Fast Tensor Factorization for Large-Scale Context-Aware Recommendation from Implicit Feedback

Szu-Yu Chou, Jyh-Shing Roger Jang, *Member, IEEE*, and Yi-Hsuan Yang, *Senior Member, IEEE*

Abstract—This paper presents a fast Tensor Factorization (TF) algorithm for context-aware recommendation from implicit feedback. For such a recommendation problem, the observed data indicate the (positive) association between users and items in some given contexts. For better accuracy, it has been shown essential to include unobserved data that indicate the negative user-item-context associations. As such unobserved data greatly outnumber the observed ones, for efficiency existing algorithms usually use only a small part of the unobserved data for model training. We show in this paper that it is possible, and beneficial, to use all the unobserved data in training a TF based context-aware recommender system. This is achieved by two technical innovations. First, we scrutinize the matrix computation of the closed-form solution and accelerate the computation by memorizing the repetitive computation. Second, we further boost the generalization and scalability by dropping out some pairwise interactions when updating user, item or context factors to prevent overfitting and to reduce the training time. The resulting whole-data based learning algorithm, referred to as DropTF in the paper, is efficient and scale well. Our evaluation on two small benchmark datasets and a million-scale large dataset demonstrates improved accuracy over some existing algorithms for context-aware recommendation.

Index Terms—Recommender system, collaborative filtering, context-aware recommendation, tensor factorization, implicit feedback

1 Introduction

Recommendation has many applications in real-world problems [1], [2], [3]. Depending on the data available, a collaborative filtering (CF) based recommender system can be learned from either users' explicit feedback or implicit feedback [4], [5], [6]. *Explicit feedback*, such as user ratings and thumbs up/down, provides direct indication of a user's preference for an item [7]. If an item is rated low, we assume that the user has less interest in that item. In contrast, *implicit feedback* comprises of traces of user behavior logged by an online system, such as item clicks, play counts, and dwell time [5], [8], [9], [10]. We can assume that a user likes an item more if the user visits (or interacts with) the item more often, but we do not know whether the user dislikes an item or not even if the user visits the item only once. Sometimes, all the observed user-item associations in an implicit feedback dataset are considered as positive data, leaving no negative data at all. This has been referred to as the *one-class* problem [11].

In practice, for learning a recommender system from implicit feedback, it has been shown beneficial to view the unobserved user-item pairs as negative data and add them to the training set. This avoids the one-class problem and makes better use of the vast amount of unobserved data. There are mainly two approaches: the *whole-data based* approach uses all the unobserved data as negative data [5], [12], [13], [14], [15], whereas the *negative sampling* approach employs only a subset of unobserved data [16], [17], [18]. The former approach uses more data for learning and therefore usually performs better [12], [13] in metrics such as mean average precision (MAP) [5], but it is also slower for the unobserved data usually outnumber the observed data greatly. It gets even worse when we want to take into account the *contextual information* of the user-item association, such as location, time, user mood, and user activity [19], [20]. For such a context-aware recommendation problem, typically only a very small portion of the possible user-item-context tuples is observed, prohibiting the use of the whole-data based approach when the number of users, items and contexts is large. Accordingly, well-known algorithms for context-aware recommendation from implicit feedback are mostly based on negative sampling [18], [21], [22].

In light of the above observations, the goal of this paper is to develop an efficient whole-data based learning algorithm for context-aware recommendation from implicit feedback. In addition to efficiency improvement, we also desire the new algorithm to be better than existing negative sampling based algorithms in performance metrics such as MAP. Therefore, we choose to develop the new algorithm based on *Tensor Factorization* (TF) with pairwise interaction [23]. Given an implicit feedback dataset, it is easy to use both observed and unobserved data in TF. However, to the best of our knowledge, few attempts, if any, have been made to develop a TF-based algorithm that can efficiently learn from all the unobserved data. We believe such an extension is important for two reasons. First, it opens new doors to capitalize unobserved data by means of variants of a well-known method (i.e., TF). Second, in many real-world recommendation problems, it is easier to amass implicit feedback rather than explicit feedback [13], [24]. Therefore, it is desirable to have a TF-based algorithm that is specifically designed for implicit feedback data.

Specifically, the new algorithm we propose is a generalization of the weighted regularized matrix factorization (WRMF) algorithm proposed by Hu et al. [5]. WRMF is a whole-data based learning algorithm to learn from implicit feedback for CF. It represents user-item associations as a *matrix* and uses a few techniques to speed up learning from the whole matrix. For context-aware recommendation, the user-item-context associations have to be represented as a high-order *tensor* [23], [25], [26]. For example, we need a fourth-order tensor (i.e., a four-dimensional, or four-way array) to describe the associations between users, items, location and time. We generalize WRMF from the two-dimensional user-item case to multiple dimensions, making it amenable to the formulation of TF. In addition, we propose a method based on the binomial theorem to avoid massive repeated computation in the model training process. The resulting algorithm can efficiently learn from the whole tensor, exploiting both observed and unobserved data. We call this algorithm the *faster Tensor factorization* (FTF^{++}).

Furthermore, we find the learning process can be further sped up by dropping out some second-order interaction in the training process. In addition to being faster, this technique reduces the chance of overfitting and further improves the accuracy of the recommender system. We call the new algorithm *DropTF*.

We evaluate the efficiency and effectiveness of the proposed algorithms on three implicit feedback datasets for context-aware recommendation—the two benchmark datasets CoMoDa [27] and Frappe [28], and a larger, million-scale dataset obtained from a music streaming company. For baseline methods, we consider a few well-known factorization-based and learning-to-rank based

- *S.-Y. Chou is with the Research Center for IT Innovation, Academia Sinica, Taipei 11529, Taiwan, and also with the Graduate Institute of Networking and Multimedia, National Taiwan University, Taipei 10617, Taiwan. E-mail: fearofchou@citi.sinica.edu.tw.*
- *J.-S.R. Jang is with the Graduate Institute of Networking and Multimedia, National Taiwan University, Taipei 10617, Taiwan. E-mail: jang@csie.ntu.edu.tw.*
- *Y.-H. Yang is with the Research Center for IT Innovation, Academia Sinica, Taipei 11529, Taiwan. E-mail: yang@citi.sinica.edu.tw.*

Manuscript received 15 Mar. 2018; revised 12 Oct. 2018; accepted 7 Dec. 2018. Date of publication 21 Dec. 2018; date of current version 28 Feb. 2020.
(Corresponding author: Szu-Yu Chou.)
Recommended for acceptance by R. Jin.
Digital Object Identifier no. 10.1109/TBDATA.2018.2889121

TABLE 1
Main Notations Used in the Paper

Notation	Description
N	Number of possible user-item-context tuples
N_+	Number of observed tuples ($N_+ \ll N$)
M	Number of objects (users, items, contexts)
G	Number of context types ($G \ll M$)
K	Length of the latent representations
$\|U\|$	Number of users
$\|I\|$	Number of items
$\mathbf{x}_n \in \{0,1\}^M$	An instance of input in TF
$\mathbf{v}^{(m)} \in \mathbb{R}^K$	Latent representation for the mth object
y_n	Target output in TF
p_n	Target output in weighted TF
w_n	Weight on the nth target output

algorithms [7], [16], [18], [21], [22], all of which use negative sampling. We also consider a whole-data based method we developed recently in a previous work [15]. Our experiments show that DropTF performs better in MAP and other precision-related metrics than these prior arts in top-N recommendation [29] for the three datasets. Moreover, while many of these prior arts are computationally too expensive to be applied to the music dataset, we show that DropTF (which uses all the unobserved data) runs fairly fast and is even one to two folds faster than the fastest competing method (which uses only a handful of unobserved data). We also note that the learning of DropTF is based on alternating least squares (ALS) [7] and is therefore easily parallelizable [30], which is good for industrial applications.

2 Related Work

Our survey focuses on algorithms for recommendation from implicit feedback data, and specifically on context-aware recommendation algorithms that learn from user-item-context associations.

2.1 Learning from Implicit Feedback

Factor models that learn latent representations of users and items have been used widely in building recommender systems [5], [6], [12], [13], [14], [31]. For recommendation from implicit feedback, a common approach is to use all the observed data as positive examples, and treat the unobserved data as negative examples. A learning method is then used to learn user preference by discriminating positive items from negative items.

As stated in Section 1, there are mainly two approaches for recommendation from implicit feedback. The negative sampling approach uses a subset of unobserved data as the negative data. This can be done by either a *uniform sampling strategy* (e.g., [16]) that randomly draws a subset of unobserved data and then uses the same set throughout the learning process, or an *adaptive sampling strategy* (e.g., [24]) that draws different subsets in different learning iterations. The latter strategy has been claimed to have better convergence rate [24]. Most learning-to-rank based methods for recommendation from implicit feedback take the negative sampling approach [16], [18], [22].

The whole-data based approach, on the other hand, treats all the unobserved data as negative data in learning user preference. A number of methods have been proposed for improving the efficiency of whole-data based matrix factorization (MF). The most notable example is the WRMF method presented by Hu et al. [5], which finds a way to separate the computation related to observed and unobserved data and thereby identify repetitive computations that can be easily avoided. A few methods have been proposed to further improve the efficiency of WRMF [12], [13], [14]. However, these methods are still designed for MF and are therefore not applicable to context-aware recommendation.

2.2 Context-Aware Recommender System

A context-aware recommender system exploits contextual information to better model the preference of a user [19]. Early work utilized contextual information based on pre-filtering to construct the data with specific context or post-filtering by adapting the contextual information from recommendation result [32]. Recent advances in context-aware recommendation focused on modeling user-item-context associations directly to predict the user preference under a specific context. A popular approach is to model the user behavior data as a tensor and apply TF to extend the factor model from second orders to higher orders. Factorization machine (FM) [7] is possibly the most successful factorization-based method for explicit feedback data. To deal with implicit feedback in context-aware recommendation, several learning-to-rank based methods have been proposed. We consider four such methods [16], [18], [21], [22] as baseline methods in our experiment and provide a brief description of them in Section 5.3.

Another notable related work is the one presented by Hidasi and Tikk [25], which improves the efficiency for learning from all the unobserved data for a factorization-based method called Tucker Decomposition (TD). Although it is an interesting method, it has been known that FM usually performs better than TD for context-aware recommendation [23] and, possibly because of that, our literature survey shows that this TD-based method is rarely used as a baseline in recent work [12], [13], [18], [21], [22], [24], [31]. We therefore decided to develop a new whole-data based method based on TF (with pairwise interaction) instead of TD.

3 Preliminaries

3.1 Formalization

We begin with a formalization of context-aware recommendation. Table 1 lists the main notations used in this paper.

User behavior data for context-aware recommendation can be represented as a tensor, $\mathcal{X} \in \mathbb{R}^{|U|\times|I|\times|C_1|\times|C_2|\times\cdots\times|C_G|}$, where U represents the set of users, I the set of items, and C_g the set of possible values for the gth type of context, and G the number of context types. The symbol $|\cdot|$ denotes the cardinality of a set. $\mathcal{X}$ is a $(G+2)$-dimensional tensor, with typically only a few number of entries observed. We denote the total number of possible entries in the tensor as $N = |U| \cdot |I| \cdot |C_1| \cdot \cdots \cdot |C_G|$, the number of observed entries as N_+, and typically $N_+ \ll N$. The goal of context-aware recommendation is to recommend items to a specific user, given knowledge of the context of the user.

Another way to represent the data for context-aware recommendation, as adopted by FM [7], flattens the tensor into a binary matrix $\mathbf{X} \in \{0,1\}^{N\times M}$ and a vector $\mathbf{y} \in \mathbb{R}^N$, where $M = |U| + |I| + |C_1| + \cdots + |C_G|$ denotes the total number of *objects* (i.e., users, items or contexts). Each row of the matrix, $\mathbf{X}_{n,:} \triangleq \mathbf{x}_n$, points to an entry in the tensor, as exemplified below:

$$\mathbf{x}_n = (\overbrace{1,0,\ldots,0}^{|U|},\overbrace{0,1,\ldots,0}^{|I|},\overbrace{1,\ldots,0}^{|C_1|},\overbrace{1,\ldots,0}^{\ldots},\overbrace{0,\ldots,1}^{|C_G|}). \quad (1)$$

In other words, the values in $\mathbf{x}_n$ can be divided into $G+2$ *groups*, and only an entry in each group is non-zero (i.e., it is a one-hot representation each group). The total number of nonzero entries in $\mathbf{x}_n$ is equal to $G+2$. On the other hand, y_n denotes the value corresponding to the entry pointed by $\mathbf{x}_n$. It can be either explicit or implicit feedback. We further use $\mathbf{X}_{:,m} \triangleq \mathbf{x}^{(m)}$ to denote a column in $\mathbf{X}$, indicating which data instances are related to the mth object. Moreover, we use $\mathbf{X}_+ \in \{0,1\}^{N_+\times M}$ and $\mathbf{y}_+ \in \mathbb{R}^{N_+}$ to denote the collection of observed data. We use $\mathcal{N}$ and $\mathcal{N}_+$ to denote the set of all

the entries and the set of observed entries, respectively. That is, we know $\mathbf{x}_n$ represents a row in $\mathbf{X}+$, if $n \in \mathcal{N}_+$.

For explicit feedback data, it is good enough to use $\mathbf{X}_+$ and $\mathbf{y}_+$ to train the recommender system. However, for implicit feedback data, oftentimes we need to draw negative data $\mathbf{X}_- \in \{0,1\}^{N_- \times M}$ and $\mathbf{y}_- \in \mathbb{R}^{N_-}$ from the unobserved data, and add these negative data to the training set. N_- denotes the number of unobserved data we use. For the whole-data based approach, $N_- = N - N_+$. As typically $y_n > 0$ for the observed data in an implicit feedback dataset, we set $y_n = 0$ for the unobserved data.

3.2 Tensor Factorization with Pairwise Interaction

The core idea of TF with pairwise interaction is to learn a latent representation $\mathbf{v}^{(m)} \in \mathbb{R}^K$ for each of the M objects, and use the inner product between the latent representations (a.k.a. second-order or pairwise interactions) to predict the association between users, items and contexts [23].The prediction function of TF with pairwise interaction can be defined as

$$\hat{y}_n = f(\mathbf{x}_n) = \sum_{m=1}^{M} \sum_{m'>m}^{M} x_{n,m} x_{n,m'} \langle \mathbf{v}^{(m)}, \mathbf{v}^{(m')} \rangle \,, \tag{2}$$

where $\hat{y}_n$ is the prediction of the input $\mathbf{x}_n$, $x_{n,m} \in \{0,1\}$ denotes the mth element in $\mathbf{x}_n$, and $\langle \mathbf{a}, \mathbf{b} \rangle \triangleq \mathbf{a}^T \mathbf{b}$ is the inner product between the two vectors $\mathbf{a}$ and $\mathbf{b}$. As $\mathbf{x}_n$ can be viewed as a concatenation of $G+2$ one-hot vectors, there are only $(G+2)(G+1)/2 \ll M^2$ such pairwise interactions. Therefore, the complexity of computing Eq. (2) is $\mathcal{O}(G^2K)$.

For explicit feedback dataset, we only need to consider the observed entries $\mathcal{N}_+$. To learn the latent representations, Rendle [7] proposed to use the following objective function:

$$\mathcal{L}_{\text{explicit}-\text{TF}} = \sum_{n \in \mathcal{N}_+} \big(y_n - f(\mathbf{x}_n)\big)^2 + \sum_{m=1}^{M} \lambda^{(m)} (\|\mathbf{v}^{(m)}\|^2) \,, \tag{3}$$

where the second term is a regularization term used to prevent overfitting, and $\lambda^{(m)} > 0$ is a regularization coefficient. Without loss of generalizability, we set $\lambda^{(m)} = \lambda$, $\forall m$. The computational complexity in computing Eq. (3) is $\mathcal{O}(G^2KN_+ + MK^2)$.

For implicit feedback dataset, it is easy to extend Eq. (3) to include all the unobserved data, by using

$$\mathcal{L}_{\text{implicit}-\text{TF}} = \sum_{n \in \mathcal{N}} \big(y_n - f(\mathbf{x}_n)\big)^2 + \lambda \sum_{m=1}^{M} (\|\mathbf{v}^{(m)}\|^2) \,. \tag{4}$$

Yet, as $N \gg N_+$, efficient ways to update the latent representations are needed. This is the subject of the next section.

4 Proposed Methods

We propose three methods to accelerate the learning of latent representations in TF for using all the unobserved data from implicit feedback, i.e., by reducing the computational cost associated with Eq. (4). These methods are extended from WRMF [5], with special designs to speed up learning from multi-dimensional user-item-context associations. As WRMF uses an ALS based solver, our methods also use ALS as the optimization method.

4.1 Fast TF (FTF)

WRMF speeds up MF by two core ideas: 1) applying a logarithmic scale to weigh both observed and unobserved data, and 2) avoiding repetitive computations by means of memorization [5].

FTF is a generalization of WRMF from the two-dimensional case (only users and items) to the multi-dimensional case (adding contexts). Following WRMF [5], we replace Eq. (4) with the following objective function to reduce the outliers and noises from implicit feedback

$$\mathcal{L}_{\text{FTF}} = \sum_{n \in \mathcal{N}} w_n \big(p_n - f(\mathbf{x}_n)\big)^2 + \lambda \sum_{m=1}^{M} (\|\mathbf{v}^{(m)}\|^2) \,, \tag{5}$$

where $p_n \in \{0,1\}$ is the preference value, $p_n = 1$ if $y_n > 0$ (i.e., observed) and $p_n = 0$ otherwise (i.e., unobserved); $w_n \geq 1$ is a weighting on (or the confidence value associated with) the nth data instance, computed by

$$w_n = 1 + \gamma \log(1 + y_n) \,, \tag{6}$$

where $\gamma > 0$ scales the influence of observed data. Clearly, $w_n = 1$ for all the unobserved data. This is a key observation we will make use of later on.

Taking the partial derivative of $\mathcal{L}_{\text{FTF}}$ with respect to $\mathbf{v}^{(m)}$ yields

$$\begin{aligned} \frac{\partial \mathcal{L}_{\text{FTF}}}{\partial \mathbf{v}^{(m)}} &= \sum_{n \in \mathcal{N}^{(m)}} -(\mathbf{z}_n'^m)^T w_n \Big(p_n - \mathbf{v}^{(m)} (\mathbf{z}_n'^m) - f(\mathbf{x}_n'^m)\Big) + \lambda \mathbf{v}^{(m)} \\ &= \Big(\sum_{n \in \mathcal{N}^{(m)}} w_n (\mathbf{z}_n'^m)^T (\mathbf{z}_n'^m) + \lambda \mathbf{I}_K \Big) \mathbf{v}^{(m)} - \sum_{n \in \mathcal{N}^{(m)}} e_n'^m \mathbf{z}_n'^m \,, \end{aligned}$$

where we have used the following shorthands for brevity:

- $\mathcal{N}^{(m)} \in \mathcal{N}$ denotes the set of entries in the tensor that is related to the mth object, including observed and unobserved data. For example, if the mth object corresponds to a user, $|\mathcal{N}^{(m)}| = |I| \cdot |C_1| \cdots |C_G|$. If it corresponds to an item, $|\mathcal{N}^{(m)}| = |U| \cdot |C_1| \cdots |C_G|$.
- $\mathbf{x}_n'^m \in \{0,1\}^M$ is a binary vector which is almost the same as $\mathbf{x}_n$, except that the mth element in $\mathbf{x}_n'^m$ is set to zero (in contrast, the mth element in $\mathbf{x}_n$ [i.e., $x_{n,m}$] must be equal to 1, because $n \in \mathcal{N}^{(m)}$).
- $\mathbf{z}_n'^m \in \mathbb{R}^K$ is a shorthand for $\sum_{j=1}^{M} x_{n,j}'^m \mathbf{v}^{(j)}$.
- $e_n'^m$ is a shorthand for $w_n(p_n - f(\mathbf{x}_n'^m))$, the weighted error when we use $\mathbf{x}_n'^m$ to estimate p_n.

Setting the above equation to zero and re-arranging the terms for $\mathbf{v}^{(m)}$ leads to the following update rule for learning $\mathbf{v}^{(m)}$

$$\mathbf{v}^{(m)} = \underbrace{\Big(\sum_{n \in \mathcal{N}^{(m)}} w_n \langle \mathbf{z}_n'^m, \mathbf{z}_n'^m \rangle + \lambda \mathbf{I}_K \Big)^{-1}}_{\Phi^{(m)}} \underbrace{\Big(\sum_{n \in \mathcal{N}^{(m)}} e_n'^m \mathbf{z}_n'^m \Big)}_{\Psi^{(m)}} \,, \tag{7}$$

where $\mathbf{I}_K$ is a K-by-K identify matrix, $\Phi^{(m)} \in \mathbb{R}^{K \times K}$ and $\Psi^{(m)} \in \mathbb{R}^K$ are another two shorthands. The complexity of computing $\Phi^{(m)}$ and $\Psi^{(m)}$ is $\mathcal{O}(GK^2|\mathcal{N}^{(m)}|)$ and $\mathcal{O}(G^2K|\mathcal{N}^{(m)}|)$, respectively. Taking into account the complexity in computing matrix inverse, the complexity of updating all the M latent representations (via Eq. (7)) is $\mathcal{O}(MK^3 + MGK^2N + MG^2KN)$, which increase linearly with MN. Footnote 1 gives an example of the time complexity for this for an artificial dataset.[1]

For efficiency, we follow the second core idea of WRMF and introduce an auxiliary variable $\tilde{w}_n = w_n - 1$ to identify repeated computations. In this way, we have

$$\Phi^{(m)} = \underbrace{\sum_{n \in \mathcal{N}^{(m)}} \tilde{w}_n \langle \mathbf{z}_n'^m, \mathbf{z}_n'^m \rangle}_{\ddot{\phi}^{(m)}} + \underbrace{\sum_{n \in \mathcal{N}^{(m)}} 1 \langle \mathbf{z}_n'^m, \mathbf{z}_n'^m \rangle}_{\dot{\phi}^{(m)}} \,. \tag{8}$$

This auxiliary variable is helpful because $\tilde{w}_n = 0$ for the unobserved variables. As can been seen, the computation of $\Phi^{(m)}$ is divided into two terms. The first term $\ddot{\phi}^{(m)}$ concerns only the

1. For a dataset that concerns the preference of 10^4 users for 10^4 items in 10^2 different locations, we have $N = 10^{10}$, $M \simeq 10^4$, and $G = 1$. Assuming 99.999 percent sparsity, we have $N_+ = 10^5$. For $K = 10$, the complexity of updating all the latent representations via Eq. (7) is in the order of 10^{16} (dominated by the term MGK^2N). With the two tricks in Eqs. (8) and (9), this is greatly reduced to 10^{12} (dominated by the term G^2K^2N). For DropTF, this is further reduced to 10^{11} (dominated by the term $MG^2K^2N_+$).

TABLE 2
Computational Complexity (of One Learning Iteration) of Different Methods for Context-Aware Recommendation

Method	Time complexity
FM (ALS) [7]	$\mathcal{O}(MK^3 + 2MG^2K^2(N_+ + N_-))$
GPPW [18]	$\mathcal{O}(GK(N_+ + N_-)^3/\lvert U\rvert^2)$
TF	$\mathcal{O}(MK^3 + MGK^2N + MG^2KN)$
FTF	$\mathcal{O}(MK^3 + MG^2K^2N_+ + G^2K^2N + G^3KN)$
FTF^{++}	$\mathcal{O}(MK^3 + MG^2K^2N_+ + G^2K^2M + G^3KN)$
DropTF	$\mathcal{O}(MK^3 + MG^2K^2N_+ + G^2K^2M)$

observed data, so the complexity is $\mathcal{O}(MGK^2N_+)$. The second term $\dot{\phi}^{(m)}$ concerns all the data, but we note that $\dot{\phi}^{(m)}$ is the same for all the objects belonging to the same group, because $\mathbf{x}_n^{/m} = \mathbf{x}_n^{/m'}$ (and therefore $\mathbf{z}_n^{/m} = \mathbf{z}_n^{/m'}$) if the mth and m'th objects are within the same group. We only need to compute $\dot{\phi}^{(m)}$ once (and then have it "memorized") for each of the G groups, so the complexity becomes $\mathcal{O}(G^2K^2N)$. Overall, the complexity is reduced from $\mathcal{O}(MGK^2N)$ to $\mathcal{O}(MGK^2N_+ + G^2K^2N)$, avoiding the dominant MN term.

Similarly, the computation of $\Psi^{(m)}$ can be rewritten as

$$\Psi^{(m)} = \underbrace{\sum_{n\in\mathcal{N}^{(m)}} w_n \langle \mathbf{z}_n^{/m}, \mathbf{z}_n^{/m}\rangle p_n}_{\ddot{\psi}^{(m)}} - \underbrace{\sum_{n\in\mathcal{N}^{(m)}} w_n \langle \mathbf{z}_n^{/m}, \mathbf{z}_n^{/m}\rangle f(\mathbf{x}_n^{/m})}_{\dot{\psi}^{(m)}} . \quad (9)$$

By definition of p_n, the computation of $\ddot{\psi}^{(m)}$ concerns only observed data. Moreover, the computation of $\dot{\psi}^{(m)}$ can be memorized and computed only once per group. Overall, the complexity is reduced from $\mathcal{O}(MG^2KN)$ to $\mathcal{O}(MG^2KN_+ + G^3KN)$. See Footnote 1 for an example of the effect of avoiding the MN term by Eqs. (8) and (9).

Although the term $\dot{\phi}^{(m)}$ and $\dot{\psi}^{(m)}$ can be memorized, the complexity still depends linearly on N, which can be extremely large for context-aware recommendation. We describe a faster FTF method below to tackle this bottleneck.

4.2 Faster FTF (FTF^{++})

The main idea of this new method is to use the binomial theorem to speed up the computation of $\dot{\phi}^{(m)}$ in Eq. (8)

$$\begin{aligned}\dot{\phi}^{(m)} &= \sum_{n\in\mathcal{N}^{(m)}} \langle \mathbf{z}_n^{/m}, \mathbf{z}_n^{/m}\rangle = \sum_{n\in\mathcal{N}^{(m)}} \left(\sum_{j=1}^{M}\sum_{j'=1}^{M} x_{n,j}^{/m} x_{n,j'}^{/m} \langle \mathbf{v}^{(j)}, \mathbf{v}^{(j')}\rangle \right) \\ &= \sum_{g}\sum_{g'} (\mathcal{V}^{(g)})^T \mathbf{H}^{(g,g')} (\mathcal{V}^{(g')}) ,\end{aligned} \quad (10)$$

where both g and g' run through all the $G+2$ groups, excluding the group that the mth object belongs to. We note that Eq. (10) contains the following new variables:

- $\mathcal{G}^{(g)} \in [1, M]$ denotes the set of objects that belong to the gth group. For example, $\mathcal{G}^{(1)} = [1, |U|]$, $\mathcal{G}^{(2)} = [|U|+1, |U|+|I|]$, $\mathcal{G}^{(G+2)} = [M - |C_G| + 1, M]$.
- $\mathcal{V}^{(g)} \in \mathbb{R}^{|\mathcal{G}^{(g)}|\times K}$ is a matrix that contains all the latent representations of the objects in $\mathcal{G}^{(g)}$. For example, $\mathcal{V}^{(1)} = [\mathbf{v}^{(1)}; \ldots; \mathbf{v}^{(|U|)}]$
- $\mathbf{H}^{(g,g')} \in \mathbb{R}^{|\mathcal{G}^{(g)}|\times|\mathcal{G}^{(g')}|}$ is a matrix that memorizes the number of repeated computations in computing $\dot{\phi}^{(m)}$ by the original equation $\sum_n \langle \mathbf{z}_n^{/m}, \mathbf{z}_n^{/m}\rangle$. For example, $H_{i,j}^{(1,2)}$ denotes how many times we need to (repetitively) compute $x_{n,i}^{/m} x_{n,j+|U|}^{/m} \langle \mathbf{v}^{(i)}, \mathbf{v}^{(j+|U|)}\rangle$.

These repetitive matrix-vector computations can be avoided. For $g = g'$, $\mathbf{H}^{(g,g')}$ will be a scalar matrix $\alpha \mathbf{I}_{|\mathcal{G}^{(g)}|}$, where $\alpha = |\mathcal{N}^{(m)}|/|\mathcal{G}^{(g)}|$. Accordingly, we have

$$(\mathcal{V}^{(g)})^T \mathbf{H}^{(g,g')} (\mathcal{V}^{(g)}) = \alpha (\mathcal{V}^{(g)})^T (\mathcal{V}^{(g)}) . \quad (11)$$

For $g \neq g'$, $\mathbf{H}^{(g,g')}$ will be a matrix where every element is equal to $\beta = |\mathcal{N}^{(m)}|/(|\mathcal{G}^{(g)}| \cdot |\mathcal{G}^{(g')}|)$, leading to

$$\begin{aligned}(\mathcal{V}^{(g)})^T \mathbf{H}^{(g,g')} (\mathcal{V}^{(g')}) &= \left(\mathcal{V}^{(g)} \begin{bmatrix} \beta \\ \vdots \\ \beta \end{bmatrix} \right) ([1 \quad \cdots \quad 1] \mathcal{V}^{(g')}) \\ &= \beta \left(\sum_{h=1}^{|\mathcal{G}^{(g)}|} \mathcal{V}_h^{(g)} \right)^T \left(\sum_{h=1}^{|\mathcal{G}^{(g')}|} \mathcal{V}_h^{(g')} \right) ,\end{aligned} \quad (12)$$

where $\mathcal{V}_h^{(g)} \in \mathbb{R}^K$ is the hth column of the matrix $\mathcal{V}_h^{(g)}$. We need to compute Eq. (11) $G+1$ times, with total complexity $\mathcal{O}(K^2 \sum_{g=1}^{G+2} |\mathcal{G}^{(g)}|) = \mathcal{O}(K^2M)$. And, we need to compute Eq. (12) approximately G^2 times, with total complexity $\mathcal{O}(\sum_g \sum_{g\neq g'} (K^2 + |\mathcal{G}^{(g)}| + |\mathcal{G}^{(g')}|) = \mathcal{O}(G^2K^2M)$. Accordingly, the complexity of computing $\dot{\phi}^{(m)}$ can be greatly reduced from $\mathcal{O}(G^2K^2N)$ to $\mathcal{O}(K^2M + G^2K^2M) = \mathcal{O}(G^2K^2M)$.

Table 2 summarizes the computational complexity (of one learning iteration) of different methods.

Algorithm 1. DropTF

Require: $\mathcal{N}_+$, $\mathbf{X}$ and λ
Randomly initialize $\mathbf{v}^{(1)}, \mathbf{v}^{(2)}, \ldots, \mathbf{v}^{(M)}$ and $k = 0$
repeat
  for $m = 1, \ldots, M$ **do**
    if $m > \sum_{j=0}^{k} |\mathcal{G}^{(j)}|$ **then** ▷ $|\mathcal{G}^{(0)}| = 1$
      $\dot{\phi}^{(m)} \leftarrow K \times K$ zero matrix
      for $g \in \{1, \ldots, G+2\} \wedge g \neq k+1$ **do**
        for $g' \in \{1, \ldots, G+2\} \wedge g' \neq k+1$ **do**
          if $g = g'$ **then**
            $\alpha \leftarrow \alpha \cdot (|\mathcal{N}^{(m)}|/|\mathcal{G}^{(g)}|)$
            $\dot{\phi}^{(m)} \leftarrow \dot{\phi}^{(m)}$+ Eq. (11)
          else
            $\beta \leftarrow \beta \cdot (|\mathcal{N}^{(m)}|/(|\mathcal{G}^{(g)}| \cdot |\mathcal{G}^{(g')}|))$
            $\dot{\phi}^{(m)} \leftarrow \dot{\phi}^{(m)}$+ Eq. (12)
          end if
        end for
      end for
      $k \leftarrow k + 1$
    end if
    $\mathbf{v}^{(m)} \leftarrow (\ddot{\phi}^{(m)} + \dot{\phi}^{(m)} + \lambda \mathbf{I}_\mathbf{K})^{-1} \ddot{\psi}^{(m)}$
  end for
until stopping criterion is met
return $\mathbf{v}^{(1)}, \mathbf{v}^{(2)}, \ldots, \mathbf{v}^{(M)}$

4.3 DropTF

We can further reduce the cost of computing $\dot{\psi}^{(m)}$ in Eq. (9) by *dropping out* some pairwise interactions between users, items and contexts in learning the latent representation $\mathbf{v}^{(m)}$. Specifically, instead of considering all the pairwise interactions in predicting p_n, as in the case of the TF (cf. Eq. (2)), we use *only the pairwise interactions that involve a target object (i.e., users, items or contexts), whose latent representation is to be updated*. That is, to update $\mathbf{v}^{(m)}$, we use the following prediction function:

$$f^{(m)}(\mathbf{x}_n) = \sum_{j=1}^{M} x_{n,m} x_{n,j}^{/m} \langle \mathbf{v}^{(m)}, \mathbf{v}^{(j)}\rangle . \quad (13)$$

Accordingly, the loss function has a formulation that depends on m and looks like

TABLE 3
Statistics of the Datasets Used in Our Experiment

Dataset	$\lvert U\rvert$	$\lvert I\rvert$	G	$\sum_{g=1}^{G}\lvert C_g\rvert$	N_+
CoMoDa [27]	111	894	3	17	1,492
Frappe [28]	953	4,073	1	42	95,888
KKBOX	37 k	743 k	1	70 k	82 M

$$\mathcal{L}^{(m)}_{\text{DropTF}} = \sum_{n\in\mathcal{N}} w_n\big(p_n - f^{(m)}(\mathbf{x}_n)\big)^2 + \lambda\sum_{m=1}^{M}(\|\mathbf{v}^{(m)}\|^2)\,. \tag{14}$$

In this way, the prediction becomes "thinner". Moreover, we are now using in total M such thinner sub-models now, one for the update of each $\mathbf{v}^{(m)}$. Although they are updated separately, the M sub-models share the same latent representations for all objects $\{\mathbf{v}^{(m)}\}_{m=1}^{M}$ throughout the training process. This way, we may avoid co-adapting the variables too much during training, akin to the case of using the so-called *dropout* mechanism for training deep neural networks [33], [34]. But, while the dropout mechanism *randomly* drops units out from a neural network, in our DropTF algorithm the selection of pairwise interactions to be excluded is *deterministic* and depends on the target object.

Taking the partial derivative of $\mathcal{L}^{(m)}_{\text{DropTF}}$ with respect to $\mathbf{v}^{(m)}$ leads to the following update rule:

$$\mathbf{v}^{(m)} = (\Phi^{(m)} + \lambda\mathbf{I}_K)^{-1}\ddot{\psi}^{(m)}\,. \tag{15}$$

Comparing to Eq. (7), everything remains the same, except that the term $\dot{\psi}^{(m)}$ in Eq. (9) is gone. This is because this particular term is from the computation of pairwise interactions that are not related to the target object.

Following the idea of dropout [34], in the testing step we can also aggregate the prediction of the ensemble of all these M sub-models by $\frac{1}{G+2}\sum_{m=1}^{M} f^{(m)}(\mathbf{x}_n)$, which is actually equivalent to $\frac{1}{2}f(\mathbf{x}_n)$. Therefore, the cost for testing is the same as FTF^{++}.

We show in Algorithm 1 the pseudo code of DropTF. We will open source our implementation of DropTF upon acceptance.

5 Experimental Setup

5.1 Dataset

Our experiments were conducted using three datasets. Key statistics of the datasets are listed in Table 3.

CoMoDa [27] is an explicit dataset and it contains the user ratings (from 1 to 5) for 1,232 movies by 121 users, totaling 2,296 ratings. In addition to the ratings, the users also provided contextual information of 12 different types, from which we considered only the following three for simplicity: ending and dominant emotions of the users, and mood factors of the movies. Following the setting used by Nguyen et al. [18], we converted this dataset into an implicit feedback one by treating ratings larger than 3 as positive data and all the others as negative data. Moreover, we randomly split the data into three subsets: 70 percent for training, 10 percent for validation, and 20 percent for testing. The validation set was used to determine hyper-parameters, e.g., we empirically set $\lambda = 100$ and $\gamma = 100$ throughout the experiment.

Frappe [28] is an implicit feedback dataset that records whether a user has used an App in a specific context. There are 953 users, 4,073 Android apps, and 42 different contexts, which are related to time-of-the-day, day-of-the-week, location (unknown, home, workplace), and weather. There are in total 61,465 observed data instances. For Frappe, we followed the settings suggested by Shi et al. [22] to split the data into training, validation, and test sets.

Finally, as existing public domain datasets are mostly not big, we obtained an implicit feedback dataset through a collaboration with an international music streaming company called KKBOX (https://www.kkbox.com/). The resulting dataset contains the playcounts for 653 k songs by 37 k users from Taiwan during eight consecutive months, totaling 57 M observed data entries. For this dataset, we used the artist ID of the most recently played song as the contextual information. The ratio of training, validation and test sets is the same as that of CoMoDa.

5.2 Performance Metric

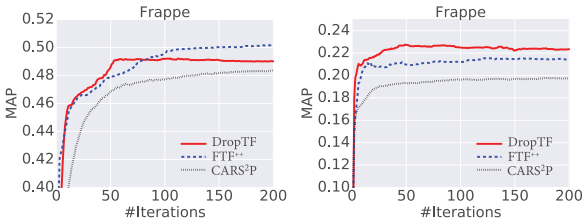
A common approach to evaluate the accuracy (or relevance) of a recommender system that uses implicit feedback is through the so-called top-N recommendation task [29]. From a candidate set that includes both the positive and negative items for a specific user in the test set, the task is to compute a ranking list of the items in descending order of estimated user preference, with the positive data at the top of the list. For Frappe, we followed the settings used by Shi et al. [22] and randomly selected 1,000 negative items to add to the candidate set for user under each context, during both the validation and testing period. We did the same for the KKBOX dataset. For the smallest CoMoDa dataset, we used only 10 negative items per user per context, again following the setting used by Nguyen et al. [18].

5.3 Baseline Methods

We compared the performance of DropTF against a few baselines.

- *POP*: A simple method that generates the recommended list by sorting the items according to item popularity, which is measured by counting the number of observed user-item pairs for each item.
- *FM* [7]: A state-of-the-art method for context-aware recommendation from explicit feedback. We took the negative sampling approach and used Eq. (4) as the cost function.
- Bayesian personalized ranking (*BPR*) [16]: A widely-used learning-to-rank based method for general, context-agnostic recommendation from implicit feedback. In other words, the contextual information is not considered at all. We note that POP is also context-agnostic.
- Tensor factorization for MAP (*TFMAP*) [21]. This method is designed for context-aware recommendation from implicit feedback, using a smoothed version of MAP to learn latent representations via tensor factorization. For efficiency, it exploits several intrinsic properties of average precision (AP) to design a fast learning algorithm.
- *CARS*^{2} [22]. This method builds on two layers of latent factors to learn context-aware user/item representations. They first layer models the factors of user-item association, whereas the second layer learns two parameters to predict the user-item association under a specific context. This method uses either a pairwise or a listwise ranking loss functions for learning. We refer to the resulting models as CARS^{2}P and CARS^{2}L, respectively.
- Gaussian process pairwise preference model (*GPPW*) [18]. A pairwise learning model that uses Gaussian process (which allows the use of flexible covariance functions) to model complex, non-linear user-item-context interactions. Scalability issues are addressed by using a stochastic gradient descent (SGD) based learner.
- Low-rank matrix completion problem over finite Abelian group algebras (*MC-AGA*) [15]. The method uses matrices over finite Abelian group algebra (AGA) to model context-aware interactions between users and items. It considers all the unobserved data as negative in the learning process.

We used the performance reported by the respective authors in their papers if the evaluation setting is the same. Otherwise, we implemented these methods based on the source codes shared by



(a) Training set (b) Test set

Fig. 1. Performance of MAP on either the (a) training or (b) test set of Frappe [28], as a function of learning iterations.

TABLE 4
Experimental Result on the Frappe Dataset

Method	MAP	P@5	P@10	R@5	R@10
POP	7.6%	4.3%	2.8%	12.1%	15.3%
BPR [16]	12.1%	6.1%	3.7%	16.0%	18.4%
TFMAP [21]	12.7%	6.1%	3.9%	15.9%	19.5%
CARS^{2}P [22]	14.0%	6.8%	4.4%	18.3%	22.6%
CARS^{2}L [22]	12.9%	6.2%	4.0%	16.5%	20.2%
FM [7]	13.9%	5.8%	3.8%	20.7%	26.5%
GPPW [18]	14.7%	8.1%	5.2%	26.3%	30.8%
MC-AGA [15]	15.5%	**8.9%**	5.7%	19.9%	24.1%
DropTF	**16.3%**	8.4%	**5.8%**	**26.6%**	**33.6%**

the authors and tuned the hyper-parameters using the validation set. Unless otherwise specified, we fixed $K = 10$, as this is the value widely used in these prior arts. Moreover, for all the latter five learning-based methods, we randomly drew 10 unobserved user-item-context tuples for each user as the negative data.

6 Experimental Result

We addressed the following questions in this evaluation:

- Does DropTF indeed learn better than FTF^{++}?
- How does DropTF perform in effectiveness and efficiency as compared with the baseline methods?

6.1 Evaluation on Effectiveness

6.1.1 DropTF versus FTF^{++}

Fig. 1 compares the MAP obtained by DropTF, FTF^{++}, and CARS^{2}P as a function of learning iterations, for the training and test sets of Frappe.[2] For this comparison, we set $K = 30$ as these methods run fast enough to use a larger K. As Fig. 1 shows, the learning curves follow a similar trend—the MAP improves rapidly in the first few iterations and then reaches a plateau. We saw that DropTF and FTF^{++} achieve higher MAP than CARS^{2}P in both the training and test sets. More interestingly, we saw that FTF^{++} outperforms DropTF in later iterations for the training set, but the situation reverses for the test set. This result provides an evidence of the effectiveness of the proposed "dropping out" technique in preventing overfitting. We also compared DropTF with FTF^{++} on a subset of KKBOX and found similar result (not reported here).

6.1.2 DropTF versus Baseline Methods

Tables 4, 5, and 6 compare the accuracy of DropTF against that of different baseline methods on the three datasets, respectively. We evaluate the accuracy (in %) in terms of MAP, recall (R@) and precision (P@) at a particular rank. For the Frappe dataset, Table 4 shows that DropTF performs remarkably better than all the prior arts. Notably, we can find whole-data based approaches (i.e., MC-AGA and DropTF) outperform negative sampling approach,[3] suggesting that the performance can be boosted by considering more unobserved data.[4] Similarly, as Table 5 shows, DropTF outperforms FM and GPPW for the CoMoDa dataset in most performance metrics.[5] These results demonstrate the effectiveness of a whole-data based learning method for recommendation from implicit feedback.

TABLE 5
Experimental Result on the CoMoDa Dataset

Method	MAP	P@5	P@10	R@5	R@10
POP	40.7%	13.5%	8.2%	52.8%	65.0%
BPR [16]	41.6%	15.0%	10.2%	59.3%	80.5%
FM [7]	42.5%	15.8%	10.9%	64.4%	86.2%
GPPW [18]	46.2%	16.3%	**12.2%**	63.2%	**90.9%**
DropTF	**50.2%**	**17.1%**	11.0%	**71.5%**	87.2%

TABLE 6
Experimental Result on the KKBOX Dataset

Method	MAP	P@5	P@10	R@5	R@10
POP	11.1%	7.1%	6.4%	22.6%	38.0%
BPR [16]	11.2%	7.4%	6.3%	23.6%	37.7%
FM [7]	18.1%	10.3%	7.7%	30.3%	43.8%
DropTF	**22.3%**	**11.8%**	**8.1%**	**33.1%**	**44.2%**

Finally, for the much larger KKBOX dataset, we were only able to compare DropTF against POP, BPR and FM, for all the other methods (including the other whole-date based method MC-AGA) are too slow to run on this dataset, despite that many of these methods do not use all the unobserved data for learning.[6] Table 6 again shows that DropTF is superior to FM and the two context-agnostic methods, with great performance margin.

To sum up, some similar trends can be observed from Tables 4, 5, and 6. First, as expected, context-aware methods outperform context-agnostic models (i.e., POP and BPR; first two rows in these tables) most of the time. Second, learning-to-rank based methods can reach performance on par with, or even superior to, FM. Finally, the proposed DropTF method enjoys high scalability as its precursor TF and it consistently performs the best in most performance metrics across the three datasets.

6.2 Evaluation on Efficiency

Next, we compared the runtime of different methods in one iteration of the training process. To see how the runtime increases as a function of data size, we sampled different proportions (from 10 to 100 percent) of the training set of either Frappe or KKBOX for training. We compared the runtime of DropTF and FTF^{++} against that of the better performing methods, including FM, CARS^{2}P and GPPW. The following observations can be made from Fig. 2. First, although GPPW performs well in performance metrics such as MAP for Frappe (and also the smaller CoMoDa), it is much slower than the other methods. Moreover, we saw that its runtime

2. We did not consider FTF in this evaluation for it is just a slower version of FTF^{++} with presumably no difference in performance metrics such as MAP.

3. DropTF and MC-AGA also outperform approaches that do not use any negative data at all. For example, the performance of FM using only observed data achieved only 12.4 percent MAP on the Frappe dataset.

4. The result of the two context-agnostic methods (POP and BPR) and four context-aware models (TFMAP, CARS^{2}P, CARS^{2}L and MC-AGA) were directly cited from that reported in [15].

5. We did not show the result of TFMAP, CARS and MC-AGA in Table 5, for they did not work well for CoMoDa, possibly because the dataset is small.

6. We note that the optimization procedures of the learning-to-rank based methods are usually quite complicated [35]. As a result, their scalability turns out to be not as good as methods such as FM.

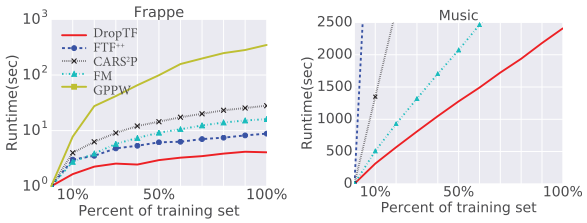


Fig. 2. Training time (in seconds) of an iteration of different methods on the (a) Frappe and (b) KKBOX datasets.

increases almost exponentially along with the data size (noting that the y-axis in Fig. 2a is in logarithmic scale). It is therefore not surprising that it cannot be used for the KKBOX dataset. Second, the runtime of the other methods increases mostly linearly as the training data expands, which is fine. Third, while FTF^{++} runs fast for Frappe, it is quite slow for KKBOX. Finally, FM and DropTF run fairly fast for both Frappe and KKBOX. In particular, DropTF is the most time-efficient one among the five methods, and it is faster than FM by one to two folds.

7 Conclusion and Future Work

In this paper, we have presented a fast tensor factorization method, called DropTF, for context-aware recommendation from implicit feedback data. As a whole-data based learning algorithm, DropTF can efficiently learn from the vast amount of unobserved data. Our experiment on three datasets shows that DropTF outperforms several well-known methods for recommendation in performance metrics such as MAP, and it is also the fastest one. Our experiment already shows that DropTF performs well in both efficiency and effectiveness for a million-scale dataset. If needed, the training process of DropTF can be further parallelized, making it a good choice for large-scale industrial applications.

We have several ideas to further improve DropTF. For example, we can adopt the non-uniform weight strategy proposed by He et al. [13] to weigh the unobserved data differently. To use more variables in the latent representation (i.e., larger K), we can employ the element-wise ALS method [13] for better efficiency. We are also interested in combining DropTF with other context-aware recommendation methods such as item embedding [31]. We can also use neural network based methods (e.g., [36], [37]) to include numerical, content-related features to build a hybrid recommender system. The use of content features can deal with the so-called cold-start problem [38], [39], which is not addressed in this paper. It can also potentially make the recommendation result more diverse [40]. Finally, it is also possible to use DropTF to get an initial estimate of the latent representations and then employ more sophisticated methods such as deep neural network based methods [41], [42], [43] for fine-tunning and better accuracy.

Acknowledgments

We thank company KKBOX Inc. for providing the KKBOX dataset used in our experiment. This work was supported by a grant from the Ministry of Science and Technology of Taiwan under contract MOST 106-3114-E-002-007.

References

[1] S. Jiang, X. Qian, T. Mei, and Y. Fu, "Personalized travel sequence recommendation on multi-source big social media," *IEEE Trans. Big Data*, vol. 2, no. 1, pp. 43–56, Mar. 2016.

[2] J. D. West, I. Wesley-Smith, and C. T. Bergstrom, "A recommendation system based on hierarchical clustering of an article-level citation network," *IEEE Trans. Big Data*, vol. 2, no. 2, pp. 113–123, Jun. 2016.

[3] S. Huang, J. Zhang, L. Wang, and X. S. Hua, "Social friend recommendation based on multiple network correlation," *IEEE Trans. Multimedia*, vol. 18, no. 2, pp. 287–299, Feb. 2016.

[4] J. Bennett and S. Lanning, "The Netflix prize," in *Proc. KDD Cup Workshop*, 2007, pp. 3–6.

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

[6] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Comput.*, vol. 42, no. 8, pp. 30–37, 2009.

[7] S. Rendle, "Factorization machines with libFM," *ACM Trans. Intell. Syst. Technol.*, vol. 3, no. 3, pp. 57:1–57:22, May 2012.

[8] D. Parra and X. Amatriain, "Walk the talk: Analyzing the relation between implicit and explicit feedback for preference elicitation," in *Proc. Int. Conf. User Model. Adaption Personalization*, 2011, pp. 255–268.

[9] X. Yi, L. Hong, E. Zhong, N. N. Liu, and S. Rajan, "Beyond clicks: Dwell time for personalization," in *Proc. ACM Conf. Recommender Syst.*, 2014, pp. 113–120.

[10] D. Liang, R. G. Krishnan, M. D. Hoffman, and T. Jebara, "Variational autoencoders for collaborative filtering," in *Proc. Int. Conf. World Wide Web*, 2018, pp. 51–61.

[11] R. Pan, Y. Zhou, B. Cao, N. N. Liu, R. M. Lukose, M. Scholz, and Q. Yang, "One-class collaborative filtering," in *Proc. IEEE Int. Conf. Multimedia Expo*, 2008, pp. 502–511.

[12] H.-F. Yu, M. Bilenko, and C.-J. Lin, "Selection of negative samples for one-class matrix factorization," in *Proc. SIAM Int. Conf. Data Mining*, 2017, pp. 363–371.

[13] X. He, H. Zhang, M.-Y. Kan, and T.-S. Chua, "Fast matrix factorization for online recommendation with implicit feedback," in *Proc. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2016, pp. 549–558.

[14] I. Pilászy, D. Zibriczky, and D. Tikk, "Fast ALS-based matrix factorization for explicit and implicit feedback datasets," in *Proc. ACM Conf. Recommender Syst.*, 2010, pp. 71–78.

[15] C.-A. Yu, T.-S. Chan, and Y.-H. Yang, "Low-rank matrix completion over finite Abelian group algebras for context-aware recommendation," in *Proc. ACM Int. Conf. Inf. Knowl. Manage.*, 2017, pp. 2415–2418.

[16] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thiem, "BPR: Bayesian personalized ranking from implicit feedback," in *Proc. Conf. Uncertainty Artif. Intell.*, 2009, pp. 452–461.

[17] R. He, W.-C. Kang, and J. McAuley, "Translation-based recommendation," in *Proc. ACM Conf. Recommender Syst.*, 2017, pp. 161–169.

[18] T. V. Nguyen, A. Karatzoglou, and L. Baltrunas, "Gaussian process factorization machines for context-aware recommendations," in *Proc. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2014, pp. 63–72. [Online]. Available: http://trungngv.github.io/gpfm/

[19] G. Adomavicius, B. Mobasher, F. Ricci, and A. Tuzhilin, "Context-aware recommender systems," *AI Mag.*, vol. 32, no. 3, pp. 67–80, 2011.

[20] S. Wang, L. Hu, L. Cao, X. Huang, D. Lian, and W. Liu, "Attention-based transactional context embedding for next-item recommendation," in *Proc. AAAI Conf. Artif. Intell.*, 2018, pp. 14–22.

[21] Y. Shi, A. Karatzoglou, L. Baltrunas, M. Larson, A. Hanjalic, and N. Oliver, "TFMAP: Optimizing MAP for top-N context-aware recommendation," in *Proc. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2012, pp. 155–164.

[22] Y. Shi, A. Karatzoglou, L. Baltrunas, M. Larson, and A. Hanjalic, "CARS2: Learning context-aware representations for context-aware recommendations," in *Proc. ACM Int. Conf. Inf. Knowl. Manage.*, 2014, pp. 291–300. [Online]. Available: http://baltrunas.info/research-menu/frappe

[23] S. Rendle and L. Schmidt-Thieme, "Pairwise interaction tensor factorization for personalized tag recommendation," in *Proc. ACM Int. Conf. Web Search Data Mining*, 2010, pp. 81–90.

[24] S. Rendle and C. Freudenthaler, "Improving pairwise learning for item recommendation from implicit feedback," in *Proc. ACM Int. Conf. Web Search Data Mining*, 2014, pp. 273–282.

[25] B. Hidasi and D. Tikk, "Fast ALS-based tensor factorization for context-aware recommendation from implicit feedback," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discovery Databases*, 2012, pp. 67–82.

[26] A. Karatzoglou, X. Amatriain, L. Baltrunas, and N. Oliver, "Multiverse recommendation: N-dimensional tensor factorization for context-aware collaborative filtering," in *Proc. ACM Conf. Recommender Syst.*, 2010, pp. 79–86.

[27] A. Košir, A. Odic, M. Kunaver, M. Tkalcic, and J. F. Tasic, "Database for contextual personalization," *Elektrotehniški Vestnik*, vol. 78, no. 5, pp. 270–274, 2011.

[28] L. Baltrunas, K. Church, A. Karatzoglou, and N. Oliver, "Frappe: Understanding the usage and perception of mobile app recommendations in-the-wild," *CoRR*, vol. abs/1505.03014, 2015. [Online]. Available: http://arxiv.org/abs/1505.03014

[29] P. Cremonesi, Y. Koren, and R. Turrin, "Performance of recommender algorithms on top-N recommendation tasks," in *Proc. ACM Conf. Recommender Syst.*, 2010, pp. 39–46.

[30] H.-F. Yu, C.-J. Hsieh, S. Si, and I. S. Dhillon, "Parallel matrix factorization for recommender systems," *Knowl. Inf. Syst.*, vol. 41, pp. 793–819, 2014.

[31] D. Liang, J. Altosaar, L. Charlin, and D. M. Blei, "Factorization meets the item embedding: Regularizing matrix factorization with item co-occurrence," in *Proc. ACM Conf. Recommender Syst.*, 2016, pp. 59–66.

[32] G. Adomavicius, R. Sankaranarayanan, S. Sen, and A. Tuzhilin, "Incorporating contextual information in recommender systems using a multidimensional approach," *ACM Trans. Inf. Syst.*, vol. 23, no. 1, pp. 103–145, 2005.

[33] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
[34] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, pp. 1929–1958, 2014.
[35] D. Sculley, "Large scale learning to rank," in *Proc. NIPS Workshop Advances Ranking*, 2009.
[36] A. V. D. Oord, S. Dieleman, and B. Schrauwen, "Deep content-based music recommendation," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2013, pp. 2643–2651.
[37] P. Covington, J. Adams, and E. Sargin, "Deep neural networks for YouTube recommendations," in *Proc. ACM Conf. Recommender Syst.*, 2016, pp. 191–198.
[38] M. Zhang, J. Tang, X. Zhang, and X. Xue, "Addressing cold start in recommender systems: A semi-supervised co-training algorithm," in *Proc. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2014, pp. 73–82.
[39] S.-Y. Chou, Y.-H. Yang, J.-S. Jang, and Y.-C. Lin, "Addressing cold start for next-song recommendation," in *Proc. ACM Conf. Recommender Syst.*, 2016, pp. 115–118.
[40] S.-Y. Chou, Y.-H. Yang, and Y.-C. Lin, "Evaluating music recommendation in a real-world setting: On data splitting and evaluation metrics," in *Proc. IEEE Int. Conf. Multimedia Expo*, 2015, pp. 1–6.
[41] X. He and T.-S. Chua, "Neural factorization machines for sparse predictive analytics," in *Proc. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2017, pp. 355–364.
[42] T. Chen, Y. Sun, Y. Shi, and L. Hong, "On sampling strategies for neural network-based collaborative filtering," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2017, pp. 767–776.
[43] S. Okura, Y. Tagami, S. Ono, and A. Tajima, "Embedding-based news recommendation for millions of users," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2017, pp. 1933–1942.