An Improved User-model-based Collaborative Filtering Algorithm

Qian Wang*, Min Sun, Cong Xu

College of Computer Science, Chongqing University, Chongqing 400030, China

Abstract

Collaborative filtering is an algorithm successfully and widely used in recommender system. However, it suffers from data sparsity, recommendation accuracy and system scalability problems. This paper proposes an improved user model for collaborative filtering to explore a solution to these problems. The ratings are firstly been normalized by decoupling normalization method, and then a nonlinear forgetting function is introduced to assign the ratings different time weights to mimic the users’ interest drift. In similarity computation, an effective weighting factor is added to the Pearson correlation similarity computation to get more accurate neighbor users. The algorithm is tested on MovieLens dataset and the comparative experiment shows that the algorithm proposed in this paper can provide a better performance.

Keywords: Collaborative Filtering; Recommender System; User Model; Decoupling Normalization; Time Weight; MAE

1 Introduction

Rapid growth of Information technology has given our modern life a new vital force and energy, and meanwhile it puts forward a severe problem of information overload [1]. Recommender system is developed to find what each individual prefers through the rating records and online shopping behavior, which helps users a lot to get their interested information. Generally there are three kinds of filtering methods applied in recommender system. They are content filtering [2], collaborative filtering [3] and combination filtering [4, 5]. Content filtering, also named information filtering, recommends users the items that similar to what they used to like, and it is used in some early systems for its simplicity.

Collaborative filtering is widely used in recommender systems. The method of collaborative filtering can be memory-based [3] or model-based [5]. Memory-based collaborative filtering makes recommendation on all the data gathered, and new generated data can also be taken into count

*Project supported by Chongqing Municipal Natural Science Foundation (CSTC 2009BB2046) and Chongqing University Postgraduates’ Science and Innovation Fund, grant CDJXS10180009.
*Corresponding author.
Email address: wangqian@cqu.edu.cn (Qian Wang).
for recommendation, so it can make high recommendation accuracy. However, it is costly in
computation, and meanwhile, with the data expanding, the method performs worse for its bad
scalability. Model-based collaborative filtering firstly needs to construct a model on all the data
offline, and then load the model to memory to generate recommendation results online. It sig-
nificantly improves system scalability while makes some concessions on accuracy. And at present
clustering, Bayesian Network, machine learning, association rules and some other technologies
have been applied into the model constructing.

Combination filtering absorbs advantages of kinds of recommendation methods, and properly
combines them together to obtain a better result. Thus, how to choose the “proper” methods
and in what ways combine them together becomes a challenge for research. The recent researches
on combination filtering are mainly in these three ways, which are respectively Post-combination,
Middle-combination and Pre-combination [5-7].

The algorithm proposed in this paper is based on a user model. The user model is improved
because, first, the ratings are processed by a normalization method. since users with similar
interest may have different rating records to a same item. In the early research of collabora-
tive filtering, Resnick et al. normalize the user ratings to Gaussian distribution [8]. J. Rong
and S. Luo propose decoupled models for collaborative filtering [9], and then they make a study
of normalization methods in collaborative filtering [10] and find that decoupling normalization
method outperforms Gaussian normalization. In this paper, we adopt the decoupling normal-
ization method to make the ratings into the same scale. Second, it takes consideration of time
weight to user ratings in the process of model constructing. In reference [11], the authors brought
up a user model to help improving collaborative filtering, yet they did not take time effect on
ratings for consideration. Actually, users’ preferences drift dynamically as time going. Reference
[12] introduces a linear gradually forgotten function to mimic the users’ interest drift. Reference
[13] adopts a nonlinear logistic function to improve the accuracy of the algorithm. This paper
refers to the Ebbinhaus Forgetting Curve and uses a different logistic function to simulate the
forgetting phenomenon. These two steps concentrate on improving the accuracy of user ratings
in the model. However, there are still much missing data in the model matrix. Therefore in this
paper, we add an effective weighting factor [14] to user similarity computation, which can not
only avoid the missing data, but also helps to get more accuracy neighbor users.

The rest of this paper is arranged as follows: Section 2 gives how the new algorithm works
a brief description. The details of this new method for collaborative filtering are discussed in
Section 3 and 4. The empirical part, experimental results are presented in Section 5. Section 6
concludes this work.

2 An Improved User-model-based Collaborative Filtering

Conventional collaborative filtering computes user similarity directly on the original user rating
records and therefore the quality of recommendation is not so good. Besides, it is memory based
and with the increase of the number of users and items, the online computation takes too much
time. However, the proposed user-model-based collaborative filtering computes user similarity
on the processed user rating model. The user ratings are firstly been normalized by decoupling
normalization method. And then a nonlinear forgetting function is introduced to assign the ratings
different time weights to mimic the users’ interest drift. Thus, we get the processed user rating
matrix, which is actually the user model. After the finish of user model constructing, the similarity
between users can be computed. And in the new method of user similarity computation, we add an effective weighting factor to the Pearson correlation similarity computation to get more accurate neighbor users. After that, collaborative filtering is adopted to generate the final recommendation results. In this algorithm, only the filtering based on the model has to be executed online, and all the computation for constructing of the user model and getting the neighbor users are finished offline. Although the offline model constructing is on the data that is not in the current time, it is reasonable because the users' preference cannot change so fast and the contribution of data change during the model constructing can be sacrificed to get more efficiency and accuracy. The Fig. 1 shows the work flow of the improved user-model-based collaborative filtering.

3 User Model Constructing

For a model-based collaborative filtering, the quality of recommendation depends greatly on the model [15]. This paper takes rating normalization and time weight into model constructing work.

3.1 Decoupling Normalization for Ratings

Since the user ratings serve as an important basis in collaborative filtering, whether the ratings reflect the true interest of users becomes very critical to the quality of recommendation. Just as what proposed in [8], these two aspects of reasons can lead to the rating variance for users share the same interest.

- **Different rating scales** This is noted as the phenomenon that some users tend to assign ratings for items in a wide range while some users tend to give items the ratings in a narrow range. For example, if the ratings are set to be scaled from 1 to 5, some users are used to take ratings from 1 to 5 to express how much they like the items while some are used to take ratings from 2 to 4 to show their degree of preference.

- **Different average rating levels** This comes with the fact that some users are “tolerant” and their ratings are found higher than some other “strict” users, although they actually share the same interest in the same items. For instance, both user A and user B like item
M very much, that is to say, they share the same taste for item M. However, “tolerant” user A takes 5 to express that he or she likes it the best while “strict” user B just assigns 3 or 4 for the favorite item to show his or her preference.

Due to the different rating habits of users, an extra transformation is needed to make ratings a more closely reflection for users’ preference. In this paper, the decoupling normalization method [9] is applied to normalize the ratings. The decoupling normalization is based on a probabilistic mechanism. And it comes from these two assumptions:

First, when there are a large portion of items rated by a user \(i\) as no more than rating \(R\), items with the rating \(R\) are more likely to be liked by the user \(i\).

Second, when there are a large portion of items rated by a user \(i\) as exactly the rating \(R\), items with the rating \(R\) have less chance to be liked by the user \(i\).

According to these two assumptions, and following “halfway accumulation distribution”, the equation of decoupling normalization is given like:

\[
\hat{R} = P_i(R \text{ is preferred}) = P_i(Rating < R) - P_i(Rating = R)/2
\] (1)

The following gives a simple example to show how the user ratings are normalized by this decoupling normalization method. Here, \(P_i(R \text{ is preferred})\) represents how items with the rating \(R\) is liked by user \(i\). \(R\) is a rating category, \(P_i(Rating < R)\) and \(P_i(Rating = R)\) stand for the probability which implement how the user \(i\) likes the items with the rating \(R\). Now for the item \(j\), since it is rated by user \(i\) as \(R\), we can take \(P_i(R \text{ is preferred})\) as the normalized rating result that user \(i\) assigned for item \(j\), which is marked as \(\hat{R}\).

Assume there are three users \(U_1, U_2\) and \(U_3\), who shared the similar interest in these five items, and their ratings for the five items are shown in Table 1 (a).

Table 1: Related tables in section 3

<table>
<thead>
<tr>
<th></th>
<th>I1</th>
<th>I2</th>
<th>I3</th>
<th>I4</th>
<th>I5</th>
</tr>
</thead>
<tbody>
<tr>
<td>(u_1)</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>(u_2)</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>(u_3)</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

(a) Ratings before normalization

<table>
<thead>
<tr>
<th></th>
<th>I1</th>
<th>I2</th>
<th>I3</th>
<th>I4</th>
<th>I5</th>
<th>TP</th>
</tr>
</thead>
<tbody>
<tr>
<td>(u_1)</td>
<td>0.1</td>
<td>0.3</td>
<td>0.5</td>
<td>0.7</td>
<td>0.9</td>
<td></td>
</tr>
<tr>
<td>(u_2)</td>
<td>0.2</td>
<td>0.2</td>
<td>0.5</td>
<td>0.8</td>
<td>0.8</td>
<td></td>
</tr>
<tr>
<td>(u_3)</td>
<td>0.2</td>
<td>0.2</td>
<td>0.6</td>
<td>0.6</td>
<td>0.9</td>
<td></td>
</tr>
</tbody>
</table>

(b) Ratings after normalization

<table>
<thead>
<tr>
<th></th>
<th>I1</th>
<th>I2</th>
<th>I3</th>
<th>I4</th>
<th>I5</th>
</tr>
</thead>
<tbody>
<tr>
<td>(u_1)</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>T3</td>
</tr>
<tr>
<td>(u_2)</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>–</td>
<td>5</td>
</tr>
<tr>
<td>(u_3)</td>
<td>–</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>(u_4)</td>
<td>1</td>
<td>–</td>
<td>2</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

(c) Ratings in different period

From the rating information, we can see that \(U_1\) tends to assign ratings for items in a wide scale while \(U_2\) and \(U_3\) are used to give items the ratings in a relative narrow scale. Besides, \(U_2\) seems more “tolerant” than \(U_3\), because ratings by \(U_2\) are higher than \(U_3\). Therefore, we need a conversion to the ratings. The Table 2 shows the rating distribution information according to decoupling method.

From Table 2, we can obtain the normalized result, which is shown in Table 1 (b). It is clear that after decoupling normalization method, their ratings are more similar than before. In Table 1 (a), the rating \(U_1\) assigned for \(I_3\) is 3 while \(U_2\) assigned for \(I_3\) is 4, and clearly the two users’ different rating habits lead to the rating difference. However, after the decoupling normalization, we can see that in Table 1 (b), the two users’ ratings are normalized to the same 0.5, which reflects the users’ preference better.
### Table 2: Rating distribution information from U1, U2 and U3

<table>
<thead>
<tr>
<th>Rating(R)</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1(Rating= R)</td>
<td>0.2</td>
<td>0.2</td>
<td>0.2</td>
<td>0.2</td>
<td>0.2</td>
</tr>
<tr>
<td>P1(Rating &lt;= R)</td>
<td>0.2</td>
<td>0.4</td>
<td>0.6</td>
<td>0.8</td>
<td>1.0</td>
</tr>
<tr>
<td>P1(Rating &lt;= R) - P1(Rating= R) / 2</td>
<td>0.1</td>
<td>0.3</td>
<td>0.5</td>
<td>0.7</td>
<td>0.9</td>
</tr>
<tr>
<td>P1(Rating= R)</td>
<td>-</td>
<td>-</td>
<td>0.4</td>
<td>0.2</td>
<td>0.4</td>
</tr>
<tr>
<td>P1(Rating &lt;= R)</td>
<td>-</td>
<td>-</td>
<td>0.4</td>
<td>0.6</td>
<td>1.0</td>
</tr>
<tr>
<td>P1(Rating &lt;= R) - P1(Rating= R) / 2</td>
<td>-</td>
<td>-</td>
<td>0.2</td>
<td>0.5</td>
<td>0.8</td>
</tr>
<tr>
<td>P1(Rating= R)</td>
<td>-</td>
<td>0.4</td>
<td>0.4</td>
<td>0.2</td>
<td>-</td>
</tr>
<tr>
<td>P1(Rating &lt;= R)</td>
<td>-</td>
<td>0.4</td>
<td>0.8</td>
<td>1.0</td>
<td>-</td>
</tr>
<tr>
<td>P1(Rating &lt;= R) - P1(Rating= R) / 2</td>
<td>-</td>
<td>0.2</td>
<td>0.6</td>
<td>0.9</td>
<td>-</td>
</tr>
</tbody>
</table>

### 3.2 Time Weight Assigning

As time goes by, users’ interest drift, therefore, ratings in different time period should not be treated in the same way.

To illustrate the problem, an example is given as follows.

Table 1 (c) shows the ratings for five items by four users, and the time that users give ratings for these five items are in different period.

From Table 1 (c), if the rating time information is ignored, we can easily find that \( \text{sim}(U1, U2) > \text{sim}(U1, U3) > \text{sim}(U1, U4) \), which indicates that \( U2 \) is the nearest neighbor of \( U1 \). However, when taking time difference into consideration, it is not reasonable to give a conclusion that \( U2 \) is the nearest neighbor of \( U1 \), because comparing the preference in the past time period with the preference in the present time period doesn’t make any sense. And in Table 1 (c), if considering the time contribution, the nearest neighbor of \( U1 \) would be \( U4 \).

That’s an example of different users, and ratings of one individual should also take time into account. For one certain user, the old ratings that he or she assigned for items is questionable because the old interest may change now, and the ratings in the past time cannot be weighted equally as the ratings in the present time. Therefore, in the collaborative filtering, a time weight function is needed to mimic the users’ interest drift.

Ebbinghaus’s research on memory, especially the forgetting curve gives us a reference to forgetting function design. Since the process of forgetting is nonlinear and gradually declined, we use a logistic function to simulate the interest drift. The much more recent the rating is, the greater time weight value it obtains. Besides, a user’s interest is relatively stable in a time period, so we define a time period to modify the logistic function. Ratings in the same time period have the same time weight, instead of each rating owns a different time weight although the ratings are very near in time. In general, the logistic function \( f(t_{i,j}) \) is designed to find what the users really like, and the following gives its expression:

\[
f(t_{i,j}) = \frac{1}{1 + e^{-\frac{t_l - t_{i,j}}{T}}}
\]

Here \( t_l \) is the latest time that user \( i \) has got a rating, and \( t_{i,j} \) is the rating time for item \( j \) by
user $i$. $T$ represents the defined time period. And what value should $T$ be set depends on the experiment results.

Now the user rating can be assigned for a time weight as the following.

$$R_{i,j} = \frac{R_{i,j}}{t_{i,j}}$$

Thus, a user-rating matrix can be obtained, which has got the ratings of the computation results $R_{i,j}$ in equation (3). The matrix is our improved user model for collaborative filtering.

4 Collaborative Filtering Recommendation

Section 3 has discussed the work of user model constructing. And the following work for the proposed algorithm describes how to find the nearest neighbors and predict to give the final recommendation.

4.1 Find the Nearest Neighbors

The main work for finding the nearest neighbors is the similarity computation between users. The notion of similarity is used to identify users having common “preferences”. There are several methods for similarity measurement, among which, Pearson correlation, cosine vector similarity, and revised cosine vector similarity are widely used in collaborative filtering.

1. Cosine Vector Similarity: Users are regarded as vectors in n-dimension space, and the elements of vectors are rating values. Similarity of user $a$ and user $b$ is measured by the cosine value of their rating vectors’ included angle. The expression is given like this:

$$\text{sim}(a, b) = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \cdot \|\vec{b}\|}$$

2. Revised Cosine Vector Similarity: it revises the cosine vector similarity by introducing the average rating of user:

$$\text{sim}(a, b) = \frac{\sum_{j \in I_{ab}} (R_{a,j} - \bar{R}_a)(R_{b,j} - \bar{R}_b)}{\sqrt{\sum_{j \in I_a} (R_{a,j} - \bar{R}_a)^2} \sqrt{\sum_{j \in I_b} (R_{b,j} - \bar{R}_b)^2}}$$

Where $I_{ab}$ stands for the items that both user $a$ and $b$ rated. And $I_a$ and $I_b$ are respectively the items rated by user $a$ and $b$.

3. Pearson Correlation Similarity: Pearson correlation is a correlation-based similarity computation method. And it is described as:

$$\text{sim}(a, b) = \frac{\sum_{j \in I_{ab}} (R_{a,j} - \bar{R}_a)(R_{b,j} - \bar{R}_b)}{\sqrt{\sum_{j \in I_a} (R_{a,j} - \bar{R}_a)^2} \sqrt{\sum_{j \in I_b} (R_{b,j} - \bar{R}_b)^2}}$$
Where $sim(a, b)$ is measurement for similarity between user $a$ and $b$. $R_{a,j}$ is processed rating value of item $j$ by user $a$. $R_{b,j}$ is processed rating value of item $j$ by the remaining user $b$. $\overline{R}_a$ is the average value of processed item ratings by user $a$, while $\overline{R}_b$ is the average value of processed item ratings by user $b$. $I_{ab}$ stands for the items both user $a$ and $b$ have ratings.

In this paper, we compute user similarity on the basis of the method of Pearson Correlation Similarity. Similarity computation aims to find the true neighbors who share the similar preference with the target user, however, in equation (6), when two users happen to have a few rating items in common, it can overestimate the similarity between these two users. In that case, the similarity value we get cannot be a good reference to predicting one user’s future preference, which influences the accuracy of the collaborative filtering algorithm.

To overcome this problem, some changes should be made to the similarity computation method. So an effective weighting factor is added to the equation (6), and the similarity between users is defined as:

$$sim'(a, b) = \text{Min}(|I_{ab}|, \delta) \times sim(a, b)$$

(7)

Where $|I_{ab}|$ stands for the number of items that user $a$ and user $b$ rated in common, and $\delta$ is an adjusting parameter which is used to set the threshold of the number of items that user $a$ and $b$ have in common. In the experiment, we can seek a proper value of $\delta$, so as to get a better result of the algorithm. The added effective weighting factor aims to obtain the more accurate similarity through the number of items users rated in common. If two user have the number of common rated items more than the threshold $\delta$, then the weight is 1, and the similarity depends on Pearson correlation similarity computation. Otherwise, the added effective weighting factor takes effect, and the weight is lower than 1. The less the number of items that the two users both rated, the lower the weight becomes, which is much closer to the reality.

After computing the similarities of users for a target user, we select the top $M$ users as the nearest neighbors who share greater similarities with the target user.

### 4.2 Predict to Give the Final Recommendation

Since we have got the nearest neighbors of a target user, we can compute the weighted average of neighbors’ ratings, weighted by their similarity to the target user. The predicted rating of the target item $j$ by the target user $a$ is given as follows:

$$p(a, j) = \overline{R}_a + \frac{\sum_{i \in UN_a} sim(a, i) \times (R_{i,j} - \overline{R}_i)}{\sum_{i \in UN_a} sim(a, i)}$$

(8)

Here $\overline{R}_a$ is the average rating of target user $a$, and $UN_a$ is the nearest neighbor set of the target user $a$.

This step is as the same with conventional collaborative filtering, the top $N$ predicted ratings are the final recommendation items to the target user.
5 Dataset and Performance Measurement

In this paper, we adopt MovieLens dataset ml to measure the performance of the proposed algorithm. It is collected by MovieLens Recommender System, which devoted to the research of recommendation technology. The data set ml contains 100,000 rating records from 943 users on 1682 movies. The ratings are on a numeric scale from 1 to 5, and each user has at least 20 ratings for the movies. In the experiment, 80% of the data in ml are for the base training of the algorithm, while the left 20% are for the performance measurement.

5.1 Performance Measurement

Several statistical metrics have been proposed for measuring the accuracy of collaborative filtering algorithm [16-18]. And in this paper, Mean Absolute Error (MAE) is employed for the algorithm performance measurement.

MAE evaluates the accuracy of prediction by comparing the average absolute difference between the predicted ratings and the actual user ratings. Assuming that $I_a$ is the set of items that each has a predicted rating and an actual rating, $p(a, i)$ is the predicted rating of target user $a$ to item $i$, and $r_{a,i}$ is the actual rating of user $a$ to item $i$, then the definition for $MAE$ is given as the following:

$$MAE = \frac{\sum_{i \in I_a} |P_{a,i} - r_{a,i}|}{|I_a|} \quad (9)$$

The smaller the $MAE$ is, the more accurate the prediction would be.

1. Impact of parameters

In our proposed algorithm, there are two parameters that can be adjusted. One is time period $T$ in logistic function (2), the other is adjusting parameter $\delta$ of effective weighting factor in equation (7). The impacts that these two parameters have on the $MAE$ are showing in Fig. 2 (a) and Fig. 2 (b) respectively.

![Fig. 2: Results of experiments](image)

In Fig. 2 (a), parameter $T$ is taking the unit of week, which is a conversion to the unit of second in logistic function. The experiment is done in the condition that the user is fixed, and just $T$ is variable, so as to test the $MAE$ results by different value of $T$. The Fig. 2 (a) gives a suggestion that in this proposed algorithm based on MovieLens dataset, a relative prior value for parameter $T$ is 2 weeks. Of course when the algorithm is adopted in a new recommendation system, the value of parameter $T$ should be set by experiment.
The experiment for Fig. 2 (b) is done on one fixed user, and the parameter $T$ takes its prior value of two weeks. Now from Fig. 2 (b), we can see that for the algorithm proposed in this paper, it is better to set the parameter $\delta$ the value from 16 to 20. Also, in the new practical recommender system, the value of $\delta$ should be set in new circumstance by experiment.

2. Comparison to other methods

The comparison experiment is done respectively on three methods. CCF stands for Conventional Collaborative Filtering, while HUMCF, Hybrid User Model based Collaborative Filtering, is the method in reference [11]. It is also based on a hybrid user model, which combines and integrates item ratings, item detailed description and demographic information together to construct the user model. And genetic algorithm is adopted to learn a best feature weight vector in computation of the nearest neighbor set. IUMCF, Improved User-Model-based Collaborative Filtering, is our proposed algorithm in this paper. Here are the MAE results by the three algorithms. Parameters are all getting their prior value, but the number of the nearest neighborhood users is different. The number of nearest neighborhood users ranges from 10 to 50, which is according to times of performance test. The result is shown in Fig. 2 (c). It is clear that both HUMCF and IUMCF perform better than conventional collaborative filtering with lower MAE. However, the proposed IUMCF has the lowest MAE, which indicates that the improved user-model-based collaborative filtering has a better performance on recommendation accuracy.

6 Conclusion

The method presented in this paper improves the collaborative filtering by the user model, which takes rating normalization and time weight into computation, and besides, the effective weighting factor added to the Pearson correlation similarity computation helps to get a higher accuracy in commendation. The experiment shows the proposed method provides a better recommendation in accuracy and efficiency.

References


