IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SPECIAL SECTION ON ADVANCED DATA MINING METHODS FOR SOCIAL COMPUTING

Received August 25, 2019, accepted September 19, 2019, date of publication September 26, 2019, date of current version October 9, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2943959

Matrix Factorization for Personalized Recommendation With Implicit Feedback and Temporal Information in Social Ecommerce Networks

MINGYANG LI, HONGCHEN WU, AND HUAXIANG ZHANG
School of Information Science and Engineering, Shandong Normal University, Jinan 250014, China

Corresponding author: Hongchen Wu (wuhongchen@sdnu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61702312, in part by the Natural Science Foundation of Shandong Province of China under Grant ZR2017BF019, and in part by the Project of Shandong Province Higher Educational Science and Technology Program under Grant J17KB178. The work of H. Wu was supported by the China Scholarship Council (CSC) under Grant 201306220132.

ABSTRACT Collaborative filtering with implicit feedback is regarded as one of the most challenging issues in social ecommerce networks. However, the scarcity of negative feedback and the impact of time makes collaborative filtering difficult to use. Most models assign a uniform weight to missing data, but this method is invalid in the real world and leads to a biased representation of user profiles. In social ecommerce networks, the popularity of an item is implicit social information that is dynamic and can affect the preferences of a user. In this paper, we propose a smart model named TimeMF to address the above issues by incorporating implicit feedback and temporal information into social ecommerce recommendation. The weighting scheme is based on the dynamic popularity of an item. Then, we present an objective function and adopt an optimization strategy to enhance the efficiency. The experimental results for a real-world dataset reveal that our model outperforms the baselines on several metrics.

INDEX TERMS Personalized recommendation in social computing, implicit feedback, collaborative filtering, temporal information, dynamic weight, matrix factorization, ecommerce.

I. INTRODUCTION

Although social computing has achieved considerable success in user identification [1] and human behavior [2], [3], information overload is a problem that cannot be neglected. Recommender systems based on collaborative filtering are an efficient method to address this problem in social ecommerce networks. Matrix factorization (MF) is one of the most popular technologies applied for this purpose. However, early recommender systems based on MF focused on explicit feedback [4]–[7], where the ratings of items reflect the preference of a user. Therefore, due to the unicity of data (i.e., item ratings), the recommendation is formulated as a rating prediction problem for abundant unobserved ratings (i.e., missing entries), and in order to improve the recommendation efficiency, missing entries are ignored because they are supposed to be unrelated to the preferences of the user. The modeling workload is reduced in this way, and some models based on explicit feedback, such as SVD++ [8] and TimeSVD [9], have been devised.

Matrix factorization, which is based on implicit feedback, is topical [10]. Explicit feedback is not available in many applications, such as trust information, but implicit feedback is the main method of interaction between users and items, such as a user's video viewing, purchase history, and log of sharing. The sparsity of explicit feedback results in a biased representation of user profiles [9], [11]. Compared with explicit feedback, implicit feedback is easily captured by the service provider, and rich information that reflects the user profile be given serious attention. But there is a problem

The associate editor coordinating the review of this manuscript and approving it for publication was Huizhi Liang.

related to a lack of negative feedback [12]. In order to solve this problem, a traditional solution assigns uniform weights to missing data [10], [11], [13]–[15], and overall missing data are considered to be negative feedback [10]. However, this approach is not realistic in the real world. Another strategy based on the law of implicit feedback assigns nonuniform weights to missing data, but the dynamicity of implicit feedback is not considered.

Existing models that incorporate temporal information [9], [16]–[20] focus on explicit feedback and find the regulation between rating and time to improve user bias or item bias. However, temporal implicit feedback information is usually ignored because it is not obvious in the dataset and is difficult to modeled. However, time is a significant factor in data mining, which has numerical and consistent nature, and is usually viewed as an entirety instead of a single digital field [18]. Temporal information can accurately represent the transformation of a user's activity over time, and such changes can indirectly reflect the variety of user preferences. Many factors can lead to these changes, such as artificial effects and changes in environment. For example, in a music recommender system, a user can change their preferred style of music with the release of a movie or a friend's recommendation, or a user may like a melody that they have never listened to [21]. Thus, incorporating time information into implicit feedback is important when building a recommender system.

Implicit feedback and temporal information can promote the recommendation accuracy, and it is important to use such information. The problem is to discover and model the relationship between implicit feedback and time. Unlike the relationship between explicit feedback and time (i.e., time and ratings), time-based bins are not suitable for implicit feedback because the interactive time for missing entries is non-unique. The selection of the interactive time for each item would impact the algorithm's efficiency.

In this paper, a novel recommendation model named TimeMF is proposed, which makes use of both implicit feedback and temporal information. The temporal information of a real-world dataset was analyzed in this paper, and some regulations were found. In particular, we assign the weight of missing data based on the dynamic popularity of items, which makes use of the regulations from temporal information. This method improves the results. The experimental results on a real-world dataset suggest that the proposed model is state-of-the-art.

The rest of this paper is organized as follows. Section II introduces related work on collaborative filtering of implicit feedback and time information. Section III presents an analysis of time information and proposes the TimeMF model. Section IV provides the results of an experiment and comparison. Finally, the paper is concluded.

II. RELATED WORK

Matrix factorization (MF) used in recommender systems is different from traditional singular value decomposition. Matrix factorization is inspired by machine learning and aims to obtain a feature matrix by training a model. To promote the predictive accuracy of recommendation, Koren et al. propose SVD++ [8], which incorporates implicit feedback. Henceforth, some algorithms have been developed based on SVD++, such as TrustSVD [9], TimeTrustSVD [18], and EnhancedSVD [22]. Despite incorporating social information in addition to rating scores, such as trust information, reviews of items, and the relationship of the user, these models largely focus on explicit feedback and result in biased representations of user profiles.

Compared with explicit feedback, implicit feedback can accurately reflect a user's preferences because of the large number of data that record the history of a user's activity. Thus, some matrix factorization methods have been developed based on implicit feedback [10], [23]. Recommendation is formulated by these works as a rating prediction problem for a large volume of unobserved ratings. Moreover, recommender systems are widely used to correctly rank unknown ratings for each user [11], [24]–[30], with little attention given to the actual ratings. However, the lack of negative feedback cannot be avoided. It is difficult to distinguish unknown values and negative feedback in missing data. A popular solution is to uniformly weight all missing data [10], [11], [13]–[15] and treat all unobserved ratings as negative feedback. But the weight strategy is not suitable for real applications because inefficiency is an issue. Another strategy is non-uniform weighting [12], [27]–[29] based on rules of implicit feedback. However, non-uniform weighting is difficult to utilize in real scenarios because the strategy considers only one aspect and simplifies the problem of the lack of missing data. Meanwhile, the dynamicity of implicit feedback is ignored.

Temporal information should not be neglected and exists in a wide variety of real-world datasets. Although temporal information can intuitively represent the change in a user's preference, it is not easy to utilize because temporal information is ambiguous. Recommender systems attach importance to temporal information. Some models that incorporates temporal dynamic, such as TimeSVD [9], TimeTrustSVD [18], TimeAware [16], TimeSQRC [31], and TimeAwareQOS [32], have been developed, but these methods still focus on the relationship between ratings and temporal information. Furthermore, the recommendation performance is influenced by the size of the time bins, which represent a fragment of the time span. In recent years, some models that integrate implicit feedback and temporal information have been designed [33]. But these models are developed for specific domains and need a large space to store the data that reflect the associating between implicit feedback and time.

III. MODEL

As shown in Fig.1, the proposed TimeMF model first cleans the data which is past behaviors to obtain the implicit feedback and temporal information. Afterwards, the implicit feedback and temporal information will be analyzed to find

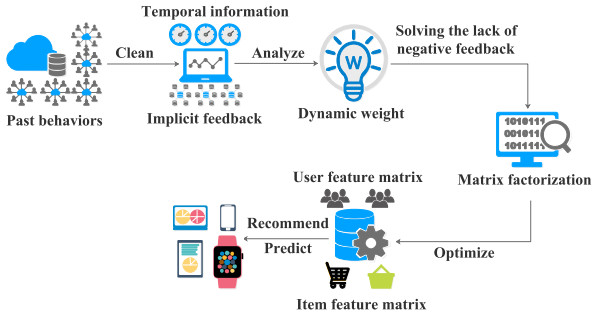

FIGURE 1. Workflow of our proposed TimeMF model.

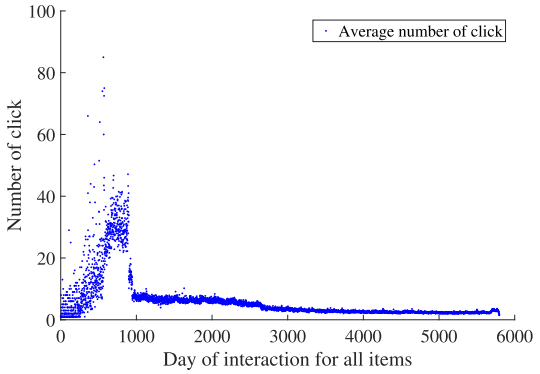

FIGURE 2. The distribution of the average number of clicks for all items.

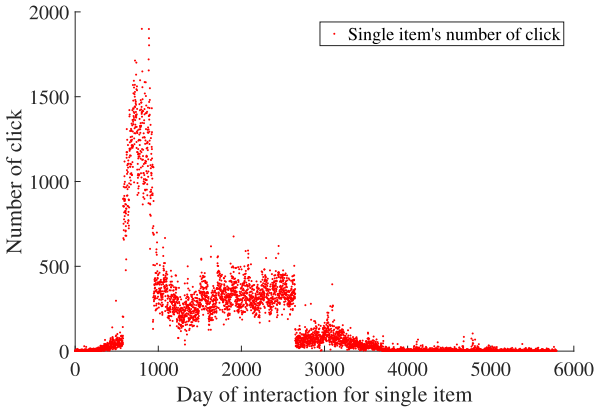

FIGURE 3. The distribution of the number of clicks for a random item.

their correspondences. Accordingly, the dynamic weight strategy is proposed to solve the lack of negative feedback. Finally, the user feature matrix and item feature matrix are obtained by training the model, and using them to recommend top-k items for each user. In what follows, we describe the analysis of temporal information.

A. THE ANALYSIS OF TEMPORAL INFORMATION

Temporal information is an important factor in recommender systems. A service provider can collect a user's information in real time, such as explicit feedback and implicit feedback, and use the information to improve the quality of service. However, the randomness and opacity of temporal information makes it difficult to use in recommender systems. Next, we will analyze the relationship between time and the popularity of an item in a real-world dataset, which will be beneficial to building the model.

The popularity of an item is an important factor in a social ecommerce network. Unlike explicit social interactions, such as trust information, the recommendation of a friend, and a user's social status, the popularity of an item is implicit. The popularity reflects the probability that an item will be exhibited on the home page; moreover, the user's activity can influence the popularity of an item. Explicit social information originates from the user's friend and affects the user's preferences; however, each user who interacts with the item can affect the popularity of the item, and thereby influence the preferences of a user who views the item exhibited on the home page. Therefore, item popularity is social information.

Our analysis is based on amazon-movie, which is a real-world dataset spanning 18 years from 1995 to 2013. Compared with the sparsity of ratings, the time density of popularity is large. Clearly, the temporal information is very dense, which creates a solid foundation to utilize these data.

First, we analyze the variation in the average popularity of all items over time to observe the influence of time on popularity. In Fig.2, we observe a strong correlation between average popularity and time. The X-axis represents the time span, and the Y-axis represents the average popularity of all items. Clearly, the average popularity of all items increases for 900 days and reaches a peak on the 1000th day. This phenomenon suggests that items experience a hot stage before the 1000th day. But after 1000 days, at which point the average popularity drops significantly and begins to decrease slowly. This tendency indicates that items gradually become obscure.

Next, consider the variation in the popularity of a single item over time. We randomly select an item. In Fig.3, the X-axis represents the time span, and the Y-axis represents the popularity of a single item. The popularity increases slowly for approximately 500 days, but after the 500th day, it increases significantly and reaches a peak on the 1000th day. However, the popularity the decreases in a short time. After 1000 days, the remaining time span can be split into two periods. The first period is approximately 1700 days, from the 1000th day to 2700th day, and the second period is approximately 3300 days, from the 2700th day to the 6000th day. Although fluctuating in the two periods, the popularity of a single item still shows an overall decreasing trend.

Based on the above analysis, we can draw the following conclusions:

First, there is a strong correlation between time and popularity. Popularity is dynamic and changes over time. In general, the stage of popularity can be split into two periods: the hot period in which the popularity is increasing and the cold period when the popularity is declining.

Second, the discrimination of negative feedback is transformed with time. If an item is popular, the item should be

recommended and shown on the home page [34]. Therefore, if a popular item is not interacted with by users, we should treat the lack of interaction as negative feedback. However, the discrimination of negative feedback is influenced by temporal information; therefore, we should consider time information to discriminate negative feedback from missing data.

B. PRELIMINARIES

We being by introducing some basic notation. User-item interaction matrix $\mathcal{R}$ has a volume of $M \times N$, where M and N denote the numbers of users and items, respectively. R denotes the set of user-item pairs that are not missing values. We use index u to denote a user, and an item is denoted by index i. Vector x_u denotes the latent feature vector of u, and vector y_i denotes the latent feature vector of i. $R(u)$ denotes the set of items that u interacts with. Similar notation is used for y_i and $R(i)$. Matrices $X \in \mathbb{R}^{(K\times M)}$ and $Y \in \mathbb{R}^{(K\times N)}$ denote the latent feature matrices for users and items, respectively. x_{uk} is the kth feature of x_u. Similar notation is used for y_{ik}.

Formally, the squared loss is used to formulate the objective function of weighted matrix factorization.

$$J = \sum_{(u,i)\in\mathcal{R}} d_{ui}\left(r_{ui} - \hat{r_{ui}}\right)^2 + \lambda\left(\sum_{u}^{M}\|x_u\|^2 + \sum_{i}^{N}\|y_i\|^2\right) \tag{1}$$

where d_{ui} denotes the weight of each user-item pair. r_{ui} denotes each entry of $\mathcal{R}$, $\hat{r_{ui}}$ is the prediction score following matrix factorization, $\hat{r_{ui}} = {x_u}^T y_i$. λ is a regularization parameter, which is usually the L_2 norm, to prevent overfitting. Note that in the implicit feedback, missing data are usually assigned a zero value, but the value of the weight is nonzero, which is important for the performance.

C. WEIGHT STRATEGY

According to a previous analysis, we know how to find negative feedback in unobserved values. Therefore, we propose a new weight strategy based on the dynamic popularity of items. Thus, we parametrize $d\left(i, stage_i\right)$ based on the following strategy:

$$d(i, stage_i) = d_0 \frac{{c_i}^{sig(stage_i)\times\alpha}}{\sum_{j}^{N} {c_j}^{sig(stage_i)\times\alpha}} \tag{2}$$

where c_i denotes the popularity of $item_i$ based on the number of clicks; $stage_i$ denotes the period of $item_i$; d_0 denotes the weight of overall missing values; and α is an exponent ranging from 0 to 1 that controls the level of popularity of an item and helps in discriminating negative feedback.

On the basis of the above analysis of time information, item popularity is influenced by temporal dynamics, e.g., the period of time determines whether the item is popular. Therefore, how to split the granularity of time to determine popularity is important. Existing models, such as TimeSVD [9] and TimeTrustSVD [18], focus on the relationship between ratings and time. They usually adopt the final time as the rating's time; thus, there is a one-to-one correspondence between ratings and time. Compared with explicit feedback, implicit feedback is composed of a user's activity history. The relationship between the activity of a user and time is many-to-many, and activity history does not have any tag to indicate who executes the activity. In implicit feedback, whether a user performs an interactive action on an item is a criterion used to distinguish the item type as interactive or non-interactive. If we adopted the time strategy of explicit feedback, we would be confronted with several problems. First, the granularity of time is fine, so we need a large volume for storage. Second, according to the previously defined objective function, if an item is not interacted with by any user, the interactive time of the item is difficult to determine.

To address these problems, we include a parameter t_{stage}, which is a time node that places a boundary on popularity. An item is popular when its time of issuance is less than t_{stage}; otherwise, the item is out of fashion. Based on a previous analysis, a popular item should be well known in general: a popular item that is ignored is probably irrelevant to the user. Thus, a miss on a popular item should be given a high weight to be treated as negative feedback. $stage_i$ is a parameter used to define the stage of an item. It has two values, -1 or 1. $item_i$ is popular when the value of $stage_i$ is 1, and its weight is high; otherwise, the item is not popular, and its weight is lower than that of a popular item.

In addition to t_{stage}, the length of the release time of an item is an important factor in defining the value of $stage_i$. The time of the first interaction with the item is the start point, and the end point is the time of the last interaction with the item. Based on the two time points, we can calculate the time span of the item. If the time span is less than t_{stage}, the value of $stage_i$ is 1; otherwise, the value is -1.

D. TRAINING THE MODEL

Due to the large volume of items, missing values for a user are a mixture of unknown values and negative feedback. However, negative feedback and unknown values are difficult to discriminate in missing data because of the nature of implicit feedback. Therefore, we can set a matrix to weight missing values to identify negative feedback. Considering the influence of temporal information, we devise a fine-grained objective function:

$$\begin{aligned} J = &\sum_{u}\sum_{i\in R(u)} d_e\left(r_{ui} - \hat{r_{ui}}\right)^2 \\ &+ \sum_{u}\sum_{i\notin R(u)} d(i, stage_i){\hat{r_{ui}}}^2 \\ &+ \lambda\left(\sum_{u}^{M}\|x_u\|^2 + \sum_{i}^{N}\|y_i\|^2\right) \end{aligned} \tag{3}$$

where d_e denotes the confidence of explicit feedback and $d(i, stage_i)$ is a function used to calculate the weight of

implicit feedback. The objective function can be split into two parts. The first part denotes the loss of observed data, and the second part denotes the loss of missing data.

We analyze the second term of objective function, $\sum_u \sum_{i \notin R(u)} d(i, stage_i) \hat{r_{ui}}^2$, which denotes the square loss of missing entries. The term is inefficient because most items are not rated by the user. However, the term can be reformulated between the sum on all entries and the sum on the items rated by the user. Thus, the sum of missing entries can be removed from the computations:

$$\sum_u \sum_{i \notin R(u)} d\,(i, stage_i)\, \hat{r_{ui}}^2 = \sum_u^M \sum_i^N d\,(i, stage_i)\, \hat{r_{ui}}^2 - \sum_u \sum_{i \in R(u)} d\,(i, stage_i)\, \hat{r_{ui}}^2 \tag{4}$$

There are two terms in the formulation, and the first term, $\sum_u^M \sum_i^N d(i, stage_i) \hat{r_{ui}}^2$, is the core on which the subsequent optimization is based. The first term can be transformed as:

$$\sum_u^M \sum_i^N d\,(i, stage_i)\, \hat{r_{ui}}^2 = \sum_u^M \sum_i^N [d(i, stage_i) \times (x_u{}^T y_i)(x_u{}^T y_i)] \tag{5}$$

$\sum_u^M \sum_i^N d\,(i, stage_i)\, \hat{r_{ui}}^2$ can be reformulated:

$$\sum_u^M \sum_i^N d\,(i, stage_i)\, \hat{r_{ui}}^2 = \sum_u^M \sum_i^N [d(i, stage_i) \times x_u{}^T y_i y_i{}^T x_u] \tag{6}$$

We have posed $S^I = \sum_i^N d(i, stage_i) y_i y_i{}^T$, a $K \times K$ matrix that is independent of i. Assuming S^I is known, we can compute the derivative of x_{uk}:

$$\frac{\partial J}{\partial x_{uk}} = \sum_{i \in R(u)} \left[d_e\left(\hat{r_{ui}} - r_{ui}\right) - d(i, stage_i) \cdot \hat{r_{ui}}\right] \cdot y_{ik} + \sum_f^K x_{uf} \cdot s_{kf}^I + \lambda x_{uk} \tag{7}$$

where s_{kf}^I denotes the factor of S^I, and K is the number of features of the latent feature matrix of X.

Symmetrically,

$$S^U = \sum_u^M x_u x_u{}^T \tag{8}$$

$$\sum_u^M \sum_i^N d\,(i, stage_i)\, \hat{r_{ui}}^2 = \sum_u^M \sum_i^N [d(i, stage_i) \times y_i{}^T x_u x_u{}^T y_i] \tag{9}$$

We assume that the feature of an item is known, and we can compute the derivative of y_{ik}:

$$\frac{\partial J}{\partial y_{ik}} = \sum_{u \in R(i)} \left[d_e\left(\hat{r_{ui}} - r_{ui}\right) - d(i, stage_i) \cdot \hat{r_{ui}}\right] \cdot x_{uk} + \sum_f^K y_{if} \cdot s_{kf}^U + \lambda y_{ik} \tag{10}$$

where s_{kf}^U denotes the factor of S^U and K is the number of features of the latent feature matrix of Y.

According to the derivation, our optimization is based on the features of the latent feature matrix. Therefore, we use adaptive gradient descent to determine the size of the gradient step for each feature of the latent feature matrix. Meanwhile, the gradient steps of features are dynamic; in other words, they decrease with increasing feature gradient. Thus, we can obtain the optimal solution with respect to the objective function, and *lrate* denotes learn rate.

Algorithm 1 TimeMF

Input: $\mathcal{R}$, K, λ, t_{stage}, α, d_0, *lrate*
Output: the latent feature Matrices X and Y

```
Initialize X and Y
Calculate the vector of weights with the popularity of item and stage_i
for the epoch is less than the number of iterations do
    for u is less than the number of users do
        for f is less than K do
            Adopt AadGRAD to update the lrate of x_uf
            Update the x_uf in lrate
            Update the gradient of x_uf
        end for
    end for
    for i is less than the number of items do
        for f is less than K do
            Adopt AdaGRAD to update the lrate of y_if
            Update the y_if in lrate
            Update the gradient of y_if
        end for
    end for
end for
return X and Y
```

Algorithm1 summarizes TimeMF. For convergence, one can either monitor the value of the objective function on the training set or check the prediction performance on hold-out validation data.

IV. EXPERIMENT

In this section, we begin by introducing the experimental setting, followed by the traditional offline protocol.

A. EXPERIMENTAL SETTING

In this part, we introduce the experimental setting.

1) DATASET

We evaluate a real-world dataset: Amazon Movie. In order to transform the review data into implicit feedback, we use a strategy in which each entry is marked as 0/1 to indicate whether the item was viewed by the user. Note that if the value of the entry is 1, the user interacted with the item; otherwise, the user did not view the item. For the purpose of availability of temporal information, we filter out the information with invalid time stamps.

2) METHODOLOGY

In order to build the training matrix and testing list, we use leave-one-out evaluation in which the latest interaction of the user is reserved for prediction and the remaining data are used to train the model. We adopt the traditional offline protocol to evaluate the performance of our algorithm; therefore, the strategy is based on the historical data of recommended items for the user.

Compared with traditional matrix factorization, matrix factorization based on implicit feedback focuses on correctly ranking the unobserved items for each user, which is known as the top-k problem. The evaluation of the performance is different from that of traditional matrix factorization. To assess the ranked list based on test items that the user interacted with, we use the hit ratio (HR) and normalized discounted cumulative gain (NGCD). The HR indicates whether the ground truth is in the ranked list, and the NDCG denotes the position of the ground truth in the ranked list. In fact, relevant items ranked at high positions contribute to the final score.

3) BASELINES

We compare our method with the following models:

ALS [10]: A conventional matrix factorization model based on complete data.

RCD [11]: A state-of-art matrix factorization model based on implicit feedback. RCD determines the step size of the vector of features by line search, and the value we use comes from the author's implementation.

4) PARAMETER SETTING

With regard to the weight of observed entries, we uniformly set the value to 1. We set the regularization parameter to 0.1 for all models, which is convenient for paired comparisons. For the initialization of the latent feature matrix, we use a Gaussian distribution. Note that we set the mean to 0 and the standard deviation to 0.01. All models are implemented in java8 and run on the same machine (Intel core i7 CPU and 16 GB RAM). The performance increases with increasing number of features, and we use K to denote the number of features. Thus, we discuss the result only for $K = 128$.

B. WEIGHT OF MISSING VALUES

In section III, we propose the strategy of assigning weights based on the dynamic popularity of an item based on time information. There are three parameters: d_0, α, and t_{stage}.

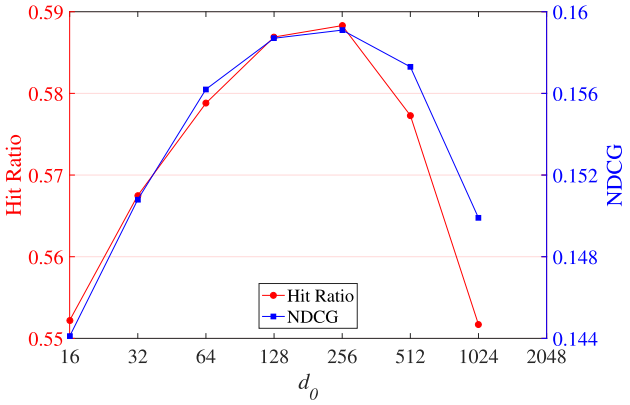

FIGURE 4. The impact of weighting parameter d_0.

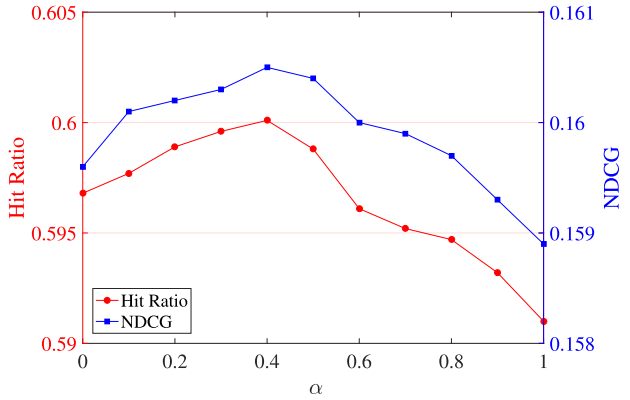

FIGURE 5. The impact of controlling parameter α.

d_0 denotes the confidence of overall missing entries, α controls the level of popularity for each item, and t_{stage} denotes the period in which the item is popular, to determine the value of $stage_i$.

First, we set $\alpha = 0$ and $t_{stage} = 1000$ to study how d_0 impacts the performance. In Fig.4, it is clear that the performance degrades significantly when d_0 is very small, such as $d_0 = 0$. As d_0 increases, the performance improves. This result demonstrates the necessity of considering missing entries when building implicit feedback about item recommendations. However, the performance suffers significantly when the value of d_0 is very large.

Then, we keep the value of t_{stage} constant, set d_0 to the best value, and vary α to study the impact on the performance. In Fig.5, when $\alpha = 0$, we assign uniform weights to missing values, missing entries are treated as negative feedback, and the performance degrades seriously. The performance improves as α increases. However, if α is too large, such as $\alpha > 1$, the performance suffers because a large α has an adverse effect on the ability to discriminate negative feedback. As a result, the item weights are imbalanced between popular items and less popular items. Thus, it is important to assign a proper weight for less popular items.

Finally, we set d_0 and α to the best values and vary t_{stage} to assess the change in performance. In Fig.6, when t_{stage}

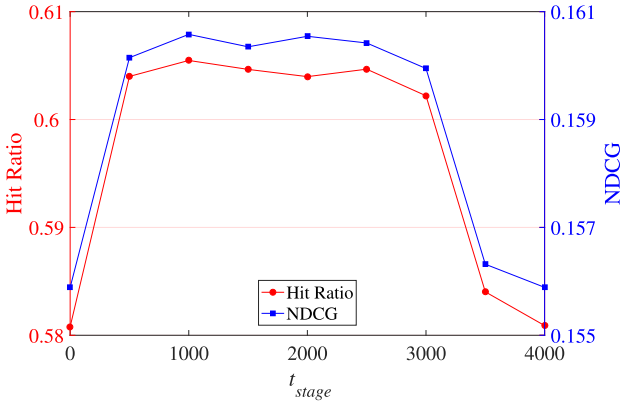

FIGURE 6. The impact of time parameter t_{stage}.

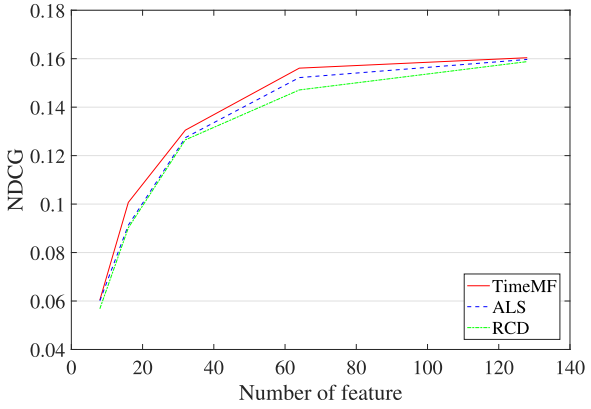

FIGURE 7. The impact of the number of features.

is large or small, such as $t_{stage} = 0$ or $t_{stage} > 4000$, the performance drops substantially because t_{stage} is an essential component for controlling the level of item popularity. Extreme values degrade the ability to discriminate negative feedback. Therefore, finding a correct time point is crucial for the performance.

In the following experiments, we fix d_0, α, and t_{stage} to their best values, i.e., $d_0 = 256$, $\alpha = 0.4$, $t_{stage} = 1000$, and we set the learn rate according to grid search, i.e., $lrate = 0.012$.

C. THE IMPACT OF THE NUMBER OF FEATURES

Fig.7 shows the relationship between the number of features and the prediction accuracy in terms of NDCG. TimeMF consistently outperforms ALS and RCD, which demonstrates that the proposed model is efficient. The prediction accuracy improves significantly with increasing K. In real applications, there is a large volume of users and items, and a large number of features is important to improve the prediction accuracy.

D. TRAINING TIME

The training time per iteration is shown in Table 1. With the increase of the number of features, ALS spends longer time than TimeMF and RCD. Specially, when the number of features is 128, ALS spends 29 minutes for one iteration on amazon-movie, while TimeMF takes 4.7 minutes. The speed-up is significant. TimeMF and RCD have the same running time, the minor difference can be caused by the data structures.

TABLE 1. Training time per iteration of different methods with varying the number of features.

number of features	TimeMF	ALS	RCD
8	7s	9s	15s
16	15s	25s	17s
32	34s	1.4m	25s
64	80s	6m	47s
128	4.7m	29m	2.3m

s and m denote seconds and minutes.

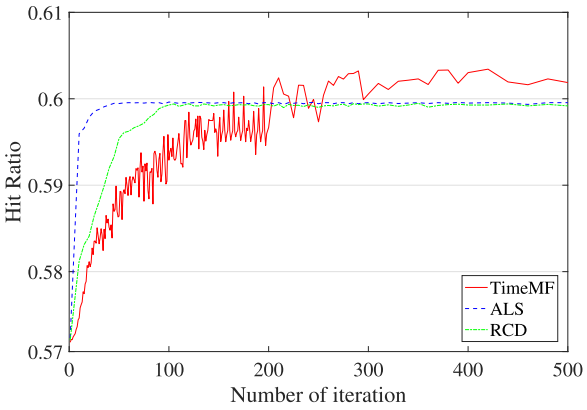

FIGURE 8. The prediction accuracy of three MF models in each iteration (Hit Ratio).

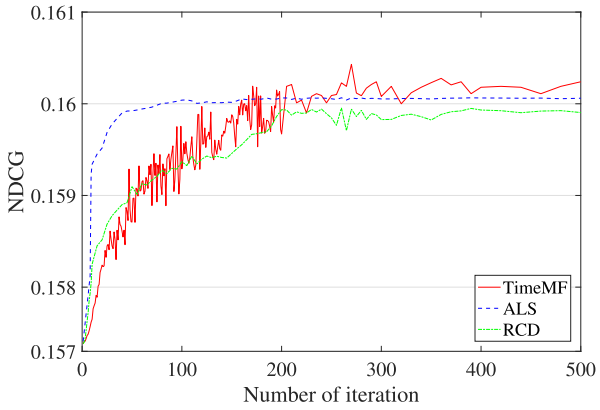

FIGURE 9. The prediction accuracy of three MF models in each iteration (NDCG).

E. COMPARISON

Fig.8 and Fig.9 show the prediction accuracy with regard to the iteration number. Note that to carefully show the impact of iteration for TimeMF, the iteration is split into four stages: in the first stage (iterations 1 to 199) the span of iteration is 1; in the second stage (iterations 200 to 300), the span is 5; in the third stage (iterations 300 to 400), the span is 10; and in the fourth state (all remaining iterations), the span

is 20. After the number of iterations exceeds 300, the accuracy of these models, TimeMF, ALS, and RCD become stable, and TimeMF achieves the best performance. Thus, TimeMF outperforms ALS and RCD. Meanwhile, the prediction accuracy of TimeMF is lower than that of the other models when the number of iterations is less 300. ALS focuses on optimizing the feature vector, so it does not need to learn the rate to train the features of the latent feature matrix via gradient descent and can therefore rapidly reach a stable state. RCD adopts a line search to determine the best step size for each feature in the feature vector, so it can find the best step size to train the features for each epoch. By contrast, TimeMF uses adaptive gradient descent to train the latent feature matrix; therefore, due to the nature of adaptive gradient descent, we cannot obtain the optimal solution when the number of iterations is small. This is the reason that the other methods converge faster than does TimeMF. However, once the number of iterations exceeds 300, the proposed model outperforms the other state-of-the-art models. There are two reasons. First, we adopt the dynamic popularity of items in which the level of popularity is controlled by the temporal information and is used to weight missing entries. The weighting strategy addresses the lack of negative feedback to facilitate the utilization of missing data. Second, each feature in the feature vector is assigned a step size for training, and the step size decreases with increasing feature gradient; thus, we can obtain the optimal solution for each feature, which contributes to the prediction accuracy.

V. CONCLUSION

In this work, we propose a smart model called TimeMF, which is based on implicit feedback and incorporates temporal information, to address the lack of negative feedback to solve the problem of information overload in social ecommerce networks. We study the relationship between item popularity and time and devise a new weighting strategy based on the dynamic item popularity to assign non-uniform weights to missing data. The optimal model assigns an exclusive learning rate to each feature of the latent feature matrix and adopts adaptive gradient descent to update the learning rate to enhance the accuracy. The experimental results reveal that our model outperforms the baselines with respect to ranking-oriented evaluation. Furthermore, temporal information can be used to improve the prediction accuracy in matrix factorization based on implicit feedback.

REFERENCES

[1] K. Deng, L. Xing, L. Zheng, H. Wu, P. Xie, and F. Gao, "A user identification algorithm based on user behavior analysis in social networks," *IEEE Access*, vol. 7, pp. 47114–47123, 2019.

[2] M. A. Al-garadi, M. R. Hussain, N. Khan, G. Murtaza, H. F. Nweke, I. Ali, G. Mujtaba, H. Chiroma, H. A. Khattak, and A. Gani, "Predicting cyberbullying on social media in the big data era using machine learning algorithms: Review of literature and open challenges," *IEEE Access*, vol. 7, pp. 70701–70718, 2019.

[3] L. Nie, X. Song, and T.-S. Chua, "Learning from multiple social networks," *Synthesis Lectures Inf. Concepts, Retr., Services*, vol. 8, no. 2, pp. 1–118, Apr. 2016.

[4] Y. Koren and R. Bell, "Advances in collaborative filtering," in *Recommender Systems Handbook*. Cham, Switzerland: Springer, 2015, pp. 77–118.

[5] H. Zhang, F. Shen, W. Liu, X. He, H. Luan, and T.-S. Chua, "Discrete collaborative filtering," in *Proc. 39th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jul. 2016, pp. 325–334.

[6] Y. Zhou, D. Wilkinson, R. Schreiber, and R. Pan, "Large-scale parallel collaborative filtering for the netflix prize," in *Proc. Int. Conf. Algorithmic Appl. Manage.* Cham, Switzerland: Springer, 2008, pp. 337–348.

[7] T. Man, H. Shen, J. Huang, and X. Cheng, "Context-adaptive matrix factorization for multi-context recommendation," in *Proc. 24th ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2015, pp. 901–910.

[8] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 1, no. 8, pp. 30–37, Aug. 2009.

[9] Y. Koren, "Collaborative filtering with temporal dynamics," in *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jun./Jul. 2009, pp. 447–456.

[10] Y. Hu, Y. Koren, and C. Volinsky, "Collaborative filtering for implicit feedback datasets," in *Proc. 8th IEEE Int. Conf. Data Mining*, Dec. 2008, pp. 263–272.

[11] R. Devooght, N. Kourtellis, and A. Mantrach, "Dynamic matrix factorization with priors on unknown values," in *Proc. 21th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2015, pp. 189–198.

[12] R. Pan, Y. Zhou, B. Cao, N. N. Liu, R. Lukose, M. Scholz, and Q. Yang, "One-class collaborative filtering," in *Proc. 8th IEEE Int. Conf. Data Mining*, Dec. 2008, pp. 502–511.

[13] I. Pilászy, D. Zibriczky, and D. Tikk, "Fast ALS-based matrix factorization for explicit and implicit feedback datasets," in *Proc. 4th ACM Conf. Recommender Syst.*, Sep. 2010, pp. 71–78.

[14] H. Steck, "Training and testing of recommender systems on data missing not at random," in *Proc. 16th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2010, pp. 713–722.

[15] M. Volkovs and G. W. Yu, "Effective latent models for binary feedback in recommender systems," in *Proc. 38th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Aug. 2015, pp. 313–322.

[16] H. Liang, Y. Xu, D. Tjondronegoro, and P. Christen, "Time-aware topic recommendation based on micro-blogs," in *Proc. 21st ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2012, pp. 1657–1661.

[17] Y. Ding and X. Li, "Time weight collaborative filtering," in *Proc. 14th ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2005, pp. 485–492.

[18] C. Tong, J. Qi, Y. Lian, J. Niu, and J. J. Rodrigues, "TimeTrustSVD: A collaborative filtering model integrating time, trust and rating information," *Future Gener. Comput. Syst.*, vol. 93, pp. 933–941, Apr. 2017.

[19] D. Sánchez-Moreno, Y. Zheng, and M. N. Moreno-García, "Incorporating time dynamics and implicit feedback into music recommender systems," in *Proc. IEEE/WIC/ACM Int. Conf. Web Intell. (WI)*, Dec. 2018, pp. 580–585.

[20] P. Zhang, Z. Zhang, T. Tian, and Y. Wang, "Collaborative filtering recommendation algorithm integrating time windows and rating predictions," *Appl. Intell.*, vol. 49, no. 8, pp. 3146–3157, Aug. 2019.

[21] B. Marlin, R. S. Zemel, S. Roweis, and M. Slaney, "Collaborative filtering and the missing at random assumption," 2012, *arXiv:1206.5267*. [Online]. Available: https://arxiv.org/abs/1206.5267

[22] X. Guan, C.-T. Li, and Y. Guan, "Matrix factorization with rating completion: An enhanced SVD model for collaborative filtering recommender systems," *IEEE Access*, vol. 5, pp. 27668–27678, 2017.

[23] B.-W. Chen, W. Ji, S. Rho, and Y. Gu, "Supervised collaborative filtering based on ridge alternating least squares and iterative projection pursuit," *IEEE Access*, vol. 5, pp. 6600–6607, 2017.

[24] S. Balakrishnan and S. Chopra, "Collaborative ranking," in *Proc. 5th ACM Int. Conf. Web Search Data Mining*, Feb. 2012, pp. 143–152.

[25] P. Cremonesi, Y. Koren, and R. Turrin, "Performance of recommender algorithms on top-n recommendation tasks," in *Proc. 4th ACM Conf. Recommender Syst.*, Sep. 2010, pp. 39–46.

[26] J. Lee, S. Bengio, S. Kim, G. Lebanon, and Y. Singer, "Local collaborative ranking," in *Proc. 23rd Int. Conf. World Wide Web*, Apr. 2014, pp. 85–96.

[27] R. Pan and M. Scholz, "Mind the gaps: Weighting the unknown in large-scale one-class collaborative filtering," in *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jun./Jul. 2009, pp. 667–676.

[28] X. He, H. Zhang, M.-Y. Kan, and T.-S. Chua, "Fast matrix factorization for online recommendation with implicit feedback," in *Proc. 39th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jul. 2016, pp. 549–558.

[29] H. Li, X. Diao, J. Cao, and Q. Zheng, "Collaborative filtering recommendation based on all-weighted matrix factorization and fast optimization," *IEEE Access*, vol. 6, pp. 25248–25260, 2018.

[30] C.-Y. Lin, L.-C. Wang, and K.-H. Tsai, "Hybrid real-time matrix factorization for implicit feedback recommendation systems," *IEEE Access*, vol. 6, pp. 21369–21380, 2018.
[31] F. Zhang, "A personalized time-sequence-based book recommendation algorithm for digital libraries," *IEEE Access*, vol. 4, pp. 2714–2720, 2016.
[32] S. Li, J. Wen, F. Luo, and G. Ranzi, "Time-aware QoS prediction for cloud service recommendation based on matrix factorization," *IEEE Access*, vol. 6, pp. 77716–77724, 2018.
[33] V. W. Anelli, T. Di Noia, E. Di Sciascio, A. Ragone, and J. Trotta, "Local popularity and time in top-n recommendation," in *Proc. Eur. Conf. Inf. Retr.* Cham, Switzerland: Springer, 2019, pp. 861–868.
[34] X. He, M. Gao, M.-Y. Kan, Y. Liu, and K. Sugiyama, "Predicting the popularity of Web 2.0 items based on user comments," in *Proc. 37th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jul. 2014, pp. 233–242.

MINGYANG LI received the B.S. degree in computer science and technology from the Harbin University of Science and Technology, in 2016. He is currently pursuing the master's degree with the School of Information Science and Engineering, Shandong Normal University, China. His research interests include recommender systems, network security, and data mining.

HONGCHEN WU received the Ph.D. degree in computer science and technology from Shandong University, in 2016. He studied at the University of California at Irvine, USA, as a joint Ph.D. student supported by the CSC for two years, since 2013. He is currently a Lecturer with the School of Information Science and Engineering, Shandong Normal University, China. His research interests include machine learning, network security, and data management.

HUAXIANG ZHANG received the Ph.D. degree from Shanghai Jiaotong University, in 2004. He is currently a Professor and the Dean of the School of Information Science and Engineering, Shandong Normal University, China. He has authored over 180 journal and conference papers. His research interests include machine learning, pattern recognition, evolutionary computation, and web information processing. He also served as a Review Expert for the National Natural Science Foundation of China, the Doctoral Found of the Ministry of Education of China, and the China Postdoctoral Science Foundation.