Gradient Boosting Factorization Machines

Chen Cheng[1,2]* , Fen Xia[3], Tong Zhang[3],Irwin King[1,2], Michael R. Lyu[1,2]
[1] Shenzhen Key Laboratory of Rich Media Big Data Analytics and Applications,
Shenzhen Research Institute, The Chinese University of Hong Kong, Shenzhen, China
[2] Department of Computer Science and Engineering,
The Chinese University of Hong Kong, Shatin, N.T., Hong Kong
[3] Big Data Lab, Baidu Inc., No.10 Shangdi Street, Haidian, Beijing, China
{ccheng,king,lyu}@cse.cuhk.edu.hk {xiafen,tongzhang}@baidu.com

ABSTRACT

Recommendation techniques have been well developed in the past decades. Most of them build models only based on user item rating matrix. However, in real world, there is plenty of auxiliary information available in recommendation systems. We can utilize these information as additional features to improve recommendation performance. We refer to recommendation with auxiliary information as context-aware recommendation. Context-aware Factorization Machines (FM) is one of the most successful context-aware recommendation models. FM models pairwise interactions between all features, in such way, a certain feature latent vector is shared to compute the factorized parameters it involved. In practice, there are tens of context features and not all the pairwise feature interactions are useful. Thus, one important challenge for context-aware recommendation is how to effectively select "good" interaction features. In this paper, we focus on solving this problem and propose a greedy interaction feature selection algorithm based on gradient boosting. Then we propose a novel Gradient Boosting Factorization Machine (GBFM) model to incorporate feature selection algorithm with Factorization Machines into a unified framework. The experimental results on both synthetic and real datasets demonstrate the efficiency and effectiveness of our algorithm compared to other state-of-the-art methods.

Categories and Subject Descriptors

H.1.1 [**Models and Principles**]: Systems and Information Theory; J.4 [**Computer Application**]: Social and Behavior Sciences

Keywords

Gradient Boosting, Factorization Machines, Recommender Systems, Collaborative filtering

1. INTRODUCTION

Recommendation systems have been well studied in the past decades. Most of them mainly focus on context unaware methods, e.g. only consider user and item interactions. Among them, matrix factorization methods [16, 27] have become popular due to their good performance and efficiency in dealing with larger dataset. These methods focus on approximating user item rating matrix using low rank representations, and use them to make further predictions. However, in real world scenarios, plenty of auxiliary information is available and is proved to be useful especially in large industry datasets. For example, in the Weibo celebrity recommendation scenario, both the user's and celebrity's meta data (such as age and gender), the popularity of a celebrity, the recent following behavior of the user, etc. can help make better recommendations. Recent work in KDDCup 2012 [1] [23, 7] show the effectiveness of utilizing auxiliary information for recommendation.

In regarding of utilizing auxiliary information, several methods have been studied to incorporate meta data (e.g., user profile, movie genre,etc.) [18, 30] and more general auxiliary information such as session information [32, 2, 14, 22, 25]. In these methods, auxiliary information is encoded as features and together with user and item they are mapped from feature space into a latent space. The Factorization Machine (FM) model [25] is currently a very strong and flexible method which easily incorporates any categorical features. However, it is common that there are tens of context features in real data. In FM, all features are assumed to be interacted with all other features. For example, assuming there are n features, then for a certain feature i, the latent vector $\boldsymbol{v}_i$ is shared with $n-1$ interaction features. It is not always the case that all the feature interactions are useful. Useless feature interactions will introduce noise in learning latent feature vector $\boldsymbol{v}_i$. Thus it is challenging to automatically select useful interaction features to reduce noise.

The most recent work in [6] introduced an automatic feature construction method in matrix factorization using gradient boosting. In their method, feature functions are constructed using greedy gradient boosting method and then incorporated into the matrix factorization framework. Different from their method, in our paper, we focus on selecting useful interaction features under factorization machines framework. At each step, we propose a greedy gradient boosting method to efficiently select interaction features, and then we additively optimize the selected latent vector by

*The work was performed when the first author was an intern at Big Data Lab of Baidu Inc..

[1] http://www.kddcup2012.org/

RecSys'14, October 6–10, 2014, Foster City, Silicon Valley, CA, USA.
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-2668-1/14/10 ...$15.00.
http://dx.doi.org/10.1145/2645710.2645730.

optimizing the residual loss. Another difference is that our method is more efficient in selecting categorical feature interactions compared to the binary decision tree construction algorithm in [6]. The contribution of our paper is summarized as follows:

- We propose an efficient feature interaction selection algorithm using gradient boosting which can reduce noise compared to factorization machines methods.
- We proposed a novel Gradient Boosting Factorization Machines Model (GBFM) by incorporating the feature selection algorithm with factorization machines into a unified framework.
- The experiment results in both synthetic and real data show the effectiveness of our proposed methods compared to factorization machines and other state-of-the-art methods.

The rest of the paper is organized as follows. Section 2 introduces the related work. Section 3 gives the details of our methods. Section 4 presents the experiment results and discussions. The paper is concluded in Section 5.

2. RELATED WORK

The work present in this paper is closely related to traditional matrix factorization models, context-aware recommendation and gradient boosting. In the following, we briefly review the related work.

2.1 Matrix Factorization

Matrix factorization techniques [29, 4, 16, 27, 2] have been shown to be particularly effective in recommender system as well as the well-known Netflix prize competitions[2]. They usually outperform traditional item-based methods [28]. The main idea behind matrix factorization is to learn two low rank latent matrices $U \in \mathbb{R}^{k\times m}$ and $V \in \mathbb{R}^{k\times n}$ to approximate the observed user item rating matrix $R \in \mathbb{R}^{m\times n}$ so that

$$R \approx U^T V, \tag{1}$$

where m, n are the number of users and items respectively and k is the dimension of low rank matrices.

We assume that the conditional distribution over the observed rating is:

$$p(R|U,V,\sigma_R^2) = \prod_{i=1}^{m}\prod_{j=1}^{n}[\mathcal{N}(R_{ij}|U_iV_j^T,\sigma_R^2)]^{I_{ij}^R}, \tag{2}$$

where $\mathcal{N}(x|\mu,\sigma^2)$ is the probability density function of the Gaussian distribution with mean μ and variance σ^2, and I_{ij}^R is the indicator function that is equal to 1 if user u_i rated item v_j and equal to 0 otherwise. The zero-mean spherical Gaussian priors are also placed on user and item latent feature vectors:

$$p(U|\sigma_U^2) = \prod_{i=1}^{m}\mathcal{N}(U_i|0,\sigma_U^2\boldsymbol{I}), p(V|\sigma_V^2) = \prod_{j=1}^{n}\mathcal{N}(V_j|0,\sigma_V^2\boldsymbol{I}), \tag{3}$$

through a Bayesian inference, we have the following objective function:

$$\min_{U,V}\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n}I_{ij}^R(R_{ij}-U_iV_j^T)^2+\frac{\lambda_1}{2}\|U\|_F^2+\frac{\lambda_2}{2}\|V\|_F^2. \tag{4}$$

The above equation can be easily solved either by stochastic gradient descent (SGD) or alternating least squares (ALS).

2.2 Context-aware recommendation

Most traditional matrix factorization method mainly analyze the user item rating matrix thus they are context unaware. Contextual information has proved to be useful in recommender systems and already have been widely studied. In order to incorporate auxiliary information, several variants of matrix factorization have been proposed. In [17, 33], temporal features are explored to help capture user preference more precisely. Socical [20, 13] or location information [35, 8, 9] also have been explored. The meta-data like user age or item genre are incorporated into the matrix factorization model [30, 18]. In addition to the meta-data which is attached to user or item itself, the context features include information attached to the whole recommendation event such as user's mood, day of the week, etc. as well.

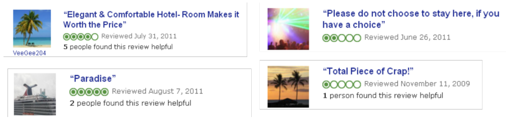
The most basic approach for context aware recommendation is to conduct pre-filtering or post-filtering where a standard context-unaware method is applied [21, 1]. Baltrunas et al. [3] proposed a simple model that introduced a basis term for each context feature or item context interaction feature. Context information is encoded in these additional parameters. More generally, Karatzoglou et al. [14] proposed a multiverse recommendation model by modeling the data as a user-item-context N-dimension tensor. Then Tucker decomposition [31] is applied to factorize the tensor. However, the computation complexity of this model is $\Theta(k^m)$ where k is the dimension of low rank vectors and m is the number of features, which is intolerable in practice. Rendle et al. [25] proposed to apply factorization machines (FM) [22] to overcome the problem in Multiverse recommendation. They transform the recommendation data into a prediction problem and FM models all interactions between pairs of features with the target. They further proposed to deal with relational data in [24] through block structure within a feature which have repeating patterns.

The idea of tree based random partition has been explored in [36, 19]. Zhong et al. [36] assumed that contextual information is reflected by user and item latent vectors. In their method, the random tree partition is conducted to split the user item matrix by grouping users and items with similar contexts. Then matrix factorization is applied on the sub-matrices. Liu et al. [19] employ the similar idea but explicitly use context information to split the user item matrix into sub matrices according to specific context values. The prediction is the average values of each prediction from T generate decision trees. However, they fail to discuss how to select useful features especially when there are tens of features.

2.3 Gradient Boosting

Gradient Boosting have been successfully used in classifications [12] and learning to rank [34, 5]. In each step, gradient boosting greedily conducts coordinate descent in the function space to select a feature function. Chen et al. [6] proposed to use gradient boosting method to automatically

[2] http://www.netflixprize.com



"Elegant & Comfortable Hotel- Room Makes it Worth the Price"
●●●●○ Reviewed July 31, 2011
5 people found this review helpful
VeeGee204

"Please do not choose to stay here, if you have a choice"
●●○○○ Reviewed June 26, 2011

"Paradise"
●●●●● Reviewed August 7, 2011
2 people found this review helpful

"Total Piece of Crap!"
●○○○○ Reviewed November 11, 2009
1 person found this review helpful

(a) Online rating example

	v_1	v_2	v_3	v_4	v_5	v_6
u_1		5	2		3	
u_2	4			3		4
u_3			2			2
u_4	5			3		
u_5		5	5			3

(b) User item rating matrix

	User			Movie				Mood			…	R
$\mathbf{x}^{(1)}$	1	0	0	1	0	0	0	1	0	0	…	4
$\mathbf{x}^{(2)}$	0	1	0	0	1	0	0	0	0	1	…	2
$\mathbf{x}^{(3)}$	1	0	0	0	1	0	0	0	1	0	…	5
$\mathbf{x}^{(4)}$	0	0	1	0	0	1	0	0	0	1	…	1

(c) Context-aware prediction

Figure 1: Context-aware recommendation

construct feature function for each user(item) latent vector at each step. Time dependent feature function and demographic feature function construction is discussed in their work. Different from their work, in our paper, we employ gradient boosting algorithm to find the best feature interaction at each step, then similar to gradient boosting, we additively optimize the latent feature vectors. Besides, in real world, there are many categorical features with large size, the binary tree splitting algorithm in their work is not efficient in deal with such features.

3. GRADIENT BOOSTING FACTORIZATION MACHINES

In this section, we first describe the context-aware recommendation problem we study in this paper and define the notations, and then we briefly review context-aware FMs which is closely related to our work. Then we present the details of our proposed Gradient Boosting Factorization Machines Model followed by complexity analysis and discussions.

3.1 Preliminaries and Problem Definition

Figure 1(a) shows an example of context-aware online rating system. Traditional recommendation systems consider only the user rating matrix 1(b) to make recommendations. However, rich context information is available and easy to obtain in real world. For example, in Fig. 1(a) we can easily get the rating time and the rating comments. These information provide a new information dimension for recommendation. We can encode the context information as well as user and item as either real value or categorical features, and the rating as the target value. In such way, we can transform the context aware recommendation into a prediction problem as shown in Fig.1(c). The figure shows an example about users $\mathcal{U}$ watch movies $\mathcal{I}$ in mood $\mathcal{M}$:

$$\begin{aligned}\mathcal{U} &= \{u_1, u_2, u_3\}\\ \mathcal{I} &= \{i_1, i_2, i_3, i_4\}\\ \mathcal{M} &= \{Happy, Normal, Sad\}\end{aligned}$$

Then the first tuple in Fig.1(c) states that user u_1 gave movie i_1 4 stars in a *Happy* mood.

Next we give the formal definition of context-aware recommendation problem. We denote the user set as $\mathcal{U}$ and the item set as $\mathcal{V}$. Assume there are another $m-2$ context features, we further denote the context features as $\mathcal{C}_3, \ldots, \mathcal{C}_m$. In fact, user and item can be regarded as the first and second "context" feature. For simplicity, we denote user set as $\mathcal{C}_1$ and item set as $\mathcal{C}_2$. In our paper, we consider only categorical features for simplicity, since in practice most features are categorical [23] and for real value features can also be segmented into categorical features. The training data can be encoded as feature vectors as shown in Fig 1(c) by transforming categorical features to indicator variables. We denote n_i as the number of different feature values for context feature $\mathcal{C}_i$. Each context feature set is $\mathcal{C}_i = \{c_{i,0}, \ldots, c_{i,n_i}\}$. We further denote the length of feature vector as d which equals to $n_1 + \ldots + n_m$. Training data is denoted as $\mathcal{S} = \sum_{i=1}^{N}(\mathbf{x}_i, y_i)$, where N is the total training instance number, $\mathbf{x}_i \in \mathbb{R}^d$, $y_i \in \mathbb{R}$ are feature vector and target value for instance i respectively. Our problem is to estimate the following rating function

$$\hat{y} : \mathbb{R}^d \to \mathbb{R}, \tag{5}$$

that minimize the following objective function:

$$\arg\min_{\Theta} \sum_{i=1}^{N} l(\hat{y}_i, y_i) + \Omega(\hat{y}), \tag{6}$$

where l is a differentiable convex loss function that measures the difference between the prediction rating $\hat{y}_i$ and the target rating y_i, Θ is parameter set to be estimated. The second term Ω measures the complexity of the model to avoid overfitting.

3.2 Context-aware FM

Factorization Machines [22] is a generic model class that subsumes many well-known recommendation methods including SVD++[16], matrix factorization [29] and PITF[26]. Rendle et al. [25] proposed to apply FM to solve context-aware recommendation problem and it has proven to be effective in KDDCup 2012 [23] as well.

In [25], factorization machines is restricted to be 2-way FMs. In such setting, the FM models all interactions between pairs of variables with the target including nested ones, by using factorized interaction parameters. The rating prediction function is:

$$\hat{y}(\mathbf{x}) := w_0 + \sum_{i=1}^{d} w_i x_i + \sum_{i=1}^{d} \sum_{j=i+1}^{d} \hat{w}_{i,j} x_i x_j. \tag{7}$$

From the perspective of classification problem, w_0 is the global bias, w_i is the weight for feature x_i and $\hat{w}_{i,j}$ is the weight for feature $x_i x_j$. We refer feature $x_i x_j$ as interaction feature which indicates the instance have both feature value x_i and x_j. For example, the first tuple in Fig. 1(c) one interaction feature is $u_1 v_1$ since we have user u_1 and item v_1 in the tuple.

The factorized parameters $\hat{w}_{i,j}$ is defined as:

$$\hat{w}_{i,j} := \langle \mathbf{v}_i, \mathbf{v}_j \rangle = \sum_{f=1}^{k} v_{i,f} \cdot v_{j,f}. \tag{8}$$

The model parameters Θ that need to be estimated are:

$$w_0 \in \mathbb{R}, \; \mathbf{w} \in \mathbb{R}^d, \; \mathbf{V} \in \mathbb{R}^{d \times k}. \tag{9}$$

Note that the latent matrix $\mathbf{V}$ can be regarded as the concatenation of all m latent feature matrices $\mathbf{V}_i \in \mathbb{R}^{n_i \times k}, i = 1, \ldots, m$. The final objective function is

$$\arg\min_{\Theta} \sum_{i=1}^{N} l(\hat{y}(\mathbf{x}_i), y) + \sum_{\theta \in \Theta} \lambda_{(\theta)} \theta^2, \tag{10}$$

where $\lambda_{(\theta)}$ is the regularization parameters. In practice, l can be logit loss for binary classification problems:

$$l(\hat{y}, y) = \log(1 + \exp(-\hat{y}y)) \tag{11}$$

or least square loss for regression

$$l(\hat{y}, y) = (\hat{y} - y)^2. \tag{12}$$

In practice, it is not surprising that we can have tens of context features [3] and not all interaction features are useful for rating prediction. Note that FM models all pairwise interactions between context features, while the weight of interaction feature is defined in Eq. 8. The latent vector $\mathbf{v}_i$ is shared by all other feature vector $\mathbf{v}_j$ in order to estimate the feature weight $\hat{w}_{i,j}$. If the feature interaction feature $x_i x_j$ is not useful i.e. in practice, the estimate function $\hat{y}$ does not have the item $\hat{w}_{i,j} x_i x_j$, in such case, the estimation for parameter $\mathbf{v}_j$ and $\mathbf{v}_j$ will be affected. In order to effectively select "good" interaction features, we propose our Gradient Boosting Factorization Machines model.

3.3 Gradient Boosting Factorization Machines

3.3.1 GBFM Training

In this section, we present our proposed GBFM training algorithm. To relieve the problem in FM discussed in previous section, we borrow the idea of boosting methods [10] to select one interaction feature at each step and additively optimize the target function. The rating prediction function in our model is defined as:

$$\hat{y}_s(\mathbf{x}) := \hat{y}_{s-1}(\mathbf{x}) + \sum_{i \in \mathcal{C}_p} \sum_{j \in \mathcal{C}_q} \mathbb{I}[i, j \in \mathbf{x}] \langle \mathbf{V}_p^i, \mathbf{V}_q^j \rangle, \tag{13}$$

where s is the iteration step of the learning algorithm. At step s, we greedily select two interaction features $\mathcal{C}_p$ and $\mathcal{C}_q$ where $\mathbb{I}$ is the indicator function, the value is 1 if the condition holds otherwise 0. The feature selection algorithm we will introduce later. $\mathbf{V}_p \in \mathbb{R}^{n_p \times k}$ and $\mathbf{V}_q \in \mathbb{R}^{n_q \times k}$ are the low rank matrices for feature $\mathcal{C}_p$ and $\mathcal{C}_q$, k is the low rank dimension. After interaction features $\mathcal{C}_p$ and $\mathcal{C}_q$ are selected, we estimate the parameters $\mathbf{V}_p$ and $\mathbf{V}_q$ at step s. For example, at step s we select the interaction between user and item, we need to learn the low rank latent matrices $\mathbf{U}$ and $\mathbf{V}$. The objective function for estimate $\mathbf{V}_p, \mathbf{V}_q$ is:

$$\arg\min_{\mathbf{V}_p, \mathbf{V}_q} \sum_{i=1}^{N} l(\hat{y}_s(\mathbf{x}_i), y_i) + \|\mathbf{V}_p\|_F^2 + \|\mathbf{V}_q\|_F^2. \tag{14}$$

Assume we totally have S steps, we denote $\mathcal{C}_{sp}$ and $\mathcal{C}_{sq}$ as interaction feature selected at step s, then the final prediction function is:

$$\hat{y}_S(\mathbf{x}) = \hat{y}_0(\mathbf{x}) + \sum_{s=1}^{S} \sum_{i \in \mathcal{C}_{sp}} \sum_{j \in \mathcal{C}_{sq}} \mathbb{I}[i, j \in \mathbf{x}] \langle \mathbf{V}_{sp}^i, \mathbf{V}_{sq}^j \rangle, \tag{15}$$

where $\hat{y}_0(x)$ is the initialized prediction function.

The details of the algorithm are shown in Algorithm 1.

Algorithm 1 Gradient Boosting Factorization Machines Model

```
Input: Training Data $\mathcal{S} = \{\mathbf{x}_i, y_i\}_{i=1}^N$
Output: $\hat{y}_S(x) = \hat{y}_0(x) + \sum_{s=1}^S \langle \mathbf{v}_{si}, \mathbf{v}_{sj} \rangle$
Initialize rating prediction function as $\hat{y}_0(x)$
for $s = 1 \to S$ do
    Select interaction feature $C_p$ and $C_q$ from Greedy Feature Selection Algorithm
    Estimate latent feature matrices $\mathbf{V}_p$ and $\mathbf{V}_q$
    Update $\hat{y}_s(\mathbf{x}) := \hat{y}_{s-1}(\mathbf{x}) + \sum_{i \in \mathcal{C}_p} \sum_{j \in \mathcal{C}_q} \mathbb{I}[i, j \in \mathbf{x}] \langle \mathbf{V}_p^i, \mathbf{V}_q^j \rangle$
end for
```

3.3.2 Greedy Feature Selection Algorithm

In this section, we show how to effectively select "good" interaction features at each step which is the core part of our model. From the view of gradient boosting machine, at each step s, we would like to search a function f in the function space $\mathcal{F}$ that minimize the objective function:

$$\mathcal{L} = \sum_{i=1}^{N} l(\hat{y}_s(\mathbf{x}_i), y_i) + \Omega(f), \tag{16}$$

where $\hat{y}_s(\mathbf{x}) = \hat{y}_{s-1}(\mathbf{x}) + \alpha_s f_s(\mathbf{x})$. In our GBFM, the function f is set to factorized feature interactions like in FM. However, it is impossible to search all feature interactions to find the best (i.e. decrease the objective function most) due to high computation complexity. In order to find the desirable interaction features, we propose a greedy layer-wised algorithm to find the n-way interaction features. Our idea is as follows: there are n layers in total, at each layer, we greedily select the feature $\mathcal{C}_i$ that makes the objective function decrease fastest. At the end, we will get the n-way interaction feature. we heuristically assume that the function f has the following form:

$$f_l(\mathbf{x}) = \prod_{t=1}^{l} q_{\mathcal{C}_{i(t)}}(\mathbf{x}), \tag{17}$$

where $q_{\mathcal{C}_{i(t)}}(\mathbf{x})$ is the function learned at layer t with $i(t)$-th context feature selected. The function q maps latent feature vector $\mathbf{x}$ to real value domain. There are d elements in feature set, for each element we assign a weight w_{tj} to it. The function $q_{\mathcal{C}_{i(t)}}(\mathbf{x})$ is defined as:

$$q_{\mathcal{C}_{i(t)}}(\mathbf{x}) = \sum_{j \in \mathcal{C}_{i(t)}} \mathbb{I}[j \in \mathbf{x}] \cdot w_{tj}, \tag{18}$$

where $\mathbb{I}$ is the indicator function. Although the q looks very complex, in fact, in each instance there is only one non-zero element corresponding to feature $\mathcal{C}_{i(t)}$, the function value just takes the weight corresponding to the non-zero element. Take the instances in Fig. 1(c) for example again, suppose we select feature $\mathcal{C}_1$ at layer t, then the q function for first tuple is w_{t1}.

Searching function f to optimize the objective function in Eq. 16 can be hard for a general convex loss function l. The most common way is to approximate it by least-square minimization [11]. We denote the negative first derivative

[3] In KDDCup 2012 Task 1 we can easily extract around 20 features, there are more features in real industry world.

and the second derivative at instance i as g_i and h_i:

$$g_i = -\frac{\partial l(\hat{y}_s(x_i), y_i)}{\partial \hat{y}_s(x_i)}|_{\hat{y}_s(x_i)=\hat{y}_{s-1}(x_i)} \tag{19}$$

$$h_i = \frac{\partial^2 l(\hat{y}_s(x_i), y_i)}{\partial \hat{y}_s(x_i)^2}|_{\hat{y}_s(x_i)=\hat{y}_{s-1}(x_i)}. \tag{20}$$

The first part of Eq. 16 can be approximate as:

$$\begin{aligned} \mathcal{L} &= \sum_{i=1}^{N} l(\hat{y}_{s-1}(\mathbf{x}_i) + \alpha_s f_s(\mathbf{x_i}), y_i) \\ &\approx \sum_{i=1}^{N} l(\hat{y}_{s-1}(\mathbf{x}_i), y_i) - g_i(\alpha_s f_s(\mathbf{x}_i)) + \frac{1}{2} h_i(\alpha_s f_s(\mathbf{x}_i))^2 \end{aligned} \tag{21}$$

The Eq. 21 is equivalent to:

$$\mathcal{L}(f) = \sum_{i=1}^{N} h_i(g_i/h_i - f_s(\mathbf{x}_i))^2 + \Omega(f_s) \tag{22}$$

Replace the f_s function with the heuristic f defined in Eq. 18, we get the objective function for selecting n-way interaction features. Even using heuristic functions, finding the best interaction feature is still impossible. Instead, we learn the function f_l layer by layer. At layer t, we assume that the function $q_{\mathcal{C}_{i(1)}}, \ldots, q_{\mathcal{C}_{i(t-1)}}$ have been learned, i.e. f_{t-1} has been learned. Suppose at layer t we select $i(t)$-th feature, then we have:

$$f_t(\mathbf{x}) = f_{t-1}(\mathbf{x}) \cdot q_{\mathcal{C}_{i(t)}}(\mathbf{x}). \tag{23}$$

Our problem is finalized to find the $i(t)$-th feature which:

$$\underset{i(t)\in\{1,\ldots,m\}}{\arg\min} \sum_{i=1}^{N} h_i(\frac{g_i}{h_i} - f_{t-1}(\mathbf{x}_i) \cdot q_{\mathcal{C}_{i(t)}}(\mathbf{x}_i))^2 + \lambda \sum_{\theta\in\Theta} \theta^2, \tag{24}$$

where here we use L2-regularization to control the model complexity. To obtain the feature $i(t)$ minimize Eq. 24, we calculate the function q for all features. Without loss of generality, we assume the selected feature at layer t is $\mathcal{C}_{i(t)}$. The problem is actully transformed to estimating the weight for each $n_{i(t)}$ item in feature $\mathcal{C}_{i(t)}$. For a certain element j in $\mathcal{C}_{i(t)}$, we denote its corresponding weight as w_{ij}. The solution for w_{ij} is:

$$w_{ij} = \underset{w}{\arg\min} \sum_{i=1}^{N} h_i(g_i/h_i - f_{t-1}(\mathbf{x}_i) \cdot \mathbb{I}(j \in \mathbf{x}_i) \cdot w)^2 + \lambda w^2 \tag{25}$$

We denote $z_i = g_i/h_i$ and let

$$\begin{aligned} a &= \sum_{i=1}^{N} \sum_{j=1}^{d} \mathbb{I}(\mathbf{x}_j == \mathbf{z}_j) z_i h_i f_{t-1}(\mathbf{x}_i) \\ b &= \sum_{i=1}^{N} \sum_{j=1}^{d} \mathbb{I}(\mathbf{x}_j == \mathbf{z}_j) h_i (f_{t-1}(\mathbf{x}_i))^2 \end{aligned} \tag{26}$$

Then the solution for w_{ij} is:

$$w_{ij} = \frac{a}{b + \lambda}. \tag{27}$$

Note that although we need to calculate q function for all features, we can compute a, b for all features at the same time by scanning the training data just once. After we get the q function for all features, it is easy to select the best feature which satisfies Eq. 24. We repeat this process at each layer, at the end we can obtain the heuristic n-way interaction feature. Like FM, in our method, we consider 2-way interaction feature only. The details of the algorithm are shown in Algorithm 2.

Algorithm 2 Greedy Feature Selection Algorithm

```
Input: Training Data $\mathcal{S} = \{\mathbf{x}_i, y_i\}_{i=1}^{N}$,context feature set $\mathcal{C}$
Output: $n$-way interaction feature $\mathcal{C}_{i(1)}, \ldots, \mathcal{C}_{i(n)}$.
for $l = 1 \to n$ do
    $\mathcal{A} = \emptyset$ // $\mathcal{A}$ is the set of context features already selected
    Maintain two vectors a and b for all categorical values in $\mathcal{C}$, both initialized to 0
    for $(\mathbf{x}_i, y_i)$ in $\mathcal{S}$ do
        compute tempa = $z_i h_i f_{t-1}(\mathbf{x}_i)$ and tempb = $h_i(f_{t-1}(\mathbf{x}_i))^2$
        for $j = 1 \to d$ do
            if $\mathbf{x}_{ij}$ is non-zero and not in $\mathcal{A}$ then
                add tempa to $\mathbf{a}_j$ and tempb to $\mathbf{b}_j$
            end if
        end for
    end for
    Compute weight for all categorical features in $\mathcal{C} - \mathcal{A}$ according to Eq. 25.
    Select the feature $\mathcal{C}_{i(l)}$ according to Eq. 24.
    Add feature $\mathcal{C}_{i(l)}$ into $\mathcal{A}$
end for
```

3.3.3 *Complexity Analysis*

The computation complexity for *Greedy Feature Selection Algorithm* is $\mathcal{O}(n \cdot N)$, where N is the training data size, n is the number of layers. At each layer, the q function can be computed though scanning the training dataset once as described in Algorithm 2. Then best feature selection according to Eq. 24 can also be carried out by the training dataset once. Usually, $n \ll N$, in 2-way FM, $n = 2$. So the computation cost for Algorithm 2 is $\mathcal{O}(N)$.

In Algorithm 1, the estimation for $\mathbf{V}_p$ and $\mathbf{V}_q$ is usually carried out by stochastic gradient descent (SGD). The complexity for this part is $\mathcal{O}(kN)$, where k is number of iterations. In total, the complexity for GBFM is $\mathcal{O}(SN + kSN)$, S is number of boosting steps as stated in the algorithm, the computation complexity is still linear to the number of training dataset.

In addition, GBFM can be speedup by multi-threading and parallelization. The computation of first and second derivative can be decoupled thus can be easily computed though multi-threading and distributed to a cluster of computers. The gradient of Eq. 14 also can be decoupled and parallelization is possible for Algorithm 1.

3.3.4 *Discussions*

We discuss the insights of our heuristic function f in Eq. 17 which is the key part of Algorithm 2 as well as the relationship between the proposed GBFM and other state-of-the-art methods. At last, we discuss some variants of our model.

Insights of heuristic function f: The main idea of our algorithm is that at each layer we greedily select a context feature $\mathcal{C}_i$ according to Eq. 24 and we compute the corresponding weight vector, e.g. $q_{\mathcal{C}_i}$. We can regard it as the low rank latent feature matrix for feature $\mathcal{C}_i$ like in FM with latent dimension $k = 1$. Then the heuristic function f is an intance of CANDECOMP/PARAFAC (CP) decomposition

[15] with $k = 1$. We greedily use this f function to choose the interaction feature. In practice, for large dataset in industry world, we can additively use this function f as the "weak learner" instead of $\langle \mathbf{V}_p, \mathbf{V}_q \rangle$, to quickly find useful interaction features since the computation cost is relatively low.

Relation to Factorization Machines: Factorization Machines is a strong baseline method for context-aware recommendation [25]. The main difference between our model and FM is that FM models all interactions between context features while our method only consider part of them. For example, in Fig. 1(c), we have 3 context features, the rating prediction function of FM is:

$$\begin{aligned} \hat{y}(\mathbf{x}(u,i,c_3)) &= w_0 + w_u + w_i + w_{c_3} + \langle \mathbf{v}_u, \mathbf{v}_i \rangle \\ &+ \langle \mathbf{v}_u, \mathbf{v}_{c_3} \rangle + \langle \mathbf{v}_i, \mathbf{v}_{c_3} \rangle. \end{aligned} \tag{28}$$

While in our GBFM, we may only consider the (user,item) and (user,mood) interaction pair. If the interaction feature (item,mood) is actually not useful, then the term $\langle \mathbf{v}_i, \mathbf{v}_{c_3} \rangle$ which is the weight for feature $\mathbf{x}_i \mathbf{x}_{c_3}$ will introduce noise for the prediction function. Another difference is that in our algorithm we additively learn the latent feature matrices which are not shared to compute other factorization weights. For example, in first step, we select (user,item) pair, then the second step we select (user,mood) pair, the latent feature matrix $\mathbf{V}_u$ is not the same. It may lose the advantage of generalization compared to FM, we can regard our GBFM as a feature selection algorithm and only model the interaction on selected features.

Relation to GBMF: Gradient Boosting Matrix Factorization [6] is the state-of-the-art model which is a general functional matrix factorization using gradient boosting. GBMF is under the framework of matrix factorization [27]. They assume that the user/item latent low rank matrix is functional, each time a function f is added to latent dimension $\mathbf{U}_k$. While our model is under the framework of factorization machines, we use gradient boosting to greedily select "good" interaction features. Another difference is the construction method of high-order categorical features. In our algorithm, we can efficiently find the "best" features according to Algorithm 2, while the binary splitting tree algorithm may fail for categorical features since the cost for finding the best binary split is exponential.

Variants of our GBFM: There are several variants of our proposed GBFM. In our paper, we only use the 2-way interaction feature like in FM. It can be easily extended to n-way FM by selecting n-way interaction feature. Since our model is an additive model, we can first consider linear features, i.e. 1-way feature, then 2-way interaction and more high order features. Another variant is that we can fully optimize the selected interaction features instead of additively optimize interaction features one by one, we refer this variant as GBFM-Opt. The difference between GBFM-Opt and FM is that GBFM-Opt only consider some "good" selected 2-way interaction features.

4. EXPERIMENTS

In this section, we empirically investigate whether our proposed GBFM can achieve better performance compared to other state-of-the-art methods with large number of context features. Furthermore we would like to examine whether the interaction features selected by our algorithm is more effective compared to pairwise interactions in FM.

4.1 Datasets

We conduct our experiments on two dataset: a synthetic dataset and a real world dataset, i.e., the Tencent Microblog[4].

Synthetic data: Since there are few public datasets that have many context features. We construct a synthetic dataset for comparison. The data generation process is as follows: assume we have m context features, each context feature $\mathcal{C}_i$ have n_i values, we generate the latent context features from zero-mean spherical Gaussian as follows:

$$\mathbf{V}_i^j \sim \mathcal{N}(\mathbf{0}_K, \sigma^2 \mathbf{I}_K),$$

where $j = 1, \ldots, n_i$, $\mathbf{0}_K$ is a K-dimension vector with all elements set to 0, and $\mathbf{I}_K$ is the $K \times K$ identity matrix. We also generate the weight vector all categorical feature values $\mathbf{w} \sim \mathcal{N}(\mathbf{0}_{d+1}, \sigma^2 \mathbf{I}_{d+1})$, where $d = \sum_{i=1}^m n_i$, we incorporate the global bias into the weight vector. Then we select several 2-way interaction features. We denote the interaction feature set as $\mathcal{F}$. Then the rating is obtained by rescale the sigmoid value to 1 to D by:

$$\begin{aligned} \hat{y}(\mathbf{x}) &= \sum_{i=0}^{d} w_i x_i + \sum_{(p_1,p_2) \in \mathcal{F}} \sum_{i \in \mathcal{C}_{p_1}} \sum_{j \in \mathcal{C}_{p_2}} \mathbb{I}[i,j \in \mathbf{x}] \langle \mathbf{V}_{p_1}^i, \mathbf{V}_{p_2}^j \rangle \\ \hat{y}(\mathbf{x}) &= \lceil g(\hat{y}) \times D \rceil, \end{aligned}$$

where D is the rating scale, $g(x) = 1/(1 + \exp(-x))$. In our experiment, we set number of context features $m = 10$, latent dimension $K = 5$, rating scale $D = 5$, feature value size $n_i = 1000$.

Table 1: Statistics of datasets

Dataset	# Users	#Items	#Observed Entries
Synethic data	1000	1000	16270
Tencent microblog	2.3 M	6095	73 M

Tencent microblog dataset: Tencent microblog is one of the largest social media services in China like Sina Weibo and Twitter. The dataset is designed for KDDCup 2012 competition and it contains the celebrity recommendation records of about 2.3 million users over a time period of about two months. In this dataset, the celebrities are regarded as items for recommendation. The system recommends a celebrity to a user at a certain time and the user's response is either "accept" or "reject". The dataset contains rich context information such as user's age, gender, item's category, time information, etc.. We can also extract the session information such as the number of recommendation records before current recommendation. The dataset splits into training and testing data by time. The test data furthermore splits into public and private set for independent evaluations. The dataset is extremely sparse with only about two positive records (e.g. accept the recommendation) for each user. Besides, nearly 70% of users in the test dataset are never occurred in the training data.

Table 1 shows the statistic for both our synthetic data and real data.

[4] http://kddcup2012.org/c/kddcup2012-track1/data

4.2 Setup and Metrics

We randomly remove 20% of dataset as testing data, and the remaining 80% data as the training for the synthetic data. We repeat the experiment 5 times and report the average results. For Tencent Microblog data, the dataset is already splitted into training and testing set. We further use 1/5 training data as validation data to tune parameters and we conduct the evaluation on pubic test dataset. We extract 18 features from the data, including user, item, tweets number, follower/followee number, tweet time etc. We treat all of the features as categorical features.

For synthetic dataset, we use two metrics, the Mean Absolute Error (MAE) and Root Mean Square Error (RMSE), to measure the prediction quality of different methods. MAE and RMSE are defined as follows:

$$MAE = \frac{\sum_i |\hat{y}(i) - y_i|}{N} \tag{29}$$

$$RMSE = \sqrt{\frac{\sum_i (\hat{y}(i) - y_i)^2}{N}}, \tag{30}$$

where N is the number of training instance.

For Tencent microblog data, MAP@k is used as the metric:

$$MAP@k = \sum_{i=1}^{N} ap@k_i / N, \tag{31}$$

where N is the number of users and $ap@k$ is the average precision at k for the user:

$$ap@k = \sum_{i=1}^{k} P(k)/(number\, of\, items\, clicked\, in\, m\, items) \tag{32}$$

where $P(k)$ is the precision at cut-off k in the item list.

4.3 Performance Comparsion

In our experiments, we compare the following methods:

- **PMF**: this method is well known in recommender systems and proposed in [27]. It only uses user-item matrix for recommendation.
- **Context-aware FM**: this method is proposed in [25]. It is a strong baseline method we introduced in Section 3.2.
- **GBFM**: this method is our newly proposed model which is described in Algorithm 1 and Algorithm 2.
- **GBFM-Opt**: this method is a variant of our GBFM which we discussed in Section 3.3.4. After S steps when the training process stops, we will obtain S interaction features. We fully optimize these S interaction features.

For GBFM, we use 1-way feature linear model as the initialized prediction function. Grid search is applied to find regularization parameter λ, and we set it to 0.1 for synthetic data and 0.8 for Tencent microblog data. The latent dimension k is set to 5 and 10 for synthetic data and Tencent microblog data respectively. We use square loss to train synthetic data and logit loss for Tencent microblog data. The detailed comparison results are shown in Table 2 and Table 3.

From the table, we can observe that:

Table 2: Results on Synthetic data in RMSE and MAE

Method	RMSE	MAE
PMF	1.9881	1.7650
FM	1.9216	1.6981
GBFM	1.8959	1.6354
GBFM-Opt	**1.8611**	**1.5762**

Table 3: Results on Tencent Microblog data in MAP

Method	MAP@1	MAP@3	MAP@5
PMF	22.88%	34.50%	37.95%
FM	24.36%	36.77%	40.32%
GBFM	24.62%	37.17%	40.90%
GBFM-Opt	**24.66%**	**37.23%**	**40.98%**

- Both our proposed GBFM and GBFM-Opt model achieve better performance on both synthetic data and Tencent microblog data in terms of all metrics compared with PMF and FM. On synthetic dataset, FM gives 0.066 reduction over PMF in terms of RMSE, and GBFM further gives 0.026 reduction over FM. Since the synthetic data is generated from part of 2-way feature interactions, the results reveal that our proposed GBFM can learn “good” interaction features. While on the Tencent microblog data, FM improves 2.25% in terms of MAP@3 compared to PMF. GBFM can still be able to improve the performance by 0.4%. This result verifies our assumption that selecting “good” features is better than considering all of pairwise interaction features.
- It is not surprising that the performance of FM, GBFM and GBFM-Opt is much better than PMF. It reveals the importance of utilizing auxiliary information on context-aware recommendation. It is even more critical on Tencent microblog data since most of users in test dataset do not exist in the training data which means PMF cannot deal with them at all.
- The performance of GBFM-Opt can illustrate whether the selected interaction features are useful for recommendation. We can observe that on both datasets GBFM-Opt can achieve even better performance than GBFM. On synthetic dataset, GBFM-Opt improves a lot (0.035) compared to GBFM in terms of RMSE while on Tencent microblog data GBFM-Opt is slightly better than GBFM. The results reveal that the features selected by our GBFM is quite useful compared to consider all the pairwise interactions. Further, recall the discussion we conducted in Section 3.3.4, compared to GBFM-Opt, GBFM loses the advantage of generalization, which may be the main reason why GBFM-Opt is better than GBFM. Compared to synthetic data, the Tencent microblog data is much sparser thus it is not easy to benefit from generalization which explains why GBFM-Opt only improves a little compared to GBFM.

5. CONCLUSION

In this paper, we have proposed a novel model called GBFM which incorporates feature interaction selection algorithm with Factorization Machines into a unified framework to solve context-aware recommendation problems. Experiments on both synthetic and real datasets show that our model can effectively select "good" interaction features and achieve better performance compared to other state-of-the-art methods.

There are several interesting directions worthy of considering in the further study: 1) we would like to explore how to find high order features, 2) we are interested to extend our GBFM with better high order feature selection algorithm. 3) it is also interesting to explore how to effectively deal with other features apart from categorical features.

6. ACKNOWLEDGEMENTS

The work described in this paper was fully supported by the National Grand Fundamental Research 973 Program of China (No. 2014CB340405 and No. 2014CB340401), the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. CUHK 413212 and CUHK 415113), and Microsoft Research Asia Regional Seed Fund in Big Data Research (Grant No. FY13-RES-SPONSOR-036).

7. REFERENCES

[1] G. Adomavicius, R. Sankaranarayanan, S. Sen, and A. Tuzhilin. Incorporating contextual information in recommender systems using a multidimensional approach. pages 103–145, 2005.

[2] D. Agarwal and B.-C. Chen. Regression-based latent factor models. In *KDD*, pages 19–28, 2009.

[3] L. Baltrunas, B. Ludwig, and F. Ricci. Matrix factorization techniques for context aware recommendation. In *RecSys*, pages 301–304, 2011.

[4] R. M. Bell and Y. Koren. Lessons from the netflix prize challenge. *SIGKDD Explorations*, 9(2):75–79, 2007.

[5] C. J. Burges. From ranknet to lambdarank to lambdamart: An overview. *Learning*, 11:23–581, 2010.

[6] T. Chen, H. Li, Q. Yang, and Y. Yu. General functional matrix factorization using gradient boosting. In *Proceedings of The 30th International Conference on Machine Learning*, pages 436–444, 2013.

[7] T. Chen, L. Tang, Q. Liu, D. Yang, S. Xie, X. Cao, C. Wu, E. Yao, Z. Liu, Z. Jiang, et al. Combining factorization model and additive forest for collaborative followee recommendation. *KDD CUP*, 2012.

[8] C. Cheng, H. Yang, I. King, and M. R. Lyu. Fused matrix factorization with geographical and social influence in location-based social networks. In *AAAI*, 2012.

[9] C. Cheng, H. Yang, M. R. Lyu, and I. King. Where you like to go next: Successive point-of-interest recommendation. In *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, pages 2605–2611. AAAI Press, 2013.

[10] J. Friedman, T. Hastie, and R. Tibshirani. Special invited paper. additive logistic regression: A statistical view of boosting. *Annals of statistics*, pages 337–374, 2000.

[11] J. H. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, pages 1189–1232, 2001.

[12] T. Hastie, R. Tibshirani, J. Friedman, T. Hastie, J. Friedman, and R. Tibshirani. *The elements of statistical learning*, volume 2. Springer, 2009.

[13] M. Jamali and M. Ester. A matrix factorization technique with trust propagation for recommendation in social networks. In *RecSys*, pages 135–142, 2010.

[14] A. Karatzoglou, X. Amatriain, L. Baltrunas, and N. Oliver. Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering. In *RecSys*, pages 79–86, 2010.

[15] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009.

[16] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD*, pages 426–434, 2008.

[17] Y. Koren. Collaborative filtering with temporal dynamics. In *KDD*, pages 447–456, 2009.

[18] N. D. Lawrence and R. Urtasun. Non-linear matrix factorization with gaussian processes. In *ICML*, page 76, 2009.

[19] X. Liu and K. Aberer. Soco: a social network aided context-aware recommender system. In *WWW*, pages 781–802, 2013.

[20] H. Ma, I. King, and M. R. Lyu. Learning to recommend with explicit and implicit social relations. *ACM TIST*, 2(3):29, 2011.

[21] U. Panniello, A. Tuzhilin, M. Gorgoglione, C. Palmisano, and A. Pedone. Experimental comparison of pre- vs. post-filtering approaches in context-aware recommender systems. In *RecSys*, pages 265–268, 2009.

[22] S. Rendle. Factorization machines. In *ICDM*, pages 995–1000, 2010.

[23] S. Rendle. Social network and click-through prediction with factorization machines. In *KDD-Cup Workshop*, 2012.

[24] S. Rendle. Scaling factorization machines to relational data. *PVLDB*, 6(5):337–348, 2013.

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

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

[27] R. Salakhutdinov and A. Mnih. Probabilistic matrix factorization. In *NIPS*, 2007.

[28] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *WWW*, pages 285–295, 2001.

[29] N. Srebro, J. D. M. Rennie, and T. Jaakkola. Maximum-margin matrix factorization. In *NIPS*, 2004.

[30] D. H. Stern, R. Herbrich, and T. Graepel. Matchbox: large scale online bayesian recommendations. In *WWW*, pages 111–120, 2009.

[31] L. R. Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311, 1966.

[32] J. Weston, C. Wang, R. J. Weiss, and A. Berenzweig. Latent collaborative retrieval. In *ICML*, 2012.

[33] L. Xiong, X. Chen, T.-K. Huang, J. G. Schneider, and J. G. Carbonell. Temporal collaborative filtering with bayesian probabilistic tensor factorization. In *SDM*, pages 211–222, 2010.

[34] J. Xu and H. Li. Adarank: a boosting algorithm for information retrieval. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 391–398. ACM, 2007.

[35] M. Ye, P. Yin, W.-C. Lee, and D. L. Lee. Exploiting geographical influence for collaborative point-of-interest recommendation. In *SIGIR*, pages 325–334, 2011.

[36] E. Zhong, W. Fan, and Q. Y. 0001. Contextual collaborative filtering via hierarchical matrix factorization. In *SDM*, pages 744–755, 2012.