Postprint: Collaborative Filtering Algorithm Combining Rating Scale Factor and Item Attributes

Authors: Li Shuzhi, Zhijun Li, Deng Xiaohong

Date: 2019-01-03T00:00:00+00:00

Abstract

To address the problems of sparse user rating matrices and the neglect of relationships among item attributes in traditional collaborative filtering algorithms, we propose a collaborative filtering algorithm that integrates rating proportion factors and item attributes. First, we derive a ratio matrix of common versus non-common rating users between items from the rating matrix, thereby enhancing the influence of co-rated users and mitigating errors in item similarity calculation caused by the sparsity of the user-item rating matrix. Subsequently, we quantify item attributes to determine their influence weights on item similarity, which improves the accuracy of similarity computation. Based on these two aspects, we propose an algorithm that employs a combination of rating proportion factors and item attribute weights as the weighting scheme for item similarity. Experimental results show that the algorithm achieves improvements of 5.1% and 4.7% in recall and precision, respectively, compared to existing methods, and is applicable to personalized recommendations on e-commerce websites.

Full Text

Preamble

Collaborative Filtering Algorithm Combined with Score Scale Factor and Item Attribute

Li Shuzhi , Li Zhijun , Deng Xiaohong (College of Information Engineering; College of Applied Science, Jiangxi University of Science & Technology, Ganzhou Jiangxi 341000, China)

Abstract: Traditional collaborative filtering algorithms suffer from the sparsity of user rating matrices and fail to consider relationships between item attributes. To address these issues, this paper proposes a collaborative filtering algorithm

that integrates a rating proportion factor and item attributes. First, the algorithm derives a ratio matrix of common versus non-common rating users between items from the rating matrix, thereby increasing the influence of co-rated users and reducing errors in item similarity calculation caused by matrix sparsity. Second, item attributes are quantified to determine their influence weights on item similarity, improving the accuracy of similarity computation. Based on these two aspects, a novel algorithm is proposed that combines the rating proportion factor and item attribute weights as item similarity weights. Experimental results demonstrate that the algorithm improves recall and precision by 5.1% and 4.7% respectively compared to existing methods, making it suitable for personalized recommendations on e-commerce websites.

Keywords: collaborative filtering; sparse matrix; scoring scale factor; item attribute

0 Introduction

With the continuous expansion of e-commerce scale and rapid growth in product variety, the "information overload" problem has emerged. To address this challenge, personalized recommendation systems have arisen \cite{1~4}. Collaborative filtering recommendation algorithms are currently the most widely used personalized recommendation algorithms, valued for their excellent speed and robustness in the global Internet domain. The principle of collaborative filtering is to recommend information that users may find interesting based on the preferences of like-minded groups with shared experiences. Individuals provide feedback (such as ratings) through a collaborative mechanism, which is recorded to achieve filtering and help others screen information. Collaborative filtering algorithms can be divided into two categories: user-based collaborative filtering and item-based collaborative filtering \cite{5~8}, both belonging to nearest-neighbor collaborative filtering recommendations that generate recommendations for users through ratings from multiple nearest neighbors with similar preferences.

However, as the scale of users (or items) expands dramatically, data becomes increasingly sparse, and collaborative filtering recommendation algorithms exhibit several defects, primarily including inaccurate recommendations caused by matrix sparsity and the item cold-start problem—how to recommend for newly joined users. To address these issues, many researchers have improved and refined collaborative filtering recommendation algorithms from various perspectives. Ha et al. [**?**] proposed constructing a project network graph from items rated by individual users, predicting preferences for unrated items based on the user' s rated items in the project network graph, and generating recommendations through descending ranking, which improved recommendation accuracy. However, when users have rated few items or new users have no ratings, the project network graph cannot be constructed, leading to a cold-start problem.

Yagci et al. [**?**] proposed using frequent subsets from rated items to replace traditional item similarity matrices, reducing computational complexity and avoiding errors from traditional similarity matrices while improving recommendation quality. However, the support threshold requires dynamic iteration, limiting the algorithm' s generality. Polato et al. [**?**] proposed a kernel-based collaborative filtering algorithm that considered the long-tail distribution of item popularity and individual user preferences when making recommendations, achieving improvements in precision and recall. Kong et al. [**?**] presented recommendation results to users in the form of tag-weighted scores with reasonable explanations, demonstrating the effectiveness of recommendations. Yu et al. [**?**] proposed a new item similarity measure composed of rating similarity and structural similarity, which penalized inverse item frequency for active users and considered the impact of co-rated users on similarity, effectively improving recommendation accuracy. However, this algorithm did not consider relationships between item attributes. Zhou et al. [**?**] proposed an online collaborative filtering algorithm based on a confidence-weighted bias model that introduced user personal interests into the similarity calculation formula, improving similarity computation accuracy, alleviating data sparsity, and enhancing recommendation accuracy, but failed to consider connections between user and item attributes.

To address the failure to consider item attributes and the item cold-start problem in the aforementioned literature when calculating similarity, this paper proposes a collaborative filtering algorithm that uses item attribute weights and user rating proportion factors as item similarity weights. This overcomes the inaccuracies in traditional item similarity calculations that fail to consider intrinsic relationships between item attributes and the impact of matrix sparsity. The algorithm simultaneously addresses the item cold-start problem by identifying users similar to most users in the system and recommending items selected by these users to new users.

1.1 User-Item Rating Data Table Construction

Assume we have a user set, an item set, and S_{ui} represents user u' s rating for item i. User rating values range from 1 to 5, with unrated items marked as "-" in the rating data.

The constructed user-item rating data table S is shown in Table 1.

Table 1 User-item score table
Table 1 User-item score table

1.2 Similarity Calculation Methods

Similarity calculation is a critical step in collaborative filtering recommendation algorithms. The most commonly used similarity calculation methods include

cosine similarity, Pearson correlation, and adjusted cosine similarity. In subsequent experiments, this paper will use Pearson correlation to calculate similarity between items.

The similarity between items i and j calculated using Pearson correlation coefficient is shown in Equation (1).

Reasonable. To solve this problem, the concept of a ratio factor between the number of common and non-common rating users for two items is introduced. The definition of the user rating proportion factor is shown in Equation (3).

Where: Factor is the user rating proportion factor with a value range of [0,1]; N_i is the number of users who rated target item i; N_j is the number of users who rated similar item j; $N_{i,j}$ is the number of users who rated both items i and j; $N_i + N_j - 2 \times N_{i,j}$ represents the number of non-common rating users after removing co-raters from those who rated items i and j.

Traditional item-based collaborative filtering algorithms fail to consider intrinsic relationships between item attributes. Therefore, this paper introduces the concept of item attribute weights. For item attributes, based on the relationship between each item and its most similar item' s attributes, item attributes are quantified and assigned a weight. The steps for quantifying item attribute weights are as follows:

a) Transpose the user-item rating matrix data table from Table 1 into an item-user rating matrix data table. Calculate the most similar item for each item using similarity calculation Equation (1), with the most similar item set denoted as belonging to the item set I defined in 1.1.

b) Assume the item attribute table is as shown in Table 2 .

Table 2 Item attribute table
Table 2 Item attribute table

Where: $\sim$ represents k attributes of an item, with 1 indicating the item has the corresponding attribute and 0 indicating it does not.

Based on Table 2, item attribute similarity can be calculated. The method for calculating attribute similarity between items i and j is shown in Equation (4).

Where: U_{ij} represents the set of common users for items i and j; r_{ui} and r_{uj} represent user u's rating values for items i and j respectively; $\bar{r}_i$ and $\bar{r}_j$ represent the average ratings for items i and j respectively.

1.3 Recommendation Set Generation

Recommendation set generation is divided into user-based and item-based approaches. User-based methods predict ratings for unrated items and rank them in descending order for recommendation. The item-based approach for generating recommendation sets is as follows: given user u and user u' s preference matrix for m items as shown in Equation (2).

First, remove items already rated by user u from the calculation results of Equation (2). Then, sort the remaining items in descending order of user preference and select the top-N ranked items to form the recommendation set for user u. In subsequent experiments, this paper will use the item-based recommendation set generation approach.

2.1 Improved Item Similarity Calculation Method

Traditional similarity measurement methods yield excessively high similarity between items when the number of common users who rated both items is very small, making the results unreasonable according to Equation (1). To address this issue, the concept of a ratio factor between the number of common and non-common rating users for two items is introduced. The definition of the user rating proportion factor is shown in Equation (3).

Where: Factor is the user rating proportion factor with a value range of [0,1]; N_i is the number of users who rated target item i; N_j is the number of users who rated similar item j; $N_{i,j}$ is the number of users who rated both items i and j; $N_i + N_j - 2 \times N_{i,j}$ represents the number of non-common rating users after removing co-raters from those who rated items i and j.

Traditional item-based collaborative filtering algorithms fail to consider intrinsic relationships between item attributes. Therefore, this paper introduces the concept of item attribute weights. For item attributes, based on the relationship between each item and its most similar item' s attributes, item attributes are quantified and assigned a weight. The steps for quantifying item attribute weights are as follows:

a) Transpose the user-item rating matrix data table from Table 1 into an item-user rating matrix data table. Calculate the most similar item for each item using similarity calculation Equation (1), with the most similar item set denoted as $\{i_1, i_2, i_3, ..., i_m\}$ belonging to the item set I defined in Section 1.1.

b) Assume the item attribute table is as shown in Table 2 .

Table 2 Item attribute table
Table 2 Item attribute table

Where: $p_1 \sim p_k$ are k attributes of an item, with 1 indicating the item has the corresponding attribute and 0 indicating it does not.

Based on Table 2, item attribute similarity can be calculated. The method for calculating attribute similarity between items i and j is shown in Equation (4).

Where: PN_i and PN_j represent the quantities of attributes possessed by items; $PN_i \cap PN_j$ represents the quantity of attributes common to both items i and j; $PN_i \cup PN_j$ represents the total quantity of attributes possessed by items i

and j. For example, to calculate attribute similarity between items i and j, if item i has attributes $\{p_1, p_3, p_4\}$ and item j has attributes $\{p_1, p_2, p_3, p_4\}$, the calculation method is shown in Equation (5).

c) Perform full permutations of $1 \sim k$ assigned to $p_1 \sim p_k$, assigning each attribute a weight, then recalculate the similarity between each item and its most similar item using Equation (4). The similarity between each item and its most similar item under each permutation is denoted as M_i, with the similarity calculated under the 1st permutation denoted as $M_1^{(1)}$. Other items are marked similarly to item 1. The sum of similarities for all items under the y-th permutation is denoted as $V(y)$. For example, the calculation for item 1 and its most similar item under the y-th permutation is shown in Equation (6).

d) The maximum value among all permutation cases is calculated as shown in Equation (7).

Under the h-th full permutation, the sum of similarities reaches its maximum value, where most items achieve maximum similarity with their similar items. That is, under this permutation, $p_1 \sim p_k$ represents the optimal solution for user attribute weights, making recommendations more realistic.

e) Based on the attribute weights obtained in step d), the calculation of item attribute similarity is shown in Equation (8).

Where w_p represents the weight of a particular attribute of an item.

Finally, based on the introduced user rating proportion factor and item attribute weights, the final improved item similarity is derived as shown in Equation (9).

Where $Peasim(i, j)$ is the item similarity obtained using Pearson correlation coefficient from Equation (1), and $EIsim(i, j)$ is the final improved item similarity calculation formula.

The improved algorithm first needs to calculate similarity between n items, then compute on user rating vectors of dimension m, resulting in a time complexity of $O(n \times n \times m)$. Since m and n are of the same order of magnitude, the final time complexity is $O(n^3)$.

2.2 Item Cold-Start Solution

To address the item cold-start problem, this paper introduces the concept of the intersection of similar users for each user. The specific definition is as follows: Let the user set be $\{u_1, u_2, u_3, ..., u_n\}$. Use Equation (1) to calculate the top k most similar users for user i, denoted as $\{u_{i1}, u_{i2}, u_{i3}, ..., u_{ik}\}$. The collection of similar user sets is $\{U_1, U_2, U_3, ..., U_n\}$. Record the L users with the highest appearance frequency in the Union set $\{union_1, union_2, ..., union_l\}$. Recommend items selected by these L users to newly joined system users, with items

aggregated by category and sorted by frequency or total rating in descending order.

2.3 Algorithm Implementation

The flowchart of the improved algorithm is shown in Figure 1.

Figure 1 Algorithm flow
Fig. 1 [Figure 1: see original paper] Algorithm flow

Algorithm: Collaborative Filtering Algorithm Combined with Score Scale Factor and Item Attribute

Input: Item-user rating matrix A, user-item rating matrix B, user set U, item set I, number of recommended items k.

Output: Top-N recommendation set.

a) Calculate item similarity using Pearson correlation coefficient from Equation (1).

b) Calculate item attribute similarity using Equation (8).

c) Calculate rating proportion factor using Equation (3).

d) Select top-N set.

The algorithm pseudocode is as follows:

```
1. for i_index, i in enumerate(I) do
2.    for j_index, j in enumerate(I) do
3.        // Item similarity = Pearson coefficient × attribute similarity × rating proportio
4.        C[i_index][j_index] = getPearson(i,j) * getProperty(i,j) * getFactor(i,j);
5.    end for
6. end for

7. // Calculate each user's preference degree for items
8. for u in U do
9.    result[0..I.length]; // User's preference degree for all items
10.   for i in I do
11.       // Preference degree = item similarity matrix · user rating matrix
12.       result[i] = dot(C[i], B[u]);
13.   end for
14.   // Sort user preferences in descending order and select top k items
15.   top-N = sorted(result.value, reverse=True, num=k);
16. end for
```

3.1 Experimental Environment

1) Experimental Datasets and Platform

The experiments employ two classic collaborative filtering datasets: MovieLens [**?**] and Jester. The MovieLens dataset contains personal information for 943 users (age, gender, occupation, etc.), basic information for 1,628 movies, and 100,000 user-movie ratings with a range of 1-5. The dataset sparsity is $1 - 100000/(943 \times 1682) = 93.7\%$. The Jester dataset contains 24,983 users' ratings for 100 jokes, with a total of 1,761,439 ratings ranging from -10 to 10. The dataset sparsity is $1 - 1761439/(24983 \times 100) = 29.5\%$. Experiments were conducted on a Windows 10 operating system using Python 3.6.4.

2) Experimental Evaluation Metrics

Recommendation system evaluation standards mainly include statistical accuracy metrics and decision support accuracy metrics. The commonly used statistical accuracy metric is Mean Absolute Error (MAE), while decision support accuracy metrics primarily include recall, precision, and F1-measure \cite{16~18}. This paper adopts precision and recall to evaluate experimental results.

Assuming user u' s preferred item set in the test set is T_i and the system' s recommended item set is N_i, the recommendation recall and precision for user u are calculated using Equations (10) and (11) respectively.

The overall system precision and recall are computed by calculating precision and recall for each user in the user set and then averaging them.

3.2 Experimental Results and Analysis

To verify the effectiveness of the proposed algorithm, training and testing were conducted on the training and test sets of MovieLens and Jester datasets, with top-N values set to $\{10, 15, 20, 25, 30, 40, 50\}$. Experiments consist of two parts: analysis of results with only the rating proportion factor introduced, and analysis of results combining both rating proportion factor and item attributes. The proposed algorithm is compared with the algorithm from literature [**?**], Pearson correlation coefficient, Sigmoid function correlation improvement algorithm [**?**], and Jaccard coefficient improvement algorithm [**?**].

Experiment 1: Effectiveness Analysis of User Rating Proportion Factor

First, recall and precision are used to verify the method of improving item similarity by introducing the user rating proportion factor from Equation (3), validated using the aforementioned top-N sets. Experimental results on the MovieLens dataset are shown in Figures 2 [Figure 2: see original paper] and 3 [Figure 3: see original paper].

When the ratio of common rating users to non-common rating users between two items is larger, it indicates greater similarity between the two items, thereby avoiding imprecision from relying solely on rating values for similarity calculation and improving the accuracy of item similarity computation.

As shown in Figures 3 [Figure 3: see original paper] and 5 [Figure 5: see original paper], when top-N values range from 10 to 50, recall continuously increases, and the recall obtained after introducing the user rating proportion factor improvement is higher than that of Pearson coefficient and algorithms from literature [**?**, **?**, **?**]. Analysis of Figures 2–5 confirms that introducing the user rating proportion factor is effective.

Experiment 2: Analysis of Combined Item Attributes and Rating Proportion Factor

Then, on Jester and MovieLens datasets, recall and precision are used to verify whether the improved item similarity calculation method combining item attributes and rating proportion factor from Equation (9) is correct, validated using the aforementioned top-N sets. Precision and recall experimental results on the MovieLens dataset are shown in Figures 6 [Figure 6: see original paper] and 7 [Figure 7: see original paper].

As shown in Figures 2 and 4, on both Jester and MovieLens datasets, when top-N ranges from 10 to 50, the precision after introducing the user rating proportion factor improvement is higher than Pearson coefficient and literature [**?**, **?**, **?**]. The declining trend of the improved curve occurs because when top-N is small (e.g., 10 in the tests), high precision is already achieved, causing the denominator in Equation (10) to increase while the numerator remains nearly unchanged, resulting in decreasing precision. At top-N = 10, the improved precision is 2.8% higher than the current better algorithm in literature [**?**].

After introducing the user rating proportion factor, the impact of co-rated users on item similarity calculation is considered. When the ratio of common to non-common rating users is larger, it indicates greater similarity between projects, avoiding imprecision from relying solely on rating magnitudes and improving similarity calculation accuracy.

Figures 4 [Figure 4: see original paper] and 5 [Figure 5: see original paper] show experimental results on the Jester dataset.

As shown in Figures 6 and 8, on both Jester and MovieLens datasets, the combined improvement of rating proportion factor and item attributes can recommend items preferred by users even when top-N is small. Compared with introducing only the user rating proportion factor, recommendation accuracy improves by 1.9% on MovieLens and approximately 4.7% on Jester. The declining trend in Figures 6 and 8 for the proposed method when top-N ranges from 10 to 50 occurs for the same reason as in Experiment 1.

Equation (9) builds upon Equation (3) by additionally introducing item attribute similarity as an item similarity weight, improving the accuracy of item

similarity calculation, reducing errors from using only sparse rating matrices, and enhancing recommendation accuracy.

As shown in Figures 7 and 9 [Figure 9: see original paper], when top-N ranges from 10 to 50, Equation (9) achieves higher recall than Equation (7), literature [**?**, **?**, **?**], and traditional item-CF. Since more recommended items are preferred by users, the numerator in Equation (11) continuously increases, causing the recall curve to show an upward trend. This demonstrates that the improved recommendation combining rating proportion factor and item attributes performs better with the same recommendation quantity.

Comprehensive analysis of Experiments 1 and 2 shows that the proposed collaborative filtering algorithm combining rating proportion factor and item attributes considers both the intrinsic relationships between item attributes and the relationship between common and non-common rating user quantities for the same item. This reduces errors from using only rating matrices and improves recommendation accuracy, achieving high precision with smaller recommendation sets, which meets practical recommendation system requirements.

---

4 Conclusion

To address the limitations of traditional item-based collaborative filtering algorithms that use only the co-occurrence matrix of items and user rating matrix product as the final user preference degree without considering the impact of popular items and intrinsic connections between item attributes, this paper proposes a collaborative filtering algorithm combining rating proportion factor and item attributes. The algorithm uses user rating proportion factor, item attribute weights, and Pearson correlation coefficient to replace the item co-occurrence matrix for calculating item similarity. Experimental results show that the proposed algorithm improves recall and precision compared to existing collaborative filtering recommendation algorithms, proving its correctness. Future work will consider further optimization of the combined rating proportion factor and item attribute collaborative filtering algorithm in terms of recommendation speed and accuracy.

References

[1] Hoic-Bozic N, Dlab M H, Mornar V. Recommender system and Web 2.0 tools to enhance a blended learning model [J]. IEEE Trans on Education, 2016, 59(1): 39-44.

[2] Aguilar J, Valdiviezo-Díaz P, Riofrio G. A General framework for intelligent recommender systems [J]. Applied Computing & Informatics, 2016, 13(2): 147-160.

[3] Liu Ang, Lu Stephen, Zhang Zhinan, et al. Function recommender system product planning design Annals-Manufacturing Technology, 2017, 66(1) 181-184.

[4] Sun Zhoubao, Han Lixin, Huang Wenliang, et al. Recommender systems based on social networks [J]. Journal of Systems & Software, 2015, 99 (1): 109-119.

[5] Sun Ping, Li Zhengyu, Han Ziyang, et al. An overview of collaborative filtering recommendation algorithm [J]. Advanced Materials Research, 2013, 756-759: 3899-3903.

[6] Shi Yue, Larson M, Hanjalic A. Collaborative filtering beyond the user-item matrix: a survey of the state of the art and future challenges [J]. ACM Computing Surveys, 2014, 47(1): 1-45.

[7] Elahi M, Braunhofer M, Ricci F, et al. Personality-based active learning for collaborative filtering recommender systems[C]//Advances in Artificial Intelligence. Cham: Springer International Publishing, 2013:

[8] Kurdija A S, Silic M, Vladimir K, et al. Efficient global correlation measures for a collaborative filtering dataset [J]. Knowledge-based systems, 2018, 147(3): 36-42.

[9] Ha Taehyun, Lee Sangwon. Item-network-based collaborative filtering: A personalized recommendation method based on a user' s item network [J]. Information Processing and Management, 2017, 53(5): 1171-1184.

[10] Yagci A M, Aytekin T, Gurgen F S. Scalable and adaptive collaborative filtering by mining frequent item co-occurrences in a user feedback stream [J]. Engineering Applications of Artificial Intelligence, 2017, 58 (1): 171-184.

[11] Polato M, Aiolli F. Exploiting sparsity to build efficient kernel based collaborative filtering top-N recommendation ChinaXiv 合作期刊 Neurocomputing, 2016, 268 (6) 17-26.

[12] 孔欣欣, 苏本昌, 王宏志, 等. 基于标签权重评分的推荐模型及算法研究 [J]. 计算机学报, 2017, 40(6): 1440-1452. (Kong Xinxin, Su Benchang, Wang Hongzhi, et al. Research on the modeling and related algorithms of label-weight rating based recommendation system [J]. Chinese Journal of Computers, 2017, 40(6): 1440-1452.)

[13] 于金明, 孟军, 吴秋峰. 基于改进相似性度量的项目协同过滤推荐算法 [J]. 计算机应用, 2017, 37(5):1387-1391, 1406. (Yu Jinming, Meng Jun, Wu Qiufeng. Item collaborative filtering recommendation algorithm based on improved similarity measure [J]. Journal of Computer Applications, 2017, 37(5): 1387-1391, 1406.)

[14] Zhou Xiuze, Shu Weibo, Lin Fan, et al. Confidence-weighted bias model for online collaborative filtering [J]. Applied Soft Computing, 2017, 17(4): 1-12.

[15] Harper F M, Konstan J A. The MovieLens datasets [J]. ACM Trans on Interactive Intelligent Systems, 2016, 5(4): 1-19.

[16] 朱郁筱, 吕琳媛. 推荐系统评价指标综述 [J]. 电子科技大学报, 2012, 41(2): 163-175. (Zhu Yuxiao, Lyu Linyuan. Evaluation metrics for recommender systems [J]. Journal of University of Electronic Science and Technology of China, 2012, 41(2): 163-175.)

[17] 肖文强, 姚世军, 吴善明. 一种改进的 top-N 协同过滤推荐算法 [J]. 计算机应用研究, 2018, 35(1): 105-108, 112. (Xiao Wenqiang, Yao Shijun, Wu Shanming. Improved top-N collaborative filtering recommendation algorithm [J]. Application Research of Computer, 2018, 35(1): 105-108, 112.)

[18] Zhang Feng, Gong Ti, Zhao Gansen, et al. Fast algorithms to evaluate collaborative filtering recommender systems[J]. Knowledge-Based Systems, 2016, 96(2): 96-103.

[19] Yang Chong, Yu Xiaohui, Liu Yang, et al. Collaborative filtering with weighted opinion aspects [J]. Neurocomputing, 2016, 210(5): 185-196.

[20] Hernando A, Ortega F. Collaborative filtering based on significances [J]. Information Sciences, 2016, 185(1): 1-17.

Note: Figure translations are in progress. See original paper for figures.