ChinaRxiv

AI translation · View original & related papers at
chinarxiv.org/items/chinaxiv-201904.00047

Postprint: Binomial Matrix Factorization Recommendation Algorithm with Rating Bias

Authors: Zhang Xiaohong, Zhang Qizhi, Zhou Yali

Date: 2019-04-01T00:00:00+00:00

Abstract

For the rating prediction problem in recommendation systems, we propose a modified binomial matrix factorization algorithm built upon matrix factorization. Assuming that user ratings for items follow a binomial distribution, differences in users' rating habits and variations in item popularity introduce bias into the user-item rating matrix. By incorporating bias terms to correct matrix factorization and rating prediction, we employ maximum a posteriori estimation for modeling and optimize the model using stochastic gradient descent. Experimental results demonstrate that on the MovieLens 100K dataset, the binomial matrix factorization algorithm with rating bias outperforms the traditional binomial matrix factorization algorithm in terms of recommendation accuracy, offline computation time, and other aspects.

Full Text

Binomial Matrix Factorization Recommendation Algorithm Incorporating Rating Bias

Zhang Xiaohong, Zhang Qizhi, Zhou Yali
(School of Automation, Beijing Information Science & Technology University, Beijing 100192, China)

Abstract: To address the rating prediction problem in recommender systems, this paper implements a modified binomial matrix factorization algorithm based on matrix decomposition. Assuming that user ratings for items follow a binomial distribution, differences in user rating habits and item popularity introduce bias into the user-item rating matrix. By incorporating bias terms to adjust the matrix decomposition and rating prediction, the model is constructed using maximum a posteriori estimation and optimized via stochastic gradient descent. Experimental results on the MovieLens 100K dataset demonstrate that the proposed binomial matrix factorization algorithm with rating bias outperforms tra-

ditional binomial matrix factorization in both recommendation accuracy and offline computation time.

Key words: recommender system; binomial matrix factorization; rating bias

0 Introduction

One hallmark of the big data era is "data rich, but information poor." For information consumers, locating personally relevant content amid massive amounts of data has become extremely difficult. For information producers, capturing user interests and understanding how they evolve to produce engaging content presents a significant challenge [1].

The core solution to this dilemma involves using a "funnel" to reduce the volume of information ultimately presented to users. Recommender systems serve as a crucial tool for addressing this contradiction by filtering and ranking information to deliver useful content to users. The essence of recommender systems lies in connecting users with items through specific methods—discovering user needs and interests, then employing recommendation algorithms to mine potentially interesting items from vast datasets for recommendation. Currently, recommender systems have been successfully applied across numerous Internet domains [14].

Traditional recommendation methods primarily include collaborative filtering techniques [5,6], content-based approaches [7,12], and hybrid methods [8]. Reference [2] provides a formal definition of recommendation algorithms: let U represent the set of users and I the set of items, with f defined as a utility function that calculates the recommendation score of item i for user u. The recommendation algorithm identifies the most interesting item $i \quad I$ for user $u \quad U$ by computing this score, as shown in Equation (1).

A key challenge for recommender system effectiveness is that utility function f is typically defined on a subspace of $U \times I$, yet the algorithm must extrapolate f to the entire $U \times I$ space [14]. For instance, many applications define recommendation scores as user ratings for items. However, in practice, users only rate a small fraction of items, resulting in data sparsity. In real applications, unknown ratings are calculated from known rating data through machine learning algorithms, and through candidate selection and ranking, a Top-N list is ultimately generated for users—this is the extrapolation process. The most classic algorithm in this process is collaborative filtering, which leverages interaction information between users and items to generate recommendations. This includes three main approaches: neighborhood-based methods, latent factor models (LFM), and graph-based random walk algorithms [15]. Among these, the most famous and widely adopted in industry are neighborhood-based methods [5].

Latent factor models have become one of the hottest research topics in recommender systems in recent years, with their core idea being to connect user interests with items through latent features. In recommender systems, latent fac-

tor models align conceptually with matrix factorization models—both complete matrices through dimensionality reduction. The earliest matrix factorization model originated from the mathematical singular value decomposition (SVD). Following the 2006 Netflix Prize, Funk [16] published an algorithm (Funk-SVD) on his blog, sparking intense academic interest in matrix factorization methods. Subsequent improvements include the Biased-SVD algorithm with added bias terms, the SVD++ algorithm incorporating users' historically rated items [10], probabilistic matrix factorization based on the assumption that user rating data follows a normal distribution, and the binomial matrix factorization (BMF) algorithm for discrete rating datasets [11].

This paper focuses on investigating certain issues in the binomial matrix factorization algorithm within latent factor models, conducting offline experiments on the MovieLens 100k movie rating dataset.

1 Traditional Matrix Factorization Algorithms

Matrix factorization algorithms are primarily applied to rating prediction problems. Consider I items and N users, with the $N \times I$ matrix $\mathbf{R}$ representing the rating matrix, where element r_ui denotes user u' s rating for item i. Assuming D latent features for users and items, the $D \times N$ matrix $\mathbf{p}$ represents the user latent feature matrix, with $\mathbf{p\_\_u}$ denoting user u' s latent feature vector; the $D \times I$ matrix $\mathbf{q}$ represents the item latent feature matrix, with $\mathbf{q\_\_i}$ denoting item i' s latent feature vector. The predicted rating for user u on item i is therefore calculated through the dot product of their latent vectors. Consequently, the rating matrix $\mathbf{R}$ can be approximated by two low-rank matrices $\mathbf{p}$ and $\mathbf{q}$. This paper argues that as long as the approximation error is minimized, predicted values can replace actual values, transforming the rating prediction problem into an optimization problem.

SVD is one of the most classical matrix factorization algorithms, with the basic formulation $\mathbf{R} = \mathbf{U\Sigma V}\hat{}\mathrm{T}$, where $\mathbf{U}$ and $\mathbf{V}$ represent user and item latent factor matrices, respectively, and $\mathbf{\Sigma}$ is a diagonal matrix of singular values. Based on the premise that a small portion of the data carries most of the information while the rest is either noise or irrelevant, we only need to select the top D factors to represent a user or item. In the Netflix Prize, SVD and its variants demonstrated excellent performance [13]. Although simple to implement, SVD requires a dense matrix—all elements must be non-null for decomposition to proceed. Traditional SVD algorithms typically address this by filling missing values in the rating matrix with the global average. However, due to the large numbers of users and items, traditional matrix factorization algorithms are difficult to deploy in production environments.

To address SVD' s requirements for matrix filling, dimensionality reduction, and particularly the $O(N^3)$ time complexity of matrix inversion, Funk proposed the Funk-SVD algorithm. By factorizing the matrix into low-rank user and item matrices, Funk-SVD reduces computational complexity. Drawing on linear re-

gression concepts, it minimizes the squared errors of observed data to obtain optimal latent vector representations for users and items, as shown in Equation (5). Despite its simplicity, Funk-SVD performs well in practice, offering advantages such as ease of implementation, low complexity, good prediction accuracy, and scalability. Many subsequent notable models were developed through improvements to the Funk-SVD framework [1].

2 Modified Binomial Matrix Factorization Algorithm

Traditional matrix factorization algorithms leverage machine learning principles to overcome rating data sparsity and introduce L2 regularization to reduce overfitting. However, these algorithms assume that rating random variables follow a normal distribution given user and item latent factors. In most practical collaborative filtering scenarios, rating data takes discrete integer values [4]. For instance, in the MovieLens 100k and Netflix Prize datasets, rating values $S = \{1, 2, 3, 4, 5\}$, making the normal distribution assumption clearly inappropriate.

Therefore, we employ a binomial distribution assumption to replace the normal distribution assumption in matrix factorization algorithms, as shown in Equation (8). Here, $B(k|n, p)$ is the binomial distribution function with parameters n and p; S is a constant defining the allowed rating range (for MovieLens 100k and Netflix Prize, $S = 5$). $\_ui$ represents some function value of the dot product of latent factor random vectors for user u and item i. Consequently, the probability P that user u rates item i with rating r ($r = 1, 2, \cdots, S$) is given by the binomial probability mass function. This assumes that users rate all items, with each rating being at least 1, and that each point a user assigns to an item can be viewed as "like" or "dislike" based on their preferences, making movie ratings satisfy a binomial distribution. The log-posterior distribution for **P** and **Q** in BMF is therefore formulated, and L2 regularization is introduced for matrix factorization to reduce overfitting through structural risk minimization. Selecting appropriate values for the variances, maximizing Equation (11) is equivalent to minimizing the objective function shown in Equation (12). The optimal solution is sought simultaneously through stochastic gradient descent. After obtaining user and item factor matrices **p** and **q**, the predicted rating for user u on item i is calculated using Equation (7).

Following the acquisition of **P** and **Q**, this paper employs a novel rating prediction formula to obtain the predicted rating for user u on item i.

Algorithm 1: Funk-SVD Algorithm
Initialize latent factor number K, regularization parameter , learning rate , and model parameters **p** and **q** (randomly drawn from a normal distribution).
a) For each user-item rating pair (u, i):
(a) Compute rating residual
(b) Update factor vectors **p__u** and **q__i** for user u and item i
b) Compute new RMSE. If the new RMSE is smaller than the previous RMSE, continue the update process; otherwise, terminate the algorithm.

While this formula connects users and items through latent classes, in practice, some inherent properties of a rating system are independent of users and items. Users have attributes unrelated to items, and items have attributes unrelated to users.

Analysis of movie rating data reveals that among 1,682 movies, 600 movies account for 83,715 of the 100,000 user ratings—meaning most user interest concentrates on these 600 popular movies, while the remaining 1,082 movies receive little to no attention. In other words, most users are only interested in a small number of movies, and the vast majority of movies have been rated by very few users. This indicates that item popularity follows a long-tail distribution [9] and exhibits the "Harry Potter problem" [1], where different items have vastly different popularity levels. To address these issues, this paper implements a modified binomial matrix factorization algorithm (Biased-BMF) that introduces user and item rating biases into the original binomial matrix factorization model, assuming these bias terms follow a uniform or normal distribution. The rating prediction formula for user u on item i incorporating rating bias is given by Equation (14). The corresponding minimization objective function for this biased binomial matrix factorization algorithm is shown in Equation (15).

Here, and are regularization parameters; $\bar{r}$ represents the global average rating across all records in the training set. In different applications, the overall rating distribution varies due to differing application positioning and items; the global average captures the application' s inherent influence on user ratings. b_u denotes the user bias term, representing rating habits independent of item factors—specifically, patterns unique to the user' s rating behavior. b_i denotes the item bias term, representing factors in an item' s received ratings unrelated to users—specifically, characteristics intrinsic to the item. Similarly, we employ stochastic gradient descent to minimize the objective function. Algorithm 2 details the iterative process for solving the optimal solution using stochastic gradient descent.

Algorithm 2: Biased-BMF
Initialize: latent factor number D, regularization parameters and , learning rate , iteration steps, training-test split ratio, and global rating mean.
Initialize model parameters $\mathbf{P}$, $\mathbf{Q}$, and bias values b_u and b_i (obtained by randomly sampling $\mathbf{P}$, $\mathbf{Q}$, b_u, b_i from a uniform distribution).
a) For each user-item rating pair (u, i):
(a) Compute rating residual
(b) Update factor vectors **p__u** and **q__i** for user u and item i (computed using Equation (14))
(c) Update and adjust biases for user u and item i
b) Compute new RMSE. If the new RMSE is smaller than the previous RMSE or iteration count is below the default parameter value, continue updating; otherwise, terminate the algorithm.

3 Experimental Results and Evaluation

The three common methods for evaluating recommender systems are offline experiments, user studies, and online experiments. Since this paper cannot provide a real system environment for testing, we adopt offline experiments. The offline experimental procedure is as follows:
a) Data preprocessing: process the dataset into a standard format.
b) Split the dataset into training and test sets at a 9:1 ratio.
c) Train the user-item rating model on the training set and make predictions on the test set.
d) Evaluate algorithm prediction accuracy using offline metrics.

This paper employs offline experiments to evaluate the algorithm on the MovieLens 100k movie rating dataset provided by GroupLens. The MovieLens 100k dataset contains 100,000 ratings (1-5) from 943 users on 1,682 movies, with each user rating at least 20 movies. The sparsity rate is 94.11%, and we randomly sample 90% of the data as the training set and 10% as the test set for each experiment.

We evaluate recommendation accuracy using RMSE, the most common metric in offline testing—lower values indicate higher algorithmic precision and better recommendation quality. Let K be the number of rating records, r_u,i the actual rating, and *î_u,i* the predicted rating; RMSE is calculated as shown in Equation (16). We also assess computational efficiency by monitoring runtime.

We compare the performance of Biased-SVD, SVD++, BMF, and our proposed Biased-BMF algorithm. Key parameters include latent factor number D, regularization parameters and , learning rate , iteration steps, and training-test split ratio. Based on theoretical analysis and extensive parameter tuning, we fix the learning rate at = 0.02, regularization parameters at = = 0.1, and split ratio at 0.9 to investigate how iteration steps and latent factor number D affect algorithm accuracy and runtime on the test set.

[Figure 1: see original paper] and [Figure 2: see original paper] illustrate the impact of different step values on RMSE and runtime, respectively. In terms of robustness, Biased-BMF' s RMSE fluctuates within a narrow range of 0.02, whereas BMF exhibits significant volatility. Regarding recommendation accuracy, Biased-BMF clearly outperforms BMF and Biased-SVD, approaching the performance of SVD++. Notably, at step = 300, Biased-BMF achieves RMSE = 0.89, substantially better than the other three algorithms. Considering time complexity, SVD++ incorporates implicit feedback such as user browsing history and movie viewing history, making it more complex. Consequently, BMF and Biased-BMF require less time than SVD++ and Biased-SVD. Therefore, Biased-BMF can achieve higher recommendation accuracy in shorter time while maintaining stable performance across iteration counts.

[Figure 2: see original paper] Algorithm time curves under different step values

[Figure 3: see original paper] demonstrates the effect of latent factor number

D on prediction accuracy. Experiments show that BMF and Biased-BMF significantly outperform BMF and Biased-SVD in recommendation accuracy, with Biased-BMF generally surpassing SVD++. At $D = 300$, RMSE(Biased-BMF) = 0.89 while RMSE(BMF) = 0.91.

Since SVD++ considers both neighborhood models and baseline rating predictions for unknown data, it achieves higher accuracy. Biased-BMF, however, delves deeper into user and item rating biases to mine latent semantic information, yielding prediction accuracy close to SVD++ while offering superior time complexity. Thus, when the latent factor number D is determined, Biased-BMF can more accurately represent the latent feature matrices of items or users, thereby improving recommendation precision.

4 Conclusion

After reviewing and summarizing the fundamental concepts and related work on matrix factorization-based collaborative filtering recommendation algorithms, this paper implements a Biased-BMF algorithm that incorporates bias information. Building upon the BMF algorithm and considering the discrete nature of rating data, we investigate how user and item rating biases affect recommendation accuracy. Experimental results on the MovieLens 100K dataset demonstrate that Biased-BMF outperforms the original BMF algorithm and several other classical matrix factorization algorithms in both recommendation accuracy and robustness. Although this paper provides an in-depth study of rating bias issues in matrix factorization recommendation algorithms, much work remains, such as developing a dynamic Biased-BMF model that incorporates evolving user interests and integrating deep learning concepts to further mine hidden features of users and items.

References

[1] Xiang L. *Recommender Systems in Action* [M]. Beijing: Posts and Telecom Press, 2012: 24-26.

[2] Tuzhilin A, Adomavicius G. Towards the next generation of recommender systems: a survey of the state-of-the-art and possible extensions[J]. *IEEE Trans on Knowledge and Data Engineering*, 2005, 17(6): 734-749.

[3] Salakhutdinov R, Mnih A. Probabilistic matrix factorization [C]//Proc of International Conference on Neural Information Processing Systems. [S.l.]: Curran Associates Inc, 2007: 1257-1264.

[4] Liu F, Hu X. Improvement of binomial matrix decomposition in discrete scoring recommender algorithm [J]. *Computer Applications and Software*, 2016, 33(1): 81-84.

[5] Smith B, Linden G. Two decades of recommender systems at Amazon.com [M]. [S.l.]: IEEE Educational Activities Department, 2017, 21(3).

[6] He X, Liao L, Zhang H, et al. Neural collaborative filtering [C]//Proc of International World Wide Web Conference Committee. 2017: 173-182.

[7] Mooney R J, Roy L. Content-based book recommending using learning for text categorization [C]//Proc of the 5th ACM Conference on Digital Libraries. New York: ACM Press, 2000: 195-204.

[8] Balabanović M, Shoham Y. Fab: content-based, collaborative recommendation [J]. *Communication of the ACM*, 1997, 40(3): 66-72.

[9] Zhang L, Yu L. Who can cooperate with new users in the recommender system? [J]. *Computer Science*, 2015, 42(b11): 80-82.

[10] Koren Y. Factor in the neighbors: Scalable and accurate collaborative filtering [J]. *ACM Trans on Knowledge Discovery from Data*, 2010, 4(1): 1-24.

[11] Wu J. Collaborative filtering algorithm in the Netflix Prize [D]. Beijing: Peking University, 2010.

[12] Wu C Y, Ahmed A, Beutel A, et al. Recurrent recommender networks [C]//Proc of the 10th ACM International Conference on Web Search and Data Mining. New York: ACM Press, 2017: 495-503.

[13] Li Y, Li P, et al. *Machine Learning in Action* [M]. Beijing: Posts and Telecom Press, 2013: 253-257.

[14] Huang L, Jiang B, Lyu S, et al. A review of research on recommender systems based on deep learning [M]. *Chinese Journal of Computers*, 2018, 41(7): 1619-1647.

[15] Zhao H, Zhang J, Cao J. App recommendation algorithm based on topic grouping and random walk [J]. *Application Research of Computers*, 2018, 35(8): 2277-2280.

[16] Funk S. http://sifter.org/~simon/journal/20061211.html[EB/OL].

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

Source: ChinaXiv —Machine translation. Verify with original.