Action Space Learning for Heterogeneous User Behavior Prediction

Dongha Lee[1] , **Chanyoung Park**[2] , **Hyunjun Ju**[1] , **Junyoung Hwang**[1] and **Hwanjo Yu**[1,*]
[1]Pohang University of Science and Technology, Pohang, Republic of Korea
[2]University of Illinois at Urbana-Champaign, Urbana, IL, USA
{dongha.lee, hyunjunju, jyhwang, hwanjoyu}@postech.ac.kr, pcy1302@illinois.edu

Abstract

Users' behaviors observed in many web-based applications are usually heterogeneous, so modeling their behaviors considering the interplay among multiple types of actions is important. However, recent collaborative filtering (CF) methods based on a metric learning approach cannot learn multiple types of user actions, because they are developed for only a single type of user actions. This paper proposes a novel metric learning method, called METAS, to jointly model heterogeneous user behaviors. Specifically, it learns two distinct spaces: 1) *action space* which captures the relations among all observed and unobserved actions, and 2) *entity space* which captures high-level similarities among users and among items. Each action vector in the action space is computed using a non-linear function and its corresponding entity vectors in the entity space. In addition, METAS adopts an efficient triplet mining algorithm to effectively speed up the convergence of metric learning. Experimental results show that METAS outperforms the state-of-the-art methods in predicting users' heterogeneous actions, and its entity space represents the user-user and item-item similarities more clearly than the space trained by the other methods.

1 Introduction

In many web-based applications, it is challenging to understand and predict users' future behaviors because their actions are heterogeneous in nature. For example, e-commerce users perform several types of actions on products (e.g., click, add to cart, add to favorite, purchase), and DB users request heterogeneous DB operations on records (e.g., query, insert, delete, update) in a web database. Therefore, a successful behavior model should jointly consider the interplay among heterogeneous types of user actions rather than separately modeling user behaviors within each action type.

As alluded to above, user behavior logs are generally collected in *implicit* form in which only positive actions are observed. That is, negative actions and unobserved positive actions are mixed together, so a key challenge is to identify them based on collaborative filtering; this problem is referred to as one-class collaborative filtering (OCCF) [Pan *et al.*, 2008]. The conventional approach to OCCF is based on *score learning* such as matrix factorization (MF) and tensor factorization (TF), which aims to approximate the score of each matrix entry (or tensor entry) close to the observed one. The state-of-the-art TF method for user behavior modeling [Yin *et al.*, 2017] is also based on this approach. However, score learning methods have an intrinsic limitation in that user-user and item-item similarities are not captured correctly in their latent spaces, because they only learn the scores of user-item interactions that do not guarantee the similarities among users and among items.

To overcome this limitation, several recent work focus on a metric space where *triangle inequality* is satisfied, and they try to learn the distance rather than the score. Specifically, collaborative metric learning (CML) [Hsieh *et al.*, 2017] is the first work to apply metric learning to OCCF, in which the Euclidean distance between a user and an item vector in the metric space is considered inversely proportional to the strength of the user's preference on that item. CML can reflect user-user and item-item similarities as well as users' preferences on items in the metric space, because placing a user close to its interacted items means that the items are also getting closer to themselves. For this reason, CML shows better accuracy in predicting users' future behaviors than other score learning methods.

However, the existing CML approach is not suitable for modeling heterogeneous user behaviors for the following two reasons. First, it assumes that there exists only a single type of users actions in data, which makes it impossible to jointly consider the interplay among heterogeneous types of user actions. Second, its metric space learns the *naive similarity* among users and among items rather than *high-level similarity*; for example, two users who have completely different behavior patterns might be placed close to each other in the metric space because of a few items on which both users commonly do an action.

In this paper, we propose METAS, a novel METric learning method for Action Space where heterogeneous user behaviors are jointly embedded. To successfully incorporate the action type information into our metric space, we design a learning framework that optimizes two distinct spaces at the same time: 1) *Action space* where all possible actions are embedded – it learns the relations between observed-unobserved action pairs in order to model how likely each action is performed, and 2) *Entity space* where all entities (i.e., users,

*Corresponding author

items, and action types) are embedded – it learns high-level similarities among the entities.

In the action space, observed actions sharing a similar behavior pattern gather together while pushing unobserved actions away until they are beyond the predefined margin. Each action (in the action space) is obtained from a nonlinear embedding function which takes a user, an item, and an action type (in the entity space) as its input, to capture the interaction among the three entities. This non-linear function tunes the entity space to reflect the indirect and high-level similarities among users and among items. To effectively train both the spaces using sparse data, METAS adopts an hard triplet mining algorithm which efficiently selects triplets helpful for learning the spaces.

Our experiments demonstrate that METAS outperforms all other baselines. METAS more accurately predicts the top-N items on which users will perform actions of different types than the state-of-the-art method does, and also the entity space obtained by METAS turns out to reflect the similarities among entities more clearly compared to other methods. Furthermore, the proposed hard triplet mining algorithm significantly boosts the convergence.

2 Related Work

In this section, we briefly review two different approaches to one-class collaborative filtering (OCCF) for user behavior modeling: 1) score learning approach that learns the interaction among entities from a score function, and 2) metric learning approach that learns it from a distance function.

2.1 Implicit Tensor Factorization

Tensor factorization (TF) is widely used to analyze latent relationships among entities in multi-aspect data, which are naturally represented as high-order tensors. For heterogeneous user behavior modeling, it factorizes a third-order tensor of users' historical behaviors, composed of the three aspects: users, items, and action types.

In early work, RTF [Rendle *et al.*, 2009a] uses an optimization criterion to maximize the ranking AUC for TF, and BPR-PITF [Rendle *et al.*, 2009b; Rendle and Schmidt-Thieme, 2010] incorporates the concept of Bayesian personalized ranking (BPR) into TF, which learns the pairwise ranking so that an observed entry has larger score than an unobserved entry. BPR-PITF also introduces a pairwise interaction factorization (PIF) by formulating the score function for each tensor entry of a user i, an item j, and an action type k as

$$f(i,j,k) = \mathbf{u}_i \cdot \mathbf{v}_j + \mathbf{v}_j \cdot \mathbf{t}_k + \mathbf{t}_k \cdot \mathbf{u}_i, \tag{1}$$

and empirically shows that PIF outperforms the other factorization schemes such as Tucker decomposition (TD) and canonical decomposition (CD).

SPTF [Yin *et al.*, 2017] is the state-of-the-art TF method for OCCF. SPTF computes the score of each entry using PIF as well, but it designs its loss function based on probabilistic generative model rather than BPR framework. With the help of its elaborate sampling strategy for stochastic gradient descent (SGD), it achieves the best performance among all existing score learning methods developed for heterogeneous user behavior modeling.

2.2 Collaborative Metric Learning

Recently, [Hsieh *et al.*, 2017] criticizes that existing score learning methods for OCCF cannot correctly reflect user-user and item-item similarities in their latent spaces, and proposes collaborative metric learning (CML) which focuses on computing the distance between a user-item pair rather than a score. The goal of CML is to learn a metric space in which the Euclidean distance between a user and an item is considered inversely proportional to the strength of the user's preference on that item. In other words, the trained metric space satisfies that the distance between a user and its interacted item is smaller than the distance between the user and its non-interacted items. The loss function is described as follows:

$$\mathcal{L} = \sum_{(i,j)\in S} \sum_{(i,k)\notin S} [\|\mathbf{u}_i - \mathbf{v}_j\|_2^2 - \|\mathbf{u}_i - \mathbf{v}_k\|_2^2 + \alpha]_+, \tag{2}$$

where α is the margin size and S is the set of observed user-item interactions. As the metric space satisfies the *triangle inequality*, user-user and item-item similarities are captured together with the similarity in terms of user-item interactions; two users sharing many interacted items (or two items sharing many interacted users) also get closer in the metric space. For this reason, CML shows better accuracy in predicting users' behaviors than other score learning methods.

However, CML focuses on the user historical data containing only a single type of user actions, so it learns the distance between a user and an item regardless of interaction types. Thus, it cannot jointly model the interplay among multiple types of user actions. In addition, user-user and item-item similarities captured from user-item interactions are *naive*, which in turn makes the metric space hard to learn *high-level* similarities among users and among items. For example, two users who have completely different behavior patterns might be placed close to each other in the space because of a few items on which both users commonly do an action. This is because CML directly uses the Euclidean distance to measure the similarity of a user-item interaction pair. To overcome these limitations of CML, we need to newly design a distance function considering both heterogeneous action types and non-linear relations between entities at the same time.

3 METAS

In this section, we describe METAS which jointly learns heterogeneous types of user behaviors in a metric space. First, we formally describe the problem of heterogeneous behavior prediction. Then, we introduce a key concept of our proposed metric spaces where both entities and actions are embedded, and explain the details about it. Furthermore, we design a hard triplet mining algorithm to efficiently search triplets which produce non-zero gradients.

3.1 Problem Formulation

In this work, we focus on predicting users' future behaviors given implicit user historical data. Let $\mathcal{U} = \{u_1, u_2, \ldots, u_I\}$ be the set of all users, $\mathcal{V} = \{v_1, v_2, \ldots, v_J\}$ be the set of all items, $\mathcal{T} = \{t_1, t_2, \ldots, t_K\}$ be the set of all action types in a dataset, where I, J and K are the number of users, items and action types, respectively. They are components of each action and we call them *entity*. A single action is represented by a triple of (user index, item index, action type index), and

it could be observed or not yet observed. Note that the number of all possible actions is large (i.e., $I \times J \times K$) and observed actions account for only a small part of them. The entire dataset containing all observed actions is denoted by $\mathcal{X}$. Based on the heterogeneous user behavior dataset $\mathcal{X}$, the problem is defined as below.

Problem: (**Heterogeneous Behavior Prediction**) Given a target user u_i and an action type t_k, we aim to predict the top-N items on which u_i will perform the action t_k.

We use neither any explicit rating data nor any contextual (or auxiliary) information about entities. For the rest of this paper, in most cases, we use the term "action" to denote a single possible instance and "behavior" to denote a particular way of acting, but sometimes these two terms are used interchangeably when it is understandable from the context.

3.2 Metric Learning for Behavior Prediction

In order to incorporate heterogeneous types of user actions into the model, we first define a multi-layer representation space, which consists two hierarchical metric spaces (Figure 1). The key difference of our multi-layer space from other existing spaces is the decoupling of the space capturing the interactions among entities from the space capturing the similarities among entities. This structure facilitates METAS to learn more accurate spaces; each of them learns the interaction or the similarity for its own purpose, and the relation between the two spaces is also additionally trained.

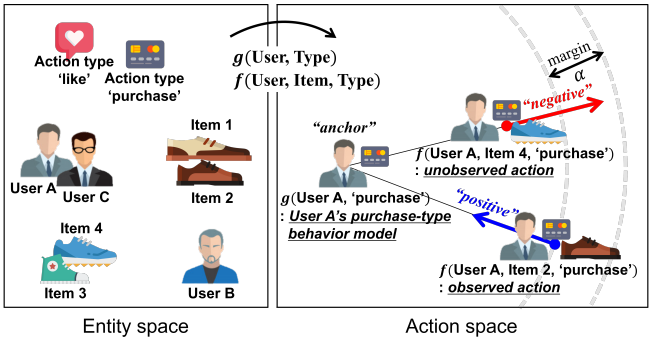

Figure 1: Multi-layer representation space for behavior prediction

The first space is *entity space* ($\in \mathbb{R}^{D_1}$) where all entities are embedded and their similarities are represented as the distances among entities. The vectors of user i, item j, and action type k in this space are denoted by bold letters (i.e., $\mathbf{u}_i$, $\mathbf{v}_j$, and $\mathbf{t}_k$, respectively). The second space is *action space* ($\in \mathbb{R}^{D_2}$) which embeds all possible actions and behavior models of all users. We use the notations $\mathbf{a}_{i,j,k}$ to denote the vector of a single action (i, j, k) and $\mathbf{b}_{i,k}$ to denote the behavior model vector of user i for action type k. Each behavior model $\mathbf{b}_{i,k}$ can be interpreted as a representative vector of its all relevant actions $\{\mathbf{a}_{i,j,k} | j = 1, ..., J\}$. In this sense, all observed actions gather together around its corresponding behavior model, whereas all unobserved actions are pushed away from its behavior model. Finally, the Euclidean distance between a target action $\mathbf{a}_{i,j,k}$ and its behavior model $\mathbf{b}_{i,k}$ in the space implies how likely user i do an action k on item j; the closer the distance between them is, the more likely the action will be performed. Figure 1 illustrates our representation space.

A key challenge here is how to obtain $\{\mathbf{a}_{i,j,k}\}$ and $\{\mathbf{b}_{i,k}\}$ so that they reflect the interactions among the entities. We note that introducing new variables for them requires a large number of parameters (i.e., $IJKD_2$ and IKD_2, respectively), which eventually results in poor scalability and overfitting problem. For this reason, METAS defines action vectors and behavior model vectors as a non-linear transformation of the corresponding entity vectors.

We employ two non-linear mapping functions f and g to embed all possible actions and behavior models of users into the action space. To be specific, $f(\mathbf{u}_i, \mathbf{v}_j, \mathbf{t}_k)$ computes the vector for the action (i, j, k), and $g(\mathbf{u}_i, \mathbf{t}_k)$ computes the representative vector for the k-th type actions of user i. Both functions take the vectors of user, item and action type embedded in the entity space as their input, and generate the vector of each action or user's behavior model that considers non-linear interactions among the entities.

Then, we can define a new distance function d that computes the Euclidean distance between a target action and its behavior model, i.e.,

$$\begin{aligned} d(i, j, k) &= \|\mathbf{b}_{i,k} - \mathbf{a}_{i,j,k}\|_2 \\ &= \|g(\mathbf{u}_i, \mathbf{t}_k) - f(\mathbf{u}_i, \mathbf{v}_j, \mathbf{t}_k)\|_2. \end{aligned} \tag{3}$$

To make this distance reflect how likely the action will be performed, we learn our action space in a way that an observed action gets closer to its behavior model than an unobserved action does. Namely, we create a triplet that consists of an *anchor* (a behavior model of user i and action type k), a *positive* (a relevant observed actions, i.e., $(i, j, k) \in \mathcal{X}$), and a *negative* (a relevant unobserved actions, i.e., $(i, j', k) \notin \mathcal{X}$), as illustrated in Figure 1. Our triplet loss function is formulated as follows:

$$\mathcal{L} = \sum_{(i,j,k) \in \mathcal{X}} \sum_{(i,j',k) \notin \mathcal{X}} [d(i, j, k) - d(i, j', k) + \alpha]_+, \tag{4}$$

where $[z]_+ = \max(z, 0)$ is the standard hinge loss, and α is the margin size. METAS uses l_2 distance rather than l_2^2 distance in each triplet loss, following [Wu *et al.*, 2017] that pointed out l_2 triplet loss is more stable and less likely to build a collapsed model compared to l_2^2 triplet loss.

Multi-layer perceptron (MLP) with dropout is employed as our non-linear mapping functions f and g due to its effectiveness in learning non-linear interactions among input features and the simple structure for training. There could be a variety of design choices for MLP networks depending on how to model the interactions between the entities. In this work, we construct the input of these functions as the concatenation of the three vectors (i.e., $[\mathbf{u}_i; \mathbf{v}_j; \mathbf{t}_k]$).

We also add a constraint which bounds all entities within a unit sphere to prevent entities from spreading too widely. We obtain the entities satisfying this constraint by normalizing each entity vector if its l_2-norm becomes larger than 1; i.e., $\mathbf{u}_* \leftarrow \mathbf{u}_* / \max(1, \|\mathbf{u}_*\|_2)$, $\mathbf{v}_* \leftarrow \mathbf{v}_* / \max(1, \|\mathbf{v}_*\|_2)$, and $\mathbf{t}_* \leftarrow \mathbf{t}_* / \max(1, \|\mathbf{t}_*\|_2)$, as done in CML.

3.3 Hard Triplet Mining

As the model gets closer to the convergence, it becomes more difficult to reduce the loss (Eq. (4)) because of the increasing

Algorithm 1: Hard triplet mining algorithm

Input: User behavior dataset $\mathcal{X}$, mini-batch size b, candidate itemset size c, the number of generated triplets per observed action at a time n
Output: Mini-batch of hard triplets B

$B \leftarrow \phi$
while $|B| < b$ **do**
  $u_i, t_k \leftarrow$ Sample a target user and action type
  $V^+ \leftarrow \{v_j | (i,j,k) \in \mathcal{X}\}$
  $V^- \leftarrow$ Sample c negative items from the set of all negative items (= $\{v_j | (i,j,k) \notin \mathcal{X}\}$)
  for $v_j \in V^+$ **do**
    $D^+[v_j] \leftarrow \|g(\mathbf{u}_i, \mathbf{t}_k) - f(\mathbf{u}_i, \mathbf{v}_j, \mathbf{t}_k)\|_2$
  for $v_j \in V^-$ **do**
    $D^-[v_j] \leftarrow \|g(\mathbf{u}_i, \mathbf{t}_k) - f(\mathbf{u}_i, \mathbf{v}_j, \mathbf{t}_k)\|_2$
  ▷ D^+, D^- : Map from each item to the distance
  $V^+_{sort}, D^+_{sort} \leftarrow$ Sort D^+ in descending order of values
  $V^-_{sort}, D^-_{sort} \leftarrow$ Sort D^- in descending order of values
  ▷ V_{sort}, D_{sort} : Array of sorted keys and values
  $ptr_{lb} \leftarrow 1$
  for $j = 1, ..., |V^+|$ **do**
    while $D^+_{sort}[j] + \alpha < D^-_{sort}[ptr_{lb}]$ ***and*** $ptr_{lb} < |V^-|$ **do**
      $ptr_{lb} \leftarrow ptr_{lb} + 1$
    $V^-_{select} \leftarrow$ Sample n items from $V^-_{sort}[ptr_{lb} : |V^-|]$
    for $v_{j'} \in |V^-_{select}|$ **do**
      $B \leftarrow B \cup \{((u_i, V^+_{sort}[j], t_k), (u_i, v_{j'}, t_k))\}$

number of observed-unobserved action pairs (i.e., $(i,j,k) \in \mathcal{X}, (i,j',k) \notin \mathcal{X}$) that have already satisfied the following margin condition,

$$d(i,j,k) + \alpha \leq d(i,j',k). \tag{5}$$

A naive triplet selection algorithm, which randomly samples an observed-unobserved action pair, cannot generate *hard triplets* that contribute to the training of the model parameters. This eventually leads to slow convergence and the waste of computing resources consumed for calculating the zero gradients.

In order to effectively train the model while saving the time and computing resources, we propose a hard triplet mining algorithm that efficiently searches triplets producing non-zero gradients. Checking whether a given triplet violates the inequality in Eq. (5) requires pre-computed action vectors, but it is infeasible to keep all possible action vectors (i.e., $f(\mathbf{u}_i, \mathbf{v}_j, \mathbf{t}_k)$) on the memory and compute all the distances among them. To tackle this challenge, our algorithm builds a *candidate set* of unobserved actions given a user and an action type, and some of them are selected and included in each triplet.

Figure 2 illustrates the toy example of our algorithm. First, given a user and an action type, it constructs 1) the set of observed actions (V^+) and 2) the set of unobserved actions (V^-) by sampling a small part of negative items on which the user has not perform the action. After sorting the two sets (V^+_{sort}, V^-_{sort}) in decreasing order of their distances, it starts to mine triplets from the observed action with the largest distance. As the unobserved actions are sorted in decreasing order, we can easily identify the range of unobserved actions that make the triplet "hard" ($V^-_{sort}[ptr_{lb} : |V^-|]$) by finding the farthest unobserved action violating the inequality ($V^-_{sort}[ptr_{lb}]$). This can be efficiently done by moving the *idx-lower-bound* pointer (ptr_{lb}) from begin to end just once per a target user and action type. Finally, we sample n unobserved actions from the identified range, and add the constructed hard triplets to our mini-batch. The detailed algorithm is presented in Algorithm 1.[1]

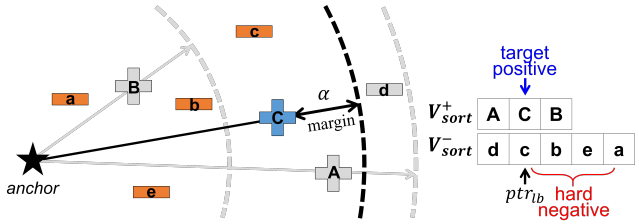

Figure 2: Efficient identification of unobserved actions violating the negative margin inequality; + and − represent observed and unobserved action, respectively.

4 Experiments

In this section, we present experimental results supporting that METAS outperforms other baselines in various aspects.

4.1 Datasets

For our experiments, we use two large-scale and real-world datasets: Tmall and Taobao. Tmall is the behavior dataset collected by Tmall of Alibaba, one of the biggest e-commerce platforms, and it is published by [Yin *et al.*, 2017]. Taobao is also users-commodities behavior data on Alibaba's mobile-commerce platforms, publicly available from Ali Mobile Recommendation Algorithm Competition. Both of them contain four types of user actions; click, add to favorite, add to cart, and purchase. Details about the datasets are presented in Table 1. It is worth noting that the distribution of observed actions with respect to behavior types is heavily skewed. Specifically, t_1 actions account for 83.06% of the entire observed actions in case of the Tmall dataset, and 88.75% in case of the Taobao dataset.

	Tmall	Taobao
# users	9,896	10,000
# items	548,999	2,876,947
# act. types	4	4
# actions	2,141,805 (19,219)	5,248,048 (21,559)
# t_1 actions	1,779,013 (9,796)	4,657,587 (9,957)
# t_2 actions	126,383 (2,615)	222,132 (3,054)
# t_3 actions	167,489 (4,503)	271,761 (5,334)
# t_4 actions	68,920 (2,305)	96,568 (3,214)

Table 1: Statistics of two real-world datasets. (Parentheses: the number of observed actions used for test and validation)

[1] To prevent any confusions between an array index and an entity index, we use the entity itself u_i, v_j, t_k instead of the entity index i, j, k to represent each entity in the algorithm.

4.2 Baselines

To evaluate the performance of METAS, we choose two state-of-the-art methods from two categories: score learning and metric learning.

- **SPTF** [Yin *et al.*, 2017]: The state-of-the-art TF method to model heterogeneous user behaviors. It shows the best accuracy in predicting users' future behaviors among all score learning methods, including BPR-PITF [Rendle and Schmidt-Thieme, 2010], BPTF [Xiong *et al.*, 2010], and RESCAL [Nickel *et al.*, 2011].
- **CML** [Hsieh *et al.*, 2017]: The state-of-the-art metric learning method developed for OCCF. As CML was originally developed to consider a single type of actions, we train a separate model for each action type. That is, each user or item is represented by multiple independent vectors, each of which corresponds to each action type.

Since SPTF and CML surpassed other implicit feedback-based baselines such as WRMF [Pan *et al.*, 2008; Hu *et al.*, 2008], BPR [Rendle *et al.*, 2009b] and WARP [Weston *et al.*, 2010], we do not further compare with them.

4.3 Evaluation Protocol and Metrics

We evaluate the prediction performance of METAS and other baselines based on *leave-one-out* protocol widely used in OCCF [He *et al.*, 2017; Rendle *et al.*, 2009b; He and McAuley, 2016b; Xue *et al.*, 2017; He and McAuley, 2016a; He *et al.*, 2016]. For each user, we leave out a single observed action per action type for testing, and use the rest for training. In our experiments, we leave out an additional observed action for a validation set. Basically, we compute how close an observed action put aside for testing is located to its behavior model compared to other unobserved actions.

As it is time consuming to rank all the triples, for each (user, action type) pair we sample 100 items on which the user has not performed the actions (i.e., $\{(i, j', k)|(i, j', k) \notin \mathcal{X}\}$), and compute the distance function (Eq. (3)) for the test triple and all the sampled triples [He *et al.*, 2017; Xue *et al.*, 2017; Xue *et al.*, 2017]. After forming a ranked list of the distances, we compute several ranking metrics: 1) hit ratio H@N simply measures whether the target action is present in the top-N list, and 2) normalized discounted cumulative gain (NDCG) N@N assigns higher scores to hits at upper ranks in the top-N list.

4.4 Experimental Setting

We implement METAS and CML using Tensorflow [Abadi *et al.*, 2016] to run on GPU, and utilize the JAVA source code of SPTF released by [Yin *et al.*, 2017]. To train the models, we use the Adam optimizer [Kingma and Ba, 2014] supported by Tensorflow. The MLP networks of METAS are equipped with a single layer, 50% dropout probability, and *ReLU* activation. For each dataset, we tune the hyper-parameters by using a grid search and use the optimal values that show the best H@10 on the validation set. We set the margin size $\alpha = 2$ and the mini-batch size (i.e., the number of triplets in a mini-batch) $b = 200$.[2] We set $D_1 = D_2 = D$ to reduce the model complexity, and investigate the performance changes with respect to the dimension size D with a range of $\{50, 100, 150, 200, 250\}$. We only report the results of $D = 250$ for Tmall and $D = 100$ for Taobao[3], as we observe the same trend across the dimension sizes.

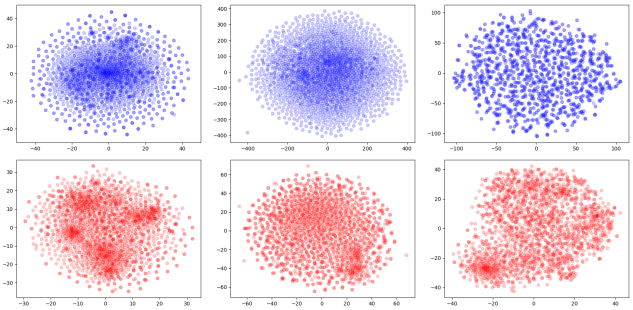

Figure 3: Users (Top) and items (Bottom) in the representation spaces from SPTF (Left), CML (Center), and METAS (Right).

4.5 Experimental Results

Prediction Accuracy

We first compare the prediction accuracy of METAS and other methods in terms of various ranking metrics (Table 2). For both the datasets, METAS achieves the best performance among all the baselines, and specifically, shows the improvement in terms of H@10 up to 20% compared to the state-of-the-art method (i.e., SPTF).

From the point of view of modeling heterogeneous user behaviors, CML shows poor performance in predicting users' behaviors of the action types with very few observed actions (i.e., t_2, t_3, and t_4). Because CML learns distinct vectors of users and items for each action type, a small number of observed actions related to t_2, t_3, and t_4 is insufficient for training the entire users and items. On the other hand, SPTF shows relatively good performance on those action types because it jointly learns multiple types of user actions; user and item vectors are shared and jointly trained across the action types, and the observed actions of t_1 type are helpful to learn and predict the other types of actions. However, we observe that the performance of SPTF on action type t_1 is worse than that of CML, whereas SPTF considerably outperforms CML on the rest of the action types. This implies that the observed actions in the majority type (i.e., t_1), are used to help improve the performance on the minor types (i.e., t_2, t_3, and t_4), at the expense of the performance on the majority class itself.

Unlike SPTF, METAS successfully learns the interplay among heterogeneous types of user actions without compromising the performance on the action type t_1. In terms of hit ratio, METAS outperforms all the other methods on all types of actions. In terms of NDCG, METAS shows slightly worse performance than SPTF on the action types t_2, t_3, and t_4, but it is still the best and the most powerful method considering all types of actions in total.

Metric Visualization

We visualize users and items by using t-SNE [Maaten and Hinton, 2008] to investigate how well each representation space reflects user-user and item-item similarities. We randomly sample 2000 users and 2000 items, and use the same sets for all methods.

[2] We empirically found that the performances are hardly affected by these hyper-parameters.

[3] We used the smaller dimension size for Taobao than that for Tmall, due to our limited GPU memory.

Datasets		Tmall					Taobao				
Method	Metric	Total	Type1	Type2	Type3	Type4	Total	Type1	Type2	Type3	Type4
SPTF	H@10	0.6692	0.3829	0.9728	0.9771	0.9397	0.6662	0.3806	0.8939	0.9280	0.8998
	H@20	0.7219	0.4845	0.9740	0.9787	0.9432	0.7059	0.4543	0.9011	0.9411	0.9095
	N@10	0.5997	0.2797	**0.9289**	**0.9479**	**0.9057**	0.5747	0.2939	**0.7868**	**0.8305**	**0.8189**
	N@20	0.6129	0.3052	**0.9292**	**0.9483**	**0.9066**	0.5848	0.3124	**0.7886**	**0.8339**	**0.8214**
CML	H@10	0.2814	0.4910	0.0730	0.0762	0.0278	0.3099	0.5327	0.1473	0.1264	0.0790
	H@20	0.3586	0.6176	0.1155	0.1053	0.0282	0.3598	0.6391	0.1483	0.1282	0.0793
	N@10	0.1865	0.3200	0.0506	0.0578	0.0249	0.2172	0.3597	0.1189	0.1019	0.0608
	N@20	0.2059	0.3518	0.0613	0.0650	0.0250	0.2298	0.3865	0.1191	0.1024	0.0609
METAS	H@10	**0.7523**	**0.5253**	**0.9908**	**0.9905**	**0.9809**	**0.7987**	**0.6185**	**0.9434**	**0.9700**	**0.9356**
	H@20	**0.8142**	**0.6417**	**0.9962**	**0.9944**	**0.9892**	**0.8695**	**0.7298**	**0.9905**	**0.9944**	**0.9798**
	N@10	**0.6298**	**0.3619**	0.8951	0.9264	0.8883	**0.6031**	**0.4600**	0.6939	0.7648	0.6918
	N@20	**0.6455**	**0.3912**	0.8964	0.9274	0.8905	**0.6211**	**0.4882**	0.7062	0.7711	0.7033
Improv.	H@10	12.42%	6.99%	1.85%	1.37%	4.38%	19.90%	16.10%	5.53%	4.53%	3.98%
	H@20	12.79%	3.90%	2.28%	1.60%	4.88%	23.18%	14.19%	9.92%	5.66%	7.73%
	N@10	5.02%	13.09%	-3.64%	-2.27%	-1.92%	4.94%	27.90%	-11.81%	-7.91%	-15.52%
	N@20	3.79%	11.20%	-3.53%	-2.20%	-1.78%	6.22%	26.32%	-10.45%	-7.53%	-14.38%

Table 2: Test performance of METAS and other methods. The best performing methods is boldfaced. (*Improv.* denotes the improvement of METAS over the best baseline method.)

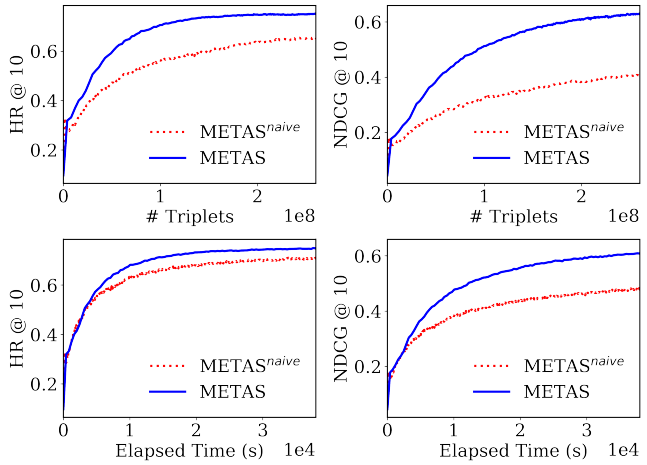

Figure 4: The convergence speed of METAS in terms of the number of triplets (Top) and the elapsed time (Bottom)

Figure 3 shows that users and items in the space obtained from CML and SPTF gather together around a certain centric point, which is analogous to Gaussian distribution. On the other hand, in the entity space learned by METAS, users and items are relatively scattered throughout the space while forming multiple small colonies. We argue that these colonies can be interpreted as fine-grained clusters that share similar behavior pattern. More precisely, our entity space is indirectly learned from the relationship among actions in the action space, which captures the non-linear interactions among users, items, and action types; thus two users (or items) need to share not only common items (or users), but also the action types in order to be similar to each other in the entity space (high-level similarity). The above experiment demonstrates that METAS successfully learned the high-level similarity among users and among items by decoupling the two representation spaces of different purposes as well as taking into account heterogeneous types of user actions.

Convergence Speed

To investigate the effectiveness of our hard triplet mining algorithm, we compare the convergence speed of METAS with METAS^{naive} that randomly selects triplets for training. In Algorithm 1, we set the candidate itemset size c to the size of mini-batch, and $n = 3$. We plot the convergence curves of METAS and METAS^{naive} in terms of 1) the number of triplets and 2) the elapsed time.

In Figure 4, our hard triplet mining algorithm considerably boosts the convergence of METAS by selecting only the hard triplets producing non-zero gradients. On the contrary, most triplets selected by METAS^{naive} are useless triplets, which do not contribute to the training of the model parameters, thus its convergence is much slower. In terms of the elapsed time, the improvement becomes slightly less significant, because mining the hard triplets requires relatively higher overhead cost compared with randomly selecting the triplets: obtaining the action vectors by the functions f and g, computing their distances, and sorting them. Despite the above overhead costs, the hard triplet mining eventually shortens the training time and makes METAS achieve higher accuracy.

5 Conclusion

This paper proposes METAS, a novel metric learning method to embed the heterogeneous types of user behaviors in the action space. METAS trains the two distinct metric spaces and the non-linear mapping functions from one to the another; all entities are embedded in the entity space, and all possible actions are embedded in the action space via the mapping function whose input consists of the combination of entities. METAS also adopts a hard triplet mining algorithm to efficiently select triplets producing non-zero gradients. Our evaluation results on two real-world datasets demonstrate that 1) METAS is more accurate in predicting users' future behaviors compared to other baselines, 2) the metric space obtained by METAS reflects high-level similarities among users and among items, and 3) our hard triplet mining algorithm significantly boosts the training process of METAS.

Acknowledgements

This research was supported by the NRF grant funded by the MSIT: (No. 2016R1E1A1A01942642) and (No. 2017M3C4A7063570), the IITP grant funded by the MSIT: (No. 2018-0-00584) and (IITP-2019-2011-1-00783).

References

[Abadi *et al.*, 2016] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: a system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016.

[Bordes *et al.*, 2011] Antoine Bordes, Jason Weston, Ronan Collobert, Yoshua Bengio, et al. Learning structured embeddings of knowledge bases. In *AAAI*, volume 6, page 6, 2011.

[Chen *et al.*, 2018] Hongxu Chen, Hongzhi Yin, Weiqing Wang, Hao Wang, Quoc Viet Hung Nguyen, and Xue Li. Pme: projected metric embedding on heterogeneous networks for link prediction. In *KDD*, pages 1177–1186, 2018.

[He and McAuley, 2016a] Ruining He and Julian McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *WWW*, pages 507–517, 2016.

[He and McAuley, 2016b] Ruining He and Julian McAuley. Vbpr: Visual bayesian personalized ranking from implicit feedback. In *AAAI*, pages 144–150, 2016.

[He *et al.*, 2016] Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. Fast matrix factorization for online recommendation with implicit feedback. In *SIGIR*, pages 549–558, 2016.

[He *et al.*, 2017] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *WWW*, pages 173–182, 2017.

[Hoffer and Ailon, 2015] Elad Hoffer and Nir Ailon. Deep metric learning using triplet network. In *International Workshop on Similarity-Based Pattern Recognition*, pages 84–92, 2015.

[Hsieh *et al.*, 2017] Cheng-Kang Hsieh, Longqi Yang, Yin Cui, Tsung-Yi Lin, Serge Belongie, and Deborah Estrin. Collaborative metric learning. In *WWW*, pages 193–201, 2017.

[Hu *et al.*, 2008] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *ICDM*, pages 263–272, 2008.

[Kingma and Ba, 2014] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[Lin *et al.*, 2015] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*, volume 15, pages 2181–2187, 2015.

[Maaten and Hinton, 2008] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *JMLR*, 9(Nov):2579–2605, 2008.

[Nickel *et al.*, 2011] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. In *ICML*, volume 11, pages 809–816, 2011.

[Pan *et al.*, 2008] Rong Pan, Yunhong Zhou, Bin Cao, Nathan N Liu, Rajan Lukose, Martin Scholz, and Qiang Yang. One-class collaborative filtering. In *ICDM*, pages 502–511, 2008.

[Rendle and Schmidt-Thieme, 2010] Steffen Rendle and Lars Schmidt-Thieme. Pairwise interaction tensor factorization for personalized tag recommendation. In *WSDM*, pages 81–90, 2010.

[Rendle *et al.*, 2009a] Steffen Rendle, Leandro Balby Marinho, Alexandros Nanopoulos, and Lars Schmidt-Thieme. Learning optimal ranking with tensor factorization for tag recommendation. In *KDD*, pages 727–736, 2009.

[Rendle *et al.*, 2009b] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *UAI*, pages 452–461, 2009.

[Schroff *et al.*, 2015] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *CVPR*, pages 815–823, 2015.

[Tay *et al.*, 2018] Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. Latent relational metric learning via memory-based attention for collaborative ranking. In *WWW*, pages 729–739, 2018.

[Weinberger and Saul, 2009] Kilian Q Weinberger and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. *JMLR*, 10(Feb):207–244, 2009.

[Weston *et al.*, 2010] Jason Weston, Samy Bengio, and Nicolas Usunier. Large scale image annotation: learning to rank with joint word-image embeddings. *Machine learning*, 81(1):21–35, 2010.

[Wu *et al.*, 2017] Chao-Yuan Wu, R Manmatha, Alexander J Smola, and Philipp Krähenbühl. Sampling matters in deep embedding learning. In *ICCV*, 2017.

[Xiong *et al.*, 2010] Liang Xiong, Xi Chen, Tzu-Kuo Huang, Jeff Schneider, and Jaime G Carbonell. Temporal collaborative filtering with bayesian probabilistic tensor factorization. In *SDM*, pages 211–222, 2010.

[Xue *et al.*, 2017] Hong-Jian Xue, Xinyu Dai, Jianbing Zhang, Shujian Huang, and Jiajun Chen. Deep matrix factorization models for recommender systems. In *IJCAI*, pages 3203–3209, 2017.

[Yin *et al.*, 2017] H. Yin, H. Chen, X. Sun, H. Wang, Y. Wang, and Q. V. H. Nguyen. Sptf: A scalable probabilistic tensor factorization model for semantic-aware behavior prediction. In *ICDM*, pages 585–594, 2017.