Advances in Engineering Research, volume 166
3rd International Conference on Automation, Mechanical Control and Computational Engineering (AMCCE 2018)

Recommendation Algorithm Based on Restricted Boltzmann Machine and Item Type

Fan He[a], Na Li[b,*] and Zhi-gang Zhang[c]
University of Science and Technologh Beijing, Beijing 100083, China
[a]beikehf_2012@163.com, [b]lena@ustb.edu.cn, [c]zggcyf@263.net

Keywords: Restricted Boltzmann Machine, collaborative filtering, recommendation system, item type.

Abstract. Because of the sparsity of the ratings in the recommendation system, the calculation of the neighbors will be affected. The common method is to predict the missing ratings and calculate the neighbors with the prediction ratings. However, due to the deviation between prediction ratings and true ratings, it will also lead to the inaccuracy of nearest neighbors. In order to solve this problem, we use RBM to predict the missing ratings. Considering that the type or label of the item has certain influence on the rating, we introduce the type similarity of the item to modify the original neighbors. So that we get the neighbors which is closer to the target user. In this paper, the new model is applied to the MovieLens data set. The result shows that the results of the new model are better than collaborative filtering based on RBM and collaborative filtering based on SVD.

1. Introduction

Since the recommendation system was proposed, its development is very fast, and now it has been applied to e-commerce, search engines, sociality and many other fields. They recommend different items for different users by analyzing their interests, preferences, and behavior characteristics. Among many recommendation algorithms, collaborative filtering is widely used because of its easy to understand, easy to implement and high recommendation accuracy. So it has become one of the mainstream algorithms. Collaborative filtering has been improved during its development. There are many different collaborative filtering in different directions, which can be applied to different forms of data sets and fields.

Collaborative filtering is to calculate the similarity among users or items, and recommend them to the target user through the relationship between the neighbors. Collaborative filtering is divided into memory based collaborative filtering and model based collaborative filtering. Among them, model based collaborative filtering is widely used and we need to apply model technology to the algorithms. We establish a suitable model according to the specific recommendation system, then load the established model into memory when the recommendation algorithm is running. A good model can effectively solve the shortcomings of collaborative filtering algorithm, but the establishment of the model is very time-consuming, and it must be carried offline, and the latest data can not be used effectively. Because of the large number of users and items in the system, only a few users rate a very small number of items, so there is a problem of data sparseness. To solve this problem, a variety of machine learning algorithms are used to predict missing ratings. Singular value decomposition (SVD) is one of the most widely used algorithms [1].

The proposal of deep belief network (DBN) indicates the emergence of deep learning [2]. With the development of deep learning [3], it has been applied to many fields, and has achieved great success in image recognition [4], speech recognition [5], natural language processing [6] and so on. Subsequently, deep learning are also used in recommender systems [7]. As early as 2007, Ruslan Salakhutdinov [8] first applied the Restricted Boltzmann Machine (RBM) to the prediction of ratings in recommender systems. The accuracy of this algorithm was higher than that of SVD. Unlike traditional RBM, this algorithm uses different RBM for different users, and each user has a separate RBM that shares the same hidden layer with RBM. Although each RBM has only one training case,

the weights and biases among all RBM are shared. So if two users rate the same item, then two RBM share the same weight and bias. In addition, the visible layer uses the 0-1 vector of length K to represent the ratings (K is the range of the rating data), and the item without users' rating will be processed into a special missing unit and it is not connected to any hidden layer.

In addition, Sedhain (2015) [9] combined autoencoder and recommendation algorithm. In this paper, the algorithm is composed of multilayer autoencoder, and the output of former autoencoder is the input of the latter layer. Adding noise to the original data allows the network to be more robust against noise, thus extracting more efficient content information and then tagging it. Liu (2014) [10] added the information of users, items and other related contents in the RBM, and combined it with the naive Bayes method to predict the missing data in the rating matrix. Yu (2003) [11] improved the problem of non weight in traditional collaborative filtering, and set weights for neighbors and items. Liu (2017) [12] proposed a new method for calculating the similarity, which found the target user's neighbors more accurately. Then, the clustering algorithm was used to put similar users in a class. Collaborative filtering algorithms were run only within the class of the user's own class. Zhao (2016) [13] added deep belief network to recommendation system. And Zhao combined collaborative filtering algorithm and K nearest neighbor two methods to solve the data sparse problem; Zhao first used deep belief network to extract the user characteristics, and then used the K nearest neighbor method to predict.

2. Algorithm Description

The Restricted Boltzmann Machines for collaborative filtering transforms the integer ratings into a matrix with a value of 0-1, and fills the integer ratings matrix with the RBM. This solves the problem of the sparsity of the rating matrix. But the training time is too long due to too many training parameters and complicated training process. Therefore, on the basis of the Georgiev[14], this paper uses the Real_value Restricted Boltzmann Machine (R_RBM) to fill the missing ratings. It not only shortens the training time, but also applicable to all kinds of data types.

This section introduces the basic idea of the new model in this article: first, a Real_value Restricted Boltzmann Machine model is trained by rating matrix, and then the neighbors is modified according to the item type similarity among the users. Finally, the predictive ratings are calculated by the collaborative filtering.

2.1 Training Real_value Restricted Boltzmann Machine

Similar to the RBM, the Real_value Restricted Boltzmann Machine is an undirected graphical model. R_RBM consists of two layers, namely the visible layer and the hidden layer. The visible layer is also the input layer, which represents the input data, and the hidden layer is the feature extraction layer. The figure is shown below.

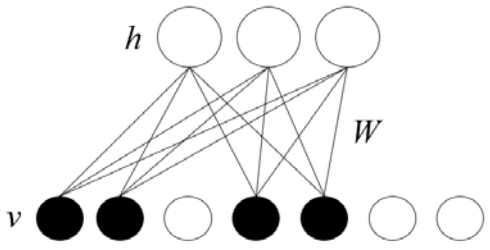

Fig.1 R_RBM model

In figure 1, *v* represents visible layer. Among them, the solid circle represents the item that user has evaluated, that is, the unit has input; the hollow circle represents the item that user has not evaluated, that is, the unit has no input. *h* represents hidden layer. Each unit in the hidden layer is independent of each other. The units in the hidden layer only connect with the input units in the visible layer. *W* represents the weight of the connection between the visible layer and the hidden layer.

Due to most of the ratings are missing, we use a different R_RBM for each user. Because each user has different evaluation items, the number of visible units in each R_RBM is different. But all

R_RBM have the same number of hidden layer units. If a user evaluated few items, the number of visible units in the R_RBM is also very small.

Every R_RBM only has a single training case, but all of the corresponding weights and biases are tied together. So, if two users have rated the same movie, their two R_RBMs must use the same weights between the visible unit for that movie and the hidden units.

As the input layer, the visible layer represents the rating of target user. The visible layer is initialized to the real number between 0 and 1. Then, the *j*th probability of turning on a hidden unit is computed by applying the logistic function[15]

$$p(h_j=1)=\sigma(b_j+\sum_{i=1}^{m} v_i W_{ij}) \tag{1}$$

where $\sigma(x)=\frac{1}{1+e^{-x}}$ is the logistic function.

A random number with a uniform distribution between 0 and 1 is generated. The value of the hidden unit is that, if the turning on probability of the hidden unit is greater than that random number, the value of the unit is 1, otherwise its value is 0.

After determining the state of hidden layer units, the *i*th value of unit in the visible layer is:

$$v_i=\sigma(a_i+\sum_{j=1}^{n} h_j W_{ij}) \tag{2}$$

From the Contrast Divergence[16], the weights and biases update as:

$$\Delta W_{ij}=\varepsilon(\langle v_i h_j\rangle_{data}-\langle v_i h_j\rangle_{mod\,el}) \tag{3}$$

$$\Delta a_i=\varepsilon(\langle v_i\rangle_{data}-\langle v_i\rangle_{mod\,el}) \tag{4}$$

$$\Delta b_j=\varepsilon(\langle h_j\rangle_{data}-\langle h_j\rangle_{mod\,el}) \tag{5}$$

where ε is learning rate.

Because a different RBM is constructed for each user, weights and biases update *M* times in one iteration. *M* is the number of users.

$$W=W+\Delta W \tag{6}$$

$$a=a+\Delta a \tag{7}$$

$$b=b+\Delta b \tag{8}$$

2.2 Recommendation algorithm based on RBM and item type

After training the R_RBM, the common method is to calculate the similarity between users by using the prediction ratings, so as to get the neighbors. In this paper, a collaborative filtering based on RBM and item type (RBM_ITCF) is proposed. Considering that there is a deviation between the prediction ratings and the real ratings, the neighbors calculated by prediction ratings is also deviated. Therefore, we introduce the item type. Considering the influence of item type of users evaluated on the neighbors, we modify the neighbors calculated by the prediction rating.

Based on this, the similarity between users is composed of two aspects: the similarity of ratings, or rating similarity for short, and the similarity between users evaluated items, or item type similarity for short.

1) Rating similarity

Rating similarity is calculated by Person correlation coefficient. Its range is [0, 1]. The higher the Person correlation coefficient, the greater the rating similarity is. The calculation method is as follows.

$$sim_R(i,j)=\frac{\sum_{c\in I_{ij}}(R_{i,c}-\bar{R}_i)(R_{j,c}-\bar{R}_j)}{\sqrt{\sum_{c\in I_{ij}}(R_{i,c}-\bar{R}_i)^2}\sqrt{\sum_{c\in I_{ij}}(R_{j,c}-\bar{R}_j)^2}} \tag{9}$$

Among them, *R* is rating matrix, $sim_R(i,j)$ is the rating similarity of user *i* and *j*, $R_{i,c}$ is prediction rating of user *i* on item *c*, $\bar{R}_i$ and $\bar{R}_j$ represent the average ratings which user *i* and *j* on all items. The common item sets scored by user *i* and user *j* are represented by I_{ij} .

2) Item type similarity

Matrix of item types evaluated by users, or item type matrix for short, is represented by S (M, K). Where *M* is the number of users, *K* is the number of item types. The value of matrix is 0 or 1, indicating whether the user has evaluated this type of item. s_{ij} indicates whether the user *i* has evaluated the type item *j*. If the user has evaluated, then $s_{ij}=1$, if the user has not evaluated, then $s_{ij}=0$.

Because the value of the item type matrix is 0 or 1. The Jaccard correlation coefficient is used to calculate the item type similarity between two users, indicating that two users jointly evaluated the proportion of all types of items that two users have evaluated. The range of the item type similarity is [0, 1]. The higher the Jaccard correlation coefficient is, the greater the item type similarity between two users is. The calculation method is as follows.

$$sim_S(i,j)=\frac{|S_i \cap S_j|}{|S_i \cup S_j|} \tag{10}$$

Among them, $sim_S(i,j)$ is the item type similarity between user *i* and user *j* evaluated item types. S_i is the set of item types that the user *i* has evaluated, $|S_i|$ is the number of elements in the set.

3) Getting the neighbors

The neighbors of the target user are obtained based on the rating similarity. A threshold is set for the item type similarity. The user who has the larger item type similarity with the target user than threshold and has similar ratings to the target user can be the neighbors of the target user.

The method of obtaining the neighbors of the target user is: first, we rank the rating similarity between target users and other users in a descending order. Then, starting from the nearest user with the most rating similarity, we compare the item type similarity between the nearest user and the target user with threshold. If the item type similarity is greater than the threshold, this user is the neighbor of target user. If the item type similarity is less than threshold, then we continue to compare the next user. Until the specified number of neighbors are found.

According to the neighbors of the target user, the prediction of user *u* on item *k* is:

$$P_{u,k}=\bar{R}_u+\frac{\sum_{m=1}^{G} sim_R(u,m)(R_{m,k}-\bar{R}_m)}{\sum_{m=1}^{G} sim_R(u,m)} \tag{11}$$

Among them, $P_{u,k}$ is the prediction rating of user *u* on item *k*, $\bar{R}_u$ is average rating of user *u* on all items, *G* is the number of neighbors, *m* is all the neighbors of target user, $sim_R(u,m)$ is rating similarity of user *u* and user *m*, $R_{m,k}$ is rating of user *u* on item *k*.

After calculating the prediction rating $P_{u,k}$ of target user *u* on item *k*, we sort $P_{u,k}$ by descending order; we provide *n* top ranking items to target user as the user's *top-n* recommendation set.

3. experiment and analysis

3.1 description of dataset and evaluation index

According to MovieLens, the data were collected between September, 1997 and April, 1998 and represent the distribution of all ratings MovieLens obtained during this period. The data set consists of 100 thousand ratings of 943 users on 1682 movies, and ratings are integers between 1 and 5. The higher the rating is, the more users liked the item. Rating 0 indicating that the user didn't rate the item. Before the experiment, the data are divided by 5 to make the rating data are all in the range of 0-1, and finally multiplying the predicted data by 5 to get the final prediction. The experiment divides all the data into ten disjoint parts, in which the nine part is the training set and the remaining part is the test set.

In this paper, the evaluation index is root mean squared error (RMSE). The larger the result of the RMSE is, the worse the performance of the proposed algorithm will be. If the actual ratings are $p_1, p_2, \cdots p_k$, the prediction ratings are $q_1, q_2, \cdots q_k$, then:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{k}(p_i - q_i)^2}{k}} \tag{12}$$

We train the RBM with ε =0.01. The weights were initialized with small random values sampled from a zero-mean normal distribution with standard deviation 0.01. The biases of visible layer and hidden layer were initialized with 0. All models were trained for 50 epochs through the entire training dataset. The number of visible units is 943, The item is divided into 18 types.

3.2 determining the number of hidden layer units of RBM

In order to determine the number of hidden layer units, the number of hidden layer units is taken as a different value under the premise that the other variables remain unchanged, and the recommendation results are calculated.

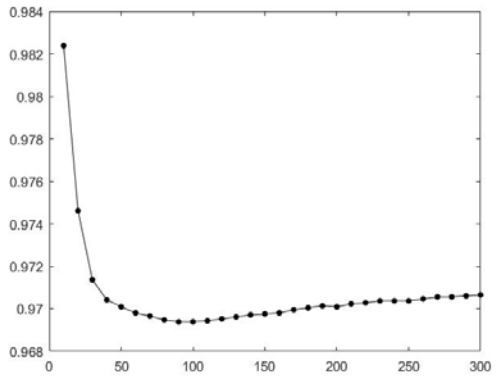

Fig.2 The influence of the number of hidden units on RMSE

In figure 2, the x-label represents the number of hidden layer units, and the y-label represents the value of RMSE. When the number of hidden layer units is less than 90, the value of RMSE decreases monotonically. When the number of hidden layer units is greater than 90, the value of RMSE increases monotonically. When the number of hidden layer units is 90, the model has the best performance. The number of hidden layer units in the following experiments is initialized to 90.

3.3 determining the threshold y where calculating the neighbors

When calculating neighbors, we need to determine the threshold of item type similarity. In order to determine the threshold, the threshold is taken as a different value under the premise that the other variables remain unchanged, and the recommendation results are calculated.

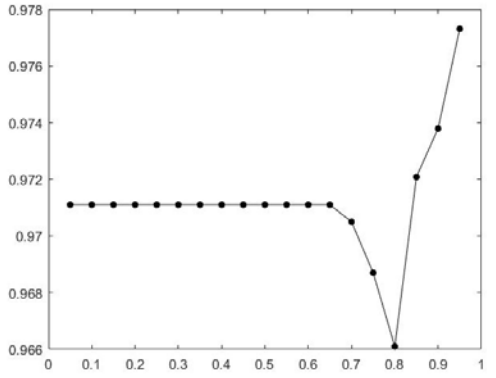

Fig.3 The influence of the threshold on RMSE

In figure 3, the x-label represents the value of the threshold, and the y-label represents the value of RMSE. When the threshold is less than 0.6, the value of RMSE remains unchanged. When the threshold is less than 0.8 and more than 0.6, the value of RMSE decreases monotonically. When the threshold is greater than 0.8, the value of RMSE increases monotonically.. When the threshold is 0.8, the model has the best performance. The threshold in the following experiments is initialized to 0.8.

When the threshold is between 0 and 0.6, the item type similarity between the neighbors calculated by rating similarity and the target user is greater than the threshold. Therefore, in this interval, the

threshold does not affect the computation of the neighbors, and thus has no effect on the results of the algorithm, so the accuracy of the algorithm remains unchanged.

3.4 determining the number of neighbors

In order to determine the number of neighbors, the number of neighbors is taken as a different value under the premise that the other variables remain unchanged, and the recommendation results of three models are calculated.

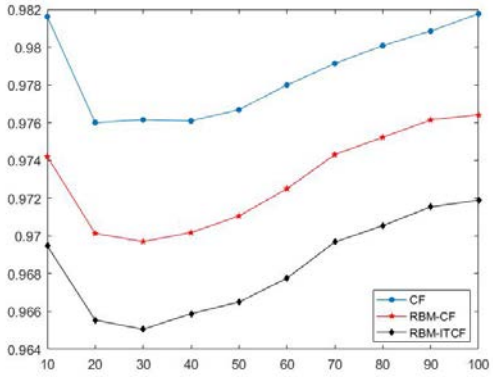

Fig.4 The influence of the number of neighbors on RMSE

In figure 4, x-label represents the number of neighbors, y-label represents the value of RMSE. With the change of the number of neighbors, the trend of three models is the same. When the number of neighbors is less than 30, the value of RMSEs decreases monotonically. When the number of neighbors is greater than 30, the value of RMSEs increases monotonically. When the number of neighbors is 30, three models have the best performance. The number of neighbors in the following experiments is initialized to 30.

3.5 comparing with other algorithms

In this experiment, when calculating neighbors, the threshold of item type similarity is set to 0.8, and the number of neighbors is set to 30. The number of hidden units in the R_RBM is set to 90. The recommended performance of several recommendation algorithms is compared, as shown in the following figure:

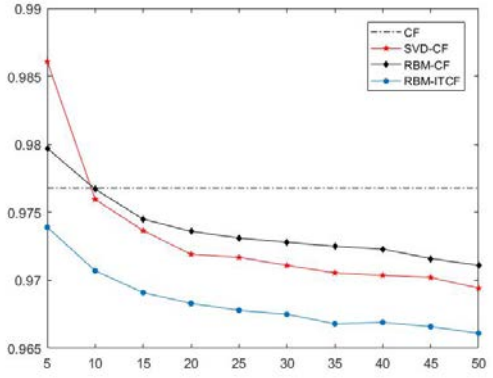

Fig.5 Comparison of RMSE of four algorithms

In figure 5, x-label represents the number of iteration, y-label represents the value of RMSE. As the number of iterations increases, RMSE decreases. The more the number of iterations is, the better the performance of the algorithm will be. Since RBM_ITCF and RBM_CF use same method to fill in the rating matrix, they get the same prediction ratings. However, the neighbors of RBM_ITCF is obtained by item type similarity combined with score similarity, the prediction ratings are closer to the real ratings. With the change of iterations, the trend of RBM_ITCF and RBM_CF is the same, but the recommendation effect of RBM_ITCF is better.

4. Summary

At present, the research of RBM has been a great success and has been widely used. But the previous models only aim at the sparsity problem of the rating matrix to fill missing ratings. The RBM_ITCF algorithm proposed in this paper calculates the neighbors based on the nearest neighbor and considers the rating similarity and item type similarity. Experiments show that the new model not only solves the sparsity problem of the rating matrix, but also gets more accurate neighbors. When using the same method to calculate the prediction ratings, the neighbors calculated by the new model is more accurate than the neighbors calculated by the general method.

References

[1] Zhang S, Wang W, Ford J, et al. 2005. Using singular value decomposition approximation for collaborative filtering. *Institute of Electrical and Electronics Computer Society*.
[2] Hinton G E, Salakhutdinov R R. 2006. Reducing the dimensionality of data with neural networks. *Science*, Vol. 313.
[3] Lecun Y, Bengio Y, Hinton G E. 2015. Deep learning. *Nature*, Vol. 521.
[4] Krizhevsky A, Sutskever I, Hinton G E. 2012. ImageNet classification with deep convolutional neural networks. *Association for Computing Machinery*, Vol. 60(2).
[5] Hinton G E, Deng L, Yu D, et al. 2012. Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups. *Institute of Electrical and Electronics Engineer*, Vol. 29(6).
[6] Collobert R, Weston J, Karlen M, et al. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, Vol. 12(1).
[7] Wang X, Wang Y. 2014. Improving content-based and hybrid music recommendation using deep learning. *Association for Computing Machinery*.
[8] Salakhutdinov R, Mnih A, Hinton G E. 2007. Restricted boltzmann machines for collaborative filtering. *Association for Computing Machinery*, Vol. 227.
[9] Sedhain S, Menon A K, Sannery S, et al. 2015. AutoRec: autoencoders meet collaborative filtering. *Association for Computing Machinery*.
[10] Liu Y, Tong Q, Du Z, et al. 2014. Content-boosted restricted boltzmann machine for recommendation. *Springer Verlag*, Vol. 8681.
[11] Yu K, Xu X, Ester M, et al. 2003. Feature weighting and instance selection for collaborative filtering: an information-theoretic approach. *Knowledge and Information Systems*, Vol. 5(2).
[12] Liu X. 2017. An improved clustering-based collaborative filtering recommendation algorithm. *Cluster Computing*, Vol. 20(1).
[13] Zhao C, Shi J, Jiang T, et al. 2016. Application of deep belief nets for collaborative filtering. *Institute of Electrical and Electronics Engineers*.
[14] Georgiev K, Nakov P. 2013. A non-IID framework for collaborative filtering with restricted boltzmann machines. *International Machine Learning Society*.
[15] Hinton G E. 2012. A Practical Guide to Training Restricted Boltzmann Machines. *Momentum*, Vol. 9(1).
[16] Carreira-Perpinan M A, Hinton G E. 2015. On contrastive divergence learning. *Society for Artificial Intelligence and Statistics*.