SPTF: A Scalable Probabilistic Tensor Factorization Model for Semantic-Aware Behavior Prediction

Hongzhi Yin[1] Hongxu Chen[1] Xiaoshuai Sun[2] Hao Wang[3] Yang Wang[4] Quoc Viet Hung Nguyen[5]
[1]The University of Queensland, School of Information Technology and Electrical Engineering
db.hongzhi@gmail.com hongxu.chen@uq.edu.au
[2]School of Computer Science and Technology, Harbin Institute of Technology, xiaoshuaisun.hit@gmail.com
[3]360 Search Lab, Qihoo 360 Inc., cashenry@126.com
[4] School of Computer Science and Engineering, The University of New South Wales, wangy@cse.unsw.edu.au
[5] School of Information and Communication Technology, Griffith University, quocviethung.nguyen@griffith.edu.au

***Abstract*—With the rapid rise of various e-commerce and social network platforms, users are generating large amounts of heterogenous behavior data, such as purchase history, adding-to-favorite, adding-to-cart and click activities, and this kind of user behavior data is usually binary, only reflecting a user's action or inaction (i.e., implicit feedback data). Tensor factorization is a promising means of modeling heterogenous user behaviors by distinguishing different behavior types. However, ambiguity arises in the interpretation of the unobserved user behavior records that mix both real negative examples and potential positive examples. Existing tensor factorization models either ignore unobserved examples or treat all of them as negative examples, leading to either poor prediction performance or huge computation cost. In addition, the distribution of positive examples w.r.t. behavior types is heavily skewed. Existing tensor factorization models would bias towards the type of behaviors with a large number of positive examples.**

In this paper, we propose a scalable probabilistic tensor factorization model (SPTF) for heterogenous behavior data, and develop a novel negative sampling technique to optimize SPTF by leveraging both observed and unobserved examples with much lower computational costs and higher modeling accuracy. To overcome the issue of the heavy skewness of the behavior data distribution, we propose a novel adaptive ranking-based positive sampling approach to speed up the model convergence and improve the prediction accuracy for sparse behavior types. Our proposed model optimization techniques enable SPTF to be scalable to large-scale behavior datasets. Extensive experiments have been conducted on a large-scale e-commerce dataset, and the experimental results show superiority of our proposed SPTF model in terms of prediction accuracy and scalability.

I. Introduction

As e-commerce and social network platforms (e.g., Amazon.com, Ebay.com and Tmall.com) are growing, an important challenge is helping customers sort through a large variety of available products to easily find the ones they will prefer and purchase. One essential means is to learn users' preferences and produce personalized recommendation by exploiting their historical behavior records [12], [31], [33], [32], [34], and the core techniques are collaborative filtering (CF) [1].

User historical behaviors can be divided into two types: *explicit feedback* and *implicit feedback*. The explicit feedback includes explicit input by users regarding their interests in items, which is often expressed in rating scores. For example, Netflix collects star ratings for movies. However, explicit feedback is not always available. In many more situations, implicit ratings of users expressed by users' behaviors such as click or not-click and purchase or not-purchase are more common and easier to obtain. A drawback of implicit feedback is that only positive feedback (e.g., purchase, click and add to cart) is observed. The collaborative filtering with only positive examples or observations was called One-Class Collaborative Filtering (OCCF) [16].

Almost all recent work on OCCF has been focused on developing new matrix factorization (MF) models to predict users' preferences from the implicit feedback data including both point-wise models (e.g., weighted matrix factorization) [7], [16] and pair-wise ranking models [22], [17] (e.g., BPR-based matrix factorization). However, these existing works focus on the homogenous user behaviors and do not consider the variety and heterogeneity of user behaviors in the real world. For example, there are at least four types of user behaviors in the typical e-commerce platforms (e.g., Amazon or Tmall): click, add-to-cart, add-to-favorite and purchase. Different types of user behaviors imply different semantics and user intentions. The way people interact with the items is critical in understanding user intents and modeling user interests. In a word, what is important is not just what users interact with, but how they interact with them as well. Moreover, user implicit feedback data on a single behavior type (e.g., purchase) is extremely sparse, therefore it is essential to collectively exploit the rich heterogenous behavior data to enhance each type of behavior prediction, especially prediction for sparse behavior types (e.g., purchase prediction).

In this paper, we propose a scalable probabilistic tensor factorization model (SPTF) for semantic-aware behavior modeling and prediction. In our SPTF, each user behavior record is represented by a triple instead of a tuple, i.e., (user, behavior-type, item), and we distinguish different behavior types as they convey different semantics. SPTF learns a latent vector representation for each user, item and behavior type. Tensors are useful ways of multiple relationships between objects, and

2374-8486/17 $31.00 © 2017 IEEE
DOI 10.1109/ICDM.2017.68

tensor factorization is seen as an essential means of predicting their possible future relationships. However, existing tensor factorization models such as BPTF [30] and RESCAL [15] ignore all unobserved examples, just as the matrix factorization models [8], [25] have done on the explicit feedback datasets. Due to the extreme sparsity of positive examples, it is difficult for these tensor factorization models to accurately model and predict semantic-aware user behaviors.

In light of this, we propose a novel negative-sampling technique to optimize the SPTF model to precisely predict semantic-aware user behaviors. The unobserved examples, e.g., a user has not purchased an item yet, are a mixture of real negative examples (e.g., the user is not interested in buying the item) and potential positive examples (e.g., the user might want to buy the item in the future). It is extremely challenging to identify real representative negative examples. One widely adopted solution is to treat all unobserved examples as negative examples, and this solution empirically works well in weighted MF and BPR-based factorization models on small datasets [7], [26], [36], [24]. But, this solution limits the prediction accuracy because some of the missing data might be positive [16]. Moreover, this solution cannot apply to large-scale datasets due to the huge computational cost. For example, BPR-PITF [24] combines all observed and unobserved examples to construct a training dataset with pairwise constraints, and the size of the training data in our problem would be $\mathcal{O}(|\mathcal{D}^+||\mathcal{V}|)$ where $\mathcal{D}^+$ is the set of observed examples (i.e., triples) and $\mathcal{V}$ is the set of items. To avoid the huge computation cost, we propose a novel bidirectional popularity-biased negative-sampling approach to sample very few informative unobserved examples for each observed example and correct the bias of treating all them as negative examples.

Another challenge is that user behaviors are heterogenous in our problem, and the distribution of positive examples w.r.t. behavior types is *heavily skewed*. For example, in our collected T-mall dataset, click behaviors take up 86.58% of the observed examples, while the proportions for add-to-favorite, add-to-cart and purchase behaviors are 4.93%, 5.91% and 2.57%, respectively. If we uniformly draw a training case and perform stochastic gradient descent on the drawn case, just as done in most stochastic gradient descent-based optimization methods [24], most of sampled positive examples would be associated with click behaviours and the trained model would heavily bias towards click behaviors. To overcome the heavy skewness of the heterogenous behavior data distribution, we propose a novel adaptive ranking-based sampling approach to draw positive examples. The basic idea is that positive examples at a lower rank should have a higher probability to be sampled, as this kind of positive examples are more informative and helpful to correct the current model parameters.

Our contributions are summarized as follows.

- We propose a new problem - Semantic-Aware Behavior Prediction that aims to predict top-n items on which the target user will perform a type-specific action.
- We develop a scalable probabilistic tensor factorization model (SPTF) to model semantic-aware heterogenous behavior data, and propose a novel bidirectional popularity-biased negative sampling technique to optimize SPTF with much higher modeling accuracy.
- To overcome the heavy skewness of the heterogenous behavior data distribution, we propose a novel adaptive ranking-based sampling approach to draw training cases.

II. Semantic-Aware User Behavior Modeling

In this section, we first formulate our problem of semantic-aware user behavior prediction, and then introduce a probabilistic generative model to account for semantic-aware heterogenous user behaviors.

A. Problem Formulation

Before proceeding, let us first define our mathematical notations for the ease of presentation. Let $\mathcal{U} = \{u_1, \ldots, u_I\}$ be the set of all users, $\mathcal{V} = \{v_1, \ldots, v_J\}$ be the set of all items, and $\mathcal{T} = \{t_1, \ldots, t_K\}$ be the set of all behavior types in a dataset. I, J and K are used to denote the numbers of users, items and behavior types, respectively. We use a triple to represent a semantic-aware behavior record, i.e., $x_{ikj} = (u_i,\ t_k,\ v_j)$. We model each possible triple $x_{ikj} = (u_i,\ t_k,\ v_j)$ over this set of users, items and behavior types as a binary random variable $y_{ikj} \in \{0, 1\}$ that indicates its existence (i.e., being observed) or not. All possible triples in $\mathcal{U} \times \mathcal{T} \times \mathcal{V}$ can be grouped naturally in a third-order tensor $\boldsymbol{Y} \in \{0, 1\}^{I \times K \times J}$ whose entries are set as in Equation (1). We use $\mathcal{D}^+$ to denote the set of observed triples and $\mathcal{D}^-$ to denote the set of unobserved triples. Here, following the common symbolic notation, upper case bold letters denote matrices or tensors, and lower case bold letters denote column vectors.

$$y_{ikj} = \begin{cases} 1, & \text{if the triple } (u_i, t_k, v_j) \text{ is observed} \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Based on the semantic-aware heterogenous user behavior dataset $\boldsymbol{Y}$, we define our problem as:

Problem 1: **(Semantic-Aware 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 action t_k.

B. Model Description

To model semantic-aware heterogenous behaviors, we propose a novel probabilistic tensor factorization model in which each user, item and behavior-type are assigned with a D-dimensional latent factor vector, denoted as $\boldsymbol{u}_i$, $\boldsymbol{t}_k$ and $\boldsymbol{v}_j$, respectively. We use the notation Θ to denote the parameter set in our model, i.e., $\Theta = \{\boldsymbol{u}_i, \boldsymbol{t}_k, \boldsymbol{v}_j | u \in \mathcal{U}, t \in \mathcal{T}, v \in \mathcal{V}\}$. In our model, we assume that all y_{ikj} are conditionally independent given latent vectors of users, items and behavior types. Our model predicts the existence of a triple x_{ikj} via a score function $f(x_{ikj}; \Theta)$ which represents the model's confidence that a triple exists (i.e., a specific user behavior happens) given the parameters Θ. The conditional independence assumption of our model allows the probability model to be written as follows:

$$P(\boldsymbol{Y}|\Theta) = \prod_{i=1}^{I} \prod_{k=1}^{K} \prod_{j=1}^{J} Ber(y_{ikj} | \sigma(f(x_{ikj}; \Theta))) \tag{2}$$

$$P(\boldsymbol{U},\boldsymbol{T},\boldsymbol{V}|\boldsymbol{Y},\rho_U^2,\rho_V^2,\rho_T^2) \propto P(\boldsymbol{Y}|\boldsymbol{U},\boldsymbol{T},\boldsymbol{V})P(\boldsymbol{U}|\boldsymbol{0},\rho_U^2)P(\boldsymbol{T}|\boldsymbol{0},\rho_T^2)P(\boldsymbol{V}|\boldsymbol{0},\rho_V^2)$$
$$=\prod_{i=1}^{I}\prod_{k=1}^{K}\prod_{j=1}^{J} Ber(y_{ikj}|\sigma(f(x_{ikj};\Theta)))\prod_{i=1}^{I}\mathcal{N}(\boldsymbol{u}_i|\boldsymbol{0},\rho_U^2\boldsymbol{I})\prod_{k=1}^{K}\mathcal{N}(\boldsymbol{t}_k|\boldsymbol{0},\rho_T^2\boldsymbol{I})\prod_{j=1}^{J}\mathcal{N}(\boldsymbol{v}_j|\boldsymbol{0},\rho_V^2\boldsymbol{I}) \tag{5}$$

$$\mathcal{L} = -\sum_{x_{ikj}\in\mathcal{D}^+}\log\sigma(f(x_{ikj};\Theta)) - \sum_{x_{ikj}\in\mathcal{D}^-}\log(1-\sigma(f(x_{ikj};\Theta))) + \frac{\lambda_U}{2}\sum_{i=1}^{I}\|\boldsymbol{u}_i\|^2 + \frac{\lambda_T}{2}\sum_{k=1}^{K}\|\boldsymbol{t}_k\|^2 + \frac{\lambda_V}{2}\sum_{j=1}^{J}\|\boldsymbol{v}_j\|^2 \tag{6}$$

$$\mathcal{O} = -\sum_{x_{ikj}\in\mathcal{D}^+}\Big(\log\sigma(f(x_{ikj};\Theta)) + \sum_{n=1}^{N}E_{v_n\sim P(v_n)}\log(1-\sigma(f(x_{ikn};\Theta))) + \sum_{n=1}^{N}E_{u_n\sim P(u_n)}\log(1-\sigma(f(x_{nkj};\Theta)))\Big)$$
$$+\frac{\lambda_U}{2}\sum_{i=1}^{I}\|\boldsymbol{u}_i\|^2 + \frac{\lambda_T}{2}\sum_{k=1}^{K}\|\boldsymbol{t}_k\|^2 + \frac{\lambda_V}{2}\sum_{j=1}^{J}\|\boldsymbol{v}_j\|^2 \tag{7}$$

where $\sigma(x) = 1/(1+e^{-x})$ is the sigmoid (logistic) function, and $Ber(y|p)$ is the Bernoulli distribution, defined as follows.

$$Ber(y|p) = \begin{cases} p, & \text{if } y = 1 \\ 1-p, & \text{if } y = 0. \end{cases} \tag{3}$$

There are many tensor factorization methods such as Tucker Decomposition (TD), Canonical Decomposition (CD) and Pairwise Interaction Factorization (PIF) [24] that can be used to implement the score function $f(x_{ikj};\Theta)$ in our SPTF model. Rendle et al. [24] empirically compared all these tensor factorization methods based on BPR optimization framework [22] and found that PIF consistently achieves the best performance with lower time cost of model training. Therefore, we adopt the pairwise interaction factorization to implement $f(x_{ikj};\Theta)$ in our SPTF model, as follows.

$$f(x_{ikj};\Theta) = \sum_{d=1}^{D} u_{i,d}\cdot t_{k,d} + \sum_{d=1}^{D} u_{i,d}\cdot v_{j,d} + \sum_{d=1}^{D} t_{k,d}\cdot v_{j,d} \tag{4}$$

Inspired by the success of probabilistic matrix factorization [25], we introduce the Gaussian priors over parameters Θ. Finally, the probabilistic generative process of our model is listed as follows:

1) For each $i \in (1\dots I)$, generate $\boldsymbol{u}_i \sim \mathcal{N}(\boldsymbol{0},\rho_U^2\boldsymbol{I})$.
2) For each $k \in (1\dots K)$, generate $\boldsymbol{t}_k \sim \mathcal{N}(\boldsymbol{0},\rho_T^2\boldsymbol{I})$.
3) For each $j \in (1\dots J)$, generate $\boldsymbol{v}_j \sim \mathcal{N}(\boldsymbol{0},\rho_V^2\boldsymbol{I})$.
4) For each triple x_{ikj}, generate $y_{ikj} \sim Ber(y_{ikj}|\sigma(f(x_{ikj};\Theta)))$.

Based on the above probabilistic generative process of our model, through a simple Bayesian inference, the posterior distribution of the latent vectors of users, items and behavior-types is formulated as in Equation (5), where $\boldsymbol{U},\boldsymbol{T},\boldsymbol{V}$ are three matrices that are formed by $\boldsymbol{u}_i$, $\boldsymbol{t}_k$ and $\boldsymbol{v}_j$ as their column vectors, respectively. Our goal is to maximize the likelihood in Equation (5), which is equivalent to minimizing the negative log likelihood in Equation (6) where the regularization parameters $\lambda_U = \rho_U^{-2}$, $\lambda_T = \rho_T^{-2}$ and $\lambda_V = \rho_V^{-2}$.

C. Model Optimization

Directly maximizing Eqn. (6) is computationally expensive, as the number of unobserved examples is cubic to the number of users or items. Besides, not all unobserved examples are real negative examples. Inspired by the negative sampling techniques [13], [28], [29], instead of using all unobserved examples, we select some most likely negative examples for the model optimization.

In the negative sampling-based optimization method, the sampling probability for negative examples is critical to the model performance. Given a positive example (u_i,t_k,v_j), all existing negative sampling techniques [13], [22], [21], [28], [29] construct negative examples from only one side, i.e., the negative sampling is unidirectional. Specifically, they fix user u_i and sample some negative items v_n (i.e., items with which u_i has never interacted via action t_k) according to a noisy distribution $P(v_n)$, and treat (u_i,t_k,v_n) as the negative examples. If we construct negative examples only from the user perspective, as done in Bayesian personalized ranking (BRP) [22], [21], [24], we cannot accurately learn latent vector representation for item v_j, because only positive users (i.e., users who have performed action t_k on v_j) are considered and thus the learned latent vector $\boldsymbol{v}_j$ could not discriminate between its positive users and negative users. Thus, constructing negative examples from one side is insufficient and would lead to the slow convergence and low modeling accuracy.

Bidirectional Negative Sampling Strategy. Therefore, we introduce a bidirectional sampling strategy to construct negative examples from both user and item sides. Specifically, given a positive example (u_i,t_k,v_j), we first fix u_i and t_k, and sample negative items v_n to form negative examples (u_i,t_k,v_n) from the user perspective. Then, we fix v_j and t_k, and sample negative users u_n to form negative examples (u_n,t_k,v_j) from the item perspective. Thus, how to sample negative items v_n and negative users u_n is critical. One naive idea is to use the uniform random sampling method to generate negative items and negative users. But, this naive method has been shown to be ineffective in the related tasks [21], [16]. In this paper, we propose the following popularity-biased method to sample both negative items and negative users.

- Popularity-biased Item Sampling (PIS): Inspired by [13], [28], we adopt $P(v_n) \propto (\sum_i Y_{ikn})^{0.75}$ as the negative sampling probability distribution of items, where $\sum_i Y_{ikn}$ indicates the popularity of item v_n with action t_k. Those popular items that a given user has not interacted via action t_k could be truly negative with a high probability.
- Popularity-biased User Sampling (PUS): Similar to negative item sampling, we use $P(u_n) \propto (\sum_j Y_{nkj})^{0.75}$ as the negative sampling probability distribution of users. Those

active users who have performed action t_k on many items but have not interacted with v_j would not be interested in v_j with a high probability.

Let N be the number of negative items and negative users sampled for a positive example (u_i, t_k, v_j), and our objective function can be redefined as in Eqn. (7). Thus, the task is to distinguish the positive example from their corresponding negative examples using the logistic regression method. We minimize the objective function in Eqn. (7) with Mini-Batch Stochastic Gradient Descent (SGD) and control the learning rate using *AdaGrad* [4], as suggested in [9]. A positive example and $2N$ sampled negative examples are treated as a mini batch in our algorithm. Given an example x_{ikj} which can be either positive or negative, the value of the loss function at this example is defined as:

$$\begin{aligned}\mathcal{O}(x_{ikj}) = -\big(&y_{ikj}\log\sigma(f(x_{ikj};\Theta)) \\ &+ (1-y_{ikj})\log(1-\sigma(f(x_{ikj};\Theta)))\big),\end{aligned} \tag{8}$$

where we omit the regularization terms. Then, the gradients of $\mathcal{O}(x_{ikj})$ w.r.t. $\boldsymbol{u}_i$, $\boldsymbol{t}_k$ and $\boldsymbol{v}_j$ are computed as follows:

$$\frac{\partial\mathcal{O}(x_{ikj})}{\partial\boldsymbol{u}_i} = \big(\sigma(f(x_{ikj};\Theta)) - y_{ikj}\big)(\boldsymbol{t}_k + \boldsymbol{v}_j) \tag{9}$$

$$\frac{\partial\mathcal{O}(x_{ikj})}{\partial\boldsymbol{t}_k} = \big(\sigma(f(x_{ikj};\Theta)) - y_{ikj}\big)(\boldsymbol{u}_i + \boldsymbol{v}_j) \tag{10}$$

$$\frac{\partial\mathcal{O}(x_{ikj})}{\partial\boldsymbol{v}_j} = \big(\sigma(f(x_{ikj};\Theta)) - y_{ikj}\big)(\boldsymbol{t}_k + \boldsymbol{u}_i) \tag{11}$$

Let $\mathcal{D}_{u_i}$ denote the set of examples (including both positive and negative ones) associated with user u_i in a mini batch. Note that u_i can be either positive or negative user in the mini batch. When u_i is a sampled negative user, there is only one example in $\mathcal{D}_{u_i}$. Similarly, let $\mathcal{D}_{v_j}$ and $\mathcal{D}_{t_k}$ denote the sets of examples associated with item v_j and behavior type t_k, respectively. The gradients of the objective function w.r.t. $\boldsymbol{u}_i$, $\boldsymbol{t}_k$ and $\boldsymbol{v}_j$ for a mini-batch are updated as follows:

$$\frac{\partial\mathcal{O}}{\partial\boldsymbol{u}_i} = \sum_{x_{ikj}\in\mathcal{D}_{u_i}} \frac{\partial\mathcal{O}(x_{ikj})}{\partial\boldsymbol{u}_i} + \lambda_U\boldsymbol{u}_i \tag{12}$$

$$\frac{\partial\mathcal{O}}{\partial\boldsymbol{t}_k} = \sum_{x_{ikj}\in\mathcal{D}_{t_k}} \frac{\partial\mathcal{O}(x_{ikj})}{\partial\boldsymbol{t}_k} + \lambda_T\boldsymbol{t}_k \tag{13}$$

$$\frac{\partial\mathcal{O}}{\partial\boldsymbol{v}_j} = \sum_{x_{ikj}\in\mathcal{D}_{v_j}} \frac{\partial\mathcal{O}(x_{ikj})}{\partial\boldsymbol{v}_j} + \lambda_V\boldsymbol{v}_j \tag{14}$$

The algorithm of model training is illustrated in Algorithm 1. To further accelerate the training process, we adopt the asynchronous stochastic gradient descent to parallelize the model optimization procedure on GPUs, which is very efficient and effective on sparse tensors [19], [28]. This is because when different threads sample different positive examples for model updating on a sparse tensor, the elements of the sampled examples in different threads seldom overlap, i.e., the latent vectors of the elements (e.g., users and items) usually do not conflict across different threads.

Time Complexity Analysis. In our model optimization method, each stochastic gradient step takes $O(2ND) = O(D)$,

Algorithm 1: Training SPTF Model

Input: Heterogenous user behavior dataset $\mathcal{D}^+$, number of stochastic gradient steps M, number of negative samples $2N$ for each positive sample;
Output: The parameter set $\Theta = \{\boldsymbol{u}_i, \boldsymbol{t}_k, \boldsymbol{v}_j\}$, i.e., the latent vectors for each user u_i, behavior type t_k and item v_j;

```
iter ← 0;
while iter ≤ M do
    /* Constructing a mini batch:                    */
    Sample a positive example x_ikj from D+;
    Sample N negative examples x_ikn using Popularity-biased
      Item Sampling strategy;
    Sample N negative examples x_nkj using Popularity-biased
      User Sampling strategy;
    /* Update the model parameters:                  */
    Update the gradients of the associated users, items and
      behavior type in the mini-batch according to
      Equations 12-14;
    Update the learning rate using AdaGrad[6];
    Update the model parameters by Mini-batch SGD;
    iter ← iter + 1;
end
```

where $2N$ is the number of negative examples and N is often very small (e.g., 2 or 3) in large-scale datasets [28], [13] and thus can be ignorable; D is the dimension of the latent vector and typically smaller than 100. We assume that our model needs M samples (i.e., M stochastic gradient steps) to reach convergence, thus, the overall time complexity is $O(DM)$. In practice, the required number of stochastic gradient steps M is typically proportional to the number of positive triples in the tensor [28]. Therefore, the time complexity of our model optimization algorithm is linear to the number of positive observations in the dataset, which is scalable to large-scale sparse semantic-aware behavior datasets.

D. Adaptive Positive Sampling Approach

In the model optimization described above, we assume that the positive examples are uniformly sampled, just as most stochastic gradient descent-based methods have done [13], [22], [21]. However, user behaviors are heterogenous in our problem, and the distribution of positive examples w.r.t. behavior types is heavily skewed and imbalanced. For example, click behaviors take up 86.58% of the observed examples in our collected T-mall dataset, while the proportions for add-to-favorite, add-to-cart and purchase behaviors are 4.93%, 5.91% and 2.57%, respectively. If the uniform sampling strategy is adopted in our method, most sampled positive examples would be click behaviours. Thus, the trained model cannot predict other three types of user behaviors accurately. Furthermore, even for the same type of user behaviors, the importance and informativeness of each positive example is different and may change during the model training process. A desirable positive sampler is expected to choose such positive examples with high probabilities which are informative at the current state of learning (i.e., for the current values of the model parameter set Θ) and more helpful to correct the model. However, the widely adopted uniform samplers [13], [22], [21] and the fixed weight-based samplers [28], [29] do not reflect this aspect. They are static and thus do not take into

account that the estimated score for a specific example x_{ikj} (i.e., $f(x_{ikj}; \widehat{\Theta})$) changes during the learning process. For example, the estimated score for example x_{ikj} is high at the beginning, but after several steps of learning it becomes low.

To address the above issues, we propose a novel adaptive sampling scheme to draw positive examples. Our intuition is that when sampling a positive example x_{ikj}, the smaller the score $f(x_{ikj}; \widehat{\Theta})$ is, the more informative and helpful the positive example is to correct the model, because the score represents the model's confidence that a triple exists given the current parameter values $\widehat{\Theta}$. Actually, we only concern the relative scores rather than the absolute scores, and more precisely we only care about the relative scores of x_{ikj} compared with the negative examples associated with u_i and t_k in our problem. Thus, we propose to compute the weight of a positive example x_{ikj} based on its normalized rank as:

$$w_{ikj} = exp(rank(x_{ikj})/(N_{ik}+1)), \quad \lambda \in \mathbb{R}^+ \tag{15}$$

where $rank(x_{ikj})$ is the rank of the positive example x_{ikj} when compared with all negative examples associated with user u_i and behavior type t_k, and N_{ik} is the total number of negative examples associated with user u_i and behavior type t_k. The ranking is computed based on the current state of learning (i.e., the current values of Θ). We adopt the exponential function to compute the weight according to its normalized rank, following [21]. This ranking-based weighting scheme favors the positive examples at a lower rank much more than the ones at the top. As only the examples with the same user and behavior type are comparable in our problem, we restrict the examples in a ranked list to be associated with the same user and behavior type. Finally, the sampling probability for a positive example x_{ikj} is defined as follows.

$$P(x_{ikj}) = \frac{w_{ikj}}{\sum_{x_{i'k'j'} \in \mathcal{D}^+} w_{i'k'j'}}. \tag{16}$$

Obviously, the above positive sampler will change while model parameter values Θ are updated because the changes in Θ result consecutively in changes in the weights of positive examples. Thus, our proposed sampler is *adaptive*. For the efficient implementation of the above sampler, we use the alias table method [10], which takes only $\mathcal{O}(1)$ time when repeatedly drawing samples from the same discrete distribution.

Even so, an exact implementation of the above positive sampler would need to compute a score for each positive and negative example and then do the ranking at each gradient descent step, which is still rather expensive and infeasible in practice. Inspired by [18], we propose to estimate $rank(x_{ikj})$ through a sequential sampling procedure that repeatedly samples a negative example until we find a negative example x_{ikn} that has a larger score than x_{ikj}. Such negative example is called imposter. Specifically, let N_{ikj} denote the number of negative examples we need to sample to find an imposter, then the $rank(x_{ikj})$ is approximated as $\lfloor \frac{N_{ik}+1}{N_{ikj}} \rfloor$. To further accelerate the positive sampling process, the number of negative samples N_{ikj} is bounded by a constant 15 or 20 to avoid an extended sampling time [6], [37], and we replace this sequential procedure with a parallel procedure to utilize the massive parallelism enabled by modern GPUs:

1) For each positive example x_{ikj}, uniformly sample N_{ikj} (N_{ikj} is limited to be a small constant) negative examples x_{ikn} in parallel and compute their scores.

2) Let Q_{ikj} denote the number of imposters in N_{ikj} samples, $rank(x_{ikj})$ is then approximated as $\lfloor \frac{(N_{ik}+1)\times Q_{ikj}}{N_{ikj}} \rfloor$, and the weight w_{ikj} is approximated as $exp(\frac{Q_{ikj}}{N_{ikj}})$.

It might seem wasteful to always sample N_{ikj} negative examples and compute their scores. Empirically, we find that our proposed model optimization method based on the adaptive positive sampler only needs few epochs to push the positive examples to a high rank. Therefore, we only need to adopt the adaptive ranking-based positive sampler in the first few epochs, and then use a two-step method to successively sample a behavior type and a positive example with the drawn behavior type at a random manner, following [29].

E. Semantic-Aware User Behavior Prediction

Once the model parameters Θ have been estimated, we can make semantic-aware user behavior prediction. Specifically, given a user u_i and a behavior type t_k, we can predict top-n items on which u_i will perform action t_k according to the predicted scores calculated by $f(x_{ikj}; \Theta)$. This kind of semantic-aware behavior prediction has many practical and important applications in e-commerce and social network platforms, e.g., to recommend top-n products that the target user will most likely purchase in the e-commerce platforms.

III. Experiments

In this section, we first describe the experimental settings and then report the experimental results.

A. Dataset

We conduct our experiments on a large-scale & real-life dataset provided by T-mall of Alibaba in 2014. This dataset contains 480,723 products, 10000 users and their generated twenty-million behavior records during 18/11/2014 – 18/12/2014. Each behavior record contains a user-id, item-id, behavior-type and time stamp. In this dataset, there are four types of user behaviors: click, add-to-favorite, add-to-cart and purchase. Table I lists the numbers of behavior records for each behavior type. To make our experiments repeatable, we make our dataset and codes publicly available at http://pan.baidu.com/s/1mhQ0ifQ with the password `jrlb`.

TABLE I
Statistics of user behavior records.

Click	Add-to-favorite	Add-to-cart	Purchase
1213836	69073	82985	36073

B. Evaluation Method

Evaluation of Prediction Accuracy. Given a collection of behavior records $\mathcal{D}_u^+$ generated by each user u, we first rank these behavior records according to their timestamps. Then, we use the 80-th percentile as the cut-off point so that the behavior records before this point will be used for training, and the rest are for testing. In the training dataset, we choose the last

TABLE II
CHARACTERISTICS OF THE SPTF VARIANTS.

Variations	Negative Sampling Strategy				Positive Sampling Strategy	
	Uniform Sampling	Popularity-biased Sampling	Bidirectional Sampling	Unidirectional Sampling	Uniform Sampling	Adaptive Sampling
SPTF-V1	✓		✓			✓
SPTF-V2		✓		✓		✓
SPTF-V3		✓	✓		✓	
SPTF-V4	✓			✓	✓	
SPTF		✓	✓			✓

10% records as the validation data to tune the model hyper-parameters such as the dimension of latent feature vectors and the regularization parameters.

According to the above dividing strategies, we split the dataset $\mathcal{D}^+$ into the training set $\mathcal{D}^+_{training}$ and the test set $\mathcal{D}^+_{test}$. To evaluate the prediction models, we employ the methodology and measurement Hits ratio which have been widely adopted by recommender system and knowledge graph communities [2], [11], [3]. Specifically, for each user behavior record $x_{ikj} \in \mathcal{D}^+_{test}$:

1) We randomly choose 10000 items on which user u_i has never performed action t_k to replace item v_j and form 10000 negative examples.
2) We compute a score for x_{ikj} as well as the 10000 negative examples by the score function in Equation 4.
3) We form a ranked list by ordering these 10001 examples according to their scores. Let $rank(x_{ikj})$ denote the position of x_{ikj} in the ranking list.
4) We form a top-n prediction list by picking the n top ranked examples from the list. If $rank(x_{ikj}) \leq n$, we have a hit. Otherwise, we have a miss.

The computation of Hits ratio proceeds as follows. We define hit@n for a single test case as either 1, if the positive example x_{ikj} appears in the top-n results, or 0, if otherwise. The overall Hits@n is defined by averaging over all test cases:

$$Hits@n = \frac{\#hit@n}{|\mathcal{D}^+_{test}|}$$

where $\#hit@n$ denotes the number of hits in the test set, and $|\mathcal{D}^+_{test}|$ is the number of all test cases. A good predictor should achieve higher Hits@n. We can further divide $\mathcal{D}^+_{test}$ into four groups of triples according to the behavior types, and then analyze the performance of prediction models on each specific type of user behaviors.

Besides Hits ratio, we also adopt the commonly used metric in information retrieval Mean Reciprocal Rank (MRR) to measure the prediction accuracy, and it is defined as follows.

$$MRR = \frac{1}{|\mathcal{D}^+_{test}|} \sum_{x_{ikj} \in \mathcal{D}^+_{test}} \frac{1}{rank(x_{ikj})}$$

MRR is an average of the reciprocal rank of a positive example over all sampled negative examples, and a good prediction model should have a bigger MRR value. In contrast to mean rank, MRR is less sensitive to outliers.

Hyper-parameter Optimization. There are some hyper parameters in our SPTF model, and we perform the cross-validation. Specifically, we use the grid search algorithm to obtain the optimal hyper parameter setup on the validation dataset: $D = 50$, $\lambda_U = 0.001$, $\lambda_T = 0.0001$ and $\lambda_V = 0.01$. We also selectly present the parameter sensitivity analysis for some novel and important hyper-parameters, such as the number of negative examples sampled for each positive example N, the number of training samples M (i.e., the number of stochastic gradient steps) and the number of negative examples needed in the adaptive ranking-based sampler N_{ikj}.

Evaluation of Model Efficiency and Scalability. In addition to the prediction accuracy, we also evaluate the efficiency of the model training in terms of time cost. As the asynchronous stochastic gradient descent algorithm is employed to perform model optimization, we also test the speedup ratio of the model training process by varying the number of threads.

C. Comparison Methods

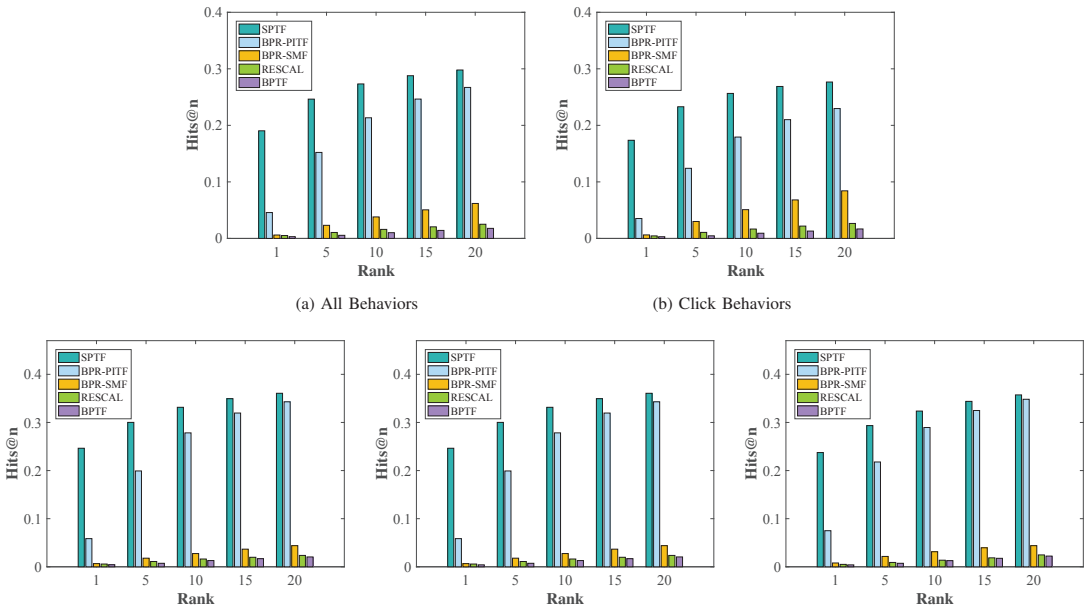
We compare our proposed SPTF model with the following powerful tensor factorization and matrix factorization models that can be applied to semantic-aware user behavior prediction.

BPTF. BPTF [30] provides a Bayesian treatment of probabilistic tensor factorization and adopts the Canonical Decomposition (CD) where the score function $f(x_{ikj}; \Theta)$ is defined as the inner-product of the three latent vectors $\boldsymbol{u}_i$, $\boldsymbol{t}_k$ and $\boldsymbol{v}_j$. BPTF was designed for rating prediction in explicit feedback datasets, thus BPTF only considers the observed examples.

RESCAL. RESCAL [14], [15] is the state-of-the-art tensor factorization model that was proposed for factoring knowledge graph. Different from BPTF, a "predicate" (e.g., the behavior-type in our problem) is represented by a matrix rather than a vector. Thus, it has more powerful expressive capacity and more parameters to be estimated. Once again, RESCAL only learns from observed examples.

BPR-PITF. BPR-PITF [24] is a special case of the Tucker Decomposition model and explicitly models the pairwise interactions between users, items and behavior-types. The model is learned with an adaption of the Bayesian personalized ranking (BPR) criterion. In this model, all unobserved examples are treated as negative examples, and each training case is uniformly drawn. Although our SPTF adopts the same pairwise interaction approach to decompose the tensor, we use different loss functions and optimization methods to learn the model. Specifically, we develop both popularity-biased negative sampling method and adaptive ranking-based positive sampling approach to perform model optimization in SPTF.

BPR-SMF. In contrast to BPR-PITF, BPR-SMF means separate BPR-based matrix factorization. In our dataset, there are four user-item matrices, and each of them stores a specific type of user behaviors. We apply BRP-based matrix factorization model to each matrix separately, and four latent vectors are learned for each user or item that align with four types of



(a) All Behaviors (b) Click Behaviors

(c) Add-to-Cart Behaviors (d) Add-to-Favorite Behavior (e) Purchase Behavior

Fig. 1. Semantic-Aware Behavior Prediction Accuracy in Terms of $Hits@n$

user behaviors. Thus, the latent vectors for the same users or items are not shared across the four matrices. We employ BPR-based matrix factorization model [22] due to its excellent performance in the one-class collaborative filtering.

To validate the benefits brought by our proposed popularity-biased negative sampling strategy, bidirectional negative sampling strategy, and adaptive ranking-based positive sampling approach respectively, we design four variants of SPTF to carry out an ablation study. Their adopted sampling strategies are shown in Table II. Compared with our SPTF model, the first variant version **SPTF-V1** simply adopts the uniform sampling strategy for the generation of negative examples instead of the popularity-biased sampling strategy. Although the second variant **SPTF-V2** adopts our proposed popularity-biased sampling strategy, it generates the negative examples only from the user perspective (i.e., given a positive example x_{ikj}, the corresponding negative examples are generated by fixing u_i and t_k) rather than in a bidirectional way. The third variant **SPTF-V3** uses the naive uniform sampling strategy to draw positive examples rather than our proposed adaptive ranking-based sampling approach. The last variant **SPTF-V4** is a simple baseline model, since it utilizes the uniform sampling strategy to draw both positive and negative examples.

D. Effectiveness of Semantic-Aware User Behavior Prediction

In this subsection, we report the comparison results between our proposed SPTF and other competitor models with well-tuned parameters in Figure 1 and Table III. All differences between our model and the other comparison models are statistically significant ($p < 0.01$).

In Figure 1, we present the prediction accuracy of all comparison models in terms of $Hits@n$. Specifically, Figure 1 (a) shows their overall prediction performance on the whole test set that consists of four types of user behaviors, and Figures 1 (b) - (e) further detail the prediction performance on each specific type of user behaviors. We only show the performance where n is set to 1, 5, 10, 15, 20, as a greater value of n is usually meaningless. Clearly, our proposed SPTF significantly and consistently outperforms other models in various types of user behavior prediction, and more importantly, the improvement is more significant when n is smaller (i.e., $n \leq 5$). For example, the overall relative improvements in terms of $Hits@1$, shown in Figure 1 (a), are 3.2×, 32.1×, 35.6× and 59.4× compared with BPR-PITF, BPR-SMF, RESCAL and BPTF, respectively.

Several other observations are also made from the results. First, SPTF, BPR-PITF and BPR-SMF achieve much higher prediction accuracy than RESCAL and BPTF, showing the importance of exploiting negative examples, as both RESCAL and BPTF only consider the positive examples in the model optimization and BPTF is designed for explicit rating prediction. Second, both SPTF and BPR-PITF outperform BPR-SMF significantly. This demonstrates the advantage of collective factorization for modeling multi-relational data over the separate factorization-based method, as the collective factorization models can effectively address the data sparsity issue by sharing the same latent vector across all relations. We further observe that BPR-SMF achieves its best prediction performance on the click behaviors due to the sufficient training data on click behaviors, while SPTF and BPR-PITF achieve their highest prediction accuracy for Add-to-cart and Purchase behaviors respectively. Third, SPTF beats BPR-PITF. Although both them adopt the same pair-wise tensor decomposition form, they use different loss functions and model optimization methods. Moreover, we develop advanced popularity-biased negative sampling strategy and adaptive positive sampling

strategy, while BPR-PITF simply treats all unobserved examples as negative examples and uses the simple uniform sampling strategy to draw training cases. Fourth, the prediction performances of all the models in terms of MRR, shown in Table III, are highly consistent with their performances in terms of $Hits@n$ shown in Figures 1.

TABLE III
PREDICTION ACCURACY IN TERMS OF MRR.

	SPTF	BPR-PITF	BPR-SMF	RESCAL	BPTF
All Behaviors	0.195	0.097	0.018	0.010	0.005
Click	0.181	0.079	0.022	0.010	0.005
Add-to-Favorite	0.128	0.081	0.006	0.011	0.004
Add-to-Cart	0.245	0.126	0.015	0.010	0.005
Purchase	0.239	0.142	0.016	0.010	0.006

E. Impact of Different Factors

In this experiment, we study the impact of various sampling techniques and the hyper-parameter setup on the whole test set.

Analysis of Various Sampling Strategies. We carry out an ablation study to show the benefits of each proposed sampling strategy in SPTF model, i.e., the popularity-biased negative sampling strategy, the bidirectional negative sampling strategy and the adaptive ranking-based positive sampling approach. We compare SPTF with its four variant versions shown in Table II, and the comparison results are shown in Figure 2. From the results, we first observe that SPTF consistently outperforms the four variant versions, indicating the benefits brought by each of our developed sampling strategies in this paper, respectively. For instance, the performance gap between SPTF and SPTF-V3 validates the significant benefit brought by our adaptive ranking-based positive sampling approach. Second, all SPTF, SPTF-V1, SPTF-V2 and SPTF-V3 achieve higher prediction accuracy than the baseline model SPTF-V4, as SPTF-V4 simply adopts the common uniform sampling strategy to draw both positive and negative examples. In addition, it constructs negative examples from only the user perspective. Third, we find that the contribution of each developed sampling strategy to improving prediction accuracy is different. The relative improvements brought by the adaptive ranking-based positive sampling approach and the popularity-biased negative sampling strategy are the most significant.

Sensitivity Analysis. In this experiment, we investigate the sensitivity of the model hyper-parameters in our SPTF, including the number of negative examples N, the number of stochastic gradient steps M (i.e., the number of training samples) and the number of negative examples needed in the adaptive ranking-based sampler N_{ikj}. Figure 3 shows the results w.r.t. the number of negative examples (N). From the figure, we observe that when the number of negative examples is larger than 4, the performance becomes very stable. For each positive example, we do not need to sample many negative examples and just need to sample a few negative ones according to our proposed bidirectional popularity-biased sampling strategy. Figure 4 shows the results w.r.t. the number of negative examples needed in the adaptive ranking-based sampler N_{ikj}. We find that when N_{ikj} becomes larger than 15, the performance of our SPTF model becomes stable. When we continue to increase the values of N_{ikj}, the improvement

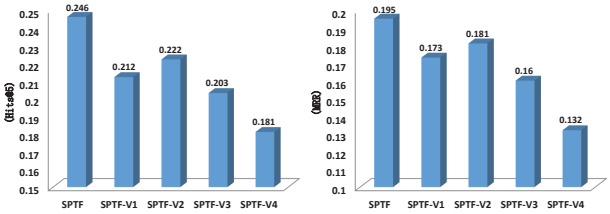


Fig. 2. The Performance of SPTF with different sampling strategies.

is almost ignorable. Figure 5 presents the performance of our SPTF model w.r.t. the number of training samples (i.e., the number of stochastic gradient steps). When the number of samples is larger than 30 millions, our model converges quickly and its performance becomes very stable.

F. Evaluation of Efficiency and Scalability

In this experiment, we first investigate the scalability of SPTF model optimized by the asynchronous stochastic gradient descent, which deploys multiple threads for model optimization on GPUs. Figure 6(a) shows the speedup ratio w.r.t. the number of threads. The speedup ratio is quite close to linear. Figure 6(b) shows that the prediction performance does not obviously degrade when using multiple threads for model training. The two figures together show that the optimization algorithm of the SPTF is quite scalable. In addition, we also compare the training time costs of our SPTF model and other comparison models on the same hardware environments. All models were trained on a Windows Server 2008 with $256G$ RAM and $24G$ GDDR5 (NVIDIA K80). Table IV shows the training time of different models on the T-mall dataset where SPTF-10 means that we use 10 threads to train our SPTF model. From the results, we observe that although the basic implementation of SPTF with a single thread is a little costly, our parallel implementation guarantees the efficiency. As both BPTF and RESCAL only consider the positive examples, their training time costs are slightly lower than our SPTF. But their prediction performances are much worse than our SPTF. BPR-PITF treats all unobserved examples as negative examples and needs more stochastic gradient steps to reach convergence, thus its training time cost is most expensive among all comparison models.

TABLE IV
MODEL TRAINING TIME (HOURS).

SPTF	SPTF-10	BPR-PITF	RESCAL	BPTF
5.92	0.65	19.55	4.65	3.21

G. Similarity Analysis of Behavior Semantics

In this experiment, we study the semantic similarity between any two types of user behaviors on T-mall dataset. To be more specific, we investigate the similarity between the learned latent vectors of the four behavior types (i.e., click, add-to-favorite, add-to-cart, purchase). Given any two behavior types, t_i and t_j, we compute their inner-product and Euclidean distance to represent their similarity. The results are shown in Tables V and VI where we highlight three most similar behavior type pairs, and the similarity results computed by

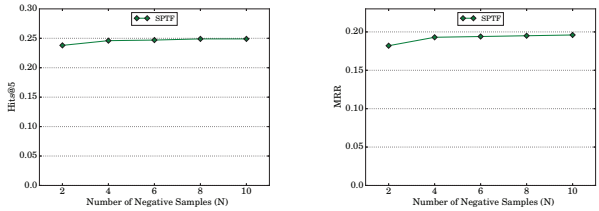


Fig. 3. Performance of SPTF w.r.t. the number of negative samples (N).

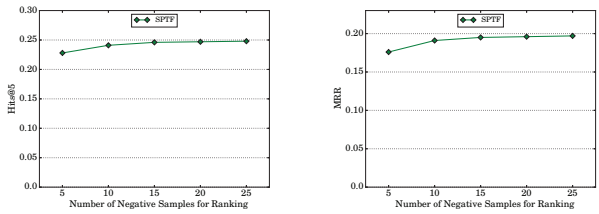


Fig. 4. Performance of SPTF w.r.t. the number of negative samples (N_{ikj}).

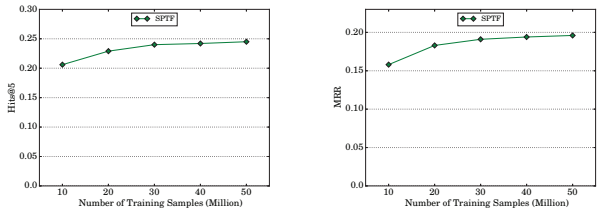
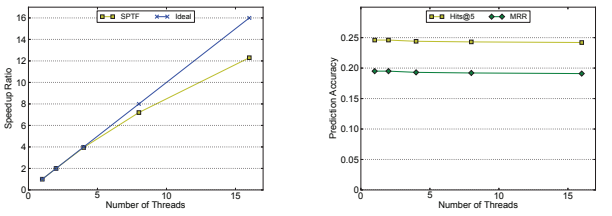


Fig. 5. Performance of SPTF w.r.t. the number of training samples (M).



(a) Speedup vs. #threads (b) Prediction accuracy vs. #threads

Fig. 6. Performance w.r.t. the number of threads.

inner product and Euclidean distance are highly consistently. One interesting finding is that both add-to-favorite and add-to-cart behaviors are more similar to purchase behaviors than click behaviors, which implies that add-to-favorite and add-to-cart behavior data have more powerful potential to improve the prediction of user purchase behaviors than click data. This observation is consistent with our intuition and common sense. Another interesting yet novel finding is that the similarity between add-to-favorite and purchase is slightly higher than the similarity between add-to-cart and purchase.

IV. Related Work

One-class Collaborative Filtering. Pan et al. [16] first proposed the concept of One-Class Collaborative Filtering (OCCF) that allows Collaborative Filtering (CF) methods, especially Matrix Factorization (MF), to model users's preferences from the implicit feedback data where only positive

TABLE V
Similarity Analysis of User Behavior Types (I).

Inner Product	Click	Add-to-cart	Add-to-favorite	Purchase
Click		9.0769	14.5250	7.5877
Add-to-cart	9.0769		**22.3425**	**22.8463**
Add-to-favorite	14.5250	**22.3425**		**23.7204**
Purchase	7.5877	**22.8463**	**23.7204**	

TABLE VI
Similarity Analysis of User Behavior Types (II).

Euclidean distance	Click	Add-to-cart	Add-to-favorite	Purchase
Click		5.6834	5.1763	6.0207
Add-to-cart	5.6834		**5.0732**	**4.2983**
Add-to-favorite	5.1763	**5.0732**		**4.1367**
Purchase	6.0207	**4.2983**	**4.1367**	

examples (e.g., click, download and purchase) are observed. This line of work includes point-wise models [7], [16], [35] which implicitly treat unobserved interactions as negative examples. Rendle et al. [22], [21] refined such ideas to develop a Bayesian Personalized Ranking (BPR) framework to optimize pair-wise rankings of positive versus unobserved examples. BPR-MF combines the strengths of the BPR framework and the efficiency of MF and forms the basis of many state-of-the-art personalized OCCF methods [17]. All these existing matrix factorization models only focus on a single type of user behaviors and ignore the variety and heterogeneity of user behaviors in the real world. Therefore, they inevitably encounter the severe issue of data sparsity when applying to our problem of semantic-aware behavior prediction.

Tensor Factorization Techniques. Tensor factorization methods have been used in many applications such as recommender systems [30], [24], [23] and knowledge graph completion [15]. For example, BPTF [30] treats the temporal user behavior data as a user-item-time tensor to provide temporal context-aware recommendations. Tensor factorization models have been the most successful techniques in the tag recommendation task [20], [24], [5], [27]. Higher Order Singular Vector Decomposition (HOSVD) [27] and RTF [20] are based on Tucker Decomposition (TD), while PITF [24], BPTF [30] and NLTF [5] are based on Canonical Decomposition (CD). Although TD-based methods outperform other state-of-the-art tag recommendation approaches, they require a lot of computation power and time. Therefore, they are not feasible for online recommendation and large-scale datasets. Compared with TD-based methods, CD methods have a huge advantage in running time, because they can be computed in linear time. PITF and NLTF, an extension of CD, splits the ternary relation of users, items and tags into three relations, user-tag and item-tag and user-item.

As for the model optimization, HOSVD and BPTF only consider the positive examples and cannot deal with unobserved values. RTF, PITF and NLTF adopt the BPR optimization criterion and handle the unobserved values by adding pairwise ranking constraint. However, to perform BPR optimization, they need to create a training dataset for pairwise constraints, and the size of the training dataset is very huge, i.e., $|\mathcal{V}|$ times of the number of observed examples where $|\mathcal{V}|$ is the number of items in our problem. In contrast, our

proposed model optimization method only needs to create $2N$ informative negative examples for a positive example instead of $|\mathcal{V}|$ negative examples where N is a very small number (e.g., 2 or 3). Obviously, the convergence of these BPR-based tensor factorization models slows down considerably if the number of items $|\mathcal{V}|$ is large. Thus, these BPR-based tensor models cannot be deployed to the real e-commerce and social network platforms. Besides, these BPR-based tensor factorization models uniformly sample a training case and performs stochastic gradient descent for the drawn case. This sampling method cannot address the issue of the heavy skewness of heterogenous behavior data distribution w.r.t. behavior types.

V. Conclusions

In this paper, we developed a scalable probabilistic tensor factorization model (SPTF) to model and predict semantic-aware user behaviors. To optimize the model of SPTF, we proposed a novel bidirectional popularity-biased negative sampling technique to leverage both observed and unobserved examples with much lower computational costs and higher modeling accuracy. In addition, we proposed a novel adaptive ranking-based sampling approach to overcome the heavy skewness of the heterogenous behavior data distribution w.r.t. behavior types and further improve the model convergence speed. Extensive experiments were conducted on a large-scale e-commerce dataset, and our proposed SPTF significantly outperforms state-of-the-art behavior prediction models in terms of prediction accuracy and scalability.

Acknowledgement

This work was supported by ARC Discovery Early Career Researcher Award (Grant No. DE160100308), ARC Discovery Project (Grant No. DP140103171) and New Staff Research Grant of The University of Queensland (Grant No.613134). It was also partially supported by National Natural Science Foundation of China (Grant No. 61572335 and 61672501).

References

[1] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. on Knowl. and Data Eng.*, 17(6):734–749, June 2005.

[2] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko. Translating embeddings for modeling multi-relational data. In *NIPS*, pages 2787–2795. 2013.

[3] P. Cremonesi, Y. Koren, and R. Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *RecSys*, pages 39–46, 2010.

[4] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12:2121–2159, July 2011.

[5] X. Fang, R. Pan, G. Cao, X. He, and W. Dai. Personalized tag recommendation through nonlinear tensor factorization using gaussian kernel. In *AAAI*, pages 439–445, 2015.

[6] C.-K. Hsieh, L. Yang, Y. Cui, T.-Y. Lin, S. Belongie, and D. Estrin. Collaborative metric learning. In *WWW*, pages 193–201, 2017.

[7] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *ICDM*, pages 263–272, 2008.

[8] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, Aug. 2009.

[9] M. Kula. Lightfm. https://github.com/lyst/lightfm/, 2016.

[10] A. Q. Li, A. Ahmed, S. Ravi, and A. J. Smola. Reducing the sampling complexity of topic models. In *KDD*, pages 891–900, 2014.

[11] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*, pages 2181–2187, 2015.

[12] Q. Liu, X. Zeng, C. liu, H. Zhu, E. Chen, H. Xiong, and X. Xie. Mining indecisiveness in customer behaviors. In *ICDM*, pages 281–290, 2015.

[13] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119, 2013.

[14] M. Nickel, V. Tresp, and H.-P. Kriegel. A three-way model for collective learning on multi-relational data. In *ICML*, pages 809–816, 2011.

[15] M. Nickel, V. Tresp, and H.-P. Kriegel. Factorizing yago: Scalable machine learning for linked data. In *WWW*, pages 271–280, 2012.

[16] R. Pan, Y. Zhou, B. Cao, N. N. Liu, R. Lukose, M. Scholz, and Q. Yang. One-class collaborative filtering. In *ICDM*, pages 502–511, 2008.

[17] W. Pan and L. Chen. Gbpr: Group preference based bayesian personalized ranking for one-class collaborative filtering. In *IJCAI*, pages 2691–2697, 2013.

[18] K. Pramod Sankar, R. Manmatha, and C. V. Jawahar. Large scale document image retrieval by automatic word annotation. *Int. J. Doc. Anal. Recognit.*, 17(1):1–17, Mar. 2014.

[19] B. Recht, C. Re, S. Wright, and F. Niu. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *NIPS*, pages 693–701, 2011.

[20] S. Rendle, L. Balby Marinho, A. Nanopoulos, and L. Schmidt-Thieme. Learning optimal ranking with tensor factorization for tag recommendation. In *KDD*, pages 727–736, 2009.

[21] S. Rendle and C. Freudenthaler. Improving pairwise learning for item recommendation from implicit feedback. In *WSDM*, pages 273–282, 2014.

[22] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *UAI*, pages 452–461, 2009.

[23] S. Rendle, Z. Gantner, C. Freudenthaler, and L. Schmidt-Thieme. Fast context-aware recommendations with factorization machines. In *SIGIR*, pages 635–644, 2011.

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

[25] R. Salakhutdinov and A. Mnih. Probabilistic matrix factorization. In *NIPS*, volume 20, pages 1–8, 2011.

[26] N. Srebro, T. Jaakkola, et al. Weighted low-rank approximations. In *ICML*, volume 3, pages 720–727, 2003.

[27] P. Symeonidis, A. Nanopoulos, and Y. Manolopoulos. Tag recommendations based on tensor dimensionality reduction. In *RecSys*, pages 43–50, 2008.

[28] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei. Line: Large-scale information network embedding. In *WWW*, pages 1067–1077, 2015.

[29] M. Xie, H. Yin, H. Wang, F. Xu, W. Chen, and S. Wang. Learning graph-based poi embedding for location-based recommendation. In *CIKM*, pages 15–24, 2016.

[30] L. Xiong, X. Chen, T.-K. Huang, J. Schneider, and J. G. Carbonell. *Temporal Collaborative Filtering with Bayesian Probabilistic Tensor Factorization*, pages 211–222. 2010.

[31] H. Yin and B. Cui. *Spatio-Temporal Recommendation in Social Media*. Springer, 2016.

[32] H. Yin, B. Cui, L. Chen, Z. Hu, and X. Zhou. Dynamic user modeling in social media systems. *ACM Trans. Inf. Syst.*, 33(3):10:1–10:44, Mar. 2015.

[33] H. Yin, B. Cui, X. Zhou, W. Wang, Z. Huang, and S. Sadiq. Joint modeling of user check-in behaviors for real-time point-of-interest recommendation. *ACM Trans. Inf. Syst.*, 35(2):11:1–11:44, Oct. 2016.

[34] H. Yin, Z. Hu, X. Zhou, H. Wang, K. Zheng, Q. V. H. Nguyen, and S. Sadiq. Discovering interpretable geo-social communities for user behavior prediction. In *ICDE*, pages 942–953, May 2016.

[35] H. Yin, W. Wang, H. Wang, H. Wang, L. Chen, and X. Zhou. Spatial-aware hierarchical collaborative deep learning for poi recommendation. *TKDE*, 2017.

[36] S. Zhang, W. Wang, J. Ford, F. Makedon, and J. Pearlman. Using singular value decomposition approximation for collaborative filtering. In *CEC*, pages 257–264, 2005.

[37] T. Zhao, J. McAuley, and I. King. Improving latent factor models via personalized feature projection for one class recommendation. In *CIKM*, pages 821–830, 2015.