVD, it project user and item feature to low-dimension space. As the rating matrix is given, so both User Based CF and FunkSVD works well in feature extracting.

NCF utilize the strength of deep learning framework on feature extraction, and it do works better than all traditional methods. But the analysis in section 3, the feature from NCF adn User Based CF are actually from different 'perspective', so we can combine then to gain better performance. As we can see, out hybrid system has a considerable improvement

compared with the NCF model. Furthermore, the merge of type feature also has a little improvement on the model.

Fig.3 shows performance of Hybrid NCF HR and NDCG with different embedding size. Different embedding size performances are similar, while embedding 16 performs the best among all settings.

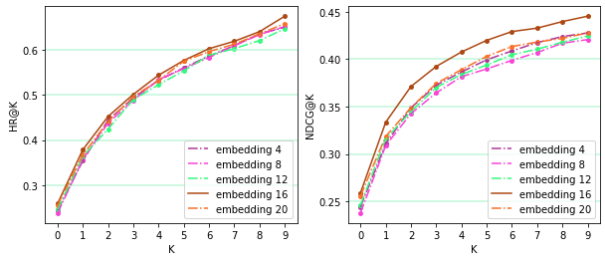

Figure 3. Evaluation of Top-K item recommendation

4. Conclusion

In this work, we test multi models containing both traditional and deep learning methods on recommending systems. We see that deep learning do perform better on our tasks than all the traditional methods. However, by analyzing result of the recommending, we find traditional and deep learning methods have different strength in global grouping for user and specific feature extraction, respectively. Therefore, we construct a hybrid recommending system combining User Based CF and NCF framework, taking movie type feature into consideration. As a result, the new model outperform all other models, showing the combination is successful.

For further improvement, a possible direction is to explore more effective feature extracting deep learning networks to extract user-specific information. Another idea is to combine matrix factoring idea into NCF framework to gain new feature from different perspective.

References

[1] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proceedings of the 26th international conference on world wide web*, 2017, pp. 173–182.

[2] X. Su and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques," *Advances in artificial intelligence*, vol. 2009, 2009.

[3] M. Naumov, D. Mudigere, H.-J. M. Shi, J. Huang, N. Sundaraman, J. Park, X. Wang, U. Gupta, C.-J. Wu, A. G. Azzolini *et al.*, "Deep learning recommendation model for personalization and recommendation systems," *arXiv preprint arXiv:1906.00091*, 2019.

[4] S. Bostandjiev, J. O'Donovan, and T. Höllerer, "Tasteweights: a visual interactive hybrid recommender system," in *Proceedings of the sixth ACM conference on Recommender systems*, 2012, pp. 35–42.

[5] T. K. Paradarami, N. D. Bastian, and J. L. Wightman, "A hybrid recommender system using artificial neural networks," *Expert Systems with Applications*, vol. 83, pp. 300–313, 2017.