Received November 28, 2018, accepted December 27, 2018, date of publication January 11, 2019, date of current version January 29, 2019.

Digital Object Identifier 10.1109/ACCESS.2018.2890480

Improving Recommendations by Embedding Multi-Entity Relationships With Latent Dual-Metric Learning

YUANBO XU[1], YONGJIAN YANG[1], JIAYU HAN[1], XIANG LI[2], AND EN WANG[1]
[1]College of Computer Science and Technology, Jilin University, Changchun 130012, China
[2]High Performance Computing Center, Jilin University, Changchun 130012, China

Corresponding author: En Wang (wangen@jlu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61772230, in part by the Natural Science Foundation of China for Young Scholars under Grant 61702215, and in part by the China Postdoctoral Science Foundation under Grant 2017M611322 and Grant 2018T110247.

ABSTRACT Recently, latent vector embedding has become a research hotspot, with its great representative ability to measure the latent relationships among different views. However, most researches utilize the inner product of latent vectors as the representation of relationships, and they develop some embedding models based on this theory. In this paper, we take deep insight into the existing embedding models and find that utilizing the inner product may increase several problems: 1) in latent space, the inner product among three vectors may violate triangle principle; 2) the inner product cannot measure the relationships between vectors in the same category, such as user and user and item and item; and 3) the inner product cannot catch the collaborative relationships (user–user and item–item) for collaborative filtering. Along with this line, we propose a latent vector embedding model for collaborative filtering: latent dual metric embedding (LDME), which utilizes the dual-Euclidean distance in latent space, instead of the inner product, to represent different types of relationships (user–user, item–item, and user–item) with a uniform framework. Specifically, we design an embedding loss function in LDME, which can measure the close and remote relationships between entities, tackle the above problems, and achieve a more clear, well-explained embedding result. Extensive experiments are conducted on several real-world datasets (Amazon, Yelp, Taobao, and Jingdong), where the expiring results demonstrate that LDME can overperform some state-of-the-art user–item embedding models and can benefit the existing collaborative filtering models.

INDEX TERMS Latent vector embedding, metric learning, collaborative filtering, recommender system.

I. INTRODUCTION

Recently latent vector embedding has become a popular research spot, by its powerful ability to represent the complex relationships, where can be applied in various areas, such as text mining [1], POI prediction [2] and recommender system [3]. Basically, latent vector embedding model makes efforts to measure the non-linear relationships, by mapping existing items (words, POIs, or users and items) into a k-dimension latent space, then a non-linear relationship in original space can be transferred into a relatively linear relationship in this latent space. Latent vector embedding borrows the idea of SVM and always achieves satisfying results for recommendation and relationship visualization.

However, traditional latent embedding models usually utilize inner product to represent the relationship between different vectors as default, which may cause some important problems. Traditional latent embedding models always utilize inner products, which is shown in Fig.1. In this example,

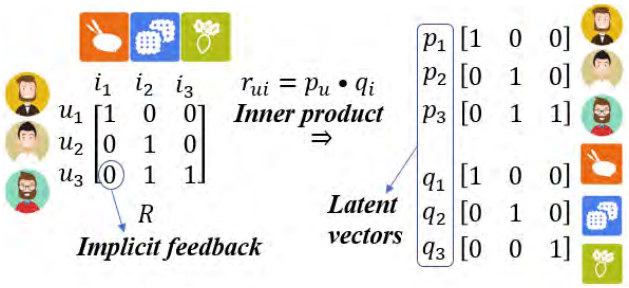

FIGURE 1. A stable embedding results with inner product function.

matrix R stands for the relationships between entity set $U = \{u_1, u_2, u_3\}$ and $I = \{i_1, i_2, i_3\}$. The relationships of u and i can be represented as r_{ui}. $r_{ui} = 1$ means that there is an implicit relationship between u and i, and $r_{ui} = 0$ means there is no relationship or the relationship is not observed between u and i. We utilize inner product as loss function $r_{ui} = p_u \bullet q_i$ and embeds all the entities U, I into a same 3-dimension latent space, where p_u stands for the latent vector of user u and q_i stands for the latent vector of item i and $\bullet$ stands for inner product. The results of a latent vector embedding with inner product loss function are shown in Fig.1.

A. INSIGHT OF TRADITIONAL EMBEDDING MODEL

When taking deep inside of this embedding with the inner product, we notice this embedding is stable because all the latent vectors meet the constraint of r_{ui} and inner product. It is important for embedding with inner product that $r_{ui} = p_u \bullet q_i, i \in I, u \in U$. Though R can be embedded into various P, Q, this is an exactly stable and representative one because it meets whole constraints perfectly.

However, if we treat the matrix R as a user-item matrix in the real world, and prepare to utilize the embedding vectors to make recommendations with collaborative filtering model [4], [5], several important issues occur. First, in latent space, the inner product among three vectors may violate triangle principle. We treat all the embedding entities (P, Q) as vectors in a 3-dimension latent space. So their relationships can be measured as ''distance'' among them. If we utilize inner product as a metric for ''distance'', we find that in some situation, the embedding vector does not meet the triangle principle (the sum of any two sides of a triangle is greater than the third side). We still utilize the example in Fig.1 (Hsieh *et al.* [6] have utilized 2-dimension embedding as an example, but we believe 3-dimension is a more common situation). We select p_1, p_2, q_1 as three points in latent space. $p_1 \bullet p_2 = 0$, $p_1 \bullet q_1 = 1$ and $p_2 \bullet q_1 = 0$. Notice $p_1 \bullet p_2 + p_2 \bullet q_1 = 0 < p_1 \bullet q_1 = 1$, which violates the triangle principle. This may cause chaos when we utilize the embedding vector to calculate the relationships in latent space.

Second, utilizing inner product as embedding metric can not represent the relationships between the different categories of entities, such as user to user, or item to item. In recommender systems, we always want to find the latent relationships between different users or items. The embedding vectors in Fig.1 with inner product only consider the relationships between users and items and ignore item to item, user to user relationships. Let us focus on user u_3. u_3 has relationships with i_2 and i_3, so there should be a relationship between them. However, when we check the embedding q_2 and q_3, we find that $q_2 \bullet q_3 = 0$, which means there is no relationship between i_2 and i_3. Moreover, $p_1 \bullet p_2 = 0$ and $p_1 \bullet p_3 = 0$, we cannot tell which of u_3 or u_2 is closer to u_1. So utilizing inner product cannot tackle relationships between the different categories of entities well.

Finally, the inner product cannot catch the collaborative relationships (user-user, item-item) for collaborative filtering. If we want to choose i_1 or i_3 as a recommendation for u_2, we cannot decide based on latent vectors in Fig.1 because $p_2 \bullet q_1 = 0$ and $p_2 \bullet q_3 = 0$. But notice that u_2 and u_3 both have relationships with i_2, which means that u_2 shares a same preference as u_3. So i_3 should be a better recommendation than i_1, and we cannot find it through embedding vectors with inner product.

B. OUTLINES OF THIS WORK

With all the issues above, the inner product seems to be an improper metric to measure the relationships between different entities for collaborative filtering in recommender system because 1) in latent space, the inner product among three vectors may violate triangle principle. 2) inner product cannot measure the relationships between vectors in the same category, such as user and user, item and item. 3) The inner product cannot catch the collaborative relationships (user-user, item-item) for collaborative filtering. It is quite an important issue to find a proper metric and design a proper loss function to measure the relationship and make accurate embeddings.

In this paper, we focus on the issues above and attempt to utilize the idea of metric learning to measure the relationships between users and users, items and items as an additional compensation for implicit collaborative filtering. Metric learning is usually applied on multimedia area, such as image and video [7]. And traditional metric learning is designed to measure the relationship between users and items only, which cannot meet the requirement of implicit collaborative filtering. Along with this line, we propose a Latent Dual Metric Embedding (LDME), which utilizes the dual Euclidean distance in latent space, instead of inner product, to represent the different types of relationships (user-user, item-item, user-item) with a uniform framework. With calculating Euclidean distance in latent space, all the entities (users and items) can be embedded into a same latent space where their relationships can be directly seen through distance. In LDME, a loss function is designed for constructing the relationships between users and users, items and items, where LDME can follow the triangle principle, cluster users who share the same preferences and items that consume by same user, and benefit the implicit collaborative filtering. Extensive experiments are conducted on real-world datasets and the results show that LDME can achieve a superior performance than traditional inner product embedding models and benefit the existing state-of-the-art implicit collaborative filtering model.

The contributions are summarized as follows:

- We notice the problems that inner product occurs in some latent vector embedding models: 1) the inner product among three vectors may violate triangle principle. 2) the inner product cannot measure the relationships between vectors in the same category. 3) the inner product cannot catch the collaborative relationships

(user-user, item-item) for collaborative filtering. And we propose that utilizing dual Euclidean distance can embed the entities into the same space which can save the problems above.
- We propose a Latent Dual Metric Embedding (LDME), which utilizes the dual Euclidean distance in latent space, instead of inner product, to represent the different types of relationships (user-user, item-item, user-item) with a uniform framework. A loss function which measures the relationships between different entities is also proposed in LDME. This model is theoretical so it can be applied in different scenarios.
- We apply the proposed model LDME to four large-scale datasets. The effectiveness experiments demonstrate our LDME model is significantly better than the baselines for measuring relationships between users and users, items and items. The effect experiment demonstrates that LDME can benefit state-of-the-art implicit collaborative filtering.

The rest of the paper is organized as follows: in the next section, we give a brief introduction to the related works. Basic definitions and problem definitions are given in Section 3. Then, we introduce the details in Section 4. In Section 5, we conduct experiments to evaluate our proposed model. Finally, we conclude our work in Section 6.

II. RELATED WORKS

A. METRIC LEARNING

Metric learning is a research spot in recent years, as in image recolonization, clustering and recommendation system [7]–[17]. Metric learning is delighted by the inner product, which is a traditional way to measure the relationships between different entities. The idea of metric learning is to utilize different metrics (such as Euclidean distance or other distance metrics) to represent the relationships between different entities. Metric learning first occurred in image and computer vision area [9], [14], where Liu *et al.* [14] proposed deep transfer metric learning (DTML) method to learn a set of hierarchical nonlinear transformations for cross-domain visual recognition by transferring discriminative knowledge from the labeled source domain to the unlabeled target domain, and Wang and Tan [9] utilized a novel semi-supervised region metric learning method to improve person re-identification performance under imbalanced unlabeled data.

Metric learning also can be applied in other areas, such as traffic detection [13], classification [16] and recommendation [15], [17]. Hu *et al.* [13] proposed a locality constraint distance metric learning for traffic congestion detection and achieved a state-of-the-art performance. Yu *et al.* [16] focused on binary classification problems, formulated the sub-space learning problem as a particular Burg Matrix optimization problem that of minimizing the Burg Matrix divergence with distance constraints, then solved the problem with metric learning. Especially for recommender systems, Shen *et al.* [17] tried to utilize metric learning and matrix factorization to make a recommendation, which is a good application for a traditional metric learning. However, researchers always focus on the application of traditional metric learning and do not consider the insights of its ability to represent relationships, which is exactly what we want to study in this paper.

B. EMBEDDING OF RECOMMENDER SYSTEM

The embedding model in recommender systems and the neural network is becoming a hot research trend [18]–[25]. Researchers attempt to use neural network to measure the relationships between users and items. He *et al.* [18] utilized NLP to design a network, which is named NeuCF to tackle implicit feedback recommendation problems. NeuCF is a model which can cover some basic MF and CF models. Yang *et al.* [23] proposed a novel concept: Serendipity and they utilized an MLP-based network to tackle the serendipity issues in recommender systems, also with the embedding model NeuCF proposed. Attention vectors are also employed by some researchers for embedding. Chen *et al.* [21] embedded users and items with different attention layers and achieved a good performance. Seo *et al.* [26] used local and global attention vectors to optimize user embedding results in recommender systems. These models put more attention on the methodology of neural network itself rather than the applications in real scenarios, which also achieve a satisfying performance on various real-world datasets. Moreover, Bai *et al.* [19] embedded the relations between neighbors and proposed a neural-network based recommender system. Also, some researchers try to combine neural models with traditional machine learning to make recommendations. Yang *et al.* [20] combined semi-supervised and neural network, bridged them and reinforced mutually.

These works have made improvement in accuracy and efficiency, however, they almost validate their on standard datasets like movielens [27], or yelp [28] with the inner product as loss function, or traditional metric learning model which does not consider to put the relations between users and users, items and items explicitly into loss function. Therefore, we propose LDME in this paper, which utilizes the dual Euclidean distance in latent space, instead of inner product, to represent the different types of relationships (user-user, item-item, user-item) with a uniform framework. A loss function which measures the relationships between different entities is also proposed in LDME. This model is theoretical so it can be applied in different scenarios.

III. PRELIMINARIES

In this section, we give some definitions about embedding with the inner product, embedding with metric learning and problem definitions.

A. BASIC DEFINITION

In a recommender system, let U be a set of m users $U = \{u_1, u_2 ... u_m\}$, and I be a set of n items $I = \{i_1, i_2 ... i_n\}$. r_{ui} means the rating user u marked for item i, and in implicit

feedback, $r_{ui} = 0$, or 1. So we build a rating user-item matrix $R_{m\times n}$, whose entries are 1s for items with feedbacks, 0s for items without feedbacks.

For latent embedding, we utilize $P = \{p_1, p_2...p_m\}$ as the users' latent embedding vectors in latent k_P dimension space, and $Q = \{q_1, q_2...q_n\}$ as the items' latent embedding vectors in latent k_Q dimension space. To simply the computation, we assume users and items are embedded into a same k dimension latent space, $k = k_P = k_Q$.

B. EMBEDDING WITH INNER PRODUCT

Given a user-item matrix R, the embedding models with inner product always employ the following loss function:

$$\arg\min_{P,Q} \sum_{P,Q}^{U,I} \Big((r_{ui} - p_u \bullet q_i)^2 + \theta(P, Q) \Big), \tag{1}$$

where $p_u \in P, q_i \in Q, u \in U, i \in I, r_{ui} \in R$. $\theta(P, Q)$ is a penalty term to avoid overfitting.

With this loss function, user u is embedded into a k dimension vector p_u, and item i is embedded into q_i, which is also a k dimension vector. To explain the inner product, we can treat every entry of p_u as u's preference on different views of an item, and every entry of q_i as item i's distributions on different views. Then inner product of p_u and q_i are treated as the predictions. When applied with collaborative filtering, we utilize the embedding vectors P, Q, compute the top-k neighbors of target user or item, then make recommendations.

$$Top@k \left(\min_{u\in U} Euc(p_u, p_t) \right). \tag{2}$$

Note that when we compute the top-k neighbors, we always utilize the Euclidean distance of different users and items, which is also an illogical default because we compute the relationships with the inner product between user and item, but utilize Euclidean distance between user and user, item and item.

C. EMBEDDING WITH METRIC LEARNING

Given a user-item matrix R, the embedding models with metric learning always employ the following loss function:

$$\arg\min_{P,Q} \sum_{P,Q}^{U,I} \left(\underset{r_{ui}=1}{L_{pull}(p_u, q_i)} - \underset{r_{ui}=0}{L_{push}(p_u, q_i)} + \theta(P, Q) \right), \tag{3}$$

where $p_u \in P, q_i \in Q, u \in U, i \in I, r_{ui} \in R$. $\theta(P, Q)$ is a penalty term to avoid overfitting.

Note that there are two loss functions in metric learning: L_{pull} and L_{push}. The core idea for metric learning is to gather the entities with relationships and push away the entities without relationships. So L_{pull} is employed to calculate the Euclidean distance between user u and item i, where $r_{ui} = 1$, and try to pull the user and item together. And L_{push} is employed to push away the user and item where $r_{ui} = 0$. Metric learning also employs Eq.2 to select neighbors and make recommendations. However, traditional metric learning only focuses on measuring the relationships between users and items without considering the relationships between user and user, item and item, which can be improved by our proposed model.

D. PROBLEM DEFINITION

Given a user-item matrix R with implicit feedbacks, the problems we want to utilize the embedding model to solve are:

- Represent the relationships between different categories (d-rel) of entities (such as user and item) in a latent space with a uniform embedding framework.

$$d - rel(p_u, q_i) = ||p_u - q_i||; \quad u \in U, \ i \in I. \tag{4}$$

- Represent the relationships between same category (s-rel) of entities (such as user and user, item and item) in a latent space with a uniform embedding framework.

$$s - rel(p_u, p_t) = ||p_u - p_t||; \quad u, t \in U. \tag{5}$$
$$s - rel(q_i, q_w) = ||q_i - q_w||; \quad i, w \in I. \tag{6}$$

  where $||p - q||$ means some distance measure metric of vector p and q.
- The loss function of metric learning to measure the relationships should be stable enough to follow the triangle principle, benefit collaborative filtering models and make accurate Top-k recommendations.

Some important notations are shown in Table1. To solve the problem above, we propose LDME, whose details are introduced in the following section.

TABLE 1. Notations in this paper.

Notation	Description
U	User set in our recommender system
I	Item set in our recommender system
U^+	Similar user pair set
I^+	Similar item pair set
I_u	Item set with $r_{ui} = 1$
U_i	User set with $r_{ui} = 1$
m, n	Number of users/items
k	Latent embedding dimension
r_{ui}	u's feedbacks on item i
R	Matrix with implicit feedbacks r_{ui}
P	User latent embedding vector set
Q	Item latent embedding vector set
$s/d - rel$	Same/Different categories relationships
p_u, q_i	user u/item i's latent embedding vectors
λ, β	hyperparameter and threshold of LDME

IV. PROPOSED MODEL-LDME

The above discussion highlights the fact that, the focus of implicit collaborative filtering is no longer about estimating a specific rating matrix but about capturing users relative preferences for different items, and the relations between different and same categories.

Based on the definitions above, we first introduce the details about LDME, including the loss function and the

embedding structures, considering how to measure the relationships between different categories (user-item) and same category (user-user, item-item). Then we discuss the similarity and difference among our proposed model and traditional inner-product based models and metric based models. Finally, we discuss how LDME can benefit the implicit collaborative filtering and the extension ability of LDME to be applied on other models.

A. LATENT DUAL METRIC EMBEDDING-LDME

1) EMBEDDING LOSS FUNCTION

We propose LDME, which utilizes dual-metric learning to measure the relationships between different users and items. First, we decide to utilize the Euclidean distance to measure the relationships among entities (users and items), where the embedding results will follow the triangle principle and easy to explain. The relationship functions are shown as follows:

$$d - rel(p_u, q_i) = ||p_u - q_i||_e; \quad u \in U,\ i \in I. \tag{7}$$

$$s - rel(p_u, p_t) = ||p_u - p_t||_e; \quad u, t \in U. \tag{8}$$

$$s - rel(q_i, q_w) = ||q_i - q_w||_e; \quad i, w \in I. \tag{9}$$

where $||p - q||_e$ means Euclidean distance of vector p and q.

Traditional metric learning models only consider the relationship between different categories, for example, users and reviews. While in our proposed model LDME, we not only consider the relations as traditional metric learning model does, but also take the relationships between entities in same category into consideration. So the style of our embedding loss function is as follows:

$$L_{LDME} = \sum_{P,Q}^{U,I} (\alpha_1 L_D(p_u, q_i) + \alpha_2 L_S(p_u, p_t) + \alpha_3 L_S(q_i, q_w)) + \theta_{pen}, \tag{10}$$

where $u, t \in U$, $i, w \in I$, $p, q \in P, Q$ and L_D stands for the relationships between different categories and L_S stands for the relationships between same categories. α stands for the weights and θ_{pen} stands for the penalty term.

We define L_D following the traditional metric learning style, which is composed with two components: L_{push} and L_{pull} as follows:

$$L_D(p_u, q_i) = \sum_{P,Q}^{U,I} \left(\lambda_{d1} \underset{r_{ui}=1}{L_{pull}}(p_u, q_i) + \lambda_{d2} \underset{r_{ui}=0}{L_{push}}(p_u, q_i) + 1 \right), \tag{11}$$

where $\underset{r_{ui}=1}{L_{pull}}(p_u, q_i) = d - rel(p_u, q_i)$, $\underset{r_{ui}=0}{L_{push}}(p_u, q_i) = -(d - rel(p_u, q_i))$. L_{pull} gathers the items around the user if $r_{ui} = 1$ and L_{push} tries to add the distance between users and items if $r_{ui} = 0$. $\lambda_{d1}, \lambda_{d2}$ are scalar hyper parameters ranging in (0,1). We add 1 at the end to avoid overfitting and chaos noise.

The major difference between our proposed model and traditional metric learning models is that we take the relationships among the same category into consideration, which is represented by L_s. In implicit collaborative filtering scenarios, we have two categories: users and reviews. So we design two similar sets: similar user pair set U^+ and similar item pair set I^+, which are defined as follows:

Definition U^+: The similar user pair $< u, t >$ is in the similar user pair set U^+, if and only if $\delta_{u,t} = 1$, where $\delta_{u,t}$ is an indicator which is predefined to restrict the user's similarity.

Definition I^+: The similar user pair $< i, w >$ is in the similar user pair set I^+, if and only if $\delta_{i,w} = 1$, where $\delta_{i,w}$ is an indicator which is predefined to restrict the item's similarity.

And based on U^+ and I^+, we define $L_S(p_u, p_t)$ and $L_S(q_i, q_w)$ as follows:

$$L_S(p_u, p_t) = \sum_{P,Q}^{U,I} \left(\lambda_{s1} \underset{<u,t>\in U^+}{L_{pull}}(p_u, p_t) + \lambda_{s2} \underset{<u,t>\notin U^+}{L_{push}}(p_u, p_t) + 1 \right), \tag{12}$$

$$L_S(q_i, q_w) = \sum_{P,Q}^{U,I} \left(\lambda_{s3} \underset{<i,w>\in I^+}{L_{pull}}(q_i, q_w) + \lambda_{s4} \underset{<i,w>\notin I^+}{L_{push}}(q_i, q_w) + 1 \right), \tag{13}$$

where $L_{pull}(p_u, p_t) = s - rel(p_i, p_t)$, and $L_{push}(p_u, p_t) = s - rel(p_i, p_t)$. So do $L_{pull}(q_i, q_w)$ and $L_{push}(q_i, q_w)$. So our proposed model LDME's embedding goal is shown as follows:

$$\underset{P,Q}{\arg\min} (L_{LDME} | \alpha, \lambda, \delta,) \tag{14}$$

2) REGULARIZATION AND MODEL TRAINING

a: REGULARIZATION

A proper regularization is quite crucial to the feasibility of the proposed model. Our model LDME essentially projects users and items to a same joint k-dimensional latent space. The number of dimensions determines the representational capacity of the model. However, an embedding model like what we propose is known to be ineffective in a high-dimensional space if the data points spread too widely (i.e., the curse of dimensionality) [6], [29]. Therefore, we bound all the user/item latent vectors within a unit sphere, i.e.,

$$\|p_u\| \le 1, \quad \|q_i\| \le 1,\ u, i \in U, I;\ p, q \in P, Q. \tag{15}$$

to ensure the robustness.

Another important point of LDME is the parameter α, λ and the indicator δ. We need to make sure that the different relationships play same roles in LDME. However, we notice that the relationships between users and items always overaffect the other relations, so we utilize α, λ to make limitations as follows:

$$\alpha_1 \le \alpha_2 + \alpha_3, \alpha_1 + \alpha_2 + \alpha_3 = 1. \tag{16}$$

$$\lambda_{d1} + \lambda_{d2} = \lambda_{s1} + \lambda_{s2} = \lambda_{s3} + \lambda_{s4} = 1. \tag{17}$$

Then we take consideration of threshold δ, we define the threshold with the scale of users and items, which are shown as follows:

$$\left\{\delta_{u,t} = 1 | (I_u \subseteq I_t) \quad or \ (I_t \subseteq I_u) \quad or \quad \frac{|I_u \cap I_t|}{|I_u \cup I_t|} > \frac{|I^{avg}|}{|I|}\right\}, \tag{18}$$

$$\left\{\delta_{i,w} = 1 | (U_i \subseteq U_w) \quad or \ (U_w \subseteq U_i) \quad or \quad \frac{|U_i \cap U_w|}{|U_i \cup U_w|} > \frac{|U^{avg}|}{|U|}\right\}, \tag{19}$$

where I^{avg}, U^{avg} stands for the average items/users that $r_{ui} = 1$. It means that only when a user's feedbacks are covered by another user, or they share a similar preference over average level, we put them into the similar user set U^+, so does I^+.

b: MODEL TRAINING

The object function of our proposed model is show as follows:

$$\min_{\theta,P,Q} \{L_{LDME} | \alpha, \lambda, \delta\}$$
$$\text{s.t. Eq.16, Eq.17, Eq.18, Eq.19.}$$

According to this object function, we should minimize this constrained objective function with Mini-Batch Stochastic Gradient Descent (SGD), achieve the latent embedding vectors P, Q and control the learning rating using AdaGrad [30], as suggested in [31].The details are introduces in experimental section. Our training procedure is as follows:

1) Randomly select negative samples to set $r_{ui} = 0$;
2) Predefine α, λ and calculate δ;
3) Hold the negative items and form a mini-batch of size N;
4) Compute gradients and update parameters with AdaGrad;
5) Repeat with regularizations until convergence.

3) RECOMMENDATION

After the computation about latent vectors $\hat{P}$, $\hat{Q}$, we can easily compute the relations between different users and items, then we can apply this embedding results to make a better recommendation.

$$score_{ui} = ||\widehat{p}_u - \widehat{q}_i||_e$$

V. EXPERIMENTAL RESULTS

A. DATASETS

To validate the effectiveness of LDME, we conduct abundant experiments on Amazon.com dataset[1] and Yelp for RecSys.[2] Amazon and Yelp datasets are two public datasets. Moreover, we also collect two real-world datasets from Taobao[3] and Jindong.[4] And we use 5-cross validation to divide the datasets, with 80% as training set, 10% as test set and 10% as validation set. The details of datasets are shown in Table 2. From Table 2, we can see that these datasets are extremely sparse. We set $r_{ui} = 1$ if there is a rating that user u rated item i.

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
Sparsity	0.051%	0.043%	0.049%	0.12%

B. BASELINES

We compare our model with the following baselines:

1) Weighted Regularized Matrix Factorization (WRMF) [32]: the implicit MF model that uses an additional case weight to model unobserved interactions, which utilizes inner products.

2) Attentive Collaborative Filtering (A-CF) [21]: A-CF utilizes item- and component-level attention models to assign attentive weights for inferring the underlying users preferences encoded in the implicit user feedback. And A-CF can achieve a superior performance over traditional collaborative filtering methods, which utilizes inner products.

3) Adaptive Matrix Factorization (A-MF) [33]: A-MF is designed to learn personal models based on adapting the popular gradient descent optimization techniques. And A-MF can achieve a superior performance over traditional matrix factorization methods, which utilizes inner products.

4) Neural Collaborative Filtering (NeuCF) [18]: NeuCF borrows the idea from Matrix Decomposition to build a neural network. It generalizes a framework of neural networks to cover basic collaborative filtering, which utilizes direct combination and MLP.

5) Collaborative Metric Learning (CMF) [6]: Collaborative Metric Learning (CML) which learns a joint metric space to encode not only users preferences but also the user-user and item-item similarity. But CMF doesn't consider the fact that the similarity should also be measured in the loss function and make effects in the training process like LDME does.

C. PARAMETER SETTING

We initialize some parameters: Word Embedding Dimension $k = 32$, learning rate $= 0.0001$ with optimizer AdaGrad. To make the experiment simply, we set $\alpha_1 = \alpha_2 + \alpha_3 = 0.5$, which means that the importance of user-item is the same as the sum of user-user and item-item. Then we set parameter $\lambda = 0.5$ and threshold $\delta = 0.3$. We also set the parameters of baselines as [6], [21], [32], and [33] to make a plain comparison. Our experiments are operated with Pytorch running on GPU: GeForce GTX TITAN X with mini-batch N = 60. If we do not introduce specifically, we utilize the default parameters.

[1] https://jmcauley.ucsd.edu/data/amazon
[2] https://www.kaggle.com/c/yelp-recsys-2013
[3] https://www.taobao.com
[4] https://www.jd.com

D. EXPERIMENTAL RESULTS AND DISCUSSIONS

1) RECOMMENDATION ACCURACY

We employ *hitting ratio* (*HR*) and *Normalized Discounted Cumulative Gain* (**NDCG**) [34] as the evaluation metric for recommendation list. The following figure shows the results of *HR* and *NDCG* on different datasets with different embedding dimensions (4, 8, 16, 32, 64).

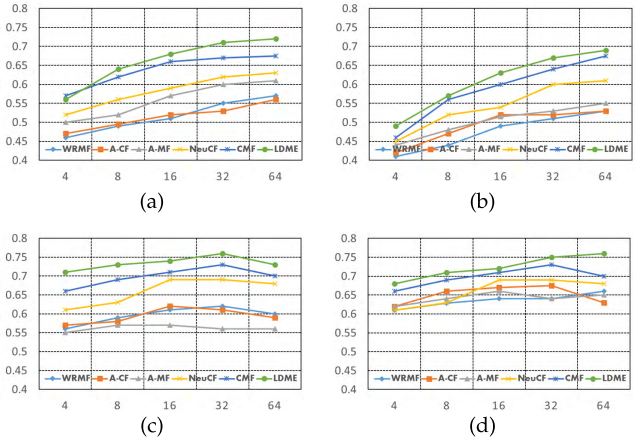

FIGURE 2. Performance of HR@10 *w.r.t* the number of latent dimension *k*. (a) HR@10 with Amazon. (b) HR@10 with Yelp. (c) HR@10 with Taobao. (d) HR@10 with Jingdong.

From Fig.2, we can clearly see the models perform better in small and dense datasets (Taobao and Jingdong) than in big and sparse datasets (Amazon and Yelp). Moreover, our proposed model overperforms the baselines across four datasets with different embedding latent dimensions. Note that when the latent dimension increases, HR performance also increases. The reason is that the number of latent dimension means the ability of the latent vector to express the latent relationships of users and items. Meanwhile, we find that because WRMF, A-CF, A-MF utilize the inner product as their loss function and make recommendations, they perform not as well as the other baselines and our proposed model LDME. The results also show that the inner product is not a good metric and loss function for user-item embedding. NeuCF employs a raw direct of user and item latent vectors and utilizes MLP to make the embedding, which is a smart idea. However, as the output of NeuCF is the feedback (r_{ui}), which cannot measure the importance of different entities. So it can be a result of inner product style, or Euclidean distance style, which makes the NeuCF's performance unstable, especially on big and sparse datasets (Amazon and Yelp). CML is a state-of-the-art metric learning model for collaborative filtering. However, because CML doesn't take user-user and item-item relationship explicitly in the loss function and control the importance through a predefined weight, the multiple relationships may affect the embedding results weaker than our proposed model, which leads to a low performance.

From Fig.3, the results of NDCG also show us some interesting phenomena. First, we can clearly see the models perform better of NDCG in small and dense datasets (Taobao and Jingdong) than in big and sparse datasets (Amazon and Yelp), like HR. Moreover, our proposed model also overperforms the baselines across four datasets with different embedding latent dimensions. However, the distance between LDME and baselines is not so far like HR. The reason is that NDCG is a metric of evaluating a rank or a list, while in implicit feedbacks, we focus on HR better than NDCG. Even though, our proposed model LDME is still average 7% over other baselines, which also demonstrates the accuracy of our proposed model.

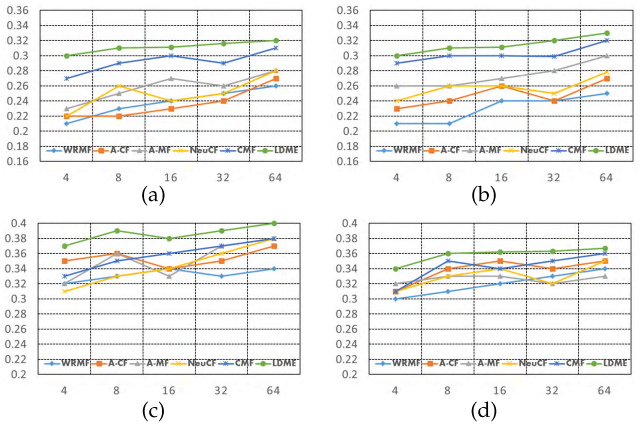

FIGURE 3. Performance of NDCG@10 *w.r.t* the number of latent dimension *k*. (a) NDCG@10 with Amazon. (b) NDCG@10 with Yelp. (c) NDCG@10 with Taobao. (d) NDCG@10 with Jingdong.

Then we conduct some performance to evaluate the accuracy of our proposed model with changing the recommendation list number **K** (**HR@K**). We change **K** from 1 to 10 on a big dataset Amazon and a small dataset Taobao. And the results are shown in Fig.4:

Because we focus on HR better than NDCG, Fig.4 shows the performance of Top-K recommendation lists, where the *K* ranges from 1 to 10. To make the results more clear and direct, we show the performance of all different baselines models. As we can point out that, LDME demonstrates consistent improvements over other models across different recommendation list lengths. This is consistent with the results shown in Fig.2, Fig.3, demonstrating that LDME can achieve a strong performance for recommendations.

2) PARAMETER DECISION

As we introduced before, there are three important parameters in our proposed model LDME α, λ and δ. For a simple description, α is for the weight of use-item relationships, and $1-\alpha$ stands for other relationships, λ is the weight of pull function and $1-\lambda$ stands for push function. δ is a threshold to control the similarity user and item sets. So we need to find out the best parameters according to the experiments. We utilize HR@10 as the performance metric and conduct our experiment on two datasets Amazon and Taobao. Without any exception, if we change one of the three parameters, we fix the other parameters. The results are shown in Fig.5.

From the results, we can clearly see that different parameters make different efforts on the results. First, we think

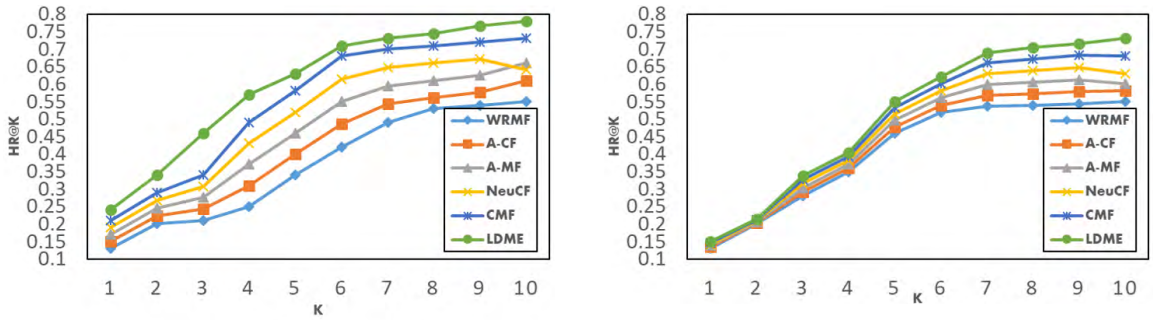

FIGURE 4. Performance of HR@K *w.r.t* the number of recommendation list *K*. (a) HR@K with Amazon. (b) HR@K with Taobao.

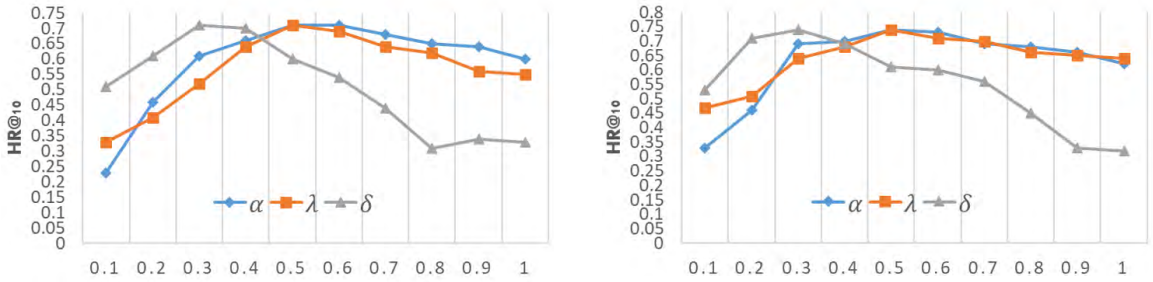

FIGURE 5. Performance of HR@10 *w.r.t* parameter α, θ, δ. (a) HR@10 of LDME with Amazon. (b) HR@10 of LDME with Taobao.

TABLE 3. Effect of LDME.

	Amazon		Yelp		Taobao		Jindong	
HR@3	P, Q	$\hat{P}, \hat{Q}$	P, Q	$\hat{P}, \hat{Q}$	P, Q	$\hat{P}, \hat{Q}$	P, Q	$\hat{P}, \hat{Q}$
Item-CF	0.131	**0.170**	0.172	**0.200**	0.091	**0.161**	0.133	**0.169**
User-CF	0.141	**0.176**	0.177	**0.205**	0.089	**0.156**	0.102	**0.146**
NeuCF	0.150	**0.218**	0.174	**0.213**	0.094	**0.185**	0.094	**0.164**
HR@5	P, Q	$\hat{P}, \hat{Q}$	P, Q	$\hat{P}, \hat{Q}$	P, Q	$\hat{P}, \hat{Q}$	P, Q	$\hat{P}, \hat{Q}$
Item-CF	0.211	**0.224**	0.179	**0.216**	0.102	**0.154**	0.122	**0.147**
User-CF	0.171	**0.199**	0.189	**0.231**	0.104	**0.136**	0.112	**0.126**
NeuCF	0.23	**0.218**	0.222	**0.243**	0.143	**0.176**	0.134	**0.209**
HR@10	P, Q	$\hat{P}, \hat{Q}$	P, Q	$\hat{P}, \hat{Q}$	P, Q	$\hat{P}, \hat{Q}$	P, Q	$\hat{P}, \hat{Q}$
Item-CF	0.325	**0.346**	0.314	**0.333**	0.349	**0.371**	0.322	**0.369**
User-CF	0.247	**0.277**	0.217	**0.245**	0.145	**0.176**	0.201	**0.256**
NeuCF	0.456	**0.501**	0.433	**0.471**	0.494	**0.518**	0.501	**0.564**

about α, which controls the weight of user-item relationship. When α is too small, we lose the ability to embed user-item relationship, which leads to a bad performance. If it is too close to 1, LDME fades to CML and achieves the same level performance as CML, worse than LDME. The same situation happens to θ. When θ is close to 0, the push function makes sense and tries to overperform the pull function, so LDME achieves its best performance when pull and push function both work.

Finally, we consider threshold δ, which controls the set of similar user set and item set. If the threshold is too small, it loses the ability to control the scale. Meanwhile, if it is too close to 1, almost all the item and user will not be put into the similar set, which makes LDME fade to traditional CF model. According to the parameter decision, we finally set default parameter that $\alpha = 0.5$, $\theta = 0.5$ and $\delta = 0.5$.

3) EFFECT OF LDME

To validate the embedding vector of LDME, we feed the embedding results P^*, Q^* of LDME and traditional inner product embedding results into different CF models

(Item-based CF, User-based CF, and NeuCF). The results are shown in Table 3:

From the results, we can clearly see that our proposed model LDME's results $P*$, $Q*$ can improve the performance of traditional CF models (user-based CF, item-based CF, and NeuCF). Across all the different datasets, our embedding results can improve the performance about average 18.9% of HR, which proves the effectiveness of our proposed model.

VI. CONCLUSION

Many researchers of the recommendation area have focused on embedding user and item with considering only the relationships between them and employing inner product as the metric. But they always failed because of ignoring the representation limitation of inner product and the importance of the relationships between user and user, item and item. In this study, we provide insights into the metric used in embedding models: inner product and Euclidean distance. Moreover, we point out the weakness of inner product when applying for collaborative filtering, and the limitation of traditional metric learning.

Along with this line, we proposed a latent vector embedding model for collaborative filtering: Latent Dual Metric Embedding (LDME), which utilizes the dual-Euclidean distance in latent space, instead of inner product, to represent the different types of relationships (user-user, item-item, user-item) with a uniform framework. Specifically, we design an embedding loss function in LDME, which can measure the close and remote relationships between entities across different categories. Finally, extensive experiments are conducted to demonstrate that our proposed model can make an excellent trade-off performance compared with state-of-the-art baselines, especially with sparse and large-scale datasets.

REFERENCES

[1] R. de Groof and H. Xu, "Automatic topic discovery of online hospital reviews using an improved LDA with variational Gibbs sampling," in *Proc. IEEE Int. Conf. Big Data*, Dec. 2018, pp. 4022–4029.

[2] S. Zhao, T. Zhao, I. King, and M. R. Lyu, "Geo-teaser: Geo-temporal sequential embedding rank for point-of-interest recommendation," in *Proc. Int. Conf.*, 2017, pp. 153–162.

[3] L. Zheng, V. Noroozi, and P. S. Yu, "Joint deep modeling of users and items using reviews for recommendation," in *Proc. 10th ACM Int. Conf. Web Search Data Mining*, 2017, pp. 425–434.

[4] W.-S. Hwang, J. Parc, S.-W. Kim, J. Lee, and D. Lee, "'Told you i didn't like it': Exploiting uninteresting items for effective collaborative filtering," in *Proc. IEEE Int. Conf. Data Eng.*, May 2016, pp. 349–360.

[5] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proc. Int. Conf. World Wide Web*, 2001, pp. 285–295.

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

[7] P. Shen, X. Du, and C. Li, "Distributed semi-supervised metric learning," *IEEE Access*, vol. 4, pp. 8558–8571, 2017.

[8] H. J. Ye, D. C. Zhan, and Y. Jiang, "Fast generalization rates for distance metric learning," *Mach. Learn.*, pp. 1–29, Jun. 2018.

[9] 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.

[10] 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.

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

[12] 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.

[13] Q. Wang, J. Wan, and Y. Yuan, "Locality constraint distance metric learning for traffic congestion detection," *Pattern Recognit.*, vol. 75, pp. 272–281, Mar. 2017.

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

[15] Y. Liu, D. Pi, and L. Cui, "Metric learning combining with boosting for user distance measure in multiple social networks," *IEEE Access*, vol. 5, pp. 19342–19351, 2017.

[16] Y. Wang and H.-X. Li, "Burg matrix divergence-based hierarchical distance metric learning for binary classification," *IEEE Access*, vol. 5, pp. 3423–3430, 2017.

[17] J. Yu, M. Gao, W. Rong, Y. Song, and Q. Xiong, "A social recommender based on factorization and distance metric learning," *IEEE Access*, vol. 5, pp. 21557–21566, 2017.

[18] 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*. Geneva, Switzerland: International World Wide Web Conferences Steering Committee, 2017, pp. 173–182.

[19] T. Bai, J.-R. Wen, J. Zhang, and W. X. Zhao, "A neural collaborative filtering model with interaction-based neighborhood," in *Proc. ACM Conf. Inf. Knowl. Manage.*, 2017, pp. 1979–1982.

[20] C. Yang, L. Bai, C. Zhang, Q. Yuan, and J. Han, "Bridging collaborative filtering and semi-supervised learning: A neural approach for POI recommendation," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2017, pp. 1245–1254.

[21] J. Chen, H. Zhang, X. He, L. Nie, W. Liu, and T.-S. Chua, "Attentive collaborative filtering: Multimedia recommendation with item- and component-level attention," in *Proc. 40th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, 2017, pp. 335–344.

[22] J. Lian, F. Zhang, X. Xie, and G. Sun, "CCCFNet: A content-boosted collaborative filtering neural network for cross domain recommender systems," in *Proc. 26th Int. Conf. World Wide Web Companion*. Geneva, Switzerland: International World Wide Web Conferences Steering Committee, 2017, pp. 817–818.

[23] Y. Yang, Y. Xu, E. Wang, J. Han, and Z. Yu, "Improving existing collaborative filtering recommendations via serendipity-based algorithm," *IEEE Trans. Multimedia*, vol. 20, no. 7, pp. 1888–1900, Jul. 2017.

[24] H. Ying *et al.*, "Sequential recommender system based on hierarchical attention networks," in *Proc. 27th Int. Joint Conf. Artif. Intell. (IJCAI)*, 2018, pp. 3926–3932.

[25] F. Zhuang, D. Luo, N. J. Yuan, X. Xie, and Q. He, "Representation learning with pair-wise constraints for collaborative ranking," in *Proc. 10th ACM Int. Conf. Web Search Data Mining (WSDM)*, New York, NY, USA, 2017, pp. 567–575, doi: 10.1145/3018661.3018720.

[26] S. Seo, J. Huang, H. Yang, and Y. Liu, "Interpretable convolutional neural networks with dual local and global attention for review rating prediction," in *Proc. 11th ACM Conf. Recommender Syst.*, 2017, pp. 297–305.

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

[28] M. Luca and G. Zervas, "Fake it till you make it: Reputation, competition, and yelp review fraud," *Manage. Sci.*, vol. 62, no. 12, pp. 3412–3427, 2016.

[29] T. Hastie, J. Friedman, and R. Tibshirani, "The elements of statistical learning," *Technometrics*, vol. 45, no. 3, pp. 267–268, 2010.

[30] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *J. Mach. Learn. Res.*, vol. 12, no. 7, pp. 257–269, 2010.

[31] Y. Cheng, L. Jiao, Y. Tong, Z. Li, Y. Hu, and X. Cao, "Directional illumination estimation sets and multilevel matching metric for illumination-robust face recognition," *IEEE Access*, vol. 5, pp. 25835–25845, 2017.

[32] R. Pan *et al.*, "One-class collaborative filtering," in *Proc. 8th IEEE Int. Conf. Data Mining*, Dec. 2008, pp. 502–511.

[33] Y. Ning, Y. Shi, L. Hong, H. Rangwala, and N. Ramakrishnan, "A gradient-based adaptive learning framework for efficient personal recommendation," in *Proc. 11th ACM Conf. Recommender Syst.*, 2017, pp. 23–31.

[34] Y. Yang, Y. Xu, J. Han, E. Wang, W. Chen, and L. Yue, "Efficient traffic congestion estimation using multiple spatio-temporal properties," *Neurocomputing*, vol. 267, pp. 344–353, Dec. 2017.

YUANBO XU received the B.Sc. and M.S. degrees from the College of Computer Science and Technology, Jilin University, Changchun, China, where he is currently pursuing the Ph.D. degree with the Key Laboratory of Symbol Computation and Knowledge Engineering, Ministry of Education. His research interests include the applications of data mining, recommender systems, and mobile computing.

YONGJIAN YANG received the B.E. degree in automatization from the Jilin University of Technology, Changchun, China, in 1983, the M.E. degree in computer communication from the Beijing University of Post and Telecommunications, Beijing, China, in 1991, and the Ph.D. degree in software and theory of computer from Jilin University, Changchun, in 2005, where he is currently a Professor and a Ph.D. Supervisor. He is also the Director of the Key Laboratory under the Ministry of Information Industry, the Standing Director of the Communication Academy, and a member with the Computer Science Academy of Jilin Province. He participated in three projects of NSFC and 863 and was funded by the National Education Ministry for Doctoral Base Foundation. He has authored 12 projects of NSFC, key projects of the Ministry of Information Industry, Middle and Young Science and Technology Developing Funds, Jilin provincial programs, Shenzhen, Zhuhai, and Changchun. His research interests include the theory and software technology of network intelligence management, and the key technology research of wireless mobile communication and services.

JIAYU HAN received the B.Sc. and M.S. degrees from the College of Computer Science and Technology, Jilin University, Changchun, China, where she is currently pursuing the Ph.D. degree with the Key Laboratory of Symbol Computation and Knowledge Engineering, Ministry of Education. Her research interests include data mining, data fusion, recommender systems, and machine learning.

XIANG LI received the B.S. and M.S. degrees from the College of Computer Science and Technology, Jilin University. He is currently an Intermediate Engineer with the High Performance Center, Jilin University. His research interests include stream data processing and deep learning.

EN WANG received the B.E. degree in software engineering, the M.E. degree in computer science and technology, and the Ph.D. degree in computer science and technology from Jilin University, Changchun, in 2011, 2013, and 2016, respectively, where he is currently an Associate Professor with the Department of Computer Science and Technology. He is also a Visiting Scholar with the Department of Computer and Information Sciences, Temple University, Philadelphia. His current research focuses on the efficient utilization of network resources, scheduling and drop strategy in terms of buffer management, energy-efficient communication between human-carried devices, and mobile crowdsensing.