A Unified Collaborative Representation Learning for Neural-Network Based Recommender Systems

Yuanbo Xu⍟, En Wang⍟, Yongjian Yang⍟, and Yi Chang⍟, *Senior Member, IEEE*

Abstract—With the boosting of neural networks, recommendation methods become significantly improved by their powerful ability of prediction and inference. Existing neural-network based recommender systems (NN-RSs) usually first employ matrix embedding (ME) as a pre-process to learn users' and items' representations (latent vectors), then input these representations to a specific modified neural network framework to make accurate Top-k recommendations. Obviously, the performance of ME has a significant effect on RS models. However, most NN-RSs focus on accuracy by building representations from the direct user-item interactions (e.g., user-item rating matrix), while ignoring the underlying relatedness between users and items (e.g., users who rate the same ratings for the same items should be embedded into similar representations), which is an ideological disadvantage. On the other hand, ME models directly employ inner products as a default loss function metric that cannot project users and items into a proper latent space, which is a methodological disadvantage. In this paper, we propose a supervised collaborative representation learning model - Magnetic Metric Learning (MML) - to map users and items into a unified latent vector space, enhancing the representation learning for NN-RSs. First, MML utilizes dual triplets to model not only the observed relationships between users and items, but also the underlying relationships between users as well as items to overcome the ideological disadvantage. Specifically, a modified metric-based dual loss function is proposed in MML to gather similar entities and disperse the dissimilar ones. With MML, we can easily compare all the relationships (user to user, item to item, user to item) according to the weighted metric, which overcomes the methodological disadvantage. We conduct extensive experiments on four real-world datasets with large item space. The results demonstrate that MML can learn a proper unified latent space for representations from the user-item matrix with high accuracy and effectiveness, and lead to a performance gain over the state-of-the-art RS models by an average of 17 percent.

Index Terms—Latent vectors, collaborative representation learning, metric learning, recommender systems

---

1 INTRODUCTION

IN recent years, popular online commercial websites such as Netflix, Amazon, Yelp, and Taobao provide a wide spectrum of recommendation services to help the customers filter their preferences out of enormous product space [1]. However, the performance of traditional recommendation models, such as collaborative filtering (CF) [2], matrix factorization (MF) [3] is highly restricted by the large scale of product space. With the development of neural networks and computation theory, the technology of recommender systems has been taken to the next stage [4]. To tackle large scale products for recommendations, most neural-network-based recommender systems first extract latent vectors of users and items from a user-item matrix. This extraction procedure is called matrix embedding (ME) [5], which is a critical factor in getting accurate recommendations, especially

for learning meaningful, measurable latent vectors. With these latent vectors, some traditional recommendation models are enhanced for real-world applications, such as CF to NCF [6], MF to NeuMF [7]. Some novel NN-based recommendation models are also proposed, such as GERL [8], NeuO [9] and HERec [10].

However, most researches assume that these latent vectors learned by existing ME models are insufficient and biased [4], without taking the interpretability into consideration [4], [11]. In other words, traditional ME only utilizes the relationships between users and items, while ignoring that the collaborative relationships between users and users, items and items, which is an *ideological disadvantage*. Moreover, most existing works directly utilize inner products to measure the relationships between users and items. This simple metric may cause chaos when computing similarities, which is a *methodological disadvantage*.

To make the above two disadvantages clear, we give a recommendation scenario in Fig. 1, where we employ basic matrix factorization as ME model and user-based collaborative filtering (UBCF [12]) as recommendation model. In this example, $U(u_1, u_2, u_3)$ and $I(i_1, i_2, i_3)$ represent users and items, respectively. R is a user-item rating matrix with ratings r_{ij}. P and Q are built with 3-dimensional embedding results (p_i for user i, q_j for user j), extracted from matrix R with existing ME models, such as SVD or other matrix factorization methods. To pick a proper item recommended to u_2, we employ a popular recommendation model (user-based CF,

- *Yuanbo Xu, En Wang and Yongjian Yang are with the Department of Computer Science and Technology, Jilin University, Changchun 130012, China, and also with the Key Laboratory of Symbolic Computation and Knowledge Engineering for the Ministry of Education, Jilin University, Changchun 130012, China. E-mail: {yuanbox, wangen, yyj}@jlu.edu.cn.*
- *Yi Chang is with the School of Artificial Intelligence, Jilin University, Changchun, Jilin 130012, China. E-mail: yichang@jlu.edu.cn.*
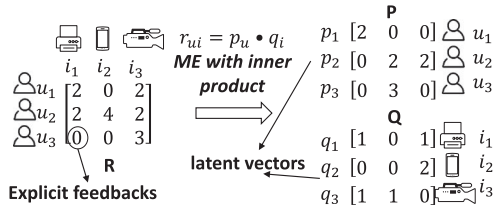
Fig. 1. An example to illustrate the disadvantages of traditional ME models.

Fig. 2. Comparison between traditional metric learning and our proposed model MML. Traditional ML (upper part) only focuses on user-item relationships (rectangle to triangle), while MML (bottom part) also takes the underlying user-user (rectangle to rectangle), item-item (triangle to triangle) relationships into consideration.
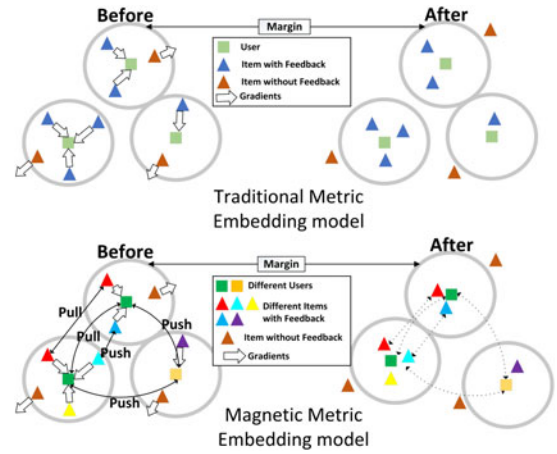
UBCF) with users' latent vectors. User-based CF calculates the similarities among u_1, u_2, u_3 to pick the Top-1 user neighbor for u_2. Then it recommends items that this Top-1 user has consumed to u_2.

As a result in Fig. 1, in ideology (note that UBCF model uses inner products to calculate latent vectors while the similarity between latent vectors is measured by euclidean distance [11]). Intuitively, UBCF should pick u_3 as u_2's Top-1 neighbor because $(p_1, p_2)_E = \sqrt{12} > (p_3, p_2)_E = \sqrt{5}$. But in fact, when considering the underlying relationship (between u_1 and u_2) hidden in matrix R, it is obvious that u_1 should be a better choice (u_1 and u_2 share the same preference of i_1 and i_3 according to their explicit feedbacks r_{11}, r_{13}, r_{21} and r_{23}). Hence, choosing u_3 as u_2's neighbor is an inaccurate decision caused by the *ideological disadvantage*, which is partly mentioned in [13], [14].

In methodology, directly choosing inner products as the metric may cause a dilemma, especially for CF models. In general, CF models employ euclidean distance between latent vectors as the similarity to find the nearest neighbor [4], where the latent vectors are learned by using inner products in traditional ME models [15]. To ensure the metric-satisfying non-negativity in latent space, the latent vector calculation should obey the triangle inequality (the sum length of any two sides must be greater than or equal to the remaining side, and the reason why embeddings should obey this is detailed introduced in [6], [11]). However, the relationships measured by inner products may violate the triangle inequality. For example, as shown in Fig. 1, if ME models use inner products • to learn latent vectors for i_1, u_2 and u_3 as q_1, p_2 and p_3, then, $p_2 • q_1 + p_3 • q_1 < p_2 • p_3$, which violates the triangle inequality. If we conduct calculations in a latent space with a metric that violates the triangle inequality, it may lead to uncertainty and inaccuracy of computing, and finally, result in a biased recommendation. Therefore, only applying inner products in ME models is not a suitable choice when learning latent vectors for recommendations. This *methodological disadvantage* damages the performance of recommendation models tremendously. For recommender systems, it's still a challenge to learn a proper latent space, where all kinds of relationships (users/items/user-item) can be measured by a unified style of the metric.

To relieve the limitation of inner products, metric learning has been proved to be useful in the multimedia area [13], [16], [17]. The core of metric learning is to learn a proper metric for the measurement between latent vectors. However, metric learning is only designed to measure user-item relationships in recommender systems [13], [14], which cannot simultaneously tackle the *ideological* and *methodological* disadvantage (as shown in Fig. 2). To this end, we propose a supervised collaborative representation learning model for

matrix embedding: Magnetic Metric Learning (MML), which utilizes the dual triplets to represent the different types of relationships (user-user, item-item, user-item) with a uniform latent space in a uniform framework. MML can learn not only the explicit relationships but also the latent relationships, which overcomes the ideological disadvantages. Meanwhile, the relationships between users and items are directly measured by weighted metric distance, which overcomes the methodological disadvantage.

The contributions of this paper are summarized as follows:

- We first argue that existing matrix embedding methods for neural-network-based recommendation models are not sufficient and unbiased. Then we explore the ideological and methodological disadvantages of traditional ME models and propose a representation learning model for matrix embedding: Magnetic Metric Learning, to overcome the above disadvantages.
- For the ideological disadvantage, we utilize dual triplets to model explicit and latent collaborative relationships among users and items in a uniform latent space. For the methodological disadvantage, a modified metric-based dual loss function is proposed to learn weighted metric and latent vectors at the same time.
- The experimental results on four real-world datasets demonstrate that MML can learn a proper unified latent space from the user-item matrix, and improve the accuracy of the state-of-the-art models.

The paper is organized as follows. We provide preliminaries in Section 2. Then we elaborate on the proposed method MML, including theory, regularization, and training process in Section 3. We report the experimental results in Section 4. Lastly, we review related work in Section 5 and conclude this paper in Section 6.

2 PRELIMINARIES

2.1 Basic Definitions

In recommender systems, U denotes a set of m users $U = \{u_1, u_2 \ldots u_m\}$, and I denotes a set of n items $I = \{i_1, i_2 \ldots i_n\}$.

A user-item rating matrix, whose entries are r_{ui}, is built as R. For items with ratings, we set r_{ui} as the rating, while for items without ratings, $r_{ui} = 0$. If $r_{ui} = 0$, we treat (u, i) as a negative pair, otherwise a positive pair.

Definition 1: Matrix Embedding. Given a matrix $R \in \mathbb{R}^{m \times n}$, the matrix embedding (ME) model is to depose the matrix into two low-dimension k spaces, which are also called latent vector spaces $P \in \mathbb{R}^{m \times k}$, $Q \in \mathbb{R}^{n \times k}$. Especially, in recommender systems, R is the user-item rating matrix. $p_i \in P$ is the latent vector for user i, while $q_j \in Q$ is for item j. Note that in real-world scenarios, the scale of users and items is huge, which means that $m, n \gg k$. Existing ME models usually utilize some matrix factorization methods, such as pureSVD [17] and NMF [17] to learn latent vectors. However, these models directly employ inner products in their loss function, which may lead to inaccurate and biased embedding results. Our proposed model aims to solve this problem, which is demonstrated in detail in Section 3.

Definition 2: Metric Learning. Given two different latent vectors $p, q \in \mathbb{R}^{1 \times k}$, the metric learning (ML) model is to learn a proper weighted metric matrix $W \in \mathbb{R}^{k \times k}$ to measure the relationships between p and q [18]. The different weights in W stand for the importance of each element in latent vectors. Specifically, in recommender systems, the distance between p_i and q_j can be treated as the measurement between user i and item j, as well as the user i's preference for item j. Existing metric learning models usually focus on the explicit feedbacks and models' optimizations in recommender systems, such as CML [13] and IML [19]. However, these models usually ignore the underlying relationships hidden in the user-item matrix R, which is a restriction to the ML models' performance.

Definition 3: Neural-Network-Based Recommendation Models. A typical neural-network-based recommendation model is a two-stage framework: the basic input is the user-item rating matrix R, and some other side information, including text, videos, and images. The first stage is named representation learning. In this stage, the inputs are mapped into latent vectors, including user latent vectors P, item latent vectors Q, and side information latent vectors SI, which extracts the latent features hidden in the multi-modal information. In the second stage for the recommendation, the latent vectors are feed into a modified neural network, which outputs the predicted ratings $\hat{r}_{ui}$. Finally, according to the ranking of $\hat{r}_{ui}$, the model gives a Top-k recommendation list. A general framework is shown in Fig. 3. Some popular recommendation models are based on this framework with different embedding models and neural networks, including Neural CF [6], NeuO [9]. However, most models use inner products as default, where we argue it is not always stable.

Note that some existing models integrate and implement joint learning framework. However, we argue there are some disadvantages: 1) overfitting. If we co-train the two stages, we have only one loss function on the recommedation stage, which may lead to the potential overfitting problem in representation learning stage [20]. 2) flexibility. The users' and items' latent representations learned by first stage could be combined with different recommendation model, or other models (such as user profiling, slanderous user detection), which is flexible for different application scenarios.
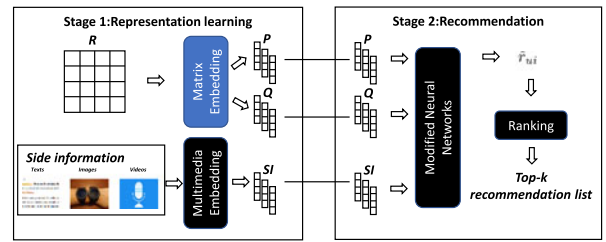
A Typical Two Stage NN-based Recommendation model

Fig. 3. A typical two-stage neural network based recommendation model. Note that in this paper we focus on the matrix embedding part of this framework.

2.2 Matrix Embedding With Inner Products

Given a user-item matrix R, matrix embedding models with inner products usually minimize this loss function L_{IP} to learn latent vectors P and Q:

$$L_{\text{IP}} = \sum_{u \in U, i \in I}^{P,Q} (r_{ui} - p_u \bullet q_i)^2 + pen(P, Q), \quad (1)$$

where $pen(P, Q)$ is a penalty term to avoid overfitting. Then we can use these latent vectors to make recommendations: 1) for user-based collaborative filtering [21], we need to find the nearest k-neighbor for target user t with the following function:

$$N_k \left(\min_{u \in U} |p_u, p_t|_{Euc} \right). \quad (2)$$

Then some common items in these neighbors can be recommed to the target user. 2) for neural network based models, we input the latent vectors and make recommedations as shown in Fig. 3.

2.3 Matrix Embedding With Metric Learning

Given a user-item matrix R, matrix embedding models with metric learning usually minimize this loss function L_{ML}:

$$L_{\text{ML}} = \sum_{\substack{u \in U, i, j \in I \\ r_{ui} \neq 0}}^{P,Q} \left(L_{\text{pull}}(p_u, q_i) - L_{\text{push}}(p_u, q_j) \right) + pen(P, Q). \quad (3)$$

Note that there are two loss functions in traditional metric learning: L_{pull} and L_{push}. The core idea for metric learning is to gather the user-item pair with explicit feedbacks and disperse the pair without them. So L_{pull} is employed to calculate the weighted distance between user u and item i, where $r_{ui} \neq 0$. By minimizing L_{pull}, L_{ML} tries to pull the user and item together. Meanwhile, by minimizing $-L_{\text{push}}$, L_{ML} tries to push away the user and item where $r_{ui} = 0$. Specifically, in the training process of metric learning, the model can learn not only the latent vectors P, Q, but also the weighted metric matrix W. The important notations are shown in Table 1.

3 MAGNETIC METRIC LEARNING (MML) MODEL

Magnetic Metric Learning model (MML) employs a unified style of metric learned through embedding and recommendation, and learns a unified latent space for users and items, which overcomes the methodological disadvantage.

TABLE 1
Notation List

Notation	Description
U	user set with u in recommender systems
I	item set with i in recommender systems
R	rating matrix $R \in \mathbb{R}^{m \times n}$ with rating r_{ui}
$\mathbb{R}, \mathbb{E}$	notations for latent spaces
m, n	number of users/items
r_{ui}	u's rating on item i
P	k-dimension user latent vector set $P \in \mathbb{R}^{m \times k}$
Q	k-dimension item latent vector set $Q \in \mathbb{R}^{n \times k}$
p_u, q_i	k-dimension latent vectors for u and i
e	uniformed latent vector (a user or an item)
W	metric matrix $W \in \mathbb{R}^{k \times k}$ with w
W^U, W^I, W^{UI}	W for users, items and user-item
$|e_a, e_b|_{\text{Euc}}$	euclidean distance between e_a, e_b
$|e_a, e_b|_{\text{W}}$	metric distance between e_a, e_b with W
$L_{\text{MML}}^{\text{EX}}$	explicit relationship loss function
S^U, S^I	user/item similar-pair buffer sets
mr	learning margin for metric learning
$L_{\text{MML}}^{\text{LA}}$	latent relationship loss function
$\alpha, \lambda, \theta, \omega$	hyper parameters

Meanwhile, MML considers both explicit and latent relationships and makes a direct embedding to overcome the ideological disadvantage (shown in Fig. 2).

Specifically, MML treats users and items as the same entities in a unified latent space, where all the relationships between users and items can be represented by their distance (in MML, it is measured by learned metric W). Moreover, MML can learn users' and items' latent vectors in a uniform framework with a uniform metric across all the procedures (embedding and recommendation) and overcome the limitation of inner products. MML does not need to distinguish latent user space and latent item space. All the users and items are embedded into the same dimension latent space. In this way, we could more easily optimize MML's loss function and calculate its gradient compared with other NN-based embedding models, such as autoencoder.

3.1 Learning Metric: Foundation of MML

We define a k-dimensional uniformed latent space $\mathbb{E}$, where $e_i \in \mathbb{E}$ stands for an extracted latent vector for a user or an item, i stands for an entity which can be either a user or an item. First, we define the function F for calculating the relationships between entities a, b as the following euclidean function:

$$F^E(a, b) = \|e_a - e_b\|_{\text{Euc}}^2. \tag{4}$$

While in MML, we use a learned metric $W \in \mathbb{R}^{k \times k}$ as a substitute for euclidean, as shown in Eq. (5):

$$F^*(a, b) = \|e_a - e_b\|_{W^*}^2. \tag{5}$$

Note that we consider learning different weighted matrix W^* for measuring user-user (W^U), item-item (W^I) and user-item (W^{UI}) relationships, which is an improvement over

other metric learning models, such as CML [13], IML [19] and CRML [22]. With these learned metrics, all the relationships can be measure as follows:

$$\|e_a - e_b\|_{W^*}^2 = \sqrt{(e_a - e_b)^T W^*(e_a - e_b)}. \tag{6}$$

To ensure that the W^* we learned is a metric-satisfying non-negative metric and obeys the triangle inequality, we need to require W^* to be positive semi-definite. Note that setting $W^* = I$ gives euclidean distance. And if we set W^* to be diagonal, it corresponds to learning a metric in which different axes are given different weights upon euclidean distance. Generally, W^* parameterizes a family of Mahalanobis distance over $\mathbb{R}^{k \times k}$ [22], [23]. With different restrictions to W^*, we can tune our proposed model MML for different application scenarios.

3.2 Explicit Relationships Formulation

MML is designed to gather similar entities and disperse the dissimilar ones with learned metrics. In recommender systems, we treat the feedback $r_{ui} \in R$ as the indicator of explicit relationships. If $r_{ui} \neq 0$, we define that there is an explicit relationship between user u and item i. To consider this for enhancing matrix embedding process, we sample the dual triplets $< a, b, c >$ and $< c, d, a >$, where $a, b \in U$, $c, d \in I$, and $r_{ac} \neq 0, r_{bc} = 0, r_{ad} = 0$. To ensure the structural consistency, we can learn that a, c should be embedded closer than b, c and a, d. Meanwhile, according to a, b's different preferences on c, it is obvious that they should not be embedded closely. The same deduction is applied on (b, c), (c, d) and (a, d). In this way, MML maximizes the effect of metric learning with a modified enhanced metric-based dual loss function (EMDL):

$$L_{\text{MML}}^1 = \sum_{a, b \in U; c \in I} t_{a,b,c} |mr_1 + F^{UI}(e_a, e_c) - F^U(e_a, e_b) - F^{UI}(e_b, e_c)|_+, \tag{7}$$

$$L_{\text{MML}}^2 = \sum_{a \in U; c, d \in I} t_{a,c,d} |mr_2 + F^{UI}(e_a, e_c) - F^I(e_c, e_d) - F^{UI}(e_a, e_d)|_+, \tag{8}$$

where notation $|J|_+$ satisfies that: $|J|_+ = max(J; 0)$. t is a ranking weight calculated as suggested in [13]. And $mr_1, mr_2 > 0$ is the safety margin size. With this dual loss function, (a, c) is embedding closer than (a, d), (b, c) with metric W^{UI}. (a, b) and (c, d) are embedded far with metric W^U and W^I. Finally, we get the EMDL loss function of MML for explicit relationships:

$$L_{\text{MML}}^{\text{EX}} = \lambda L_{\text{MML}}^1 + (1 - \lambda) L_{\text{MML}}^2, \tag{9}$$

where λ is the balance weight between users and items. By minimizing EMDL, we can not only pull the user-item pair together with explicit relationships ($r_{ac} \neq 0$) and push away the user-item pairs with no feedback ($r_{bc}, r_{ad} = 0$), but also push away the user-user pair (a, b) and item-item pair (c, d), as shown in the lower part in Fig. 2.

3.3 Latent Relationships Formulation

Different from explicit relationships between users and items which are indicated by $r_{ui} \in R$, latent relationships always occur between users and users, items and items, which can not be directly observed. So many existing matrix embedding models only consider explicit relationships while ignoring the latent ones. However, the latent relationships should be an important factor in matrix embedding because they also reflect the users' preferences and items' features, as the example we have given in Introduction. In order to utilize the latent relationships, we first extract the user pairs and item pairs according to the following rules:

- Users who rate the same items should be embedded closer in latent vector space, and vice versa.
- Items rated by the same users should be embedded closer in latent vector space, and vice versa.

With the rules above, we first build two similar-pair buffer sets: S^U and S^I, which contain user pairs and item pairs, respectively. We treat user pair in S^U as the same category, so do the item pair in S^I. A user-user or item-item pair (a, b) is assigned to similar-pair buffer sets according to the following restriction:

$$\begin{cases} (a,b) \in S^{(*)}, \text{if } \frac{|\text{list}(a) \cap \text{list}(b)|}{|\text{list}(a) \cup \text{list}(b)|} > \theta; \\ (a,b) \notin S^{(*)}, \text{else}, \end{cases} \tag{10}$$

where $\text{list}(a)$ means the list of items that user a has rated, or the users who have rated item a, and $S^{(*)}$ is either S^U or S^I. θ is a control threshold to decide the partition of same preference that the users or the items share. So the loss function of latent relationships is as follows:

$$L_{\text{MML}}^{\text{LA}} = \sum_{a,f \in S^{(*)}} \sum_{a,g \notin S^{(*)}} t_{a,f,g} |mr_3 + F(e_a, e_f) - F(e_a, e_g)|_+. \tag{11}$$

In Eq. (11), f is the similar entity of a, while g is not. F could be either F^U or F^I in one formulation. mr_3 is the safety margin size. With this formulation, the user pair or item pair (a, f) in S are embedded closer than (a, g) not in S. The matrix embedding is more enhanced by considering the latent relationships for both users and items.

3.4 Magnetic Metric Learning Formulation

Finally, we combine explicit relationship loss $L_{\text{MML}}^{\text{EX}}$ and latent relationship loss $L_{\text{MML}}^{\text{LA}}$ linearly with a combination weight α:

$$L_{\text{MML}} = \alpha L_{\text{MML}}^{\text{EX}} + (1-\alpha) L_{\text{MML}}^{\text{LA}}. \tag{12}$$

Note that in $L_{\text{MML}}^{\text{EX}}$ and $L_{\text{MML}}^{\text{LA}}$, all the $+F$ functions are the realizations of L_{pull} in Eq.(2), which means pulling the similar entities together in learned metric space. While the $-F$ functions mean L_{push}, which pushes the dissimilar entities away.

3.5 Regularization and Optimization

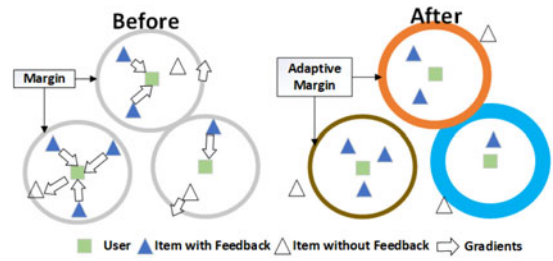We add two regularizations to make MML efficient and feasible.

Fig. 4. Effect of applying adaptive margins in MML.

To avoid overfitting and biased parameters, we bound all the embedding results $e^{(*)}$ (users' and items' latent vectors) in a unit sphere: $\|e^{(*)}\|^2 < 1$, to ensure the robustness of our model.

Moreover, we utilize a covariance regularization proposed by [24] to restrict the embedding results. First, we calculate a $k \times k$ matrix E for an O size of k-dimension vector e:

$$E_{ij} = \frac{1}{O} \sum_o (e_i^o - \eta_i)(e_j^o - \eta_j), \tag{13}$$

where o denotes the index in O, i, j is an index pair in a range of k. $\eta_i = \frac{1}{O} \sum_o e_i^o$. Then we define penalty loss L_P:

$$L_{\text{P}} = \frac{1}{O} (\|E\|_{\text{f}} - \|\text{diag}(E)\|_2^2); \\ \text{Subject to } \|e^{(*)}\|^2 < 1, \tag{14}$$

where $\|E\|_{\text{f}}$ is F-norm of E, $\text{diag}(E)$ is a diagonal matrix.

Moreover, to optimize the model, we first define the user-user weighted metric matrix W^U and item-item weighted metric matrix W^I to be symmetric because the relationships among users or items are undirected. With this restriction, we can save up the running time when calculating gradients.

To add personality into our model, we employ adaptive margins in MML (Fig. 4). Specially, there are three different margins in our model: mr_1, mr_2 and mr_3. Inspired by [14], we prefer to use adaptive margin to reduce the variations, which utilizes mr^u, mr^i and mr^l to replace the origin margin mr_1, mr_2 and mr_3, respectively for different categories of relationships. Note that the less items the users have rated, the larger margin should be applied to avoid overfitting. Thus, the adaptive margins could be achieved by minimizing the following loss function L_R:

$$L_{\text{R}} = -\left(\frac{1}{m} \sum_u mr^u + \frac{1}{n} \sum_u mr^i + \frac{1}{m+n} \sum_u mr^l \right); \\ \text{Subject to } mr^u \in (0,1], mr^i \in (0,1], mr^l \in (0,1], \tag{15}$$

where m, n are the size of U, I.

3.6 Training Process

In summary, our complete loss function of MML is shown as follows:

$$\text{Minimize}_{e^{(*)}}(L_{\text{MML}} + \omega_{\text{P}} L_{\text{P}} + \omega_{\text{R}} L_{\text{R}});$$

$$\text{Subject to } ||e^{(*)}||^2 < 1,$$
$$mr^u \in (0, 1],$$
$$mr^i \in (0, 1], \quad (16)$$
$$mr^l \in (0, 1],$$

where ω_{P}, ω_{R} are the hyperparameters for controlling L_{P} and L_{R}.

We minimize this constrained objective function above with Mini-Batch Stochastic Gradient Descent (SGD) and control the learning rating using AdaGrad. We keep the negative pair that maximizes the distance with the target user-item pair ($\max F(e_{\text{tar}}^{(*)}, e_{\text{ne}}^{(*)})$) when we sample negative pairs. Our training process is shown as Algorithm 1.

Algorithm 1. MML Training process

Input: User set U; item set I; user-item rating matrix R; margins mr^u, mr^i, and mr^l; hyperparameters α, λ, θ, and ω
Output: User / Item latent vector set E_U / E_I, metric matrix W^U, W^I, W^{UI}.
1: Select a batch B with N positive user-item pairs.
2: **for all** $B \in U, I$ **do**
3:    **for all** user-item positive pair (a, c) **do**
4:       Sample 2 negative user-item (a, d), (b, c) pairs to build two triplets.
5:       Calculate $L_{\text{MML}}^{\text{EX}}$ with Eq. (9).
6:       For a, sample 1 similar user f and 1 dissimilar user g with Eq. (10). Also sample a similar item and a dissimilar item for c.
7:       Calculate $L_{\text{MML}}^{\text{LA}}$ with Eq. (11).
8:       Calculate L_{MMA} across batch B.
9:    **while** not converge **do**
10:      Calculate gradients.
11:      Update p_u and q_i with AdaGrad on Eq. (16).
12:      Update W^U, W^I and W^{UI} with AdaGrad on Eq. (16).
13:      Update mr^u, mr^i and mr^l with AdaGrad on Eq.(16).
14: **return** User / Item latent vector set E_U / E_I; metric matrix W^U, W^I, W^{UI}.

3.7 Comparison With Collaborative Metric Learning

We compare our proposed model with a representative model, Collaborative Metric Learning (CML) [13] in detail. MML borrows the idea of metric learning, which is similar to CML. However, our model has essential differences compared with CML (shown in Fig. 5):

First, CML utilizes only the user-item pair to build the objective function, which focuses on the explicit relationships in the user-item matrix. As shown in Fig. 5, CML pulls the items i_1, i_2 to the user u and pushes away item i_3. However, note that there are latent relationships hidden in the user-item matrix. So the items in the same similar-pair set (i_2, i_3) should be embedded closer, while i_1, i_2 should be embedded with a longer distance. MML considers this situation, using Push and Pull for both explicit and latent relationships, to achieve more accurate and unbiased embedding results.

Second, as shown in the right part of Fig. 5, MML can learn a direct and visible embedding result because of the latent relationship formulation $L_{\text{MML}}^{\text{LA}}$. In $L_{\text{MML}}^{\text{LA}}$, MML considers the relationships between same categories (users or
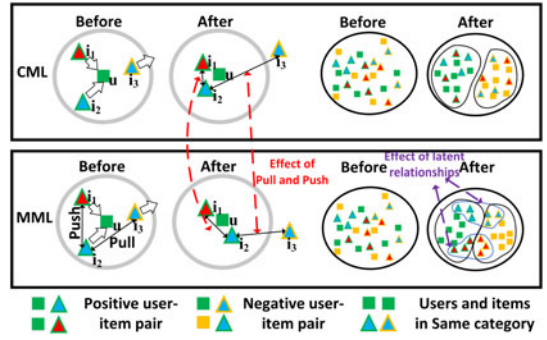

Fig. 5. Comparison between CML and MML. The red dashed line shows the effect of Pull and Push. The purple dashed line shows the effect of consideration of latent relationships.

items). So MML is able to gather the entities of the same category closer than CML, which is also a great improvement on explainability.

Moreover, CML directly employs euclidean distance to measure the relationships between different users and items, ignoring the importance variety of different vectors. While MML is able to learn a more accurate metric W for users, items, and user-item respectively, which fine-grained measures the relationships. CML utilizes the fixed margin for all entities, while MML considers the different criteria for different users and items, and employs the adaptive margins to add personality to our model.

Finally, the objective function of MML with a uniform format ($L_{\text{MML}}^{\text{EX}}, L_{\text{MML}}^{\text{LA}}$) does not distinguish users and items like CML, which is more feasible and effective. MML employs two regularizations to relieve overfitting situations, especially when the dataset is sparse and unbalanced.

4 EVALUATION

In this section, we first describe the experimental settings, including datasets, baselines, parameter setting, and implementation details. Subsequently, we conduct extensive experiments to answer the following research questions:

RQ1: How is the effectiveness of MML? Can it provide a competitive performance compared with baselines on the matrix embedding task at a proper running time? *RQ2*: How do the hyperparameters affect the performance of MML? Which are the optimal values? *RQ3*: How does the proposed model benefit the neural-network-based recommendation models with Top-K recommendation? *RQ4*: How do the learned metric benefit the matrix embedding and recommendations? What is the effectiveness of regularization to avoid overfitting? *RQ5*: What is the embedding performance of MML on million-scale dataset? What is the comparison between MML and other SOTA ME models?

4.1 Experimental Settings

4.1.1 Datasets

We conduct experiments on datasets from Amazon.com[1] (we use Amazon as the abbreviation of the Sports and Outdoors dataset in Amazon in this paper) and Yelp for RecSys.[2]

---

1. https://jmcauley.ucsd.edu/data/amazon
2. https://www.kaggle.com/c/yelp-recsys-2013

TABLE 2
The Datasets' Characteristics

Dataset	Amazon	Yelp	Taobao	Jingdong
#user	30,759	45,980	10,121	8,031
#item	16,515	11,537	9,892	3,025
#rating	285,644	229,900	49,053	25,152
#item labels	36	24	17	12
Sparsity	0.051%	0.043%	0.049%	0.12%

Moreover, we collect two datasets from Taobao[3] and Jingdong[4] as supplementary to validate our method [9], [25]. All the datasets contain ratings ranging from 1 to 5. We divide the datasets: 60 percent as the training set, 20 percent as the test set and 20 percent as the validation set with 5-cross-validation, and treat ratings less than 3 as negative samples for recommendations [9], [18]. Table 2 summarizes the details of datasets.

4.1.2 Baselines

To evaluate our proposed model on matrix embedding task, we compare MML with five representative metric learning models, including:

WRMF [26], [27] This implicit MF model utilizes an additional case weight to model unobserved interactions. WRMF can also be treated as a basic matrix factorization embedding model on the user-item matrix with inner products. *CML* [13] This representative CF model borrows the idea of metric learning to learn a latent space for users and items. Moreover, it is claimed that CML can outperform most state-of-the-art CF models with the metric-based loss function. *IML* [19] This efficient model applies metric learning to unbalanced data for clustering. IML's contribution is that it splits data into subsets and accelerates the process. *CRML* [22] This is a metric learning model for collaborative recommendations with co-occurrence embedding regularization. It considers the optimization problem as a multi-task learning problem which includes optimizing a primary task of metric learning and two auxiliary tasks of representation learning. *SML* [14] This is a metric learning model that symmetrically introduces a positive item-centric metric which maintains closer distance from positive items to users and pushes the negative items away from the positive items at the same time with an adaptive margin. We show the relationship measurement and loss function comparison with baselines in Table 3.

We combine MML with nine different recommendation models to make a top-k recommendation, including two basic recommendation models, and four neural-network-based recommendation models :

UBCF and *IBCF* [12] compute the similarity (Cosine or euclidean) between users (UBCF) or items (IBCF), and find the target's k-nearest neighbors to make Top-K recommendations. *NCF* [6] is a state-of-the-art neural-network-based recommendation model which directly combines the latent vectors as the input of the model. As his work claims, NCF can cover some state-of-the-art CF models. *2IPS* [20] is a typical two-stage off-policy policy gradient method. The

proposed method explicitly takes into account the ranking model when training the candidate generation model, which helps improve the performance of the whole system. *NAIS* [28] is an attention network, which is capable of distinguishing which historical items in a user profile are more important for a prediction. *KTUP* [29] jointly learns the model of recommendation and knowledge graph completion. It accounts for various preferences in translating a user to an item, and then jointly trains it with a KG completion model by combining several transfer schemes. *HERec* [10] is a heterogeneous network embedding based approach for heterogeneous information network (HIN) based recommendation. To embed HINs, it designs a meta-path based random walk strategy to generate meaningful node sequences for network embedding. *NGCF* [30] exploits the user-item graph structure by propagating embeddings on it. This leads to the expressive modeling of high-order connectivity in user-item graph, effectively injecting the collaborative signal into the embedding process in an explicit manner. *GraphRec* [31] provides a principled approach to jointly capture interactions and opinions in the user-item graph, which coherently models two graphs and heterogeneous strengths.

Some RS baselines are two-stage recommendation models which contain the matrix embedding parts. In this paper, we use ME baselines (WRMF/IML/CRML/SML/MML) to substitute these matrix embedding parts in RS models for testing.

4.1.3 Parameter Setting and Implementation Details

The implementation of the comparison methods are from the public codes that the authors provided in their papers or open source project. For MML, we set default margins $mr^u = mr^i = mr^l = 0.02$. All latent vectors in dimension $k = 32$, with random initialization (uniform distributions mean: 0.2, viariance: 0.04). The batch size B is 512. We tune the learning rate 0.01, 0.02, 0.05. Without special explanations, we set balance weight $\lambda = 0.5$, similarity threshold $\theta = 0.3$, $\omega = 0.03$ and $\alpha = 0.7$. All these parameters are determined through cross-validation.

4.2 Matrix Embedding Validation (RQ1)

In this section, we need to validate whether the models can gather the same items and disperse the different ones. Along with this line, we employ spherical k-means on embedding results, with $K = 10$ and 20 clusters. We use Normalized Mutual Information (NMI) as the protocols:

$$\mathrm{NMI}(L, C) = \frac{\mathrm{Cor}(L, C)}{[\mathrm{H}(L) + \mathrm{H}(C)]/2}, \qquad (17)$$

where L is the set of labels of items and C is the set of clusters. $\mathrm{Cor}(L, C)$ denotes the sum of mutual information between any label l in any cluster c. $\mathrm{H}(L)$ and $\mathrm{H}(C)$ denote the entropy for labels and clusters respectively. This metric evaluates the purity of clustering results from an information-theoretic perspective.

From the NMI evaluation results in Tables 4 and 5, we can see that MML outperforms all the baselines for all clustering value K in all four datasets. This result shows two advantages of MML: First, five models with metric learning are much better than traditional model WRMF, which

3. https://www.taobao.com
4. https://re.jd.com/

TABLE 3
Relationship Measurement and Loss Function Comparison With Baselines

Models	Relationship measurement	Loss Function
WRMF	$F(u,i) = p_u q_i^T$	$\sum_{u,i} (r_{ui} - p_u q_i^T)^2 + \lambda(\sum_u \|p_u\|^2) + \lambda(\sum_i \|q_i\|^2)$
CML	$F(u,i) = \|p_u - q_i\|_{\text{Euc}}^2$	$\sum_{u,i,i^-} \|F(u,i) - F(u,i^-) + m\|_+$
IML	$F(u,i) = \|p_u - q_i\|_{\text{Euc}}^2$	$\sum_{u,i,i^-} \|F(u,i) - F(u,i^-) + m\|_+$
CRML	$F(u,i) = \|p_u - q_i\|_{\text{Euc}}^2$	$\sum_{u,i,i^-} L(p_u, q_i) + L(p_u, b_{ui}) + L(q_i, b_{ui})$
SML	$F(u,i) = \|p_u - q_i\|_{\text{Euc}}^2$	$\sum_{u,i,i^-} \left(\|F(u,i) - F(u,i^-) + m_u\|_+ + \|F(u,i) - F(i,i^-) + m_i\|_+ \right) + \lambda L_{AM}$
MML	$F(u,i) = \|p_u - q_i\|_W^2$	$L_{\text{MML}} + \omega_P L_P + \omega_R L_R$ (Eq. (16))

λ is the hyperparameter, (u,i) means a positive pair $r_{ui} \neq 0$, (u,i^-) means a negative pair $r_{ui} = 0$, b_{ui} is a learned sharing parameter.

means that metric-based models are more proper for matrix embedding than inner products. Second, MML tackles both explicit and latent relationships and learns a weighed metric matrix, which leads to a more stable performance than CML, IML, CRML, and SML. Note that in Jingdong with 20 clusters, CRML, SML, and MML's performance are very close. But in Amazon and Yelp, MML outperforms both the state-of-the-art baselines. This indicates the advantage of MML in tackling sparse data.

Besides, we exploit the effect of latent vector space dimension k (4, 8, 16, 32, and 64) on NMI (Fig. 6). We notice that almost all ME models' performance is better with high-dimension latent vector space, and MML achieves the best results. High-dimensional data space has a strong representative ability to catch more hidden knowledge of users and items, which can enhance the performance of matrix embedding. Hence, the performance increases fast from 4 to 16. However, note that the increase becomes slower from 16 to 64, which shows the bottleneck of the dimension profit. Note that WRMF achieves the worst performance among baselines, which indicates that in the high dimension latent space, using metric learning is better than inner products in the matrix embedding task.

At last, we also compare the computing time among five metric learning models (Table 6). CML takes the shortest

time each epoch and IML takes the longest. CML's loss function is simple to calculate, so it achieves the best running time. While IML utilizes an iteration metric learning, which means in one epoch, IML learns metric repeatedly on different subsets. Note that two state-of-the-art models, CRML and SML use more time than our proposed model MML. SML combines two different styles of the loss function with two regularizations and three sub loss functions, which adds computation complexity. CRML and MML utilize the same formulation of loss function for explicit and latent relationship embedding, which is easy to compute derivation and speeds up the model's optimization.

4.3 Exploring Effect of Hyperparameters (RQ2)

In this section, we explore the effect of hyperparameters in MML. MML introduces four additional hyperparameters α, λ, θ, and ω. $\alpha \in (0,1)$ controls the learning of explicit and latent relationships. $\lambda \in (0,1)$ controls the learning of EMDL.

Fig. 6. Dimension effect on NMI with 10 clusters.

TABLE 4
Normalized Mutual Information With 10 Clusters

Model	Amazon	Yelp	Taobao	Jingdong
WRMF	0.3214	0.3013	0.4215	0.4317
CML	0.5310	0.5010	0.5870	0.5711
IML	0.5613	0.5522	0.5830	0.6001
CRML	0.5673	0.5444	0.6030	0.6111
SML	0.5723	0.5602	0.5933	0.6092
MML	**0.5831***	**0.5621***	**0.6321***	**0.6134***

TABLE 5
Normalized Mutual Information With 20 Clusters

Model	Amazon	Yelp	Taobao	Jingdong
WRMF	0.2943	0.3001	0.3255	0.3321
CML	0.4732	0.4638	0.5533	0.5612
IML	0.4831	0.4765	0.5545	0.5532
CRML	0.5023	0.5122	0.5732	0.6001
SML	0.5313	0.5232	0.5644	0.6011
MML	**0.5433***	**0.5564***	**0.6003***	**0.6112***

TABLE 6
Running Time for Training Process (Time Unit)

Time/Epoch	Amazon	Yelp	Taobao	Jingdong
CML	103*	124*	68*	74*
IML	349	402	156	147
CRML	112	133	79	89
SML	113	150	88	93
MML	**110**	**130**	**75**	**83**
Ours vs Best	**+7**	**+6**	**+7**	**+9**

Fig. 7. Performance of MML with respect to different values of α.
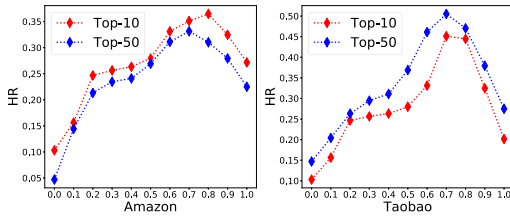

Fig. 9. Performance of MML with respect to different values of θ.

$\theta \in (0, 1)$ restricts the similar-pair set building in latent relationships formulation. ω controls the regularizations, which we discuss in the following sections (RQ4). Here we show how the three hyperparameters impact the performance and also shed light on how to set them. We only show the results on Amazon and Taobao due to the limitation of space. We use Hitting Ratio (HR) on Top-10 and Top-50 to explore the hyperparameters. We vary one parameter while fixing others as experimental settings.

As shown in Fig. 7, the optimal value of α is around 0.7 for both two datasets. And we also observe that the performance improves before α reaches 0.7, then it decreases sharply. Thus the too large value of α will ruin the learning process of metric learning. So we set α to 0.7.

As shown in Fig. 8, the optimal value of λ is around 0.5 for both two datasets. When $\lambda = 0.5$. it treats the users and items as the same category, which satisfies the assumption of our model (to map users and items into a unified latent space). So we set λ to 0.5.

As shown in Fig. 9, the optimal value of θ is around 0.3 for both two datasets. Note that When θ is too small, MML behaves minor improvements, which shows there are redundant pairs in similar-pair sets, which hurts the performance. Moreover, if θ is too large, the performance drops dramatically. So we set θ to 0.3.

4.4 Recommendation Validation (RQ3)

In this section, we validate the quality of embedding on recommendations. We treat six ME models as matrix embedding models, combining with nine popular recommendation models to make a Top-k recommendation. Hitting Ratio (HR) and Recall are employed to evaluate the recommendations. All the results, including our proposed model and baselines, achieve the best performance while keeping all hyperparameters at their optimal settings. The results on four datasets are shown in Table 7. Note that KTUP, HERec, and GraphRec are knowledge-graph based recommendation models, and the Amazon data does not provide the KGs.

In all datasets, our proposed model MML outperforms all the ME baselines with three recommendation models, which is a noticeable improvement. In detail, WRMF performs the worst, especially when it combines with NCF, 2IPS, KTUP,
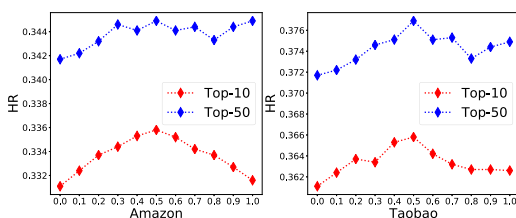

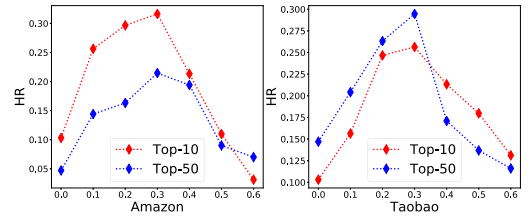Fig. 8. Performance of MML with respect to different values of λ.

and HERec. Note that WRMF is the only method that utilizes the inner products as the measurement for relationships. This result proves the effect of metric learning. When we compare UBCF and IBCF with different metric based models, it is interesting that CML's performance drops significantly, even worse than WRMF. The reason is that CML treats users as the center of embedding, which affects the items' embedding. Although IML also utilizes the idea of CML, the computation iteration of IML can make compensation to some extent. However, in our proposed model, we treat items and users as the same category to ensure accuracy. For NCF, because our models take more knowledge (the latent relationships) into consideration than CML and IML, it also improves an average of 20 percent over baselines.

Compared with two state-of-the-art models, CRML and SML, we notice that the improvement is more obvious on Amazon and Yelp than on Taobao and Jindong. Taking deep insight, MML utilizes different relationships, including explicit and latent ones. With these relationships, MML can relieve the data-sparse issue. While SML only considers user-item, item-time relationships. For CRML, it combines two style loss functions, which we argue it damage the embedding performance to some extent. For cooperating with GNN based models (GraphRec), MML could reach the best recommendation performance. At last, the most important factor is that MML learns a weighted metric matrix W, and uses W to calculate the distance in recommender systems, which is a significant improvement.

Moreover, we explore the perfromance enhancement for neural-network based recommendation models (basic MLP [32], NCF, 2IPS, KTUP, and HERec) using MML as a preprocessing for Top-10 and Top-50 recommendation. We conduct experiments on Yelp and Taobao. The results are shown in Fig. 10:

Note that with MML as a preprocessing for neural-network-based recommendation models, HR performance is enhanced over all baselines on both datasets. Specifically, MLP is the basic neural-network-based model that directly inputs latent vectors to predict ratings. The performance gain over MLP indicates the accuracy of latent vectors MML has learned. And for some of the state-of-the-art NN based models, MML can improve the HR performance by average 15 percent on Yelp, 17 percent on Taobao.

4.5 Exploring the Effect of MML's Component (RQ4)

In this section, we explore the effect of learned metric in our proposed model. We separate MML with each component, and rebuild the following models:

1) EUC-MML: Use $F^E(a, b) = \|e_a - e_b\|^2_{\text{Euc}}$ to replace F in MML (compare euclidean with Learned metric).

TABLE 7
Improvement of Recommendation Models With Different Matrix Embedding Models

RS model	ME model	Amazon		Yelp		Taobao		Jingdong	
		HR@50	Recall@50	HR@50	Recall@50	HR@50	Recall@50	HR@50	Recall@50
UBCF	WRMF	0.1372	0.2112	0.1544	0.2002	0.3721	0.3235	0.3313	0.3143
	CML	0.2112	0.2411	0.2339	0.2348	0.3826	0.4057	0.4118	0.4756*
	IML	0.2211	0.2333	0.2213	0.2453	0.4052	0.4361	0.4312	0.4123
	CRML	0.2312	0.2520	0.2439	0.2548	0.4336	0.4557	0.4780*	0.4661
	SML	0.2328*	0.2621*	0.2533*	0.2653*	0.4432*	0.4732*	0.4610	0.4711
	MML	**0.2618**	**0.2811**	**0.2794**	**0.2860**	**0.4653**	**0.4979**	**0.5167**	**0.4790**
	Ours vs Best	*+12.4%*	*+7.2%*	*+10.3%*	*+7.8%*	*+4.9%*	*+5.2%*	*+7.9%*	*+1.6%*
IBCF	WRMF	0.1433	0.2011	0.1411	0.2100	0.3543	0.3421	0.3442	0.3301
	CML	0.1634	0.1623	0.1777	0.2012	0.2972	0.3022	0.3310	0.3294
	IML	0.2011	0.2111	0.2214	0.2433	0.3911	0.3203	0.3882	0.4023
	CRML	0.1934	0.1831	0.1823	0.2213	0.3672	0.3723	0.3890	0.3684
	SML	0.2281*	0.2621*	0.2710*	0.2533*	0.4102*	0.4302*	0.4082*	0.4323*
	MML	**0.3133**	**0.3374**	**0.3332**	**0.3411**	**0.4833**	**0.5379**	**0.5100**	**0.4990**
	Ours vs Best	*+37.8%*	*+28.7%*	*+22.9%*	*+34.6%*	*+17.8%*	*+25.0%*	*+24.9%*	*+15.4%*
NCF	WRMF	0.2041	0.2210	0.1331	0.1994	0.3217	0.3433	0.3614	0.3710
	CML	0.2213	0.2561	0.2613	0.2600	0.4231	0.4313	0.4714	0.4705
	IML	0.2528	0.2722	0.2810*	0.2518	0.4303	0.4862	0.4660	0.4913
	CRML	0.2543	0.2771*	0.2653	0.2693*	0.4557*	0.4673	0.4884*	0.4745
	SML	0.2548*	0.2762	0.2810*	0.2688	0.4553	0.4879*	0.4767	0.4933*
	MML	**0.3318**	**0.3641**	**0.3700**	**0.3660**	**0.5053**	**0.5379**	**0.5288**	**0.5034**
	Ours vs Best	*+30.2%*	*+31.3%*	*+31.6%*	*+35.9%*	*+10.8%*	*+10.2%*	*+8.2%*	*+2.1%*
2IPS	WRMF	0.1137	0.1040	0.1041	0.1144	0.3091	0.2910	0.2906	0.2959
	CML	0.2220	0.3053	0.2958	0.2988	0.4151	0.4115	0.4251	0.4184
	IML	0.2234	0.3245	0.3110	0.2764	0.4312	0.4319	0.4555	0.4616
	CRML	0.2411	0.3400*	0.3253	0.3021	0.4617*	0.4714	0.4800	0.4645
	SML	0.2448*	0.3312	0.3311*	0.3452*	0.4613	0.4867*	0.4867*	0.4713*
	MML	**0.2918**	**0.3440**	**0.3706**	**0.3650**	**0.4813**	**0.5117**	**0.5012**	**0.5023**
	Ours vs Best	*+16.1%*	*+1.1%*	*+10.6%*	*+5.4%*	*+4.0%*	*+4.7%*	*+2.8%*	*+6.1%*
NAIS	WRMF	0.1184	0. 1194	0.1172	0.1193	0.2412	0.3329	0.3001	0.3200
	CML	0.2313	0.2910	0.3111	0.3200	0.4417	0.4564	0.4428	0.4511
	IML	0.2601	0.2813	0.3221	0.3226	0.4754	0.4719	0.4816	0.4776
	CRML	0.2799	0.2997*	0.3399	0.3411	0.4888	0.4814	0.4904	0.4883
	SML	0.2900*	0.2911	0.3466*	0.3551*	0.5012*	0.5003*	0.5019*	0.4933*
	MML	**0.3111**	**0.3532**	**0.3611**	**0.3588**	**0.5378**	**0.5400**	**0.5510**	**0.5410**
	Ours vs Best	*+7.2%*	*+17.8%*	*+5.6%*	*+1.0%*	*+7.3%*	*+7.9%*	*+9.7%*	*+9.6%*
NGCF	WRMF	0.1201	0.1209	0.1222	0.3102	0.3000	0.3222	0.3015	0.3132
	CML	0.2440	0.2411	0.2946	0.2945	0.4003	0.4013	0.4112	0.4113
	IML	0.2531	0.2664	0.2677	0.3011	0.5001	0.5023	0.4954	0.4333
	CRML	0.2679	0.2649	0.3216	0.3364	0.5013	0.4964	0.4755	0.5014
	SML	0.2974*	0.2874*	0.3454*	0.3461*	0.5105*	0.5009*	0.4969*	0.5110*
	MML	**0.3221**	**0.3600**	**0.3646**	**0.3654**	**0.5394**	**0.5475**	**0.5564**	**0.5433**
	Ours vs Best	*+8.3%*	*+25.2%*	*+5.5%*	*+5.5%*	*+5.6%*	*+9.3%*	*+11.9%*	*+6.3%*
KTUP	WRMF	-	-	0.1002	0.1083	0.2842	0.3178	0.2936	0.3132
	CML	-	-	0.2583	0.2584	0.4635	0.4753	0.4500	0.4347
	IML	-	-	0.2677	0.2711	0.4853	0.4879*	0.4700	0.4613
	CRML	-	-	0.2813	0.2693	0.4777	0.4773	0.4801*	0.4645*
	SML	-	-	0.2817*	0.2788*	0.4892*	0.4879*	0.4767	0.4633
	MML	-	-	**0.3411**	**0.3510**	**0.5211**	**0.5321**	**0.5388**	**0.4910**
	Ours vs Best	-	-	*+17.4%*	*+20.6%*	*+6.1%*	*+10.2%*	*+10.9%*	*+5.3%*
HERec	WRMF	-	-	0.1044	0.1027	0.2950	0.2915	0.2945	0.2888
	CML	-	-	0.2568	0.2526	0.4571	0.4364	0.4204	0.4429
	IML	-	-	0.2671	0.2505	0.4509	0.4284	0.4174	0.4468
	CRML	-	-	0.2700	0.2713*	0.4717*	0.4773*	0.4814*	0.4712
	SML	-	-	0.2813*	0.2698	0.4652	0.4679	0.4712	0.4813*
	MML	-	-	**0.3542**	**0.3711**	**0.5333**	**0.5279**	**0.5408**	**0.5112**
	Ours vs Best	-	-	*+20.5%*	*+26.8%*	*+11.5%*	*+9.5%*	*+10.9%*	*+5.8%*
GraphRec	WRMF	-	-	0.1112	0.1113	0.1942	0.2188	0.2711	0.3009
	CML	-	-	0.2333	0.2534	0.4112	0.4342	0.4432	0.4232
	IML	-	-	0.2577	0.2600	0.4723	0.4631	0.4564	0.4513
	CRML	-	-	0.3013	0.3023*	0.4917	0.5000	0.5101	0.5003
	SML	-	-	0.3117*	0.3000	0.5011*	0.5001*	0.5123*	0.5188*
	MML	-	-	**0.3655**	**0.3659**	**0.5400**	**0.5521**	**0.5601**	**0.5531**
	Ours vs Best	-	-	*+17.2%*	*+21.9%*	*+7.7%*	*+10.3%*	*+9.1%*	*+6.6%*

* marks the best performance among baselines.

2) W-MML: Use one W to replace W^U, W^I and W^{UI} in MML (compare fixed metric matrix with multi-metric matrix).

3) M-MML: Use one fix margin mr to replace mr^u, mr^i and mr^l (compare fixed margin with adaptive margin).

4) NP-MML: Use L_{MML} without restriction L_P (ω_P=0).

5) NR-MML: Use L_{MML} without restriction L_R (ω_R=0).

We conduct experiments on four datasets with NMI with 10 clusters and HR@50. The effect of different component in MML is shown in Table 8.

We notice that MML achieves the best performance (NMI and HR) over all four datasets. Specifically, EUC-MML performs worst than other models, which indicates that in our proposed model, euclidean is not the proper metric for matrix embedding tasks and recommendations. The simple euclidean metric may be not suitable for measuring the distance in high-dimension latent space. So metric learning for matrix embedding is necessary for complex NN-based recommendation models to tackle large scale sparse data. According to the comparison between W-MML and MML, it indicates that the metric between users, items, and user-item should be learned respectively to achieve a better result. It is obvious that the features of users and items are different, so learning different W is reasonable. The same explanation can be applied for the comparison between M-MML and MML which indicates the advantage of adaptive margins.

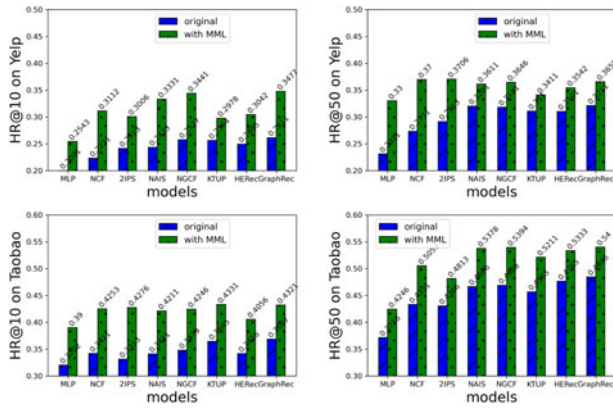For NP-MML and NR-MML, we can ensure the effect of regularization. Although the performance of MML is better

Fig. 10. Performance gain with MML for neural-network based recommendation models.

Fig. 11. Overfitting analysis on NMI and HR.

than NP-MML and NR-MML with a small gap, both regularization can enhance the model by avoiding overfitting.

To evaluate the effect for L_P, L_R to relieve the overfitting situation, we run NP-MML, NR-MML and MML on Amazon and Taobao to see the performance (NMI with 10 clusters and HR@50) changing with different epochs. The performance changing with epochs is shown in Fig. 11.

From the results, we can see that NR-MML achieves its best NMI performance within 20 epochs, while NP-MML and MML achieve their best within 30 epochs. Although the best performance of these three models is in the same level, we notice that NR-MML's HR decreases rapidly after 20 epochs, which is a significant overfitting phenomenon. L_R is to restrict the margins. When the epochs add up, a fixed margin can not measure the detailed distance between high-dimension latent vectors, which leads to the overfitting situation. While L_P is to restrict the latent vectors. Without L_P, our proposed model suffers the biased and de-centered embedding results. So NR-MML's performance can not be improved after 20 epochs by these biased latent vectors.

Compared with both models, MML can restrict the embedding results and tune the margins over epochs, with two regularizations L_R and L_P. Note that MML's performance is stable without sharp fluctuation, which also indicates robustness and effectiveness. Comprehensively, with the consideration of accuracy, efficiency, and overfitting, MML achieves a more stable and feasible performance than all these rebuilt models.
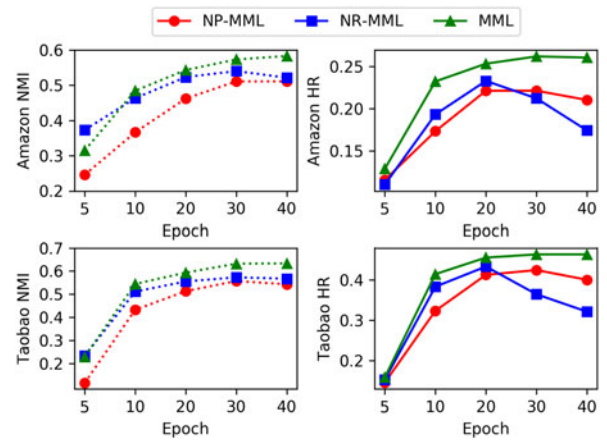
4.6 Million-Scale Embedding Validation (RQ5)

We conduct WRMF, ConvMF [33] (a SOTA model which combines GCN with MF for recommendations, widely employed as benchmarks), SML and our proposed MML on a million-scale dataset (Amazon Beauty, with 6,403,006 users, 1,660,119 items, 14,771,988 ratings, with 2.3070 ratings per user and 0.0001 percent sparsity) [34]. Note that this work focuses on the embedding procedure, we only utilize the models' embedding results for validations. Specifically, we use NMI with 20 clusters and running time unit as metrics, as shown in Table 9:

Note that ConMF performs better on NMI than MML (2.12 percent). The reason is that ConvMF enriches the dataset by convolution operations with CNN framework. However, limited by the scale of dataset, ConMF need more running time (almost 100 percent) than metric learning-based model (SML and MML) for computing parameters. Considering the trade-off between effectiveness and accuracy, MML achieves a stable performance with acceptable running time on million-scale datasets.

5 RELATED WORKS

5.1 Matrix Embedding (ME)

Matrix Embedding (ME) is usually employed as a pre-procedure for recommender systems, which projects the user-item matrix into latent spaces for users and items [35], [36].

TABLE 8
Effect of Each Component in MML (Including Learned Metric, Adaptive Margin and Regularizations) on NMI and HR Performance

Models	Datasets	Performance		Models	Datasets	Performance		Models	Datasets	Performance	
EUC-MML	Amazon	NMI	0.3132	W-MML	Amazon	NMI	0.4212	M-MML	Amazon	NMI	0.4309
		HR@50	0.2231			HR@50	0.2744			HR@50	0.2823
	Yelp	NMI	0.3923		Yelp	NMI	0.4832		Yelp	NMI	0.4992
		HR@50	0.2702			HR@50	0.3212			HR@50	0.3212
	Taobao	NMI	0.4734		Taobao	NMI	0.5621		Taobao	NMI	0.5712
		HR@50	0.3823			HR@50	0.4222			HR@50	0.4332
	Jingdong	NMI	0.5012		Jingdong	NMI	0.5432		Jingdong	NMI	0.5543
		HR@50	0.4011			HR@50	0.4532			HR@50	0.4733
Models	Datasets	Performance		Models	Datasets	Performance		Models	Datasets	Performance	
NP-MML	Amazon	NMI	0.5637	NR-MML	Amazon	NMI	0.5766	MML	Amazon	NMI	0.5831
		HR@50	0.3132			HR@50	0.3213			HR@50	0.3318
	Yelp	NMI	0.5600		Yelp	NMI	0.5431		Yelp	NMI	0.5621
		HR@50	0.3611			HR@50	0.3550			HR@50	0.3700
	Taobao	NMI	0.5833		Taobao	NMI	0.6131		Taobao	NMI	0.6321
		HR@50	0.4979			HR@50	0.5051			HR@50	0.5053
	Jingdong	NMI	0.6014		Jingdong	NMI	0.5932		Jingdong	NMI	0.6134
		HR@50	0.5098			HR@50	0.5132			HR@50	0.5288

TABLE 9
Million-Scale Performance (With Amazon Beauty Dataset)

Model	WRMF	ConvMF	SML	MML
NMI-20	0.0832	0.2026*	0.1813	**0.1983**
Time	19,331($\pm$43)	21,334($\pm$178)	11,864($\pm$57)	**10,333**($\pm$62)*

In general, recommender systems without neural networks always use matrix factorization with inner products to get the users' and items' latent representations in a learned latent vector space [4], [11]. There are some popular matrix factorization based matrix embedding models, such as WRMF [26] and SVD [17]. Matrix factorization with inner products works well with some small datasets like Movielens [37]. However, because of some limitations that we introduced above, inner products weaken the performance of recommender systems (collaborative filtering models, user or item-based models) in many aspects.

Recently, as a powerful tool of deep learning, the neural network has been widely applied in recommender systems [5], [38]. The ability of neural networks enhances the recommender system to the next level. As a preprocessing for recommender systems, traditional matrix embedding models can be enhanced by neural networks. [7] develops the neural network framework for MF, and proposes a neural-network based MF model. However, because of weak interpretability [39] and the strong fitting ability for neural networks, most researchers focus on the neural networks' framework rather than the quality of the matrix embedding. Checking the existing recommendation models [7], [17], [29], [39], [40], they usually treat the embedding results as a default and limit the explanation for matrix embedding in details, like LightGCN [41]. While in this paper, we argue that as important representative vectors for users and items, matrix embedding models do affect the performance of recommendations and should be more focused.

5.2 Metric Learning (ML)

Metric learning (ML) is a research spot for image recognition, clustering, and recommendation system [16], [42], [43], [44], [45], [46]. The key to metric learning is how to learn different metrics (such as euclidean distance or other distance metrics) to represent the relationships between different entities instead of inner products. Metric learning is usually applied in the computer vision area, in which a deep transfer metric learning method for cross-domain visual recognition was proposed [47]. For recommender systems, CML [13] directly uses metric learning to embed the relationships between users and items, as shown in the upper part of Fig. 2. And IML [19] proposes a practical framework to accelerate the embedding process.

Recently, some researchers combine metric learning with other existing models to improve performance. Combined with multi-task learning, CRML [22] is a metric learning model proposed for collaborative recommendations with co-occurrence embedding regularization. It considers the optimization problem as a multi-task learning problem which includes optimizing a primary task of metric learning and two auxiliary tasks of representation learning. To combine different styles of loss functions, SML [14] symmetrically introduces a positive item centric metric which maintains a closer distance from positive items to the user and pushes the negative items away from the positive items at the same time with an adaptive margin. Few researches focus on how to utilize metric learning to embed matrix, which is an open issue in the recommender system area.

5.3 Neural-Network Based Recommendation Models (NN-RSs)

The combination of recommender systems and the neural network is becoming a hot research trend [5], [8], [28], [43]. Researchers attempt to utilize the non-linear activation functions in the neural network to measure the relationships between users and reviews. [6] utilizes a Multilayer perceptron (MLP) to design a network NeuCF to tackle implicit feedback recommendation problems. NeuCF is a rating-based model that can cover basic MF and CF and also achieve state-of-the-art performance. [14] combines semi-supervised and neural networks, bridges them, and reinforces mutually.

To tackle the sparse data in real-world scenarios, most existing neural-network based models use two-stage framework: first, it employs the matrix embedding or other models to embed the data into vectors. Then they input these vectors to achieve recommendations [6], [10], [20], [29], [48], [49]. [20] proposes 2IPS, which is a two-stage off-policy policy gradient method. The proposed method explicitly takes into account the ranking model when training the candidate generation model, which helps improve the performance of the whole system. KTUP [29] jointly learns the model of recommendation and knowledge graph completion by combining several transfer schemes. It is an embedding-based recommender model with matrix embeddings. HERec [10] is a heterogeneous network embedding based approach for heterogeneous information network (HIN) based recommendation. It is a path-based recommender model with matrix embeddings. Also some GNN based models [28], [30], [31], [41], [50], [51], [52] are boosting recently, including GraphRec [31], NGCN [30] and LRGCCF [51], which greatly improve the recommender system.

5.4 Relations Among ME, ML and NN-Based RSs

NN-based RSs is an important branch of recommender systems, which utilizes the strong computing ability of neural networks. According to the structure of NN, matrix embedding should be employed to project the abundant information in the user-item matrix into latent vectors. In this paper, we argue that existing ME models are not sufficient, and propose a representation learning model MML, which utilizes the idea of metric learning to enhance the ME performance, and benefit the NN-based RSs.

6 CONCLUSION

The quality of matrix embedding is an imperceptible but important factor in achieving a good recommendation. In this paper, we propose a matrix embedding model: Magnetic Metric Learning, which utilizes dual triplets to embed users and items with a metric-based loss function. With this model, we can achieve a unified embedding in a unified latent vector space. Through the experimental results on four datasets, our model is proved to be superior not only

when compared with state-of-the-art models on all evaluation metrics, but also when trying to find a more stable latent space with the consideration of accuracy, efficiency, and overfitting. Our future work is to apply MML with some context and side information about users and items, to construct a more reasonable similar-pair set for latent relationships.

ACKNOWLEDGMENTS

REFERENCES

[1] G. K. Patro, A. Biswas, N. Ganguly, K. P. Gummadi, and A. Chakraborty, "FairRec: Two-sided fairness for personalized recommendations in two-sided platforms," in *Proc. Web Conf.*, 2020, pp. 1194–1204. [Online]. Available: https://doi.org/10.1145/3366423.3380196

[2] W. Fan, Y. Ma, D. Yin, J. Wang, J. Tang, and Q. Li, "Deep social collaborative filtering," in *Proc. 13th ACM Conf. Recommender Syst.*, 2019, pp. 305–313. [Online]. Available: https://doi.org/10.1145/3298689.3347011

[3] E. Bugliarello, S. Jain, and V. Rakesh, "Matrix completion in the unit hypercube via structured matrix factorization," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, 2019, pp. 2038–2044. [Online]. Available: https://doi.org/10.24963/ijcai.2019/282

[4] F. Ricci, L. Rokach, and B. Shapira, "Recommender systems: Introduction and challenges," in *Recommender Systems Handbook*. Berlin, Germany: Springer, 2015, pp. 1–34.

[5] S. Zhang, L. Yao, and A. Sun, "Deep learning based recommender system: A survey and new perspectives," 2017, *arXiv: 1707.07435*.

[6] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proc. 26th Int. Conf. World Wide Web*, 2017, pp. 173–182.

[7] J. Fan and J. Wang, "A collective neurodynamic optimization approach to nonnegative matrix factorization," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 10, pp. 2344–2356, Oct. 2017. [Online]. Available: https://doi.org/10.1109/TNNLS.2016.2582381

[8] S. Ge, C. Wu, F. Wu, T. Qi, and Y. Huang, "Graph enhanced representation learning for news recommendation," in *Proc. Web Conf.*, 2020, pp. 2863–2869. [Online]. Available: https://doi.org/10.1145/3366423.3380050

[9] Y. Xu et al., "NeuO: Exploiting the sentimental bias between ratings and reviews with neural networks," *Neural Netw.*, vol. 111, pp. 77–88, 2019. [Online]. Available: https://doi.org/10.1016/j.neunet.2018.12.011

[10] C. Shi, B. Hu, W. X. Zhao, and P. S. Yu, "Heterogeneous information network embedding for recommendation," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 2, pp. 357–370, Feb. 2019. [Online]. Available: https://doi.org/10.1109/TKDE.2018.2833443

[11] C. He, D. Parra, and K. Verbert, "Interactive recommender systems: A survey of the state of the art and future research challenges and opportunities," *Expert Syst. Appl.*, vol. 56, pp. 9–27, 2016.

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

[13] C.-K. Hsieh, L. Yang, Y. Cui, T.-Y. Lin, S. Belongie, and D. Estrin, "Collaborative metric learning," in *Proc. 26th Int. Conf. World Wide Web*, 2017, pp. 193–201.

[14] M. Li, S. Zhang, F. Zhu, W. Qian, L. Zang, J. Han, and S. Hu, "Symmetric metric learning with adaptive margin for recommendation," in *Proc. 34th AAAI Conf. Artif. Intell., 32nd Innovative Appl. Artif. Intell. Conf., 10th AAAI Symp. Educ. Adv. Artif. Intell.*, 2020, pp. 4634–4641. [Online]. Available: https://aaai.org/ojs/index.php/AAAI/article/view/5894

[15] A. Acharya, R. Goel, A. Metallinou, and I. S. Dhillon, "Online embedding compression for text classification using low rank matrix factorization," in *Proc. 33rd AAAI Conf. Artif. Intell., 31st Innovative Appl. Artif. Intell. Conf., 9th AAAI Symp. Educ. Adv. Artif. Intell.*, pp. 6196–6203. [Online]. Available: https://doi.org/10.1609/aaai.v33i01.33016196

[16] D. Wang and X. Tan, "Robust distance metric learning via Bayesian inference," *IEEE Trans. Image Process.*, vol. 27, no. 3, pp. 1542–1553, Mar. 2018.

[17] A. N. Nikolakopoulos, V. Kalantzis, E. Gallopoulos, and J. D. Garofalakis, "Eigenrec: Generalizing puresvd for effective and efficient top-n recommendations," *Knowl. Inf. Syst.*, vol. 58, no. 1, pp. 59–81, 2019. [Online]. Available: https://doi.org/10.1007/s10115-018-1197-7

[18] M. Li et al., "Symmetric metric learning with adaptive margin for recommendation," in *Proc. 34th AAAI Conf. Artif. Intell., 32nd Innovative Appl. Artif. Intell. Conf., 10th AAAI Symp. Educ. Adv. Artif. Intell.*, 2020, pp. 4634–4641. [Online]. Available: https://aaai.org/ojs/index.php/AAAI/article/view/5894

[19] N. Wang, X. Zhao, Y. Jiang, and Y. Gao, "Iterative metric learning for imbalance data classification," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, pp. 2805–2811. [Online]. Available: https://doi.org/10.24963/ijcai.2018/389

[20] J. Ma et al., "Off-policy learning in two-stage recommender systems," in *Proc. Web Conf.*, 2020, pp. 463–473. [Online]. Available: https://doi.org/10.1145/3366423.3380130

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

[22] H. Wu, Q. Zhou, R. Nie, and J. Cao, "Effective metric learning with co-occurrence embedding for collaborative recommendations," *Neural Netw.*, vol. 124, pp. 308–318, 2020. [Online]. Available: https://doi.org/10.1016/j.neunet.2020.01.021

[23] E. P. Xing, M. I. Jordan, S. J. Russell, and A. Y. Ng, "Distance metric learning with application to clustering with side-information," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2003, pp. 521–528.

[24] M. Cogswell, F. Ahmed, R. B. Girshick, L. Zitnick, and D. Batra, "Reducing overfitting in deep networks by decorrelating representations," in *Proc. 4th Int. Conf. Learn. Representations*, 2016. [Online]. Available: http://arxiv.org/abs/1511.06068

[25] Y. Xu et al., "Neural serendipity recommendation: Exploring the balance between accuracy and novelty with sparse explicit feedback," *ACM Trans. Knowl. Discov. Data*, vol. 14, no. 4, pp. 50:1–50:25, 2020. [Online]. Available: https://doi.org/10.1145/3396607

[26] Q. Gu, J. Zhou, and C. Ding, "Collaborative filtering: Weighted nonnegative matrix factorization incorporating user and item graphs," in *Proc. SIAM Int. Conf. Data Mining*, 2010, pp. 199–210.

[27] R. Gemulla, E. Nijkamp, P. J. Haas, and Y. Sismanis, "Large-scale matrix factorization with distributed stochastic gradient descent," in *Proc. 17th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2011, pp. 69–77.

[28] X. He, Z. He, J. Song, Z. Liu, Y. Jiang, and T. Chua, "NAIS: Neural attentive item similarity model for recommendation," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 12, pp. 2354–2366, Dec. 2018. [Online]. Available: https://doi.org/10.1109/TKDE.2018.2831682

[29] Y. Cao, X. Wang, X. He, Z. Hu, and T. Chua, "Unifying knowledge graph learning and recommendation: Towards a better understanding of user preferences," in *Proc. World Wide Web Conf.*, 2019, pp. 151–161. [Online]. Available: https://doi.org/10.1145/3308558.3313705

[30] X. Wang, X. He, M. Wang, F. Feng, and T. Chua, "Neural graph collaborative filtering," in *Proc. 42nd Int. ACM SIGIR Conf. Res. Develop. Inf.. Retrieval*, 2019, pp. 165–174. [Online]. Available: https://doi.org/10.1145/3331184.3331267

[31] W. Fan et al. "Graph neural networks for social recommendation," in *Proc. World Wide Web Conf.*, 2019, pp. 417–426. [Online]. Available: https://doi.org/10.1145/3308558.3313488

[32] L. Yang, E. Bagdasaryan, and H. Wen, "Modularizing deep neural network-inspired recommendation algorithms," in *Proc. 12th ACM Conf. Recommender Syst.*, 2018, pp. 533–534. [Online]. Available: https://doi.org/10.1145/3240323.3241618

[33] D. H. Kim, C. Park, J. Oh, S. Lee, and H. Yu, "Convolutional matrix factorization for document context-aware recommendation," in *Proc. 10th ACM Conf. Recommender Syst.*, 2016, pp. 233–240. [Online]. Available: https://doi.org/10.1145/2959100.2959165

[34] J. Han *et al.*, "Adaptive deep modeling of users and items using side information for recommendation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 3, pp. 737–748, Mar. 2020. [Online]. Available: https://doi.org/10.1109/TNNLS.2019.2909432

[35] M. Nilashi, O. Ibrahim, and K. Bagherifard, "A recommender system based on collaborative filtering using ontology and dimensionality reduction techniques," *Expert Syst. Appl.*, vol. 92, pp. 507–520, 2018.

[36] S. Wang, J. Tang, Y. Wang, and H. Liu, "Exploring hierarchical structures for recommender systems," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 6, pp. 1022–1035, Jun. 2018. [Online]. Available: https://doi.org/10.1109/TKDE.2018.2789443

[37] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *ACM Trans. Interactive Intell. Syst.*, vol. 5, no. 4, 2016, Art. no. 19.

[38] Y. Xu, Y. Yang, J. Han, E. Wang, F. Zhuang, and H. Xiong, "Exploiting the sentimental bias between ratings and reviews for enhancing recommendation," in *Proc. IEEE Int. Conf. Data Mining*, 2018, pp. 1356–1361.

[39] N. Senthilkumaran and R. Rajesh, "Image segmentation-a survey of soft computing approaches," in *Proc. Int. Conf. Adv. Recent Technol. Commun. Comput.*, 2009, pp. 844–846.

[40] S. Kabbur, X. Ning, and G. Karypis, "FISM: Factored item similarity models for top-N recommender systems," in *Proc. 19th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2013, pp. 659–667. [Online]. Available: https://doi.org/10.1145/2487575.2487589

[41] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, "LightGCN: Simplifying and powering graph convolution network for recommendation," in *Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2020, pp. 639–648. [Online]. Available: https://doi.org/10.1145/3397271.3401063

[42] H. J. Ye, D. C. Zhan, and Y. Jiang, "Fast generalization rates for distance metric learning," *Mach. Learn.*, vol. 108, pp. 267–295, 2019.

[43] J. Li, A. J. Ma, and P. C. Yuen, "Semi-supervised region metric learning for person re-identification," *Int. J. Comput. Vis.*, vol. 126, no. 8, pp. 855–874, 2018.

[44] X. Sui, E. L. Xu, X. Qian, and T. Liu, "Convex clustering with metric learning," *Pattern Recognit.*, vol. 81, pp. 575–584, 2018.

[45] W. Zuo *et al.*, "Distance metric learning via iterated support vector machines," *IEEE Trans. Image Process.*, vol. 26, no. 10, pp. 4937–4950, Oct. 2017.

[46] S. Chen, C. Gong, J. Yang, Y. Tai, L. Hui, and J. Li, "Data-adaptive metric learning with scale alignment," in *Proc. AAAI Conf. Artif. Intell.*, 2019, pp. 3347–3354.

[47] J. Hu, J. Lu, and Y. P. Tan, "Deep transfer metric learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 325–333.

[48] F. Yuan *et al.*, "Future data helps training: Modeling future contexts for session-based recommendation," in *Proc. Web Conf.*, 2020, pp. 303–313. [Online]. Available: https://doi.org/10.1145/3366423.3380116

[49] C. Chen, M. Zhang, W. Ma, Y. Liu, and S. Ma, "Efficient non-sampling factorization machines for optimal context-aware recommendation," in *Proc. Web Conf.*, 2020, pp. 2400–2410. [Online]. Available: https://doi.org/10.1145/3366423.3380303

[50] S. Wu, Y. Tang, Y. Zhu, L. Wang, X. Xie, and T. Tan, "Session-based recommendation with graph neural networks," in *Proc. 33rd AAAI Conf. Artif. Intell., 31st Innovative Appl. Artif. Intell. Conf., 9th AAAI Symp. Educ. Adv. Artif. Intell.*, 2019, pp. 346–353. [Online]. Available: https://doi.org/10.1609/aaai.v33i01.3301346

[51] L. Chen, L. Wu, R. Hong, K. Zhang, and M. Wang, "Revisiting graph based collaborative filtering: A linear residual graph convolutional network approach," in *Proc. 34th AAAI Conf. Artif. Intell., 32nd Innovative Appl. Artif. Intell. Conf., 10th AAAI Symp. Educ. Adv. Artif. Intell.*, 2020, pp. 27–34. [Online]. Available: https://aaai.org/ojs/index.php/AAAI/article/view/5330

[52] J. Zhang, X. Shi, S. Zhao, and I. King, "STAR-GCN: Stacked and reconstructed graph convolutional networks for recommender systems," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, 2019, pp. 4264–4270. [Online]. Available: https://doi.org/10.24963/ijcai.2019/592

Yuanbo Xu received the BE, ME, and PhD degrees in computer science and technology from Jilin University, Changchun, in 2012, 2015, and 2019, respectively. He is currently a postdoc with the Department of Artificial Intelligence at Jilin University, Changchun. His research interests include applications of data mining, recommender system, and mobile computing. He has published some research results on journals such as *IEEE Transactions on Multimedia*, *ACM Transactions on Knowledge Discovery from Data*, *IEEE Transactions on Neural Networks and Learning Systems* and conference as ICDM, SECON.
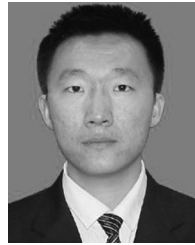
En Wang received the BE degree in software engineering from Jilin University, Changchun, in 2011, the ME and PhD degrees in computer science and technology from Jilin University, Changchun, in 2013, and 2016, respectively. He is currently an associate professor with the Department of Computer Science and Technology at Jilin University, Changchun. He is also a visiting scholar with the Department of Computer and Information Sciences at Temple University, in Philadelphia. His current research interest include efficient utilization of network resources, scheduling and drop strategy in terms of buffer-management, energy-efficient communication between human-carried devices, and mobile crowdsensing.

Yongjian Yang received the BE degree in automatization from the Jilin University of Technology, Changchun, Jilin, China, in 1983, the ME degree in computer communication from the Beijing University of Post and Telecommunications, Beijing, China, in 1991, and the PhD degree in software and theory of computer from Jilin University, Changchun, Jilin, China, in 2005. He is currently a professor and a PhD supervisor with Jilin University, director of the Key lab under the Ministry of Information Industry, standing director of the Communication Academy, member of the Computer Science Academy of Jilin Province. His research interests include theory and software technology of network intelligence management, Key technology research of wireless mobile communication and services. He participated 3 projects of NSFC, 863 and funded by National Education Ministry for Doctoral Base Foundation. He has authored 12 projects of NSFC, key projects of Ministry of Information Industry, Middle and Young Science and Technology Developing Funds, Jilin provincial programs, ShenZhen, ZhuHai, and Changchun.

Yi Chang (Senior Member, IEEE) is dean of the School of Artificial Intelligence, Jilin University. His research interests include information retrieval, data mining, machine learning, natural language processing, and artificial intelligence. He is an associate editor of *IEEE Transactions on Knowledge and Data Engineering*, and he served as one of the conference general chairs for ACM WSDM'2018 and ACM SIGIR'2020. He is ACM distinguished scientist.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/csdl.