Optimizations for Item-based Collaborative Filtering Algorithm

Shuang Xia
WeST, Tsinghua University
Beijing, China
xs06@mails.tsinghua.edu.cn

Yang Zhao
WeST, Tsinghua University
Beijing, China
zhaoyang07@mails.tsinghua.edu.cn

Yong Zhang
WeST, Tsinghua University
Beijing, China
zhaoyong05@tsinghua.edu.cn

Chunxiao Xin
WeST, Tsinghua University
Beijing, China
xingcx@tsinghua.eud.cn

Scott Roepnack
Florida Atlantic University
Florida, United State
sroepnac@fau.edu

Shihong Huang
Florida Atlantic University
Florida, United State
shihong@cse.fau.edu

Abstract:

Collaborative Filtering (CF) is widely used in the Internet for recommender systems to find items that fit users' interest by exploring users' opinion expressed on other items. However there are two challenges for CF algorithm, which are recommendation accuracy and data sparsity. In this paper, we try to address the accuracy problem with an approach of deviation adjustment in item-based CF. Its main idea is to add a constant value to every prediction on each user or each item to modify the uniform error between prediction and actual rating of one user or one item. Our deviation adjustment approach can be also used in other kinds of CF algorithms. For data sparsity, we improve similarity computation by filling some blank rating with a user's average rating to help decrease the sparsity of data. We run experiments with our optimization of similarity computation and deviation adjustment by using MovieLens data set. The result shows these methods can generate better predication compared with the baseline CF algorithm.

Keywords:

Personalized Services; Recommender Systems; Item-based Collaborative Filtering

1. Introduction

With the incredible growth of the information on the Internet, it's more and more difficult for us to find resources that match our interests. Therefore recommender systems are developed in which collaborative filtering (CF) is commonly used[1]. Although CF algorithm has widely used and succeeded, some problems are still under research such as predication accuracy, data sparsity and scalability. In order to address these issues, several kinds of CF algorithm are proposed. User-based CF, item-based CF and singular value decomposition(SVD)[2] are three commonly used algorithms.

User-based CF generates prediction by exploring the target user's similar users' opinions on the target item. As their historical preferences on the same items are quite alike, their preference on the target item may be also quite similar[3]. However, in most conditions, a recommender system has much more users than items, and most users only rated very small part of items, which makes it much harder to find similar users than to find similar items. Take this into consideration, item-based CF is proposed to address the previous issue.

Item-based CF's main idea is to predict a target user's preference on a target item by finding similar items of the target item[4]. When computing two items' similarity, we explore users who rated both items, and ignore users who rated only one of them, which makes a sparse data matrix even sparser. In this paper, we try to avoid this limitation by filling the unrated item with the user's average rating when computing item similarity, based on that a user will rate most items with his normal rating. Experimental result shows that this method can effectively increase the accuracy of item-based CF.

After finding similar items, we can predict ratings for items. The predicted rating is usually different from the actual one. For all the ratings we already knew, if we cover one of them, we can use other ratings to predict the covered one, and then we can know the deviation of our predication to the actual rating. Furthermore, by repeating the same step on each rating, we can know our algorithm's predication deviation on all the ratings of one item or one user. This deviation value can be used to adjust the prediction algorithm. In this paper, we propose an approach to adjust the deviation in item-based CF. We name this approach as *Deviation Adjustment*. This approach can be also applied in other kinds of CF algorithm such as user-based CF. Experimental result shows this approach is effective in both item-based and user-based CFs.

978-1-4244-6899-7/10/$26.00 ©2010 IEEE

The rest of this paper is organized as following. The next section provides a brief introduction for the related work. Then we propose our modifications on similarity computation and deviation adjustment method in Section 3. In Section 4, we test our modified CF algorithms using MovieLens data set and compare them with the baseline CF algorithm. Last section is the conclusion and future work.

2. Related Work

In this section, we briefly introduce some previous research work on CF algorithm.

The objective of CF algorithm is to generate an accurate prediction of a target user on a target item in real time. Compared with other recommendation technology, CF algorithm mainly considers the users' opinion on the items instead of the items or users' own character like movie's content or user's gender[5]. There are two main kinds of CF algorithms, user-based CF and item-based CF, both of which estimate correlation between users or items and then use it to compute predication. There are two procedures of item-based or user-based CF, which are estimating similarity and generating prediction, the following content will take item-based CF as an example.

2.1. Estimate Similarity

There are several different baseline ways to compute item similarity. In order to compute item i and j's similarity, all these algorithms first isolate the users who rated both items. And then treat their ratings on each item as a vector in the N-dimension user space to compute their similarity. Adjust cosine has been proved the best one of them for item-based CF.[4][6]

a) Cosine based similarity

This method takes the cosine value of two vectors' angle as similarity. Formally, the similarity between item i and j, denoted by $sim(i, j)$ is measured as

$$sim(i,j) = \frac{\sum_{u \in User} R_{u,i} R_{u,j}}{\sqrt{\sum_{u \in User} R_{u,i}^2} \sqrt{\sum_{u \in User} R_{u,j}^2}} \quad (1)$$

Where *User* is a collection of users who rated both item i and item j.

b) Correlation-based similarity

This kind of similarity is measured by Pearson correlation that similarity between item i and j is given by equation(2) Where $R_{u,i}$ denotes user u's rating on item i, *and* $\bar{R}_i$ is the average rating on item i.

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

c) Adjusted cosine similarity

Considering users' different scale of rating, like user u_1's score is between 3 and 5 while user u_2's rating is between 2 and 4, so the user u_2's rating will have more influence on cosine value computation which makes no sense. In order to avoid this drawback, adjusted cosine[4] is proposed to unify different user's rating scale as

$$sim(i,j) = \frac{\sum_{u \in User} (R_{u,i} - \bar{R}_u)(R_{u,j} - \bar{R}_u)}{\sqrt{\sum_{u \in User} (R_{u,i} - \bar{R}_u)^2} \sqrt{\sum_{u \in User} (R_{u,j} - \bar{R}_u)^2}} \quad (3)$$

2.2. Generate Prediction

After calculating item similarities we can generate the predication of the target user u's rating on the target item i using weighted sum as the equation below

$$P_{u,i} = \frac{\sum_{j \in Item} (R_{u,j} \times sim(i,j))}{\sum_{j \in Item} sim(i,j)} \quad (4)$$

Where *Item* is the set of items that the target user u has rated. This formula equals to

$$P_{u,i} = \frac{\sum_{j \in Item} ((R_{u,j} - \bar{R}_u) \times sim(i,j))}{\sum_{j \in Item} sim(i,j)} + \bar{R}_u \quad (5)$$

Although the previous formula is widely used[4][6], there is a more accurate[5] method to generate the prediction which uses weighted sum to modify item average rating instead of user average rating introduced by [7] as below

$$P_{u,i} = \frac{\sum_{j \in Item} ((R_{u,j} - \bar{R}_j) \times sim(i,j))}{\sum_{j \in Item} sim(i,j)} + \bar{R}_i \quad (6)$$

3. Optimized Item-based Collaborative Filtering

In this section, we proposed two kinds of optimizations. The first one is called average filling, the second is deviation adjustment.

3.1. Average Filling

In the scenario of baseline item-based CF, when computing similarities between item i and j, we only take the users who have rated both item i and j into account. Consider the situation shown in Table 1. In baseline similarity computation, item i_4 and i_5 are completely the same. But we can also get some rough information from the users who have only rated one of them like user u_1 and u_2. For example, user u_1 likes rating a normal movie as 3 point, but he rated item i_4 5 point. In user u_1's opinion, there is a great chance that i_4 and i_5 are quite different. On the contrary, user u_2 usually rate his normal movie as 4 point and he gives item i_5 a 4 point which means item i_4 and i_5 are the same in user u_2's opinion most probably. Therefore, in order to make use of this part of information, we can simply fill the unrated item with the user's average rating when we process similarity computation which shown in Table 2. Note that we can

get no information from a user who rated none of the item *i* and *j* when computing the similarity between them, so it's not necessary to fill the entire unrated field.

Table 1.The baseline method of computing similarity between item i_4 and i_5 only considers user u_3's rating

	i_1	i_2	i_3	i_4	i_5
u_1	3	3	3	5	
u_2	3	4	5		4
u_3	4	5	3	4	4

Table 2.We can get all 3 users opinion by filling average value of u1 and u2

	i_1	i_2	i_3	i_4	i_5
u_1	3	3	3	5	**3.5**
u_2	3	4	5	**4**	4
u_3	4	5	3	4	4

3.2. Deviation Adjustment

The goal of CF algorithm is to minimize the error between predicted and actual user preference[8]. In order to fix the deviation in a CF algorithm, we propose *Deviation Adjustment* approach to rectify it. The deviation indicates the error which has the same behavior on one user or one item due to some drawbacks of the algorithm that generates the prediction. A typical situation shows in the Table 3.

Table 3.The predicted ratings of training data for item i1 are 0.1 point high in average. So we can minus prediction for i1 by 0.1 when generating prediction for test data.

i_1 rating	training data				test data
user	u_1	u_2	u_3	u_4	u_5
actual	4	3	5	3	4
predicted	4.1	3.2	5	3.1	4.1

Generally, we use the CF algorithm with the training data as input to predict the rating and evaluate the result with the test data. However, we can know the algorithm's error by using the training data to predict the training data. For example, imagining we use 8000 user ratings to train a CF algorithm, we can cover one rating and use the other 7999 ratings to predict the covered one. By covering every rating in the training set, we can get a complete predicted training data. So the user deviation can be measured by

$$\varepsilon(u) = \frac{\sum_i (R_{u,i} - p(u,i))}{n_u} \tag{7}$$

Where n_u denotes the amount of user who rated item *i*. On the other hand, we can rectify user deviation with $\varepsilon(i)$ in user-based CF. The item deviation can be measured by

$$\varepsilon(i) = \frac{\sum_u (R_{u,i} - p(u,i))}{n_i} \tag{8}$$

Where n_i denotes the amount of user who rated item *i*.

When applying deviation adjustment in item-based CF, after calculating item similarities, covering one rating barely changes item similarities. Therefore we don't have to recalculate the item similarity after covering one rating.

In item-based CF, a predicted rating differs from the actual rating is mainly caused by not-well-calculated item correlations. But we can't assert there is no user deviation in item-based CF. Therefore in our experiment we change the prediction formula to

$$p'(u,i) = p(u,i) + \alpha\varepsilon(u) + (1-\alpha)\varepsilon(i) \tag{9}$$

Where α ranges from 0 to 1, indicates the weight of user deviation adjustment and item deviation adjustment.

4. Experiments and Evaluation

In this section, we present our experiment of the proposed algorithms. In order to evaluate our method, we implement all the previous similarity measurements and prediction algorithms and add our optimization to them. In addition, we combine deviation adjustment with user-based algorithm to evaluate its performance.

4.1. Data Set

The experimental data set is provided by the MovieLens recommender system. We use the publically available data set which includes 100,000 ratings on1682 movies from 943 users who rated at least 20 movies. The ratings range from 1 to 5 point. We divide the data set into two parts, 80,000 ratings as training data and 20,000 ratings as test data.

4.2. Evaluation Metrics

The evaluation of recommender systems mainly divides into two categories, statistical accuracy metrics and decision support metrics[9]. We will use the mean absolute error (MAE) which is widely used as a statistical accuracy metric. The MAE metric evaluates each rating-prediction pair $< p_i, q_i >$'s absolute difference. For all the N ratings in test data, MAE is formally given by

$$MAE = \frac{\sum_{i=1}^{N} |p_i - q_i|}{N} \tag{10}$$

4.3. Experimental Results and Analysis

a) Average Filling

We apply the average filling method with the three kinds of similarity algorithms introduced in Section 2.1 and the baseline prediction algorithm introduced in Section 2.2 equation (4). The experimental result shows in the following figures.

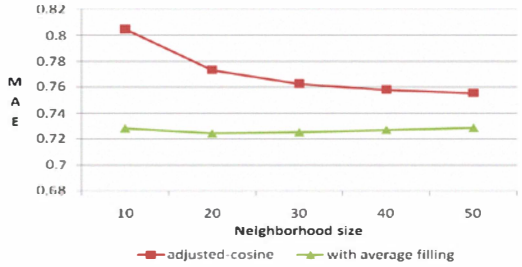

Figure 1.Comparison of adjusted cosine similarity and adjusted cosine similarity with average filling

As Figure 1 shows, adjusted cosine with average filling has decreased the MAE, especially in low neighborhood size. In Figure 2 and Figure 3 we can see that it also optimized the correlation-based similarity, but it doesn't work on cosine similarity as the filled user average rating enlarges the drawback of cosine similarity which ignores the difference of users' rating scale.

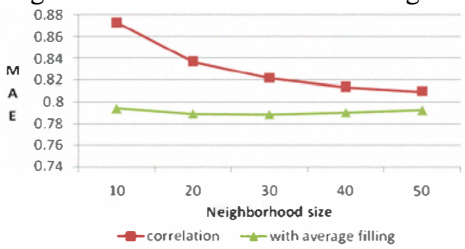

Figure 2.Comparison of correlation similarity and correlation similarity with average filling

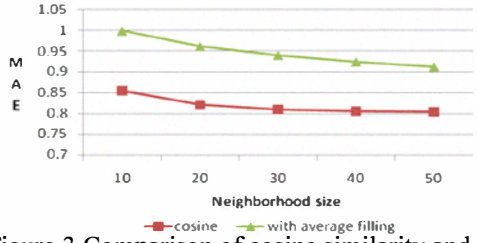

Figure 3.Comparison of cosine similarity and cosine similarity with average filling

From Figure 1-3, we can see that adjusted-cosine similarity has the best performance, and with average filling the MAE decreases from 0.773 to 0.724 at a neighborhood size of 20 which is the most accurate one. As similarity computation and prediction generation are two separated steps for item-based CF, we will use the best similarity calculated by adjusted cosine similarity with average filling and 20 neighbors in the next experimental procedure

b) Deviation Adjustment

There are two kinds of prediction algorithms in item-based CF, and we will apply our deviation adjustment with both of them to run the experiment in order to evaluate its performance. The result shows in the following figures.

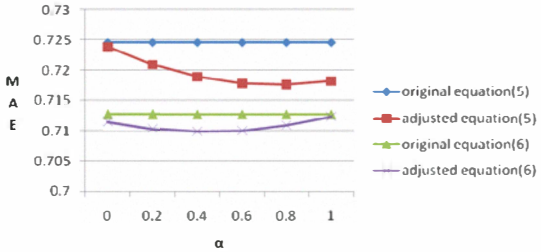

Figure 4. Comparison of item-based CF with equation (5) and (6) prediction with deviation adjustment

From Figures 4, we can see that item-based can generate more accurate prediction with equation (6) than with equation (5). And with $\alpha = 0.4$ as about half user deviation and half item deviation adjustment, the MAE is down to 0.709, while predication with the equation (5) has a large item deviation error as $\varepsilon(i)$ has an effective rectification.

c) User-based CF with Deviation Adjustment

We also apply the deviation adjustment to a baseline user-based CF. We still use the adjusted cosine similarity and a prediction algorithm similar with equation (6)[7] for the user-based CF. The result is given by the following figure which shows deviation modification can also optimize user-based CF. This figure also shows that user-based CF has a large user deviation as $\varepsilon(u)$ has an effective modification.

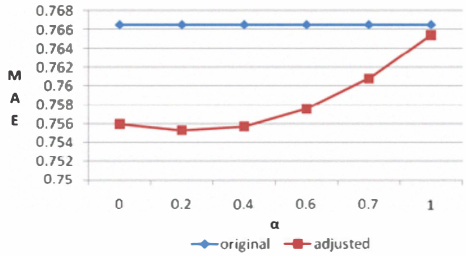

Figure 5.User-based CF with deviation adjustment.

5. Conclusion and Future Work

Collaborative filtering is an effective technology for recommender systems. In this paper, we proposed two novel methods to optimize the quality of item-based CF. Experimental results show that the average filling method can provide more accurate similarity estimation. The deviation adjustment approach can rectify the uniform error in several kinds of collaborative filtering algorithms. We think there might be an approach which uses the predicted training data to adjust the similarity computation. We are now working on it.

Acknowledgements

This paper is supported by the National High-tech R&D Program of China under Grant No. 2009AA01Z143 and the Research Foundation of the

Ministry of Railways and Tsinghua University under Grant No. 20091111068

References

[1] P. Resnick, H.R. Varian, "Recommender systems - introduction to the special section," Commun. ACM 40(3), pp. 56 – 58 , 1997

[2] D. Billsus, M. J. Pazzani, "Learning collaborative information filters. In 15th International Conference on Machine Learning ," Madison, WI, 1998

[3] J.S. Breese, D. Heckerman, C. Kadie, "Empirical Analysis of Predictive Algorithms for Collaborative Filtering," Proc of the 14th Conf on Uncertainty in Artificial Intelligence. Madison. WI: Morgan Kaufmann Publisher,1998

[4] B. Sarwar, B. Karypis, J. Konstan, "Item-based collaborative filtering recommendation algorithms," Proceeding of the 10th International World Wide Web Conference, 2001

[5] M. Papagelis, D. Rousidis, "Qualitative Analysis of Userbased and Item-based Prediction Algorithms for Recommendation Systems," CIA 2004, pp.152-166, 2004

[6] M.L. Clemente, "Experimental Results on Item-Based Algorithms for Independent Domain Collaborative Filtering," International Conf on Automated solutions for Cross Media Content and Multi-channel Distribution, 2008

[7] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, J. Riedl, "GroupLens: an open architecture for collaborative filtering of netnews," In Proc. of the ACM Conf. on Computer supported Cooperative Work , pp.175–186, 2004

[8] T. Jambor. J. Wang. "Goal-Driven Collaborative Filtering – A Directional Error Based Approach," ECIR 2010, pp. 407–419, 2010

[9] H. N. Kim, A. T. Ji, G.S. Jo, "Enhanced Prediction Algorithm for Item-Based Collaborative Filtering Recommendation," EC-Web 2006, pp.41 – 50, 2006