One-class Field-aware Factorization Machines for Recommender Systems with Implicit Feedbacks

BOWEN YUAN, National Taiwan University
MENG-YUAN YANG*, ETH Zurich
JUI-YANG HSIA, National Taiwan University
HONG ZHU, ZHIRONG LIU, and ZEHNHUA DONG, Noah's ark lab, Huawei
CHIH-JEN LIN, National Taiwan University

Recommender systems with implicit feedbacks is a typical one-class scenario, where only positive labels are available. Such positive-unlabeled (PU) learning problems can be solved by one-class matrix factorization (OCMF). Recently, OCMF with side information (OCMFSI) on users and items has been proposed as a powerful extension of OCMF. Interestingly, OCMFSI is strongly related to Factorization Machines (FM), which is a general classification and regression model. This link motivates us to investigate if models superior to FM (e.g., Field-aware Factorization Machines) can be effectively extended to PU learning. In this paper, we propose a novel One-class Field-aware Factorization Machines (OCFFM) model. An efficient optimization algorithm is developed such that OCFFM can be trained on the large-scale data sets. Finally, through experiments on four data sets, OCFFM shows its superiority over other one-class models.

CCS Concepts: • **Information systems** → **Recommender systems**; *Collaborative filtering*; • **Computing methodologies** → **Learning from implicit feedback**; **Factor analysis**; • **Mathematics of computing** → **Nonconvex optimization**.

Additional Key Words and Phrases: Field-aware factorization machine, Block coordinate descent, PU learning

1 INTRODUCTION

Many real-world applications involve recommendation with implicit user feedbacks [15, 24, 30, 32]. This topic, as part of the broader field of positive-unlabeled (PU) learning problems, is now very important in the study of recommender systems.

In a typical PU-learning scenario, we observe some positive (user, item) pairs as the incomplete information of a $0/1$ matrix $Y \in \mathcal{R}^{m \times n}$ of m users and n items. The set of positive pairs, denoted as Ω^+, has $Y_{ij} = 1, \forall (i, j) \in \Omega^+$ and $|\Omega^+| \ll mn$. The goal is to predict whether unobserved pairs are zero or one. To accomplish this, one-class extensions of matrix factorization (MF) have been proposed as an effective technique [15, 24, 30, 32]. It differs from traditional rating-based MF for recommender systems because only one-class information (i.e., observed positive pairs) is available. The rating-based MF finds two low-rank latent matrices

$$W \in \mathcal{R}^{m \times k} \text{ and } H \in \mathcal{R}^{n \times k},$$

such that

$$\boldsymbol{w}_i^T \boldsymbol{h}_j \approx \text{ observed rating of } (i, j), \tag{1}$$

*Work done at Huawei.

Authors' addresses: Bowen Yuan, Department of Computer Science, National Taiwan University, f03944049@csie.ntu.edu.tw; Meng-Yuan Yang, Department of Computer Science, ETH Zurich, meyang@ethz.ch; Jui-Yang Hsia, Department of Computer Science, National Taiwan University, hsiajuiyang5174@gmail.com; Hong Zhu; Zhirong Liu; Zehnhua Dong, Noah's ark lab, Huawei, zhuhong8@huawei.com, liuzhirong@huawei.com, dongzhenhua@huawei.com; Chih-Jen Lin, Department of Computer Science, National Taiwan University, cjlin@csie.ntu.edu.tw.

where k is a specified latent dimension with $k \ll m$ and $k \ll n$, and $\boldsymbol{w}_i \in R^k$ and $\boldsymbol{h}_j \in R^k$ are respectively the ith row of W and jth row of H. That is,

$$W = \begin{bmatrix} \boldsymbol{w}_1^T \\ \vdots \\ \boldsymbol{w}_m^T \end{bmatrix} \text{ and } H = \begin{bmatrix} \boldsymbol{h}_1^T \\ \vdots \\ \boldsymbol{h}_n^T \end{bmatrix}.$$

1.1 Selection of Negative Pairs: Subsampled versus Non-subsampled

Unfortunately, for the one-class scenario finding a model achieving (1) leads to the positive prediction for all unobserved pairs. Existing one-class extensions of MF thus additionally take the unobserved pairs into consideration. Specifically, an intuitive idea is to treat some unobserved pairs as negative. By considering a point-wise loss on each pair, the task is converted to the following conventional rating-based MF problem.

$$\sum_{(i,j)\in\Omega^+\cup\Omega^-} C_{ij}\ell(Y_{ij}, \hat{Y}_{ij}) + \frac{\lambda}{2}(\|W\|_{\mathrm{F}}^2 + \|H\|_{\mathrm{F}}^2), \tag{2}$$

where Ω^- is the set of unobserved pairs selected as negative, we assume true class labels are

$$Y_{ij} = \begin{cases} 1\ , \forall (i,j) \in \Omega^+, \\ r\ , \forall (i,j) \in \Omega^-, \end{cases} \tag{3}$$

the predicted values are $\hat{Y}_{ij} = \boldsymbol{w}_i^T \boldsymbol{h}_j$, $\|\cdot\|_{\mathrm{F}}$ is the Frobenius norm, λ is the regularization parameter, $\ell(y, \hat{y})$ is the loss function, and C_{ij} is a cost parameter. Because all $(i,j) \notin \Omega^+$ are treated as negative, in general $r = 0$ or 1 in (2) is considered. For the selection of Ω^-, currently two approaches are commonly considered in past studies. The first approach [24, 25] is "subsampled" approach. It constructs Ω^- by subsampling partial unobserved pairs, where $|\Omega^+ \cup \Omega^-|$ is generally much smaller than mn. Another widely-used setting [12, 32–34] is to include all unobserved pairs into Ω^-. By following this non-subsampled setting, also named as "non-subsampled" or "full" approach in some literatures, (2) becomes the following optimization problem.

$$\min_{W,H} \quad \sum_{i=1}^{m}\sum_{j=1}^{n} C_{ij}\ell(Y_{ij}, \hat{Y}_{ij}) + \frac{\lambda}{2}(\|W\|_{\mathrm{F}}^2 + \|H\|_{\mathrm{F}}^2). \tag{4}$$

It has been shown that the non-subsampled setting of considering all unobserved pairs gives a better model than the subsampled setting [32]. However, a difficulty to solve (4) is that the loss term involves $O(mn)$ values, which are prohibitive for most large-scale applications. Recently, some training algorithms have been proposed [1, 6, 12, 32, 33] to efficiently solve (4) without any $O(mn)$ cost. With the success of the non-subsampled approach, in this work we focus on (4) and its extensions. Thus in our description we refer to (4) as the one-class MF (OCMF) problem.

Instead of using the point-wise loss, another popular approach called BPR (Bayesian personalized ranking) [30] considers a pair-wise ranking loss. It assumes that for each user i, all observed pairs $(i,j) \in \Omega^+$ should be ranked higher than other unobserved pairs. With a logistic ranking loss, BPR-MF solves the following optimization problem.

$$\min_{W,H} \quad \sum_{(i,j_1)\in\Omega^+,(i,j_2)\notin\Omega^+} \log(1 + e^{-(\hat{Y}_{ij_1} - \hat{Y}_{ij_2})}) + \frac{\lambda}{2}(\|W\|_{\mathrm{F}}^2 + \|H\|_{\mathrm{F}}^2). \tag{5}$$

Because the number of $(i,j) \notin \Omega^+$ is huge, BPR-MF considers stochastic gradient (SG) methods for optimizing (5). At each SG step, an observed pair $(i, j_1) \in \Omega^+$ and an unobserved pair $(i, j_2) \notin \Omega^+$ are sampled for updating $\boldsymbol{w}_i$, $\boldsymbol{h}_{j_1}$ and $\boldsymbol{h}_{j_2}$. As $|\Omega^+| \ll mn$, in the optimization process with a

limited number of epochs, in general only a subset of unobserved pairs can be touched. From this perspective, both the subsampled setting in (2) and the BPR approach can be considered as subsampled approaches but with different loss functions. Experiments in [32] indicate that BPR is also inferior to the non-subsampled setting in (4).

1.2 One-class Models Incorporating Side Information

Although the MF-based approaches discussed above have been useful techniques for recommender systems with implicit feedbacks, there are two main drawbacks for real-world applications. First, MF only learns the latent vectors of the users in Ω^+. For new users, MF is not applicable to provide meaningful recommendations (i.e., a cold-start problem for recommender systems [9]). Second, context information such as user or item features, if available, may improve the model predictability, but MF can not take them into consideration.

To address these issues, OCMF with side information (OCMFSI) has been proposed in [1, 33]. Assume that Y is now a matrix of m contexts and n items with the observed set Ω^+. Each context i is associated with a feature vector $\boldsymbol{u}^i \in \mathcal{R}^{D_u}$ including user and other information (e.g., weather, time) and $\boldsymbol{v}^j \in \mathcal{R}^{D_v}$ is a feature vector of the item j. OCMFSI lets the latent matrices be

$$W \in \mathcal{R}^{D_u \times k} \text{ and } H \in \mathcal{R}^{D_v \times k},$$

and considers

$$Y_{ij} \approx \hat{Y}_{ij} = (W^T \boldsymbol{u}^i)^T (H^T \boldsymbol{v}^j), \forall (i,j) \in \Omega^+. \tag{6}$$

The idea of (6) is to follow the non-subsampled approach in (4) but treat $W^T \boldsymbol{u}_i$ and $H^T \boldsymbol{v}_j$ rather than $\boldsymbol{w}_i$ and $\boldsymbol{h}_j$ as the new latent vectors. We can further consider (6) from a viewpoint of data classification and regression. Let

$$Y_{ij} \text{ as a label and } \boldsymbol{x}^{i,j} = \begin{bmatrix} \boldsymbol{u}^i \\ \boldsymbol{v}^j \end{bmatrix} \text{ as a feature vector.} \tag{7}$$

Then for any test data $\boldsymbol{x} = \left[\begin{smallmatrix} \boldsymbol{u} \\ \boldsymbol{v} \end{smallmatrix}\right]$, the output function is

$$\hat{y}(\boldsymbol{x}) = (W^T \boldsymbol{u})^T (H^T \boldsymbol{v}) = \sum_{j_1=1}^{D_u} \sum_{j_2=D_u+1}^{D_u+D_v} (\boldsymbol{w}_{j_1}{}^T \boldsymbol{h}_{j_2-D_u}) x_{j_1} x_{j_2}. \tag{8}$$

Interestingly, a recent work [1] has noticed that this function is a simplification of a general classification and regression model called Factorization Machines (FM) [28], which has the following output function.[1]

$$\hat{y}_{\text{FM}}(\boldsymbol{x}) = \sum_{j_1=1}^{D_u+D_v} \sum_{j_2=j_1+1}^{D_u+D_v} (\boldsymbol{w}_{j_1}{}^T \boldsymbol{w}_{j_2}) x_{j_1} x_{j_2}, \tag{9}$$

where $\boldsymbol{w}_j, \forall j$ are the latent vectors. If we split the latent vectors to two groups respectively of contexts and items, (9) can be expanded as

$$\sum_{j_1=1}^{D_u} \sum_{j_2=j_1+1}^{D_u} (\cdots) + \sum_{j_1=1}^{D_u} \sum_{j_2=D_u+1}^{Du+D_v} (\cdots) + \sum_{j_1=D_u+1}^{D_u+D_v} \sum_{j_2=j_1+1}^{D_u+D_v} (\cdots). \tag{10}$$

We can see that there exists two kinds of feature conjunctions in (10). The first kind includes those in the first and the third summations, which are respectively self-conjunctions between context features and between item features. The second kind includes those in the second summations, which are cross-conjunctions across context and item features. Clearly, if we drop the self-conjunctions, then (9) is reduced to (8). Following the above connection, Bayer et al. [1] propose a generic

[1] We omit the linear and bias terms of FM in this paper

Table 1. The connections among various one-class models for recommender systems with implicit feedbacks, where OCMF, OCMFSI and OCFM are special cases of the last one, OCFFM.

Approach	Model	Side information	Self conjunction	Field information
Subsampled	BPR-FM	✓	✓	✗
	BPR-FFM	✓	✓	✓
	BPR-DeepFM	✓	✓	✗
Non-subsampled	OCMF [15]	✗	✗	✗
	OCMFSI [1, 33]	✓	✗	✗
	OCFM [1]	✓	✓	✗
	OCFFM (this paper)	✓	✓	✓

framework so that various classification/regression methods (e.g., FM and tensor factorization) can be extended to one-class scenarios. It has been shown in [1] that with side information, the performance of OCMFSI and one-class FM (OCFM) is significantly improved upon OCMF. The techniques developed in [1] to solve optimization problems are coordinate-descent (CD) methods that iteratively update variables corresponding to one latent dimension.

1.3 Motivation and Goal of This Work

Recently, for applications with highly sparse feature (e.g., computational advertising) several classification/regression methods (e.g., filed-aware factorization machine (FFM) [16], DeepFM [11], and deep interest network (DIN) [38]) superior to FM have been proposed. The superiority of these advanced models over FM motivates us to investigate if they can be extended to one-class scenario so that they are better than existing approaches such as OCMFSI and OCFM for PU-learning. Currently, the only optimization method that has been used for these advanced models is stochastic gradient (SG). A straightforward way to extend them to the one-class scenario is to follow approaches applying the subsampled setting. By taking FM, FFM, and DeepFM as examples, a detailed discussion of extending BPR-MF to BPR-FM, BPR-FFM, and BPR-DeepFM is given in Section 5. We will include them in our empirical comparison. However, for the non-subsampled approach shown the superiority over subsampled-based approaches, the task turns out to be challenging. For example, [32, 33] has pointed out issues in applying stochastic gradient for handling the $O(mn)$ instances in OCMF and OCMFSI. Because these advanced models are even more complicated, stochastic gradient is likely not practically viable.

Among these advanced models, FFM extends FM to consider "field" information, so the output function of FFM is

$$\hat{y}(\boldsymbol{x}) = \sum_{j_1=1}^{D} \sum_{j_2=j_1+1}^{D} \boldsymbol{w}_{j_1,f_{j_2}}{}^T \boldsymbol{w}_{j_2,f_{j_1}} x_{j_1} x_{j_2}, \tag{11}$$

where D is the number of features, f_{j_1} and f_{j_2} are the fields of x_{j_1} and x_{j_2}, respectively, $\boldsymbol{w}_{j_1,f_{j_2}} \in \mathcal{R}^k$ is a k-dimensional latent vector of the feature j_1, which learns the latent effect with the interacted features that belong to the field f_{j_2}, and k is a pre-specified value. From (9) and (11), the close relation between FFM and FM inspires us to investigate if we can develop a non-SG optimization method for FFM by referring to past similar efforts on FM. Then through this new optimization method, we can efficiently train one-class FFM (OCFFM) under the non-subsampled setting.

For training general FM for classification/regression, many optimization methods have been proposed. Some of them have been further developed to handle the one-class scenario. For example,

Table 2. Main notation. Upper: for settings of classification and regression. Lower: for settings of recommender systems

L, D, F, k	numbers of instances, features, fields, latent factors
$\boldsymbol{x}, \boldsymbol{x}_{f_1}, y$	feature vector, feature vector of field f_1 and label
D_{f_1}	numbers of features belonging to field f_1
$W_{f_1}^{f_2}$	embedding matrix of field f_1 encoding field f_2
$H_{f_2}^{f_1}$	embedding matrix of field f_2 encoding field f_1
Y	rating matrix
Ω^+	set of observed positive entries
Ω_i^+	set of user (context) i's observed positive entries
m, n	numbers of contexts and items
F_u, F_v	number of fields of contexts and items
$\boldsymbol{u}, \boldsymbol{v}$	feature vector of a context and an item
D_u, D_v	numbers of features belonging to contexts and items

CD has been a common optimization method for FM [3], and [1] investigates how to alleviate the $O(mn)$ cost in the case of OCFM. To choose the method to work on for OCFFM, we discuss existing optimization methods for FM in Section 2, and investigate their advantages and disadvantages. In the end we decide to consider the optimization approach in [8], which follows [3] to use a multi-block convex reformulation of FM.

The method in [8] is a block CD approach. Our main algorithmic developments are in Sections 3 and 4:

- While the eventual goal is to train one-class FFM, a byproduct in Section 3 is that we must develop a block CD algorithm for general FFM first. We believe that this is the first opimization method other than stochastic gradient developed for FFM.
- For one-class scenarios, as in the situation of other one-class models, if an optimization alogrithm is directly applied, a prohibitive $O(mn)$ cost occurs. We have successfully removed this cost in Section 4

In Section 5, we review related works and several related models in the literature, where a summary of their relationship is shown in Table 1. In section 6, based on extensive experiments of comparing one-class models on some large-scale data sets, we show OCFFM performs better than other existing models. Thus it is a novel and useful models for recommender systems with implicit feedbacks. Finally, conclusions and future directions are in Section 7. Table 2 gives main notation in this paper.

2 FIELD-AWARE FACTORIZATION MACHINES AND THEIR OPTIMIZATION

In this section, we review FFM and discuss options for solving its optimization problem.

2.1 A Review of FFM

Both FM and FFM aim to learn coefficients of feature conjunctions. As indicated in Section 1, FFM [16] extends FM by considering the field information. Specifically, the feature vector $\boldsymbol{x}$ is considered as a concatenation of F sub-vectors

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_F \end{bmatrix},$$

where $\boldsymbol{x}_f \in \mathcal{R}^{D_f}$ includes D_f features which belong to the field f. Therefore, (11) can be also written as

$$\hat{y}(\boldsymbol{x}) = \sum_{f_1=1}^{F} \sum_{f_2=f_1}^{F} (W_{f_1}^{f_2}{}^T \boldsymbol{x}_{f_1})^T (W_{f_2}^{f_1}{}^T \boldsymbol{x}_{f_2}) - \sum_{f=1}^{F} \sum_{j=1}^{D_f} \boldsymbol{w}_{j,f}^T \boldsymbol{w}_{j,f} x_j^2, \tag{12}$$

where

$$W_{f_1}^{f_2} = [\boldsymbol{w}_{1,f_2}, \cdots, \boldsymbol{w}_{D_{f_1},f_2}]^T \in \mathcal{R}^{D_{f_1} \times k} \tag{13}$$

is the embedding matrix of the features in field f_1, encoding their interactions to the features in field f_2. To encode all interactions among F fields, there are overall F^2 blocks of parameters. Note that by setting $F = 1$, we can ignore the field information, and FFM is essentially reduced to FM.

Assume $(y_l, \boldsymbol{x}^l), l = 1, ..., L$ are the training set, where y_l is the label or target value, and $\boldsymbol{x}^l$ is a feature vector. The parameters of FFM are determined by solving the following optimization problem,

$$\min_{W_{f_1}^{f_2}, \forall f_1, f_2} \quad \frac{\lambda}{2} \sum_{f_1=1}^{F} \sum_{f_2=1}^{F} \|W_{f_1}^{f_2}\|_{\mathrm{F}}^2 + \sum_{l=1}^{L} C_l \ell(y_l, \hat{y}_l), \tag{14}$$

where

$$\hat{y}_l = \hat{y}(\boldsymbol{x}^l), \tag{15}$$

λ is the regularization parameter, $\boldsymbol{C} \in \mathcal{R}^L$ is the cost parameter, and $\ell(y, \hat{y})$ is a convex loss function in $\hat{y}$ to ensure that $\hat{y}$ can be used to predict or approximate y. For practical regression and classification, the squared loss

$$\ell(y, \hat{y}) = \frac{1}{2}(y - \hat{y})^2 \tag{16}$$

and the logistic loss

$$\ell(y, \hat{y}) = \log(1 + e^{-y\hat{y}}) \tag{17}$$

are commonly used. To solve the optimization problem, Juan et al. [16] propose an algorithm based on Stochastic Gradient (SG) methods. At each SG step, an instance is randomly selected for update with a cost of $O(\mathrm{nnz}(\boldsymbol{x}^l)^2 k)$, where $\mathrm{nnz}(\boldsymbol{x}^l)$ is the number of non-zeros in $\boldsymbol{x}^l$. For SG usually an epoch refers to L updates, so the cost of per epoch is

$$O(\mathrm{nnz}(\boldsymbol{x})^2 L k), \tag{18}$$

where $O(\mathrm{nnz}(\boldsymbol{x}))$ is the average number of non-zeros in all training instances $\boldsymbol{x}^1, \cdots, \boldsymbol{x}^L$.

Different from FM where each feature has only one latent vector to learn the latent effect with any other features, by considering the filed information, in FFM, each feature has several latent vectors for other features belonging to different fields. It has been reported [16] that this design can greatly boost the performance for data sets with highly sparse features such as identity and categorical features widely used in in recommender systems.

2.2 Issues for One-class Extension of FFM

If we follow (7) to treat OCFFM as a special case of FFM, the instance index l in Section 2.1 is changed to (i, j) and the following optimization problem can be considered.

$$\min_{W_{f_1}^{f_2}, \forall f_1, f_2} \quad \frac{\lambda}{2} \sum_{f_1=1}^{F} \sum_{f_2=1}^{F} \|W_{f_1}^{f_2}\|_{\mathrm{F}}^2 + \sum_{i=1}^{m} \sum_{j=1}^{n} C_{ij} \ell(Y_{ij}, \hat{Y}_{ij}), \tag{19}$$

where Y_{ij} is the label,

$$\hat{Y}_{ij} = \hat{y}(\boldsymbol{x}^{i,j}), \text{ and } \boldsymbol{x}^{i,j} = \begin{bmatrix} \boldsymbol{u}^i \\ \boldsymbol{v}^j \end{bmatrix}. \tag{20}$$

Here the output function $\hat{y}(\cdot)$ is the FFM in (11). We follow past works [32, 33] to set C_{ij} as

$$C_{ij} = \begin{cases} 1 & (i,j) \in \Omega^+, \\ \omega & (i,j) \notin \Omega^+. \end{cases} \tag{21}$$

By the above setting, we transform a recommendation problem to a regression or classification problem of mn training instances.

From (18), the cost of one epoch in solving (19) by SG is $O(\text{nnz}(\boldsymbol{x})^2 mnk)$. Because mn is huge, the cost per SG epoch is extremely high. The same issue has occurred in training OCMF by SG [32], where they made efforts to make SG practical but failed. We think the same situation still applies here. Therefore, designing some new algorithms for FFM is a essential precondition to make OCFFM a feasible approach for recommender systems.

2.3 Existing Optimization Methods for Training FM

To develop methods other than SG for FFM, we review techniques that have been considered for FM because as mentioned in Section 2.1, FM is a special case of FFM by setting $F = 1$. In particular, we check the following two efficient non-SG algorithms for learning FM, coordinate decent (CD) algorithm [3, 29] and block CD algorithm [8].

For the CD algorithm, at each step variables corresponding to one latent dimension are updated by solving an associated subproblem while other variables are fixed. In one cycle all latent dimensions are considered and the algorithm runs through cycles until some stopping condition is met. On the other hand, the main idea of the block CD algorithm for FM is to slightly modify the output function from (9) to

$$\hat{y}_{\text{FM}}(\boldsymbol{x}) = (W^T\boldsymbol{x})^T(H^T\boldsymbol{x}), \tag{22}$$

where $H \in \mathcal{R}^{D\times k}$ is introduced such that (22) is a multi-block convex function [3]. Then some efficient algorithms alternatively minimizing over W and H can be applied for training FM. That is, for the two blocks W and H, if one is considered as the variable while the other is fixed, then (22) is linear and the optimization problem is convex. Some dtailed comparisons between CD and block CD for FM have been given in [8]. They show that block CD possesses the following advantages.

- The sub-problem of the CD approach involves only a vector variable associated with a latent dimension, while the sub-problem of block CD involves about half of all the variables stored as a matrix. Thus in solving the sub-problem of block CD, it is easier to apply matrix-based operations. For example, [8] considers a truncated Newton method to solve each sub-problem. Then not only is optimized implementations for matrix operations (e.g., optimized BLAS such as Intel® Math Kernel Library) can be applied, but also the parallelization is easier.
- For some loss functions such as the logistic loss defined in (17), CD suffers from a relatively large number of exponential/logarithmic operations. In this case, CD is not competitive because each exponential/logarithmic operation costs much more than a basic operation. This issue has been well studied in [8, 35].

From these past works, we can see that both CD and block CD algorithms may be extended into solving FFM. However, in consideration of CD's limitations discussed above, instead of investigating CD to solve (19), our strategy here is to follow [8] to slightly modify the FFM problem so that it may be easier to create effective optimization methods based on block CD.

3 BLOCK CD ALGORITHMS FOR LEARNING FFM

We investigate if FFM can be slightly modified so that the optimization problem becomes multi-block convex. We then develop an efficient optimization method for training FFM.

Algorithm 1 Solving the FFM problem (25) via a block coordinate descent method.

Given initial $W_{f_1}^{f_2}, H_{f_2}^{f_1}, \forall 1 \le f_1 \le f_2 \le F$.
while stopping condition is not satisfied **do**
  for $f_1 \leftarrow \{1 \cdots F\}$ **do**
    for $f_2 \leftarrow \{f_1 \cdots F\}$ **do**
      Solve (25) by fixing all blocks except $W_{f_1}^{f_2}$
      Solve (25) by fixing all blocks except $H_{f_2}^{f_1}$
    end for
  end for
end while

3.1 Multi-block Convex Reformulation of FFM

Inspired by the idea of [8], we investigate how FFM can be modified to a multi-block convex form. Interestingly, if $f_1 \neq f_2$, then each

$$(W_{f_1}^{f_2 T} \boldsymbol{x}_{f_1})^T (W_{f_2}^{f_1 T} \boldsymbol{x}_{f_2}) \tag{23}$$

in (12) is already in the same form in (22). Thus all we need is to modify the term of $f_1 = f_2$ by introducing a new block. Let

$$H_{f_2}^{f_1} = \begin{cases} W_{f_2}^{f_1} & f_2 > f_1, \\ [\boldsymbol{h}_{1,f_1}, \cdots, \boldsymbol{h}_{D_{f_1},f_1}]^T & f_1 = f_2, \end{cases}$$

where $[\boldsymbol{h}_{1,f_1}, \cdots, \boldsymbol{h}_{D_{f_1},f_1}]^T$ is new. Then

$$\begin{aligned} \hat{y}(\boldsymbol{x}) &= \sum_{f_1=1}^{F} \sum_{f_2=f_1}^{F} (W_{f_1}^{f_2 T} \boldsymbol{x}_{f_1})^T (H_{f_2}^{f_1 T} \boldsymbol{x}_{f_2}) \\ &= \sum_{f_1=1}^{F} \sum_{f_2=f_1}^{F} \boldsymbol{p}_{f_1,f_2}{}^T \boldsymbol{q}_{f_2,f_1}, \end{aligned} \tag{24}$$

where

$$\boldsymbol{p}_{f_1,f_2} = W_{f_1}^{f_2 T} \boldsymbol{x}_{f_1} \text{ and } \boldsymbol{q}_{f_2,f_1} = H_{f_2}^{f_1 T} \boldsymbol{x}_{f_2}.$$

The optimization problem over all blocks is

$$\min_{W_{f_1}^{f_2}, H_{f_2}^{f_1}, \forall f_1 \le f_2} \quad \frac{\lambda}{2} \sum_{f_1=1}^{F} \sum_{f_2=f_1}^{F} (\|W_{f_1}^{f_2}\|_{\mathrm{F}}^2 + \|H_{f_2}^{f_1}\|_{\mathrm{F}}^2) + \sum_{l=1}^{L} C_l \ell(y_l, \hat{y}_l), \tag{25}$$

which is multi-block convex. The total number of blocks is only slightly increased from F^2 to $F^2 + F$.

3.2 Block CD Method for FFM with the Squared Loss

With the reformulation, (25) becomes a multi-block convex optimization problem and can be solved by block CD methods. We obtain $(F + 1)F$ convex sub-problems by sequentially selecting one block and fixing the others. Then (25) can be solved by cyclically minimizing sub-problems. We summarize the procedure in Algorithm 1. For simplicity, we assume that the squared loss is used though extensions to other losses are possible.

Next we discuss how to solve each sub-problem. The objective function of the sub-problem is now convex quadratic because of the squared loss. We analyze the quadratic function and show that a linear system must be solved. If $W_{f_1}^{f_2}$ is the block to be updated and S is the imposed change, then the new value is

$$W_{f_1}^{f_2} + S \in \mathcal{R}^{D_{f_1} \times k}. \tag{26}$$

The output value $\hat{y}_l$ is changed to

$$\hat{y}_l + (S^T \boldsymbol{x}_{f_1}^l)^T \boldsymbol{q}_{f_2,f_1}^l = \hat{y}_l + \boldsymbol{x}_{f_1}^{l\,T} S \boldsymbol{q}_{f_2,f_1}^l. \tag{27}$$

From (26)-(27), the objective function of the corresponding sub-problem can be written as

$$\frac{\lambda}{2} \|W_{f_1}^{f_2} + S\|_{\mathrm{F}}^2 + \frac{1}{2} \sum_{l=1}^{L} C_l \big(\hat{y}_l + \boldsymbol{x}_{f_1}^{l\,T} S \boldsymbol{q}_{f_2,f_1}^l - y_l\big)^2. \tag{28}$$

Therefore, the sub-problem is a quadratic minimization over a matrix variable S. For easy analysis, we rewrite (28) to a vector form in (31) by defining

$$\tilde{\boldsymbol{s}} = \mathrm{vec}(S), \tilde{\boldsymbol{w}} = \mathrm{vec}(W_{f_1}^{f_2}), \tilde{\boldsymbol{x}}^l = \boldsymbol{q}_{f_2,f_1}^l \otimes \boldsymbol{x}_{f_1}^l, \tag{29}$$

and

$$\tilde{\boldsymbol{y}} = [\hat{y}_1 - y_1, \cdots, \hat{y}_L - y_L]^T \in \mathcal{R}^L, \tag{30}$$

where $\mathrm{vec}(\cdot)$ stacks all columns of a matrix to a vector and $\otimes$ is the Kronecker product.

$$(28) = \frac{\lambda}{2} \|\tilde{\boldsymbol{w}} + \tilde{\boldsymbol{s}}\|^2 + \frac{1}{2} \sum_{l=1}^{L} C_l \big(\tilde{y}_l + \tilde{\boldsymbol{s}}^T \tilde{\boldsymbol{x}}^l\big)^2. \tag{31}$$

Note that we use

$$\boldsymbol{v}_1^T M \boldsymbol{v}_2 = \mathrm{vec}(M)^T (\boldsymbol{v}_2 \otimes \boldsymbol{v}_1) \tag{32}$$

to have

$$\tilde{\boldsymbol{s}}^T \tilde{\boldsymbol{x}}^l = \boldsymbol{x}_{f_1}^{l\,T} S \boldsymbol{q}_{f_2,f_1}^l.$$

Now (31) is a simple quadratic function of $\tilde{\boldsymbol{s}}$. Taking the gradient (i.e., the first derivative) to zero leads to the following linear system

$$\lambda(\tilde{\boldsymbol{w}} + \tilde{\boldsymbol{s}}) + \sum_{l=1}^{L} C_l (\tilde{y}_l + \tilde{\boldsymbol{s}}^T \tilde{\boldsymbol{x}}^l) \tilde{\boldsymbol{x}}^l = \boldsymbol{0}.$$

We can rewrite it in a matrix-based form by defining

$$\tilde{X} = \big[\tilde{\boldsymbol{x}}^1, \cdots, \tilde{\boldsymbol{x}}^L\big]^T \in \mathcal{R}^{L \times D_{f_1} k} \text{ and } \tilde{C} = \begin{bmatrix} C_1 & & \\ & \ddots & \\ & & C_L \end{bmatrix}$$

as a diagonal matrix.

$$(\lambda I + \tilde{X}^T \tilde{C} \tilde{X}) \tilde{\boldsymbol{s}} = -(\tilde{X}^T \tilde{C} \tilde{\boldsymbol{y}} + \lambda \tilde{\boldsymbol{w}}), \tag{33}$$

where I is identity matrix.

Up to now we have shown that solving a sub-problem is equivalent to simply solving a linear system. However, for large-scale applications, not only is the matrix in (33) too large to be stored in the computer memory, but also the matrix inversion is computationally expensive. Following past works (e.g., [8, 21]), by taking the special structure of the linear system (33) into account, the conjugate gradient (CG) method can be applied to solve the linear system without explicitly storing $\tilde{X}^T \tilde{C} \tilde{X}$. CG [13] is one of the most used iterative methods for solving a linear system. The main

Algorithm 2 Conjugate gradient procedure for approximately solving the sub-problem of $W_{f_1}^{f_2}$. Note that $\langle \cdot, \cdot \rangle_{\mathrm{F}}$ is the Frobenius inner product between two matrices.

Given $\xi \le 1$, X_{f_1}, $\tilde{C}$, $\tilde{\boldsymbol{y}}$, $Q_{f_2}^{f_1}$, λ, and current $W_{f_1}^{f_2}$.
$G \leftarrow X_{f_1}^T \operatorname{diag}(\tilde{C}\tilde{\boldsymbol{y}}) Q_{f_2}^{f_1} + \lambda W_{f_1}^{f_2}$
Let $R \leftarrow -G$, $V \leftarrow R$, $S \leftarrow \mathbf{0}_{D_{f_1} \times k}$ and $\gamma \leftarrow \|R\|_{\mathrm{F}}^2$.
while True **do**
  if $\|R\|_{\mathrm{F}} \le \xi \|G\|_{\mathrm{F}}$ **then**
    break
  end if
  $\boldsymbol{z} \leftarrow ((X_{f_1} V) \odot Q_{f_2}^{f_1}) \mathbf{1}_{k \times 1}$
  $Z \leftarrow X_{f_1}^T \operatorname{diag}(\tilde{C}\boldsymbol{z}) Q_{f_2}^{f_1} + \lambda V$
  $\alpha \leftarrow \gamma / \langle V, Z \rangle_{\mathrm{F}}$
  $S \leftarrow S + \alpha V$
  $R \leftarrow R - \alpha Z$
  $\gamma^{\mathrm{new}} \leftarrow \|R\|_{\mathrm{F}}^2$
  $\beta \leftarrow \gamma^{\mathrm{new}} / \gamma$
  $V \leftarrow R + \beta V$
  $\gamma \leftarrow \gamma^{\mathrm{new}}$
end while
$W_{f_1}^{f_2} \leftarrow W_{f_1}^{f_2} + S$

operation is a sequence of products between the matrix of the linear system and a vector $\tilde{\boldsymbol{v}}$, where $\tilde{\boldsymbol{v}}$ is the variable used in the CG procedure. By the following computation sequence,

$$(\lambda I + \tilde{X}^T \tilde{C} \tilde{X}) \tilde{\boldsymbol{v}} = \lambda \tilde{\boldsymbol{v}} + \tilde{X}^T \left(\tilde{C} (\tilde{X} \tilde{\boldsymbol{v}}) \right), \tag{34}$$

the explicit storing of the matrix $\tilde{X}^T \tilde{C} \tilde{X}$ can be avoided.

3.3 Solving the Linear System (33) by Matrix Operations

In Section 3.2, we derive the sub-problems (31) so that the variable is a vector $\tilde{\boldsymbol{s}}$. Then we can easily get the gradient (i.e., the first derivative) with respect to $\tilde{\boldsymbol{s}}$ for obtaining the linear system (33). However, for efficient implementations, we should not directly consider the matrix $\tilde{X}$, which is a large matrix coming from $\tilde{\boldsymbol{x}}^l$ defined in (29). In fact $\tilde{X}$ never needs to be formed. A similar situation has occurred in [8] for training FM. Here we give a general description so that not only is the procedure in [8] covered, but also we can use it for the development in one-class scenarios in Section 4. The main result is in the following theorem.

Theorem 3.1. *Consider*

$$\boldsymbol{x}^l \in \mathcal{R}^D, \boldsymbol{q}^l \in \mathcal{R}^k, \ l = 1, \cdots, L,$$

and

$$\tilde{X} = \left[\boldsymbol{q}^1 \otimes \boldsymbol{x}^1, \cdots, \boldsymbol{q}^L \otimes \boldsymbol{x}^L \right]^T \in \mathcal{R}^{L \times Dk}.$$

Then for any $\tilde{\boldsymbol{y}} \in \mathcal{R}^L$ and any diagonal matrix $\tilde{C} \in \mathcal{R}^{L \times L}$, we have

$$\tilde{X}^T \tilde{C} \tilde{\boldsymbol{y}} = \operatorname{vec}(X^T \operatorname{diag}(\tilde{C}\tilde{\boldsymbol{y}}) Q) = \operatorname{vec}(\sum_{l=1}^{L} \boldsymbol{x}^l C_l \tilde{y}_l \boldsymbol{q}^{l^T}), \tag{35}$$

where

$$X = \left[\boldsymbol{x}^1, \cdots, \boldsymbol{x}^L\right]^T \in \mathcal{R}^{L\times D}, Q = \left[\boldsymbol{q}^1, \cdots, \boldsymbol{q}^L\right]^T \in \mathcal{R}^{L\times k},$$

and $\operatorname{diag}(\tilde{C}\tilde{\boldsymbol{y}})$ *is a diagonal matrix with elements of the vector* $\tilde{C}\tilde{\boldsymbol{y}}$ *on the diagonal.*

Further, for any matrix $V \in R^{D\times k}$,

$$\tilde{X}^T\tilde{C}\tilde{X}\operatorname{vec}(V) = \operatorname{vec}(X^T\operatorname{diag}(\tilde{C}\boldsymbol{z})Q), \tag{36}$$

where

$$\boldsymbol{z} = ((XV)\odot Q)\mathbf{1}_{k\times 1} \text{ with } z_l = \boldsymbol{x}^{l^T}V\boldsymbol{q}^l, l = 1, \cdots, L, \tag{37}$$

the $\odot$ *symbol is the Hadamard (i.e., element-wise) product between two matrices, and* $\mathbf{1}_{k\times 1}$ *is the vector of ones.*

The proof is in Section A of supplementary materials. From the definition of $\tilde{X}$, X, and Q in Theorem 3.1, clearly if $\tilde{X}$ is formed, the storage is k-fold of X and Q. Thus our results in Theorem 3.1 effectively reduces the memory consumption.

Now we are ready to present in Algorithm 2 the CG procedure to solve the linear system (33). The algorithm is the same as the CG procedure in any textbook (e.g., [10]), but every operation is matrix rather than vector-based. In particular, by defining

$$X_{f_1} = \left[\boldsymbol{x}^1_{f_1}, \cdots, \boldsymbol{x}^L_{f_1}\right]^T \text{ and } Q^{f_1}_{f_2} = \left[\boldsymbol{q}^1_{f_2,f_1}, \cdots, \boldsymbol{q}^L_{f_2,f_1}\right]^T, \tag{38}$$

we can apply Theorem 3.1 to obtain the right-hand side of the linear system at line 2 and conduct each matrix-vector product in line 8-9 of Algorithm 2.

For the stopping condition of CG iterations, a common setting is to check if the error is relatively smaller than the norm of the right-hand side of the linear system in (33).

$$\|(\tilde{X}^T\tilde{C}\tilde{\boldsymbol{y}} + \lambda\tilde{\boldsymbol{w}}) + (\lambda I + \tilde{X}^T\tilde{C}\tilde{X})\tilde{\mathbf{s}}\| \le \xi\|\tilde{X}^T\tilde{C}\tilde{\boldsymbol{y}} + \lambda\tilde{\boldsymbol{w}}\|, \tag{39}$$

where $0 < \xi \le 1$ is a small stopping tolerance. For Algorithm 3 that conducts the CG procedure in a form of matrix operations, we also tranform (39) to a matrix form. For example, from (35) and (38), right-hand side in (39) is the same as

$$\|X^T_{f_1}\operatorname{diag}(\tilde{C}\tilde{\boldsymbol{y}})Q^{f_2}_{f_1} + \lambda W^{f_2}_{f_1}\|_{\mathrm{F}};$$

see line 5-7 in Algorithm 2.

4 BLOCK CD METHOD FOR OCFFM

Because OCFFM is a special case of FFM, block CD can be directly applied. That is, all sub-problems in Algorithm 1 can be formed in (31) and solved by Algorithm 2. However, the total number of instances in the one-class scenario is mn, which is tremendously large. The reason is that from (20) each instance $\boldsymbol{x}^l$ in FFM is now indexed by (i, j):

$$\boldsymbol{x}^l \rightarrow \boldsymbol{x}^{i,j}.$$

From and (37) the cost of the matrix products in (35) and (36) are prohibitively proportional to mn. Subsequently we show that properties of one-class problems can be used to avoid the $O(mn)$ complexity.

To begin, we check the right-hand side of the linear system (33). From (29), (30) and (35), by rewriting index l to i, j, we have

$$\tilde{y}_l \rightarrow \hat{Y}_{ij} - Y_{ij}, C_l \rightarrow C_{ij}$$

and what we must calculate now is

$$\sum_{i=1}^{m}\sum_{j=1}^{n} \boldsymbol{x}_{f_1}^{i,j} C_{ij}(\hat{Y}_{ij} - Y_{ij}) \boldsymbol{q}_{f_2,f_1}^{i,j}{}^{T}. \tag{40}$$

To avoid the $O(mn)$ summations, the main idea is that in (40), some values indexed by i, j may depend only on i or j. Suppose the features of contexts and items can be respectively grouped into F_u, F_v fields. Accordingly, we have

$$\boldsymbol{u}^i = \begin{bmatrix} \boldsymbol{u}_1^i \\ \vdots \\ \boldsymbol{u}_{F_u}^i \end{bmatrix} \text{ and } \boldsymbol{v}^j = \begin{bmatrix} \boldsymbol{v}_{F_u+1}^j \\ \vdots \\ \boldsymbol{v}_{F_u+F_v}^j \end{bmatrix}.$$

From (7), $\boldsymbol{x}_f^{i,j}$ can be represented by considering the range of the field f as:

$$\boldsymbol{x}_f^{i,j} = \begin{cases} \boldsymbol{u}_f^i & f \le F_u, \\ \boldsymbol{v}_f^j & f > F_u. \end{cases} \tag{41}$$

Now we see for example, if $f \le F_u$, then $\boldsymbol{x}_f^{i,j}, \forall j$ share the same context feature vector $\boldsymbol{u}_f^i$. Similarly, $\boldsymbol{q}_{f_2,f_1}^{i,j}$ can be just written as $\boldsymbol{q}_{f_2,f_1}^{i}$ or $\boldsymbol{q}_{f_2,f_1}^{j}$ depending on f_2:

$$\boldsymbol{q}_{f_2,f_1}^{i,j} \to \begin{cases} \boldsymbol{q}_{f_2,f_1}^{i} = H_{f_2}^{f_1} \boldsymbol{u}_{f_2}^i & f_2 \le F_u, \\ \boldsymbol{q}_{f_2,f_1}^{j} = H_{f_2}^{f_1} \boldsymbol{v}_{f_2}^j & f_2 > F_u. \end{cases} \tag{42}$$

Therefore, the calculation of (40) is separated to the following possible situations of f_1 and f_2:

$$\begin{cases} f_1, f_2 \le F_u, \\ f_1, f_2 > F_u, \\ f_1 \le F_u, f_2 > F_u. \end{cases} \tag{43}$$

The calculation is very complicated, so we leave most details in supplementary materials. Here we only check the first situation as an illustration.

With (41) and (42), if $1 \le f_1, f_2 \le F_u$, (40) can be written as

$$\sum_{i=1}^{m}(\sum_{j=1}^{n} C_{ij}(\hat{Y}_{ij} - Y_{ij})) \boldsymbol{u}_{f_1}^i \boldsymbol{q}_{f_2,f_1}^{i}{}^{T}. \tag{44}$$

We investigate the summation over j in detail. From the definition of Y_{ij} and C_{ij} in (3) and (21) respectively,

$$\begin{aligned} &\sum_{j=1}^{n} C_{ij}(\hat{Y}_{ij} - Y_{ij}) = \sum_{j \in \Omega_i^+} (\hat{Y}_{ij} - 1) + \sum_{j \notin \Omega_i^+} \omega(\hat{Y}_{ij} - r) \\ &= \sum_{j \in \Omega_i^+} (\hat{Y}_{ij} - 1 - \omega(\hat{Y}_{ij} - r)) + \omega \sum_{j=1}^{n} (\hat{Y}_{ij} - r) \\ &= \sum_{j \in \Omega_i^+} ((1-\omega)\hat{Y}_{ij} - 1 + \omega r) + \omega \sum_{j=1}^{n} \hat{Y}_{ij} - \omega n r, \end{aligned} \tag{45}$$

where Ω_i^+ is the set including the observed items of the ith context. The bottleneck is the $O(n)$ term $\sum_{j=1}^{n} \hat{Y}_{ij}$ because $|\Omega_i^+| \ll n$. To address this bottleneck we show that each Y_{ij} can be decomposed to terms solely related to i or j. From (24),

$$\hat{Y}_{ij} = \sum_{\alpha=1}^{F} \sum_{\beta=\alpha}^{F} \boldsymbol{p}_{\alpha,\beta}^{i,j\ T} \boldsymbol{q}_{\beta,\alpha}^{i,j}. \tag{46}$$

Similar to (42) for $\boldsymbol{q}_{\beta,\alpha}^{i,j}$, we also have that $\boldsymbol{p}_{\alpha,\beta}^{i,j}$ only depends on either i or j.

$$\boldsymbol{p}_{\alpha,\beta}^{i,j} \rightarrow \begin{cases} \boldsymbol{p}_{\alpha,\beta}^{i} = W_{\alpha}^{\beta T} \boldsymbol{u}_{\alpha}^{i} & \alpha \leq F_u, \\ \boldsymbol{p}_{\alpha,\beta}^{j} = W_{\alpha}^{\beta T} \boldsymbol{v}_{\alpha}^{j} & \alpha > F_u. \end{cases} \tag{47}$$

Then we can split the summation in (46) to three parts according to (43).

$$\sum_{\alpha=1}^{F} \sum_{\beta=\alpha}^{F} \boldsymbol{p}_{\alpha,\beta}^{i,j\ T} \boldsymbol{q}_{\beta,\alpha}^{i,j} = a_i + b_j + \sum_{\alpha=1}^{F_u} \sum_{\beta=F_u+1}^{F} \boldsymbol{p}_{\alpha,\beta}^{i\ T} \boldsymbol{q}_{\beta,\alpha}^{j}, \tag{48}$$

where

$$a_i = \sum_{\alpha=1}^{F_u} \sum_{\beta=\alpha}^{F_u} \boldsymbol{p}_{\alpha,\beta}^{i\ T} \boldsymbol{q}_{\beta,\alpha}^{i} \text{ and } b_j = \sum_{\alpha=F_u+1}^{F} \sum_{\beta=\alpha}^{F} \boldsymbol{p}_{\alpha,\beta}^{j\ T} \boldsymbol{q}_{\beta,\alpha}^{j}. \tag{49}$$

Then

$$\sum_{j=1}^{n} \hat{Y}_{ij} = n a_i + \sum_{j=1}^{n} b_j + d_i, \tag{50}$$

where

$$d_i = \sum_{\alpha=1}^{F_u} \sum_{\beta=F_u+1}^{F} \boldsymbol{p}_{\alpha,\beta}^{i\ T} \Big(\sum_{j=1}^{n} \boldsymbol{q}_{\beta,\alpha}^{j} \Big). \tag{51}$$

A crucial observation is that

$$\sum_{j=1}^{n} b_j \text{ and } \sum_{j=1}^{n} \boldsymbol{q}_{\beta,\alpha}^{j}$$

are now independent of i. Thus if they are pre-calculated, then $\sum_{j=1}^{n} \hat{Y}_{ij}$ can be obtained in $O(1)$ cost and therefore the $O(mn)$ number of summations in (44) no longer occurs. Of course how to obtain and maintain values like a_i, b_j, and $\boldsymbol{d}$ is an important issue. See the discussion below and details in supplementary materials. We now put (50) back to (45). By defining

$$U_{f_1} = \left[\boldsymbol{u}_{f_1}^{1}, \cdots, \boldsymbol{u}_{f_1}^{m} \right]^T, Q_{f_2}^{f_1} = \left[\boldsymbol{q}_{f_2,f_1}^{1}, \cdots, \boldsymbol{q}_{f_2,f_1}^{m} \right]^T, \tag{52}$$

we can calculate (44) by the following matrix product.

$$U_{f_1}^T \operatorname{diag}(\boldsymbol{z}) Q_{f_2}^{f_1}, \tag{53}$$

where

$$z_i = \sum_{j \in \Omega_i^+} \left((1-\omega) \hat{Y}_{ij} - 1 + \omega r \right) + \omega \big(n a_i + \sum_{j=1}^{n} b_j + d_i - n r \big). \tag{54}$$

For the matrix-vector products in (36), details are shown in Section B of supplementary materials. Here we give only the results of the first case ($1 \leq f_1, f_2 \leq F_u$).

$$\tilde{X}^T \tilde{C} \tilde{X} \operatorname{vec}(V) = \operatorname{vec}(U_{f_1}^T \operatorname{diag}(\boldsymbol{z}) Q_{f_2}^{f_1}), \tag{55}$$

where

$$\boldsymbol{z} = D\left((U_{f_1}V) \odot Q^{f_1}_{f_2}\right)\mathbf{1}_{k\times 1}, \tag{56}$$

and $D \in \mathcal{R}^{m\times m}$ is a diagonal matrix with $D_{ii} = (1-\omega)|\Omega^+_i| + \omega n$. Similar to (53), the calculation no longer depends on $O(mn)$ summations.

With (53) and (55), we extend the CG procedure in Algorithm 2 for the one-class scenario in Algorithm 3.

To have a complete block CD procedure, we must address the following implementation issues.

(1) Under the same f_1 and f_2, two sub-problems on variable blocks $W^{f_2}_{f_1}$ and $H^{f_1}_{f_2}$ are solved. The two sub-problems are in the same form, so Algorithm 3 can be applied on both. So far we have discussed details when $1 \le f_1, f_2 \le F_u$. The situation is similar if $f_1, f_2 > F_u$, though suitable input information for each call of Algorithm 3 must be carefully specified.
For the situation of $f_1 \le F_u, f_2 > F_u$, many details are different, so an algorithm other than Algorithm 3 must be developed.
(2) We mentioned that in calculating (54), values such as $\boldsymbol{a}$, $\boldsymbol{b}$, and $\boldsymbol{d}$ are needed. How to efficiently maintain them is an import issue.

We give details in Section B of supplementary materials and show that the cost to go through all blocks of parameters is

$$\begin{aligned} &O\left((\text{nnz}(U_f \text{ or } V_f) + |\Omega^+|)k + (D_f + m + n)k^2\right) \\ &\quad \times \text{ avg. \# CG steps per sub-problem } \times F(F+1), \end{aligned} \tag{57}$$

where D_f is the average number of context/item features of a field, and $\text{nnz}(U_f)$ or $\text{nnz}(V_f)$ is the average number of non-zeros in the feature matrix corresponding to a field f. Note that U_f is defined in (52), while V_f is defined in supplementary Section B. As a comparison, for SG, from (18) and the set of mn instances, the cost of one epoch is $O(\text{nnz}(\boldsymbol{x})^2 mnk)$, where $\text{nnz}(\boldsymbol{x})$ is the average number of non-zeros in all $\boldsymbol{x}^{i,j}$. While one SG epoch and one cycle of the block CD method are not exactly comparable, we see the mn term of SG is prohibitive.

5 A DISCUSSION ON MODELS FOR RECOMMENDER SYSTEMS WITH IMPLICIT FEEDBACK

In this section, after a brief review on related works, we discuss models that will be included in our experiments.

As we summarized in Section 1, by differing on whether all unobserved pairs are considered for the model update, existing works of recommender systems with implicit feedback can be broadly categorized into two groups respectively considering the subsampled setting (e.g., negatvie sampling and BPR approaches) and the non-subsampled setting.

For works considering the subsampled setting, many of them have focused on developing different subsampled schemes. For example, the uniform sampling [24, 30], the static sampling [24, 25], the adaptive sampling [2, 37] and the dynamic sampling [36]. A recent work [20] improves some schemes by the importance subsampling technique. Another line of works focuses on distinguishing non-negative pairs from unobserved ones through auxiliary feedback [27], and collaborative and social information [22, 37, 39]. While most works only consider MF, some ones propose to leverage the side information. For example, Cai et al. [4] improve MF by additionally considering each user's neighborhood information.

On the other hand, for works considering the non-subsampled setting, the main line is to incorporate more models into the non-subsampled framework, which can range from simple MF [6, 12, 23, 32] to context-aware models (e.g., MFSI and FM [1, 5, 33]). In addition to the point-wise

Algorithm 3 Conjugate gradient procedure for approximately solving the sub-problem of $W_{f_1}^{f_2}$, where $f_1, f_2 \le F_u$. The complexity of each major operation is shown at the end of the line.

Given input U_{f_1}, $W_{f_1}^{f_2}$, $Q_{f_2}^{f_1}$, $\boldsymbol{d}, \boldsymbol{a}, \bar{b} = \sum_{j=1}^{n} b_j$, and $\hat{Y}_{ij}, \forall (i,j) \in \Omega^+$
// Calculate the right-hand side of the linear system
for $i \leftarrow \{1, \cdots, m\}$ **do** $\cdots O(m + |\Omega^+|)$
  $z_i \leftarrow \omega(na_i + \bar{b} + d_i - nr)$
  $z_i \leftarrow z_i + \sum_{j \in \Omega_i^+}((1-\omega)\hat{Y}_{ij} - 1 + \omega r)$
end for
$G \leftarrow U_{f_1}^T \operatorname{diag}(\boldsymbol{z}) Q_{f_2}^{f_1} + \lambda W$ $\cdots O(\operatorname{nnz}(U_{f_1})k + D_{f_1}k)$
Let $R \leftarrow -G$, $V \leftarrow R$ and $\gamma \leftarrow \|R\|_{\mathrm{F}}^2$
while True **do**
  if $\|R\|_{\mathrm{F}} \le \xi \|G\|_{\mathrm{F}}$ **then**
    break
  end if
  //Compute and store the matrix-vector product to Z
  $\boldsymbol{z} = \left((U_{f_1} V) \odot Q_{f_2}^{f_1}\right) \mathbf{1}_{k \times 1}$ $\cdots O(mk + \operatorname{nnz}(U_{f_1})k)$
  for $i \leftarrow \{1, \cdots, m\}$ **do** $\cdots O(m)$
    $z_i \leftarrow z_i((1-\omega)|\Omega_i^+| + \omega n)$
  end for
  $Z \leftarrow U_{f_1}^T \operatorname{diag}(\boldsymbol{z}) Q_{f_2}^{f_1} + \lambda V$ $\cdots O(\operatorname{nnz}(U_{f_1})k)$
  Step 10-16 in Algorithm 2
end while

loss applied in most works, a recent work [7] considers a pair-wise ranking loss between observed pairs and unobserved pairs. For optimization methods for the non-subsampled setting, besides the alternating-based algorithms applied in this work, some works consider the gradient-based algorithms (e.g., [5, 6, 18]). However, a study given in [8] shows that for (25) such multi-block convex problems, cyclically solving each sub-problem by a second-order method can be much more efficient than the gradient-based algorithms.

For our experiments, we first consider the following models obtained by the non-subsampled approach. Specifically, to check the effectiveness of the field information, we consider a setting without such information by using $F_u = F_v = 1$ in OCFFM. From (24), the output function becomes

$$\begin{aligned}\hat{y}(\boldsymbol{x}) =& (W_{f_C}^{f_C T}\boldsymbol{x}_{f_C})^T (H_{f_C}^{f_C T}\boldsymbol{x}_{f_C}) + (W_{f_C}^{f_I T}\boldsymbol{x}_{f_C})^T (H_{f_I}^{f_C T}\boldsymbol{x}_{f_I}) \\ &+ (W_{f_I}^{f_I T}\boldsymbol{x}_{f_I})^T (H_{f_I}^{f_I T}\boldsymbol{x}_{f_I}),\end{aligned} \tag{58}$$

where f_C and f_I indicate that all features are split to two fields corresponding to contexts and items. It is close to but slightly different from the multi-block convex formulation of FM in (22) by [3, 8]:

$$\hat{y}_{\mathrm{FM}}(\boldsymbol{x}) = \boldsymbol{x}^T W H^T \boldsymbol{x} = [\boldsymbol{x}_{f_C}^T \boldsymbol{x}_{f_I}^T] \begin{bmatrix} W_{f_C} \\ W_{f_I} \end{bmatrix} [H_{f_C}^T H_{f_I}^T] \begin{bmatrix} \boldsymbol{x}_{f_C} \\ \boldsymbol{x}_{f_I} \end{bmatrix}. \tag{59}$$

In (59), W_{f_C} is used in both the interaction between $\boldsymbol{x}_{f_C}$ and $\boldsymbol{x}_{f_C}$, and the interaction between $\boldsymbol{x}_{f_C}$ and $\boldsymbol{x}_{f_I}$. In contrast, in (58), $W_{f_C}^{f_C}$ and $W_{f_C}^{f_I}$ are separately considered. In our comparisons, we consider (58) and denote it as the OCFM model.

We mentioned in Section 1 that models like OCMFSI do not include self-conjunction information. It is important to check the effectiveness of including such information. To begin, a comparison between (6) and (58) shows that OCFM differs from OCMFSI in that OCFM includes self-conjunctions of contexts and items. Thus a comparison between them can reveal how useful self-conjunctions are. Next, because OCFFM has included self-conjunction terms, we remove them from $\hat{Y}_{ij}$ in (46) to have

$$\hat{Y}_{ij} = \sum_{f_1=1}^{F_u} \sum_{f_2=F_u+1}^{F} \boldsymbol{p}_{f_1,f_2}^{i,j\ T} \boldsymbol{q}_{f_2,f_1}^{i,j} = \sum_{f_1=1}^{F_u} \sum_{f_2=F_u+1}^{F} \boldsymbol{p}_{f_1,f_2}^{i\ T} \boldsymbol{q}_{f_2,f_1}^{j}, \tag{60}$$

where we take the properties (42) and (47) into account. Another interpretation of (60) is that we add field information to OCMFSI. Therefore, in our experiments we refer to this setting as OCMFSI+F.

In addition to the above models learnt by the non-subsampled approach, we include several BPR-based models in our experiments. All of them are obtained through solving problems modified from (5). First, by respectively considering $\hat{y}_{\text{FM}}(\boldsymbol{x}^{i,j})$ in (59) and $\hat{y}(\boldsymbol{x}^{i,j})$ in (24) for computing $\hat{Y}_{ij}$, we develop BPR-FM and BPR-FFM. Moreover, as we mentioned in Section 1, besides FFM considering the field information, some recent works improve FM by introducing techniques from deep learning. A typical example is DeepFM [11]. Its output function is

$$\hat{y}_{\text{DeepFM}}(\boldsymbol{x}) = \hat{y}_{\text{FM}}(\boldsymbol{x}) + \hat{y}_{\text{NN}}(\boldsymbol{x}), \tag{61}$$

where $\hat{y}_{\text{NN}}(\boldsymbol{x})$ is a neural network which learns non-linear embeddings of features. By considering $\hat{y}_{\text{DeepFM}}(\boldsymbol{x}^{i,j})$ for computing $\hat{Y}_{ij}$, we develop BPR-DeepFM for the comparison in our experiments.

The connections of these models compared in this work has been summarized in Table 1.

6 EXPERIMENTS

In this section, we first provide the details of the experimental settings. Then, we compare the OCFFM model with some existing one-class models discussed in Section 5.

6.1 Data Sets

We conduct experiments on the following three large-scale data sets.

- **Outbrain**[2]: This set is a binary (clicked or not) classification problem from the "Outbrain Click Prediction" competition held at Kaggle. We convert it to a one-class set by only extracting the contexts with positive labels. To save the evaluation time, we select the advertisements which have been clicked greater than 200 times as the positive entries.
- **KDD-2012**[3]: This is another advertisement data set from KDD-Cup 2012. To have a one-class set, we select the advertisements with more than 50 clicks as the positive entries. Because context information is not pre-defined, we group (User.id, Query.id, Query.depth) as the context of positive instances.
- **KKBOX**[4]: This is a recommendation set from the "KKBox's Music Recommendation Challenge." The goal is to predict whether a user will like a new song or not. We only take the positive samples and select the songs with repetitive listening events more than 5 times. Note that we add a history feature, which contains the recent 50 unique songs repetitivly listened by the user, as part of the context information.

[2]Outbrain Data Set: https://www.kaggle.com/c/outbrain-click-prediction
[3]KDD-2012 Data Set: https://www.kaggle.com/c/kddcup2012-track2
[4]KKBOX Data Set: https://www.kaggle.com/c/kkbox-music-recommendation-challenge

Table 3. Context and item features considered for each data set. The features used as the identities in OCMF are bold-faced. All features are categorical and processed with one-hot encoding. The superscript is the field information assigned to each feature.

Data set	Context	Item
Outbrain	**User.id**[1], User.platform[1], User.location[1], Page.id[2], Page.source[2], Page.publisher[2]	**Ad.id**[3], Ad.campaign[3], Ad.advertiser[3], Ad.source[4], Ad.publisher[4], Ad.page[4]
KDD-2012	**User.id**[1], Query.id[1], Context.depth[1]	**Ad.id**[2], Ad.advertiser[2], Ad.page[2], Ad.keywords[3], Ad.title[4], Ad.description[5]
KKBOX	**User.id**[1], User.location[1], User.gender[1], User.history[2], Source.screen[3], Source.tab[3], Source.type[3]	**Song.id**[4], Song.composer[4], Song.artist[4], Song.lyricist[4], Song.genre[5], Song.language[6]

The feature sets and the corresponding field information of each data set are listed in Table 3. Because we focus on comparing models, the detailed feature engineering and field assignment are out of the scope for this paper.

To select proper parameters, we split the whole data set into three independent parts, the training set (`Tr`), the validation set (`Va`) and the test set (`Te`). We choose the best parameters by training `Tr` and validating `Va`. We then train a new model with the selected parameters on the data set `TrVa`, which combines `Tr` and `Va`, and report the final score of evaluating `Te`. For the **Outbrain** and the **KKBOX** data sets, we sort all contexts by time to select `Tr`, `Va`, and `Te` in ratios of 80%, 10%, and 10%, respectively. For **KDD-2012**, as the time information is implicit, we randomly split contexts by the same ratios.

6.2 Evaluation Criteria

We consider Precision and nDCG on top-K ranked items (P@K and nDCG@K) as evaluation criteria. For an item list $\boldsymbol{z}_i$ sorted according to predicted values of context i, P@K is defined as

$$\text{P@K} \equiv \frac{1}{K} \sum_{j \in \text{top}_K(\boldsymbol{z}_i)} \mathbb{I}(j \in (\Omega^+_{\texttt{Te}})_i) \times 100\%,$$

and nDCG@K is defined as

$$\text{DCG@K} \equiv \sum_{j \in \text{top}_K(\boldsymbol{z}_i)} \frac{\mathbb{I}(j \in (\Omega^+_{\texttt{Te}})_i)}{\log(j+1)},$$

$$\text{nDCG@K} \equiv \frac{\text{DCG@K}}{\sum_{j=1}^{\min(K, |(\Omega^+_{\texttt{Te}})_i|)} \frac{1}{\log(j+1)}} \times 100\%,$$

where $(\Omega^+_{\texttt{Te}})_i$ is the observed positive entries of context i in `Te` and $\mathbb{I}$ is the indicator function.

For our experiments, we conduct grid searches by training multiple models on `Tr` with all the combinations of parameters in Table 5, and choose the combination achieving the best P@5 score evaluated on `Va`. We then train a new model with the selected parameters on the data set `TrVa`, which combines `Tr` and `Va`, and report the final score of evaluating `Te`. The parameters obtained by the selection procedure are shown in supplementary Table I.

Table 4. Data statistics: m and $\hat{m}$ are the number of contexts in the TrVa and Te, respectively. n is the number of items. D is the total number of features in all fields. $|\Omega^+_{\mathsf{TrVa}}|$ and $|\Omega^+_{\mathsf{Te}}|$ indicate numbers of observed entries in training and test sets among $m \times n$ (and $\hat{m} \times n$) pairs, respectively. The column ρ is the ratio of cold-start users in Te.

Data Set	m	$\hat{m}$	n	D	$\lvert\Omega^+_{\mathsf{TrVa}}\rvert$	$\lvert\Omega^+_{\mathsf{Te}}\rvert$	ρ
Outbrain	13M	1.4M	10,947	24,916	13M	1.4M	90.4%
KDD-2012	3.6M	0.40M	13,151	1.5M	3.8M	0.43M	64.3%
KKBOX	3.1M	0.34M	48,443	133,226	3.1M	0.34M	7.9%

Table 5. Range of parameters used in a grid search procedure: ω is the cost weight for the negatives, λ is the regularization coefficient, λ_{BPR} is the regularization coefficient in the SG's weight decay fashion for BPR-based models, k is the specified latent dimension, and T is the number of iterations to stop the training procedure.

Parameter	Range
λ	$\{1, 4, 16, 64, 256\}$
λ_{BPR}	$\{10^{-3}, 10^{-5}, 10^{-7}\}$
ω	$\{2^{-1}, 2^{-2}, \cdots, 2^{-11}\}$
k	$\{16, 32, 64\}$
T	$\{1, 2, \cdots, 100\}$

6.3 Considered Models

We consider the following models in our experiments:

- Popular: The most popular items in the training set are recommended to every user without personalization.
- OCMF: This is the basic model discussed in Sections 1 and 5, where only two identity features are considered. This model fails to make recommendations on cold-started users, so for these users top popular items are recommended.
- OCMFSI: The model proposed by [1, 33]. See the discussion in Section 1.
- OCFM: see Section 5.
- BPR-FM: The one-class extension of FM by the BPR approach. see Section 5.
- BPR-FFM: The one-class extension of FFM by the BPR approach. see Section 5.
- BPR-DeepFM: The one-class extension of DeepFM by the BPR approach; see Section 5.
- OCMFSI+F: see Section 5.
- OCFFM: the model proposed in Section 3.

6.4 Comparison on Test Performance

The results evaluated on the test set are shown in Table 6 and Figure 1.

From Table 6, we first observe that Popular is worse than others. The personalized model OCMF, though not considering context or item features, is generally better than Popular. Note that for **Outbrain** and **KDD-2012** data sets, from Table 4, the test set includes many new users not seen in training. OCMF recommends only popular items rather than personalized recommendations for such cold-start situations. Therefore, if we consider only users that have appeared in the training set, the performance gap is even bigger. However, **KKBOX** is an exception because OCMF performs only similar to Popular. We plan to investigate this set in detail to know why such results occur.

Table 6. P@K (%) and nDCG@K (%) by different models. The best model is bold-faced.

Models	P@5	P@10	P@20	P@40	n@5	n@10	n@20	n@40
Popular	0.24	0.22	0.20	0.17	0.79	1.09	1.53	2.13
OCMF	0.31	0.27	0.24	0.20	0.98	1.36	1.87	2.53
OCMFSI	3.34	2.44	1.70	1.11	11.2	13.7	16.1	18.2
OCFM	3.39	2.46	1.71	1.11	11.3	13.8	16.2	18.3
BPR-FM	3.25	2.44	1.76	1.19	10.9	13.6	16.3	18.9
BPR-FFM	3.51	2.64	1.86	**1.25**	11.8	14.6	17.4	20.0
BPR-DeepFM	3.31	2.50	1.78	1.19	11.0	13.7	16.4	18.8
OCMFSI+F	**3.84**	**2.75**	**1.87**	1.20	**13.0**	**15.7**	18.2	**20.4**
OCFFM	3.83	**2.75**	**1.87**	1.20	**13.0**	**15.7**	**18.2**	**20.4**

(a) Outbrain

Models	P@5	P@10	P@20	P@40	n@5	n@10	n@20	n@40
Popular	1.48	1.10	0.78	0.54	4.91	6.03	7.14	8.28
OCMF	2.99	2.00	1.29	0.81	10.2	11.7	13.1	14.3
OCMFSI	10.2	5.88	3.24	1.74	37.8	40.1	41.5	42.5
OCFM	10.8	6.06	3.30	1.77	41.3	43.3	44.6	45.5
BPR-FM	4.77	3.67	2.44	1.52	15.1	19.0	21.9	24.2
BPR-FFM	7.17	4.66	2.89	1.72	24.0	27.2	29.9	32.0
BPR-DeepFM	8.22	5.29	3.16	1.79	27.4	31.0	33.5	35.1
OCMFSI+F	12.7	7.01	3.72	1.95	49.3	51.2	52.3	53.0
OCFFM	**13.3**	**7.32**	**3.88**	**2.02**	**50.4**	**52.4**	**53.4**	**54.0**

(b) KDD-2012

Models	P@5	P@10	P@20	P@40	n@5	n@10	n@20	n@40
Popular	0.15	0.13	0.13	0.12	0.44	0.64	0.94	1.38
OCMF	0.15	0.14	0.13	0.12	0.44	0.64	0.93	1.38
OCMFSI	0.31	0.31	0.31	0.29	0.89	1.40	2.17	3.23
OCFM	0.38	0.38	0.37	0.32	1.10	1.71	2.59	3.68
BPR-FM	0.35	0.35	0.33	0.28	1.03	1.59	2.35	3.29
BPR-FFM	0.34	0.32	0.29	0.26	1.06	1.53	2.20	3.12
BPR-DeepFM	0.41	0.40	0.36	0.30	1.12	1.81	2.61	3.60
OCMFSI+F	0.72	0.62	0.51	**0.39**	2.26	3.10	4.12	5.21
OCFFM	**0.78**	**0.63**	**0.52**	**0.39**	**2.57**	**3.34**	**4.35**	**5.42**

(c) KKBOX

For factorization-based one-class models, we observed that OCMF is inferior to others. An important reason is that for new users in the test set, OCMF reduces to be the same as Popular. Further, for users already appeared in the training set, the side information improves the predictability. An example is **KKBOX**, where few users are new.

For models with side information, by comparing OCFM with BPR-FM, and OCFFM with BPR-FFM, we observe that models learnt with the non-subsampled setting can have better performance than those learnt with the subsampled setting. This finding is similar to [32] where OCMF performs better than other approaches involving negative subsampling. By comparing OCMFSI with OCMFSI+F,

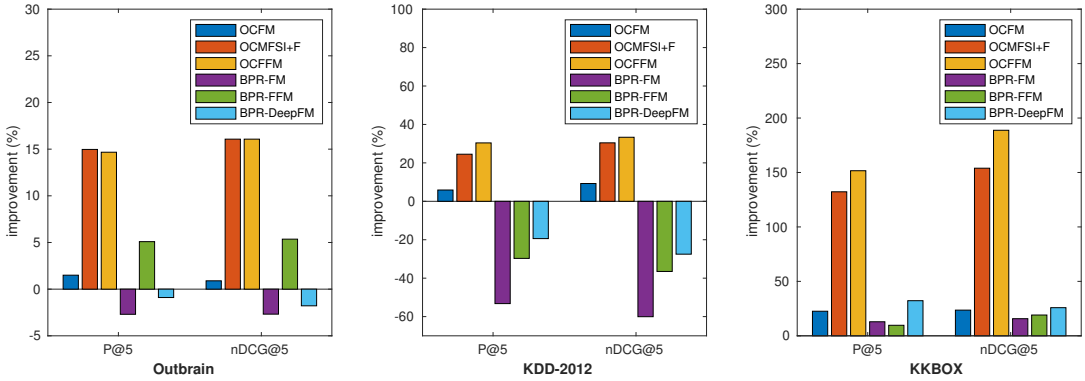


Fig. 1. Comparison on different one-class models with side information. Y-axis is the improvement over the OCMFSI model in percentage.

OCFM with OCFFM, and BPR-FM with BPR-FFM we can find that for most cases, the models considering the field information are superior. By comparing BPR-DeepFM with BPR-FM, the better performance of BPR-DeepFM confirms the better modeling ability of DeepFM over FM can be extended into the one-class scenarios. We observe that OCFFM (or OCMFSI+F) is significantly better for almost all data sets. This result confirms our conjecture that with the non-subsampled setting and the field information, OCFFM can be a competitive approach for recommender systems with implicit feedback. Finally, OCFM and OCFFM are respectively slightly better than OCMFSI and OCMFSI+F, where the differences come from whether we add self-conjunctions of context (or item) features. The results show that adding these conjunctions may be helpful to learn the biases of contexts and items.

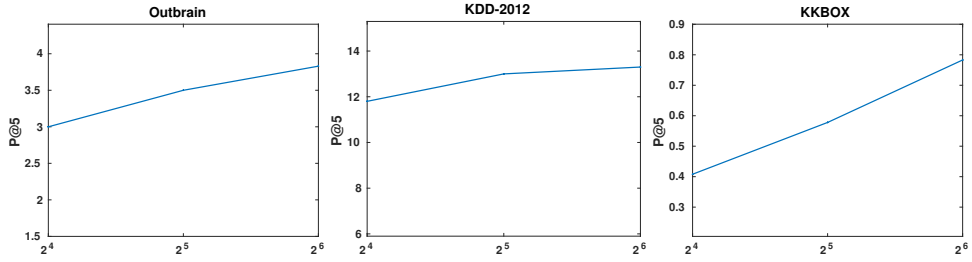
6.5 Impact of Parameters

We conduct experiments to investigate the impact of the latent dimension k, the regularization coefficient λ, the cost weight for the negatives ω, and the number of the stopping iteration T on OCFFM. Based on the selected parameters from a validation procedure discussed in Section 6.1 (see supplementary Table I), we orderly change each parameter's value and leave the others unchanged. The resulting P@5 scores are given in Figure 2.

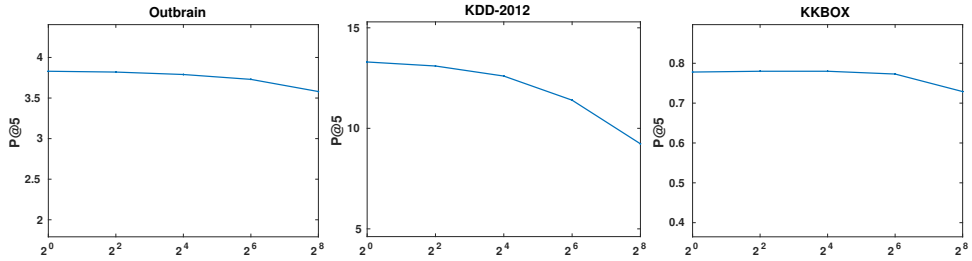
Regarding the parameter k, results in Figure 2a show that with the increasing of k, the P@5 scores gradually improves due to the stronger modeling capability. However, from (13) and (57), because the model size and the complexity of the block CD algorithm linearly and quadratically depend on k, respectively, a tradeoff between performance and the cost on memory and training time is required in the practice use of OCFFM.

In figure 2b, we present the relationship between λ and P@5 scores. If λ is too large, OCFFM is not able to achieve a good performance. On the contrary, with a small λ, OCFFM gets better results, but it may lead to a slight overfitting effect on some data sets.

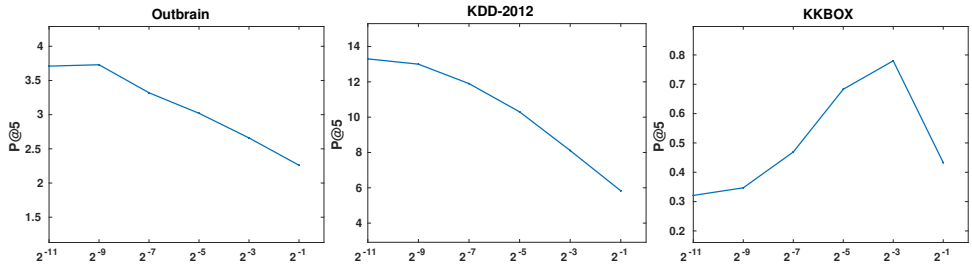
For the parameter ω, Figure 2c shows that results are the most sensitive to this parameter. Specifically, we have the similar observation to [32] that ω should not be too small nor too large. A too small ω causes models to wrongly underfit negative unobserved pairs, while a too large ω causes model to wrongly fit positive unobserved pairs. The theoretical results in [14] suggest that under the setting of $\omega = \frac{\gamma}{2-\gamma}$ an error bound can be proved, where γ is the percentage of positive observed pairs. When γ is close to one, because most positive pairs have been observed, the unobserved pairs are most negative, so ω should be larger. In contrast, a small γ indicates



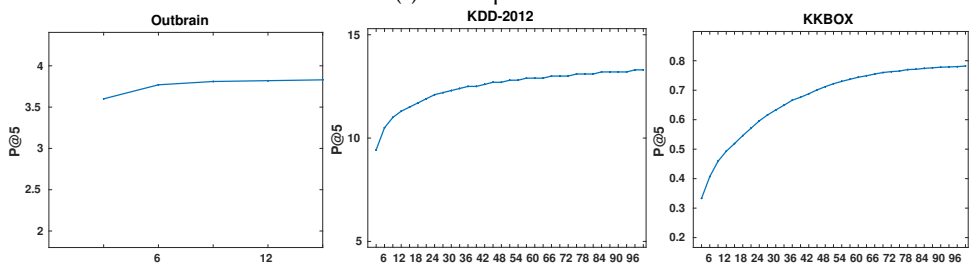
(a) The impact of k



(b) The impact of λ



(c) The impact of ω



(d) The impact of T

Fig. 2. The impact of the latent dimension k, the regularization coefficient λ, the cost weight for the negatives ω, and the number of the stopping iteration T on OCFFM

many unobserved pairs are indeed positive, so ω should be smaller in this case. This result can be used for explaining why **KKBOX** requires a larger ω. Specifically, different from the situation of **Outbrain** and **KDD-2012** where ads are recommended items, the task of **KKBOX** is to recommend songs. Because compared to ads, songs have more chances to be known by users in other ways (e.g., recommendations from friends), most positive pairs of **KKBOX** may have been included in

Ω^+, while unobserved pairs are more likely to be known but disliked by users. Thus a larger ω is more suitable for **KKBOX**.

From the relationship between T and P@5 scores shown in Figure 2d, we observe that P@5 gradually increases along iterations T. After some iterations the results are close toe the optimal one. Therefore, we can easily find a suitable iteration T to stop the training procedure.

7 CONCLUSION AND FUTURE WORKS

From Section 1, the motivation to propose OCFFM can be summarized in the following table.

General regression	Recommendation (implicit feedbacks)
FM	OCMFSI/OCFM
↓	↓
FFM	OCFFM

We saw the potential of OCFFM because of

(1) the superiority of FFM over FM on some classification/regression problems, and
(2) the strong connection on the output functions of FM and OCMFSI/OCFM.

However, to have a practically viable OCFFM model is not easy. In particular, to avoid the $O(mn)$ complexity, we develop an efficient implementation of block coordinate descent algorithms. The experiments conducted on real-world data sets show that our proposed OCFFM outperforms existing one-class models.

For future works, the algorithm in Section 3 may be extended to more general loss such as that in [33]:

$$\ell_{ij}(y, \hat{y}) = \begin{cases} \text{any differentiable function convex in } \hat{y} & (i,j) \in \Omega^+, \\ (16) & (i,j) \notin \Omega^+. \end{cases}$$

For the OCFFM implementation, some further investigation is needed. For example, the order of blocks in the block CD algorithm may affect the convergence speed. Finally, we have deployed OCFFM to our online production.

REFERENCES

[1] Immanuel Bayer, Xiangnan He, Bhargav Kanagal, and Steffen Rendle. 2017. A generic coordinate descent framework for learning from implicit feedback. In *Proceedings of the 26th International Conference on World Wide Web*. 1341–1350.

[2] Guy Blanc and Steffen Rendle. 2018. Adaptive sampled softmax with kernel based sampling. In *International Conference on Machine Learning*. 590–599.

[3] Mathieu Blondel, Masakazu Ishihata, Akinori Fujino, and Naonori Ueda. 2016. Polynomial Networks and Factorization Machines: new Insights and Efficient Training Algorithms. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*.

[4] Wanling Cai, Jiongbin Zheng, Weike Pan, Jing Lin, Lin Li, Li Chen, Xiaogang Peng, and Zhong Ming. 2019. Neighborhood-enhanced transfer learning for one-class collaborative filtering. *Neurocomputing* 341 (2019), 80–87.

[5] 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*. 2400–2410.

[6] Chong Chen, Min Zhang, Yongfeng Zhang, Yiqun Liu, and Shaoping Ma. 2020. Efficient Neural Matrix Factorization without Sampling for Recommendation. *ACM Transactions on Information Systems* 38, 2 (2020), 1–28.

[7] Jin Chen, Defu Lian, and Kai Zheng. 2019. Improving One-Class Collaborative Filtering via Ranking-Based Implicit Regularizer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 37–44.

[8] Wei-Sheng Chin, Bo-Wen Yuan, Meng-Yuan Yang, and Chih-Jen Lin. 2018. An Efficient Alternating Newton Method for Learning Factorization Machines. *ACM Transactions on Intelligent Systems and Technology* 9, 6 (2018), 72:1–72:31. https://www.csie.ntu.edu.tw/~cjlin/papers/fm/scalefm.pdf

[9] Zeno Gantner, Lucas Drumond, Christoph Freudenthaler, Steffen Rendle, and Lars Schmidt-Thieme. 2010. Learning attribute-to-feature mappings for cold-start recommendations. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*. 176–185.

[10] Gene H. Golub and Charles. F. Van Loan. 2012. *Matrix Computations* (fourth ed.). The Johns Hopkins University Press.

[11] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*. 1725–1731.

[12] Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. 2016. Fast matrix factorization for online recommendation with implicit feedback. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. 549–558.

[13] Magnus Rudolph Hestenes and Eduard Stiefel. 1952. Methods of Conjugate Gradients for Solving Linear Systems. *J. Res. Nat. Bur. Standards* 49, 1 (1952), 409–436.

[14] Cho-Jui Hsieh, Nagarajan Natarajan, and Inderjit Dhillon. 2015. PU Learning for Matrix Completion. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*. 2445–2453.

[15] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*. 263–272.

[16] Yuchin Juan, Yong Zhuang, Wei-Sheng Chin, and Chih-Jen Lin. 2016. Field-aware factorization machines for CTR Prediction. In *Proceedings of the ACM Recommender Systems Conference (RecSys)*. http://www.csie.ntu.edu.tw/~cjlin/papers/ffm.pdf

[17] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *Proceedings of International Conference on Learning Representations (ICLR)*.

[18] Walid Krichene, Nicolas Mayoraz, Steffen Rendle, Li Zhang, Xinyang Yi, Lichan Hong, Ed Chi, and John Anderson. 2018. Efficient Training on Very Large Corpora via Gramian Estimation. In *International Conference on Learning Representations*.

[19] Mu-Chu Lee, Wei-Lin Chiang, and Chih-Jen Lin. 2015. Fast Matrix-vector Multiplications for Large-scale Logistic Regression on Shared-memory Systems. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*. http://www.csie.ntu.edu.tw/~cjlin/papers/multicore_liblinear_icdm.pdf

[20] Defu Lian, Qi Liu, and Enhong Chen. 2020. Personalized Ranking with Importance Sampling. In *Proceedings of The Web Conference*. 1093–1103.

[21] Chih-Jen Lin, Ruby C. Weng, and S. Sathiya Keerthi. 2007. Trust region Newton method for large-scale logistic regression. In *Proceedings of the 24th International Conference on Machine Learning (ICML)*. Software available at http://www.csie.ntu.edu.tw/~cjlin/liblinear.

[22] Hongzhi Liu, Zhonghai Wu, and Xing Zhang. 2018. CPLR: collaborative pairwise learning to rank for personalized recommendation. *Knowledge-Based Systems* 148 (2018), 31–40.

[23] Rong Pan and Martin Scholz. 2009. Mind the Gaps: Weighting the Unknown in Large-scale One-class Collaborative Filtering. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. 667–676.

[24] Rong Pan, Yunhong Zhou, Bin Cao, Nathan N Liu, Rajan Lukose, Martin Scholz, and Qiang Yang. 2008. One-class collaborative filtering. In *IEEE International Conference on Data Mining (ICDM)*. 502–511.

[25] Ulrich Paquet and Noam Koenigstein. 2013. One-class collaborative filtering with random graphs. In *Proceedings of the 22nd international conference on World Wide Web*. 999–1008.

[26] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.). 8024–8035.

[27] Huihuai Qiu, Yun Liu, Guibing Guo, Zhu Sun, Jie Zhang, and Hai Thanh Nguyen. 2018. BPRH: Bayesian personalized ranking for heterogeneous implicit feedback. *Information Sciences* 453 (2018), 80–98.

[28] Steffen Rendle. 2010. Factorization machines. In *Proceedings of IEEE International Conference on Data Mining (ICDM)*. 995–1000.

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

[30] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence (UAI)*. 452–461.

[31] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research* 15, 1 (2014), 1929–1958.

[32] Hsiang-Fu Yu, Mikhail Bilenko, and Chih-Jen Lin. 2017. Selection of Negative Samples for One-class Matrix Factorization. In *Proceedings of SIAM International Conference on Data Mining (SDM)*. http://www.csie.ntu.edu.tw/~cjlin/papers/one-class-mf/biased-mf-sdm-with-supp.pdf

[33] Hsiang-Fu Yu, Hsin-Yuan Huang, Inderjit S. Dihillon, and Chih-Jen Lin. 2017. A Unified Algorithm for One-class Structured Matrix Factorization with Side Information. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI)*. http://www.csie.ntu.edu.tw/~cjlin/papers/ocmf-side/biased-leml-aaai-with-supp.pdf

[34] 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. In *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence (UAI)*.

[35] Guo-Xun Yuan, Chia-Hua Ho, and Chih-Jen Lin. 2012. An Improved GLMNET for L1-regularized Logistic Regression. *Journal of Machine Learning Research* 13 (2012), 1999–2030. http://www.csie.ntu.edu.tw/~cjlin/papers/l1_glmnet/long-glmnet.pdf

[36] Weinan Zhang, Tianqi Chen, Jun Wang, and Yong Yu. 2013. Optimizing top-n collaborative filtering via dynamic negative item sampling. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*. 785–788.

[37] Tong Zhao, Julian McAuley, and Irwin King. 2014. Leveraging social connections to improve personalized ranking for collaborative filtering. In *Proceedings of the 23rd ACM International Conference on Information and Knowledge Management*. 261–270.

[38] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. 1059–1068.

[39] Wang Zhou, Jianping Li, Yongluan Zhou, and Muhammad Hammad Memon. 2019. Bayesian pairwise learning to rank via one-class collaborative filtering. *Neurocomputing* 367 (2019), 176–187.

A PROOF OF THEOREM 3.1

Proof. From the definition of $\tilde{\boldsymbol{x}}$, we have

$$\tilde{X} = \left[\boldsymbol{q}^1 \otimes \boldsymbol{x}^1, \cdots, \boldsymbol{q}^L \otimes \boldsymbol{x}^L\right]^T \in \mathcal{R}^{L \times Dk}$$

and therefore

$$\begin{aligned}\tilde{X}^T \tilde{C}\tilde{\boldsymbol{y}} &= \sum_{l=1}^{L} (C_l \tilde{y}_l)(\boldsymbol{q}^l \otimes \boldsymbol{x}^l) = \sum_{l=1}^{L} (C_l \tilde{y}_l)\,\mathrm{vec}(\boldsymbol{x}^l \boldsymbol{q}^{l^T}) = \mathrm{vec}\left(\sum_{l=1}^{L} \boldsymbol{x}^l C_l \tilde{y}_l \boldsymbol{q}^{l^T}\right) \\ &= \mathrm{vec}(X^T \,\mathrm{diag}(\tilde{C}\tilde{\boldsymbol{y}})Q).\end{aligned} \tag{A.1}$$

For $\tilde{X}^T \tilde{C}\tilde{X}\,\mathrm{vec}(V)$, we let

$$\boldsymbol{z} = \tilde{X}\,\mathrm{vec}(V). \tag{A.2}$$

Then (A.1) implies that

$$\tilde{X}\tilde{C}\tilde{X}\,\mathrm{vec}(V) = \mathrm{vec}(X^T \,\mathrm{diag}(\tilde{C}\boldsymbol{z})Q).$$

For $\boldsymbol{z}$, the definition in (A.2) indicates that

$$z_l = (\boldsymbol{q}^l \otimes \boldsymbol{x}^l)^T \,\mathrm{vec}(V) = \boldsymbol{x}^{l^T} V \boldsymbol{q}^l,$$

which is equivalent to

$$\boldsymbol{z} = ((XV) \odot Q)\mathbf{1}_{k \times 1}.$$

□

B DETAILS OF BLOCK CD METHOD FOR OCFFM

B.1 The Computation of the Right-hand Side of the Linear System

Among the three situations in (43), in Section 4 we have discussed the first one of $1 \le f_1, f_2 \le F_u$. Thus we check the remaining two cases here.

To begin, from the definition of Y_{ij} and C_{ij} in (3) and (21) respectively,

$$C_{ij}(\hat{Y}_{ij} - Y_{ij}) = \begin{cases} \hat{Y}_{ij} - 1 & \text{if } (i,j) \in \Omega^+, \\ \omega(\hat{Y}_{ij} - r) & \text{otherwise.} \end{cases} \tag{B.1}$$

Then we have

$$\begin{aligned}\sum_{i=1}^{m}\sum_{j=1}^{n} \boldsymbol{x}_{f_1}^{i,j}(C_{ij}(\hat{Y}_{ij} - Y_{ij}))\boldsymbol{q}_{f_2,f_1}^{i,j\,T} &= \sum_{(i,j)\in\Omega^+} \boldsymbol{x}_{f_1}^{i,j}((1-\omega)\hat{Y}_{ij} - 1 + r\omega)\boldsymbol{q}_{f_2,f_1}^{i,j\,T} \\ &\quad + \omega \sum_{i=1}^{m}\sum_{j=1}^{n} \boldsymbol{x}_{f_1}^{i,j}(\hat{Y}_{ij} - r)\boldsymbol{q}_{f_2,f_1}^{i,j\,T}.\end{aligned} \tag{B.2}$$

The first term involves $|\Omega^+|$ summations so with $|\Omega^+| \ll mn$ the bottleneck is on the second term, which can be further written as

$$\sum_{i=1}^{m}\sum_{j=1}^{n}(a_i + b_j + \sum_{\alpha=1}^{F_u}\sum_{\beta=F_u+1}^{F} \boldsymbol{p}_{\alpha,\beta}^{i\,T}\boldsymbol{q}_{\beta,\alpha}^{j} - r)\boldsymbol{x}_{f_1}^{i,j}\boldsymbol{q}_{f_2,f_1}^{i,j\,T}, \tag{B.3}$$

by excluding ω and using (46) and (48).

B.1.1 The Situation of $f_1, f_2 > F_u$.
From (41), (42) and $f_1, f_2 > F_u$, we have

$$\boldsymbol{x}_{f_1}^{i,j} = \boldsymbol{v}_{f_1}^{j} \text{ and } \boldsymbol{q}_{f_2,f_1}^{i,j} = \boldsymbol{q}_{f_2,f_1}^{j}. \tag{B.4}$$

We can compute (B.3) as

$$\begin{aligned}
&\sum_{i=1}^{m}\sum_{j=1}^{n}(a_i + b_j + \sum_{\alpha=1}^{F_u}\sum_{\beta=F_u+1}^{F} \boldsymbol{p}_{\alpha,\beta}^{i\,T}\boldsymbol{q}_{\beta,\alpha}^{j} - r)\boldsymbol{v}_{f_1}^{j}\boldsymbol{q}_{f_2,f_1}^{j\,T} \\
&= \sum_{j=1}^{n}(mb_j - mr + \sum_{i=1}^{m} a_i + \sum_{\alpha=1}^{F_u}\sum_{\beta=F_u+1}^{F}(\sum_{i=1}^{m}\boldsymbol{p}_{\alpha,\beta}^{i\,T})\boldsymbol{q}_{\beta,\alpha}^{j})\boldsymbol{v}_{f_1}^{j}\boldsymbol{q}_{f_2,f_1}^{j\,T} \\
&= \sum_{j=1}^{n}(mb_j - mr + \sum_{i=1}^{m} a_i + e_j)\boldsymbol{v}_{f_1}^{j}\boldsymbol{q}_{f_2,f_1}^{j\,T},
\end{aligned} \tag{B.5}$$

where

$$e_j = \sum_{\alpha=1}^{F_u}\sum_{\beta=F_u+1}^{F}(\sum_{i=1}^{m}\boldsymbol{p}_{\alpha,\beta}^{i\,T})\boldsymbol{q}_{\beta,\alpha}^{j}. \tag{B.6}$$

A crucial observation is that

$$\sum_{i=1}^{m} a_i \text{ and } \sum_{i=1}^{m}\boldsymbol{p}_{\alpha,\beta}^{i}$$

are now independent of j.

We now put (B.5) back to (B.2). Let

$$V_{f_1} = \left[\boldsymbol{v}_{f_1}^{1}, \cdots, \boldsymbol{v}_{f_1}^{n}\right]^T. \tag{B.7}$$

Then (B.2) can be computed by

$$V_{f_1}^T \operatorname{diag}(\boldsymbol{z}) Q_{f_2}^{f_1}, \tag{B.8}$$

where

$$z_j = \sum_{i\in\bar{\Omega}_j^+}((1-\omega)\hat{Y}_{ij} - 1 + \omega r)) + \omega(mb_j - mr + \sum_{i=1}^{m} a_i + e_j), \tag{B.9}$$

and $\bar{\Omega}_j^+$ is the set including the observed contexts of the jth item. Note that $Q_{f_2}^{f_1}$ now contains n columns. Our earlier definition of $Q_{f_2}^{f_1}$ in (52) is for the situation of $f_2 \le F_u$ only. A complete definition should be

$$Q_{f_2}^{f_1} = \begin{cases} \left[\boldsymbol{q}_{f_2,f_1}^{1}, \cdots, \boldsymbol{q}_{f_2,f_1}^{m}\right]^T & f_2 \le F_u, \\ \left[\boldsymbol{q}_{f_2,f_1}^{1}, \cdots, \boldsymbol{q}_{f_2,f_1}^{n}\right]^T & f_2 > F_u. \end{cases} \tag{B.10}$$

Similarly, we define $P_{f_2}^{f_1}$ for the use in subsequent places.

$$P_{f_1}^{f_2} = \begin{cases} \left[\boldsymbol{p}_{f_1,f_2}^{1}, \cdots, \boldsymbol{p}_{f_1,f_2}^{m}\right]^T & f_1 \le F_u, \\ \left[\boldsymbol{p}_{f_1,f_2}^{1}, \cdots, \boldsymbol{p}_{f_1,f_2}^{n}\right]^T & f_1 > F_u. \end{cases} \tag{B.11}$$

B.1.2 The Situation of $f_1 \le F_u, f_2 > F_u$. From (41), (42), and $f_1 \le F_u$, $f_2 > F_u$, we have

$$\boldsymbol{x}_{f_1}^{i,j} = \boldsymbol{u}_{f_1}^{i} \text{ and } \boldsymbol{q}_{f_2,f_1}^{i,j} = \boldsymbol{q}_{f_2,f_1}^{j}. \tag{B.12}$$

We can split the computation of (B.3) to several parts. First,

$$\begin{aligned}\sum_{i=1}^{m}\sum_{j=1}^{n} \boldsymbol{u}_{f_1}^{i}(a_i - r)\boldsymbol{q}_{f_2,f_1}^{j\,T} &= \sum_{i=1}^{m} \boldsymbol{u}_{f_1}^{i}(a_i - r)(\sum_{j=1}^{n} \boldsymbol{q}_{f_2,f_1}^{j\,T}) \\ &= U_{f_1}^{T}(\boldsymbol{a} - r\mathbf{1}_{m\times 1})\boldsymbol{q}_{\mathrm{o}}^{T},\end{aligned} \tag{B.13}$$

where U_{f_1} is defined in (52) and

$$\boldsymbol{q}_{\mathrm{o}} = Q_{f_2}^{f_1 T}\mathbf{1}_{n\times 1}. \tag{B.14}$$

Next,

$$\sum_{i=1}^{m}\sum_{j=1}^{n} \boldsymbol{u}_{f_1}^{i} b_j \boldsymbol{q}_{f_2,f_1}^{j\,T} = (\sum_{i=1}^{m} \boldsymbol{u}_{f_1}^{i})(\sum_{j=1}^{n} b_j \boldsymbol{q}_{f_2,f_1}^{j\,T}) = (U_{f_1}^{T}\mathbf{1}_{m\times 1})\boldsymbol{q}_{\mathrm{b}}^{T}, \tag{B.15}$$

where

$$\boldsymbol{q}_{\mathrm{b}} = Q_{f_2}^{f_1 T}\boldsymbol{b}. \tag{B.16}$$

Finally,

$$\begin{aligned}&\sum_{i=1}^{m}\sum_{j=1}^{n} \boldsymbol{u}_{f_1}^{i}(\sum_{\alpha=1}^{F_u}\sum_{\beta=F_u+1}^{F} \boldsymbol{p}_{\alpha,\beta}^{i\,T}\boldsymbol{q}_{\beta,\alpha}^{j})\boldsymbol{q}_{f_2,f_1}^{j\,T} \\ =&\sum_{\alpha=1}^{F_u}\sum_{\beta=F_u+1}^{F}(\sum_{i=1}^{m}\boldsymbol{u}_{f_1}^{i}\boldsymbol{p}_{\alpha,\beta}^{i\,T})(\sum_{j=1}^{n}\boldsymbol{q}_{\beta,\alpha}^{j}\boldsymbol{q}_{f_2,f_1}^{j\,T}) \\ =&\sum_{\alpha=1}^{F_u}\sum_{\beta=F_u+1}^{F}(U_{f_1}^{T}P_{\alpha}^{\beta})(Q_{\beta}^{\alpha T}Q_{f_2}^{f_1}).\end{aligned} \tag{B.17}$$

Various ways are possible to calculate (B.17). From (B.10)-(B.11), we give the size of matrices involved.

$$U_{f_1}: m \times D_{f_1}, \quad P_{\alpha}^{\beta}: m \times k, \quad Q_{\beta}^{\alpha}: n \times k, \quad Q_{f_2}^{f_1}: n \times k.$$

Except k, other values D_{f_1}, m and n can possibly be large. Therefore, a setting such as

$$U_{f_1}^{T}(\sum_{\alpha=1}^{F_u}\sum_{\beta=F_u+1}^{F}(P_{\alpha}^{\beta}Q_{\beta}^{\alpha T}))Q_{f_2}^{f_1}, \tag{B.18}$$

is not feasible because the matrix in the middle requires $O(mn)$ space. Feasible settings with the corresponding computational complexity are

$$\sum_{\alpha=1}^{F_u}\sum_{\beta=F_u+1}^{F}(U_{f_1}^{T}P_{\alpha}^{\beta})(Q_{\beta}^{\alpha T}Q_{f_2}^{f_1}) \tag{B.19}$$

with

$$O((\mathrm{nnz}(U_{f_1})k + (n + D_{f_1})k^2)F_u F_v) \tag{B.20}$$

cost, and

$$U_{f_1}^{T}(\sum_{\alpha=1}^{F_u}\sum_{\beta=F_u+1}^{F}P_{\alpha}^{\beta}(Q_{\beta}^{\alpha T}Q_{f_2}^{f_1})), \tag{B.21}$$

with

$$O(\mathrm{nnz}(U_{f_1})k + F_u F_v(n + m)k^2) \tag{B.22}$$

cost. For the space requirement, the approach in (B.19) needs $O(D_{f_1}k)$ space, because $Q_\beta^{\alpha T}Q_{f_2}^{f_1}$ is a small $k \times k$ matrix. For (B.21), we need $O((D_{f_1}+m)k)$. Both are affordable because storing U_{f_1},P_α^β and Q_β^α may already take more than that. Note that in this analysis we assume that U_{f_1} is a sparse matrix and its product with any matrix results in a dense matrix. A comparison between (B.20) and (B.22) shows that if F_u and F_v are not small, then the term

$$\text{nnz}(U_{f_1})kF_uF_v$$

tends to cause that the first approach in (B.19) is more expensive. Therefore, in the subsequent description of our algorithms, we consider (B.21) for the implementation.

Clearly, each of the $O(mn)$ summations is nicely split to the product between one $O(m)$ term and one $O(n)$ term. Therefore, similar to the first situation, the $O(mn)$ cost is avoided.

Using (B.12) and (B.13-B.17), the following matrix form of (B.2) is what we intend to obtain.

$$U_{f_1}^T\begin{bmatrix}\boldsymbol{z}_1^T\\ \vdots \\ \boldsymbol{z}_m^T\end{bmatrix} + \omega((U_{f_1}^T(\boldsymbol{a}-r\mathbf{1}_{m\times 1}))\boldsymbol{q}_{\text{o}}^T + (U_{f_1}^T\mathbf{1}_{m\times 1})\boldsymbol{q}_{\text{b}}^T + U_{f_1}^T T),$$

where $\boldsymbol{q}_o$ and $\boldsymbol{q}_b$ are defined in (B.14) and (B.16) respectively,

$$\boldsymbol{z}_i = \sum_{j\in\Omega_i^+}((1-\omega)\hat{Y}_{ij} - 1 + r\omega)\boldsymbol{q}_{f_2,f_1}^j, \text{ and } T = \sum_{\alpha=1}^{F_u}\sum_{\beta=F_u+1}^{F} P_\alpha^\beta(Q_\beta^{\alpha T}Q_{f_2}^{f_1}).$$

B.2 The Computation of Matrix-vector Products

From Theorem 3.1 and

$$\boldsymbol{x}^l \to \boldsymbol{x}_{f_1}^{i,j}, \boldsymbol{q}^l \to \boldsymbol{q}_{f_2,f_1}^{i,j}, C_l \to C_{ij}, z_l \to z_{ij},$$

what we need to calculate is

$$\sum_{i=1}^{m}\sum_{j=1}^{n}\boldsymbol{x}_{f_1}^{i,j}(C_{ij}z_{ij})\boldsymbol{q}_{f_2,f_1}^{i,j\ T}, \tag{B.23}$$

where

$$z_{ij} = \boldsymbol{x}_{f_1}^{i,j\,T}V\boldsymbol{q}_{f_2,f_1}^{i,j}.$$

From the definition of C_{ij} in (21),

$$\begin{aligned}(\text{B.23}) &= \sum_{(i,j)\in\Omega^+}\boldsymbol{x}_{f_1}^{i,j}z_{ij}\boldsymbol{q}_{f_2,f_1}^{i,j\ T} + \sum_{(i,j)\notin\Omega^+}\boldsymbol{x}_{f_1}^{i,j}\omega z_{ij}\boldsymbol{q}_{f_2,f_1}^{i,j\ T}\\ &= (1-\omega)\sum_{(i,j)\in\Omega^+}\boldsymbol{x}_{f_1}^{i,j}z_{ij}\boldsymbol{q}_{f_2,f_1}^{i,j\ T} + \sum_{i=1}^{m}\sum_{j=1}^{n}\boldsymbol{x}_{f_1}^{i,j}z_{ij}\boldsymbol{q}_{f_2,f_1}^{i,j\ T}.\end{aligned} \tag{B.24}$$

For the first situation, $f_1, f_2 \le F_u$, (41) and (42) imply

$$\boldsymbol{x}_{f_1}^{i,j} \to \boldsymbol{u}_{f_1}^i \text{ and } \boldsymbol{q}_{f_2,f_1}^{i,j} \to \boldsymbol{q}_{f_2,f_1}^i.$$

Thus

$$z_{ij} = \boldsymbol{x}_{f_1}^{i,j\,T}V\boldsymbol{q}_{f_2,f_1}^{i,j} = \boldsymbol{u}_{f_1}^{i\ T}V\boldsymbol{q}_{f_2,f_1}^i \tag{B.25}$$

is independent of j. With $z_{ij} \to z_i$, we have

$$(\text{B.24}) = \sum_{i=1}^{m}\boldsymbol{u}_{f_1}^i((1-\omega)\sum_{j\in\Omega_i^+}z_i + \sum_{j=1}^{n}z_i)\boldsymbol{q}_{f_2,f_1}^i = U_{f_1}^T\,\text{diag}(\boldsymbol{z})Q_{f_2}^{f_1}, \tag{B.26}$$

where

$$z_i = ((1-\omega)|\Omega_i^+| + \omega n)\boldsymbol{u}_{f_1}^{i\,T} V \boldsymbol{q}_{f_2,f_1}^i,$$

or equivalently, the matrix form of $\boldsymbol{z}$ in (56). Clearly, from (B.26), the $O(mn)$ cost is avoided.

For the second situation, it is the same as (B.26) though variables must be changed; see Section B.3.

Next we consider the third situation when $f_1 \le F_u$, $f_2 > F_u$ and have (B.12). Then

$$z_{ij} = \boldsymbol{u}_{f_1}^{i\,T} V \boldsymbol{q}_{f_2,f_1}^j.$$

To get the second term in (B.24) we have

$$\begin{aligned}\sum_{i=1}^m \sum_{j=1}^n \boldsymbol{u}_{f_1}^i (\boldsymbol{u}_{f_1}^{i\,T} V \boldsymbol{q}_{f_2,f_1}^j)\boldsymbol{q}_{f_2,f_1}^{j\,T} &= (\sum_{i=1}^m \boldsymbol{u}_{f_1}^i \boldsymbol{u}_{f_1}^{i\,T}) V (\sum_{j=1}^n \boldsymbol{q}_{f_2,f_1}^j \boldsymbol{q}_{f_2,f_1}^{j\,T}) \\ &= (U_{f_1}^T U_{f_1}) V (Q_{f_2}^{f_1\,T} Q_{f_2}^{f_1}).\end{aligned}$$

Finally,

$$(\text{B.24}) = (1-\omega)\sum_{i=1}^m \boldsymbol{u}_{f_1}^i \sum_{j\in\Omega_i^+} (\boldsymbol{u}_{f_1}^{i\,T} V \boldsymbol{q}_{f_2,f_1}^j)\boldsymbol{q}_{f_2,f_1}^{j\,T} + \omega (U_{f_1}^T U_{f_1}) V (Q_{f_2}^{f_1\,T} Q_{f_2}^{f_1}). \tag{B.27}$$

To compute the first term, at each i, we obtain

$$\boldsymbol{u}_{f_1}^{i\,T} V$$

with a complexity of $O(\text{nnz}(\boldsymbol{u}_{f_1})k)$. Then the cost of computing

$$\sum_{j\in\Omega_i^+} ((\boldsymbol{u}_{f_1}^{i\,T} V)\boldsymbol{q}_{f_2,f_1}^j)\boldsymbol{q}_{f_2,f_1}^{j\,T}$$

is

$$O((\text{nnz}(\boldsymbol{u}_{f_1}^i) + |\Omega_i^+|)k).$$

Thus, the first term can be computed in

$$O((\text{nnz}(U_{f_1}) + |\Omega^+|)k). \tag{B.28}$$

To calculate the second term, several ways are possible by different orders of matrix-matrix products. To discuss them, we begin with giving the size of matrices involved.

$$U_{f_1}: m \times D_{f_1}, \quad V: D_{f_1} \times k, \quad Q_{f_2}^{f_1}: n \times k. \tag{B.29}$$

Except k, all other values D_{f_1}, m and n can possibly be very large. Therefore, a setting such as

$$(U_{f_1}^T U_{f_1}) V (Q_{f_2}^{f_1\,T} Q_{f_2}^{f_1})$$

is not feasible because $(U_{f_1}^T U_{f_1})$ requires a huge $O(D_{f_1}^2)$ space. Therefore, the three only possible settings with the corresponding computational complexity are

$$(U_{f_1}^T (U_{f_1} V))(Q_{f_2}^{f_1\,T} Q_{f_2}^{f_1}) \quad \cdots O(\text{nnz}(U_{f_1})k + D_{f_1}k^2 + nk^2) \tag{B.30}$$

$$U_{f_1}^T ((U_{f_1} V)(Q_{f_2}^{f_1\,T} Q_{f_2}^{f_1})) \quad \cdots O(\text{nnz}(U_{f_1})k + mk^2 + nk^2) \tag{B.31}$$

$$U_{f_1}^T (U_{f_1} (V (Q_{f_2}^{f_1\,T} Q_{f_2}^{f_1}))) \quad \cdots O(\text{nnz}(U_{f_1})k + D_{f_1}k^2 + nk^2) \tag{B.32}$$

where all need

$$O((D_{f_1} + m)k)$$

space. We consider to apply (B.31) if $m < D_{f_1}$. Otherwise, either (B.30) or (B.32) can be considered.

B.3 Implementation Details

We summarize the overall procedures of block CD for OCFFM in Algorithm 4. At each iteration of the algorithm, blocks of variables are sequentially updated. The order of blocks in the sequence is an important and complicated issue in the block CD method. We leave a detailed study as a future issue, so here we simply consider a setting according to the three situations in (43). That is, three loops are conducted to go through the following fields.

$$\begin{cases} \text{Steps 18-23} & f_1, f_2 \le F_u, \quad \text{(B.33a)} \\ \text{Steps 26-31} & f_1, f_2 > F_u, \quad \text{(B.33b)} \\ \text{Steps 32-39} & f_1 \le F_u, f_2 > F_u. \quad \text{(B.33c)} \end{cases}$$

Consider details of the first situation. Under the given f_1 and f_2, two sub-problems to update variable blocks $W_{f_1}^{f_2}$ and $H_{f_2}^{f_1}$ are solved. From (24), (41), (B.10) and (B.11), if we replace the following information

$$(U_{f_1}, W_{f_1}^{f_2}, P_{f_1}^{f_2}, Q_{f_2}^{f_1}) \tag{B.34}$$

in the sub-problem of $W_{f_1}^{f_2}$ with

$$(U_{f_2}, H_{f_2}^{f_1}, Q_{f_2}^{f_1}, P_{f_1}^{f_2}),$$

then we obtain the sub-problems of $H_{f_2}^{f_1}$. Thus the same Algorithm 5 (a detailed version of Algorithm 3 explained below) can be used to solve them with different input variables.

We then consider details of the second situation. By comparing (B.8) with (53), the sub-problems of $W_{f_1}^{f_2}$ in (B.33a) and (B.33b) are in the same form if the roles of

$$(m, U_{f_1}, \boldsymbol{d}, \boldsymbol{a}, \bar{b}, i, j, \hat{Y}_{ij}) \text{ and } (n, V_{f_1}, \boldsymbol{e}, \boldsymbol{b}, \bar{a}, j, i, \hat{Y}_{ji}) \tag{B.35}$$

are swapped. Therefore, Algorithm 5 can solve all sub-problems in the loop (B.33b) with different input variables; see how we call Algorithm 5 in line 26-31 of Algorithm 4. We note that the input $\hat{Y}$ is changed to $\hat{Y}^T$ and give the following explanation. Later we will show that among the variables to be maintained after each sub-problem, $\hat{Y}_{ij}, \forall (i,j) \in \Omega^+$ is one of them. With $|\Omega^+| \ll mn$, $\hat{Y}$ is stored as a sparse matrix. In a sub-problem of the second case (i.e., $f_1, f_2 > F_u$), we note that for an item j, contexts $i \in \bar{\Omega}_j^+$ are iterated in (B.9). Specifically, in (B.9) we have the following calculation.

$$\sum_{i \in \bar{\Omega}_j^+} \hat{Y}_{ij}.$$

In contrast, in (54) for a context i a loop over $j \in \Omega_i^+$ is implemented. Because i and j are swapped according to (B.35), if we input $\hat{Y}^T$ into Algorithm 5, then the same loop in Algorithm 5 can be used for sub-problems of either the first or the second situation.

For the third situation, we have shown in Sections B.1 and B.2 that the calculation of the right-hand side of the linear system and the calculation of matrix-vector products are different from those for the first/second situations. Thus we have another Algorithm 6 to give details for solving the sub-problem.

Next we discuss one sub-problem in detail as an example. We consider Algorithm 5 on the sub-problem of the block $W_{f_1}^{f_2}$ in the first loop (B.33a). From (53), (54) and (B.26), besides the input data U_{f_1}, all we need are

$$Q_{f_2}^{f_1}, \boldsymbol{a}, \sum_{j=1}^{n} b_j, \boldsymbol{d}, \text{and } \hat{Y}_{ij}, \forall (i,j) \in \Omega^+. \tag{B.36}$$

We can calculate all of them, but a careful check shows that these values may be maintained and passed from one sub-problem to another. For example, the vector $\boldsymbol{a}$ defined in (49) is changed after $P_{f_1}^{f_2}$ is updated. If

$$\Delta P = [(\Delta \boldsymbol{p})^1, \cdots, (\Delta \boldsymbol{p})^m]^T \tag{B.37}$$

is the amount of change on $P_{f_1}^{f_2}$, then easily the vector $\boldsymbol{a}$ can be maintained by

$$\boldsymbol{a} + \Delta \boldsymbol{a}, \tag{B.38}$$

where from (49)

$$(\Delta a)_i = (\Delta \boldsymbol{p})^{i^T} \boldsymbol{q}_{f_2,f_1}^i, i = 1, ..., m,$$

or equivalently,

$$\Delta \boldsymbol{a} = (\Delta P \odot Q_{f_2}^{f_1}) \mathbf{1}_{k\times 1}.$$

Similarly, from (46),$\hat{Y}_{ij}$ needed in (B.36) can be maintained by

$$\hat{Y}_{ij} \leftarrow \hat{Y}_{ij} + (\Delta a)_i, \forall (i,j) \in \Omega^+.$$

Next we discuss the vector $\boldsymbol{b}$. In the loop (B.33a) we are considering, because of $1 \le f_1, f_2, \le F_u$, (49) implies that $\boldsymbol{b}$ is a constant. Therefore, we may calculate it before the loop. However, in Algorithm 4 we see that in the second loop (B.33b), the role of $\boldsymbol{b}$ is like $\boldsymbol{a}$ in the first loop. Thus in the second loop $\boldsymbol{b}$ has been maintained by a way similar to (B.38). Therefore, $\boldsymbol{b}$ is readily available before the loop (B.33a), so all we need is to calculate the sum of $\boldsymbol{b}$ at Step 8 as an input value for solving the sub-problem. Another value $Q_{f_2}^{f_1}$ needed in (B.36) is in a situation similar to $\boldsymbol{b}$. It is a constant in the current loop (B.33a), but can be maintained in the loop (B.33b). Conversely, $P_{f_1}^{f_2}$ is not currently used, but should be updated for the use in the loop (B.33b). To this end, we need

$$P_{f_1}^{f_2} \leftarrow P_{f_1}^{f_2} + \Delta P.$$

The above discussion explains why in the end of Algorithm 5, $P_{f_1}^{f_2}, \boldsymbol{a}$ and $\hat{Y}_{ij}, \forall (i,j) \in \Omega^+$ are variables to be updated.

Regarding the vector $\boldsymbol{d}$ defined in (51), we note that it is also a constant in the loop (B.33a). Therefore, we may maintain it by a similar way to those for $\boldsymbol{a}$ and $\boldsymbol{b}$. In particular, from (51), $\boldsymbol{d}$ involves $\alpha \in \{1, \cdots, F_u\}$ and $\beta \in \{F_u + 1, \cdots, F\}$, so it should be maintained in the third loop (B.33c). However, unlike the situation in the first and the second loops, where $\boldsymbol{a}$ and $\boldsymbol{b}$ must be maintained and used, the third loop does not need $\boldsymbol{d}$ at all. Further, even if we maintain $\boldsymbol{d}$ there, because of the for loop, the cost is still proportional to $F_u F_v$ and the complexity is not reduced. Therefore, we directly calculate it at Step 16 before the loop (B.33a). On the other hand for the loop (B.33b), a similar vector $\boldsymbol{e}$ is considered; see Step 24.

We briefly discuss if the cost of (B.21) can be reduced by maintaining some variables. An issue is that the value in (B.21) depends on the current f_1 and f_2. Because the setting (B.18) is not feasible for the space concern, we can only consider (B.19) or (B.21), in which (α, β) and (f_1, f_2) are mingled together. Thus we do not see an easy setting to further reduce the complexity.

B.4 Computational Complexity

We analyze the overall complexity of Algorithm 4. In solving each sub-problem, from Algorithm 5 and Algorithm 6 we can see that each matrix-vector product costs similarly to that of calculating the right-hand side of the linear system (33).[5] Further, each matrix-vector product in Algorithm 6

[5] Assume that in the calculation of (B.21), F_u and F_v are not large.

Table I. Parameters used in Table 6. For **KDD-2012**, due to its large number of features D as shown in Table 4, the setting of $k = 32$ and $k = 64$ cause the out of GPU memory issue for BPR-based approaches.

Datasets	Outbrain	KDD-2012	KKBOX
Parameters / Models	$\{\lambda, \lambda_{\text{BPR}}, \omega, k, T\}$		
OCMF	$\{16, -, 2^{-9}, 64, 100\}$	$\{1, -, 2^{-11}, 64, 100\}$	$\{4, -, 2^{-2}, 32, 30\}$
OCMFSI	$\{1, -, 2^{-10}, 64, 100\}$	$\{1, -, 2^{-11}, 64, 100\}$	$\{16, -, 2^{-11}, 64, 100\}$
OCFM	$\{1, -, 2^{-10}, 64, 100\}$	$\{1, -, 2^{-10}, 64, 100\}$	$\{4, -, 2^{-9}, 64, 100\}$
BPR-FM	$\{-, 10^{-7}, -, 64, 40\}$	$\{-, 10^{-7}, -, 16, 50\}$	$\{-, 10^{-7}, -, 64, 30\}$
BPR-FFM	$\{-, 10^{-7}, -, 64, 40\}$	$\{-, 10^{-7}, -, 16, 40\}$	$\{-, 10^{-7}, -, 64, 30\}$
BPR-DeepFM	$\{-, 10^{-7}, -, 64, 20\}$	$\{-, 10^{-7}, -, 16, 50\}$	$\{-, 10^{-7}, -, 64, 30\}$
OCMFSI+F	$\{1, -, 2^{-10}, 64, 30\}$	$\{1, -, 2^{-10}, 64, 100\}$	$\{4, -, 2^{-5}, 64, 90\}$
OCFFM	$\{1, -, 2^{-10}, 64, 15\}$	$\{1, -, 2^{-11}, 64, 100\}$	$\{1, -, 2^{-3}, 64, 100\}$

is more expensive than that in Algorithm 5. Assume that in calculating the second term of (B.27), the setting of (B.32) is taken. Then from (B.28) and (B.32) the overall cost is

$$\begin{aligned} &O\left((\text{nnz}(U_f \text{ or } V_f) + |\Omega^+|)k + (D_f + m + n)k^2\right) \\ &\quad \times \text{ avg. \# CG steps per sub-problem } \times F(F+1), \end{aligned} \tag{B.39}$$

where D_f is average the number of context/item features of a field, and $\text{nnz}(U_f)$ or $\text{nnz}(V_f)$ is the average number of non-zeros. Clearly, the sparsity of Ω^+ decides if $|\Omega^+|k$ is the dominant term or not.

C EXPERIMENTAL DETAILS

C.1 Implementation

As discussed in Section 5, all non-subsampled approach based models to be compared are special cases of OCFFM. Therefore, we only implement OCFFM, and use it to train OCMF, OCMFSI, OCFM, OCMFSI+F and OCFFM. We tried the best to have efficient implementations for the optimization algorithm proposed in Section 3. The dense operations (e.g., matrix-matrix and matrix-vector products) are implemented using Intel® Math Kernel Library (MKL). Besides, we parallelize the Hession-vector products with an algorithm proposed by [19].

For the BPR-based models including BPR-FFM and BPR-DeepFM, we implement them by PyTorch [26], where we apply Adam [17] as the optimizer with initial learning rates 0.001 for **Outbrain**, **KKBOX** and 0.0005 for **KDD-2012**. For the neural network part of BPR-DeepFM, we apply a three-layer network, where each layer contains 400 hidden units and is equipped with dropout [31] rate 0.5 to prevent overfitting.

For all models we randomly select the initial parameters uniformly from $[-\frac{0.1}{\sqrt{k}}, \frac{0.1}{\sqrt{k}}]$. For the tolerance in the CG stopping condition (39), we set $\xi = 0.3$.

C.2 Parameters

The parameters used in our experiments are shown in Table I.

Algorithm 4 Solving the OCFFM problem via block coordinate descent method.

Given an initial $\mathcal{W}$ and $\mathcal{H}$
for $f_1 \leftarrow \{1, \cdots, F_u\}$ **do**
  for $f_2 \leftarrow \{f_1, \cdots, F_u\}$ **do**
    $P_{f_1}^{f_2} \leftarrow X_{f_1} W_{f_1}^{f_2}$, $Q_{f_2}^{f_1} \leftarrow X_{f_2} H_{f_2}^{f_1}$ $\cdots O((m+n)k)$
  end for
end for
$\boldsymbol{a} \leftarrow \sum_{f_1=1}^{F_u} \sum_{f_2=f_1}^{F_u} \left(P_{f_1}^{f_2} \odot Q_{f_2}^{f_1}\right) \mathbf{1}^{k\times 1}$ $\cdots O(F_u^2 mk)$
$\boldsymbol{b} \leftarrow \sum_{f_1=F_u+1}^{F} \sum_{f_2=f_1}^{F} \left(P_{f_1}^{f_2} \odot Q_{f_2}^{f_1}\right) \mathbf{1}^{k\times 1}$ $\cdots O(F_v^2 nk)$
$\hat{Y}_{ij} = a_i + b_j - 1, \forall (i,j) \in \Omega^+$, $\cdots O(|\Omega^+|)$
for $f_1 \leftarrow \{1, \cdots, F_u\}$ **do**, $\cdots O(F_u F_v |\Omega^+| k)$
  for $f_2 \leftarrow \{F_u + 1, \cdots, F_u\}$ **do**
    $\hat{Y}_{ij} \leftarrow \hat{Y}_{ij} + {\boldsymbol{p}_{f_1,f_2}^i}^T \boldsymbol{q}_{f_2,f_1}^j, \forall (i,j) \in \Omega^+$
  end for
end for
while Stopping condition is not satisfied **do**
  $\boldsymbol{d} \leftarrow \sum_{\alpha=1}^{F_u} \sum_{\beta=F_u+1}^{F} (P_\alpha^\beta (Q_\beta^{\alpha T} \mathbf{1}_{n\times 1}))$
    $\cdots O(F_u F_v (m+n)k)$
  $\bar{b} = \sum_{j=1}^{n} b_j$ $\cdots O(n)$
  for $f_1 \leftarrow \{1, \cdots, F_u\}$ **do**
    for $f_2 \leftarrow \{f_1, \cdots, F_u\}$ **do**
      Alg. 5 $(U_{f_1}, W_{f_1}^{f_2}, P_{f_1}^{f_2}, Q_{f_2}^{f_1}, \boldsymbol{d}, \boldsymbol{a}, \bar{b}, \hat{Y})$
      Alg. 5 $(U_{f_2}, H_{f_2}^{f_1}, Q_{f_2}^{f_1}, P_{f_1}^{f_2}, \boldsymbol{d}, \boldsymbol{a}, \bar{b}, \hat{Y})$
    end for
  end for
  $\boldsymbol{e} \leftarrow \sum_{\alpha=1}^{F_u} \sum_{\beta=F_u+1}^{F} (Q_\alpha^\beta (P_\beta^{\alpha T} \mathbf{1}_{m\times 1}))$
    $\cdots O(F_u F_v (m+n)k)$
  $\bar{a} = \sum_{i=1}^{m} a_i$ $\cdots O(m)$
  for $f_1 \leftarrow \{F_u + 1, \cdots, F\}$ **do**
    for $f_2 \leftarrow \{f_1, \cdots, F\}$ **do**
      Alg. 5 $(V_{f_1}, W_{f_1}^{f_2}, P_{f_1}^{f_2}, Q_{f_2}^{f_1}, \boldsymbol{e}, \boldsymbol{b}, \bar{a}, \hat{Y}^T)$
      Alg. 5 $(V_{f_2}, H_{f_2}^{f_1}, Q_{f_2}^{f_1}, P_{f_1}^{f_2}, \boldsymbol{e}, \boldsymbol{b}, \bar{a}, \hat{Y}^T)$
    end for
  end for
  for $f_1 \leftarrow \{1, \cdots, F_u\}$ **do**
    for $f_2 \leftarrow \{F_u + 1, \cdots, F\}$ **do**
      $T \leftarrow \sum_{\alpha=1}^{F_u} \sum_{\beta=F_u+1}^{F} (P_\alpha^\beta (Q_\beta^{\alpha T} Q_{f_2}^{f_1}))$
        $\cdots O(F_u F_v (m+n)k^2)$
      Alg. 6 $(U_{f_1}, W_{f_1}^{f_2}, P_{f_1}^{f_2}, Q_{f_2}^{f_1}, \boldsymbol{a}, \boldsymbol{b}, \hat{Y}, T)$
      $T \leftarrow \sum_{\alpha=1}^{F_u} \sum_{\beta=F_u+1}^{F} (Q_\beta^\alpha (P_\alpha^{\beta T} P_{f_1}^{f_2}))$
        $\cdots O(F_u F_v (m+n)k^2)$
      Alg. 6 $(V_{f_2}, H_{f_2}^{f_1}, Q_{f_2}^{f_1}, P_{f_1}^{f_2}, \boldsymbol{b}, \boldsymbol{a}, \hat{Y}^T, T)$
    end for
  end for
end while

Algorithm 5 Conjugate gradient procedure for approximately solving the sub-problems when f_1, $f_2 \le F_u$ or f_1, $f_2 > F_u$

Given input $U_{f_1} \in \mathcal{R}^{m \times D_{f_1}}$, $W_{f_1}^{f_2} \in \mathcal{R}^{D_{f_1} \times k}$, $P_{f_1}^{f_2} \in \mathcal{R}^{m \times k}$, $Q_{f_2}^{f_1} \in \mathcal{R}^{n \times k}$, $\boldsymbol{d} \in \mathcal{R}^m, \boldsymbol{a} \in \mathcal{R}^m, \bar{b} = \sum_{j=1}^{n} b_j$, and $\hat{Y}_{ij}, \forall (i,j) \in \Omega^+$
Let
    $U \leftarrow U_{f_1}, W \leftarrow W_{f_1}^{f_2}$
    $P \leftarrow P_{f_1}^{f_2}, Q \leftarrow Q_{f_2}^{f_1}, D \leftarrow D_{f_1}$
// Calculate the right-hand side of the linear system
for $i \leftarrow \{1, \cdots, m\}$ **do** $\cdots O(m + |\Omega^+|)$
    $z_i \leftarrow \omega(na_i + \bar{b} + d_i - nr)$
    $z_i \leftarrow z_i + \sum_{j \in \Omega_i^+}((1-\omega)\hat{Y}_{ij} - \omega(1-r))$
end for
$G \leftarrow U^T \operatorname{diag}(\boldsymbol{z}) Q + \lambda W$ $\cdots O(\operatorname{nnz}(U)k + Dk)$
Let $R \leftarrow -G$, $V \leftarrow R$ and $\gamma \leftarrow \|R\|_{\mathrm{F}}^2$
while True **do**
    if $\|R\|_{\mathrm{F}} \le \xi \|G\|_{\mathrm{F}}$ **then**
        break
    end if
    //Compute and store the matrix-vector product to Z
    $\boldsymbol{z} = ((UV) \odot Q)\, \mathbf{1}_{k \times 1}$ $\cdots O(mk + \operatorname{nnz}(U)k)$
    for $i \leftarrow \{1, \cdots, m\}$ **do** $\cdots O(m)$
        $z_i \leftarrow z_i((1-\omega)|\Omega_i^+| + \omega n)$
    end for
    $Z \leftarrow U^T \operatorname{diag}(\boldsymbol{z}) Q$ $\cdots O(\operatorname{nnz}(U)k)$
    Step 10-16 in Algorithm 2
end while
//Update cached variables
$W \leftarrow W + S$ $\cdots O(Dk)$
$\Delta P \leftarrow US$ $\cdots O(\operatorname{nnz}(U)k)$
$P \leftarrow P + \Delta P$ $\cdots O(mk)$
$\Delta \boldsymbol{a} \leftarrow (\Delta P \odot Q)\mathbf{1}_{k \times 1}$ $\cdots O(mk)$
$\boldsymbol{a} \leftarrow \boldsymbol{a} + \Delta \boldsymbol{a}$ $\cdots O(m)$
$\hat{Y}_{ij} \leftarrow \hat{Y}_{ij} + (\Delta a)_i, \forall (i,j) \in \Omega^+$ $\cdots O(|\Omega^+|)$

Algorithm 6 Conjugate gradient procedure for approximately solving the sub-problems when $f_1 \leq F_u$, $f_2 > F_u$

Given input $U_{f_1} \in \mathcal{R}^{m \times D_{f_1}}$, $W_{f_1}^{f_2} \in \mathcal{R}^{D_{f_1} \times k}$, $P_{f_1}^{f_2} \in \mathcal{R}^{m \times k}$, $Q_{f_2}^{f_1} \in \mathcal{R}^{n \times k}$,$\boldsymbol{d} \in \mathcal{R}^m, \boldsymbol{a} \in \mathcal{R}^m, \boldsymbol{b} \in \mathcal{R}^n$, $\hat{Y}_{ij}, \forall (i,j) \in \Omega^+$, and $T \in \mathcal{R}^{m \times k}$

Let

$$U = [\boldsymbol{u}^1, \cdots, \boldsymbol{u}^m]^T \leftarrow U_{f_1}, W \leftarrow W_{f_1}^{f_2}$$

$$P \leftarrow P_{f_1}^{f_2}, Q = [\boldsymbol{q}^1, \cdots, \boldsymbol{q}^n]^T \leftarrow Q_{f_2}^{f_1}, D \leftarrow D_{f_1}$$

// Calculate the right-hand side of the linear system

$\boldsymbol{q}_{\text{o}} \leftarrow Q^T \mathbf{1}_{n \times 1}$ $\cdots O(nk)$

$\boldsymbol{q}_{\text{b}} \leftarrow Q^T \boldsymbol{b}$ $\cdots O(nk)$

$\bar{\boldsymbol{z}} \leftarrow \boldsymbol{a} - r\mathbf{1}_{m \times 1}$ $\cdots O(m)$

for $i \leftarrow \{1, \cdots, m\}$ **do** $\cdots O((|\Omega^+| + \text{nnz}(U))k)$

  $\boldsymbol{z} \leftarrow \sum_{j \in \Omega_i^+} ((1-\omega)\hat{Y}_{ij} - \omega(1-r))\boldsymbol{q}^j$

  $G \leftarrow G + \boldsymbol{u}_i \boldsymbol{z}^T$

end for

$G \leftarrow G + \omega((U^T \bar{\boldsymbol{z}})\boldsymbol{q}_{\text{o}}^T + (U^T \mathbf{1}_{m \times 1})\boldsymbol{q}_{\text{b}}^T + U^T T) + \lambda W$ $\cdots O((\text{nnz}(U) + D)k)$

$B \leftarrow Q^T Q$ $\cdots O(nk^2)$

Let $R \leftarrow -G$, $V \leftarrow R$ and $\gamma \leftarrow \|R\|_{\text{F}}^2$

while True **do**

  if $\|R\|_{\text{F}} \leq \xi \|G\|_{\text{F}}$ **then**

    break

  end if

  //Compute and store the matrix-vector product to Z

  $Z \leftarrow \lambda V + \omega U^T (U(VB))$ $\cdots O(\text{nnz}(U)k + Dk^2)$

  for $i \leftarrow \{1, \cdots, m\}$ **do** $\cdots O((|\Omega^+| + \text{nnz}(U))k)$

    $\boldsymbol{\tau} \leftarrow \mathbf{0}_{k \times 1}, \boldsymbol{\gamma} \leftarrow \boldsymbol{u}^{iT} V$

    for $j \in \Omega_i^+$ **do**

      $v \leftarrow \boldsymbol{\gamma} \boldsymbol{q}^j$

      $\boldsymbol{\tau} \leftarrow \boldsymbol{\tau} + v\boldsymbol{q}^j$

    end for

    $Z \leftarrow Z + (1-\omega)\boldsymbol{u}^i \boldsymbol{\tau}^T$

  end for

  Step 10-16 in Algorithm 2

end while

//Update cached variables

$W \leftarrow W + S$ $\cdots O(Dk)$

$\Delta P \leftarrow US$, where ΔP is defined in (B.37) $\cdots O(\text{nnz}(U)k)$

$P \leftarrow P + \Delta P$ $\cdots O(mk)$

$\hat{Y}_{ij} \leftarrow \hat{Y}_{ij} + (\Delta \boldsymbol{p})^{iT} \boldsymbol{q}^j, \forall (i,j) \in \Omega^+$ $\cdots O(|\Omega^+|k)$