Improvement of Matrix Factorization-based Recommender Systems Using Similar User Index

Haesung Lee¹ and Joonhee Kwon²*

¹,²Department of Computer Science, Kyonggi University,
San 94-6, Yiui-dong, Yeongtong-ku, Suwon-si, Gyeonggi-do, Korea
{seastar0202, kwonjh}@kgu.ac.kr

Abstract

Matrix factorization-based approaches have proven to be efficient for recommender systems. However, due to the time complexity in composing recommendations, matrix factorization-based approaches are inefficient in dealing with large scale datasets. In this paper, we present a new similar user index-based matrix factorization approach for large scale recommender systems. Finding similar users is the most time-consuming phase in large scale recommender systems. To reduce time to find the similar users, we propose a similar user index in matrix factorization. This paper describes the index structure and algorithms. Several experiments are performed. The results show that our approach is more efficient in dealing with the large dataset as compared with matrix factorization approach without the similar user index.

Keywords: Recommender systems, Matrix factorization, Large scale dataset, Similar user indexing

1. Introduction

Recommender systems suggest items that are likely to interest the users. Generally recommender systems are based on CF (Collaborative Filtering) which is a technique that automatically predicts the interest of an active user by collecting user’s historical data. However, most recommender systems based on CF suffer from two limitation, data sparsity and prediction accuracy. MF(Matrix Factorization) has become one of the leading techniques for the data sparsity which is main limitation of CF. However, due to the time complexity in composing recommendations, MF-based approaches are inefficient in dealing with large scale datasets.

In this paper, we propose a similar user index in order to efficiently reduce computation time needed in composing recommendations. Also, the use of the similar user index improves the accuracy of prediction because the similar user index is composed with the social network analysis.

The remainder of this paper is organized as follows. Section 2 surveys related works. Section 3 discusses the structure and algorithms of the similar user index-based MF. Section 4 presents the experimental study and results. Finally, Section 5 concludes this paper.

2. Related Works

Collaborative filtering (CF) is the most successful recommendation technology and is used in many of the most successful recommender systems [1]. In the process of CF, it is
one of mainly important phases that a set of similar users is formed. Based on items experienced by similar users, CF calculates the predicted preference score of item $i_j$ for a specific user $u_i$. Then, CF produces recommendations which is a list of $N$ items that the user $u_i$ would like the most. However, CF has two fatal limitations which are respectively related with the data sparsity and scalability [2]. Over the last broad decade many CF algorithms have been proposed in order to solve two challenges. Among many proposed approaches, matrix factorization-based approaches are widely known for its effectiveness.

Matrix factorization (MF) was empirically shown to be a better model than traditional CF approaches. Let $A \in R^{n \times m}$ be the rating matrix in a recommender system, where $m$ and $n$ are the number of users and items, respectively. The problem of matrix factorization for recommender system is presented in the equation (1) [3].

$$\min_{W \in R^{m \times k}, H \in R^{n \times k}} \sum_{(i, j) \in \Omega} (A_{ij} - w_i^T h_j^T)^2 + \lambda (\|w\|_F^2 + \|h\|_F^2)$$ (1)

In the equation (1), $\Omega$ is the set of indices for observed ratings, $\lambda$ is the regularization parameter, and $w_i^T$ and $h_j^T$ are the $i^{th}$ and the $j^{th}$ row vectors of the matrices $W$ and $H$, respectively. The goal of the equation (1) is to approximate the incomplete matrix $A$ by $WH^T$, where $W$ and $H$ are rank-$k$ matrices. In other words, MF learns the model by fitting the previously observed ratings. However, there are overfitting issues because the goal of MF is to generalize those previous ratings in a way that predicts unknown ratings. Besides, the time complexity of MF is $O((\Omega |k^2 + (m + n)k^{-1})$ in each iterated computations. That mean MF is not efficient for dealing with the large-scale dataset.

3. MF-based Recommendations with the Use of Similar User Index

The size of user’s historical data of recommender systems are rapidly increasing. Most MF-based recommender systems suffer from data explosion. The more historical data increase, the more limitations related with the data sparsity and scalability are issued. Besides, it has known that computing similarity between a pair of users is the most time-consuming factor among other operation of recommender algorithms [4]. The huge amount of historical data means more and more computation time is needed in computing user similarity. That is, the large-scale dataset is a main cause of decreasing the performance of recommender systems. For solving the limitations of recommender systems, we try to take the index mechanism to efficiently handle huge amounts of data.

As a complement to conventional information process mechanism, the index mechanism is going to be mainly used in order to improve the performance of systems. To reduce computation time in large scale recommender systems, we propose a similarity user index. Through the similar user index, only useful data for composing recommendations are considered without looking for all data. Therefore, with the user of the similar user index, the computation time of composing similar users is efficiently reduced.

Figure 1 depicts the process of the recommender system which uses the similar user index. In the back-end, the similar user index is composed with the clustering method in advance. Then the similar user index is stored in the storage. In the front-end, when the MF-based recommendation engine receives recommendation requests from client services, the engine accesses the similar user index for looking for similar users. After that, based on similar user’s history data, recommender items are composed and then provided to client services.
For constructing the similar user index, we propose a similar user clustering. The similar user clustering technique works by identifying groups of users who appear to have similar preferences. Since each similar user cluster includes similar users, the recommender algorithm only takes data of users who are belonged in the targeted cluster without considering all users who are included in other clusters.

Among many clustering techniques, similar user clustering is extended from the k-means clustering method [5]. Major task of similar user clustering is measuring the distance between a central user and another user. If the distance between the user $i$ and the central user $j$ in the cluster $c_k$ is shorter than other central users in different clusters, the user $i$ is included in the cluster $c_k$. The distance is considered as similarity of the user and the central user. For measuring the similarity, we use the square of the Euclidean distance measurement. Equation (2) is for measuring similarity $\text{Sim}_{i,j}$ between a pair of users $i$ and $j$, based on Euclidean distance measurement.

$$\text{Sim}_{i,j} = \frac{1}{\sqrt{(TN(i) - MN(i,j)) + (TN(j) - MN(i,j))}}$$

In equation (2), $TN$ is total number of items experienced by a user. $MN(i,j)$ is the number of items experienced by a pair of users $i, j$ in common. Algorithm 1 takes the similar user clustering.

**Algorithm 1. Similar User Clustering**

**Input:** $k$ is the number of clusters, $U$ is the user dataset

**Output:** The set of clusters $C$

1. Randomly assign each user $u_i$ as central user in each cluster $c_i \in C$
2. $\text{centroid}_i \leftarrow \text{Position}(u_i)$
3. repeat
4. for each user node $u_i \in U$ do
5. for each centroid $c_i \in c_k$ do
6. $\text{similarity}_{i,k} \leftarrow \text{Euclidean}(u_i, \text{centroid}_k)$
7. $\text{centroid}_k \leftarrow \text{ReadjustCentroid}(c_k)$
8. if $\text{similarity}_{i,k}$ is biggest value among other clusters then
9. $c_i \leftarrow u_i$
10. until centroids of clusters do not change
11. return $C$
In algorithm 1, \( \text{centroid}_k \) is the position of central user in cluster \( k \). The function, \( \text{Position} \) returns the location in the cluster. \( \text{Euclidean} \) measures the distance between a pair of users of each cluster. \( \text{ReadjustCentroid} \) readjusts the center position in each cluster based on recently computed similarity. Among clusters, the cluster which has the shortest distances between the user \( u_i \) and the central user \( \text{centroid}_k \) is chosen to include the user \( u_i \). The clustering algorithm is iterated until the center position of each cluster does not change.

Through the similar user clustering, similar user clusters are composed. Then, similar users of each cluster are indexed. Similar clusters have their own cluster ID. And then, we use these cluster IDs as an identifier of index. For fast access to similar user index, we define a hash function. Figure 2 depicts the structure of the similar user index. When a user ID is given, similar users of it are retrieved through the hash function. Retrieved set of similar users means a similar user cluster. As shown in Figure 2, the similar user index consists of a cluster ID dictionary and lists of user IDs. The dictionary of cluster IDs records cluster IDs with their own value of the centroid. The hash function computes similarities between the given user ID and the centroid of each similar user cluster. And then it returns a cluster ID with which the given user has the greatest value of the similarity.

![Figure 2. The Structure of the Similar User Index](image)

Ultimately, recommender systems simply accesses the similar user index with a given user ID and the set of similar users are retrieved without needing to consider the whole user data. Then, the recommender system composes recommendations based on retrieved similar users. As a result, the use of the similar user index helps the recommender system recommends items quickly and accurately without needing to consider the whole user data and to spend computation time in composing similar users.

4. Evaluation

In this section, we compare the performance of our approach with that of the previous MF-based recommendation systems without the similar user index. In the experiment, our approach uses the similar user index in MF-based recommender systems. In this paper, our approach is called ‘MF+SimIndex’ and the previous MF-based recommendation systems is called ‘MF’. We evaluated both recommendation accuracy and recommendation time. First, we took the experiment recommendation accuracy for data
sparsity. Second, we experimented the time of composing recommended items with various data size.

Experimental environment is constructed with 2.4 GHZ dual-core Intel Core i5 and 8GB of 1600MHZ DDR3 RAM. In comparative experiments, recommender systems are implemented with Ubuntu 13.1, Hadoop 2.2 and apache mahout 0.8. For handling the large-scale dataset in the distributed environment, we use Hadoop that is the most known distributed computing open framework [6]. Also, apache mahout is the machine learning library and provides useful libraries for recommender systems [7]. For the efficient evaluation, we implemented MF-based recommender algorithms with libraries that are provided by apache mahout. Experimental data sets are composed from data of MovieLens that is online video recommendation site [8].

**Table 1. Experimental Data Sets for the Evaluation of the Recommendation Accuracy**

<table>
<thead>
<tr>
<th>Name</th>
<th>The number of ratings</th>
<th>Sparsity rate (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>1,000,000</td>
<td>95.531</td>
</tr>
<tr>
<td>A2</td>
<td>900,000</td>
<td>85.977</td>
</tr>
<tr>
<td>A3</td>
<td>810,000</td>
<td>77.380</td>
</tr>
<tr>
<td>A4</td>
<td>729,000</td>
<td>69.642</td>
</tr>
<tr>
<td>A5</td>
<td>656,100</td>
<td>62.677</td>
</tr>
<tr>
<td>A6</td>
<td>590,490</td>
<td>56.410</td>
</tr>
<tr>
<td>A7</td>
<td>531,441</td>
<td>50.769</td>
</tr>
<tr>
<td>A8</td>
<td>478,296</td>
<td>45.692</td>
</tr>
<tr>
<td>A9</td>
<td>430,467</td>
<td>41.122</td>
</tr>
<tr>
<td>A10</td>
<td>387,420</td>
<td>37.010</td>
</tr>
</tbody>
</table>

In the case of the first experiment part, we measure RMSE of two recommender systems respectively to evaluate the performance of the recommendation accuracy. RMSE (root-mean-square error) is the most widely used in evaluating the accuracy of recommendations. If the value of RMSE is high, the accuracy of prediction is considered as being incorrect. Also, we synthetically generate 10 data sets varying by data sparsity. Table 1 lists generated 10 data sets for evaluating the accuracy of recommendations.

Data sparsity rate means the degree of data sparsity. If the rate of data sparsity is low, there are much missing data. Missing data is a main cause of reducing the accuracy of recommendations. Figure 3 shows experimental results of evaluating the performance of the accuracy of recommendations with 10 data sets. With the experimental result, it is clear that the recommender system with the use of similar user index performs better than the MF-based recommender system in which the similar user index is not used. As a result, the use of similar user index improves the accuracy of recommendations regardless of data sparsity. This advantage of the similar user index is mainly attributed to the similar user clustering because similar user clustering is not affected by missing data.
In the second part of experiment, we measure the response time of two recommender systems. The response time of the recommender system means the needed computation time in composing recommendations for a specific user. If the response time is fast, the recommender system is considered to have the good performance. For the experiment about the response time, we also synthetically generate 10 datasets with various data sizes. Table 2 shows those 10 data sets.

### Table 2. Experimental Data Sets for Evaluating the Response Time of Recommendations

<table>
<thead>
<tr>
<th>Name</th>
<th>The number of users</th>
<th>Data size (MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>B1</td>
<td>3706</td>
<td>1.040</td>
</tr>
<tr>
<td>B2</td>
<td>3595</td>
<td>0.910</td>
</tr>
<tr>
<td>B3</td>
<td>3433</td>
<td>0.848</td>
</tr>
<tr>
<td>B4</td>
<td>3227</td>
<td>0.784</td>
</tr>
<tr>
<td>B5</td>
<td>2952</td>
<td>0.725</td>
</tr>
<tr>
<td>B6</td>
<td>2657</td>
<td>0.666</td>
</tr>
<tr>
<td>B7</td>
<td>2351</td>
<td>0.610</td>
</tr>
<tr>
<td>B8</td>
<td>2069</td>
<td>0.578</td>
</tr>
<tr>
<td>B9</td>
<td>2041</td>
<td>0.539</td>
</tr>
<tr>
<td>B10</td>
<td>1731</td>
<td>0.493</td>
</tr>
</tbody>
</table>

In the case of the experiment about the response time, we measure the time when the value of RMSE is converged in the certain range. The range is $0.68 \leq \text{RMSE} \leq 0.6$. Fig. 4 comparatively shows the experimental results between two recommender systems. We can see that the recommender system with the use of similar user index is also superior to the recommender system in which only general MF is applied without the use of similar user index. In order words, the use of similar user index improve the response time of recommendation in all experimental data sets. Besides, the recommender system with the use of similar user index is relatively not affected by data size. In the case of the recommender system which does not use the similar user index, however, the more data size increase, the more the response time is slower.
Therefore, we know that the use of similar user index improves the performance of recommender systems regardless of data sizes. The improvement of the response time is mainly contributed to the use of the similar user index. With the given user ID, the recommender system simply access the similar user index and received the set of similar users without needing to consider the whole user data and to compute the similarity between users.

5. Conclusion

Data sparsity and prediction quality have been recognized as one of the crucial challenges that most recommender systems confront. Matrix factorization (MF) has become one of the leading techniques for solving the issues of data sparsity. However, due to the time complexity in composing recommendations, MF-based approaches are inefficient in dealing with a huge amount of historical data. In this paper, we propose a similar user index. The similar user index is efficiently used for reducing useless data in the calculation for composing recommendations. With the given user ID, the recommender system simply access the similar user index and received the set of similar users without needing to consider the whole user data and to compute the similarity between users. With the experimental results, we knew that the use of similar user index improves the accuracy of recommendations without being affected by data sparsity. Also, the similar user index helps the recommender system composes recommendation in faster way regardless of increasing data sizes. In conclusion, the use of proposed approach improves the performance of the MF-based recommender systems. Also, the similar user index is usefully applied in the large-scale recommender systems that handle a huge amount of data.

Acknowledgments

This work was supported by the Gyonggi Regional Research Center (GRRC) of Korea and Contents Convergence Software (CCS) research center of Korea.

References


Authors

Lee Haesung, is a Ph.D. candidate of Computer Science at Kyonggi University, Korea. Her research areas include Context-aware Computing, Social Network, Information Retrieval and Mobile Computing. She works for software development on areas of data search in ubiquitous environment. She received her B.S., M.S. in Computer Science from Kyonggi University, Korea. Contact her at seastar0202@kyonggi.ac.kr.

Kwon Joonhee, is a professor of Computer Science at Kyonggi University, Korea. She was a visiting research professor at the Computer Science Department at New Jersey Institute of Technology. Her research areas include Context-aware Computing, Information Retrieval, Social Network, Web 2.0 and Mobile Database. Her research projects focus on areas of data search using social network and Web 2.0 in ubiquitous environment. She received her B.S., M.S. and Ph.D. in Computer Science from Sookmyung Women’s University, Korea. Contact her at kwonjh@kyonggi.ac.kr.