Knowledge-Based Systems 213 (2021) 106660

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Self-paced learning enhanced neural matrix factorization for noise-aware recommendation

Zhen Liu [a], Xiaodong Feng [b,*], Yecheng Wang [a], Wenbo Zuo [a]

[a] School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, Sichuan 611731, China
[b] Department of Information Management, School of Public Affairs and Administration, University of Electronic Science and Technology of China, Chengdu, Sichuan 611731, China

A R T I C L E I N F O

Article history:
Received 30 March 2020
Received in revised form 6 December 2020
Accepted 7 December 2020
Available online 13 December 2020

Keywords:
Recommendation
Deep learning
Noisy and outlier corruption
Instance weighting
Self-paced learning

A B S T R A C T

Matrix factorization-based collaborative filtering, learning user and item latent features, has been one of the powerful recommendation techniques. Due to its simply modeling of user–item interactions by inner product of two vectors as a linear model, its efficiency needs an improvement. Neural Network-based matrix factorization has been proposed to deal with this issues. Usually, these methods are proposed on clean data, but in real applications, there are possibly unexpected noises and outliers, due to many subjective or objective reasons. The noisy instances would disturb the learning of normal instances and thus cause adverse affect as the model would also be easily over-fitted. Thus, we propose an enhanced neural matrix factorization model by introducing a self-paced learning (SPL) schema, which can automatically distinguish noisy instances and learn the model mostly based on clean instances. The main contribution of our model is that we design a bounded SPL learning schema with a parameter to control how many instances will be finally induced in the model learning. Thus, different from traditional SPL that gradually selects instances until all are selected, the bounded SPL mechanism tries to learn the model mainly on clean data and exclude noisy instances. The effectiveness of proposed method on collaborative filtering is demonstrated by extensive experiments on three widely used datasets.

1. Introduction

The rapid development of Internet makes it possible to browse or buy a large number of products, which leads to an information overload issue for users to search for what users are interested in much harder or impossible. Also, it brings about a big competition for business runners to attract customers. In order to improve the likelihood of users clicking or buying some online products, such as books, music or video items, personalized recommendation has become ubiquitous, applied in numerous E-commerce platform to alleviate the information overload issues. Many firms, such as Amazon, Netflix, and Taobao, have adopted recommender systems (RS) that estimate users' potential preferences and recommend relevant items from the big number of candidate products, which have been demonstrated as a necessary part of their success [1]. Thus, how to generate more effective recommendation has attracted much attention in both E-commerce industry application and academic domain, and much efforts have been devoted to developing various approaches and algorithms to improve the performance of RS.

Among the various techniques for RS, content-based methods [2] generate recommended items by matching user profiles and product descriptions, and the performance is mainly limited by the quality of descriptors. Collaborative filtering (CF) techniques, however, recommend items through searching similar items or users from the user–item interaction records with the help of many models, which prone to generally outperform content-based methods [1]. Generally, most CF techniques fall into neighbor-based and model-based categories. Compared to neighbor-based methods that directly recommend items based on item or user pairwise similarity [2], model-based methods are more powerful to capture the users' preference hidden in the historical records and thus can achieve better performance especially on more complex tasks.

Matrix factorization (MF) [3], as a defacto latent factor model, has been so popular in the past decade since the successful use of Netflix Price. MF factorizes the user–item interaction matrix (explicit purchase or implicit click behavior) into two low dimensional matrices representing each user and item on shared latent space. As a result, an interaction of one user on a given item is predicted by computing the inner product of two corresponding latent vectors. However, it is found that the performance

* Corresponding author.
E-mail addresses: quake.liu0625@gmail.com (Z. Liu), fengxd1988@hotmail.com (X. Feng), 924172380@qq.com (Y. Wang), 251927330@qq.com (W. Zuo).

https://doi.org/10.1016/j.knosys.2020.106660
0950-7051/© 2020 Elsevier B.V. All rights reserved.

of MF is limited by the simply choose of inner production as the interaction function despite its robust effectiveness on CF tasks. To overcome this limitation, neural matrix factorization (NeuMF) [4] was proposed to make use of neural network layers to approximate the interaction function from latent vectors of users and items, generating better results than traditional MF.

The optimization problems of both MF and NeuMF are non-convex, and thus widely-used optimization approaches, such as alternating least squares (ALS) or Stochastic Gradient Descent (SGD) cannot avoid getting stuck into the local minima due to this non-convexity. Moreover, the model would also be easily over-fitted when the dataset of user–item interaction records contains much noisy instances and outliers. Actually, in the historical records of user–item interactions, some instances may be seen as noisy corruption, such as spam ratings [5] given by accident or given by malicious competitors. Therefore, before discussing the impact of noisy data, we formally give a concept of outlier in Definition 1.

Definition 1. Outlier (or noisy instance). An outlier is a negative instance (a never really happened user–item interaction) yet falsely labeled as the positive (a truly happened user–item interaction).

To alleviate the affect of non-convexity and noisy instances, a heuristic way based on the idea of cross-validation is to run the optimization process repeatedly for many times using different learning rates or initializations, and return the best results. However, this approach is generally inappropriate to implement for recommendation tasks that is processing a large dataset. Therefore, in this paper, we put forward with an enhanced neural matrix factorization by conducting a robust Self-Paced Learning (SPL) schema to overcome the above limitations caused by noisy corrupting in training instances. Specifically, SPL [6] can be understood as an instance selection framework that gradually involves easy ones to complex ones, as a simulation of the learning principle from humans/animals. This learning mechanism has been empirically shown to benefit various tasks [7,8].

To better recognize noises and mitigate the influence of noises, we propose a bounded SPL enhanced neural matrix factorization, denoted as 'NeuMF_SPL_B'. It firstly inherits the idea of the mechanism of SPL that can dynamically process instance selection from easy to complex. However, different from traditional SPL mechanism that gradually involving instances until all are completely included, the new learning method will not include all instances which would be recognized as noises during the whole learning process and finally these noisy instances will be not completely assigned with weights of 1. NeuMF_SPL_B is quite different from the references of other SPL-based works [7–9], as it is a bounded learning schema with a parameter to control how many instances will be finally included in the model learning. The target of this work is to provide a recommendation method that can bring high accuracy by alleviating the deficiency due to noisy corruption in the training ratings.

The main contributions of this paper can be summarized as follows:

(1) We define a unified framework for personalized recommendation with introducing the methodology of Self-Paced Learning schema, namely self-paced learning enhanced neural matrix factorization.

(2) Furthermore, we propose a novel bounded self-paced learning mechanism enhanced neural matrix factorization (**NeuMF_SPL_B** for short), which introduces a parameter to exclude a proportion of instances during the whole learning process.

(3) Extensive experiments on three real datasets show its superiority on alleviating the affect caused by noisy corruption on the learning.

The remainder of this paper is organized as follows. In Section 2, we briefly review related works about recommendation techniques and self-paced learning. We present the proposed model with formulation as well as learning algorithm in Section 3. Section 4 demonstrate the experimental results and a short analysis. The conclusion and discussion about possible improvement in future are summarized in Section 5.

2. Related work

Since this work lies in the cross road of collaborative filtering-based recommendation and self-paced learning, we will simply review the related works on these two fields.

2.1. Collaborative filtering models for recommendation

Generally, model-based collaborative filtering methods can be mainly categorized into three groups: similarity learning-based models, latent factor-based shallow models and deep learning-based models.

Similarity measurement has been the basic component of neighbor-based models [10], which directly use k-NN(k-nearest neighbor) based on pairwise similarity, such as cosine similarity, between each two users or items. However, pairwise similarity is sensitive to noisy data, and is subject to the curse of dimension where high-dimensional user–item matrix tends to dramatically distort the structure of the pairwise similarity. Thus, model-based similarity is studied to overcome this shortcoming, where Sparse LInear Method (SLIM) [11] can bee seen as firstly to learn the similarity between items based on sparsity regularized linear regression. Following this work, SocSLIM [12] was proposed to learn user-based similarity considering both rating information and social relationships. To reduce the high offline training complexity and capture transitive relations, Factored Item Similarity Model (FISM) [13] was developed, followed by extensions with attention mechanism [14,15]. Also, random walks and network propagation are applied to learn the similarity between users for recommendation [16].

Latent factor-based models usually approximate the user–item rating matrix by the product of two low dimensional matrices representing users and items respectively. In contrast to deep learning, these kinds of models can be seen as shallow models that predict the user–item interaction based on a linearly one-layer product multiplication. Matrix factorization (MF) [3,17] is one of the popular latent factor-based models, further studied by SVD++ [18] considering user/item basis and PMF [19] assuming each rating drawn from a conditional distribution. Also, other researches have been focused on enhancing MF by incorporating social network [20], topic model of item content [21], sparsity constraints [22], temporal behaviors [23] and cross-domain knowledge transfer [24].

Recently, inspired by the success of deep learning methods on computer vision, speech recognition and natural language processing, different deep learning models have been applied to CF tasks. Compared to the above shallow latent factor models, Wu et al. [25] learns the user latent feature from the user–item rating vectors and both user's and item's latent representation are learned to integrate additional side information by stacked denoising auto-encoder (SDAE) [26,27] or attention and gate mechanism [28]. To overcome the defect of MF that the interaction is simply predicted by summing the multiplication of corresponding value in latent features linearly, i.e, inner product, Neural Matrix Factorization (NeuMF) [4] generalizes the inner product function of MF by a nonlinear activation function on weighted sum of the element-wise product multiplication of user and item latent features and a multiple layer perception (MLP) on the concatenation of user and item latent features. Others focus on modeling the user–item interaction using graph convolutional networks (GCN) [29,30] on user–item bipartite graph to pass messages between users and items or autoencoder neural network [31] to reduce feature dimension and obtain item latent features for initializing SVD++.

2.2. Self-paced learning

Noisy samples and outliers usually exist in real data mining applications, which will harm the effectiveness of used methods. Thus, how to overcome or relieve this effect has always been an key issue, and attracts much attention from research fields. Among these robust statistic methods, curriculum learning (CL), inspired by the natural cognitive principle of humans or animals, was firstly proposed in [32]. The core idea of CL is that easier instances should be involved firstly in the model learning, and then more complex ones are gradually considered. This tactic is empirically evaluated that it can benefits the learning process for alleviating the bad local minimum, leading to the superior performance in some tasks [33].

Original CL strategy usually requires a priori identification to distinguish the easy instances from the hard ones, which is not well adapted to many tasks. Thus, rather than heuristic sample easiness ranking, Self-Paced Learning (SPL) [34] was formulated as a concise optimized model by incorporating a regularization term into the optimization objective to automatically select instances. Specifically, SPL introduces a trainable weight assigned for each instance into the loss function of model learning, along with a new regularizer to guide the weight learning. As a result, instances with higher learning error (for example prediction error or reconstruction error in different tasks) would be defined as confidence instances, or named outliers and noises.

It is exciting that SPL can automatically include instances with small errors for training the model so as to suppress the negative effect raised by outliers. Compared to CL, SPL can simultaneously optimize the learning objective of original model and the curriculum, so that the curriculum and model are consistent under the same optimization problem. Due to the virtue of its generality, SPL was introduced in a wide area of latent variable models, such as network embedding [7], clustering [8] and matrix factorization [35]. Therefore, embedding self-paced learning into a recommendation tasks can also effectively improve the performance. It is noted that our study differs from this stream of research by focusing on alleviating the affect of noisy instance by proposing a bounded SPL schema that will not include all instances in the final learning.

3. Self-paced neural matrix factorization for recommendation

In this section, we will present the details of our proposed method, with firstly introducing the MF-based collaborative filtering method and NeuMF, which is what we are mainly focusing on. Then, we show how our methods work with the present optimization objective and algorithms.

3.1. Matrix factorization for recommendation

Given the user–item historical interaction matrix $R \in \mathbb{R}^{n\times m}$ with n users and m items, collaborative filtering-based recommendation can be viewed as a matrix complete problem. In the latent factor model, each user and item are assigned with a latent vector to represent user's preference and item's feature. This can also be represented by the embedding layer with one-hot vector as input. Specifically, each user is denoted by a one-hot vector, $x_u \in \mathbb{R}^n$; each item is also represented by a one-hot vector $x_v \in \mathbb{R}^m$. All users share an embedding matrix $P \in \mathbb{R}^{d\times n}$, and items share an embedding matrix $Q \in \mathbb{R}^{d\times m}$, where d is the size of embedding vector. Accordingly, each user and item's latent vectors $p \in \mathbb{R}^d$ and $q \in \mathbb{R}^d$ can be converted by embedding matrices, as

$$p = Px_u; \quad q = Qx_v. \tag{1}$$

The embedding matrices P and Q for users and items are exactly what to learn in the training process, with supervised ground truth of user–item interaction records and sampled negative instances.

For MF [3], it approximates the user–item interaction elements r_{ij} by a simply inner product of two corresponding vectors,

$$r_{ij} = \langle u_i, v_j\rangle \text{ or } R \approx P^TQ. \tag{2}$$

Obviously R is a sparse matrix as a single user only rates a few items and also a single item is rated by a few users in the real applications. It is noted that the rank d is much smaller than the total number of the users n and the items m. To recover the large number of missing ratings in R, low rank matrix factorization is usually to solve the following unconstrained minimization problem:

$$\min_{P,Q} \frac{1}{2}\sum_{i,j} I_{ij}(r_{ij} - p_i^T q_j)^2 + \frac{\lambda}{2}(\|P\|_F^2 + \|Q\|_F^2), \tag{3}$$

where $\|\cdot\|_F$ denotes the Frobenius norm of a matrix as $\|A\|_F^2 = \sum a_{ij}^2$. I_{ij} is the indicator of ratings, where $I_{ij} = 1$ if user i rated item j and $I_{ij} = 0$ otherwise. $\|P\|_F^2$ and $\|Q\|_F^2$ are two regularization terms to avoid overfitting, with the positive regularization coefficient $\lambda > 0$ to balance the weight between the data reconstruction error term and regularization term, which is set empirically. Eq. (3) is usually solved by the gradient descent method due to its scalability on large datasets. This may be one of the most popular collaborative filtering methods and can be parallelized in distributed system for large datasets.

3.2. Neural matrix factorization for recommendation

Now, to improve the robustness of MF, based on the embeddings of all users and items, the rating score can be modeled by the framework of neural matrix factorization (NeuMF), which consists of generalized matrix factorization (GMF) and multi-layer perceptron (MLP). Firstly, given the embeddings of user i and item j, i.e, p_i and q_j, GMF aggregates these two vectors with an element-wise product, as

$$\phi_{ij}^g = p_i \odot q_j, \tag{4}$$

where '$\odot$' denotes the element-wise product of two vectors. It is noted that the element-wise product of two vectors can be seen as the first step of computing the inner product of two vectors.

For MLP, given the latent feature vectors of users and items, it is intuitive to combine them by concatenating them, which has been widely adopted in multi-modal deep learning work [36]. It is obvious that a simply concatenation of two vectors would not take into account the interactions between user and item latent vectors, which is insufficient for constructing the rating and social relations. To overcome this issue, the MLP part applies multiple hidden layers on vector concatenation to model the interaction between the latent vectors of users and items. Thus, a high level of flexibility and non-linearity to learn the interaction between $\langle p_i, q_j\rangle$ would be achieved, rather than only a fixed element-wise multiplication on them. Specially, the MLP models to predict the rating are defined as

$$\phi_{i,j}^m = ReLU(h_L^r(\ldots ReLU(h_1^r(p_i \oplus p_j)))). \tag{5}$$

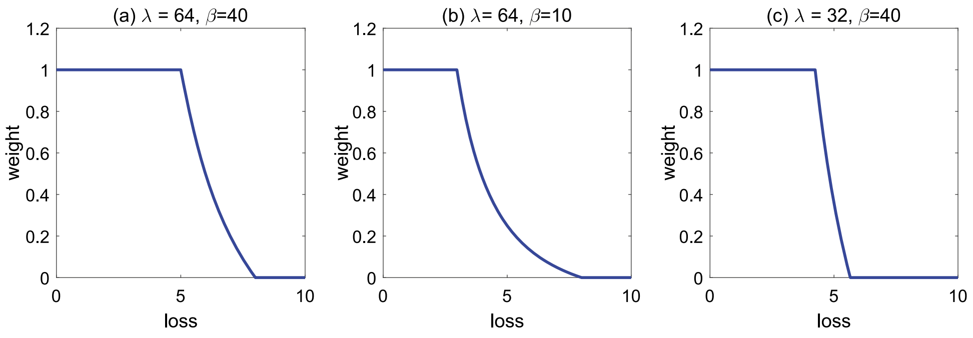

Fig. 1. Comparison of weighting function with different λ and β.

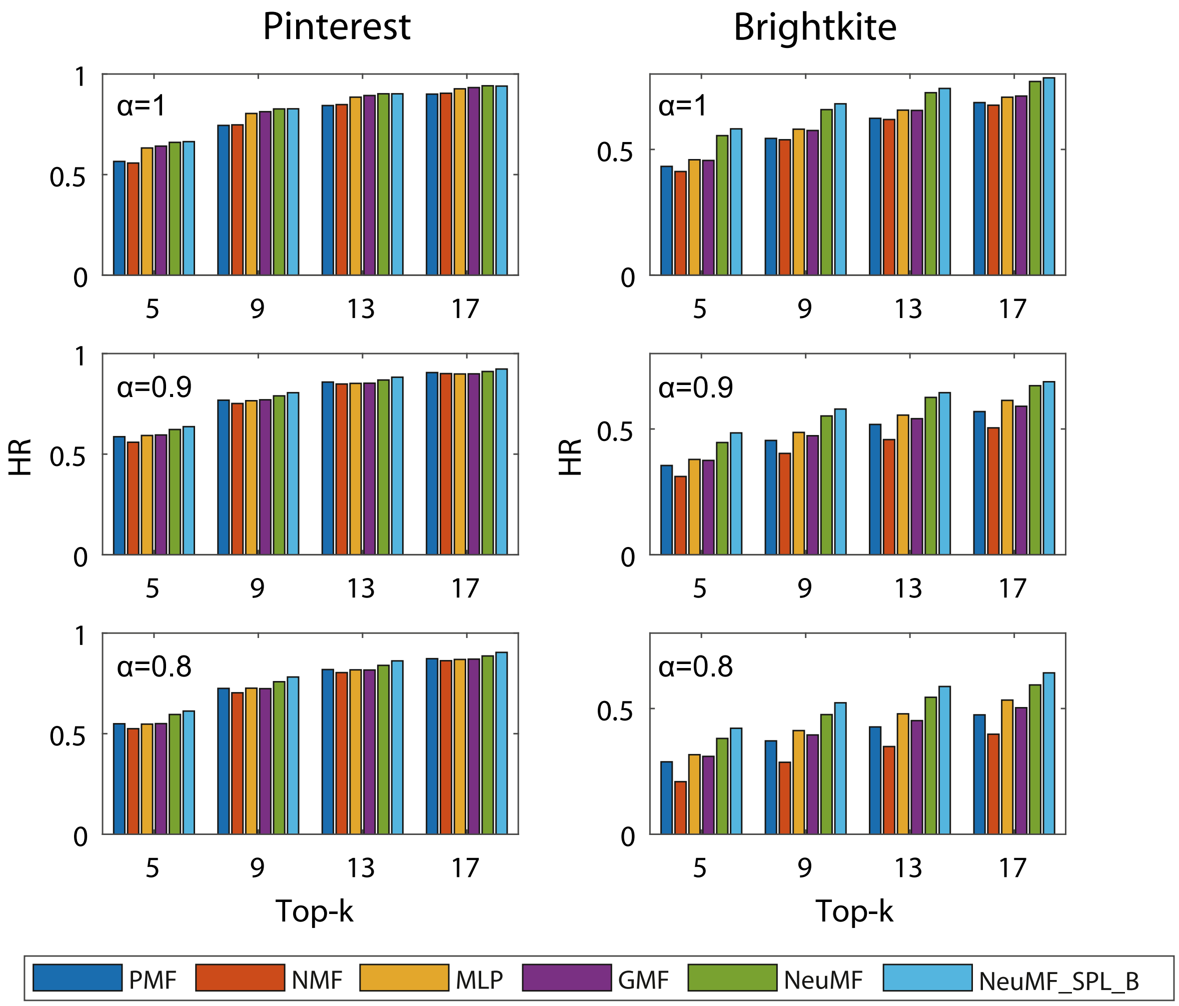

Fig. 2. (Color online) The HR comparisons of top-k recommendation for six models on datasets of Brightkite and Pinterest.

where, $h(x) = W^T x + b$ is the hidden layer (W is the weight matrix, and b is the bias vector); '$\oplus$' denotes the concatenation manipulation of two vectors. Overall, $\theta_h = \{W_l^r, W_l^a, b_l^r, b_l^a\}$ ($l = 1, 2..., L$) denotes a set of parameters of each hidden layer, which will be learned when training the model. As to the activation function, Rectified Linear Unit (ReLU) has been empirically shown [4] to yield slightly better performance that *tanh* and *sigmoid*. For the network structure, tower pattern that the bottom layer has more neurons than successive layer is a common design and it can learn more abstractive features by a small number of hidden neurons in higher layers [37]. Also, to reinforce the flexibility of the model, we allow GMF and MLP to learn separate embeddings, that is, we define two different embedding matrices (P^g and P^m, Q^g and Q^m) to get the corresponding latent features, as the optimal embedding size of the two models may vary for some datasets.

Now we can see that GMF can learn the linear interaction between latent features and MLP can model a non-linear interaction, and the results could be fused so that they can mutually enhance each other to better understand the complex user–item interactions. By using a one-layer perceptron on the concatenation of the last hidden layer of GMF and MLP, we can output the prediction of ratings as

$$\hat{r}_{ij} = \sigma(h_{out}(\phi_{ij}^g \oplus \phi_{ij}^m)), \tag{6}$$

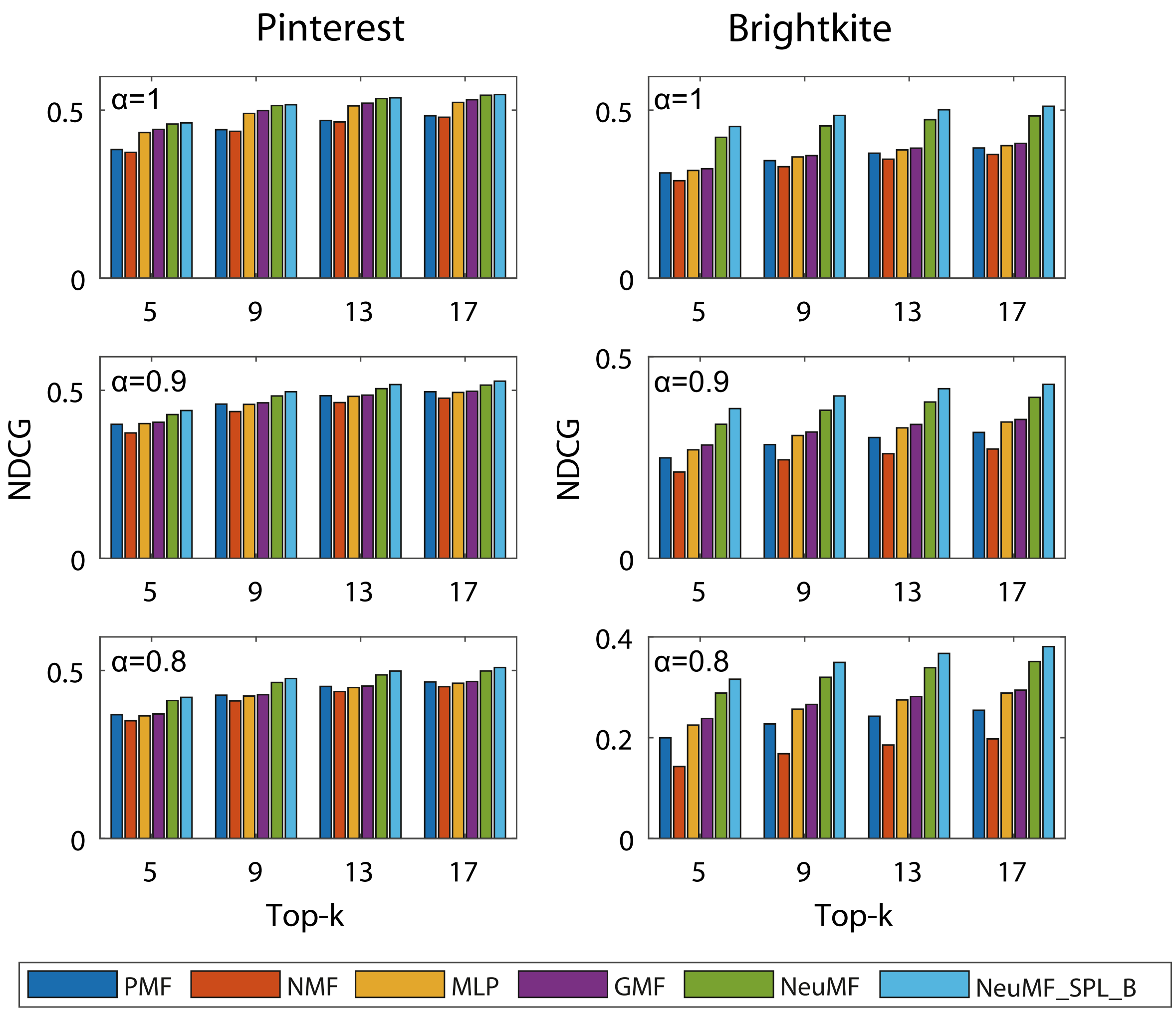

Fig. 3. (Color online) The NDCG comparisons of top-k recommendation for six models on datasets of Brightkite and Pinterest.

where, $h_{out}(x) = w^T x + b$ is the hidden layer (w is the weight vector, and b is the bias parameter) containing weight vector and bias as parameters learned from data, and $\sigma(x) = 1/(1 + e^{-x})$ is the nonlinear *sigmoid* function.

Generally, the rating score r_{ij} in collaborating filtering tasks could be real value ([1,5]). Also, if we focus on the collaborating filtering tasks with implicit data, such as browsing, clicking, or commenting, the target score r_{ij} is usually measured be a binary value. Under this setting, 1 means user u_i has interacted with item v_j and 0 otherwise. Then, the prediction score $\hat{r}_{ij}$ could be seen as how likely there will be an interaction between user i and item j. Thus, the likelihood function could be defined as

$$L(\mathcal{R}, \mathcal{R}^- | P, Q, H) = \prod_{(i,j)\in\mathcal{R}} \hat{r}_{ij} \prod_{(i,j)\in\mathcal{R}^-} (1 - \hat{r}_{ij}), \tag{7}$$

where $\mathcal{R}$ denotes the set of observed user–item interactions; $\mathcal{R}^-$ is the corresponding sets of negative instances. $\{P, Q, H\}$ are the trainable parameters to learn from the data, where H includes parameters of hidden layers h in the above introduction. Taking the negative logarithm of the above likelihood, we can reach the loss function as the binary cross-entropy loss:

$$\begin{aligned} \ell_r &= -\sum_{(i,j)\in\mathcal{R}} log\hat{r}_{ij} - \sum_{(i,j)\in\mathcal{R}^-} log(1 - \hat{r}_{ij}) \\ &= -\sum_{(i,j)\in\mathcal{R}\cup\mathcal{R}^-} [r_{ij} log\hat{r}_{ij} + (1 - r_{ij}) log(1 - \hat{r}_{ij})]. \end{aligned} \tag{8}$$

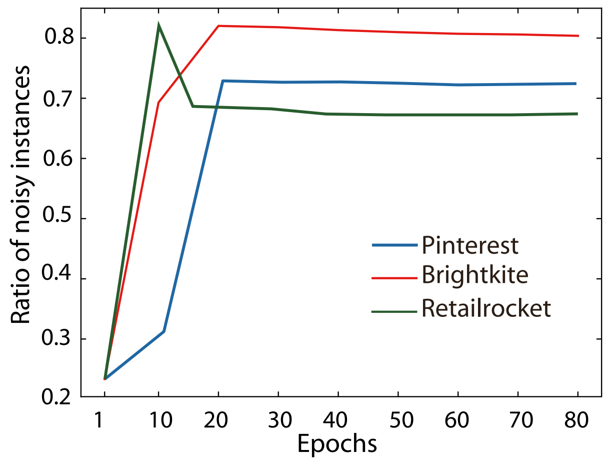

Fig. 4. (Color online) Ratios of the noisy instances assigned with soft weights to the total of the instances having soft weights vs. epochs of the model training.

Usually, Eq. (8) can be optimized by stochastic gradient descent (SGD) method. However, noises and outliers will be treated equally for the update of the parameters during the process of model training, and the outliers in the training instances will greatly harm the efficiency of the model. To overcome this issue, we will introduce self-paced learning (SPL) mechanism to

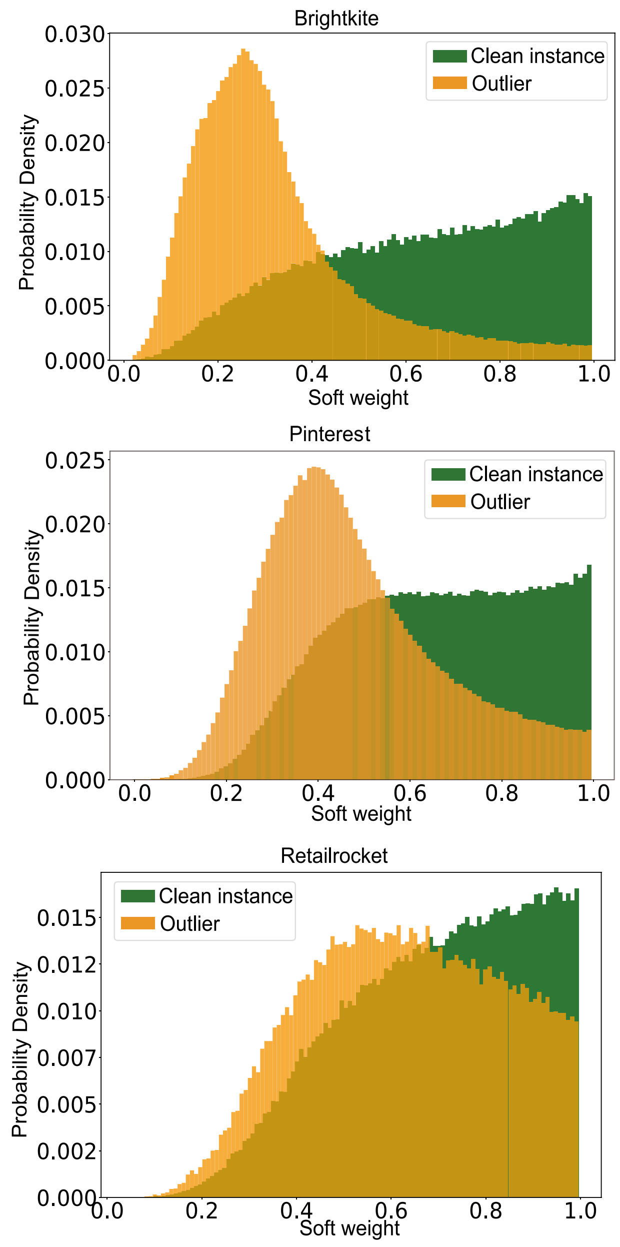

Fig. 5. (Color online) Probability density distributions for both outliers and clean instances having soft weights on datasets of Brightkite, Pinterest, and Retailrocket.

improve the robustness of NeuMF, which is a kind of learning method to simulate the cognitive nature of human/animals, that understanding of things is from simple knowledge to complex knowledge.

3.3. *Self-paced learning-based neural matrix factorization for recommendation*

The main idea of SPL is that it inclines to selecting simple instances in each iteration to update the model parameters. In the optimization-based SPL framework [38], whether an instance is included for each iteration is determined by the learning parameters, and more instances would be included by gradually increasing the parameters. Specifically, SPL introduce a new parameter to weight each instance in the traditional learning objective of different tasks, as the following objective function.

$$\min_{w,v} \quad \mathbb{E}(w, v, \lambda) = \sum_{(x_i,y_i)\in\mathcal{D}} v_i \ell(y_i; g(x_i, w)) + f(v; \lambda, \beta), \tag{9}$$

where $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$ is a training set ; $\ell(y_i; g(x_i, w))$ denotes the loss term to calculate the cost of the difference between the true label y_i and learned label $g(x_i, w)$ under a model g with w as its model parameter to learn; $v = [v^1, v^2, \ldots, v^N]^T \in [0, 1]^N$ represents the weight vector measuring the easiness of instances $\{x_i\}$ to determine whether an instance is selected; f corresponds to a self-paced regularizer function for the instance selection scheme; λ and β as parameters to control the instance selection pace, which can be interpreted as "pace age".

Incorporating the objective of NeuMF in Eq. (8) into the framework of SPL in Eq. (9), we propose the objective function of self-paced learning enhanced neural matrix factorization as follows:

$$\min_{\theta,v} \sum_{(i,j)\in\mathcal{R}\cup\mathcal{R}^-} -v_{(i,j)}[r_{ij} log \hat{r}_{ij} + (1 - r_{ij}) log(1 - \hat{r}_{ij})] + f(v; \lambda), \tag{10}$$

where each user–item pair (i, j) is assigned a weight $v_{(i,j)}$ to determine how much possible training instance (i, j) would be included; θ denotes the learned parameters $\{P, Q, H\}$.

Here, instead of hard weighting schema, in this work, we adopt the soft weighting schema [38] as it is relatively easy to implement and well adapted to complex applications, and it has been shown to perform better than hard weighting [9]. The weights of instances are linearly penalized by this regularizer in terms of the loss function. It is specifically defined as

$$f(v; \lambda, \beta) = \sum_{(i,j)\in\mathcal{R}\cup\mathcal{R}^-} v_{(i,j)} \frac{\beta^2}{v_{(i,j)} + \beta/\lambda}. \tag{11}$$

It is noting that Eq. (11) is convex with regards to $v_{(i,j)}$, and with fixed parameters θ, Eq. (10) becomes:

$$\min_{v_{(i,j)}\in[0,1]} \sum_{(i,j)\in\mathcal{R}\cup\mathcal{R}^-} v_{(i,j)}\ell_{(i,j)} + f(v; \lambda, \beta), \tag{12}$$

where the loss for each instance is defined according to Eq. (8) as:

$$\ell_{(i,j)} = -r_{ij} log \hat{r}_{ij} - (1 - r_{ij}) log(1 - \hat{r}_{ij}). \tag{13}$$

Then, the global minimum $v_{(i,j)}$ can be reached with analytical solution for the linear soft weighting objective, as

$$v^*_{(i,j)}(\lambda, \beta) = \begin{cases} 1, & \ell_{(i,j)} \le \frac{1}{\sqrt{1/\lambda + 1/\beta}} \\ 0, & \ell_{(i,j)} \ge \sqrt{\lambda} \\ \beta(\frac{1}{\ell_{(i,j)}{}^2} - \frac{1}{\lambda}), & otherwise \end{cases} \tag{14}$$

It can be seen that if an instance (i, j) is well predicted by the neural matrix factorization model as in Eq. (6), i.e, $\ell_{(i,j)} < \sqrt{\lambda}$, it will be included as an easy instance ($v_{(i,j)} > 0$), otherwise excluded ($v_i = 0$). The weight is decreasing as to the cross-entropy loss $\ell_{(i,j)}$. Obviously, more instances will be excluded when λ is smaller since their corresponding cross-entropy losses are more likely to be greater than $\sqrt{\lambda}$, and more instances with bigger losses will be probably introduced when λ becomes larger. In the optimization, λ would be gradually increased. Therefore, the parameter λ is to control the pace at which the model include new instances, and naturally it can be referred as the 'age' of the model. Another parameter β will control the smoothness of weights greater than 0. If β is smaller, the bound $\frac{1}{\sqrt{1/\lambda+1/\beta}}$ would be smaller. By this setting, cross-entropy losses are less likely to be smaller than this bound, meaning less instances are treated as a faithfully easy samples weighted by 1 and leading to a smooth weight allocation. The examples of $v^*_{(i,j)}(\lambda, \beta)$ tend curve with respect to ℓ is shown in Fig. 1.

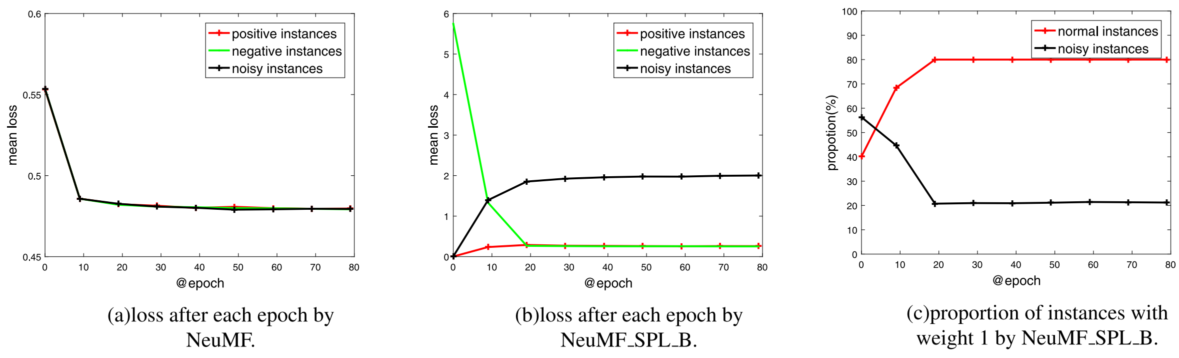

Fig. 6. (Color online) Example of the effect of NeuMF_SPL_B on noisy detection. We run the NeuMF and our proposed method on Pinterest dataset with randomly 20% noises and show the mean construction loss of normal instances and noisy instances, and how many normal instances and noisy instances are completely selected on each epoch.

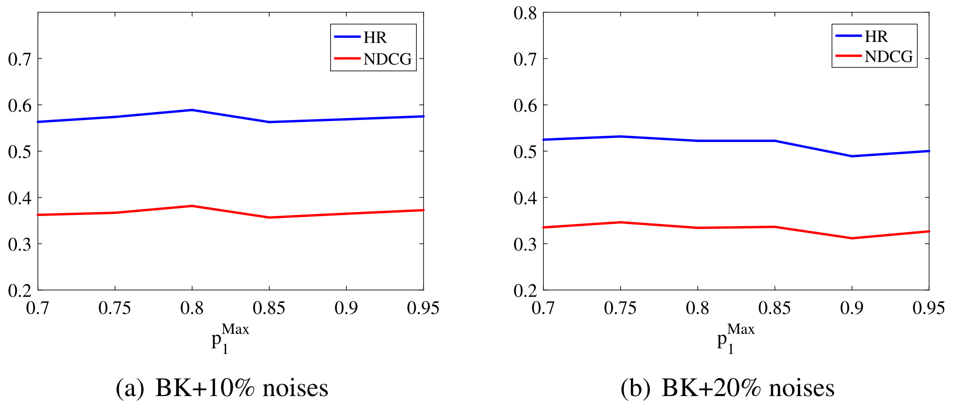

Fig. 7. (Color online) Recommendation results of proposed NeuMF_SPL with different setting of hyper-parameter p_1^{Max} on Brightkite data with noises.

3.4. Discussions on the loss of noisy data

The learning objective in Eq. (10) is designed for all the training instances that may contain noise corrupted data. Usually, we do not know which instances in the training set are noises and which are not. Thus, our goal is to recognize the noises from the normal data by this optimization process, that is, the loss of noisy instances should be larger than clean data. Here we can prove it by a simple theorem.

Theorem 1. *The expected loss ℓ in Eq. (13) of noisy instances is bigger than clean instances and the weights w of instances are thus smaller.*

Proof. Supposing that $\hat{R}$ denote the predicted user–item interactions by a machine learning model; R and R' respectively denote the user–item interactions of clean instances and noisy instances. Thus, based on the general understanding of noises, the $\hat{R}$ should be correlated to normal data R while it is independent to noises R' as noisy data are usually generated randomly, i.e, $Cov(\hat{R}, R) > 0$, $Cov(\hat{R}, R') = 0$.

According to the definition of K-L divergence,[1] the K-L divergence of a given variable between an independent variable is much bigger than a correlated variable, we have:

$$D_{KL}(\hat{R}\|R') > D_{KL}(\hat{R}\|R).$$

According to basic property of cross-entropy and K-L divergence, the relationship between these two is:

$$D_{KL}(\hat{R}\|R') = H(\hat{R}, R') - H(\hat{R}),$$

$$D_{KL}(\hat{R}\|R) = H(\hat{R}, R) - H(\hat{R}),$$

where '$H(\cdot)$' denotes the entropy of a random variable and '$H(A, B)$' denotes the cross-entropy of two random variables. Accordingly,

$$H(\hat{R}, R') > H(\hat{R}, R). \tag{15}$$

Thus, the expected cross-entropy loss $E(\ell_{R'})$ on noisy instances is bigger than $E(\ell_R)$ on clean data. According the properties of SPL regularizer as in Eq. (14) and Fig. 1, the expected weights of noisy instance would be smaller.

3.5. Optimization algorithm with bounded SPL

When looking into the process of model learning by traditional SPL, all instances will be included with increasingly λ after certain iterations, even though it is selecting easy instance in the first iterations. It means that if there exists many outliers in the training set, the model learning would consider these outliers when updating parameters, which is probably to cause low efficiency. Therefore, we propose a bounded SPL learning mechanism that only clean instances would be selected in the model learning and not all instances are completely included, which indicates that the influence of outliers on the model learning may be reduced.

Specifically, as in Algorithm 1, we introduce a parameter p_1^{Max} to control the maximum ratio of instances that are assigned to a weight as $v_{(i,j)} = 1$, that is, the remaining $1 - p_1^{Max}$ of instances would be assigned to a soft weight between 0 and 1. Accordingly, if we can well identify outliers from normal instances, those outliers would be assigned with relatively small weights among the remaining $1-p_1^{Max}$ of instances. Obviously, much bigger or smaller

[1] https://adventuresinmachinelearning.com/cross-entropy-kl-divergence/.

parameter of p_1^{Max} would not be a good choose. If we set a larger p_1^{Max}, meaning that almost all instances are completely selected and noisy instance would affect the learning process much. If we set a smaller p_1^{Max}, indicating that a smaller ratio of instances are selected and the model is only trained on few instances. Thus, the setting of p_1^{Max} for real problems should be considered to balance these two effects. We note the proposed Bounded Self-paced Learning Enhanced Neural Matrix Factorization as **NeuMF_SPL_B**.

Stochastic gradient descent (SGD) method is a popular method for optimizing the objective function in Eq. (10). Specifically, we use a robust version mini-batch gradient descent (MBSD) strategy, which is a robust version as it considers the weight of each sample when computing the gradient of a batch by summing the gradients of all samples. The gradients of $\{P, Q, H\}$ can be computed by the back propagation algorithm, which we omit due to the interest of space. Usually, the number of unobserved user–item interactions in $\mathcal{R}^-$ is much bigger than observed instances, we will perform negative sampling on R (for each positive instance (i, j) we sample ns instances from $\mathcal{R}_i^-$), described in Algorithm 1. In Algorithm 1, $\mathcal{R}_i^-$ denotes the set of items user u_i has interacted with. In the algorithm, an instance is defined as an 4-tuple $(i, j, v_{(i,j)}, r_{ij})$, meaning we assign a weight $v_{(i,j)}$ to the interaction r_{ij} of user i on in item j.

Also, different from traditional SPL learning that increases the learning pace parameter λ by a exponential function (e.g. $\lambda^{(t)} = \lambda^{(0)}\mu^{t-1}$), we introduce a heuristic function **DetermineLamadaBeta**() to update the parameters λ and β given a ratio that controls how many instances are completely selected (weight equals 1) and how many are assigned to a soft weight, as in Algorithm 2. **quickSort** denotes basic quick sort algorithm.

From the line 18 of Algorithm 1, we can see that, the parameter 'p_1^{Max}' gives a upper bound of how many instances are completely included in the learning and '$1 - p_1^{Max}$' instances are allocated with a soft weight between 0 and 1. Under the learning merchants, noisy instances are expected to belong to the part of '$1 - p_1^{Max}$', and thus the learning process can be lightly affected by the noisy instances. According, we name it as 'NeuMF_SPL_B' that could recognize noises and alleviate the influence of them for model learning.

```
Input      : The set of observed user–item interactions $\mathcal{R}$.
Parameters : Hyper-parameters: the ratio of negative
             samples $ns$; the batch size $batch\_size$; the
             maximum ratio of completed selected
             instances $p_1^{Max} < 100\%$; $\Delta r_1$ and $\Delta r_2$ for
             updating learning pace.
Output     : $\Theta = \{P,Q,H\}$
1  Initialization: $P,Q,H$; $r_1^{(0)} < p_1^{Max}$; $r_2^{(0)} < 100\%$;
2  repeat
3      $Batch \leftarrow \emptyset$; $t \leftarrow 0$
4      $(\lambda, \beta) \leftarrow$ DetermineLamadaBeta($r_1^{(t)}, r_2^{(t)}$); //updating
       the learning pace.
5      Update $v^{(t)}$ via Eq. (14);
6      for $(i, j) \in \mathcal{R}$ do
           //get a positive training instance:
7          $Batch \leftarrow Batch \cup \{(i, j, v_{(i,j)}^{(t)}, 1)\}$;
8          for $ng \leftarrow 1 : ns$ do
               //negative sampling.
9              $j \leftarrow$ draw a random item from $\mathcal{R}_i^-$ ;
               //get a negative training instance:
10             $Batch \leftarrow Batch \cup \{(i, j, v_{(i,j)}^{(t)}, 0)\}$;
11         end
12         if $|Batch| >= batch\_size$ then
13             compute gradients of $P,Q,\Theta$ with updated
               weights;
14             Update the above parameters by gradient
               descent;
15             $Batch \leftarrow \emptyset$ ;
16         end
17     end
18     $r_1^{(t+1)} = min(p_1^{Max}, r_1^{(t)} + \Delta r_1)$;
19     $r_2^{(t+1)} = min(100\%, r_2^{(t)} + \Delta r_2)$;
20     $t \leftarrow t + 1$;
21 until until convergence;
```

Algorithm 1: An Optimization Algorithm of proposed NeuMF_SPL_B

```
Input      : The model learning results at $t$ iteration.
Parameters : The ratio of completely selected instance $r_1$;
             The ratio of selected instance $r_2$.
Output     : $\lambda, \beta$
1  Compute the loss of each instance $\ell_{(i,j)}$ at $t$-th iteration by
   Eq. (13);
2  Sort all the losses $\{\ell_{(i,j)}\}_{(i,j)\in\mathcal{R}\cup\mathcal{R}^-}$ by quickSort in
   descending order;
3  $\lambda \leftarrow$ square of the $(1 - r_2)$-th biggest loss in $\{\ell_{(i,j)}\}$;
4  $\ell_{r1} \leftarrow$ the $r_1$-th smallest loss in $\{\ell_{(i,j)}\}$;
5  $\beta \leftarrow 1/(\frac{1}{\ell_{r1}^2} - \frac{1}{\lambda})$;
```

Algorithm 2: Pseudo code for **DetermineLamadaBeta**(r_1, r_2)

4. Experiment results

4.1. Data description

The datasets adopted in this article for model evaluation include Brightkite, Pinterest, and Retailrocket which are widely utilized for the study of information filtering and recommendation. Brightkite was once an LBS (Location-Based Social discovery network) which allows users to share their locations by using text messages or mobile apps and each location sharing is also known as a check-in [39]. Here, similar to the classic problem of product recommendation, we suppose the past check-ins as the "items" interacted by the users and predict the users' future check-ins on POIs (point-of-interests). Please note that some pre-treatments to the original data were done initially to reduce the sparsity of the dataset including those users with less than ten locations sharing actions and locations which were visited by no more than ten users are all removed. Pinterest is an online community focusing on image collection and sharing. A user pinning an image is recorded as an implicit feedback. Due to the sparsity of the original data, He et al. [4] removes the users who have rare actions of pin and, as such, only keeps the users having no less than 20 interactions (pins). Retailrocket is an on-line shopping dataset, which collected the daily data of real e-commerce website users including the behavior data of website visitors within 4.5 months. The behavior is divided into three categories: click, adding to shopping cart, and transaction. There are a total of

Table 1
Statistics of three real-world datasets.

Datasets	#Users	#Items	#Records	#Testing set
Brightkite	11,163	5019	2,931,055	1,116,300
Pinterest	55,187	9916	1,408,394	5,518,700
Retailrocket	7600	65,562	211,706	760,000

Table 2
Summary of HR and NDCG results for Top-10 recommendation on nine datasets. The best results are marked in bold-face. 'BK', 'PT', and 'RR' denote datasets of Brightkite, Pinterest, and Retailrocket, respectively. '*N*' is the abbreviation of noises. The last two columns correspond to the *t* and *p* of the significance test by comparing the HR and NDCG of NeuMF_SPL_B to NeuMF over all testing instances.[2]

Datasets	Metrics	PMF	NMF	MLP	GMF	NeuMF	NeuMF_SPL_B	*t*	*p*-value
BK	HR	0.5873	0.5837	0.6227	0.6209	0.6936	**0.7166**	3.8	<0.0001
	NDCG	0.3621	0.3446	0.3727	0.3778	0.4631	**0.4943**	4.7	<0.0001
BK + 10%N	HR	0.4900	0.4314	0.5239	0.5103	0.5919	**0.6127**	3.2	0.0007
	NDCG	0.2921	0.2525	0.3152	0.3239	0.3787	**0.4122**	5.1	<0.0001
BK + 20%N	HR	0.4023	0.3193	0.4479	0.4241	0.5131	**0.5577**	6.7	<0.0001
	NDCG	0.2361	0.1776	0.2668	0.2743	0.3305	**0.3593**	4.5	<0.0001
PT	HR	0.8415	0.8141	0.8510	0.8596	**0.8701**	0.8700	0	0.4803
	NDCG	0.4917	0.4599	0.5036	0.5121	0.5261	**0.5284**	0.8	0.2221
PT + 10%N	HR	0.8205	0.8084	0.8149	0.8193	0.8351	**0.8519**	7.7	<0.0001
	NDCG	0.4738	0.4529	0.4722	0.4768	0.4961	**0.5090**	4.3	<0.0001
PT + 20%N	HR	0.7787	0.7610	0.7794	0.7774	0.8042	**0.8290**	10.6	<0.0001
	NDCG	0.4419	0.4259	0.4391	0.4433	0.4775	**0.4896**	4.0	<0.0001
RR	HR	0.4160	0.3427	0.4125	0.4013	0.4391	**0.4762**	4.6	<0.0001
	NDCG	0.2318	0.1782	0.2280	0.2214	0.2566	**0.3022**	6.3	<0.0001
RR + 10%N	HR	0.3623	0.3077	0.3817	0.3732	0.3774	**0.4076**	3.8	<0.0001
	NDCG	0.1986	0.1550	0.2129	0.2103	0.2155	**0.2311**	2.3	<0.0001
RR + 20%N	HR	0.3242	0.2776	0.3682	0.3229	0.3693	**0.3957**	3.3	<0.0001
	NDCG	0.1745	0.1396	0.2079	0.1849	0.2094	**0.2228**	2.0	<0.0001

2,756,101 behavioral events from 1,407,580 visitors, of which include 2,666,312 of browsing behaviors, 69,332 of adding to shopping cart behaviors, and 22,457 of transaction behaviors. Here, what we are using is a sub-set of the data on clicks. Some detailed statistics of the three datasets are summarized in Table 1. For generating Top-*k* recommendation list, for each user, one item is randomly selected as testing set and 99 un-interacted items are randomly chosen as negative samples, then we sort the predicted scores of 100 items (99 un-interacted items and 1 ground-truth item) to generate recommendation lists.

4.2. Parameter settings

For comparison, four matrix-factorized recommendation models are selected as state-of-the-art baselines which are NMF (Non-negative Matrix Factorization) [3], PMF (Probabilistic Matrix Factorization) [41], GMF (Generalized Matrix Factorization) [4] and NeuMF (Neural Matrix Factorization) [4], respectively. At the meantime, MLP (Multi-layer perceptron) is also selected as a baseline of the recommendation model. In case of fairness, all the competing models have been set the optimal parameters which are suggested by the literature. In detail, the learning rate is set as $lr = 0.001$ and factor size $d = 8$ for all models. As for NMF, the Batch size equals 1024. For GMF, NeuMF, and NeuMF_SPL_B, the negative sampling ratio *ns* is set as 4, meaning for each observed user–item interaction records, we randomly sample 4 items from unobserved item sets of this user. In particular, for proposed NeuMF_SPL_B: we set Δr_1 and Δr_2 as 5%; we set p_1^{Max} as 80%, meaning 80% of all instances would be finally completely selected and other 20% are assigned with a weight within (0,1); the initial $r_1^{(0)}$ and $r_2^{(0)}$ are set as 20% and 50% respectively, meaning that 20% of instances are completely selected in the first iteration and 50%–20% = 30% of instances are assigned with a soft weight within (0,1) with remaining 50% excluded.

4.3. Experimental setup

We use two popular metrics including Hit ratio (HR) and Normalized Discounted Cumulative Gain (NDCG) to assess the model's performance on information recommendation. First of all, we conduct the experiments which focus on top-10 recommendation. Here, in order to test the proposed model's performance of recommendation in the noisy contexts, we specialized to introduce noisy data into the original datasets. The experimental setup can be described as follows. As the number of negative instances is far more than the positive ones in the two datasets, we devise a sampling strategy in which one positive instance corresponds to 4 randomly sampled negative instances at a sampling step. This step will be performed repeatedly for every positive instance existing in the training set. Note that the adopted sampling rate of positive instances to negative ones is based on the analyses made by He et al. [4] in their study.

Meanwhile, in each step of sampling negative instances, a random number within (0, 1) will be generated which follows the uniform distribution. If it is larger than a threshold like 0.9, the sampled negative instance is labeled intentionally as positive ones. As a result, for all the sampled negative instances, 10% of them will be converted to the outlier or noisy data as stated in Definition 1, i.e., forged positive instances. In practice, the mentioned threshold can be treated as a noise parameter α. Without loss of generality, setting α as 0.9 and 0.8 for the two tested datasets, we will generate and obtain another six new datesets with 10% and 20% corrupted negative instances. Our goal is to figure out, compared to the conventional methods, how good our model could make recommendation with these injected noisy data.

4.4. Comparison results

The results summarized in Table 2 demonstrate that, on PT dataset without noises, the performance of proposed NeuMF_SPL_B is slightly better than those for NeuMF (the improvement is not significant on PT as shown in the last two

[2] Here, we can treat the observed HR or NDCG as the proportion of users in the test set, which are provided an accurate recommended item by models. It means that HR and NDCG are a kind of proportion of successful tries of recommendation from a sample containing all testing instances, as they are both in [0,1]. Thus, according to the principle of hypothesis test of the proportion between two populations [40], t can be computed as:

$$t(n-1) = (p_1 - p_2)/\sqrt{\frac{p(1-p)}{n} + \frac{p(1-p)}{n}}, \qquad (16)$$

where p_1 and p_2 represent the HR (or NDCG) of NeuMF_SPL_B and NeuMF respectively; p represents the average of p_1 and p_2; n is the number of users in test set and $n-1$ denotes the degree of freedom of T distribution. Then, single-tailed p-value can be computed by T distribution.

Table 3
Summary of the number of epochs and required time (Seconds) for convergence of the six models on nine datasets. 'BK', 'PT', and 'RR' denote datasets of Brightkite, Pinterest, and Retailrocket, respectively. 'N' is the abbreviation of noises.

Datasets	Metrics	PMF	NMF	MLP	GMF	NeuMF	NeuMF_SPL_B
BK	Epochs	80	76	23	12	6	15
	Time (s)	4.6	95	4.6	4	4.8	11.7
BK + 10%N	Epochs	80	76	6	13	10	16
	Time (s)	4.8	96.3	4.7	4.1	4.9	12.8
BK + 20%N	Epochs	80	76	5	9	10	17
	Time (s)	4.6	97.5	4.8	4.1	5	13.1
PT	Epochs	80	70	16	5	10	10
	Time (s)	26.4	423	26.8	26.2	30.3	50.1
PT + 10%N	Epochs	80	70	13	9	9	16
	Time (s)	27.8	411.2	26.2	24.6	30.4	56.3
PT + 20%N	Epochs	80	70	13	10	11	17
	Time (s)	28.2	415.5	27.6	26.5	29	90.2
RR	Epochs	80	20	5	8	10	11
	Time (s)	4.7	45	3	2.6	3.5	7.6
RR + 10%N	Epochs	80	20	6	4	10	13
	Time (s)	4.9	48.3	2.9	2.7	3.7	7.4
RR + 20%N	Epochs	80	20	5	12	7	16
	Time (s)	5.1	48.9	2.9	2.6	3.6	7

columns), but it performs much better than others on other two datasets(the improvements all are significant on datasets with noises). When introducing more noise data, we notice that the proposed model NeuMF_SPL_B performs increasingly better than all of the competing models. The results suggest that the new model NeuMF_SPL_B can gain more accurate recommendations withstanding the perturb of the noisy data.

To further find more evidence to support the viewpoint that our model has the advantage for recommendation with the immunity of noisy data, we employ the length of recommendation list k as a changing parameter and set it as 5, 9, 13 and 17, respectively. From the Figs. 2 and 3, we can observe that the trends of the recommendation accuracy for all the tested models are mildly increasing with the growth of the length of recommendation list. When the original datasets are injected with noisy instances, as expected, our model performs best against the rest of the baseline methods on all k settings. This experiment clearly validates that our algorithm has the capability to resist the negative impact of the noisy data while training and can thus achieve a better recommender model via bounded self-paced learning compared with the traditional methods. Yet, it is worth noting that, compared to the accuracy results of recommendation on the dataset of Pinterest, the improvements of HR and NDCG on the Brightkite are more significant along with more sampled noisy instances introduced into the dataset (some more results on the dataset of Retailrocket can be found in the Section of Appendix).

4.5. Effect of noise detection

To further understand the superiority of the proposed model in detecting noisy instances, we try to look into the entire learning steps of the model. Fig. 4 shows that, for the three datasets, the ratio of the noisy instances to the total instances assigned with soft weights climbs up or has a small drop via about 15–20 of training epochs before finally reaches a steady value. It coincides with our expectation that self-paced learning indeed has the tendency to assign more and more noisy instances with soft weights which means these outliers will give less influence to the model. More deeply, we use the Probability Density Function (PDF) to observe the distribution of the soft weights on both clean instances and outliers at the final epoch as shown in Fig. 5. It turns out on all the three datasets that, compared to the clean instances, most outliers have smaller values of soft weights, indicating that outliers play a non-significant role in the model training. In particular, we also notice that most outliers for Brightkite have the smaller values of soft weights falling in (0.1, 0.5) compared with those for Pinterest whose soft weights mostly fall in (0.2, 0.7). This observation can interpret why our model performs slightly better on Brightkite than on Pinterest.

Also, for the optimization process, since all instances are treated the same, the noisy instances cannot be well distinguished from normal ones, provided that there is no significant difference of the reconstruction loss between real ratings and noisy ratings as in Fig. 6.(a). It means that the noisy instances would disturb the learning of normal instances and thus cause low efficiency. This new method can address the issues of noisy data well as reported in the experiments As shown in Fig. 6(b) that the losses on noisy instances[3] are much higher than original instances, and in the learning process less noisy instances are completely included with weight 1 as in Fig. 6(c). Extensive experimental results on three real datasets demonstrate the performance improvement of recommendation quality by proposed bounded self-paced learning, indicating that our model is superior to original neural matrix factorization model and other baselines.

4.6. Effect of hyper-parameter setting

In order to see how the performance varies with different setting of p_1^{Max}, maximum ratio of completed selected instances, we report the recommendation results of proposed NeuMF_SPL_B with different setting of hyper-parameter p_1^{Max} on Brightkite data with noises, as in Fig. 7. It can be seen that the performance measured by HR and NDCG do not change much with various parameter setting, showing the robustness of proposed method. It is noted that we only plot the robust results when p_1^{Max} in the range of [0.7, 0.95]. As it controls how many instances are completely included in the training processing, the much lower parameter (in the range of [0, 0.7]) will lead to less training instances and the performance metrics (HR, NDCG) of recommendation will be lower. Also, the empirical results from experiments also support this, and we only show the robustness on the big range of [0.7, 0.95] due to interest of space, which can provide a reference for the setting of this parameter in future application.

[3] We manually generate noises on 20% instances at random.

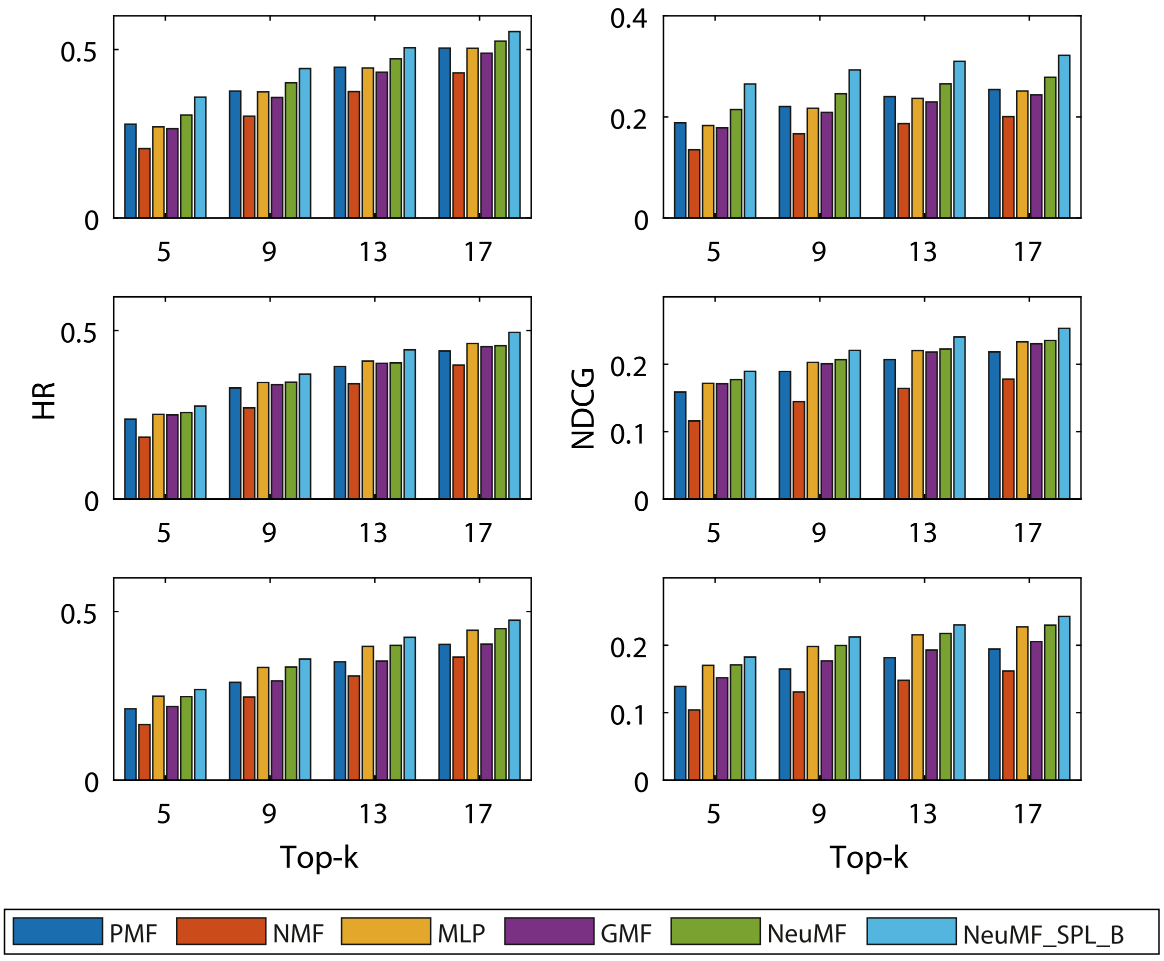

Fig. 8. (Color online) The HR and NDCG comparisons of top-k recommendation for six models on dataset of Retailrocket.

4.7. Efficiency analysis of the models

As the number of iterations for convergence is an important indicator which could reflect the efficiency of model training, we summarize the number of epochs required for convergence for the six competing models on the nine datasets and corresponding time consumption shown in Table 3. The machine used for testing is a desktop with Core i7-9750H 2.60 GHz Intel CPU, 16 GB memory, and NVIDIA GeForce GTX 1650 GPU. From the records, we do not observe the significant increase of the iterations or time consumption under the new learning framework. It suggests that the proposed model NeuMF_SPL_B has the similar training efficiency compared with the baselines. Moreover, compared to traditional matrix factorized models like NMF, NeuMF_SPL_B requires even fewer epochs and less time for convergence.

5. Conclusion

In this work, we presented a Bounded Self-paced Learning Enhanced Neural Matrix Factorization (NeuMF_SPL_B) framework to learn a robust model for personalized recommendation. Specially, we learn the latent embeddings of users and items by employing a generalized matrix factorization to model the complex nonlinear relationship of user–item interaction. To alleviate the affect of noisy instances and outliers on the model result, we propose to gradually select instances when learning the model by the mechanism of self-paced learning. More concretely, the weights of instances for learning the model would be increasingly updated after each epoch according to their prediction error, and we set a bound parameter to include a maximum number of instances for learning the model or to potentially exclude outliers recognized by previous steps. Extensive experiments on three real world datasets demonstrate the effective of proposed method compared to state-of-arts baselines, and the effects of self-paced learning for recommendation are revealed.

Currently, we only utilize the generalized neural matrix factorization and only consider the noisy instances in user–item interaction records. Other deep neural networks (such as graph neural networks, deep auto-encoder, deep metric learning) have also been shown powerful. Thus one direction of future work lies in treating the noise corruption issues in other data source (such as user or item features and users' social relationship) with more robust deep model.

CRediT authorship contribution statement

Zhen Liu: Supervision, Project administration , Writing - original draft, Funding acquisition. **Xiaodong Feng:** Conceptualization, Methodology, Investigation, Validation, Writing - review & editing. **Yecheng Wang:** Software, Data curation, Visualization. **Wenbo Zuo:** Software, Data curation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work was partially supported by grants from the National Natural Science Foundation of China (No. 72004021).

Appendix

To further evaluate the bounded self-paced learning framework proposed in this study, we also report the experimental results on the dataset of Retailrocket with varying settings of the length of the recommendation list *k*. It is found in Fig. 8 that, compared to the baselines, NeuMF_SPL_B has the best performance. It again validates that the new learning framework can improve the recommendation accuracy of traditional models in diverse contexts.

References

[1] T. Ha, S. Lee, Item-network-based collaborative filtering: A personalized recommendation method based on a user's item network, Inf. Process. Manage. 53 (5) (2017) 1171–1184.
[2] F. Narducci, P. Basile, C. Musto, P. Lops, A. Caputo, M. de Gemmis, L. Iaquinta, G. Semeraro, Concept-based item representations for a cross-lingual content-based recommendation process, Inform. Sci. 374 (2016) 15–31.
[3] Y. Koren, R. Bell, C. Volinsky, Matrix factorization techniques for recommender systems, Computer 42 (8) (2009) 30–37.
[4] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, T.-S. Chua, Neural collaborative filtering, in: International Conference on World Wide Web, International World Wide Web Conferences Steering Committee, 2017, pp. 173–182.
[5] N. Hussain, H. Turab Mirza, G. Rasool, I. Hussain, M. Kaleem, Spam review detection techniques: A systematic literature review, Appl. Sci. 9 (5) (2019) 98701–98726.
[6] D. Meng, Q. Zhao, L. Jiang, A theoretical understanding of self-paced learning, Inform. Sci. 414 (2017) 319–328.
[7] H. Gao, H. Huang, Self-paced network embedding, in: ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, ACM, 2018, pp. 1406–1415.
[8] C. Xu, D. Tao, C. Xu, Multi-view self-paced learning for clustering., in: International Joint Conference on Artificial Intelligence, IJCAI, 2015, pp. 3974–3980.
[9] Q. Zhao, D. Meng, L. Jiang, Q. Xie, Z. Xu, A.G. Hauptmann, Self-paced learning for matrix factorization, in: AAAI Conference on Artificial Intelligence, 2015, pp. 3196–3202.
[10] M. Deshpande, G. Karypis, Item-based top-n recommendation algorithms, ACM Trans. Inf. Syst. 22 (1) (2004) 143–177.
[11] X. Ning, G. Karypis, Slim: Sparse linear methods for top-n recommender systems, in: IEEE International Conference on Data Mining, IEEE, 2011, pp. 497–506.
[12] X. Feng, A. Sharma, J. Srivastava, S. Wu, Z. Tang, Social network regularized sparse linear model for top-n recommendation, Eng. Appl. Artif. Intell. 51 (2016) 5–15.
[13] S. Kabbur, X. Ning, G. Karypis, Fism: factored item similarity models for top-n recommender systems, in: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2013, pp. 659–667.
[14] X. He, Z. He, J. Song, Z. Liu, Y.-G. Jiang, T.-S. Chua, Nais: Neural attentive item similarity model for recommendation, IEEE Trans. Knowl. Data Eng. 30 (12) (2018) 2354–2366.
[15] W. Yuan, H. Wang, X. Yu, N. Liu, Z. Li, Attention-based context-aware sequential recommendation model, Inform. Sci. 510 (2020) 122–134.
[16] M. Mao, J. Lu, G. Zhang, J. Zhang, Multirelational social recommendations via multigraph ranking, IEEE Trans. Syst. Man Cybern. 47 (12) (2017) 4049–4061.
[17] S. Chen, Y. Peng, Matrix factorization for recommendation with explicit and implicit feedback, Knowl.-Based Syst. 158 (2018) 109–117.
[18] Y. Koren, Factorization meets the neighborhood: a multifaceted collaborative filtering model, in: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2008, pp. 426–434.
[19] S. Zhang, L. Liu, Z. Chen, H. Zhong, Probabilistic matrix factorization with personalized differential privacy, Knowl.-Based Syst. 183 (2019) 104864.
[20] H. Parvin, P. Moradi, S. Esmaeili, N.N. Qader, A scalable and robust trust-based nonnegative matrix factorization recommender using the alternating direction method, Knowl.-Based Syst. 166 (2019) 92–107.
[21] H. Wang, N. Wang, D.-Y. Yeung, Collaborative deep learning for recommender systems, in: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2015, pp. 1235–1244.
[22] X. Feng, S. Wu, Z. Tang, Z. Li, Sparse latent model with dual graph regularization for collaborative filtering, Neurocomputing 284 (2018) 128–137.
[23] Z. Liu, K. Tan, X.-Q. Wang, S.-H. Tang, A learning framework for temporal recommendation without explicit iterative optimization, Appl. Soft Comput. 67 (2018) 529–539.
[24] Q. Zhang, J. Lu, D. Wu, G. Zhang, A cross-domain recommender system with kernel-induced knowledge transfer for overlapping entities, IEEE Trans. Neural Netw. 30 (7) (2019) 1998–2012.
[25] Y. Wu, C. DuBois, A.X. Zheng, M. Ester, Collaborative denoising auto-encoders for top-n recommender systems, in: ACM International Conference on Web Search and Data Mining, in: WSDM '16, ACM, 2016, pp. 153–162.
[26] X. Dong, L. Yu, Z. Wu, Y. Sun, L. Yuan, F. Zhang, A hybrid collaborative filtering model with deep structure for recommender systems, in: AAAI Conference on Artificial Intelligence, AAAI, 2017, pp. 1309–1315.
[27] Y. Ma, J. Xu, B. Stenger, C. Liu, Y. Hirate, Deep heterogeneous autoencoders for collaborative filtering, in: IEEE International Conference on Data Mining, 2018, pp. 1164–1169.
[28] C. Yang, L. Miao, B. Jiang, D. Li, D. Cao, Gated and attentive neural collaborative filtering for user generated list recommendation, Knowl.-Based Syst. 187 (2020) 104839.
[29] X. Wang, X. He, M. Wang, F. Feng, T.-S. Chua, Neural graph collaborative filtering, in: ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, 2019, pp. 165–174.
[30] R. Ying, R. He, K. Chen, P. Eksombatchai, W.L. Hamilton, J. Leskovec, Graph convolutional neural networks for web-scale recommender systems, in: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2018, pp. 974–983.
[31] J. Zhao, X. Geng, J. Zhou, Q. Sun, Y. Xiao, Z. Zhang, Z. Fu, Attribute mapping and autoencoder neural network based matrix factorization initialization for recommendation systems, Knowl.-Based Syst. 166 (2019) 132–139.
[32] Y. Bengio, J. Louradour, R. Collobert, J. Weston, Curriculum learning, in: International Conference on Machine Learning, IMLS, 2009, pp. 41–48.
[33] N. Sarafianos, T. Giannakopoulos, C. Nikou, I.A. Kakadiaris, Curriculum learning of visual attribute clusters for multi-task classification, Pattern Recognit. 80 (2018) 94–108.
[34] M.P. Kumar, B. Packer, D. Koller, Self-paced learning for latent variable models, in: Advances in Neural Information Processing Systems, 2010, pp. 1189–1197.
[35] X. Zhu, Z. Zhang, Improved self-paced learning framework for nonnegative matrix factorization, Pattern Recognit. Lett. 97 (2017) 1–7.
[36] N. Srivastava, R.R. Salakhutdinov, Multimodal learning with deep boltzmann machines, in: Advances in Neural Information Processing Systems, 2012, pp. 2222–2230.
[37] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2016, pp. 770–778.
[38] L. Jiang, D. Meng, Q. Zhao, S. Shan, A.G. Hauptmann, Self-paced curriculum learning, in: AAAI Conference on Artificial Intelligence, IEEE, 2015, pp. 2694–2700.
[39] E. Cho, S.A. Myers, J. Leskovec, Friendship and mobility: user movement in location-based social networks, in: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2011, pp. 1082–1090.
[40] P. Crewson, Applied Statistics Handbook, AcaStat Software, Leesburg, 2006.
[41] H. Ma, H. Yang, M.R. Lyu, I. King, Sorec: social recommendation using probabilistic matrix factorization, in: ACM Conference on Information and Knowledge Management, ACM, 2008, pp. 931–940.