RaFM: Rank-Aware Factorization Machines

Xiaoshuang Chen [1] **Yin Zheng** [2] **Jiaxing Wang** [3] **Wenye Ma** [2] **Junzhou Huang** [2]

Abstract

Fatorization machines (FM) are a popular model class to learn pairwise interactions by a low-rank approximation. Different from existing FM-based approaches which use a fixed rank for all features, this paper proposes a Rank-Aware FM (RaFM) model which adopts pairwise interactions from embeddings with different ranks. The proposed model achieves a better performance on real-world datasets where different features have significantly varying frequencies of occurrences. Moreover, we prove that the RaFM model can be stored, evaluated, and trained as efficiently as one single FM, and under some reasonable conditions it can be even significantly more efficient than FM. RaFM improves the performance of FMs in both regression tasks and classification tasks while incurring less computational burden, therefore also has attractive potential in industrial applications.

1. Introduction

Factorization machines (FM) (Rendle, 2010; 2012) are one of the most popular models to leverage the interactions between features. It models nested feature interactions via a factorized parametrization, and achieves success in many sparse predictive areas, such as recommender systems and click-through rate predictions (Juan et al., 2016). Recently there are many follow-up researches, such as high-order FMs (Blondel et al., 2016), convex FMs (Blondel et al., 2015), neural-network-based FMs (He & Chua, 2017; Guo et al., 2017), locally linear FMs (Liu et al., 2017), etc.

Most of the existing approaches allocate a fixed rank of embedding vectors to each feature (Liu et al., 2017; Guo et al., 2017; Zheng et al., 2016b;a; Du et al., 2018; Lauly et al., 2017; Jiang et al., 2017). However, the frequencies

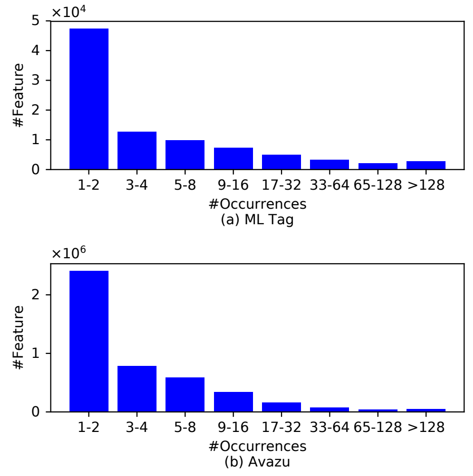

Figure 1. Occurrences of different features in (a) ML Tag; (b) Avazu. Example: in ML Tag, more than 40,000 features only occur once or twice, while less than 5,000 features occur more than 128 times.

of different features occurring in real-world datasets vary a lot. Fig. 1 shows the numbers of occurrences of different features in two public datasets, i.e. MovieLens[1] Tag and Avazu[2] respectively. It is shown that a large number of features scarsely occur in the dataset, while only a few features frequently occur. Fixing the rank of FMs may lead to overfitting problems for features with few occurrences, and underfitting problems for features with many occurrences.

There are also researches allocating with different ranks to each feature(Li et al., 2016; 2017; Juan et al., 2016). However, training and storing multiple embeddings lead to unaffordable computational burden, and the large number of parameters may even aggravate the overfitting problem (Li et al., 2017). To this end, this paper proposes a Rank-Aware FM (RaFM) model, i.e., to maintain embedding vectors with different ranks for each feature, which makes it possible to compute each pairwise interaction via proper ranks of embedding vectors. A key contribution of this paper is to prove that **although RaFM maintains multiple embeddings with different ranks for each feature, it can be stored, evaluated, and trained as efficiently as, or even more efficiently than a single FM with a fixed rank**.

[1]Department of Electrical Engineering, Tsinghua University, Beijing, China [2]Tencent AI Lab, Shenzhen, China [3]Institute of Automation, Chinese Academy of Sciences, and University of Chinese Academy of Sciences, Beijing, China. Correspondence to: Yin Zheng <yinzheng@tencent.com>.

Proceedings of the 36th International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019. Copyright 2019 by the author(s).

[1]https://grouplens.org/datasets/movielens
[2]https://www.kaggle.com/c/avazu-ctr-prediction/data

Specifically, we analyze the time and space complexity of RaFM, and show that there are many inactive factors which need not be stored and evaluated. Therefore, both the computational time and the storage can be significantly reduced, and the entire computational burden can be even smaller than that of a single FM under some reasonable conditions. Furthermore, we provide an algorithm to train all embedding vectors of RaFM in one concise model where the inactive factors will not be stored and trained, and then prove the convergence rate and the performance bound of the training algorithm. Thus, the computational burden of the training process can also be effectively reduced. Experiments show the effectiveness of RaFM in both public datasets and datasets of industrial applications. The proposed RaFM model improves the performance of FMs while incurring a comparable or even less computational burden, therefore also has attractive potential in industrial applications.

2. Problem Formulation

2.1. General Interaction Form of FMs

Here we provide a general interaction form of the FM model:

$$\hat{y} = \sum_{i,j\in\mathcal{F}, i<j} \langle \mathcal{V}_i, \mathcal{V}_j \rangle x_i x_j + \sum_i w_i x_i + bias \tag{1}$$

where $\mathcal{F}$ is the index set of features, x_i is the value of the i-th feature, w_i is the weight of the i-th feature in the linear part, and $bias$ is the bias of the model. $\mathcal{V}_i$ is the embedding of the i-th feature, describing the characteristics of this feature when interacting with other features. $\langle \mathcal{V}_i, \mathcal{V}_j \rangle$ represents the bi-interaction between the i-th feature and the j-th feature.

Clearly, the computation of the bi-interaction $\langle \mathcal{V}_i, \mathcal{V}_j \rangle$ highly depends on the formulations of the embedding term $\mathcal{V}_i$. In the original FM model, $\mathcal{V}_i$ is a vector of a fixed rank, and $\langle \mathcal{V}_i, \mathcal{V}_j \rangle$ is the dot product. Specifically, assume we have m FMs, the rank of the k-th of which is D_k, then we have

$$\langle \mathcal{V}_i, \mathcal{V}_j \rangle_{FM_k} = \boldsymbol{v}_i^{(k)} \cdot \boldsymbol{v}_j^{(k)} = \sum_{f=1}^{D_k} v_{i,f}^{(k)} v_{j,f}^{(k)} \tag{2}$$

where $\boldsymbol{v}_i^{(k)}$ is the embedding with dimension D_k, and $v_{i,f}^{(k)}$ is the f-th coordinate of $\boldsymbol{v}_i^{(k)}$. We assume the m FMs are ordered such that $D_1 \le D_2 \le \cdots \le D_m$. Let $D = D_m$. We use $\boldsymbol{v}^{(k)}$ to represent the parameter set $\{\boldsymbol{v}_i^{(k)} : i \in \mathcal{F}\}$.

According to Fig. 1, the frequencies of different features occurring in real-world datasets vary a lot, hence fixed rank FMs may have unsatisfying performance on these datasets. Specifically, a certain embedding $\mathcal{V}_i$ will be trained only when x_i are both nonzero. Therefore in the k-th FM, the embedding $\mathcal{V}_i = \boldsymbol{v}_i^{(k)}$ will suffer from underfitting if the i-th feature frequently occurs, and from overfitting if the feature scarsely occurs, as shown in Fig. 2(a)(b).

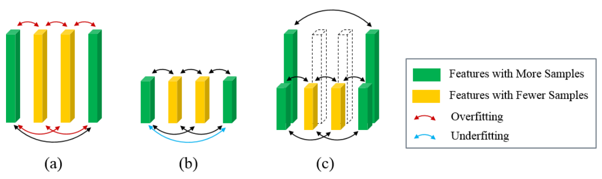

Figure 2. The intuition of RaFM. (a) FM with a high dimension; (b) FM with a low dimension; (c) RaFM

2.2. RaFM: Rank-Aware Factorization Machines

From the abovementioned discussions, it is beneficial to allocate different ranks of embedding vectors to different features. In this paper, we allocate multiple embedding vectors to each feature in RaFM, i.e.,

$$\mathcal{V}_i = \left\{ \boldsymbol{v}_i^{(1)}, \boldsymbol{v}_i^{(2)}, \cdots, \boldsymbol{v}_i^{(k_i)} \right\} \tag{3}$$

where the rank of the k-th vector is D_k. The value k_i means that the maximum rank of the embedding vectors is D_{k_i}. k_i can be interpreted as: if $k > k_i$, the rank of $\boldsymbol{v}_i^{(k)}$ will be too large compared to the occurrences of the i-th feature, which will lead to the overfitting problem. k_i can be chosen according to the number of occurrences of the i-th feature, and in this paper, we regard k_i as hyperparameters. Although some boosting methods (Li et al., 2018) might be used to select appropriate k_i, we leave it for future work.

Here we emphasize that the multiple embeddings in $\mathcal{V}_i$ are not independent in the RaFM model. Intuitively speaking, $\boldsymbol{v}_i^{(1)}$ can be regarded as a "projection" of $\boldsymbol{v}_i^{(2)}$ from the D_2-dimensional space to the D_1-dimensional space. We will return to this point in Section 4, which provides an efficient training problem maintaining this property.

The pairwise interaction can be computed as

$$\langle \mathcal{V}_i, \mathcal{V}_j \rangle_{RaFM} = \boldsymbol{v}_i^{(k_{ij})} \cdot \boldsymbol{v}_j^{(k_{ij})} \tag{4}$$

where

$$k_{ij} = \min(k_i, k_j) \tag{5}$$

which means the dimensionality to compute the pairwise interaction between the i-th and j-th features depends on the maximum common dimensionality of the embedding vectors of the i-th and the j-th feature. Therefore, the computation of $\langle \mathcal{V}_i, \mathcal{V}_j \rangle$ achieves a good trade-off between overfitting problems and underfitting problems, which is the intuition why RaFM outperforms FM. Fig. 2 also illustrates the idea of RaFM.

A key challenge of RaFM is the computational burden when storing, evaluating, and training RaFM:

1. It is well-known that FM can be evaluated efficiently in $O(D\,|\mathcal{F}|)$. However, the straight computation of RaFM, i.e. (1)(4), is $O(D\,|\mathcal{F}|^2)$, which is not satifying.

2. To train the RaFM, we need to train multiple embedding vectors for a feature, which incurs a large computational burden and storage consumption.

We separate the discussions into two sections: Section 3 shows that RaFM can be stored and evaluated as efficient, or even more efficient than a single FM, whereas Section 4 provides an efficient training algorithm.

3. Model Complexity of RaFM

This section analyzes the space and time complexity of RaFM. Trivial bounds of the space and time complexity are $O(\sum_{k=1}^m D_k |\mathcal{F}|)$ and $O(D |\mathcal{F}|^2)$ respectively. In this section, they will be significantly improved to be comparable or even less than a single FM under reasonable conditions.

3.1. Space Complexity

Before discussing the complexity of RaFM, we introduce a class of feature sets, i.e.

$$\mathcal{F}_k = \{i \in \mathcal{F} : k_i \geq k\}, k = 1, 2, \cdots, m \qquad (6)$$

then we have $\mathcal{F}_1 \supset \mathcal{F}_2 \cdots \supset \mathcal{F}_m$. We have $\mathcal{F} = \mathcal{F}_1$ due to $k_i \geq 1$ by definition.

Although the RaFM in Eq. (3) contains m group of embeddings with different ranks, not all the parameters each group of embeddings will be used. Specifically, if we use $\mathcal{F} - \mathcal{F}_k$ to denote the set difference of $\mathcal{F}$ and $\mathcal{F}_k$, then for $i \in \mathcal{F} - \mathcal{F}_k$ and $l > k$, the factor $\boldsymbol{v}_i^{(l)}$ will never be used according to Eq. (4). These factors are called *inactive factors*. In other words, only $\boldsymbol{v}_i^{(k)}, i \in \mathcal{F}_k$ need to be maintained, which are called *active factors*. Therefore we have

Proposition 1. *The space complexity of parameters in RaFM* (3) *is* $O\left(\sum_{k=1}^m D_k |\mathcal{F}_k|\right)$.

3.2. Time Complexity

To derive the time complexity of RaFM, we introduce two notations, i.e. $\mathcal{A}_{l,k}$ and $\mathcal{B}_{l,k}$. Specifically,

$$\begin{aligned}\mathcal{A}_{l,k} &= \sum_{i,j \in \mathcal{F}_k, i<j} \boldsymbol{v}_i^{(l)} \cdot \boldsymbol{v}_j^{(l)} x_i x_j \\ &= \frac{1}{2}\left(\left\|\sum_{i \in \mathcal{F}_k} \boldsymbol{v}_i^{(l)} x_i\right\|_2^2 - \sum_{i \in \mathcal{F}_k}\left\|\boldsymbol{v}_i^{(l)} x_i\right\|_2^2\right)\end{aligned} \qquad (7)$$

And when $l \leq k$, $\mathcal{B}_{l,k}$ is defined as

$$\mathcal{B}_{l,k} = \sum_{i<j} \boldsymbol{v}_i^{(k_{ij}|_{[l,k]})} \cdot \boldsymbol{v}_j^{(k_{ij}|_{[l,k]})} x_i x_j \qquad (8)$$

where $k_{ij}|_{[l,k]} = \max[l, \min(k, k_{ij})]$.

$\mathcal{A}_{l,k}$ and $\mathcal{B}_{l,k}$ are introduced in order to transform the original RaFM in Eq. (1)(4) to a computationally efficient form. Actually, the following properties are obvious:

Proposition 2. *We have the following properties regarding* $\mathcal{A}_{l,k}$ *and* $\mathcal{B}_{l,k}$*:*

1. $\mathcal{A}_{k,1}$ *and* $\mathcal{B}_{k,k}$ *both denote the* k*-th FM model in Eq.*(2).
2. *The RaFM model in Eq.* (1)(4) *is* $\mathcal{B}_{1,m}$*;*
3. *The computational complexity of* $\mathcal{A}_{l,k}$ *is* $O(D_l |\mathcal{F}_k|)$ *due to the second equality in Eq.* (7).

The proof is omitted since these statements are obvious. According to Proposition 2, the time complexity of $\mathcal{A}_{l,k}$ is known, and all we need to do is to compute $\mathcal{B}_{l,k}$ according to $\mathcal{A}$. To achieve this, we provide the following theorem:

Theorem 3. $\mathcal{A}_{l,k}$ *and* $\mathcal{B}_{l,k}$ *satisfy the following equality:* $\mathcal{B}_{l,k+1} = \mathcal{B}_{l,k} - \mathcal{A}_{k,k+1} + \mathcal{A}_{k+1,k+1}$

Proof. It is easy to show that

$$\begin{aligned}\mathcal{B}_{l,k+1} = \mathcal{B}_{l,k} &- \sum_{i<j, k_{ij}>k} \boldsymbol{v}_i^{(k)} \cdot \boldsymbol{v}_j^{(k)} x_i x_j \\ &+ \sum_{i<j, k_{ij}>k} \boldsymbol{v}_i^{(k+1)} \cdot \boldsymbol{v}_j^{(k+1)} x_i x_j\end{aligned} \qquad (9)$$

Moreover, according to (4), the feature set $\{i < j : k_{ij} > k\}$ can be rewritten as

$$\{i < j : k_{ij} > k\} = \{i < j : i, j \in \mathcal{F}_{k+1}\} \qquad (10)$$

Then we have the 4th property according to the definition of $\mathcal{A}_{k,k+1}$ and $\mathcal{A}_{k+1,k+1}$. □

Corollary 4. *Regarding the computational complexity of RaFM, we have*

1. *The time complexity of* $\mathcal{B}_{l,k}$ *is* $O\left(D_l |\mathcal{F}| + \sum_{p=l+1}^k D_p |\mathcal{F}_p|\right)$*;*
2. *The time complexity of the RaFM model* (3)*, i.e.* $\mathcal{B}_{1,m}$*, is* $O\left(\sum_{k=1}^m D_k |\mathcal{F}_k|\right)$.

Proof. We only prove the 1st statement, and the 2nd statement is the direct corollary of the 1st when $l = 1, k = m$.

We prove by induction. When $k = l$, the time complexity of $\mathcal{B}_{l,l}$ is $O(D_l |\mathcal{F}|)$. If the time complexity of $\mathcal{B}_{l,k}$ is $O\left(D_l |\mathcal{F}| + \sum_{p=l+1}^k D_p |\mathcal{F}_p|\right)$, then the time complexity of $\mathcal{B}_{l,k+1}$ should be

$$\begin{aligned}O\left(D_l |\mathcal{F}| + \sum_{p=l+1}^{k} D_p |\mathcal{F}_p|\right) + O\left(D_k |\mathcal{F}_{k+1}|\right) \\ + O\left(D_{k+1} |\mathcal{F}_{k+1}|\right) = O\left(D_l |\mathcal{F}| + \sum_{p=l+1}^{k+1} D_p |\mathcal{F}_p|\right)\end{aligned} \qquad (11)$$

Table 1. Complexity of RaFM under different conditions

	$D_k = \Theta(\frac{k}{m}D)$	$D_k = \Theta(2^{k-m}D)$
$\lvert\mathcal{F}_k\rvert = \Theta((1-\frac{k-1}{m})\lvert\mathcal{F}\rvert)$	$O(mD\lvert\mathcal{F}\rvert)$	$O(D\lvert\mathcal{F}\rvert)$
$\lvert\mathcal{F}_k\rvert = \Theta(2^{1-k}\lvert\mathcal{F}\rvert)$	$O(D\lvert\mathcal{F}\rvert)$	$O(\frac{m}{2^{m-1}}D\lvert\mathcal{F}\rvert)$

where the fact $D_k \leq D_{k+1}$ is used. $\square$

Remark 5. Similar to FM, when the data is sparse, $|\mathcal{F}_k|$ should be replaced by $n(\mathcal{F}_k)$, which is the expected number of occurrences of $\mathcal{F}_k$ in a data sample. In such cases, the time complexity is in the sense of expectation.

3.3. Comparison with FM

This section compares the complexity of RaFM with that of FM. When using a single FM as the predictor, we usually use a large rank to ensure the performance, and use regularization to avoid overfitting. Here we use the FM with rank D_m for comparison, of which the space and time complexities are $O(D|\mathcal{F}|)$ and $O(Dn(\mathcal{F}))$ respectively.

The space and time complexity of RaFM are similar to each other except that $|\mathcal{F}_k|$ should be replaced by $n(\mathcal{F}_k)$ in the time complexity, so we discuss them together. Generally, when k increases, D_k will increase and $|\mathcal{F}_k|$ will decrease. Table 1 provides the complexity under different speeds of D_k increasing and $|\mathcal{F}_k|$ decreasing. It is shown that if D_k increases and $|\mathcal{F}_k|$ decreases moderately (e.g. linearly), the complexity of RaFM will be large. However, if one or two of them vary rapidly (e.g. exponentially), the complexity of RaFM will be comparable or smaller than the FM model.

In practice, it is widely-accepted that D_k varies rapidly when k increases (He & Chua, 2017; Li et al., 2017). As indicated by Fig. 1, $|\mathcal{F}_k|$ decreases rapidly when k increases. In contrast, the speed of $n(\mathcal{F}_k)$ decreasing may not be as rapidly as that of $|\mathcal{F}_k|$, but is still likely to be superlinear. Therefore, in such conditions, *RaFM will significantly reduce the space complexity of FM while incur a comparable or smaller time complexity.* This statement will also be validated by experiments in Section 6.1.3.

4. Efficient Learning of RaFM

This section provides a computationally efficient learning algorithm of RaFM, i.e. Algorithm 1, where the inactive factors need not be stored and trained. We first provide the objective function and the learning algorithm, then prove that the proposed algorithm is to train the upper bounds of all the m FMs simultaneously.

We emphasize that although the analysis in this section is somehow technical, the final algorithm, i.e., Algorithm 1, can be regarded as an extension to SGD methods, hence is easy to implement.

Algorithm 1 Training the RaFM

```
1: Initialize all the parameters
2: while not convergent do
3:     Sample a data point $(\boldsymbol{x}, y)$ randomly
4:     for $1 \leq p < m$ do
5:         $\boldsymbol{v}^{(p)}\big|_{\mathcal{F}_{p+1}} \leftarrow \boldsymbol{v}^{(p)}\big|_{\mathcal{F}_{p+1}} - \rho_d \frac{\partial L(\mathcal{B}_{1,p},\mathcal{B}_{1,p+1})}{\partial \boldsymbol{v}^{(p)}\big|_{\mathcal{F}_{p+1}}}$
6:         $\boldsymbol{v}^{(p)}\big|_{\mathcal{F}_p-\mathcal{F}_{p+1}} \leftarrow \boldsymbol{v}^{(p)}\big|_{\mathcal{F}_p-\mathcal{F}_{p+1}} - \rho_f \frac{\partial L(\mathcal{B}_{1,m},y)}{\partial \boldsymbol{v}^{(p)}\big|_{\mathcal{F}_p-\mathcal{F}_{p+1}}}$
7:     end for
8:     $\boldsymbol{v}^{(m)}\big|_{\mathcal{F}_m} \leftarrow \boldsymbol{v}^{(m)}\big|_{\mathcal{F}_m} - \rho_f \frac{\partial L(\mathcal{B}_{1,m},y)}{\partial \boldsymbol{v}^{(m)}\big|_{\mathcal{F}_m}}$
9: end while
```

4.1. Constrained Optimization of RaFM

The goal of the training algorithm is to obtain the parameters in $\mathcal{B}_{1,m}$. According to Eq. (7) and Theorem 3, the parameters are $\boldsymbol{v}^{(p)}\big|_{\mathcal{F}_p}, \forall 1 \leq p \leq m$. In other words, $\boldsymbol{v}^{(p)}\big|_{\mathcal{F}-\mathcal{F}_p}, \forall 1 \leq p \leq m$ are the inactive factors that need not be trained. In order to avoid the training of inactive factors, we provide the following bi-level optimization model:

$$\min \frac{1}{N}\sum_{\boldsymbol{x}} L(\mathcal{B}_{1,m}, y) \tag{12a}$$

$$\text{s.t. } \boldsymbol{v}^{(p)}\Big|_{\mathcal{F}_{p+1}} = \arg\min \frac{1}{N}\sum_{\boldsymbol{x}} L(\mathcal{B}_{1,p}, \mathcal{B}_{1,p+1}), \forall 1 \leq p < m \tag{12b}$$

where L is the loss function, and $\boldsymbol{v}^{(p)}\big|_{\mathcal{F}_{p+1}}$ denotes the $\boldsymbol{v}_i^{(p)}$ where $i \in \mathcal{F}_{p+1}$.

The variables in Eq. (12) can be classified into two groups, i.e. free variables and dependent variables. $\boldsymbol{v}^{(p)}\big|_{\mathcal{F}_p-\mathcal{F}_{p+1}}, \forall 1 \leq p < m$ and $\boldsymbol{v}^{(m)}\big|_{\mathcal{F}_m}$ are free variables, while $\boldsymbol{v}^{(p)}\big|_{\mathcal{F}_{p+1}}, \forall 1 \leq p < m$ are dependent variables since they are determined by $\boldsymbol{v}^{(p+1)}\big|_{\mathcal{F}_{p+1}}$ via the constraint (12b). The basic idea of Eq. (12) is to regard $\boldsymbol{v}_i$ in its highest dimensionality as free variables to be optimized, and to regard other lower-dimensionality counterparts as its "projections" in lower dimensions, which can be approximated by the constraint (12b). Note that inactive factors $\boldsymbol{v}^{(p)}\big|_{\mathcal{F}-\mathcal{F}_p}, 1 \leq p \leq m$ does not exist in Eq. (12). Therefore, Eq. (12) makes it possible to maintain the model size given by Proposition 1 in the training process.

4.2. Learning

This subsection will show that Eq. (12) can be efficiently trained. Due to (12b), the dependent variables $\boldsymbol{v}^{(p)}\big|_{\mathcal{F}_{p+1}}$ can be obtained by taking the stochastic gradient descent (SGD) algorithm on the first term in $L(\mathcal{B}_{1,p}, \mathcal{B}_{1,p+1})$. The major challenge is the estimation of the gradient direction of the free variables due to that the optimization problem (12) is a multi-stage optimization problem. Taking $\boldsymbol{v}^{(p)}\big|_{\mathcal{F}_p-\mathcal{F}_{p+1}}$ for a certain p as an example, the gradient with respect to $\boldsymbol{v}^{(p)}\big|_{\mathcal{F}_p-\mathcal{F}_{p+1}}$ should be

$$grad = \frac{1}{N}\sum_{\boldsymbol{x}}\left(\frac{\partial L(\mathcal{B}_{1,m},y)}{\partial\, \boldsymbol{v}^{(p)}\big|_{\mathcal{F}_p-\mathcal{F}_{p+1}}} + \frac{\partial L(\mathcal{B}_{1,m},y)}{\partial\, \boldsymbol{v}^{(p-1)}\big|_{\mathcal{F}_p}}\frac{\partial\, \boldsymbol{v}^{(p-1)}\big|_{\mathcal{F}_p}}{\partial\, \boldsymbol{v}^{(p)}\big|_{\mathcal{F}_p-\mathcal{F}_{p+1}}}\right) \tag{13}$$

where $\partial\, \boldsymbol{v}^{(p-1)}\big|_{\mathcal{F}_p} / \partial\, \boldsymbol{v}^{(p)}\big|_{\mathcal{F}_p-\mathcal{F}_{p+1}}$ is a $|\mathcal{F}_p|\, D_{p-1} \times (|\mathcal{F}_p| - |\mathcal{F}_{p+1}|) D_p$ matrix representing the relationship between the dependent variable $\boldsymbol{v}^{(p-1)}\big|_{\mathcal{F}_p}$ and the free variable $\boldsymbol{v}^{(p)}\big|_{\mathcal{F}_p-\mathcal{F}_{p+1}}$ given by the constraint (12b). According to the theorem of implicit functions,

$$\frac{\partial\, \boldsymbol{v}^{(p-1)}\big|_{\mathcal{F}_p}}{\partial\, \boldsymbol{v}^{(p)}\big|_{\mathcal{F}_p-\mathcal{F}_{p+1}}} = \left[\frac{1}{N}\sum_{\boldsymbol{x}}\frac{\partial^2 L(\mathcal{B}_{1,p-1},\mathcal{B}_{1,p})}{\partial\left(\boldsymbol{v}^{(p-1)}\big|_{\mathcal{F}_p}\right)^2}\right]^{-1} \left[\frac{1}{N}\sum_{\boldsymbol{x}}\frac{\partial^2 L(\mathcal{B}_{1,p-1},\mathcal{B}_{1,p})}{\partial\, \boldsymbol{v}^{(p-1)}\big|_{\mathcal{F}_p}\partial\, \boldsymbol{v}^{(p)}\big|_{\mathcal{F}_p-\mathcal{F}_{p+1}}}\right] \tag{14}$$

Apply Eq. (14) to Eq. (13),

$$grad = \frac{1}{N}\sum_{\boldsymbol{x}}\frac{\partial L(\mathcal{B}_{1,m},y)}{\partial\, \boldsymbol{v}^{(p)}\big|_{\mathcal{F}_p-\mathcal{F}_{p+1}}} + \frac{1}{N^2}\sum_{\boldsymbol{x},\boldsymbol{x}'}\frac{\partial L(\mathcal{B}_{1,m},y)}{\partial\, \boldsymbol{v}^{(p-1)}\big|_{\mathcal{F}_p}}\boldsymbol{G}^{-1}\frac{\partial^2 L(\mathcal{B}'_{1,p-1},\mathcal{B}'_{1,p})}{\partial\, \boldsymbol{v}^{(p-1)}\big|_{\mathcal{F}_p}\partial\, \boldsymbol{v}^{(p)}\big|_{\mathcal{F}_p-\mathcal{F}_{p+1}}} \tag{15}$$

where $\boldsymbol{G} = \frac{1}{N}\sum_{\boldsymbol{x}}\left[\partial^2 L / \partial\left(\boldsymbol{v}^{(p-1)}\big|_{\mathcal{F}_p}\right)^2\right]$ is the first term in the right of (14), which is a $|\mathcal{F}_p|\, D_{p-1} \times |\mathcal{F}_p|\, D_{p-1}$ matrix. $\mathcal{B}'_{l,k}$ denotes the $\mathcal{B}_{l,k}$ of input vector $\boldsymbol{x}'$. By exchanging $\boldsymbol{x}$ and $\boldsymbol{x}'$ in the second term of the right of (15), and using one sample $\boldsymbol{x}$ to estimate the gradient, we have

$$\widehat{grad} = \frac{\partial L(\mathcal{B}_{1,m},y)}{\partial\, \boldsymbol{v}^{(p)}\big|_{\mathcal{F}_p-\mathcal{F}_{p+1}}} + \frac{1}{N}\sum_{\boldsymbol{x}'}\frac{\partial L(\mathcal{B}'_{1,m},y)}{\partial\, \boldsymbol{v}^{(p-1)}\big|_{\mathcal{F}_p}}\boldsymbol{G}^{-1}\frac{\partial^2 L(\mathcal{B}_{1,p-1},\mathcal{B}_{1,p})}{\partial\, \boldsymbol{v}^{(p-1)}\big|_{\mathcal{F}_p}\partial\, \boldsymbol{v}^{(p)}\big|_{\mathcal{F}_p-\mathcal{F}_{p+1}}} \tag{16}$$

In the right of Eq. (16), the first term can be obtain by the chain rule, while the second term contains second order derivatives which are challenging to compute. However, we have the following theorem:

Theorem 6. *The direction of $\widehat{grad}$ is parallel to its first term $\partial L(\mathcal{B}_{1,m},y)/\partial\, \boldsymbol{v}^{(p)}\big|_{\mathcal{F}_p-\mathcal{F}_{p+1}}$.*

Proof. See Section 1 in the supplementary material. □

In SGD, the direction of the gradient vector is more important than the length. Theorem 6 shows that we can use the first term in Eq. (16) to estimate the descent direction over free variables. The same discussion can be applied to $\boldsymbol{v}^{(m)}\big|_{\mathcal{F}_m}$, therefore we can use Algorithm 1 to learn Eq. (12), where ρ_d and ρ_f are the learning rates of dependent variables and free variables, respectively.

Now we discuss the complexity of Algorithm 1. Note that in each step, only active factors are used and updated, which means the space complexity follows Proposition 1. Moreover, we do not need to compute $\mathcal{B}_{1,p}$ seperately since the it is a part of $\mathcal{B}_{1,m}$. Therefore, the time complexity also follows Corollary 4. Therefore, as discussed in Section 3.3, training RaFM via Algorithm 1 is as efficient as, or more efficient than training a single FM.

Moreover, it can be proven that the proposed learning algorithm minimizes an upper bound of each FM with a single rank $D_k, 1 \le k \le m$. We put the proof in Section 2 of the supplementary material due to space constraints.

5. Related Work

The most related researches on the combination of FMs with different ranks are DiFacto(Li et al., 2016), MRMA(Li et al., 2017), and FFM(Juan et al., 2016). We compare them with RaFM thoroughly in this section. There are many other researches to improve the performance of FMs by deep models, such as NFM(He & Chua, 2017), AFM(Xiao et al., 2017), DeepFM(Guo et al., 2017), etc. The idea of RaFM is orthogonal to these researches. It is also an attractive direction to combine RaFM with these models, and we leave it for future work.

5.1. DiFacto

DiFacto also uses multiple ranks in one FM. However, for each feature, DiFacto only allocates an embedding vector with a single rank, while RaFM allocates multiple embeddings with different ranks. In DiFacto, the pairwise interaction between features with different ranks is obtained by simply truncating the embedding with a higher rank to a lower rank. In cases such as SVD of a complete matrix, such truncations are reasonable, since the best k-rank approximation is equivalent to the k-prefix of any $k+n$-rank approximation. However, in recommender systems where the training samples are highly sparse, such truncations usually lead to worse performances, as will be shown in Section 6. Therefore, DiFacto reduces the computational burden of FM by sacrificing the performance, while RaFM improves the performance of FM with a lower computational burden.

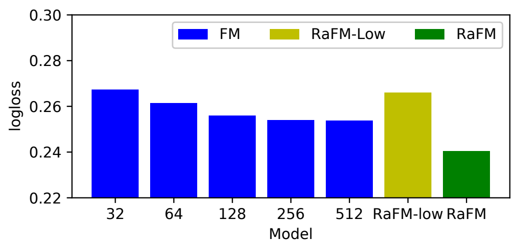

Figure 3. FM vs. RaFM in ML Tag

Table 2. Performance and Complexity under Different Settings

	logloss	#param	train time
FM(D=512)	0.2538	46.40M	1×
RaFM(S_1)	0.2405	17.89M	0.43×
RaFM(S_2)	0.2416	9.63M	0.17×
RaFM(S_3)	0.2387	9.23M	0.24×
RaFM(S_4)	0.2391	17.75M	1.35×

5.2. MRMA

MRMA is a matrix approximation model, of which the key idea is also to combine models with different ranks. The major difference is that it stores the entire models with different ranks, thus leading to a large computational burden and storage burden. Moreover, the large number of parameters may also cause severe overfitting problems, and this is why (Li et al., 2017) provides the Iterated Conditional Modes (ICM) to train MRMA. In contrast, RaFM will significantly reduce the number of parameters by eliminating inactive factors, and will not be likely to suffer from overfitting problems.

5.3. FFM

RaFM is totally different from FFM, although they both use multiple embeddings for each feature. On the one hand, FFM uses different embeddings for interactions between different field-pairs, while the concept "field" never exists in RaFM. For problems without the concept of fields or problems with only 2 fields, FFM fails or degenerates to the original FM, while RaFM still works. On the other hand, different embeddings in FFM are independent, while embeddings in a lower rank can be regarded as the projection of embeddings in higher rank, which is guaranteed by the learning algorithm. This property largely reduces the computational burden and avoids the overfitting problem, while FFM suffers from the large computational burden.

6. Experiments

This section provides the experiments of the proposed approach and other benchmarks in several datasets. In Experiment A, we test our approach on 7 public datasets while in Experiment B, we perform the RaFM approach on the news CTR data provided by Tencent to show the effectiveness of the proposed approach in industrial applications. The code of RaFM is available at https://github.com/cxsmarkchan/RaFM.

6.1. Experiment A: Public Datasets

6.1.1. Experiment Setup

We consider 3 datasets for regression tasks and 4 for classification tasks. All these datasets are randomly split into train (80%), validation (10%), and test (10%) sets. Datasets for regression tasks are the MovieLens 10M (ML 10M), 20M (ML 20M), and the Amazon movie review dataset[3] (AMovie), respectively, of which the square loss is used as the performance criterion. Datasets for classification tasks are Frappe[4], Movielens Tag (ML Tag), Avazu, and Criteo[5], respectively, of which the log loss and the area under curve (AUC) are used as the performance criteria.

We use the standard FM (Rendle, 2010) and Difacto (Li et al., 2016) as baselines for all datasets, and MRMA as a baseline for datasets containing exactly two fields, i.e. ML 10M, ML 20M, and AMovie, respectively. We do not compare with deep models since the motivation of this work is to show that FMs with different ranks can be efficiently combined and trained while incurring less computational burden. However, we argue that RaFM is a flexible framework and deep FMs can be embedded. We adopt L_2 regularizations for each model, and search the L_2 coefficient from $\{1e^{-6}, 5e^{-6}, 1e^{-5}, \ldots, 1e^{-1}\}$ on the validation set. We search the ranks from $\{32, 64, 128, 256, 512\}$ for each model, except for some discussions in Section 6.1.3, where we use more values. We compare the performance, the model size, the training time, and the test time. Since RaFM has more hyperparameters, to be fair, we tune k_i by the following equation rather than grid search:

$$k_i = \arg\min_k |\log n_i - \log D_k| \tag{17}$$

where n_i is the number of occurrences of the i-th feature. This equation means choosing k_i so that D_k is the closest to n_i in the sense of logarithm. Moreover, we use the same L_2 coefficient for all FM models in the RaFM, and search it from the same candidate set as baselines.

6.1.2. Advantages of RaFM over FM with Fixed Ranks

Fig. 3 shows the comparison between FMs with different ranks and RaFM in the ML Tag dataset. The ranks of FMs range from 32 to 512, and the hyperparameters of RaFM

[3] http://jmcauley.ucsd.edu/data/amazon/
[4] http://baltrunas.info/research-menu/frappe
[5] http://labs.criteo.com/2014/02/kaggle-display-advertising-challenge-dataset/

Table 3. Results on Regression Tasks

	ML 10M			ML 20M			AMovie		
	square loss	#param	train/test time	square loss	#param	train/test time	square loss	#param	train/test time
FM	0.8016 $\pm$0.0010	2.66M	1$\times$	0.8002 $\pm$0.0008	5.45M	1$\times$	1.0203 $\pm$0.0046	3.25M	1$\times$
DiFacto	0.7950 $\pm$0.0011	1.79M	0.82$\times$/ 0.95$\times$	0.7948 $\pm$0.0005	3.22M	0.70$\times$/ 0.80$\times$	1.0268 $\pm$0.0051	1.76M	0.75$\times$/ 0.75$\times$
MRMA	0.7952 $\pm$0.0006	4.11M	1.27$\times$/ 1.43$\times$	0.7855 $\pm$0.0011	8.43M	1.19$\times$/ 1.38$\times$	1.0071 $\pm$0.0039	5.02M	1.27$\times$/ 1.27$\times$
RaFM	**0.7870** **$\pm$0.0008**	1.57M	0.95$\times$/ 1.12$\times$	**0.7807** **$\pm$0.0009**	3.63M	0.74$\times$/ 0.85$\times$	**0.9986** **$\pm$0.0035**	1.76M	0.75$\times$/ 0.75$\times$

Table 4. Results on Classification Tasks

	Frappe				ML Tag			
	log loss	AUC	#param	train/test time	log loss	AUC	#param	train/test time
FM	0.1702 $\pm$0.0023	0.9771 $\pm$0.0008	1.38M	1$\times$	0.2538 $\pm$0.0009	0.9503 $\pm$0.0006	46.40M	1$\times$
DiFacto	0.1711 $\pm$0.0023	0.9771 $\pm$0.0004	0.61M	0.63$\times$/ 0.85$\times$	0.2529 $\pm$0.0007	0.9450 $\pm$0.004	16.97M	0.42$\times$/ 0.83$\times$
RaFM	**0.1447** **$\pm$0.0015**	**0.9811** **$\pm$0.0002**	0.71M	0.73$\times$/ 0.85$\times$	**0.2387** **$\pm$0.0005**	**0.9526** **$\pm$0.0006**	9.23M	0.24$\times$/ 0.71$\times$
	Avazu				**Criteo**			
	log loss	AUC	#param	train/test time	log loss	AUC	#param	train/test time
FM	0.3817 $\pm$0.0001	0.7761 $\pm$0.0003	18.81M	1$\times$	0.4471 $\pm$0.0002	0.8030 $\pm$0.0002	35.87M	1$\times$
DiFacto	0.3823 $\pm$0.0003	0.7778 $\pm$0.0003	10.83M	0.82$\times$/ 1.79$\times$	0.4470 $\pm$0.0002	0.8030 $\pm$0.0004	19.70M	0.63$\times$/ 0.80$\times$
RaFM	**0.3801** **$\pm$0.0002**	**0.7826** **$\pm$0.0003**	10.17M	0.85$\times$/ 1.20$\times$	**0.4451** **$\pm$0.0001**	**0.8060** **$\pm$0.0002**	20.88M	0.67$\times$/ 0.84$\times$

are $m = 2, D_1 = 32$ and $D_2 = 512$. The RaFM-low in Fig. 3 represents the $\mathcal{B}_{1,1}$, i.e. the FM model with rank 32, but trained simultaneously with $\mathcal{B}_{1,2}$ by Eq. (12b). It is shown that RaFM significantly outperforms all FMs. It is because that RaFM maintains 32 factors for most of the features with limited occurrences to avoid overfitting problems, and allocate 512 factors for features with enough occurrences to guarantee expressiveness. Moreover, RaFM-low achieves similar performance to FM with 32 factors, which means embeddings in RaFM has similar expressiveness of embeddings in FM with the same rank.

6.1.3. Performance and Complexity

Table 2 shows the performance and complexity of RaFMs under different rank settings. We compare 4 sets of ranks, namely, $S_1 = \{32, 512\}$, $S_2 = \{32, 128, 512\}$, $S_3 = \{32, 64, 128, 256, 512\}$, and $S_4 = \{32, 64, 96, 128, 160, \cdots, 480, 512\}$. Therefore, ranks in S_2, S_3 varies exponentially, while ranks in S_4 varies arithmetically. RaFMs with these 4 sets achieves similar performances, all significantly better than FM with 512 factors, which is the best out of all FMs as shown in Fig. 3. RaFMs with S_2, S_3 have fewer parameters due to the expenentially increasing rank settings. RaFM with S_4 has a comparable number of parameters to S_1, but the train time is much longer. This comparison shows the impacts of the speed of rank increasing on the computational burden, which is consistent with what is discussed in Section 3.3.

6.1.4. Results and Discussion

As evidenced by Tables 3 and 4, our algorithm significantly outperforms the baseline algorithms. For example, the relative improvements on square loss (log loss) criteria are $1\% \sim 2\%$ in ML 10M, ML 20M and AMovie, 15% on Frappe, 6% on ML Tag, and 0.5% on Avazu and Criteo. Empirically, all these improvements are regarded as significant in the researches on corresponding datasets. Taking Criteo as an example, RaFM reduces the logloss of FM by 0.002, while an improvement of 0.001 in logloss is considered as practically significant (Wang et al., 2017).

Moreover, the complexity of RaFM is reduced compared to FMs. In fact, RaFM significantly reduces both the model size and the computational time. For example, the model size of RaFM is only 20%∼66% of FM, and the training time is 24%∼95% of FM.

Although Difacto can also reduce the computational burden, it cannot guarantee the performance. The major reason is that Difacto assumes the low-dimensional counterpart of a high-dimensional feature can be obtained by parameter sharing. However, this assumption is not usually reasonable. In contrast, MRMA performs better than FM and DiFacto due to it combines models with different ranks. However, its model size and training time are significantly larger than RaFM, while its performance is a bit worse than that of RaFM due the overfitting problem caused by a large number of parameters.

In summary, RaFM not only achieves a better performance, but also reduces the model size and computational time. Therefore, the proposed RaFM model also has an attractive potential in industrial applications.

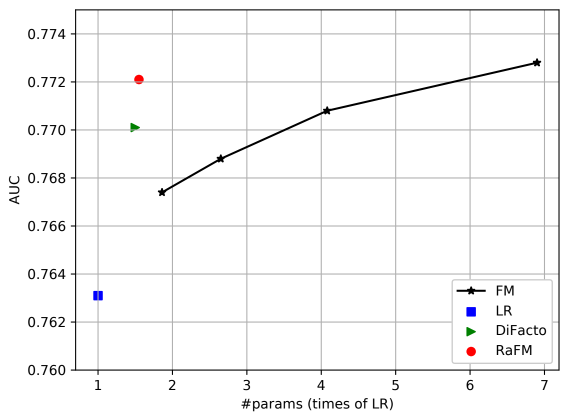

Figure 4. Comparisons of AUC and the model size on industrial level CTR dataset.

6.2. Experiment B: Industrial Level Click-Through-Rate Dataset

6.2.1. Experiment Setup

Here we perform an experiment on the news CTR data provided by Tencent in order to show the potential of RaFM in industrial applications. We use the records of 8 consecutive days, of which the first 7 days are used as the training dataset, the last day is used on the test dataset. The dataset contains 1.7 billion records and 120 millions features.

We compare the proposed RaFM approach with LR, FM and DiFacto, and the performance standard is the AUC, which is consistent with the requirement of online predictions. For FM/DiFacto/RaFM, the dimensionality of factors are chosen from $\{1, 2, 4, 8\}$. We use the FTRL algorithm to guarantee the sparsity of learned weights, and use distributed learning to accelerate the learning process.

6.2.2. Results

We show the comparisons of AUC and the model size in Fig. 4. We use the number of parameters of LR, which is 13.19M in our experiments, as the unit of the model size. The comparison of training time of these approaches is not provided here, because they are very similar (about 6 hours for training, and 30 minutes for testing in our experiments) due to the fact that the training timeis dominated by the communication time of distributed learning rather than the computational time.

Since the number of records in the training dataset is sufficiently large, it is natural to hope that the AUC will continuously increase when the model size increases, and the AUCs in Fig. 4 can be further improved if we allow a larger rank of factors. However, the model size to achieve such an improvement is the main issue, since a larger model size will undoubtedly bring a heavier burden to the parameter server. According to Fig. 4, RaFM increases the AUC by about 1% compared to LR, while its model size is only 1.55 times that of LR. In comparison, FM needs 7 times the model size of LR to achieve a similar performance. In other words, the model size of RaFM is only 22% that of FM to achieve a similar AUC, which means the proposed RaFM approach achieves a good trade-off between model size and performance, and has an attractive potential in industrial applications.

7. Conclusion and Future Work

This paper proposes an RaFM model which adopts pairwise interactions from embeddings with different ranks. RaFM can be stored and evaluated as efficiently as, or even more efficiently than FMs with fixed ranks. Moreover, we provide a learning algorithm for efficiently training all embeddings in one concise model, and prove that the training error of each FM is bounded from above. Experiments demonstrate that RaFM not only has better performance in regression and classification datasets whose different features have significantly varying frequencies of occurrence, but also reduces the computational burden of FMs.

RaFM is a flexible framework, therefore an interesting direction in future study is to combine it with deep models in order to achieve better performances. Moreover, a more effective hyperparameter tuning approach is also an attractive research direction.

References

Blondel, M., Fujino, A., and Ueda, N. Convex factorization machines. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 19–35. Springer, 2015.

Blondel, M., Fujino, A., Ueda, N., and Ishihata, M. Higher-order factorization machines. In *Advances in Neural Information Processing Systems*, pp. 3351–3359, 2016.

Du, C., Li, C., Zheng, Y., Zhu, J., and Zhang, B. Collaborative filtering with user-item co-autoregressive models. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Guo, H., Tang, R., Ye, Y., Li, Z., and He, X. Deepfm: A factorization-machine based neural network for ctr prediction. *arXiv preprint arXiv:1703.04247*, 2017.

He, X. and Chua, T.-S. Neural factorization machines for sparse predictive analytics. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*, pp. 355–364. ACM, 2017.

Jiang, Z., Zheng, Y., Tan, H., Tang, B., and Zhou, H. Variational deep embedding: An unsupervised and generative approach to clustering. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, pp. 1965–1972. AAAI Press, 2017. ISBN 978-0-9992411-0-3. URL `http://dl.acm.org/citation.cfm?id=3172077.3172161`.

Juan, Y., Zhuang, Y., Chin, W.-S., and Lin, C.-J. Field-aware factorization machines for ctr prediction. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pp. 43–50. ACM, 2016.

Lauly, S., Zheng, Y., Allauzen, A., and Larochelle, H. Document neural autoregressive distribution estimation. *The Journal of Machine Learning Research*, 18(1):4046–4069, 2017.

Li, D., Chen, C., Liu, W., Lu, T., Gu, N., and Chu, S. Mixture-rank matrix approximation for collaborative filtering. In *Advances in Neural Information Processing Systems*, pp. 477–485, 2017.

Li, L., Zhao, P., Zhou, J., and Li, X. A boosting framework of factorization machine. *arXiv preprint arXiv:1804.06027*, 2018.

Li, M., Liu, Z., Smola, A. J., and Wang, Y.-X. Difacto: Distributed factorization machines. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, pp. 377–386. ACM, 2016.

Liu, C., Zhang, T., Zhao, P., Zhou, J., and Sun, J. Locally linear factorization machines. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pp. 2294–2300. AAAI Press, 2017.

Rendle, S. Factorization machines. In *IEEE 10th International Conference on Data Mining (ICDM)*, pp. 995–1000. IEEE, 2010.

Rendle, S. Factorization machines with libfm. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 3 (3):57, 2012.

Wang, R., Fu, B., Fu, G., and Wang, M. Deep & cross network for ad click predictions. *CoRR*, abs/1708.05123, 2017. URL `http://arxiv.org/abs/1708.05123`.

Xiao, J., Ye, H., He, X., Zhang, H., Wu, F., and Chua, T.-S. Attentional factorization machines: Learning the weight of feature interactions via attention networks. *arXiv preprint arXiv:1708.04617*, 2017.

Zheng, Y., Liu, C., Tang, B., and Zhou, H. Neural autoregressive collaborative filtering for implicit feedback. In *Proceedings of the 1st workshop on deep learning for recommender systems*, pp. 2–6. ACM, 2016a.

Zheng, Y., Tang, B., Ding, W., and Zhou, H. A neural autoregressive approach to collaborative filtering. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pp. 764–773. JMLR.org, 2016b. URL `http://dl.acm.org/citation.cfm?id=3045390.3045472`.