Efficient Non-Sampling Factorization Machines for Optimal Context-Aware Recommendation

Chong Chen, Min Zhang, Weizhi Ma, Yiqun Liu, and Shaoping Ma
Department of Computer Science and Technology, Institute for Artificial Intelligence,
Beijing National Research Center for Information Science and Technology, Tsinghua University
cc17@mails.tsinghua.edu.cn,z-m@tsinghua.edu.cn

ABSTRACT

To provide more accurate recommendation, it is a trending topic to go beyond modeling user-item interactions and take context features into account. Factorization Machines (FM) with negative sampling is a popular solution for context-aware recommendation. However, it is not robust as sampling may lost important information and usually leads to non-optimal performances in practical. Several recent efforts have enhanced FM with deep learning architectures for modelling high-order feature interactions. While they either focus on rating prediction task only, or typically adopt the negative sampling strategy for optimizing the ranking performance. Due to the dramatic fluctuation of sampling, it is reasonable to argue that these sampling-based FM methods are still suboptimal for context-aware recommendation.

In this paper, we propose to learn FM without sampling for ranking tasks that helps context-aware recommendation particularly. Despite effectiveness, such a non-sampling strategy presents strong challenge in learning efficiency of the model. Accordingly, we further design a new ideal framework named Efficient Non-Sampling Factorization Machines (ENSFM). ENSFM not only seamlessly connects the relationship between FM and Matrix Factorization (MF), but also resolves the challenging efficiency issue via novel memorization strategies. Through extensive experiments on three real-world public datasets, we show that 1) the proposed ENSFM consistently and significantly outperforms the state-of-the-art methods on context-aware Top-K recommendation, and 2) ENSFM achieves significant advantages in training efficiency, which makes it more applicable to real-world large-scale systems. Moreover, the empirical results indicate that a proper learning method is even more important than advanced neural network structures for Top-K recommendation task. Our implementation has been released [1] to facilitate further developments on efficient non-sampling methods.

CCS CONCEPTS

• Information systems → Recommender systems; **• Computing methodologies → Factorization methods**.

[1]https://github.com/chenchongthu/ENSFM

KEYWORDS

Factorization Machines, Top-K Recommendation, Context-aware, Efficient Learning, Implicit Feedback

ACM Reference Format:
Chong Chen, Min Zhang, Weizhi Ma, Yiqun Liu, and Shaoping Ma. 2020. Efficient Non-Sampling Factorization Machines for Optimal Context-Aware Recommendation. In *Proceedings of The Web Conference 2020 (WWW '20), April 20–24, 2020, Taipei, Taiwan.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3366423.3380303

1 INTRODUCTION

With the continuous development of the Internet, information explosion has become a great challenge that people are faced with [4, 36]. Keeping pace with the growing requirements of customization and personalization, recommender systems that are capable of learning fine-grained individual preferences with a concise, flexible, and efficient structure are eagerly expected.

Context-aware recommender systems exploit contextual information such as user demographics, item attributes, and time/location of the current transaction to personalize item recommendation for users [2, 35, 46]. To learn from such context-rich data, several effective models have been proposed, among which Factorization Machines (FM) [31] gains significant attention from researchers [15, 16, 23, 44, 46] due to its elegant theory in seamless integration of sparse context and increasing popularity in industrial applications.

FM is originally developed for rating prediction task, which is based on explicit user feedback (e.g., ratings). However, most observed user feedback is implicit in practice [22, 34], such as views, clicks, and purchases. Besides, previous literature [12] has pointed out that algorithms optimized for rating prediction may not perform well on Top-K recommendation, which is widely used in real scenarios. To address both context-aware and implicit feedback scenarios, the ranking-based FM methods have been recently investigated [30, 44, 46]. By far, existing ranking-based FM algorithms mainly rely on negative sampling for efficient model learning. Despite effectiveness, it is reasonable to argue that sampling is not robust as it is highly sensitive to the sampling distribution. Essentially, sampling is biased, making it difficult to converge to the same performance with all training examples, regardless of how many update steps have been taken [6, 45]. Hence, sampling-based FM is still a suboptimal scheme for ranking tasks such as Top-K recommendation.

Despite the overwhelming research on FM, the majority of existing studies are conducted to apply deep neural networks for modelling high-order feature interactions, such as multi-layer perception (MLP) [15, 16], attention mechanisms [43], and Convolutional Neural Network (CNN) [25, 44], etc. However, these studies either focus on rating prediction task only, or typically adopt convenient negative sampling for optimizing the ranking performance. Although these methods have yielded great promise in many prediction tasks, their ranking performances are limited by the inherent weakness of sampling-based learning strategy.

In this paper, we propose to learn FM without sampling for ranking tasks, which is particularly intended for context-aware recommendation. In contrast to sampling, non-sampling strategy computes the gradient over the whole data (including all missing data). As such, it can easily converge to a better optimum in a more stable way [6, 22]. Unfortunately, the difficulty in applying non-sampling strategy lies in the expensive computational cost. Although some studies have been made to explore efficient non-sampling Matrix Factorization (MF) methods [6, 21, 24, 47], it is non-trivial to directly apply MF learning methods for existing FM formulations. Because compared to FM that has a huge amount of cross-feature interactions, MF only considers pure user-item ID interactions. When such a nice structure is broken, these efficient algorithms become invalid and the complexity return to intractable. This creates an impending need of efficient learning methods for non-sampling FM.

In light of the above problems of existing solutions, we design a new framework named Efficient Non-Sampling Factorization Machines (ENSFM). Through novel designs of memorization strategies, we first reformulate FM into a generalized MF framework, and then leverages the bi-linear structure of MF to achieve speedups. The proposed ENSFM framework builds up a clear bridge between the two most popular recommendation methods — MF and FM, with theoretical guarantees, and resolves the challenging efficiency issue caused by non-sampling learning strategy. As a result, ENSFM achieves two remarkable advantages over state-of-the-art context-aware recommendation methods: 1) ***effective*** non-sampling optimization and 2) ***efficient*** model training. To evaluate the recommendation performance and training efficiency of our model, we apply ENSFM on three real-world datasets with extensive experiments. The results indicate that our model significantly outperforms the state-of-the-art context-aware methods (including neural models DeepFM, NFM, and CFM) with a much simpler structure and fewer model parameters. Furthermore, ENSFM shows significant advantages in training efficiency, which makes it more practical in real E-commerce scenarios. The main contributions of this work are as follows:

(1) We highlight the importance of learning FM without sampling for context-aware recommendation, which is more effective and stable as considering all samples' information in each parameter update.
(2) We present a novel embedding-based ENSFM framework to achieve more accurate performance while maintaining low complexity. It not only complements the mainstream sampling-based context-aware models, but also provides an efficient, effective, and theoretical guaranteed solution to improve FM.
(3) Extensive experiments are conducted on three benchmark datasets. The results show that ENSFM consistently and significantly outperforms the state-of-the-art models in terms of both recommendation performance and training efficiency.
(4) This work empirically shows that a proper learning method is even more important than advanced neural networks for Top-K recommendation task.

Table 1: Summary of symbols and notation.

Symbol	Description
$\mathbf{U}$	Set of users
$\mathbf{B}$	Batch of users
$\mathbf{V}$	Set of items
$\mathbf{X}$	Set of features
$\mathbf{Y}$	User-item interactions
$\mathcal{R}$	Set of user-item pairs whose values are non-zero
$\mathbf{x}$	Sparse feature input
$\mathbf{e}_i$	Latent vector of feature i
$\mathbf{h}$	Neuron weights of the prediction layer
$\mathbf{p}_u$	Auxiliary vector of user context u
$\mathbf{q}_v$	Auxiliary vector of item context v
$\mathbf{h}_{aux}$	Auxiliary neuron weights of the prediction layer
w_i	First-order feature interaction weight
w_0	Global bias
c_{uv}	Weight of entry y_{uv}
m	The number of user context
n	The number of item context
d	Latent factor number
Θ	Set of neural parameters

2 PRELIMINARIES

In this section, we first introduce the key notations used in this work, and then provide an introduction to factorization machines and the efficient non-sampling matrix factorization methods.

2.1 Notations

Table 1 depicts the notations and key concepts . Suppose we have users $\mathbf{U}$, items $\mathbf{V}$, and features $\mathbf{X}$ in the dataset, and we use the index u to denote a user context, and v to denote an item context. The user-item data matrix is denoted as $\mathbf{Y} = [y_{uv}] \in \{0, 1\}$, indicating whether u has an interaction with item v. We use $\mathcal{R}$ to denote the set of observed entries in $\mathbf{Y}$, i.e., for which the values are non-zero. $\mathbf{x}$ denotes a real valued feature vector, which utilizes one-hot encoding to depict contextual information. An example is illustrated as follows with five feature fields:

$$\overbrace{\underbrace{[0,1,0,\ldots,0]}_{\text{user ID}}\underbrace{[\ 1,0\ ]}_{\text{gender}}\underbrace{[\ 0,1,\ldots,0\ ]}_{\text{organization}}}^{\text{user context}}\overbrace{\underbrace{[0,0,1,\ldots,0]}_{\text{item ID}}\underbrace{[0,1,0,1,\ldots,0]}_{\text{category}}}^{\text{item context}}$$

We use m and n to denote the number of user context features and item context features, respectively. To support efficient optimization, we specifically build three auxiliary vectors: $\mathbf{p}_u$—auxiliary vector of user u, $\mathbf{q}_v$—auxiliary vector of item v, and $\mathbf{h}_{aux}$—auxiliary prediction vector. More details are introduced in Section 3.

2.2 Factorization Machines

Factorization machines [31] is a generic framework which integrates the advantages of flexible feature engineering and high-accuracy prediction of latent factor models. Given a real valued feature vector $\mathbf{x}$, FM estimates the target by modelling all interactions between each pair of features via factorized interaction parameters:

$$\hat{y}_{FM}(\mathbf{x}) = w_0 + \sum_{i=1}^{m+n} w_i x_i + \sum_{i=1}^{m+n}\sum_{j=i+1}^{m+n} \mathbf{e}_i^T \mathbf{e}_j \cdot x_i x_j \tag{1}$$

where w_0 is the global bias, w_i models the interaction of the i-th feature to the target. The $\mathbf{e}_i^T \mathbf{e}_j$ term denotes the factorized interaction, $\mathbf{e}_i \in \mathbb{R}^d$ denotes the embedding vector for feature i, and d denotes the latent factor number. Note that Eq.(1) can be reformulated as:

$$\hat{y}_{FM}(\mathbf{x}) = w_0 + \sum_{i=1}^{m+n} w_i x_i + \frac{1}{2}\sum_{f=1}^{d}\left(\left(\sum_{i=1}^{m+n} \mathbf{e}_{i,f} x_i\right)^2 - \sum_{i=1}^{m+n} \mathbf{e}_{i,f}^2 x_i^2\right) \tag{2}$$

resulting in linear time complexity $O((m+n)d)$ for each training instance [31].

2.2.1 *Complexity Issue of Non-sampling FM.*

Since sampling has been shown to be non-optimal in many existing studies [6, 45, 47], we propose to learn FM without sampling to achieve optimal ranking performance. For implicit data, the observed interactions are rather limited, and non-observed examples are of a much larger scale. A commonly used non-sampling loss is as follows [22], which associates a confidence to each prediction in the implicit feedback matrix $\mathbf{Y}$:

$$\mathcal{L}(\Theta) = \sum_{u\in\mathbf{U}}\sum_{v\in\mathbf{V}} c_{uv}(y_{uv} - \hat{y}_{uv})^2 \tag{3}$$

where c_{uv} denotes the weight of entry y_{uv}. Note that in implicit feedback learning, missing entries are usually assigned a zero y_{uv} value but non-zero c_{uv} weight.

As can be seen, although FM has a linear time complexity for one training instance, the whole complexity of computing the loss in Eq.(3) is still generally unaffordable — $O((m+n)|\mathbf{U}||\mathbf{V}|d)$, because $|\mathbf{U}||\mathbf{V}|$ can easily reach billion level or even higher in real life.

Recently, many variants of FM have been developed, such as DeepFM [15], NFM [16], AFM [43], xDeepFM [25], and CFM [44], etc. However, these variants mainly focus on utilizing different neural networks to model high-order features interactions. Despite effectiveness on rating prediction, the complex network structures make them even harder to apply non-sampling learning for ranking optimization. For recommendation which is a ranking task, deeper models do not necessarily lead to better results since they are more difficult to optimize and tune [13, 16]. We empirically show the details in Section 4.

2.3 Efficient Non-sampling Matrix Factorization

To address the inefficiency issue of non-sampling matrix factorization, several methods have been proposed [6, 21, 45, 47]. Specifically, Chen et al. [6] derive an efficient loss for generalized MF, and prove:

Theorem 2.1. *For a generalized matrix factorization framework whose prediction function is:*

$$\hat{y}_{uv} = \mathbf{h}^T(\mathbf{p}_u \odot \mathbf{q}_v) \tag{4}$$

where $\mathbf{p}_u \in \mathbb{R}^d$ and $\mathbf{q}_v \in \mathbb{R}^d$ are latent vectors of user u and item v, $\odot$ denotes the element-wise product of vectors, the gradient of loss Eq.(3) is exactly equal to that of:

$$\begin{aligned}\tilde{\mathcal{L}}(\Theta) = &\sum_{u\in\mathbf{U}}\sum_{v\in\mathbf{V}^+}\left((c_v^+ - c_v^-)\hat{y}_{uv}^2 - 2c_v^+\hat{y}_{uv}\right)\\ &+ \sum_{i=1}^{d}\sum_{j=1}^{d}\left((h_i h_j)\left(\sum_{u\in\mathbf{U}} p_{u,i}p_{u,j}\right)\left(\sum_{v\in\mathbf{V}} c_v^- q_{v,i}q_{v,j}\right)\right)\end{aligned} \tag{5}$$

if the instance weight c_{uv} is simplified to c_v.

The complexity of Eq.(5) is $O((|\mathbf{U}|+|\mathbf{V}|)d^2 + |\mathcal{R}|d)$ while that of Eq.(3) is $O(|\mathbf{U}||\mathbf{V}|d)$. Since $|\mathcal{R}| \ll |\mathbf{U}||\mathbf{V}|$ in practice, the complexity of training a MF model without sampling is reduced by several magnitudes. The proof of this theorem can be made by reformulating the expensive loss over all negative instances using a partition and a decouple operation, which largely follows from that in [6, 7] with little variations. To avoid repetition, we do not prove it step by step.

Although efficient MF methods have achieved great success, it is non-trivial to directly apply them for existing FM frameworks since MF only considers pure user-item ID interactions. When such a nice structure is broken, these efficient algorithms becomes invalid and the complexity return to intractable. In this study, we design a novel ENSFM framework to address the above problems based on Theorem 2.1.

3 EFFICIENT NON-SAMPLING FACTORIZATION MACHINES (ENSFM)

This section elaborates our proposed ENSFM method, which unifies the strengths of FM and non-sampling strategy for optimal ranking optimization. We first present a general overview of ENSFM. Then we elaborate how to express a generalized FM as matrix factorization with our key designs of memorization strategies. After that an efficient non-sampling learning algorithm for our FM formulation is presented. Finally, some discussions about the learning procedure, generalization, and complexity of our ENSFM are made.

3.1 Overview

The goal of our ENSFM is to efficiently learn FM models without negative sampling, so as to achieve optimal ranking performance for context-aware recommendation. The prediction function of ENSFM follows the generalized embedding-based FM [16, 43], which is:

$$\hat{y}_{FM}(\mathbf{x}) = w_0 + \sum_{i=1}^{m+n} w_i x_i + \underbrace{\mathbf{h}^T \sum_{i=1}^{m+n}\sum_{j=i+1}^{m+n}(x_i\mathbf{e}_i \odot x_j\mathbf{e}_j)}_{f(\mathbf{x})} \tag{6}$$

where $\odot$ denotes the element-wise product of two vectors and $\mathbf{h}$ denotes the neuron weights of the prediction layer. The framework of our ENSFM is shown in Figure 1. We first make a simple high-level overview of the proposed ENSFM:

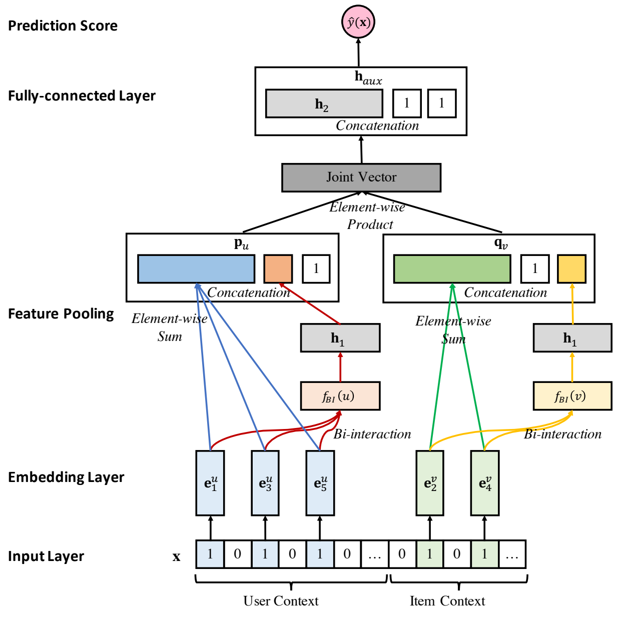

Figure 1: Illustration of our ENSFM framework, showing how to represent FM in a matrix factorization manner (for clarity purpose, the first-order linear regression part is not shown in the figure, which can be trivially incorporated).

(1) The context inputs are converted to dense vector representations through embeddings. Specifically, user context and item context are denoted as $\mathbf{e}^u$ and $\mathbf{e}^v$, respectively. The output $\hat{y}_{FM}(\mathbf{x})$ is a predicted score that indicates user u's preference for item v.

(2) Through novel designs of memorization strategies, we reformulate the FM score of Eq.(6) into a generalized matrix factorization function without any approximation: $\hat{y}_{FM}(\mathbf{x}) = \mathbf{h}_{aux}^T(\mathbf{p}_u \odot \mathbf{q}_v)$ where $\mathbf{p}_u, \mathbf{q}_v$, and $\mathbf{h}_{aux}$ are auxiliary vectors denote user u, item v, and prediction parameter, respectively. It present a new view of FM framework.

(3) We propose an efficient mini-batch non-sampling algorithm to optimize our ENSFM framework, which is more effective and stable due to the consideration of all samples in each parameter update.

3.2 ENSFM Methodology

3.2.1 ***Theoretical Analysis.*** We first present the theoretical guarantee of our proposed ENSFM in this subsection.

THEOREM 3.1. *The prediction function of a generalized factorization machines (Eq.(6)) can be reformulated into a matrix factorization function:*

$$\hat{y}_{FM}(\mathbf{x}) = \mathbf{h}^T(\mathbf{p}_u \odot \mathbf{q}_v) \tag{7}$$

where $\mathbf{p}_u$ only depends on user context u and $\mathbf{q}_v$ only depends on item context v.

Theorem 3.1 connects the relationship between the two most popular recommendation methods — Matrix Factorization (MF) and Factorization Machines (FM). Next, we prove Theorem 3.1 based on a generalized FM function (Eq.(6)) while elaborating our ENSFM framework [2].

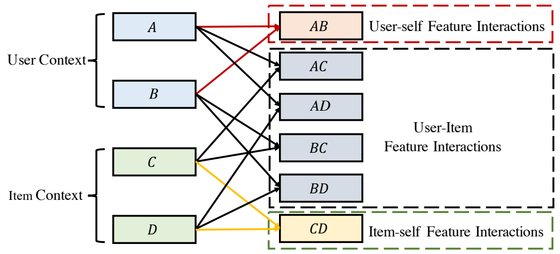
User Context
Item Context
A
B
C
D
AB
AC
AD
BC
BD
CD
User-self Feature Interactions
User-Item Feature Interactions
Item-self Feature Interactions

Figure 2: An example of feature interactions, which can be divided into three groups: user-self, item-self, and user-item. User-self feature interactions are independent of item features, while item-self interactions are also independent of user features.

PROOF. Recall the second-order feature interactions $f(x)$ in Eq.(6), it can be rearranged as follows:

$$f(\mathbf{x}) = \mathbf{h}_1^T \left(\underbrace{\sum_{i=1}^{m}\sum_{j=i+1}^{m}(x_i^u\mathbf{e}_i^u \odot x_j^u\mathbf{e}_j^u)}_{f_{BI}(u)} + \underbrace{\sum_{i=1}^{n}\sum_{j=i+1}^{n}(x_i^v\mathbf{e}_i^v \odot x_j^v\mathbf{e}_j^v)}_{f_{BI}(v)} \right) + \mathbf{h}_2^T \left(\sum_{i=1}^{m} x_i^u\mathbf{e}_i^u \odot \sum_{i=1}^{n} x_i^v\mathbf{e}_i^v \right) \tag{8}$$

where $f_{BI}(u)$ and $f_{BI}(v)$ indicate the second-order interactions among user-self features and item-self features, respectively (see Figure 2). Note that the prediction parameter $\mathbf{h}$ can be extended to $\mathbf{h}_1$ and $\mathbf{h}_2$. This setting allows more flexible modelling of self feature interactions and user-item feature interactions respectively, which also leads to better generalization ability of our framework. Other advanced structures like attention mechanism [4, 43] can also be applied, we leave it as future work as this is not the main concern of this paper.

As shown in Figure 2, user-self feature interactions are independent of item features, and item-self interactions are also independent of user features. Therefore, we could apply memorization strategy to precomputing the two terms. We detail above process by building three auxiliary vectors $\mathbf{p}_u \in \mathbb{R}^{d+2}$, $\mathbf{q}_v \in \mathbb{R}^{d+2}$, and $\mathbf{h}_{aux} \in \mathbb{R}^{d+2}$ to denote user u, item v, and prediction parameter (see Figure 3):

$$\mathbf{p}_u = \begin{bmatrix} \mathbf{p}_{u,d} \\ \mathbf{p}_{u,d+1} \\ \mathbf{p}_{u,d+2} \end{bmatrix}; \mathbf{q}_v = \begin{bmatrix} \mathbf{q}_{v,d} \\ \mathbf{q}_{v,d+1} \\ \mathbf{q}_{v,d+2} \end{bmatrix}; \mathbf{h}_{aux} = \begin{bmatrix} \mathbf{h}_{aux,d} \\ \mathbf{h}_{aux,d+1} \\ \mathbf{h}_{aux,d+2} \end{bmatrix} \tag{9}$$

[2]The proof of vanilla FM can be made similarly.

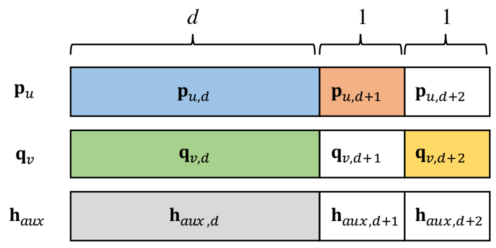

Figure 3: Illustration of the three built auxiliary vectors. $\mathbf{p}_u$, $\mathbf{q}_v$, and $\mathbf{h}_{aux}$ denote user u, item v, and prediction parameter, respectively.

where

$$\mathbf{p}_{u,d} = \sum_{i=1}^{m} x_i^u \mathbf{e}_i^u; \mathbf{p}_{u,d+1} = \mathbf{h}_1^T f_{BI}(u) + w_0 + \sum_{i=1}^{m} w_i^u x_i^u; \mathbf{p}_{u,d+2} = 1 \tag{10}$$

$$\mathbf{q}_{v,d} = \sum_{i=1}^{n} x_i^v \mathbf{e}_i^v; \mathbf{q}_{v,d+1} = 1; \mathbf{q}_{v,d+2} = \mathbf{h}_1^T f_{BI}(v) + \sum_{i=1}^{n} w_i^v x_i^v \tag{11}$$

$$\mathbf{h}_{aux,d} = \mathbf{h}_2; \mathbf{h}_{aux,d+1} = 1; \mathbf{h}_{aux,d+2} = 1 \tag{12}$$

As a result, the prediction function of a generalized FM can be reformulated as a matrix factorization function:

$$\hat{y}_{FM}(\mathbf{x}) = \mathbf{h}_{aux}^T(\mathbf{p}_u \odot \mathbf{q}_v) \tag{13}$$

where $\mathbf{p}_u$ only depends on user context u and $\mathbf{q}_v$ only depends on item context v. The result of Eq. (13) is exactly the same as Eq(6) when setting $\mathbf{h}_1 = \mathbf{h}_2 = \mathbf{h}$. □

3.2.2 ***Efficient Mini-batch Learning Algorithm.*** Here we present our mini-batch ENSFM learning algorithm, which can be derived from the following analysis.

First, $f_{BI}(u)$ and $f_{BI}(v)$ in Eq.(8) can be rewritten to achieve linear time complexity [16, 31]. Take $f_{BI}(u)$ as an example:

$$f_{BI}(u) = \frac{1}{2}\left((\sum_{i=1}^{m} x_i^u \mathbf{e}_i^u)^2 - \sum_{i=1}^{m} (x_i^u \mathbf{e}_i^u)^2 \right) \tag{14}$$

The time complexity is $O(md)$ for $f_{BI}(u)$ and $O(nd)$ for $f_{BI}(v)$.

Second, the proof of Theorem 3.1 shows that $\mathbf{p}_u$ and $\mathbf{q}_v$ in Eq.(13) are independent of each other (i.e., $\mathbf{p}_u$ does not change when u interacts with different items). Therefore, we could achieve a significant speed-up by precomputing the auxiliary vectors to avoid the massive repeated computations.

Finally, after we build the auxiliary vectors, the prediction of our ENSFM is reformulated into a MF function, which satisfies the requirements of THEOREM 2.1. Thus we have the non-sampling loss for a batch of users as follows:

$$\begin{aligned}\tilde{\mathcal{L}}(\Theta) = &\sum_{u \in \mathbf{B}} \sum_{v \in \mathbf{V}^+} \left((c_v^+ - c_v^-)\hat{y}(\mathbf{x})^2 - 2c_v^+ \hat{y}(\mathbf{x}) \right) \\ &+ \sum_{i=1}^{d} \sum_{j=1}^{d} \left((h_{aux,i} h_{aux,j}) \left(\sum_{u \in \mathbf{B}} p_{u,i} p_{u,j} \right) \left(\sum_{v \in \mathbf{V}} c_v^- q_{v,i} q_{v,j} \right) \right)\end{aligned} \tag{15}$$

Algorithm 1 ENSFM Learning algorithm

```
Require: Training data {Y, U, V, X}; weights of entries c; learning
    rate η; embedding size d
Ensure: Neural parameters Θ
 1: Randomly initialize neural parameters Θ
 2: while Stopping criteria is not met do
 3:     while An epoch is not end do
 4:         Randomly draw a training batch {Y_B, B, V, X}
 5:         Build auxiliary vectors P_B for users (Eq.(9,10))
 6:         Build auxiliary vectors Q for items (Eq.(9,11))
 7:         Build auxiliary vector h (Eq.(9,12))
 8:         Compute the loss L̃(Θ) (Eq.(15))
 9:         Update model parameters
10:     end while
11: end while
12: return Θ
```

where $\mathbf{B}$ denotes a batch of users, $\mathbf{V}$ denotes all the items in the data set, and c_{uv} is simplified to c_v that denotes the weight of entry R_{uv}. Algorithm 1 summarizes the accelerated algorithm for our ENSFM.

3.3 Discussion

3.3.1 ***Computational Complexity.*** The computational complexity of our ENSFM can be divided into two parts. The first part is to build auxiliary vectors and the second part is cost for efficient non-sampling learning. For a training batch, building auxiliary vectors $\mathbf{P_B}$ takes $O(m|\mathbf{B}|d)$, $\mathbf{Q}$ takes $O(n|\mathbf{V}|d)$. Note that $\mathbf{P_B}$ and $\mathbf{Q}$ can be updated in synchronization with the changes in $\tilde{\mathcal{L}}(\Theta)$. Therefore, updating a training batch takes $O((m|\mathbf{B}| + n|\mathbf{V}|)d^2 + |\mathcal{R}_\mathbf{B}|d)$. The total cost of Algorithm 1 for one epoch over all variables is $O((m|\mathbf{U}| + \frac{n|\mathbf{U}||\mathbf{V}|}{|\mathbf{B}|})d^2 + |\mathcal{R}|d)$, where $\mathcal{R}$ denotes the set of positive user-item interactions. For the original regression loss (Eq.(3)), one epoch takes $O((m+n)|\mathbf{U}||\mathbf{V}|d)$. Since $|\mathcal{R}| \ll |\mathbf{U}||\mathbf{V}|$ and $d \ll |\mathbf{B}|$ in practice, the computational complexity of training FM without sampling is reduced by several magnitudes. This makes it possible to apply non-sampling optimization strategy for FM models. Moreover, the optimization results are exactly the same with the original non-sampling regression loss since no approximation is introduced during the learning algorithm.

3.3.2 ***Relation to Existing Methods.*** Our ENSFM generalizes several existing context-aware solutions [9, 16, 31]. Specifically, by fixing $\mathbf{h}_1$ and $\mathbf{h}_2$ to a constant vector of (1, ..., 1), we can exactly recover the vanilla FM [31]; By fixing $\mathbf{h}_1 = \mathbf{h}_2$, we recover NFM without hidden layers [16]; By fixing $\mathbf{h}_1$ to (0, ..., 0) and $\mathbf{h}_2$ to (1, ..., 1), we can recover the SVDFeature framework [9]. In addition to the above differences, the key ingredient of our ENSFM is that we propose an efficient non-sampling algorithm for model learning, which complements the mainstream sampling-based context-aware models and provides a new approach to improve FM.

Note that some previous studies have also discussed the relationship between Matrix Factorization and Factorization Machines [32, 33]. Specifically, Rendel et al. showed that FM recovers MF when the context input only contains ID information [32] and rewrote FM into a MF model [33]: $\hat{y}_{FM}(\mathbf{x}) = \mathbf{v}_c^T \mathbf{v}_i$, however, the built auxiliary vector $\mathbf{v}_i$ is not user-independent and changes when interacting with different users. This makes it impossible to safely separate user context and item context for efficient non-sampling learning. Our ENSFM reformulates the prediction of FM into a MF function, where the two multiplied vectors only depend on user context and item context, respectively (THEOREM 3.1).

The proposed efficient learning algorithm of our ENSFM is based on THEOREM 2.1 [6], which is not applicable for models with non-linear prediction layers. Thus our current ENSFM framework has a linear prediction layer on the top. We leave the extensions as future work. Nevertheless, it is worth mention that compared to the state-of-the-art deep learning methods — the 1-layer NFM [16], 3-layer DeepFM [15], and CNN based CFM [44], our ENSFM achieves significant improvements on context-aware Top-K recommendation task, while maintaining a much simpler structure, fewer model parameters, and much fast training process. We show the details in Section 4.

3.4 Training

Modern computing units such as GPU usually provide speedups for matrix-wise float operations. Thus our mini-batch based optimization methods can be naturally implemented in modern machine learning tools like Tensorflow and PyTorch. The model parameters can be calculated with standard back-propagation. To optimize the objective function, we adopt mini-batch Adagrad [14] as the optimizer. Its main advantage is that the learning rate can be self-adapted during the training phase, which eases the pain of choosing a proper learning rate.

Dropout is an effective solution to prevent deep neural networks from overfitting [39], which randomly drops part of neurons during training. In this work, we employ dropout to improve our model's generalization ability. Specifically, we randomly drop ρ percent of latent factors after obtaining $f_{BI}(u)$, $f_{BI}(v)$, and the element-wise production in Eq.(13) to improve the model's generalization ability, where ρ is termed as the dropout ratio.

4 EXPERIMENTS

In this section, we perform experiments to verify the correctness, efficiency, and effectiveness of our proposed Efficient Non-Sampling Factorization Machines. We aim to answer the following research questions:

RQ1 Does our proposed ENSFM outperform state-of-the-art methods for context-aware Top-K recommendation task?

RQ2 How is the training efficiency of ENSFM compared with state-of-the-art FM methods?

RQ3 How do the key hyper-parameter settings impose influence on the performance of ENSFM?

4.1 Experimental Setup

4.1.1 ***Data Description.*** To evaluate the performance of the proposed ENSFM, we conduct extensive experiments on three real-world implicit feedback datasets: ***Frappe***[3], ***Last.fm***[4], and ***Movielens***[5]. We briefly introduce the three datasets:

[3]http://baltrunas.info/research-menu/frappe
[4]https://grouplens.org/datasets/hetrec-2011/
[5]https://grouplens.org/datasets/movielens/1m/

Table 2: Statistical details of the datasets.

Dataset	#User	#Item	#Feature	#Instance	#Field
Frappe	957	4,082	5,382	96,203	10
Last.fm	1,000	20,301	37,358	214,574	4
Movielens	6,040	3,706	10,021	1,000,209	6

- ***Frappe***: Frappe is a context-aware app discovery tool. This dataset is conducted by [1]. Frappe contains 96,203 app usage logs of different user contexts. Each log contains 10 contextual feature fields including user ID, item ID, daytime and some other information.
- ***Last.fm***: The Last.fm dataset is for music recommendation. In our experiments, we use the latest one day listen history of 1,000 users. The user context is described by user ID and the last music ID that the user has listened within 90 minutes. The item context includes music ID and artist ID.
- ***Movielens***: MovieLens is a dataset of movie rating which has been leveraged extensively to investigate the performance of recommendation algorithms. In our experiments, we choose the version including one million ratings and binarize it into implicit feedback. The user context is described by user ID, gender, age, and occupation. The item context is composed of movie ID and movie genres

Note that for Frappe and Last.fm, we use exactly the same splits as in [44][6]. The statistical details of these datasets are summarized in Table 2.

4.1.2 ***Baselines.*** We compare the performance of ENSFM with the following baselines:

- **PopRank**: This method returns Top-K most popular items. It acts as a basic benchmark.
- **FM** [31]: The original Factorization Machine. It has shown strong performance for context-aware prediction.
- **NFM** [16]: Neural factorization machine is one of the state-of-the-art deep learning method which uses MLP to learn nonlinear and high-order interaction signals.
- **DeepFM** [15]: This method ensembles the original FM and a MLP to generate recommendation.
- **ONCF** [17]: This method is a newly proposed algorithm which improves MF with outer product for item recommendation.
- **CFM** [44]: Convolutional Factorization Machine models higher-order interactions through outer product and CNN, which is the state-of-the-art neural extension of factorization machines.
- **ENMF** [6, 7]: Efficient Neural Matrix Factorization is a newly proposed non-sampling neural recommendation method. It is a state-of-the-art method for Top-K recommendation which only based on the historical feedback information.

Note that the official implementations of FM, NFM, and DeepFM are specifically optimized for rating prediction. Following the settings of previous work [10, 44, 46], these methods are optimized with negative sampling and Bayesian Personalized Ranking [34] objective function to fit the ranking task. As we have discussed, non-sampling optimization is generally infeasible for existing FM models (especially neural methods) since they can not finish the

[6]https://github.com/chenboability/CFM

Table 3: Performance of different models on three datasets. ** denotes the statistical significance for $p < 0.01$, compared to the best baseline. "RI" indicate the average relative improvements of our ENSFM over the corresponding baseline.

Frappe	**HR@5**	**HR@10**	**HR@20**	**NDCG@5**	**NDCG@10**	**NDCG@20**	**RI**
PopRank	0.2539	0.3493	0.4136	0.1595	0.1898	0.2060	+143.3%
FM (*Rendle et al., 2010*)	0.4204	0.5486	0.6590	0.3054	0.3469	0.3750	+39.86%
DeepFM (*Guo et al., 2017*)	0.4632	0.6035	0.7322	0.3308	0.3765	0.4092	+27.77%
NFM (*He et al., 2017*)	0.4798	0.6197	0.7382	0.3469	0.3924	0.4225	+23.64%
ONCF (*He et al., 2018*)	0.5359	0.6531	0.7691	0.3940	0.4320	0.4614	+13.24%
CFM (*Xin et al., 2019*)	0.5462	0.6720	0.7774	0.4153	0.4560	0.4859	+9.15%
ENMF (*Chen et al., 2019*)	0.5682	0.6833	0.7749	0.4314	0.4642	0.4914	+6.95%
ENSFM	**0.6094****	**0.7118****	**0.7889****	**0.4771****	**0.5105****	**0.5301****	–

Last.fm	**HR@5**	**HR@10**	**HR@20**	**NDCG@5**	**NDCG@10**	**NDCG@20**	**RI**
PopRank	0.0013	0.0023	0.0032	0.0007	0.0011	0.0013	+26566%
FM (*Rendle et al., 2010*)	0.1658	0.2382	0.3537	0.1142	0.1374	0.1665	+108.4%
DeepFM (*Guo et al., 2017*)	0.1773	0.2612	0.3799	0.1204	0.1473	0.1772	+94.59%
NFM (*He et al., 2017*)	0.1827	0.2678	0.3783	0.1235	0.1488	0.1765	+91.76%
ONCF (*He et al., 2018*)	0.2183	0.3208	0.4611	0.1493	0.1823	0.2176	+58.11%
CFM (*Xin et al., 2019*)	0.2375	0.3538	0.4841	0.1573	0.1948	0.2277	+48.05%
ENMF (*Chen et al., 2019*)	0.3188	0.4254	0.5279	0.2256	0.2531	0.2894	+15.94%
ENSFM	**0.3683****	**0.4729****	**0.5793****	**0.2744****	**0.3082****	**0.3352****	–

Movielens	**HR@5**	**HR@10**	**HR@20**	**NDCG@5**	**NDCG@10**	**NDCG@20**	**RI**
PopRank	0.0084	0.0308	0.0763	0.0041	0.0111	0.0227	+388.9%
FM (*Rendle et al., 2010*)	0.0377	0.0687	0.1164	0.0234	0.0334	0.0453	+52.32%
DeepFM (*Guo et al., 2017*)	0.0413	0.0754	0.1351	0.0247	0.0365	0.0503	+38.43%
NFM (*He et al., 2017*)	0.0421	0.0775	0.1334	0.0268	0.0381	0.0521	+33.91%
ONCF (*He et al., 2018*)	0.0491	0.0801	0.1368	0.0301	0.0402	0.0543	+24.70%
CFM (*Xin et al., 2019*)	0.0514	0.0812	0.1398	0.0318	0.0419	0.0567	+20.22%
ENMF (*Chen et al., 2019*)	0.0534	0.0867	0.1523	0.0332	0.0448	0.0606	+13.10%
ENSFM	**0.0601****	**0.1024****	**0.1690****	**0.0373****	**0.0508****	**0.0674****	–

For Frappe and Last.fm datasets, the results of FM, DeepFM, NFM, ONCF, and CFM are the same as those reported in [44] since we share exactly the same data splits and experimental settings.

training process in acceptable time and computing resources, which is the main concern of this work.

All models except PopRank are implemented with TensorFlow[7], a well-known open-source software library for deep learning. For FM, NFM, ONCF and CFM, we use the implementations released by the authors of [44][6]. For DeepFM, we use the implementation released by the authors of [15][8]. For ENMF, we use the implementation released by the authors of [6][9].

4.1.3 *Evaluation Metrics.*

The leave-one-out evaluation protocol [10, 19, 20, 44] is employed here to study the performance of item recommendation. Specifically, for Last.fm and MovieLens, the latest transaction of each user is held out for testing and the remaining data is treated as the training set. For the Frappe dataset, as there is no timestamp information, we randomly select one instance for each specific user context as the test example. We evaluate the ranking list using Hit Ratio (**HR**) and Normalized Discounted Cumulative Gain (**NDCG**). HR is a recall-based metric, measuring whether the testing item is in the Top-K list, while NDCG is position-sensitive, which assigns higher scores to hits at higher positions. The two metrics have been widely used in previous recommendation studies [10, 19–21, 44]. For both metrics, larger values indicate better performance.

[7]https://www.tensorflow.org/
[8]https://github.com/ChenglongChen/tensorflow-DeepFM
[9]https://github.com/chenchongthu/ENMF

4.1.4 *Parameter Settings.*

The parameters for all baseline methods are initialized as in the corresponding papers, and are then carefully tuned to achieve optimal performances. The learning rate for all models are tuned amongst [0.005, 0.01, 0.02, 0.05]. To prevent overfitting, we tune the dropout ratio in [0.1, 0.3, 0.5, 0.7, 0.9, 1]. The batch size is tested in [128, 256, 512], the dimension of the latent factor number d is tested in [8, 16, 32, 64]. Note that we uniformly set the weight of missing data as c_0, as the effectiveness of popularity-biased weighting strategy is beyond the scope of this paper. c_0 is tuned amongst [0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1]. After the tuning process, the batch size is set to 512, the size of the latent factor dimension d is set to 64, and the learning rate is set to 0.05. The output channels of CNN-based models (i.e., ONCF and CFM) are set as 32 according to their original paper [17, 44]. Regarding NFM, the number of MLP layers is set as 1 with 64 neurons, which is the recommended setting of their original paper [16]. For the deep component of DeepFM, we set the MLP according to their

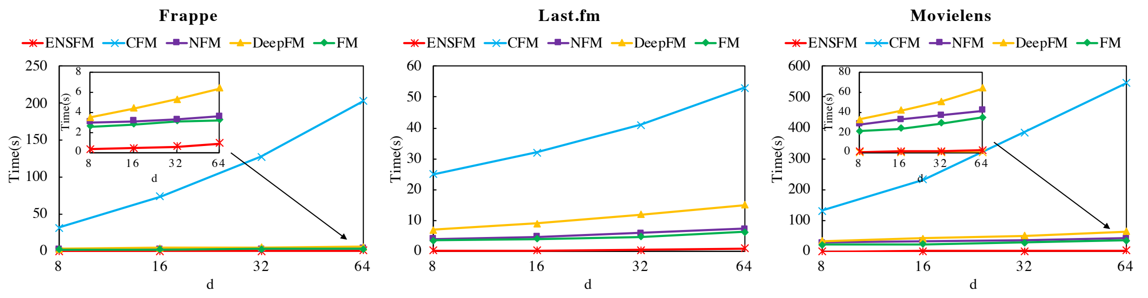

Figure 4: Comparison on the per iteration training time of FM, DeepFM, NFM, CFM, and ENSFM with different embedding size d.

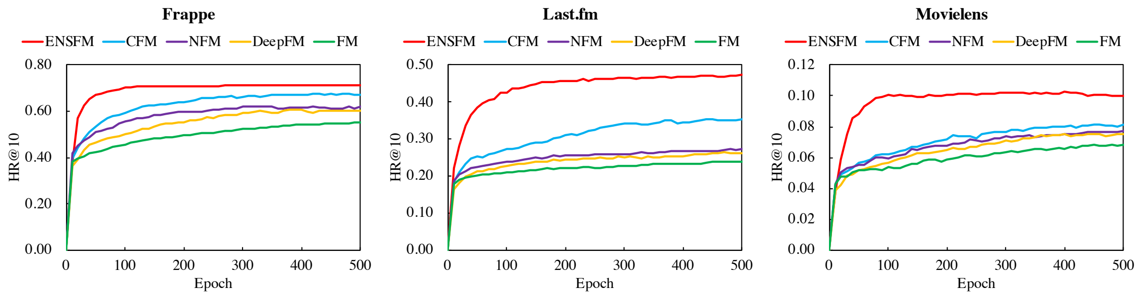

Figure 5: Performance curves of FM, DeepFM, NFM, CFM, and ENSFM on the three datasets.

original paper [15], which has 3 layers and 200 neurons in each layer. The dropout ratio ρ is set to 0.9. c_0 is set 0.05 for Frappe, 0.005 for Last.fm, and 0.5 for Movielens, respectively.

4.2 Performance Comparison (RQ1)

The results of the comparison of different methods on three datasets are shown in Table 3. To evaluate on different recommendation lengths, we set the length K = 5, 10, and 20 in our experiments. From the results, the following observations can be made:

First and foremost, our proposed ENSFM achieves the best performance on the three datasets, significantly outperforming all the state-of-the-art baseline methods. Specifically, compared to CFM — a recently proposed and very expressive deep learning-based FM model, our ENSFM exhibits average improvements of 9.15%, 48.05%, and 20.22% on the three datasets. This is very remarkable, since ENSFM is a shallow FM framework that has much fewer parameters. The substantial improvement could be attributed to the proposed non-sampling learning algorithm. The parameters in ENSFM is optimized on the whole data, while sample-based methods (FM, DeepFM, NFM, ONCF, CFM) only use a fraction of sampled data and may ignore important negative examples. The results also imply the potential of improving conventional shallow methods with a better learning algorithm. The performance of ENSFM is significantly better than ENMF, which indicates that context information is helpful in recommendation [2, 33, 44].

Second, we observe that methods using non-sampling learning strategy generally perform better than sampling-based methods. For example, in Table 3, ENMF (which utilizing no context information) and our ENSFM both perform better than the state-of-the-art methods: NFM, DeepFM, ONCF, and CFM. This is consistent with previous work [6, 45, 47], which indicates that sampling is a biased learning strategy for optimizing ranking tasks.

Lastly, although deep learning-based FM methods do achieve better performance than vanilla FM when adopting the same sampling-based learning strategy, the improvements are relatively small compared with our non-sampling ENSFM. It reveals that on ranking tasks, deeper models do not necessarily lead to optimal results. A better learning strategy is even more important than advanced neural network structures. The large performance gap between baselines and our ENSFM reflects the value of learning FM without sampling for ranking tasks.

4.3 Efficiency Analyses (RQ2)

Many deep learning studies only focused on obtaining better results but ignored the computational efficiency of reaching the reported accuracy [37]. However, expensive training cost can limit the applicability of a model to real-world large-scale systems. In this section, we conduct experiments to explore the training efficiencies of our ENSFM and four state-of-the-art FM methods: FM, DeepFM, NFM, and CFM. All experiments in this section are run on the same machine (Intel Xeon 8-Core CPU of 2.4 GHz and single NVIDIA GeForce GTX TITAN X GPU) for fair comparison on the efficiency.

We first investigate the training time of FM, DeepFM, NFM, CFM and our ENSFM with different embedding size d. The results are shown in Figure 4. From the figure, we can see that the training time cost of ENSFM is much less than other FM methods with different size of d. As d increases, the costs of baseline methods increase significantly, while our ENSFM still maintains a very fast training process (e.g., 2 seconds per iteration on Movielens with a large d of 64). We then conduct comparing among the overall training time of the above methods. The embedding size is set

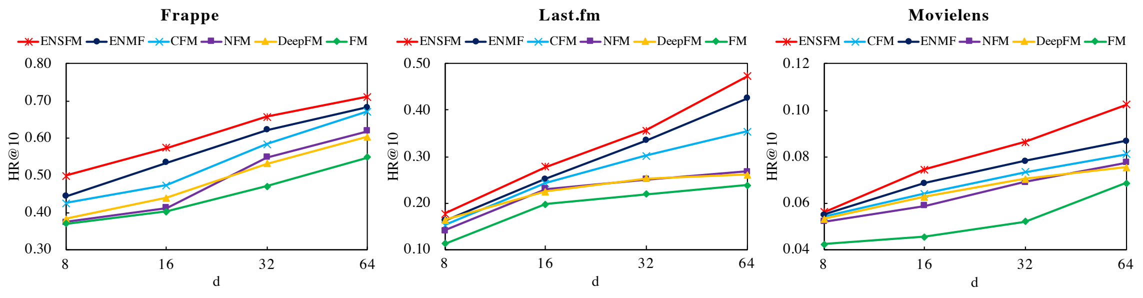

Figure 6: Performance of FM, DeepFM, NFM, CFM, ENMF, and ENSFM w.r.t the embedding size on the three datasets.

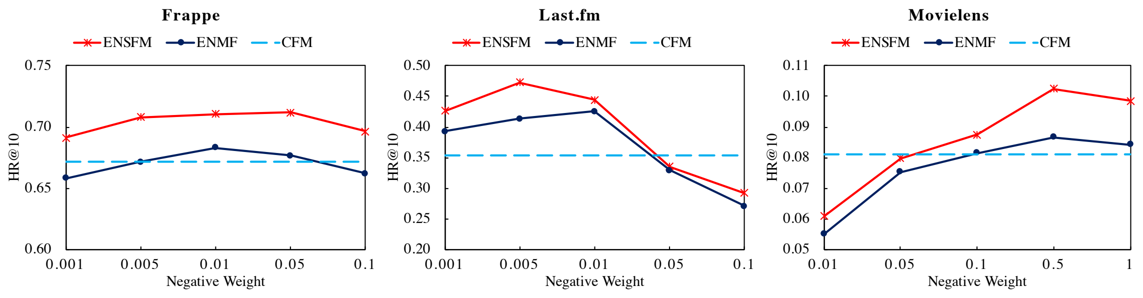

Figure 7: Performance of ENSFM w.r.t the negative weight on the three datasets.

Table 4: Comparisons of runtime (second/minute/hour/day [s/m/h/d]). "S", "I", and "T" represents the training time for a single iteration, the number of iterations to converge, and the total training time, respectively.

Model	*Frappe*			*Last.fm*			*Movielens*		
	S	I	T	S	I	T	S	I	T
FM	3.2s	500	27m	6.2s	500	52m	35s	500	5h
NFM	3.6s	500	30m	7.3s	500	61m	42s	500	6h
DeepFM	6.4s	500	54m	15s	500	324m	64s	500	9h
CFM	203s	500	28h	54s	500	125m	9m	500	3d
ENSFM	**0.9s**	200	**3m**	**1.1s**	500	**10m**	**2s**	200	**7m**

to 64 for all the methods and the results are shown in Table 4. We can obviously observe that the overall training time of our ENSFM is **several magnitudes faster** than the baseline models. In particular, for the largest dataset Movielens, our ENSFM only needs 7 minutes to achieve the optimal performance, while the state-of-the-art models NFM, DeepFM, and CFM take about 6 hours, 9 hours, and 3 days, respectively. This acceleration is over 50 times than NFM and 600 times than CFM, which is highly valuable in practice and is difficult to achieve with simple engineering efforts. For other datasets, the results of ENSFM are also remarkable. In real E-commerce scenarios, the cost of training time is also an important factor to be considered [6]. Our ENSFM shows significant advantages in training efficiency, which makes it more practical in real life.

We also investigate the training process of the baselines and our ENSFM. Figure 5 demonstrates the state of each method at embedding size 64 on three datasets. Due to the space limitation, we only show the results on HR@10 metric. For other metrics, the observations are similar. From the figure, we can see that ENSFM converges much faster than other FM methods and consistently achieves the best performance. The reason is that our ENSFM is optimized with a newly derived non-sampling algorithm, while other FM methods are based on negative sampling, which generally requires more iterations and can be sub-optimal.

4.4 Hyper-parameter Analyses (RQ3)

In this section, we conduct experiments to investigate the impact of different values of the embedding size d and different negative weight c_0 on our ENSFM method. It is worth mention that our ENSFM can be tuned very easily in practice due to: 1) The overall training process of ENSFM is very fast; 2) Unlike most existing deep learning FM methods, our ENSFM does not require pre-training from FM; 3) Generally only one hyper-parameter — negative weight c_0 needs to be tuned for different datasets.

4.4.1 ***Impact of Embedding Size.*** Figure 6 shows the performance of HR@10 with respect to the embedding size d. For other metrics, the observations are similar. As can be seen from this figure, our ENSFM outperforms all the other models with different values of d. Notably, ENSFM with d of 32 even performs better than the state-of-the-art context-aware method CFM with a larger d of 64. This further verifies the positive effect of non-sampling learning in our ENSFM method. Moreover, as the latent dimension size increases, the performance of all models increase. This indicates that a larger dimension could capture more hidden factors of users and items, which is beneficial to Top-K recommendation due to the increased modeling capability. This observation is similar to previous work [5, 18, 20]. However, for most methods, a larger dimension also requires more time for training. Thus it is crucial to increase a model's efficiency by learning with efficient optimization methods.

4.4.2 ***Impact of Negative Weight.*** We demonstrate the results of our ENSFM with different negative weights in Figure 7. Note that in our experiments we uniformly set the weight of missing data as c_0 and leave the item-dependent weighting strategy [21, 26] to a future work. For different datasets, c_0 is tuned amongst [0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1]. From the figure, we can make the following observations: 1) For Frappe, Last.fm, and Movielens, the peak performance is achieved when c_0 is around 0.05, 0.005, and 0.5, respectively. When c_0 becomes smaller or too larger, the performance of ENSFM degrades. This highlights the necessity of accounting for the missing data when modeling implicit feedback for item recommendation. 2) Considering the performance on each dataset, we find that the optimal weight of missing instance depends on the density of the dataset. The Movielens dataset is more dense compared to Frappe and Last.fm. As shown in previous work [21, 26], popular items are more likely to be known by users, thus it is reasonable to assign a larger weight to a missing popular item as it is more probable to be a truly negative instance. 3) Our ENSFM is very robust to the value of c_0. Generally, ENSFM outperforms the best context-aware baseline CFM with a wide range of c_0 on the three datasets (e.g., c_0 between 0.1 to 1 on Movielens).

5 RELATED WORK

5.1 Context-aware Recommendation

Context-aware recommendation aims to leverage rich context information such as user demographics, item attributes, and time/location of the current transaction to improve the performance of recommender systems [35, 46]. Existing solutions mainly account for the interactions among features, which can be categorized into two types [41, 44]: 1) manually construction of features and 2) automatically learning feature interactions. The first type manually constructs combinatorial features to explicitly encode feature interactions. Then the cross features are fed into predictive models such as logistic regression [11] and deep neural networks [42]. Apparently, this type requires heavy engineering efforts and domain knowledge, making the solutions less adaptable to other domains. The second type automatically learn feature interactions in a unified model. A typical paradigm is to integrate context features into factorization models. For example, FM [31] models the interaction of two features as the inner product of their embeddings, and Deep Crossing [38] concatenates feature embeddings and feeds them into a multi-layer perception (MLP) to learn high-order interactions.

Due to the popularity of FM, many efforts have been made to enhance its framework. The majority of existing studies are conducted to apply deep neural networks for modelling high-order feature interactions, such as MLP (DeepFM [15], NFM [16]), attention mechanisms (AFM [43]), and CNN (xDeepFM [25], CFM [44]), etc. However, note that previous FM studies generally focus on rating prediction task [3, 16, 27, 31, 35, 43], while the goal of recommendation is preferred as ranking task rather than rating prediction [12]. Although some methods by combining negative sampling and FM models provide promising solutions [30, 44, 46], their ranking performance can still be limited by the inherent weakness of sampling-based learning strategy.

5.2 Model Learning for Top-K Recommendation

Learning sparse features from implicit data is a fundamental ingredient for Top-K recommendation. Generally, there are two strategies: 1) negative sampling strategy [5, 20, 34] and 2) non-sampling (whole-data based) strategy [6, 21, 22]. The first strategy samples negative instances from missing entries, while the second one sees all the missing data as negative. In previous work, negative sampling is widely adopted for efficient training. However, some studies have shown that sampling can limit the performance of recommendation as it is not robust and highly sensitive to the sampling distributions [6, 21, 45, 47]. In contrast, non-sampling strategy leverages the whole data with a potentially better coverage, but inefficiency can be an issue. To retain the fidelity of FM for context-aware Top-K recommendation, we persist in non-sampling learning and develop an efficient FM framework to address the inefficiency issue.

Recently, some efforts have been devoted to resolving the inefficiency issue of non-sampling learning. For example, Pilaszy et al. [29] described an approximate solution of element-wise Alternating Least Squares (ALS). He et al. [21] proposed an efficient ALS with non-uniform missing data. The authors of [45, 47] studied fast Batch Gradient Descent (BGD) methods. Chen et al. [6, 8] derived a flexible non-sampling loss for neural recommendation models. However, existing efficient studies mainly focus on MF methods that only consider pure user-item ID interactions. To the best of our knowledge, this is the first work that provides an efficient mini-batch non-sampling algorithm to learn FM for ranking tasks.

6 CONCLUSION AND FUTURE WORK

In this work, we propose a novel Efficient Non-Sampling Factorization Machines (ENSFM) framework for context-aware Top-K recommendation. ENSFM not only seamlessly connects the relationship between factorization machines and matrix factorization, but also resolves the challenging efficiency issue of non-sampling learning. As a result, ENSFM achieves two remarkable advantages: 1) ***effective*** non-sampling optimization and 2) ***efficient*** model training. Extensive experiments have been made on three real-word datasets. The proposed ENSFM consistently and significantly outperforms the state-of-the-art methods including deep learning models NFM, DeepFM, and CFM, in terms of both recommendation performance and training efficiency.

Recently, there is a surge of interest in applying novel neural networks for recommendation tasks. However, more complex models do not always lead to better results since they are more difficult to optimize and tune [6]. Our work empirically shows that a proper learning method is even more important than advanced neural network structures. As such, we expect future research should pay more attention to design models with better learning algorithms for specific tasks, rather than only relying on complex models and expensive computational power for minor improvements. This work complements the mainstream sampling-based context-aware models and provides a new approach to improve FM methods. The proposed ENSFM framework is not limited to the recommendation task presented in this paper, it has the potential to benefit many other tasks where only positive data is observed. In the future, we will explore our ENSFM method on other related tasks like network embedding [28] and multi-label classification [40]. Also, since our mini-batch learning algorithm can be naturally implemented in modern machine learning tools like Tensorflow and PyTorch, we are interested in extending ENSFM by incorporating more advanced deep learning techniques.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their valuable comments and suggestions. This work is supported by Natural Science Foundation of China (Grant No. 61672311, 61532011) and the National Key Research and Development Program of China (2018YFC0831900).

REFERENCES

[1] Linas Baltrunas, Karen Church, Alexandros Karatzoglou, and Nuria Oliver. 2015. Frappe: Understanding the usage and perception of mobile app recommendations in-the-wild. *arXiv preprint arXiv:1505.03014* (2015).

[2] Immanuel Bayer, Xiangnan He, Bhargav Kanagal, and Steffen Rendle. 2017. A generic coordinate descent framework for learning from implicit feedback. In *Proceedings WWW*. 1341–1350.

[3] Mathieu Blondel, Akinori Fujino, Naonori Ueda, and Masakazu Ishihata. 2016. Higher-order factorization machines. In *Proceedings of NIPS*. 3351–3359.

[4] Chong Chen, Min Zhang, Yiqun Liu, and Shaoping Ma. 2018. Neural Attentional Rating Regression with Review-level Explanations. In *Proceedings of WWW*. 1583–1592.

[5] Chong Chen, Min Zhang, Yiqun Liu, and Shaoping Ma. 2019. Social Attentional Memory Network: Modeling Aspect-and Friend-level Differences in Recommendation. In *Proceedings of WSDM*.

[6] Chong Chen, Min Zhang, Chenyang Wang, Weizhi Ma, Minming Li, Yiqun Liu, and Shaoping Ma. 2019. An Efficient Adaptive Transfer Neural Network for Social-aware Recommendation. In *Proceedings of SIGIR*. ACM, 225–234.

[7] Chong Chen, Min Zhang, Yongfeng Zhang, Yiqun Liu, and Shaoping Ma. 2020. Efficient Neural Matrix Factorization without Sampling for Recommendation. *ACM Trans. Inf. Syst.* 38, 2, Article Article 14 (Jan. 2020), 28 pages. https://doi.org/10.1145/3373807

[8] Chong Chen, Min Zhang, Yongfeng Zhang, Weizhi Ma, Yiqun Liu, and Shaoping Ma. 2020. Efficient Heterogeneous Collaborative Filtering without Negative Sampling for Recommendation. In *Proceedings of AAAI*.

[9] Tianqi Chen, Weinan Zhang, Qiuxia Lu, Kailong Chen, Zhao Zheng, and Yong Yu. 2012. SVDFeature: a toolkit for feature-based collaborative filtering. *Journal of Machine Learning Research* 13, Dec (2012), 3619–3622.

[10] Yifan Chen, Pengjie Ren, Yang Wang, and Maarten de Rijke. 2019. Bayesian Personalized Feature Interaction Selection for Factorization Machines. (2019).

[11] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*. ACM, 7–10.

[12] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. 2010. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of RecSys*. ACM, 39–46.

[13] Maurizio Ferrari Dacrema, Paolo Cremonesi, and Dietmar Jannach. 2019. Are We Really Making Much Progress? A Worrying Analysis of Recent Neural Recommendation Approaches. *arXiv preprint arXiv:1907.06902* (2019).

[14] John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* 12, Jul (2011), 2121–2159.

[15] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. Deepfm: a factorization-machine based neural network for ctr prediction. *arXiv preprint arXiv:1703.04247* (2017).

[16] Xiangnan He and Tat-Seng Chua. 2017. Neural factorization machines for sparse predictive analytics. In *Proceedings of SIGIR*. 355–364.

[17] Xiangnan He, Xiaoyu Du, Xiang Wang, Feng Tian, Jinhui Tang, and Tatseng Chua. 2018. Outer Product-based Neural Collaborative Filtering. *Proceedings of IJCAI* (2018), 2227–2233.

[18] Xiangnan He, Zhankui He, Xiaoyu Du, and Tatseng Chua. 2018. Adversarial Personalized Ranking for Recommendation. *Proceedings of SIGIR* (2018), 355–364.

[19] Xiangnan He, Zhankui He, Jingkuan Song, Zhenguang Liu, Yugang Jiang, and Tatseng Chua. 2018. NAIS: Neural Attentive Item Similarity Model for Recommendation. *IEEE Transactions on Knowledge and Data Engineering* (2018).

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

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

[22] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *Proceedings of ICDM*. Ieee, 263–272.

[23] Yuchin Juan, Yong Zhuang, Wei-Sheng Chin, and Chih-Jen Lin. 2016. Field-aware factorization machines for CTR prediction. In *Proceedings of Recsys*. ACM, 43–50.

[24] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 8 (2009), 30–37.

[25] Jianxun Lian, Xiaohuan Zhou, Fuzheng Zhang, Zhongxia Chen, Xing Xie, and Guangzhong Sun. 2018. xdeepfm: Combining explicit and implicit feature interactions for recommender systems. In *Proceedings of SIGKDD*. ACM, 1754–1763.

[26] Dawen Liang, Laurent Charlin, James McInerney, and David M Blei. 2016. Modeling user exposure in recommendation. In *Proceedings of WWW*. 951–961.

[27] Xiao Lin, Wenpeng Zhang, Min Zhang, Wenwu Zhu, Jian Pei, Peilin Zhao, and Junzhou Huang. 2018. Online compact convexified factorization machine. In *Proceedings of WWW*. 1633–1642.

[28] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS*. 3111–3119.

[29] István Pilászy, Dávid Zibriczky, and Domonkos Tikk. 2010. Fast als-based matrix factorization for explicit and implicit feedback datasets. In *Proceedings of Recsys*. ACM, 71–78.

[30] Runwei Qiang, Feng Liang, and Jianwu Yang. 2013. Exploiting ranking factorization machines for microblog retrieval. In *Proceedings of CIKM*. ACM, 1783–1788.

[31] Steffen Rendle. 2010. Factorization machines. In *Proceedings of ICDM*. IEEE, 995–1000.

[32] Steffen Rendle. 2012. Factorization machines with libfm. *ACM Transactions on Intelligent Systems and Technology (TIST)* 3, 3 (2012), 57.

[33] Steffen Rendle and Christoph Freudenthaler. 2014. Improving pairwise learning for item recommendation from implicit feedback. In *Proceedings of WSDM*. ACM, 273–282.

[34] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of UAI*. 452–461.

[35] Steffen Rendle, Zeno Gantner, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2011. Fast context-aware recommendations with factorization machines. In *Proceedings of SIGIR*. ACM, 635–644.

[36] Francesco Ricci, Lior Rokach, and Bracha Shapira. 2011. Introduction to recommender systems handbook. In *Recommender systems handbook*. Springer, 1–35.

[37] Roy Schwartz, Jesse Dodge, Noah A Smith, and Oren Etzioni. 2019. Green AI. *arXiv preprint arXiv:1907.10597* (2019).

[38] Ying Shan, T Ryan Hoens, Jian Jiao, Haijing Wang, Dong Yu, and JC Mao. 2016. Deep crossing: Web-scale modeling without manually crafted combinatorial features. In *Proceedings of SIGKDD*. ACM, 255–262.

[39] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15, 1 (2014), 1929–1958.

[40] Grigorios Tsoumakas and Ioannis Katakis. 2007. Multi-Label Classification: An Overview. *International Journal of Data Warehousing and Mining* 3, 3 (2007), 1–13.

[41] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. 2017. Deep & cross network for ad click predictions. In *Proceedings of the ADKDD*. ACM, 12.

[42] Xiang Wang, Xiangnan He, Fuli Feng, Liqiang Nie, and Tat-Seng Chua. 2018. Tem: Tree-enhanced embedding model for explainable recommendation. In *Proceedings of WWW*. 1543–1552.

[43] Jun Xiao, Hao Ye, Xiangnan He, Hanwang Zhang, Fei Wu, and Tat-Seng Chua. 2017. Attentional factorization machines: Learning the weight of feature interactions via attention networks. *arXiv preprint arXiv:1708.04617* (2017).

[44] Xin Xin, Bo Chen, Xiangnan He, Dong Wang, Yue Ding, and Joemon Jose. 2019. CFM: Convolutional Factorization Machines for Context-Aware Recommendation. In *Proceedings of IJCAI*.

[45] Xin Xin, Fajie Yuan, Xiangnan He, and Joemon M Jose. 2018. Batch IS NOT Heavy: LearningWord Representations From All Samples. (2018).

[46] Fajie Yuan, Guibing Guo, Joemon M Jose, Long Chen, Haitao Yu, and Weinan Zhang. 2016. Lambdafm: learning optimal ranking with factorization machines using lambda surrogates. In *Proceedings of CIKM*. ACM, 227–236.

[47] Fajie Yuan, Xin Xin, Xiangnan He, Guibing Guo, Weinan Zhang, Chua Tat-Seng, and Joemon M Jose. 2018. fbgd: Learning embeddings from positive unlabeled data with bgd. (2018).