Sparse Factorization Machines for Click-through Rate Prediction

Zhen Pan[1], Enhong Chen[1,*], Qi Liu[1], Tong Xu[1], Haiping Ma[2], Hongjie Lin[1]
[1]School of Computer Science and Technology, University of Science and Technology of China
{pzhen, hjlin}@mail.ustc.edu.cn, {cheneh, qiliuql, tongxu}@ustc.edu.cn
[2] IFLYTEK Co.,Ltd., hpma@iflytek.com

***Abstract*—With the rapid development of E-commerce, recent years have witnessed the booming of online advertising industry, which raises extensive concerns of both academic and business circles. Among all the issues, the task of *Click-through rates* (CTR) prediction plays a central role, as it may influence the ranking and pricing of online ads. To deal with this task, the Factorization Machines (FM) model is designed for better revealing proper combinations of basic features. However, the sparsity of ads transaction data, i.e., a large proportion of *zero elements*, may severely disturb the performance of FM models. To address this problem, in this paper, we propose a novel *S*parse *F*actorization *M*achines (SFM) model, in which the *Laplace* distribution is introduced instead of traditional Gaussian distribution to model the parameters, as Laplace distribution could better fit the sparse data with higher ratio of zero elements. Along this line, it will be beneficial to select the most important features or conjunctions with the proposed SFM model. Furthermore, we develop a distributed implementation of our SFM model on Spark platform to support the prediction task on mass dataset in practice. Comprehensive experiments on two large-scale real-world datasets clearly validate both the effectiveness and efficiency of our SFM model compared with several state-of-the-art baselines, which also proves our assumption that Laplace distribution could be more suitable to describe the online ads transaction data.**

I. Introduction

In recent years, with the rapid development of E-commerce, online advertising industry has become the most popular and effective approach of promotion and marketing. As a multi-billion business mode, online ads could build novel connections between customers and merchants with lower cost and better coverage. In 2015, spending on online ads reached \$59.6 billion in the US, 20.4% over 2014. While producing opportunities, such an enormous market raises new challenges in the *real-time bidding* (RTB) scheme [32], [33], [25] which targets at designing the ads inventories, including allocation, ranking and pricing [28], [31], [19]. Thus, techniques on computational advertising are urgently required to support the decision-making and ensure the profit.

In computational advertising, the most crucial task is the price setting for each ad as it has a direct bearing on the profit. Usually, several compensation methods are utilized, such as Cost per Click (CPC), Cost per Action (CPA) or Cost per Mille (CPM). Among them, CPC, in which advertisers pay each time only a user clicks on the ad, is the most popular method as 66% of online advertising transactions are counted on a CPC basis[1]. Intuitively, if a user clicks the ad and then visits the profile of advertiser, there will be a deal more probably than just viewing the ad. Thus, CPC may accurately reveal conversion rate better than the other metrics. However, in order to arrange the pricing and ranking of each ad, expectation of clicking should be precisely estimated, i.e., the *click-through rate* (CTR) prediction task, which attracts extensive concern of both academic circles and business circles [16], [24].

In the literature, traditionally, as a simple but effective tool, Logistic regression (LR) is widely studied and utilized (e.g., [24], [4], [7], [10], [15]) to predict the CTR. However, this linear model heavily depends on manually selected features, while the latent correlations between features could hardly be revealed. It is a fatal flaw as combining features based on human expertise costs a lot, especially for large-scale datasets with high dimensional features, which may severely limit its application. Recently, some other prior arts based on neural network (NN) are proposed, like [3] and [13], in which various elements could be well summarized and complex interactions could be automatically obtained. However, the NN-methods suffer heavy burden of computation, and it is always difficult to achieve the global optimization. For CTR prediction, the *Factorization Machines* (FM) [21] model is also adopted based on feature engineering and matrix design. As a successful solution [20], [27], FM could comprehensively model the correlations between variables with the basic assumption that the first and second order parameters follow Gaussian distribution. Furthermore, to better summarize the features, *Bayesian Factorization Machines* are designed by imposing Gaussian hyper-priors for mean of each parameter, and Gamma hyper-priors for prior precision [6], [22].

However, there are some unique characteristics of ad transaction data, which impairs the performance of FM models. First, ad transaction data could be extremely sparse due to the common-used *one-hot encoding*, i.e., if N kinds of possible status exist for one feature, there will be a N-dimensional vector, which has only one *non-zero element* indicating the current status. Second, ad transaction data in real-world application could usually be large-scale. For instance, Table I lists statistics of 4 advertisers in iPinYou dataset [11], in which each record of impression is represented by features of 15,804 dimensions. We realize that for all the advertisers, they hold millions of impression records and billions of feature values.

*Corresponding Author.

[1]https://en.wikipedia.org/wiki/Online_advertising

TABLE I
SOME CHARACTERISTICS OF IPINYOU TRAINING DATASETS, WITH FEATURES OF 15804 DIMENSIONS.

Advertiser Key	Impressions	Non-zero Values	Non-zero Rate(%)
1458	3,083,056	69,550,137	0.1427
3358	1,742,104	37,915,970	0.1377
2821	1,322,561	15,106,292	0.1463
2997	312,437	5,383,633	0.1090

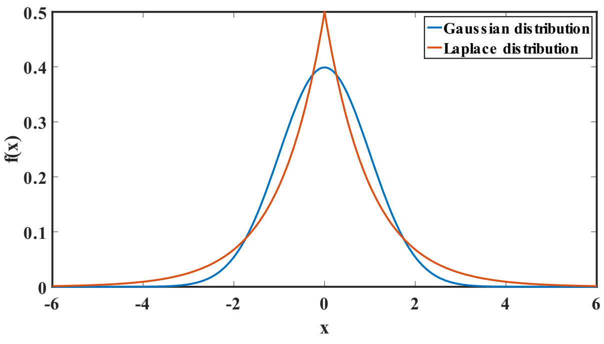


Fig. 1. The probability density function of Laplace distribution with ($\mu = 0, \rho = 1$) and Gaussian distribution with ($\mu = 0, \sigma = 1$). Specifically, the horizontal axis shows the possible value of variable x, while the vertical axis records the corresponding probability of x.

Unfortunately, the proportion of *non-zero elements* is even lower than *0.15%*. Clearly, mass data and extreme sparsity may significantly disturb the modeling, as Gaussian distribution could hardly fit the sparse data.

To address these problems, in this paper, we propose a novel **S**parse **F**actorization **M**achines (SFM) model, in which the *Laplace* distribution is introduced instead of traditional Gaussian distribution to model the parameters. As shown in Figure 1, Laplace distribution has a higher peak than Gaussian with similar parameters, which results in a higher probability of zero elements [18]. Thus, Laplace distribution could better describe the sparse data with few non-zero elements, and further, distinguish the relevant features or measure correlations between feature pairs. Along this line, to be specific, since Laplace distribution is non-smooth, the Bayesian inference of SFM will be analytically intractable. Thus, we formulate it as a scale-mixture of Gaussian distribution and exponential density, and then employ the Markov chain Monte Carlo method to perform the Bayesian inference. Furthermore, we develop a distributed implementation of our SFM model on Spark to support the prediction task on large-scale data in practice.

To the best of our knowledge, we are among the first ones who introduce the Laplace distribution into FM modeling, and further propose the distributed solution to ensure the applicability. Comprehensive experiments on two large-scale real-world datasets clearly validate both the effectiveness and efficiency of our SFM model compared with state-of-the-art baselines, including basic FM and extended Bayesian FM, which proves that our model could select the most relevant features or combinations, and also supports our assumption that Laplace distribution could be more suitable to describe the online ads transaction data.

II. RELATED WORK

As one of the most promising business models in big-data driven online advertising markets, RTB has attracted intensive research interests since its birth, and the major research issues include bidding behavior analysis and strategy optimization, market segmentation and ad performance analysis, and so on [32]. Actually, one of the main goals for analyzing and improving ad performance is to maximize user response rate for advertising campaigns, such as click through rates (CTR). Thus, a number of different models have been proposed for CTR prediction in both academia and industry [24], [21], [20], [27], [7], [30], [13]. Among them, Logistic regression (LR) is the most intuitive, analyzing and expanding model [24], where the probability that a user clicks on an ad is modeled as a logistic function of a linear combination of the features. Since LR model is highly interpretable and can be trained fast when data are large-scale, it is widely studied. For instance, [8] proposed to combine LR model and decision tree model. Specifically, this method took the output of decision trees as the input of LR model, and studied how the timeliness of training data impact on CTR accuracy. However, as a linear model, LR pays few attention to the different importance and interactions of basic features. In contrast, Factorization Machines (FM) model is a type of method that is able to model all interactions between variables (features) automatically and is a general predictor working with any real valued feature vector [21]. Therefore, both the basic FM and the extended versions of FM (e.g., Bayesian FM) have been adopted for CTR prediction [6], [20]. Recently, the authors in [27] also introduced an online learning algorithm for CTR prediction based on Factorization Machines.

Besides LR and FM, some other methods were also proposed recently. For instance, [7] presented a type of probabilistic regression model by Bayesian priors. [30] introduced a method that assigns user features and ad features with different parameters. Thus, this method can ignore the useless features of the users and ads, and the authors also built a distributed learning framework for training the model. [5] presented a combination of strategies, including a method for transfer learning and an update rule for learning rate.

As the volume of data increase, more complex methods based on neural network are proposed to predict CTR [3], [35], [13]. For instance, in [3], Artificial Neural Networks (ANN) was used for CTR prediction and performed better than LR model. As observed in the real-world sponsored search system, user's behaviors highly depend on how the user behaved along with the past time. Inspired by this observation, [35] introduced a novel framework based on Recurrent Neural Networks (RNN). In [13], Convolution Neural Networks (CNN) was applied to predict CTR, which could treat varied elements in a single ad impression as a whole and obtain complex interaction among them. However, the Neutral Network methods may get a local optimal solution easily and the time cost is high.

On the other hand, sparsity problem is usually one of the major issues that bother many data-driven studies, e.g., CTR prediction in this paper. Generally, when sparsity happens, the researchers try to effectively represent each object by the most informative latent features. For instance, [29] developed Regularized dual averaging (RDA) method, which was par-

TABLE II
SEVERAL IMPORTANT MATHEMATICAL NOTATIONS.

Notations	Description
$\mathbf{y}$	vector of the targets, where y_i represents the impression was clicked or not
$\hat{y}$	prediction of the models, a probability
$\mathbf{X}$	feature vectors, where $\mathbf{x}_i$ (or $\mathbf{x}$) is the i-th vector, x_{ij} is the j-dimension of $\mathbf{x}_i$
k	the dimension of latent vector
Θ	parameters, $\Theta = \{\theta\} = \{w_0, w_1, \cdots, w_p, v_{11}, \cdots, v_{pk}\}$
α	the precision for measuring $\hat{y}$
α_0, β_0	parameters of prior Gamma distribution
μ_θ	hyperparameter in BFM, the mean of θ
λ_θ	hyperparameter in BFM, the precision of θ
η_θ	the scale parameter of Laplace distribution
ϕ_θ	hyperparameter in SFM
a, b	parameters of prior inverse Gaussian distribution

ticularly effective in obtaining sparse solutions in the case of L_2-regularization. Indeed, prediction models with sparsity are usually obtained by employing sparsity-favoring distributions (that are highly peaked with heavy tails), e.g., Laplace distribution [18]. For instance, a variant of probabilistic matrix factorization method (PMF) was proposed in [9] which utilized a Laplace distribution to model the item/user factor vector, and similarly, a sparse (Laplace) covariance prior is adopted in [26] to enforce the user and item factors and make each latent feature reflect the semantics more properly.

However, to the best of our knowledge, none of existing studies focus on addressing the challenge of sparsity in C-TR prediction by introducing Laplace priors to improve the performance of factorization machines.

III. SPARSE FACTORIZATION MACHINES

In this section, we first briefly review FM and Bayesian FM. Then, we will introduce our sparse factorization machines model for the task of CTR prediction, namely SFM, and the inference process. Finally, we show a distributed implementation on Spark based on the Gibbs sampling technique [1]. For better illustration, Table II summarizes some math notations.

A. Preliminary

1) Factorization Machines: Factorization Machines (FM) are a model class that combines the advantages of Support Vector Machines (SVM) with factorization models [21], [12], [14]. Figure 2(a) shows the graphical model of FM using the plate convention, where shaded and unshaded variables indicate observed targets and latent variables respectively. Specifically, the model equation for a factorization machine is defined as:

$$\hat{y}(\mathbf{x}) := w_0 + \sum_{j=1}^{p} w_j x_j + \sum_{j=1}^{p} \sum_{l=j+1}^{p} \langle \mathbf{v}_j, \mathbf{v}_l \rangle x_j x_l, \qquad (1)$$

where $\mathbf{x}$ is the feature vector of one impression (exposure), x_j donates the j-th dimension of the feature vector, $\hat{y}$ represents the prediction value of FM model, i.e., the estimated probability of click. w_0 is the global bias, w_j models the importance of the j-th variable and $\mathbf{v}_j$ is a k-dimension latent vector of the j-th variable. $\langle \cdot, \cdot \rangle$ is the dot product of two vectors of size k, that models the interaction between the j-th and l-th variable. Here, k is the dimension of the latent vectors, which controls the complexity of FM.

The pairwise interactions can be written as:

$$\begin{aligned} &\sum_{j=1}^{p} \sum_{l=j+1}^{p} \langle \mathbf{v}_j, \mathbf{v}_l \rangle x_j x_l \\ =&\frac{1}{2} \sum_{f=1}^{k} \left[\left(\sum_{j=1}^{p} v_{j,f} x_j \right)^2 - \sum_{j=1}^{p} v_{j,f}^2 x_j^2 \right], \end{aligned} \qquad (2)$$

which makes FM can be computed in linear time $O(kp)$.

The FM model has an appealing $multilinearing$ trait (Actually, we also exploit this property during our inference of SFM). Specifically, for each model parameter $\theta \in \Theta$, the output of FM $y(\hat{\mathbf{x}})$ is a linear combination of two functions $g(x)$ and $h(x)$ [23]. The functions g and h are indexed with the name of the parameter θ because their form depends on the variable θ but they are independent of the value of θ.

$$\hat{y}(\mathbf{x}) = g_\theta(\mathbf{x}) + \theta h_\theta(\mathbf{x}), \quad \forall \theta \in \Theta, \qquad (3)$$

where $h_\theta(\mathbf{x})$ corresponds to the gradient term:

$$h_\theta(\mathbf{x}) = \frac{\partial \hat{y}(\mathbf{x})}{\partial \theta} = \begin{cases} 1 & if \quad \theta = w_0, \\ x_j & if \quad \theta = w_j, \\ x_j \sum_{l=1}^{p} v_{l,f} x_l - v_{j,f} x_j^2 & if \quad \theta = v_{j,f}. \end{cases} \qquad (4)$$

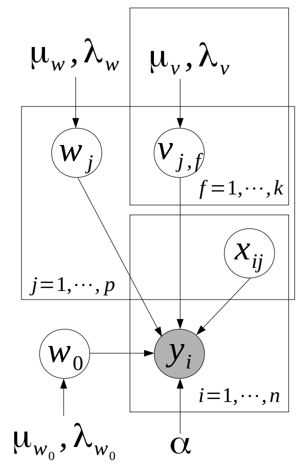
The sum $\sum_{l=1}^{p} v_{l,f} x_l$ is independent of j and thus can be precomputed for efficient learning.

In summary, FM subsumes matrix factorization which is representable by engineering the input features $\mathbf{x}$, and they are more complex than matrix factorization models as $\mathbf{x}$ may consist of real valued predictor variables. Also, they can efficiently handle relationships between these features. Therefore, FM can be used to address data with side information. However, restricting all features to the same regularization level limits the flexibility of the model [21].

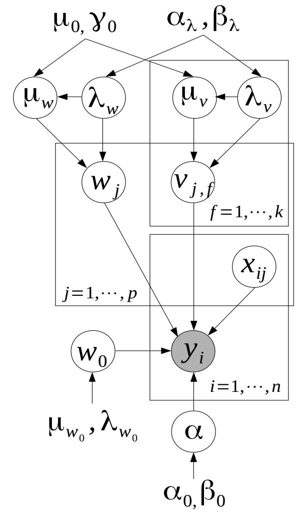
2) Bayesian Factorization Machines: To further improve the accuracy, automatic hyperparameter learning (no grid search) or no learning hyperparameter, e.g., learn rate for gradient descent, FM was enhanced with structured Bayesian inference (namely BFM) [6]. Specially, in Bayesian model, Gaussian hyper-priors are added to the standard regularized regression model (i.e., FM) for each mean μ_θ of all model parameters $\Theta = \{w_0, w_1, \cdots, w_n, v_{1,1}, \cdots, v_{n,k}\}$, as well as Gamma hyper-priors for each prior precision α, λ_θ, as shown in Figure 2(b). The standard L_2-regularized regression model with hyperparameters is for a sample of n observations:

$$p(\Theta|\mathbf{y}, \mathbf{X}, \Theta_H) \propto \prod_{i=1}^{n} \sqrt{\alpha} e^{-\frac{\alpha}{2}(y_i - y(\mathbf{x}_i, \Theta))^2} \prod_{\theta \in \Theta} \sqrt{\lambda_\theta} e^{-\frac{\lambda_\theta}{2}(\theta - \mu_\theta)^2}. \qquad (5)$$

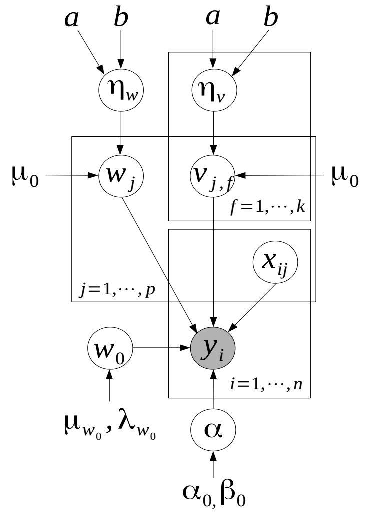
By maximizing the log-posterior of the model over both parameters and hyper parameters, BFM can automatically control the model complexity. From the Bayesian model, we can get a probability distribution of CTR rather than a fixed estimate value. Then, it is beneficial for the balance of exploration and exploitation. Furthermore, it helps us to control the speed of advertising. However, BFM assumes that the parameters follow Gaussian distribution, which may not be reasonable especially when the data are extremely sparse.



(a) FM



(b) BFM



(c) SFM

Fig. 2. Graphical models for FM, BFM and SFM, respectively.

B. SFM Model

To address sparsity, the sparsity-favoring distributions can be employed which prefer a high peak with heavy tails, e.g., Laplace distribution. Specifically, Laplace distribution is log-concave, leading to a posterior whose log density is a concave function and has a single local maximum, which is essential to design robust and easy-to-use algorithm. The probability density function (P.D.F.) of Laplace distribution is defined as:

$$\boldsymbol{L}(x|\mu,\rho) = \frac{1}{2\rho} exp(-\frac{|x-\mu|}{\rho}). \tag{6}$$

In Section I, Figure 1 shows the difference between the P.D.F.s of Laplace distribution and Gaussian distribution. We propose a novel Sparse Factorization Machines (SFM) model for CTR prediction, in which the Laplace distribution is introduced instead of traditional Gaussian distribution to model the parameters. The graphical representation is shown in Figure 2(c). In this subsection, we show the details of the parameter modeling and the overall generative process of SFM.

Parameters w **and** v: In order to effectively characterize each feature and the interaction, we try to select the most informative latent features to represent them. Thus, we employ Laplace distribution to model the first order parameter w with $Zero$-mean and η_w scale, and model the second order parameter v with $Zero$-mean and η_v scale.

$$p(\boldsymbol{w}|\eta_w) = \prod_{j=1}^{p} \boldsymbol{L}(w_j|0,\eta_w), \tag{7}$$

$$p(\boldsymbol{v}|\eta_v) = \prod_{j=1}^{p}\prod_{f=1}^{k} \boldsymbol{L}(v_{jf}|0,\eta_v). \tag{8}$$

Hyperparameters η_θ (e.g., η_w, η_v): One step further, we select the Inverse Gaussian (IG) distribution [17] to model the scales of Laplace distribution for obtaining a hierarchical Bayesian treatment of SFM and enhancing the model robustness. In [36], it has been demonstrated that Laplace mixture with IG is beneficial to define a regularization for variable selection, which is useful for SFM model to select the most informative latent features. That is,

$$\boldsymbol{IG}(a,b) = \sqrt{\frac{b}{2\pi\eta_\theta^3}} exp\left\{\frac{-b(\eta_\theta - a)^2}{2a^2\eta_\theta}\right\}, \tag{9}$$

where a is the mean and b is the scale parameter of IG distribution, respectively.

Precision α: In SFM, the conditional distribution of the observed value $\hat{y}$ is still i.i.d. normal distribution with mean $\mathbf{y}(\mathbf{X})$ and precision α. We model the precision of Gaussian distribution α by Gamma distribution with shape α_0 and scale β_0 as the same as BFM.

$$p(\alpha|\alpha_0,\beta_0) = \frac{1}{\beta_0^{\alpha_0}\Gamma(\alpha_0)} x^{(\alpha-1)} e^{-\frac{x}{\beta}}, x > 0. \tag{10}$$

In summary, according to the Bayesian theory, the posterior distribution over parameter $\theta \in \Theta = \{w_0, w_1, \cdots, w_n, v_{1,1}, \cdots, v_{n,k}\}$ can be modeled as:

$$\begin{aligned} p(\theta|\mathbf{y},\mathbf{X},\Theta\backslash\{\theta\},\alpha,\eta_\theta) =& \frac{p(\mathbf{y}|\mathbf{X},\Theta,\eta_\theta)p(\theta|\eta_\theta)}{p(\mathbf{y}|\alpha)} \\ \propto& p(\mathbf{y}|\mathbf{X},\Theta,\eta_\theta)p(\theta|\eta_\theta). \end{aligned} \tag{11}$$

Then, the overall generative process of SFM can be summarized as follows.

- Draw scales η_w and η_v from $\boldsymbol{IG}(a,b)$.
- Draw each first order parameter w from $\boldsymbol{L}(0,\eta_w)$ and each second order parameter v from $\boldsymbol{L}(0,\eta_v)$.
- Draw precision α from $\Gamma(\alpha,\beta)$.
- Draw each target value $\hat{y}$ from $\boldsymbol{N}(y(\mathbf{X}),\alpha)$.

C. Inference

In this subsection, we show the solution for "inverting" the generative process and "generating" latent variables (parameters) from given observations (e.g., y_i). Along this line, Markov chain Monte Carlo (MCMC)-based methods are widely applied to approximate the predictive distribution [9]. Its key idea is to construct a Markov chain that will evenly converge to the posterior distribution of the model with the given data, and each state of the Markov chain is used as a sample of the desired distribution. When the conditional distributions can be sampled easily, Gibbs sampling is the simplest but efficient MCMC algorithm, which samples each variable from its distribution conditionally on the current values of all other variables, i.e., it samples each variable by fixing all others. However, it is not easy to sample from a non-smooth Laplace distribution in SFM model. Fortunately, Laplace distribution can be equivalently expressed as a scaled mixture of Gaussians [2], i.e., an infinite Gaussian mixture with an exponential distribution like

$$\boldsymbol{L}(x|\mu,\rho)=\int_0^{\infty}\boldsymbol{N}(x|\mu,\epsilon)exp(\epsilon|\frac{\rho}{2})d\epsilon. \tag{12}$$

We can see that all priors on the parameters and hyperparameters in SFM are conjugate, and thus, we can develop an efficient Gibbs sampling algorithm based on Eq. (12) to infer SFM model. Next, we will introduce the proposed sampling process in detail.

Sample parameters θ: For θ, we extract all terms related to θ and use Bayes rule to obtain

$$\begin{aligned}p(\theta|\mathbf{y},\mathbf{X},\Theta\backslash\{\theta\},\alpha,\eta_\theta)&\propto p(\mathbf{y}|\mathbf{X},\Theta,\alpha)p(\theta|\eta_\theta)\\&=\prod_{i=1}^{n}\boldsymbol{N}(y_i|\hat{y}(\mathbf{X},\Theta),\alpha)\times\boldsymbol{L}(\theta|0,\eta_\theta).\end{aligned} \tag{13}$$

In order to incorporate the alternative expression of Laplace distribution, i.e., the infinite Gaussian distribution in Eq. (12), and motivated by [9], we introduce a latent variable ϕ, where each element ϕ_θ is a variable with exponential prior, i.e.,

$$p(\phi_\theta|\eta_\theta)=exp(\phi_\theta|\eta_\theta), \tag{14}$$

for the corresponding θ. We can further express Eq. (13) as

$$p(\theta|\mathbf{y},\mathbf{X},\Theta\backslash\{\theta\},\alpha,\phi_\theta)=\boldsymbol{N}(\theta|\mu_\theta^\star,\sigma_\theta^2), \tag{15}$$

where

$$\sigma_\theta^2=\left(\alpha\sum_{i=1}^{n}h_\theta(\mathbf{x}_i)^2+\phi_\theta^{-1}\right)^{-1}, \tag{16}$$

$$\mu_\theta^\star=\sigma_\theta^2\left(\sum_{i=1}^{n}\alpha(y_i-g_\theta(\mathbf{x}_i))h_\theta(\mathbf{x}_i)\right). \tag{17}$$

Sample hyperparameters η_θ and ϕ_θ: Since we introduce a latent variable ϕ for inferring θ, in the following we show how to sample each element ϕ_θ in ϕ. Specifically, for ϕ_θ, we have $p(\phi_\theta|\theta,\eta_\theta)\propto p(\theta|\phi_\theta,\eta_\theta)p(\phi_\theta)$ and each ϕ_θ has an exponential prior (Eq. (14)). According to the property of exponential distribution, ϕ_θ^{-1} follows an inverse Gaussian distribution [36]. Then, we can get the posterior probability of the hyperparameters ϕ_θ as

$$p(\phi_\theta^{-1}|\theta,\eta_\theta)=\boldsymbol{IG}\left(\frac{\sqrt{\eta_\theta}}{|\theta|},\eta_\theta\right). \tag{18}$$

For the scale of Laplace distribution, η_θ, it is modeled by an IG distribution (as shown in Eq. (9)), and thus, it can be computed by

$$p(\eta_\theta|\phi_\theta,a,b)=\boldsymbol{IG}\left(\sqrt{\frac{\phi_\theta+a}{b}},\phi_\theta+a\right). \tag{19}$$

Sample parameters α: At last, we show how to sample the precision α of the normal distribution for measuring $\hat{y}$. Based on Eq. (10), the posterior distribution of α by fixing other variables can be written as

$$\begin{aligned}&p(\alpha|\boldsymbol{y},\boldsymbol{X},\Theta)=\Gamma(\alpha^\star,\beta^\star),\\&\alpha^\star=\frac{n}{2}+\alpha_0,\\&\beta^\star=\frac{1}{2}\sum_{i=1}^{n}(y_i-(g_\theta(\mathbf{x}_i)+\theta h_\theta(\mathbf{x}_i)))^2+\beta_0.\end{aligned} \tag{20}$$

In summary, the Gibbs sampling algorithm for SFM takes the form of Algorithm 1.

Algorithm 1 Gibbs sampling for SFM

```
Input: targets $\mathbf{y}$, feature vectors $\mathbf{X}$, hyperparameters $\alpha_0, \beta_0, a, b$
Output: parameters $\Theta$
 1: Initialize model parameters $\Theta^1$ and hyperparameters $\eta_\theta^1$
 2: Sample hyperparameter $\alpha$ (Eq. (21)):
            $\alpha^t \sim p(\alpha|\mathbf{y},\mathbf{X},\Theta^t,\alpha_0,\beta_0)$;
 3: for $t = 1, \cdots, T$ do
 4:     for all $\theta \in \Theta$ do
 5:         Sample hyperparameter $\phi_\theta$ in parallel (Eq. (18)):
                $(\phi_\theta^{-1})^t \sim p(\phi_\theta^{-1}|\theta^t,\eta_\theta^t)$;
 6:         Sample hyperparameter $\eta_\theta$ in parallel (Eq. (19)):
                $\eta_\theta^{t+1} \sim p(\eta_\theta|\phi_\theta^t)$;
 7:     end for
 8:     Sample the parameters (Eq. (15)):
            $\Theta^{t+1} \sim p(\Theta|\mathbf{y},\mathbf{X},\Theta^t,\alpha^t,\phi_\theta^t)$;
 9: end for
10: return $\Theta^T$
```

D. Distributed Implementation On Spark

As the volume of advertising data is growing dramatically, Spark is an excellent tool to analyze this type of data and implement CTR prediction algorithm. In this subsection, we develop a distributed implementation of SFM on Spark.

The flowchart is shown in Figure 3. Specifically, we first broadcast the parameters collect Θ to all participating executors (there are C executors in total). Each executor compute the function $g_\theta(\mathbf{x})$ and $h_\theta(\mathbf{x})$. Then, we use the resilient distributed dataset (RDD) operates $flatMap$ and $partitionBy$ to shuffle the data. Next, the executors can sample the parameters and hyperparameters as described in Algorithm 1 and finally write back its result.

In this way, the computational complexity of inferring the parameter Θ and its corresponding hyperparameters in SFM is $O(T(kp\hat{N}))$, where T is the number of iterations, k is the number of latent features and $\hat{N}$ is the maximum number of samples in all executors (In other words, the more executors may not always lead to fewer time consuming). Thus, the computational complexity of our SFM algorithm is linearly scalable to the size of samples, so it is very practical for handling large-scale advertising data.

TABLE III
IPINYOU DATASET STATISTICS

Season	Advertiser Key	Training Dataset			Test Dataset		
		Impressions	Clicks	CTR(%)	Impressions	Clicks	CTR(%)
2	1458	3,083,056	2,454	0.080	614,638	543	0.088
2	3358	1,742,104	1,358	0.078	300,928	339	0.113
2	3386	2,847,802	2,076	0.073	542,421	496	0.091
2	3427	2,593,765	1,926	0.074	536,795	395	0.074
2	3476	1,970,360	1027	0.052	523,848	302	0.058
3	2259	835,556	280	0.034	417,179	131	0.031
3	2261	687,617	207	0.030	343,862	97	0.028
3	2821	1,322,561	843	0.064	661,964	394	0.060
3	2997	312,437	1386	0.444	153,063	533	0.348
Total	9	24,658,119	18,559	0.075	6,689,084	5,162	0.077

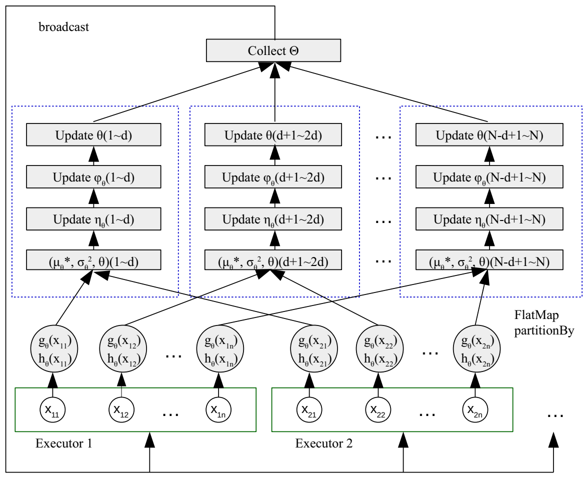

broadcast
Collect Θ
Update θ(1~d)
Update θ(d+1~2d)
Update θ(N-d+1~N)
Update φ_θ(1~d)
Update φ_θ(d+1~2d)
Update φ_θ(N-d+1~N)
Update η_θ(1~d)
Update η_θ(d+1~2d)
Update η_θ(N-d+1~N)
$(\mu_\theta^*, \sigma_\theta^2, \theta)(1\sim d)$
$(\mu_\theta^*, \sigma_\theta^2, \theta)(d+1\sim 2d)$
$(\mu_\theta^*, \sigma_\theta^2, \theta)(N-d+1\sim N)$
FlatMap
partitionBy
$g_\theta(x_{11})$ $h_\theta(x_{11})$
$g_\theta(x_{12})$ $h_\theta(x_{12})$
$g_\theta(x_{1n})$ $h_\theta(x_{1n})$
$g_\theta(x_{21})$ $h_\theta(x_{21})$
$g_\theta(x_{22})$ $h_\theta(x_{22})$
$g_\theta(x_{2n})$ $h_\theta(x_{2n})$
x_{11} x_{12} x_{1n}
x_{21} x_{22} x_{2n}
Executor 1
Executor 2

Fig. 3. Distributed implementation on Spark.

IV. EXPERIMENTS

In this section, we evaluate the performance of SFM on two large-scale real-world datasets from desktop display advertising and iOS platform advertising, respectively. Specifically, we demonstrate: (1) the effectiveness of SFM compared with state-of-the-art baselines; (2) the sensitivity of SFM with different parameters; (3) the comparison on different sparsity levels; (4) the speedup of distributed implementation.

A. Experimental Setup

Data Description. We use two real-world datasets.

1) IPinYou Dataset: IPinYou dataset is mainly a desktop display advertising dataset which is released by the DSP company iPinYou in 2014. It includes three season datasets of a global bidding algorithm competition [11]. Each season dataset contains impression, click, and conversion logs collected from several advertisers during various days and is previously divided into a training set and a test set. Because there is no advertiser ID column in season 1, we only use datasets of season 2 and season 3 for our experiments. For instance, the training datasets of season 2 contains historical bidding logs collected from 5 advertisers during the seven days from June 6th to 12th, and a dataset collected from the following three days from June 13th to 15th is used for offline testing purposes. The statistical information of this dataset is given in Table III, from which we can see that all CTRs in this desktop display advertising dataset are less than 0.1% except for advertiser 2997 (0.444%). It is the reason that advertiser 2997 is a mobile e-commerce app install related to the mobile environment, where an increased number of inadvertent clicks are easily generated by fat fingers due to the limited space of touch screens. At last, as described in [11], we know that each data record contains three types of information: user features (iPinYou ID, user profile, region, city, etc), ad features (creative ID, advertiser ID, landing page URL, domain, etc) and context features (timestamp, IP, user-agent, ad slot ID, slot width, slot height, etc). We extract the above three types of features with 15,804 dimensions by one-hot encoding.

2) IOS Dataset: IOS dataset contains bid and click data supplied by IFLYTEK Co.,Ltd., and it was collected from iOS platform between the period of 16-26 April and 9-16 May 2016. Similar to iPinYou, for this dataset we conduct 10 runs of experiments. For each run ($Trial$), we take 8 days and split the data by time into a training set (7 days) and a test set (1 day). The statistics of this dataset are shown in Table IV. Because IOS data is collected from mobile environments, the inadvertent clicks also exist and the CTRs are relatively high. We extract features of 4,550 dimensions from data in April and 5,462 dimensions in May, and the average non-zero rate of features is 0.48% in April and the sparsity is approximately between 0.38% and 0.40% in May.

Benchmark Methods and The Details of Training. In order to demonstrate the effectiveness of SFM, we compare it with two benchmark methods, FM and BFM [21], [6]. Actually, we can see that these two baselines are the related methods for SFM. It also should be noted that the widely used LR model is not listed as a baseline, since LR is a special case of FM when $k = 0$.

The FM models were trained to minimize the sum-squared error with L_2-norm regular terms, and the regularization parameters were tuned from the candidate set $\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1\}$. Following [6], we set $\alpha_0 = \beta_0 = \lambda_0 = \alpha_\lambda = \beta_\lambda = 1, \mu_0 = 0$ for BFM. In SFM, we set $\alpha_0 = \beta_0 = \eta_0 = \phi_0 = 1$. Meanwhile, we selected the mean of inverse Gaussian distribution a from the candidate set $\{1, 3, 10, 30, 100, 300, 1000, 3000, 10000\}$ and we set the scale parameter $b = 1/a$ respectively. Moreover, we randomly initialize the parameter set Θ by Gaussian distribution with $mean = 0$. We fix the dimension of latent vector $k = 8$ for FM, BFM and SFM. In practice, the parameters with the best

TABLE IV
IOS DATASET STATISTICS

Experiment	Training Dataset				Test Dataset			
	period	Impressions	Clicks	CTR(%)	period	Impressions	Clicks	CTR(%)
$Trial$ 1	11-17 April	5,046,406	57,207	1.134	18 April	1,219,926	15,229	1.248
$Trial$ 2	12-18 April	5,398,028	63,628	1.179	19 April	1,545,594	18,090	1.170
$Trial$ 3	13-19 April	6,181,866	72,186	1.168	20 April	1,862,805	37,459	2.011
$Trial$ 4	14-20 April	7,327,085	102,181	1.395	21 April	1,125,374	24,463	2.174
$Trial$ 5	15-21 April	8,051,988	121,148	1.505	22 April	1,148,904	25,262	2.199
$Trial$ 6	16-22 April	8,468,659	138,137	1.631	23 April	1,042,017	23,809	2.285
$Trial$ 7	17-23 April	8,675,763	153,552	1.770	24 April	1,025,736	25,952	2.530
$Trial$ 8	18-24 April	8,970,356	170,264	1.898	25 April	1,020,010	23,658	2.319
$Trial$ 9	19-25 April	8,770,440	178,693	2.037	26 April	1,179,900	24,042	2.038
$Trial$ 10	9-15 May	2,337,551	57,364	2.454	16 May	643,128	20,431	3.177

training performance are chosen for the model, and they are used as the default settings for performance comparison in testing.

Evaluation Metrics. A good model for CTR prediction should give good regression and classification results. Thus, we evaluate the prediction performances of each algorithm using three different metrics following [34], [20]. For regression, we use root mean square error (RMSE) and negative log likelihood (NLL). These two metrics can quantify the distance between predicted probabilities and the actual one. Specifically, for a test set with N instances:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{N}}, \tag{21}$$

$$NLL = -\sum_{i=1}^{N}(y_i \log \sigma(\hat{y}_i) + (1 - y_i)\log(1 - \sigma(\hat{y}_i))), \tag{22}$$

where y_i and $\hat{y}_i$ represent the target and predictions respectively, and σ is the sigmoid function.

At the same time, we treat the prediction problem as a binary classification task, where a click (no click) indicates a positive (negative) instance. To reflect the performance of classification result, we use the area under the receiver-operating characteristic curve (AUC).

B. Experimental Results

Performance Comparison. First, the performance results of all the models, evaluated using RMSE, NLL and AUC, are summarized in Table V. Generally, from the table it can be observed that SFM outperforms FM and BFM. Specifically, though SFM is slightly better, the results under RMSE and NLL (the smaller value the better performance) of all models are very similar. However, the AUC values (the larger value the better performance) achieved by SFM are much higher than those achieved by FM and SFM. For different data sets, our SFM model performs better than all the baselines significantly (with $P\text{-}value \ll 0.05$). As AUC is a more reasonable and straightforward metric for CTR prediction, these results suggest that the proposed SFM model can more effectively predict the click-through rate by modeling the sparsity of ad data with Laplace distribution.

Second, we compare the convergence process of BFM and SFM in terms of the number of iterations, as shown in Figure 4, with taking dataset 3476 from iPinYou and $Trial$ 1 from IOS as examples. From Figure 4, we can see that these two methods converge similarly, and SFM outperforms BFM after convergence which can be also observed in Table V. Since the main difference between SFM and BFM is the prior assumptions (Laplace distribution or Gaussian distribution) over the parameters, these observations in Figure 4 once again demonstrate the effectiveness of both Laplace distribution and SFM model in CTR prediction.

Sensitivity of Parameters. According to the setting of experiments, two parameters in SFM need to be determined, i.e., the dimension of latent vector k and the mean of inverse Gaussian distribution a.

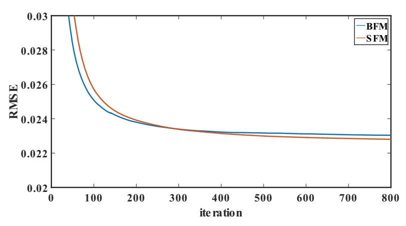
As shown in Figure 5, we evaluate the RMSE, NLL and AUC of SFM and BFM with different k. Due to the space limitation, we only select the $Trial$ 1 dataset from IOS for illustration. From the figure, we can see that the size of latent vectors k does affect the performance of the prediction models [20]. Note that, we fix $k = 8$ as the default setting for experimental evaluation. We also conduct an experiment to study the impact of hyperprior a in SFM. Parameter a is the mean of the inverse Gaussian distribution in Eq. (19), which plays an important role in controlling the sparsity of parameters. That is, larger a leads to larger η_θ and larger $1/\phi_\theta$, which causes the variance σ_θ^2 smaller in Eq. (16). One step further, the small variance makes most of the parameters of SFM model close to zero, which is helpful to distinguish the relevant and irrelevant features. Therefore, we evaluate RMSE, NLL and AUC of SFM with different a from the set $\{1, 3, 10, 30, 100, 300, 1000, 3000, 10000\}$, and the result is presented in Figure 6. Here, we randomly select dataset 3358 from iPinYou and $Trial$ 3 from IOS as examples. From Figure 6, we can see that RMSE and NLL decrease and AUC increases as a grows from 1 to 10,000. In other words, with relatively large a, our SFM model can generate sparse parameters, which could improve AUC and decline RMSE and NLL at the same time. We observe that RMSE and NLL are stable at a good level when a is larger than 30, and the model performs quite well in AUC when a is larger than 300.

To better capture the parameters for other applications, we summarize some experience. For instance, we can select proper size of latent vectors k (e.g. $k = 8$) to get the balance between accuracy and efficiency. Moreover, a can be initialized with 300, and it is better to subsample the data and obtain a small group of data for further tuning.

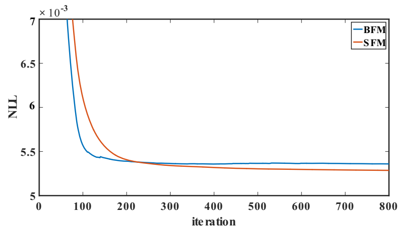
Comparison on Different Sparsity Levels. To straightforwardly demonstrate that SFM is a sparsity-favoring model, we

TABLE V
A Performance Comparison

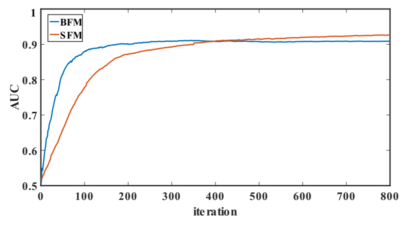
Dataset	RMSE			NLL			AUC		
	FM	BFM	SFM	FM	BFM	SFM	FM	BFM	SFM
1458	0.021	0.021	**0.020**	0.0050	0.0050	**0.0049**	0.978	0.972	**0.981**
3358	0.026	**0.025**	**0.025**	0.0055	0.0054	**0.0053**	0.965	0.966	**0.974**
3386	0.029	0.029	**0.028**	0.0060	0.0060	**0.0059**	0.716	0.747	**0.772**
3427	0.023	0.022	**0.021**	0.0052	0.0051	**0.0050**	0.935	0.931	**0.947**
3476	0.026	0.023	**0.022**	0.0054	0.0054	**0.0052**	0.884	0.923	**0.928**
2259	0.018	0.018	**0.017**	0.0050	0.0050	**0.0049**	0.665	0.649	**0.676**
2261	0.018	0.018	**0.017**	0.0050	0.0050	**0.0049**	0.552	0.571	**0.590**
2821	0.024	0.024	**0.023**	0.0055	**0.0051**	0.0055	0.596	0.601	**0.621**
2997	0.059	**0.058**	**0.058**	0.0112	0.0112	**0.0111**	0.591	0.584	**0.597**
$Trial$ 1	0.112	**0.111**	**0.111**	0.0295	0.0292	**0.0290**	0.545	0.548	**0.556**
$Trial$ 2	0.108	**0.107**	**0.107**	0.0272	0.0269	**0.0268**	0.701	0.782	**0.789**
$Trial$ 3	**0.140**	0.141	0.141	**0.0419**	0.0425	0.0425	0.630	0.629	**0.636**
$Trial$ 4	**0.145**	0.146	0.146	**0.0450**	0.0454	0.0454	0.567	0.566	**0.579**
$Trial$ 5	0.147	0.147	**0.146**	0.0457	0.0455	**0.0454**	0.580	0.577	**0.589**
$Trial$ 6	**0.149**	0.150	0.150	**0.0466**	0.0481	0.0479	0.601	0.596	**0.608**
$Trial$ 7	0.157	0.157	**0.156**	0.0506	0.0506	**0.0505**	0.591	0.590	**0.599**
$Trial$ 8	0.151	0.151	**0.150**	0.0470	0.0470	**0.0469**	0.613	0.615	**0.625**
$Trial$ 9	0.142	0.142	**0.141**	0.0434	0.0439	**0.0425**	0.587	0.564	**0.617**
$Trial$ 10	0.175	0.175	**0.174**	0.0597	0.0597	**0.0595**	0.618	0.617	**0.625**



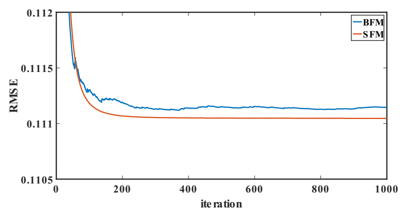
(a) RMSE of iPinYou dataset 3476



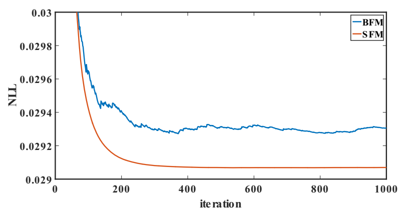
(b) NLL of iPinYou dataset 3476



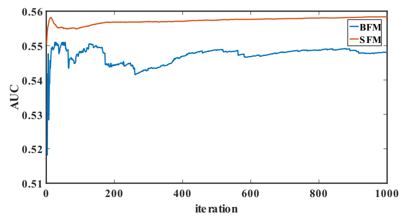
(c) AUC of iPinYou dataset 3476



(d) RMSE of IOS dataset $Trial$ 1



(e) NLL of IOS dataset $Trial$ 1



(f) AUC of IOS dataset $Trial$ 1

Fig. 4. Convergence on iPinYou dataset 3476 and IOS dataset $Trial$ 1.

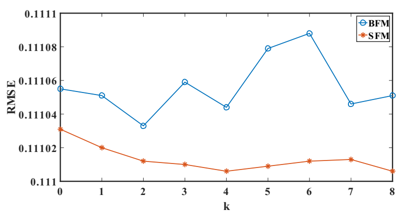
TABLE VI
Sparsity with different feature numbers

Dataset	2997			3358		
Features	5,803	15,804	35,805	5,803	15,804	35,805
Non-zero Rate(%)	0.3751	0.1377	0.0608	0.2669	0.1090	0.0481

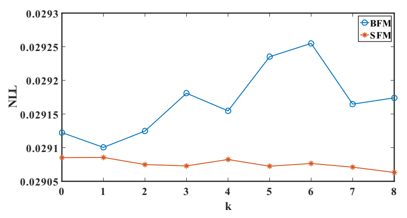
compare SFM with FM and BFM with different sparsity levels. Specifically, for two randomly chosen datasets 2997 and 3358 in iPinYou[2], we first extract features in three different ways, i.e., the features with 5,803 dimensions, 15,804 dimensions, and 35,805 dimensions, respectively. The proportion of *non-zero feature values* (sparsity) with different dimension of features are shown in Table VI. From this table we can see that the higher number of features, the higher sparsity. Then, we run FM, BFM and SFM on these datasets and their performance in terms of AUC, as shown in Figure 7. Combining Figure 7 and Table VI, we can see that when the feature values become sparser, the improvement of SFM is much higher than FM and BFM, i.e., SFM is a sparsity-favoring model as Laplace distribution could fit the sparse data very well. For better illustration, we visualize the parameter set Θ on these two datasets to present the different distributions of parameters in BFM and SFM in Figure 8 and Figure 9, respectively, where the distributions of parameters follow their prior assumptions, i.e., Gaussian distribution or Laplace distribution. We can see that the distribution of parameters can be well fitted by their prior assumptions. Besides, it should be noticed that the difference between the two distributions in 3358 is more obvious than that in 2997. Since there are more samples in 3358, the factorization models can adapt to the sparsity by sufficient training.

[2]The features of IOS dataset are previously extracted and fixed by the online advertising company, thus we only conduct this experiment on iPinYou.

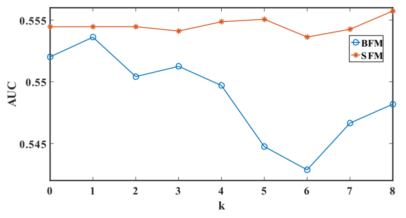
The Speedup of Distributed Implementation. Our SFM is implemented on a Spark platform consisting of 4 executors. Every executor has four 2.0GHz Intel Xeon E5-2620 CPUs and 100G memory. There are 6 cores in each CPU, so our cluster has 96 cores in total. For testing the performance of our distributed implementation of SFM on Spark, we compare



(a) RMSE

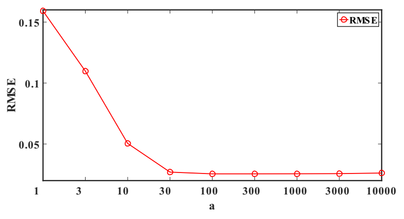


(b) NLL

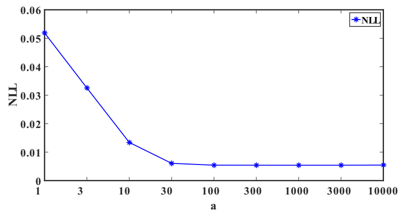


(c) AUC

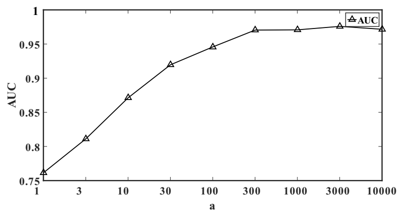
Fig. 5. The influence of parameter k on IOS dataset $Trial$ 1.



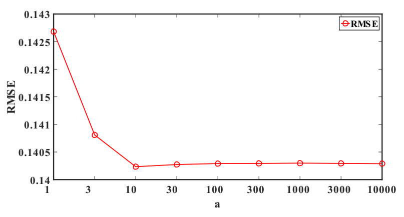
(a) RMSE of iPinYou dataset 3358



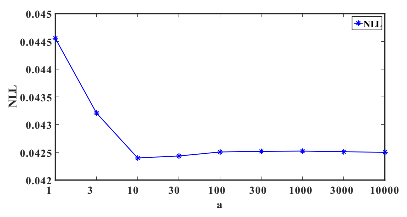
(b) NLL of iPinYou dataset 3358



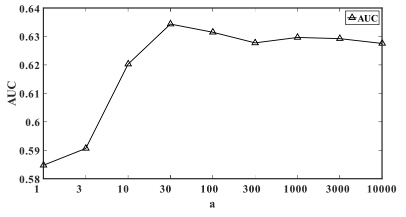
(c) AUC of iPinYou dataset 3358



(d) RMSE of IOS dataset $Trial$ 3

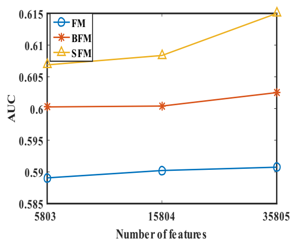


(e) NLL of IOS dataset $Trial$ 3

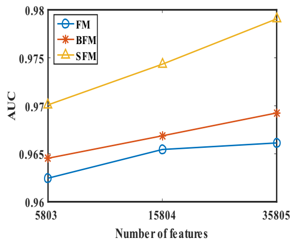


(f) AUC of IOS dataset $Trial$ 3

Fig. 6. The influence of parameter a on iPinYou dataset 3358 and IOS dataset $Trial$ 3.



(a) 2997



(b) 3358

Fig. 7. The influence of sparsity.

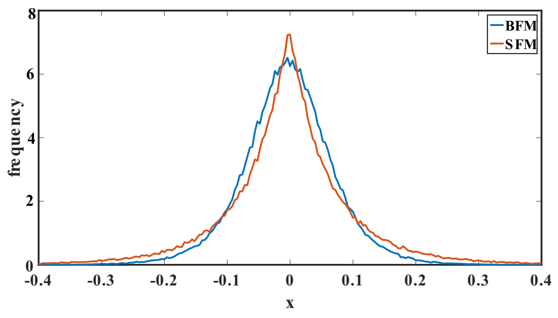


Fig. 8. The visualization of parameters of iPinYou dataset 2997.

the running time of SFM on Spark with different settings. Specifically, we set the number of executors as two extreme numbers 1 and 4, respectively, and then we record the running time of SFM. Actually, the comparison results are similar for different datasets, and for simplicity, we just illustrate three of them (i.e., datasets 2997, 2261 and 2259 from iPinYou) in Figure 10. As shown in Figure 10, the SFM with 4 executors could significantly speed up the computing process, i.e., our distributed implementation of SFM is quite effective. We should note that, we do not conduct such a running time comparison for FM and BFM, this is because we directly use the implementation of FM and BFM from libfm[3], and thus we do not have the distributed version of these two models.

[3]http://www.libfm.org/

V. Conclusion

In this paper, we provided a novel Sparse Factorization Machines (SFM) model for addressing the sparsity problem in the task of click-through rate prediction. Based on the analysis of the unique characteristics of ad data, we first proposed the idea of utilizing the highly peaked Laplace distribution in SFM to model the parameters. Since Laplace distribution is nonsmooth, we then designed a technique to make the Bayesian inference can be performed on SFM. Meanwhile, we also developed a distributed implementation of our SFM model on Spark to deal with large-scale ad data. Finally, we conducted extensive experiments on two real-world datasets from desktop display advertising and iOS platform

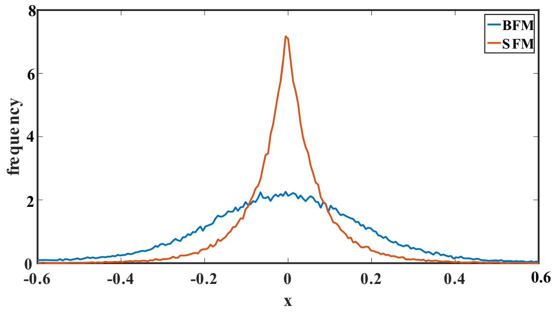


Fig. 9. The visualization of parameters of iPinYou dataset 3358.

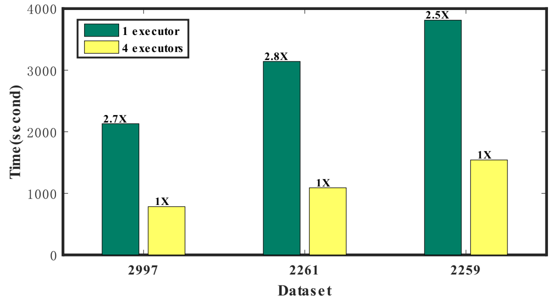


Fig. 10. The running time on Spark.

advertising, respectively. The experimental results demonstrate that Laplace distribution is very suitable to describe the online ads transaction data and SFM model can effectively predict the click-through rate.

VI. Acknowledgments

This research was partially supported by grants from the National Key Research and Development Program of China (Grant No. 2016YFB1000904), the National Science Foundation for Distinguished Young Scholars of China (Grant No. 61325010) and the Fundamental Research Funds for the Central Universities of China (Grant No. WK2350000001). Qi Liu gratefully acknowledges the support of the Youth Innovation Promotion Association of CAS (No. 2014299).

References

[1] Spark. http://spark.apache.org.
[2] D. F. Andrews and C. L. Mallows. Scale mixtures of normal distributions. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 99–102, 1974.
[3] A. I. Baqapuri and I. Trofimov. Using neural networks for click prediction of sponsored search. *arXiv preprint arXiv:1412.6601*, 2014.
[4] O. Chapelle, E. Manavoglu, and R. Rosales. Simple and scalable response prediction for display advertising. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 5(4):61, 2015.
[5] B. Dalessandro, D. Chen, T. Raeder, C. Perlich, M. Han Williams, and F. Provost. Scalable hands-free transfer learning for online advertising. In *SIGKDD*, pages 1573–1582. ACM, 2014.
[6] C. Freudenthaler, L. Schmidt-Thieme, and S. Rendle. Bayesian factorization machines. 2011.
[7] T. Graepel, J. Q. Candela, T. Borchert, and R. Herbrich. Web-scale bayesian click-through rate prediction for sponsored search advertising in microsoft's bing search engine. In *ICML*, pages 13–20, 2010.
[8] X. He, J. Pan, O. Jin, T. Xu, B. Liu, T. Xu, Y. Shi, A. Atallah, R. Herbrich, S. Bowers, et al. Practical lessons from predicting clicks on ads at facebook. In *ADKDD*, pages 1–9. ACM, 2014.
[9] L. Jing, P. Wang, and L. Yang. Sparse probabilistic matrix factorization by laplace distribution for collaborative filtering. In *IJCAI*, pages 1771–1777. AAAI Press, 2015.
[10] K.-c. Lee, B. Orten, A. Dasdan, and W. Li. Estimating conversion rate in display advertising from past erformance data. In *SIGKDD*, pages 768–776. ACM, 2012.
[11] H. Liao, L. Peng, Z. Liu, and X. Shen. ipinyou global rtb bidding algorithm competition dataset. In *ADKDD*, pages 1–6. ACM, 2014.
[12] Q. Liu, E. Chen, H. Xiong, Y. Ge, Z. Li, and X. Wu. A cocktail approach for travel package recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 26(2):278–293, 2014.
[13] Q. Liu, F. Yu, S. Wu, and L. Wang. A convolutional click prediction model. In *CIKM*, pages 1743–1746. ACM, 2015.
[14] Q. Liu, X. Zeng, H. Zhu, E. Chen, H. Xiong, X. Xie, et al. Mining indecisiveness in customer behaviors. In *Data Mining (ICDM), 2015 IEEE International Conference on*, pages 281–290. IEEE, 2015.
[15] H. B. McMahan, G. Holt, D. Sculley, M. Young, D. Ebner, J. Grady, L. Nie, T. Phillips, E. Davydov, D. Golovin, et al. Ad click prediction: a view from the trenches. In *SIGKDD*, pages 1222–1230. ACM, 2013.
[16] A. K. Menon, K.-P. Chitrapura, S. Garg, D. Agarwal, and N. Kota. Response prediction using collaborative filtering with hierarchies and side-information. In *SIGKDD*, pages 141–149. ACM, 2011.
[17] J. R. Michael, W. R. Schucany, and R. W. Haas. Generating random variates using transformations with multiple roots. *The American Statistician*, 30(2):88–90, 1976.
[18] S. Mohamed, K. Heller, and Z. Ghahramani. Bayesian and l1 approaches to sparse unsupervised learning. *arXiv preprint arXiv:1106.1157*, 2011.
[19] S. Muthukrishnan. Ad exchanges: Research issues. In *Internet and network economics*, pages 1–12. Springer, 2009.
[20] R. J. Oentaryo, E.-P. Lim, J.-W. Low, D. Lo, and M. Finegold. Predicting response in mobile advertising with hierarchical importance-aware factorization machine. In *WSDM*, pages 123–132. ACM, 2014.
[21] S. Rendle. Factorization machines. In *ICDM*, pages 995–1000. IEEE, 2010.
[22] S. Rendle. Factorization machines with libfm. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 3(3):57, 2012.
[23] S. Rendle, Z. Gantner, C. Freudenthaler, and L. Schmidt-Thieme. Fast context-aware recommendations with factorization machines. In *SIGIR*, pages 635–644. ACM, 2011.
[24] M. Richardson, E. Dominowska, and R. Ragno. Predicting clicks: estimating the click-through rate for new ads. In *WWW*, pages 521–530. ACM, 2007.
[25] J. Shen, B. Orten, S. C. Geyik, D. Liu, S. Shariat, F. Bian, and A. Dasdan. From 0.5 million to 2.5 million: Efficiently scaling up real-time bidding. In *ICDM*, pages 973–978. IEEE, 2015.
[26] J. Shi, N. Wang, Y. Xia, D. Y. Yeung, I. King, and J. Jia. Scmf: sparse covariance matrix factorization for collaborative filtering. In *IJCAI*, pages 2705–2711, 2013.
[27] A.-P. Ta. Factorization machines with follow-the-regularized-leader for ctr prediction in display advertising. In *Big Data*, pages 2889–2891. IEEE, 2015.
[28] J. Wang and S. Yuan. Real-time bidding: A new frontier of computational advertising research. In *WSDM*, pages 415–416. ACM, 2015.
[29] L. Xiao. Dual averaging method for regularized stochastic learning and online optimization. In *NIPS*, pages 2116–2124, 2009.
[30] L. Yan, W.-j. Li, G.-R. Xue, and D. Han. Coupled group lasso for web-scale ctr prediction in display advertising. In *ICML-14*, pages 802–810, 2014.
[31] S. Yuan, J. Wang, and X. Zhao. Real-time bidding for online advertising: measurement and analysis. In *ADKDD*, page 3. ACM, 2013.
[32] Y. Yuan, F. Wang, J. Li, and R. Qin. A survey on real time bidding advertising. In *SOLI*, pages 418–423. IEEE, 2014.
[33] W. Zhang, S. Yuan, and J. Wang. Optimal real-time bidding for display advertising. In *SIGKDD*, pages 1077–1086. ACM, 2014.
[34] W. Zhang, S. Yuan, J. Wang, and X. Shen. Real-time bidding benchmarking with ipinyou dataset. *arXiv preprint arXiv:1407.7073*, 2014.
[35] Y. Zhang, H. Dai, C. Xu, J. Feng, T. Wang, J. Bian, B. Wang, and T.-Y. Liu. Sequential click prediction for sponsored search with recurrent neural networks. In *AAAI*, 2014.
[36] Z. Zhang, S. Wang, D. Liu, and M. I. Jordan. Ep-gig priors and applications in bayesian sparse learning. *The Journal of Machine Learning Research*, 13(1):2031–2061, 2012.