ChinaRxiv

AI translation · View original & related papers at
chinarxiv.org/items/chinaxiv-201811.00147

Postprint: A Hybrid Recommendation Algorithm Integrating Collaborative Filtering and XGBoost

Authors: Cui Yan, Qi Wei, Pang Hailong, Zhao Hui

Date: 2018-11-29T00:00:00+00:00

Abstract

Collaborative filtering is the most widely applied technique in information filtering and recommendation systems; however, the data sparsity problem in data processing affects the accuracy of recommendation algorithms. This paper proposes a recommendation algorithm that integrates collaborative filtering and XGBoost. By leveraging user ratings for items and the inherent characteristics of items themselves, the algorithm uncovers latent relationships between items and users, thereby enhancing recommendation accuracy. Experiments are conducted on the Book-Crossings dataset using Baidu' s deep learning framework PaddlePaddle. The results demonstrate that the proposed algorithm achieves significant improvement in accuracy compared with two algorithms from the literature.

Full Text

A Recommendation Algorithm Fusing Collaborative Filtering and XGBoost

Cui Yan, Qi Wei, Pang Hailong, Zhao Hui
(College of Computer Science and Engineering, Changchun University of Technology, Changchun 130012, China)

Abstract: Collaborative filtering is the most widely used technique in information filtering and recommendation systems. However, it suffers from data sparsity issues during data processing, which affects the accuracy of recommendation algorithms. This paper proposes a recommendation algorithm that fuses collaborative filtering and XGBoost. By leveraging user ratings on items and the inherent characteristics of items themselves, the algorithm mines potential relationships between users and items to improve recommendation accuracy. Experiments were conducted on the Book-Crossings dataset using Baidu' s deep

learning framework PaddlePaddle. Experimental results demonstrate that the proposed algorithm achieves significant improvements in accuracy compared to two algorithms from the literature.

Keywords: XGBoost; collaborative filtering; accuracy; recommendation system

0 Introduction

With the rapid development of network environments in recent years, network information coverage has extended to virtually every aspect of daily life. While the amount of data available online has become increasingly abundant, this has led to exponential data growth. According to statistical results, Facebook' s active users share approximately 684,000 bits of information online every minute, Twitter users post over 100,000 tweets, and 90% of the world' s data was generated between 2010 and 2012. By 2020, the total global information volume is expected to be 22 times that of 2011, reaching 35.2 ZB [**?**]. However, much of this data is irrelevant and redundant, leading to the problem of "information overload" [**?**]. As the online world becomes inundated with information and transitions from the IT (information technology) era to the DT (data technology) era [**?**], recommendation systems [**?**] have emerged as essential tools to help users obtain effective information, serving as an important filtering technology to address information overload.

In the development of recommendation systems, collaborative filtering [**?**] remains dominant. Collaborative filtering can effectively filter information that is difficult for machines to analyze automatically and utilizes feedback from other similar users to improve the speed of personalized learning. However, practical applications face several challenges: data sparsity [**?**, **?**], scalability as data volume increases [**?**], and the cold start problem [**?**]. Data sparsity occurs when users have rated very few items, causing the similarity between users based on these ratings to fall below expected levels and ultimately affecting recommendation accuracy. The scalability problem arises when the data volume is large or when substantial new information is added, making it impossible to identify highly similar user-item relationships in a timely manner and thus preventing real-time recommendations. The cold start problem occurs when new users or items join the recommendation system without historical data, making similarity calculations impossible and preventing effective recommendations.

Numerous studies have addressed these issues. Reference [**?**] proposed a two-stage joint clustering rating method where the dimensionality of the clustered matrix is much smaller than that of the original matrix, reducing computational complexity while improving accuracy. However, this algorithm relies heavily on rating weights, resulting in unreliable predicted ratings. Reference [**?**] proposed using Singular Value Decomposition (SVD) to solve dimensionality and scalability problems, but it is extremely time-consuming for large datasets and impractical for real-world use. Reference [**?**] proposed a matrix factorization rec-

ommendation method that fuses social information, which can more accurately mine user interests and improve recommendation precision through friend relationships. Reference [**?**] explored the application of XGBoost [**?**] in e-commerce product recommendation, but its failure to incorporate users' historical behavioral characteristics led to defective recommendations.

To address data sparsity, this paper proposes a recommendation algorithm that fuses collaborative filtering and XGBoost (eXtreme Gradient Boosting recommendation algorithm with collaborative filtering, XGBCF). The algorithm calculates similarity by constructing user similarity matrices and item similarity matrices through collaborative filtering. Based on nearest-neighbor relationships, it forms similar pair sets to generate training data, then uses XGBoost to fit the training set to obtain optimal weights and learning rates, and finally employs an objective function for classification and recommendation to users, thereby improving recommendation accuracy.

1.1 Collaborative Filtering Algorithm

Collaborative filtering is a commonly used technique in information filtering and information systems, representing a typical manifestation of collective intelligence. Its principle is to analyze and mine users' historical behaviors to discover their preferences and group users based on different interests. Its primary functions are prediction and recommendation. Collaborative filtering is mainly divided into two categories: user-based collaborative filtering and item-based collaborative filtering. User-based collaborative filtering analyzes users' historical behaviors to identify content or items of interest, calculates similarity based on different users' ratings for the same items, and makes recommendations among similar users. Item-based collaborative filtering operates on a similar principle but focuses on items themselves when calculating similarity—that is, it finds similar items based on user preferences. The process involves first calculating the similarity between rated items and items to be predicted, using this similarity as a weight coefficient to control the scores of rated items, and obtaining predicted values for the target items. If the predicted values are close to the actual ratings, the items are combined for recommendation.

Existing basic methods for similarity calculation are primarily vector-based, computing the distance between vectors—the closer the distance, the greater the similarity. In recommendation scenarios, we can treat user preferences for all items as vectors to calculate user similarity or treat all users' preferences for a particular item as vectors to calculate item similarity. Commonly used similarity measures include the Pearson correlation coefficient [**?**] (PCC) and cosine similarity [**?**].

The cosine similarity formula is defined as:

$$sim(u,v) = \frac{\sum_{i \in I} R_{u,i} R_{v,i}}{\sqrt{\sum_{i \in I} R_{u,i}^2} \sqrt{\sum_{i \in I} R_{v,i}^2}}$$

where $sim(u,v)$ represents the similarity between users u and v, I is the set of items rated by both users, $R_{u,i}$ denotes user u' s rating for item i, and $R_{v,i}$ denotes user v' s rating for item i.

The Pearson correlation coefficient is defined as:

$$sim(u,v) = \frac{\sum_{i \in I} (R_{u,i} - \bar{R}_u)(R_{v,i} - \bar{R}_v)}{\sqrt{\sum_{i \in I} (R_{u,i} - \bar{R}_u)^2} \sqrt{\sum_{i \in I} (R_{v,i} - \bar{R}_v)^2}}$$

where U represents the set of all users who have rated items, $\bar{R}_i$ denotes the average rating for item i, and $\bar{R}_j$ denotes the average rating for item j. The Pearson correlation coefficient measures the strength of the linear relationship between two users, with values in the range $[-1, 1]$.

The collaborative filtering prediction rating formula is defined as:

$$\hat{R}_{u,i} = \bar{R}_i + \frac{\sum_{v \in V} sim(u,v) \cdot (R_{v,i} - \bar{R}_i)}{\sum_{v \in V} |sim(u,v)|}$$

where $\hat{R}_{u,i}$ represents the predicted rating of user u for unrated item i, and V denotes the set of nearest-neighbor users who have rated item i.

The Top-N algorithm sorts data according to certain rules. In this paper, the proposed algorithm (XGBCF) calculates predicted ratings for users who have not rated items using Equation (3), sorts these predictions in descending order, and generates a recommendation list for the user.

1.2 XGBoost Algorithm

XGBoost, short for eXtreme Gradient Boosting, is an improvement on the Boosting algorithm proposed by Dr. Tianqi Chen at the University of Washington in 2014 based on the GBDT algorithm. Its internal decision trees are regression trees, characterized by high speed, good performance, ability to handle large-scale data, and support for custom loss functions. It employs iterative computation of weak classifiers to improve classification accuracy.

The complexity of a tree is defined as:

$$\Omega(f_t) = \gamma T + \frac{1}{2} \lambda \sum_{j=1}^{T} w_j^2$$

where q represents the tree structure and w represents the leaf weights.

The improved complexity function is:

$$\Omega(f_t) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$$

where T is the number of leaf nodes, and w_j represents the L2 norm squared of the output scores on each leaf node. The parameter λ controls the weight of this term in the final model formula, enabling measurement of model complexity and effective prevention of overfitting.

The rewritten objective function is:

$$Obj^{(t)} = \sum_{i=1}^{n} [l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t)] + constant$$

where l is the loss function, Ω is the regularization term including L1 and L2 regularization, and *constant* is a constant term. As shown in the equation, the final objective function depends on the first and second-order derivatives of the error function for each data point.

Using Equation (6), the objective function is rewritten again, where l is defined for the sample set on each leaf node:

$$Obj^{(t)} \approx \sum_{i=1}^{n} [l(y_i, \hat{y}_i^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \Omega(f_t) + constant$$

where g_i and h_i are defined as:

$$G_j = \sum_{i \in I_j} g_i, \quad H_j = \sum_{i \in I_j} h_i$$

Taking the derivative of $Obj^{(t)}$ with respect to w_j and setting it to zero yields:

$$w_j^* = -\frac{G_j}{H_j + \lambda}$$

Substituting the optimal solution back into the objective function gives:

$$Obj^* = -\frac{1}{2} \sum_{j=1}^{T} \frac{G_j^2}{H_j + \lambda} + \gamma T$$

where T is the number of leaf nodes, and γ is a weight coefficient to prevent overfitting.

2.1 Algorithm Idea

Definition 5 User-Item Rating Matrix R, where $R_{p,m}$ represents user p's rating for item m.

Item Rating Matrix

2.3 Algorithm Description

The proposed algorithm consists of three steps: generating the training sample set, performing XGBoost precision classification, and generating recommendations.

1) Generating the Training Sample Set

Since each user has rated only a limited number of items, the user behavior matrix contains numerous null values when constructed. To address this issue, the algorithm mines relationships between users and user-item relationships based on users' historical behaviors. It constructs user pairs and item pairs from rated items, calculates similarity for each pair to build similarity matrices, computes the similarity between each user and other users/items, sorts the results, and forms nearest-neighbor sets. Using the collaborative filtering prediction rating formula, it obtains predicted ratings, then employs the XGBoost algorithm for classification, calculates classification error rates, and updates weights and learning rates to identify misclassified samples and reset their weights, thereby improving algorithm precision. Finally, it uses Top-N to generate recommendation lists.

Input: User feature matrix $ufeature$ after data cleaning, item feature matrix $ifeature$ after data cleaning, user-item rating matrix R, user set U, item set I.

Output: Training sample set.

Processing Flow:

a) In the user-item rating matrix, identify the set of items I that have been rated and the set of users U who have rated items.

b) Split the above I and U into user pair sets $\{upair|upair = \langle u_m, u_n\rangle, u_m, u_n \in U\}$ and item pair sets $\{ipair|ipair = \langle i_a, i_b\rangle, i_a, i_b \in I\}$.

c) For each data pair in $upair$ and $ipair$, locate their corresponding positions in $feature$ and $ufeature$, then use Equation (1) to calculate user similarity $sim(u_m, u_n)$ and Equation (2) to calculate item similarity $sim(i_m, i_n)$.

d) Loop through the first three steps to obtain all user and item similarities, constructing user similarity matrix $sim(u, u)$ and item similarity matrix $sim(i, i)$.

e) Compute the similarity of each user in the user similarity matrix, sort the results, and combine the top n most similar users into the user nearest-

neighbor set $N_u = \{u_1, u_2, ..., u_n | u_1, u_2, ..., u_n \in U\}$. Similarly, combine the item nearest-neighbor set $N_i = \{i_1, i_2, ..., i_n | i_1, i_2, ..., i_n \in I\}$.

f) Traverse the user-item rating matrix R, select the nearest-neighbor set N_i corresponding to item i. Use Equation (3) to calculate the predicted rating $\hat{R}_{p,n}$ of user p for item n, and record the difference between the predicted value and the original rating in the user-item matrix as $x_{u,i}$. Similarly, select the nearest-neighbor set N_u corresponding to user u, use Equation (3) to calculate the predicted rating $\hat{R}_{p,n}$ of user p for item n, and record the difference as $x_{i,u}$. Finally, combine the obtained differences into a new dataset $data = \{\langle x_{u,i}, x_{i,u}, R_{u,i} \rangle | u \in U, i \in I, R_{u,i} \in R\}$.

2) XGBoost Classification

Input: Training sample set.

Output: Classification results.

Processing Flow:

a) Assign equal weights w to all training samples.

b) Use the XGBoost algorithm for classification, iterating m times, and calculate the classification error rate using:

$$err_m = \frac{\sum_{i=1}^{n} w_i^{(m)} I(y_i \neq G_m(x_i))}{\sum_{i=1}^{n} w_i^{(m)}}$$

where $w_i^{(m)}$ represents the weight of the i-th sample, and G_m represents the m-th classifier.

c) Update the learning rate:

$$\alpha_m = \frac{1}{2} \log \left(\frac{1 - err_m}{err_m} \right)$$

This pruning step prevents overfitting while reducing tree size and ensuring iterative sample space, allowing subsequent learning rate calculations to be updated through iteration.

d) Repeat steps b) and c) to obtain corresponding error rates, then reset the weight of the n-th sample as:

$$w_i^{(m+1)} = w_i^{(m)} \exp(-\alpha_m y_i G_m(x_i))$$

e) After multiple iterations, identify misclassified samples by updating learning rates and weight coefficients and reset their weights, yielding more precise classification results.

3) Generating Recommendation Lists

Input: Dataset *data*, user u' s data in *data*.

Output: Recommendation list for user u.

Processing Flow:

a) Construct a regression tree with dataset labels as root node input.

b) Build an XGBoost model using the objective function shown in Equation (8). Due to XGBoost's support for custom loss functions, the loss function is defined here as logistic loss:

$$l(y_i, \hat{y}_i) = y_i \log(1 + e^{-\hat{y}_i}) + (1 - y_i) \log(1 + e^{\hat{y}_i})$$

where y_i represents the actual value and $\hat{y}_i$ represents the predicted value.

c) Compute the second-order partial derivative of the loss function, update the learning rate using Equation (10), adjust weight parameters, and substitute into Equation (8) to obtain the XGBCF model.

d) Use the XGBCF model to predict ratings for items not yet rated by user u, and generate a recommendation list using Top-N.

2.2 Algorithm Related Definitions

Definition 1 User set $U = \{u_1, u_2, ..., u_n\}$, containing n users, where subscripts represent user indices.

Definition 2 Item set $I = \{i_1, i_2, ..., i_m\}$, containing m items, where subscripts represent item indices.

Definition 3 shows the cross-product of items and features, where $if_{m,n}$ represents the n-th feature of the m-th item.

Item Feature Matrix $ifeature$

Definition 4 User feature matrix $ufeature$ where $uf_{p,q}$ represents the cross-product of user p and feature q, indicating the q-th feature of the p-th user.

User Feature Matrix $ufeature$

3.1 Experimental Data

The dataset used in this paper is the Book-Crossing dataset provided by Cai-Nicolas Ziegler [**?**], which consists of three parts: user information dataset, book information dataset, and user rating dataset. The user dataset includes user ID and demographic information (location, age) as shown in . The book dataset is identified by ISBN, with invalid information removed from the dataset. Additionally, some content-based information is provided ('book title' , 'author'

, 'publication date' , 'publisher') as shown in . The rating dataset contains 1,149,780 rating entries from 278,858 users on 271,379 books, with explicit ratings (Book-Rating) on a scale of 1-10 (higher values indicate higher interest). In the experiments, 75% of the dataset was used as the training set and 25% as the test set.

Example of User Information Characteristics

Example of Book Information Characteristics

3.2 Metrics

This experiment uses Mean Absolute Error (MAE) to measure classification accuracy. MAE is normalized over the rating interval and is defined as:

$$MAE = \frac{1}{|E|} \sum_{(u,a) \in E} |r_{u,a} - \hat{r}_{u,a}|$$

where $r_{u,a}$ represents user u' s actual rating for item a, $\hat{r}_{u,a}$ represents the predicted rating, and E represents the test set. Smaller values indicate more accurate predictions and better recommendation performance.

3.3 Experimental Results and Analysis

Since different parameters affect algorithm performance, we first determine a series of optimal parameters by fitting the dataset before comparing algorithms. As mentioned earlier, user similarity in this paper is calculated using the Pearson correlation coefficient, while item similarity is calculated using modified cosine similarity.

1) Learning Rate

The learning rate controls the speed at which the algorithm adjusts weights based on the loss gradient. A learning rate that is too small leads to slow convergence, while one that is too large causes the cost function to fluctuate. Here, we vary the learning rate from 0.1 to 1. As shown in [Figure 1: see original paper], when the learning rate $\alpha = 0.3$, the Mean Absolute Error (MAE) reaches its global minimum and no longer changes with increasing learning rate.

[Figure 1: see original paper] Influence of learning rate on mean absolute error

2) Number of Nearest Neighbors

The number of nearest neighbors is a key factor affecting recommendation accuracy. With the optimal learning rate fixed at $\alpha = 0.3$, we vary the number of nearest neighbors from 10 to 200 to find the optimal value. As shown in [Figure 2: see original paper], as the number of user nearest neighbors increases, the MAE of the proposed fusion algorithm gradually decreases and recommendation accuracy improves, reaching a global minimum at $k = 125$ and remaining

stable thereafter. Similarly, as shown in [Figure 3: see original paper], when the number of item nearest neighbors k varies from 10 to 75, the MAE of XGBCF fluctuates, but as k increases further, MAE continues to decrease and recommendation accuracy improves, reaching its minimum at $k = 125$ and maintaining stable recommendation precision.

[Figure 2: see original paper] Influence of user neighbour on mean absolute error

[Figure 3: see original paper] Influence of item neighbour on mean absolute error

3) Algorithm Analysis and Comparison

Since the proposed algorithm (XGBCF) incorporates decision trees, it performs initial classification of users and items during construction, reducing subsequent classification processes. During fitting, it can find optimal parameters, achieving more precise classification and improved recommendation accuracy. Comparing the proposed algorithm with those in references [**?**] and [**?**], [Figure 4: see original paper] shows that the MAE of the XGBCF algorithm is lower than that of the algorithms mentioned in the literature, demonstrating significantly improved accuracy.

[Figure 4: see original paper] Comparison of algorithm results

Therefore, for the XGBCF algorithm, we set the number of user nearest neighbors to 125, the number of item nearest neighbors to 125, and the learning rate $\alpha = 0.3$ for the experiments.

4 Conclusion

This paper proposes a recommendation algorithm that fuses collaborative filtering and XGBoost. The algorithm constructs user and item similarity matrices to calculate similarity, forms similar pair sets based on nearest-neighbor relationships to generate training data, uses XGBoost to fit the training set to obtain optimal weights and learning rates, and employs an objective function for classification. Finally, it generates recommendation lists for users. The proposed algorithm overcomes the problem of insufficient recommendation accuracy caused by data sparsity. Experimental validation demonstrates that its accuracy is significantly improved compared to traditional collaborative filtering. Future work will further investigate feature extraction and security issues in recommendation systems.

References

[1] Gantz J, Reinsel D. The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east. [EB/OL]. (2013-02). https://www.com/collateral/analyst-reports/idc-digital-universe-united-states.pdf.

[2] Meng Xiangwu, Hu Xun, Wang Licai, et al. Mobile recommender systems and their applications [J]. Journal of Software, 2013, 24 (1): 91-108.

[3] Lian Yuming. Human society from IT era to DT era [J]. Business Culture, 2016 (11): 66-69.

[4] Milicevic A K, Nanopoulos A Ivanovic, M. Social tagging in recommender systems: a survey of the state-of-the-art and possible extensions [J]. Artificial Intelligence Review, 2010, 33 (3): 187-209.

[5] Patil V A, Ragha L. Comparing performance of collaborative filtering algorithms [C]// Proc of International Conference on Communication, Information & Computing Technology. 2012: 1-6.

[6] Wang Cheng, Zhu Zhigang, Zhang Yuxia, et al. Recommendation efficiency and personalized improvement of user-based collaborative filtering algorithm [J]. Journal of Chinese Computer Systems, 2016, 37 (3): 428-432.

[7] Ma Hongwei, Zhang Guangwei, Li Peng. Collaborative filtering recommendation algorithm [J]. Journal of Chinese Computer Systems, 2009, 30 (7): 1282-1288.

[8] Liu Jianguo, Zhou Tao, Guo Qiang, et al. Review of individualized recommendation system evaluation methods [J]. Complex Systems and Complexity Sciences, 2009, 6 (3): 1-10.

[9] Yang Diyi, Chen Tianqi, Zhang Weinan, et al. Localimplicit feedback mining for music recommendation [C]// Proc of the 6th ACM Conference on Recommender Systems. New York: ACM Press, 2012: 91-98.

[10] Oh K J, Lee W J, Lim C G, et al. Personalized newsrecommendation using classified keywords to capture user preference [C]// Proc of the 16th International Conference on Advanced Communication Technology. Piscataway, NJ: IEEE Press, 2014: 1283-1287.

[11] Li Cong. Summary of research on scalability of collaborative filtering in electronic commerce [J]. Modern Library and Information Technology, 2010 (11): 37-44.

[12] Sun Xiaohua. The sparsity and cold start problem of collaborative filtering system [D]. Hangzhou: Zhejiang University, 2005.

[13] Wu Hu, Wang Yongji, Wang Zhe, et al. Two-stage joint clustering collaborative filtering algorithm [J]. Journal of Software, 2010, 21 (5): 1042-1054.

[14] Vozalis M G, Margaritis K G. Applying SVD on item-based filtering [C]// Proc of International Conference on Intelligent Systems Design and Applications. Washington DC: IEEE Computer Society, 2005: 464-469.

[15] Liu Huafeng, Jing Liping, Yu Jian. A review of matrix factorization recommendation methods for integrating social information [J]. Journal of Software, 2018, 29 (2): 340-362.

[16] Zhang Hao, Ji Hongchao, Zhang Hongyu. Application of XGBoost algorithm in e-business commodity recommendation [J]. Internet of Things Technologies, 2017, 7 (2): 102-104.

[17] Chen Tianqi, Carlos Guesintr. XGBoost: a scalable tree boosting system [C]// Proc of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York: ACM, 2016: 785-794.

[18] Herlocker J, Konstan J, Borchers A, et al. An algorithmic framework for performing collaborative filtering [C]// Proc of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. New York: ACM Press, 1999: 230-237.

[19] Saltong. The SMART retrieval system-experiments in automatic document processing [M]. Englewood Cliffs, New Jersey: Prentice Hall Inc, 1971.

[20] GroupLens [EB/OL]. http://www.grouplens.org/.

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

Source: ChinaXiv —Machine translation. Verify with original.