Received November 26, 2018, accepted December 3, 2018, date of publication December 7, 2018, date of current version February 6, 2019.

Digital Object Identifier 10.1109/ACCESS.2018.2885399

Research on CTR Prediction Based on Deep Learning

QIANQIAN WANG[1], FANG'AI LIU[1], SHUNING XING[1], XIAOHUI ZHAO[1,2], AND TIANLAI LI[1]
[1]School of Information Science and Engineering, Shandong Normal University, Jinan 250014, China
[2]School of Mathematical Science, Shandong Normal University, Jinan 250014, China

Corresponding author: Fang'ai Liu (lfa@sdnu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61772321, in part by the Natural Science Foundation of Shandong Province under Grant ZR2016FP07, in part by the Innovation Foundation of Science and Technology Development Center of Ministry of Education and New H3C Group under Grant 2017A15047, and in part by the CERNET Innovation Project under Grant NGII20170508.

ABSTRACT Click-through rate (CTR) prediction is critical in Internet advertising and affects web publisher's profits and advertiser's payment. In the CTR prediction, the interaction between features is a key factor affecting the prediction rate. The traditional method of obtaining features using feature extraction did not consider the sparseness of advertising data and the highly nonlinear association between features. To reduce the sparseness of data and to mine the hidden features in advertising data, a method that learns the sparse features is proposed. Our method exploits dimension reduction based on decomposition and combines the power of field-aware factorization machines and deep learning to portray the nonlinear associated relationship of data to solve the sparse feature learning problem. The experiment shows that our method improves the effect of CTR prediction and produces economic benefits in Internet advertising.

INDEX TERMS Click through rate, deep learning, factorization machines, sponsored search, tensor decomposition.

I. INTRODUCTION

Click-through rate (CTR) prediction is critical to many web applications including web search, recommender systems [1], [2], sponsored search, and display advertising. Search advertising, known as sponsored search, refers to advertisers identifying relevant keywords based on their product or service for advertising. When the user retrieves the keyword purchased by the advertiser, the corresponding advertisement is triggered and displayed. In the cost-per-click model, the advertiser pays the web publisher only when a user clicks their advertisements and visits the advertiser's site. The CTR prediction is defined to estimate the ratio of clicks to impressions of advertisements that will be displayed [3].

With the development of online promotion technology, accurate advertising delivery has become the standard for online marketing trends. In the process of purchasing and searching for ads on the demand side platform (DSP), it is necessary to assess the user's preferences for advertising, and an important indicator of this preference is the click-through rate of the advertisement [4]. The process of sponsored search is similar to a web search, including query analysis, advertising search, advertising sorting and other stages. The CTR is the most important measure of advertising revenue. In many recommender systems, the goal is to maximize the number of clicks, so the items returned to a user can be ranked by the estimated CTR. The CTR is also important for improving revenue in other application scenarios such as online advertising. Therefore, the ranking strategy can be adjusted as CTR*bid across all candidates, where "bid" is the benefit the system receives if the item is clicked by a user. It is clear that the goal is to estimate the CTR correctly.

Considering the high dimensional sparsity of advertising data and the highly non-linear association between features [5], a sparse feature learning method from the perspective of deep learning is proposed in this paper. We explore data dimension reduction and identify the relationship between features based on deep learning. Additionally, many experiments are conducted to show that this method improves the accuracy of CTR estimation.

The rest of this paper is organized as follows. Section 2 provides a brief overview of the relevant work. In Section 3, considering the problems existing in the contemporary work and the characteristics of the advertising data, the sparse feature learning method for advertising data based on deep learning (DLSFL) is proposed. In Section 4, we design the experiment and verify the prediction accuracy of the method by comparison experiment. We also analyze the experimental results in this section. Section 5 concludes the paper and lists possible future work.

II. RELATED WORK

Click-through rate, defined as the probability of an ad click from a specific user on a displayed ad, is essential in online advertising [6]. To maximize revenue and user satisfaction, online advertising platforms must predict the expected user behavior for each displayed advertisement and maximize the expectation that users will click. There are many features that affect CTR but considering the features may not receive better effects, it is necessary to identify the features that are relevant to improving click-through rate estimation. Researchers have proposed many models that are usually based on machine learning methods. We can divide them into three categories: linear, nonlinear, and fusion models. McMahan *et al.* [7] used a logistic regression model to solve the CTR problem of Google ads. Using multiple aspects of characteristics including user information, search keywords, advertising data and relative metadata with advertisement as the input of the model, the proposed the online sparse learning algorithm to train the model. Chapelle [8] proposed a machine learning framework based on Logistic Regression (LR), aiming to predict the CTR for Yahoo website. Jahrer *et al.* [9] proposed using collaborative filtering, a Bayesian network model and feature engineering to predict the click rate. He *et al.* [10] introduced a fusion model which combines decision trees with logistic regression for predicting clicks on Facebook ads. However, when the features of advertising data are sparse, the prediction accuracy is inaccurate. Shen *et al.* [11] considered the user personalization information and proposed collaborative filtering and tensor decomposition to extract personalized features. Kumar *et al.* [12] from BV Bhoomaraddi College of Engineering and Technology adopted logistic regression to predict the CTR of search engine advertising and achieved approximately 90% accuracy on a dataset of approximately 25 GB. Baqapuri and Trofimov [13] proposed a novel architecture to solve the CTR prediction for sponsored search advertising by combining artificial neural networks with boosted trees. The processing of features in the above several models requires a large number of feature engineering.

The advertising data has sparse feature. Richardon *et al.* used the known advertisements based on the same or similar items to estimate CTR. One of the biggest challenges in CTR prediction is the lack of information, especially for new ads. The historical display data is too small to provide a pre-estimated benchmark for the click-through model. Liu *et al.* [14] extended CNN for CTR prediction, but CNN-based models are biased towards the interactions between neighboring features. Shan *et al.* [15] proposed the deep crossing model which is a deep neural network that automatically combines features to produce superior models. The model did not consider the important of low-order features. Product-based neural networks with an embedding layer to learn a distributed representation of the categorical data was proposed by Qu *et al.* [16], but the model also ignored the low-order feature. The Convolutional Click Prediction Model (CCPM) is a convolutional model for click prediction [14]. This model learns local-global features efficiently. However, CCPM relies on feature alignment and lacks interpretation.

The click-through rate estimation method has been widely studied, but problems remain. The key to improving the click-through rate of advertising is to explore the relationship between features. Advertising data has high-dimensional sparse features. The effective information (non-zero value) in the high-dimensional features is minimal, and the noise disturbs the real information. Most CTR estimation methods can not accurately estimate the click-through rate in high-dimensional and highly sparse advertising data.

Another characteristic of advertising data is the non-linear association between features. The method of artificial structural features (also known as ‘‘artificial feature engineering’’) is inefficient and has poor scalability. The focus of this paper is automatically mining the association between features and reducing manual intervention.

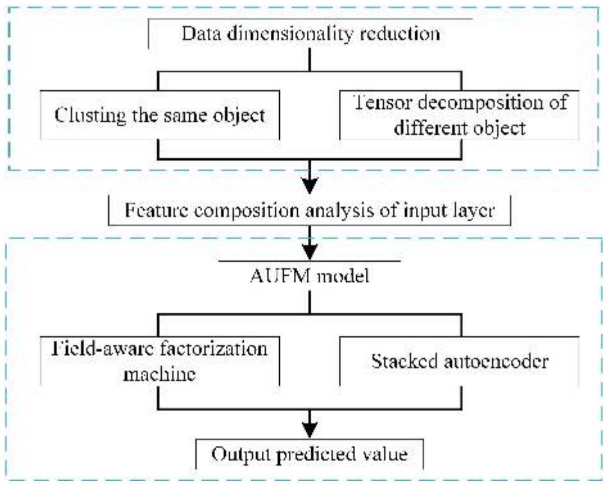

Data dimensionality reduction
Clusting the same object
Tensor decomposition of different object
Feature composition analysis of input layer
AUFM model
Field-aware factorization machine
Stacked autoencoder
Output predicted value

FIGURE 1. The structure of DLSFL method.

III. THE METHOD OF CLICK-THROUGH RATE ESTIMATION BASED ON DEEP LEARNING

The key to improving the click-through rate of advertising is to explore the relationship between features. One of the necessary steps in the click rate prediction system is to mine features that are highly correlated with the estimated task. To improve the prediction of CTR, we propose a sparse feature learning method for advertising data based on deep learning (DLSFL), and DLSFL enable reduce the high sparseness of features and characterize the non-linear relationship between features. The network structure of the proposed DLSFL method is shown in Figure. 1.

A. DATA DIMENSIONALITY REDUCTION

There are complex relationships between different types of objects. For instance, given a particular user and the query submitted by the user, it is necessary to predict whether the user will click on the advertisement and the probability. There is a complex implicit relationship between users, queries and advertising. Based on the characteristics of the click log data, dimension reduction is achieved in the following two aspects: the similarity between the internal objects and the association between different objects.

In this paper, the k-means clustering algorithm [17] based on distance is adopted. We cluster queries, advertisements and users separately, and the similar objects are aggregated into the same cluster. We use advertising frequency as the weight of the advertisement A_i and query Q_j, and create a matrix $W_{M_a \times M_q}$ of the ad-query (where M_a is the number of ads, and M_q indicates the number of queries), using the k-means algorithm to cluster the ad-query matrix. We scan the ad-query matrix to obtain the ad sets and query sets, as $A = \{a_1, a_2, \cdots, a_m\}$ and $Q = \{q_1, q_2, \cdots, q_N\}$. Then, we take K samples from the advertising set randomly as the initial point of the cluster center, record as $T = \{t_1, t_2, \cdots, t_k\}$. Next, Equation (1) is used to calculate the distance between ad a_i and each cluster center point t_j. The number of clusters of users, ads and queries is as K_u, K_a, K_q, respectively. Finally, the number of users, ads and queries in the data set is reduced from M_u, M_a, M_q to K_u, K_a, K_q.

$$Dis(a_i, t_j) = \sqrt{\sum_{c \in G_{ij}} (W_{a_i} - W_{q_j})^2} \tag{1}$$

where G_{ij} represents a collection of queries that a_i and t_j show together, W_{a_i} is the weight of a_i, W_{q_j} is the weight of t_j and $Dis(a_i, t_j)$ is the distance between a_i and t_j.

There is a ternary relationship among the user, query and ad in the click log data. In this paper, we use a three-dimensional tensor structure model [18], [19] to represent the user, query and advertisement. The tensor decomposition method is used to reduce the dimensions. The sum of the display number of ads in the cluster is used as the weight of the elements in 3D space. The three-dimensional tensor model is constructed and represented by $X \in R^{K_u \times K_q \times K_a}$. In this paper, tensor X is decomposed using Tucker factorization. Equation (2) is the decomposition formula.

$$\begin{aligned} X &= [G; A, B, C] = G \times {}_u A \times {}_q B \times {}_a C \\ &= \sum_{p=1}^{P} \sum_{m=1}^{M} \sum_{r=1}^{R} g_{pmr} u_p \circ q_m \circ a_r \end{aligned} \tag{2}$$

In Equation (2), G represents the core tensor of tensor X and $\circ$ denotes the composite operation. We use A, B and C to represent the feature matrix of the tensor X on the dimension K_u, K_q, K_a.

Figure 2 is a schematic diagram of the Tucker decomposition. The purpose of the tucker decomposition is to find an approximate tensor $\hat{X}$ with the original tensor X and to retain the original tensor information and structural information to the greatest extent [20]. The minimization formula is shown

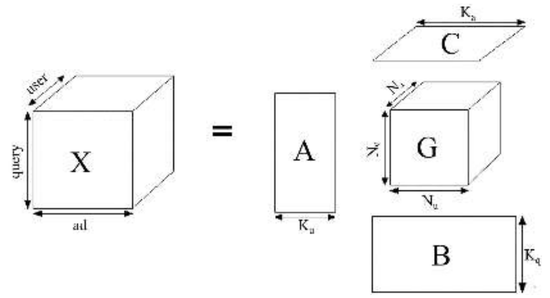


FIGURE 2. Schematic diagram of the Tucker decomposition.

below:

$$\min \left\| X - \hat{X} \right\|, \hat{X} = G \times {}_u A \times {}_q B \times {}_a C = [G; A, B, C] \tag{3}$$

Equation (3) is the objective optimization function. $\|.\|$ is the norm of the vector. According to Equation (3), the expression of the core tensor can be obtained as follows:

$$G = X \times {}_u A^{\mathrm{T}} \times {}_q B^{\mathrm{T}} \times {}_a C^{\mathrm{T}} \tag{4}$$

the objective function can be written in a squared form:

$$\begin{aligned} &\|X - [G; A, B, C]\|^2 \\ &= \|X\|^2 - 2\left\langle X \times_u A^{\mathrm{T}} \times {}_q B^{\mathrm{T}} \times {}_a C^{\mathrm{T}}, G\right\rangle + \|G\|^2 \\ &= \|X\|^2 - 2\langle G, G\rangle + \|G\|^2 \\ &= \|X\|^2 - \|G\|^2 \\ &= \|X\|^2 - \left\| X \times_u A^{\mathrm{T}} \times {}_q B^{\mathrm{T}} \times {}_a C^{\mathrm{T}} \right\|^2 \end{aligned} \tag{5}$$

where $\langle . \rangle$ denotes the inner product.

Therefore, the objective function is transformed to:

$$\begin{aligned} \max &\left\| X \times {}_u A^{\mathrm{T}} \times {}_q B^{\mathrm{T}} \times {}_a C^{\mathrm{T}} \right\|^2 \\ &\left\| A^{\mathrm{T}} W \right\|, W = X \times {}_q B^{\mathrm{T}} \times {}_a C^{\mathrm{T}}; \\ &\left\| B^{\mathrm{T}} W \right\|, W = X \times {}_u A^{\mathrm{T}} \times {}_a C^{\mathrm{T}}; \\ &\left\| C^{\mathrm{T}} W \right\|, W = X \times {}_u A^{\mathrm{T}} \times {}_q B^{\mathrm{T}}; \end{aligned} \tag{6}$$

In the process of solving the optimal solution, we need to fix the matrix of the other dimensions W, solve for A^{T}, B^{T}, C^{T}, and then perform a singular value decomposition (SVD) of A^{T}, B^{T}, C^{T}. Next, X_1, X_2, X_3 are obtained from expanding the tensor X to a matrix on the user, query, and advertising dimensions, respectively, and then we apply SVD on X_1, X_2, X_3:

$$\begin{aligned} X_1 &= A \cdot G_1 \cdot V_1^{\mathrm{T}}; \\ X_2 &= B \cdot G_2 \cdot V_2^{\mathrm{T}}; \\ X_3 &= C \cdot G_3 \cdot V_3^{\mathrm{T}}; \end{aligned} \tag{7}$$

G_1, G_2, G_3 are diagonal singular value matrices obtained using singular value decomposition of the matrices X_1, X_2, X_3. The dimensions of the singular value matrix A, B, C are obtained by calculating the diagonal singular values of G_1, G_2, G_3 in proportion. V_1^T, V_2^T, V_3^T are vectors that obtained by SVD. In the process of reducing the dimensions, the proportion of exclusion singular values is set to 50% in this paper. Therefore, the calculation of the core tensor after dimension reduction is as follows:

$$G' = X \times {}_u A_{r1}^{\mathrm{T}} \times {}_q B_{r2}^{\mathrm{T}} \times {}_a C_{r3}^{\mathrm{T}}$$
$$X' = G' \times {}_u A_{r1} \times {}_q B_{r2} \times {}_a C_{r3} \quad (8)$$

The three dimensions of the initial tensor X are K_u, K_q, K_a, and the three dimensions of the approximate tensor X' after decreasing dimension are denoted by N_u, N_q, N_a. The time complexity of the Tucker decomposition algorithm is proportional to the tensor dimension, which is expressed as $O(K_u K_q K_a)$. We previously used the clustering method to achieve the reduction of the original matrix, which reduced the cost of the Tucker decomposition is greatly reduced, and improved the efficiency and precision.

B. FEATURE COMPOSITION ANALYSIS OF INPUT LAYER

There is a high degree of non-linear correlation between the features in advertising data. Although the approximate tensor of the original tensor is reduced by Tucker decomposition, it only reflects the information between the three characteristic dimensions of user, query and ad. Other useful information in the data is not fully utilized for click-through rate estimates, such as the position of the advertisement on the page, the number of ads, and the age and gender of the user, etc. This paper combines the features of <user, query, ad> after tensor reduction and other valid information in the log data as the object of feature learning. The composition of the input layer features is summarized as follows:

1) ID FEATURE

ID feature uniquely identifies a class of entities in the actual click log, usually use a set of numeric strings to represent variables. For instance, '10110' can identify only one user group. The ID class used in this article has the UserID, QueryID, AdID, position, and the number of advertisements on the return page. UserID, QueryID, and AdID are collections of ''virtual'' ID classes that are obtained using K-means clustering and tensor dimension reduction.

2) ATTRIBUTE FEATURE

The ID feature is a symbol that cannot be obtained from the new entity data, and has weak generalization ability. Attribute features are used to describe a set of users, ad collections, etc., and have better generalization ability and apply to multiple instances. Therefore, the input layer should contain the attribute features. Commonly used attribute features are: user's URL, user's gender, user's age, advertising time to trigger and query keywords.

3) STATISTICAL FEATURE

The statistical feature uses statistical information from historical data to provide an estimate for the forecasting model. The statistical characteristics of the text are the number of advertising histories, the number of clicks on the advertising history and the click-through rate after the advertising position normalization of, denoted by Shows, Clicks and (COEC) [21], COEC is a position normalized statistic:

$$COEC(a) = \sum_p c(p,a) \Big/ \sum_p ec(p,a) \quad (9)$$

where the numerator indicates the sum of the number of clicks for advertising a in all locations, the denominator represents the sum of the expected number of clicks for advertising a at each location.

In the experiment, the features of input layer are shown in Figure 3.

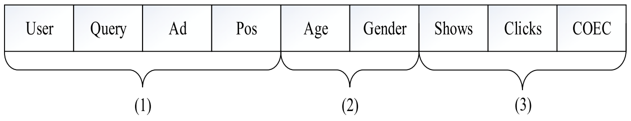


FIGURE 3. The features of input layer.

C. FEATURE LEARNING MODEL BASED ON DEEP LEARNING

The study of the machine learning field shows that the models of depth or hierarchy are more effective in characterizing non-linear relationships and complex patterns in the data. The key challenge is in effectively modelling feature interactions. Some feature interactions are known and can be designed by experts. However, most other feature interactions are hidden in data and difficult to identify a priori, and can only be captured automatically by machine learning. As a powerful approach to learn feature representation, deep learning has the potential to learn sophisticated feature interactions [22]. To model both low and high-order feature interactions, Cheng *et al.* [23], proposed an interesting hybrid network structure (Wide & Deep) that combines a linear (''wide'') model and a deep model. In this model, two different inputs are required for the ''wide part'' and ''deep part'', and the input of the ''wide part'' still relies on expertise feature engineering. We propose a new model named AUFM that integrates the architectures of field-aware factorization machines (FFM) and stacked autoencoder. This model is used to automatically learn the pattern characteristics of data, and integrate the acquired characteristics into the modeling process (such as classification and prediction) to overcome the incomplete defects of artificial feature engineering.

1) FIELD-AWARE FACTORIZATION MACHINE (FFM)

The factorization machine (FM) was proposed by Rendle *et al.* [24], [25] to learn feature interactions for recommendations. Pairwise interaction tensor factorization (PITF) [24] is a variant of FM. In KDD Cup 2012, ''Team Opera Solutions'' [26] proposed a generalization of PITF called ''factor model''. The idea of FFM originates from PITF proposed for recommender systems with personalized tags. PITF is used for recommender systems and is limited to three specific fields (User, Item, and Tag). In FM, every feature

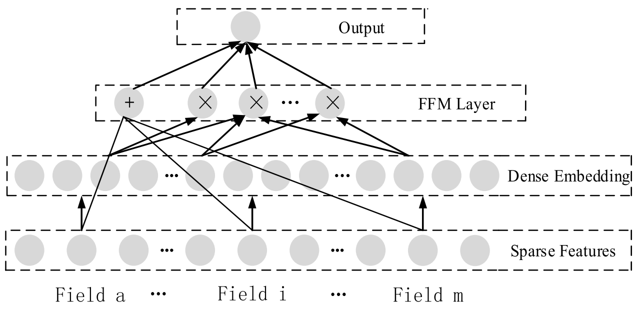


FIGURE 4. **The architecture of FFM.**

has only one latent vector to learn the latent effect with any other features. In this paper, features are grouped into fields. Each feature has several latent vectors, and each latent vector contains k latent factors, where k is a user-specified parameter. Depending on the field of other features, one of them is used to perform the inner product. Mathematically,

$$\phi FFM(w,x) = <w,x> + \sum_{j_1=1}^{n}\sum_{j_2=j_1+1}^{n}(w_{j_1,f_2}\cdot w_{j_2,f_1})x_{j_1}x_{j_2} \tag{10}$$

where f_1 and f_2 are respectively the fields of j_1 and j_2. The addition unit ($< w, x >$) reflects the importance of order-1 features. The structure is shown in Figure 4. To explain how FFM works, we consider the following new example:

Clicked	Publisher (P)	Advertiser (A)	Gender (G)
Yes	ESPN	Nike	Male

Three features ESPN, Vogue, and NBC, belong to the field Publisher, and the other three features Nike, Gucci, and Adidas, belong to the field Advertiser. Take ESPN as an example, w_{ESPN} is used to learn the latent effect with Nike ($w_{ESPN}\cdot w_{Nike}$) and Male ($w_{ESPN}\cdot w_{Male}$). However, Nike and Male belong to different fields, the latent effects of (EPSN, Nike) and (EPSN, Male) may be different. Depending on the field of other features, one of them is used to do the inner product. In our example, $\phi FFM(w, x)$ is

$$w_{ESPN,A}\cdot w_{Nike,P} + w_{ESPN,G}\cdot w_{Male,P} + w_{Nike,G}\cdot w_{Male,A}.$$

We see that to learn the latent effect of (ESPN, NIKE), $w_{ESPN,A}$ is used because Nike belongs to the field Advertiser, and $w_{Nike,P}$ is used because ESPN belongs to the field Publisher.

w is obtained by solving the following optimization problem:

$$\min_{w}\frac{\lambda}{2}\|w\|_2^2 + \sum_{i=1}^{m}\log(1+\exp(-y_i\phi FMM(w,x_i))) \tag{11}$$

where λ is the regularization parameter.

2) STACKED AUTOENCODER

The autoencoder (AE) [27] is often used to learn a better representation of the original data [28]; it consists of three network layers. The bottom is the input layer I, the middle of the hidden layer H and the output layer O or reconstruction layer. The autoencoder architecture is shown in Figure 5.

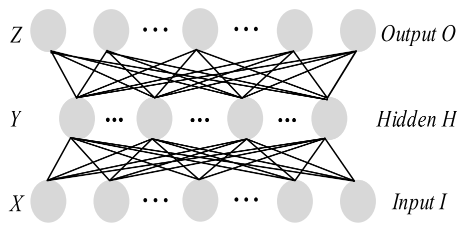


FIGURE 5. **The architecture of auto encoder.**

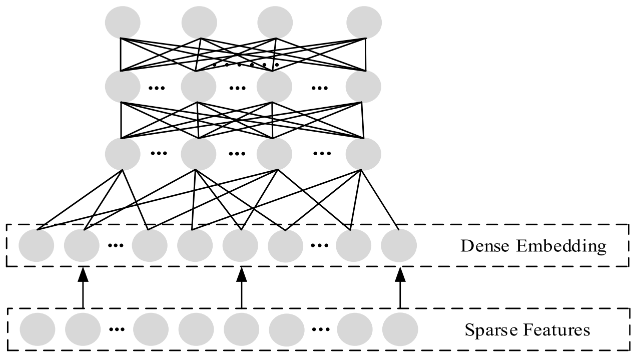


FIGURE 6. **Stacked autoencoder.**

Given a training data set $X = \{x^{(1)}, x^{(2)}, \cdots, x^{(n)}\}$, the encoder maps the sample $x^{(i)}$ to $y^{(i)}$. The process of mapping is a process of encoding, using the sigmoid function as the connection function to complete the encoding process, expressed as follows:

$$y^{(i)} = f(W^{(1)}\cdot x^{(i)} + b^{(1)}) \tag{12}$$

where $W^{(1)}$ is a coding weight matrix, and $b^{(1)}$ is the bias. From the hidden layer $y^{(i)}$ to the output layer $z^{(i)}$, the process of reconstructing the input vector is also a decoding process, which reconstructs the input vector $x^{(i)}$. It maps from $y^{(i)}$ to $z^{(i)}$ using a linear mapping as follows:

$$z^{(i)} = W^{(2)}y^{(i)} + b^{(2)} \approx x^{(i)} \tag{13}$$

where $W^{(2)}$ and $b^{(2)}$ represent the weighting matrices and bias vectors of the decoding process, respectively. The encoding algorithm minimizes the reconstruction error between the input $x^{(i)}$ and the output $z^{(i)}$, to obtain a set of parameters for the encoding and decoding process. Here, $Z = \{z^{(1)}, z^{(2)}, \cdots, z^{(m)}\}$ and $J(X, Z)$ represent the reconstruction error.

$$J(X,Z) = \frac{1}{2}\sum_{i=1}^{n}\left\|x^{(i)} - z^{(i)}\right\|^2 \tag{14}$$

In the autoencoder, multiple auto encoders form a stacked autoencoder. Each of its hidden layers is a non-linear transformation of the output of the previous layer. Figure 6 shows the structure.

3) AUFM MODEL

AUFM consists of two components, an FFM component and a stacked autoencoder component, both these two components share the same input. AUFM can be trained efficiently because its wide part and deep part, share the same input and embedding vector. In the stacked autoencoder, multiple autoencoder form a multi-layer depth network structure. Each of its hidden layers is a non-linear transformation of the output of the previous layer. The initial feature is used as the input of the model, and the non-linear feature of the initial feature is transformed to obtain the first hidden layer, the low order combination. The low-order combination of features as a new learning object and the combination features of the higher order are obtained by non-linear transformation. This process is repeated until the set number of hidden layers is reached and that is shown in Figure 7.

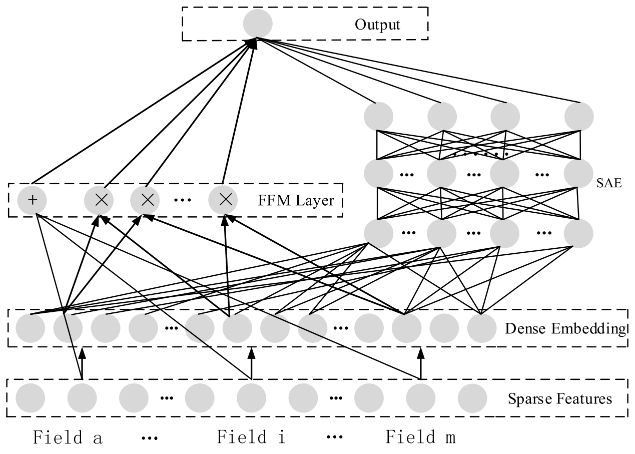


FIGURE 7. The architecture of AUFM.

The AUFM model integrates FFM and autoencoders to predict click rate. The graphical model of the AUFM model is shown in Figure 7. The objective function is minimized as follows:

$$\min_{w} \frac{\lambda}{2}\|w\|_2^2 + \sum_{i=1}^{m} \log(1+\exp(-y_i \times \sum_{j_1=1}^{n}\sum_{j_2=j_1+1}^{n}(w_{j_1,f_2}\cdot w_{j_2,f_1})x_{j_1}x_{j_2})) + \frac{1}{2}\sum_{i=1}^{n}\left\|x^{(i)}-z^{(i)}\right\|^2 \tag{15}$$

Following [25], [26], stochastic gradient methods (SG) are used to solve the local optimal problem in FFM. However, boosting the training process of SG has been proposed [29], and it is effective on matrix factorization. In this paper, we use the AdaGrad method [30] to solve the local optimal solution of the objective function. First, the sub-gradients are:

$$g_{j_1,f_2} \equiv \nabla_{w_{j_1,f_2}} f(w) = \lambda \cdot w_{j_1,f_2} + \kappa \cdot w_{j_2,f_1} \tag{16}$$

$$g_{j_2,f_1} \equiv \nabla_{w_{j_2,f_1}} f(w) = \lambda \cdot w_{j_2,f_1} + \kappa \cdot w_{j_1,f_2} \tag{17}$$

where

$$\kappa = \frac{\partial \log(1+\exp(-y\phi FFM(w,x)))}{\partial \phi FFM(w,x)} = \frac{-y}{1+\exp(y\phi FFM(w,x))}.$$

Second, for each coordinate $n = 1,\ldots,m$, the sum of the squared gradient is accumulated:

$$\begin{aligned}(G_{j_1,f_2})_m &\leftarrow (G_{j_1,f_2})_m + (g_{j_1,f_2})_m^2\\ (G_{j_2,f_1})_m &\leftarrow (G_{j_2,f_1})_m + (g_{j_2,f_1})_m^2\end{aligned} \tag{18}$$

Finally, $(w_{j_1,f_2})_m$ and $(w_{j_2,f_1})_m$ are updated by:

$$\begin{aligned}(w_{j_1,f_2})_m &\leftarrow (w_{j_1,f_2})_m - \frac{\eta}{\sqrt{(G_{j_1,f_2})_m}}(g_{j_1,f_2})_m\\ (w_{j_2,f_1})_m &\leftarrow (w_{j_2,f_1})_m - \frac{\eta}{\sqrt{(G_{j_2,f_1})_m}}(g_{j_2,f_1})_m\end{aligned} \tag{19}$$

where η is a user-specified learning rate. The initial values of W are randomly sampled and G is set to one.

In the stacked autoencoder component, we select the square error as the objective function and adopt the gradient descent to train the parameters. The related definition of the jth node in the hidden layer (y) of the AE can be described as follows:

s_h is the number of nodes in the hidden layer (H) of the AE.

w_{ji}^h is the connection weight between the jth node of hidden layer (H) and the ith node of input layer (I).

b_j^h is the bias of the jth node in the hidden layer (H).

$net_j^h = b_j^h + \sum_{i=1}^{s_x} w_{ji}^h o_i^x$ is the weight sum of the input of the jth node in the hidden layer (H).

o_j^h is the output value of the jth node in the hidden layer (H). The activation function of every neuron node is $\sigma(x) = {}^1/_{1+e^{-x}}$.

The output value of the jth node in the hidden layer (H) can be represented by formula (20).

$$o_j^h = f(net_j^h) = \sigma(b_j^h + \sum_{i=1}^{s_x} w_{ji}^h o_i^x) \tag{20}$$

When the feature of the hidden layer H is decoded, the feature of the reconstruction layer O is obtained. The output value of the jth node in the reconstruction layer O can be represented by formula (21).

$$\begin{aligned}o_j^o &= g(net_j^o) = g(b_j^o + \sum_{i=1}^{s_h} w_{ji}^o o_i^h)\\ &= \sigma(b_j^o + \sum_{i=1}^{s_h} w_{ji}^o(\sigma(b_i^h + \sum_{k=1}^{s_x} w_{ik}^h o_k^x)))\end{aligned} \tag{21}$$

We use the square error as the objective function. It can be described by formula (22).

$$J(X,Z) = \frac{1}{2}\sum_{i=1}^{n}\left\|x^{(i)}-z^{(i)}\right\|^2 = \frac{1}{2}\sum_{j=1}^{s_x}(o_j^x - o_j^o)^2 \tag{22}$$

To easily calculate and deduce the formulae, we define the residual error δ_j^l of the jth node in the lth layer. The residual error δ_j^h of the neuron node of the reconstruction layer can be calculated using formula (23) according to chain rule.

$$\begin{aligned}\delta_j^h &= \frac{\partial J}{\partial net_j^h} = \frac{\partial J}{\partial o_j^h}\frac{\partial o_j^h}{\partial net_j^h} \\ &= \sum_{i=1}^{s_o}\left(\frac{\partial J}{\partial net_i^o}\frac{\partial net_i^o}{\partial o_j^h}\right)\cdot\frac{\partial o_j^h}{\partial net_j^h} \\ &= \left(\sum_{i=1}^{s_o}\delta_i^o w_{ji}^o\right)\frac{\partial \delta(net_j^h)}{\partial net_j^h} \\ &= \left(\sum_{i=1}^{s_o}\delta_i^o w_{ji}^o\right)\delta(net_j^h)(1-\delta(net_j^h)) \\ &= \left(\sum_{i=1}^{s_o}\delta_i^o w_{ji}^o\right)o_j^h(1-o_j^h) \end{aligned} \tag{23}$$

The parameters w_{ji}^l and b_j^l can be calculated by formulae (24) and (25).

$$\frac{\partial J}{\partial w_{ji}^l} = \frac{\partial J}{\partial net_j^l}\frac{\partial net_j^l}{\partial w_{ji}^l} = \delta_j^l \cdot o_i^{l-1} \tag{24}$$

$$\frac{\partial J}{\partial b_j^l} = \delta_j^l \tag{25}$$

The parameters w_{ji}^l and b_j^l can be updated as the following formulae, where ε is the learning rate.

$$w_{ji}^l = w_{ji}^l - \beta\frac{\partial J}{\partial w_{ji}^l} = w_{ji}^l - \beta\cdot\delta_j^l\cdot o_i^{l-1} \tag{26}$$

$$b_j^l = b_j^l - \beta\frac{\partial J}{\partial b_j^l} = b_j^l - \beta\cdot\delta_j^l \tag{27}$$

IV. EXPERIMENTS

In this section, we compare our proposed DLSFL method and the other models empirically. The evaluation result indicates that our proposed DLSFL is more efficient than other methods and the estimate is more effective. At the same time, we observe the impact of different data sizes on the click-through rate estimation result.

A. DATASETS

The experimental data in this paper were from SIGKDD Cup2012 track2, which is the advertising click log data provided by soso (Tencent's search engine). The KDD2012 CUP track2 corresponding research question was based on the actual click data information to predict the click rate of the advertisement, which included click data information, user query, return advertising information, return page information, etc.

The training dataset provided by the competition had a total of 149,639,105 records, and the size of 9.8GB. In addition to the number of click and the number of displays, the test dataset was consistent with the training dataset, a total of 20,257,594 records, 1.28GB in size. A record in the dataset represents all the information contained in the advertisement shown by the user's search behavior, also known as an instance.

B. EXPERIMENTAL DESIGN PROCESS

1) EXPERIMENTAL DATA DIVISION

After data cleaning and data pre-processing, a total of 3.5 million samples were randomly selected from the candidate dataset for the experiment. The data statistics used in the experiment are shown in table 1.

TABLE 1. Dataset.

	Sample	User	Query	Ad
Dataset	3500,000	127,385	150,672	284,379
Training data	3000,000			
Test data	500,000			

During the experiment, we used the training data to train the model in the dataset of seven different scales, and verified the prediction performance of different methods on the same test set. The samples of seven different scale datasets were 150000, 200000, 300000, 500000, 600000, 750000 and 1 million. The training data were grouped randomly, and the final result was the average of all the experimental results to ensure the reliability of the experimental results.

2) THE METHOD COMPARISON

LR [31]: LR is the most widely used linear model in industrial applications. It is easy to implement and fast to train. However, it is unable to capture non-linear information.

HPCM [11]: HPCM used matrix resolution to obtain the user-query related information by tensor decomposition. Then using Bayesian network modelling, the obtained probability model was solved using the EM algorithm. However, the method could not capture the relationship between features effectively.

Human_LR [32]: Human_LR used LR as the estimation model. It was extracted by artificial feature and obtained the characteristics related to the click rate estimation.

FM [22]: FM was successfully applied to the recommended system and user response prediction task. FM explores feature interaction, which is effective on sparse data.

Wide&Deep [33]: This model combines a linear (''wide'') model and a deep model. The deep part is a three-layer MLP that first concatenates feature embedding. The wide part (which is a linear regression model) is subject to design to incorporate cross-features.

DeepCross [15]: It applies a multi-layer residual network on a feature embedding cascade for learning feature interactions. This model is a deep neural network that automatically combines features to produce superior models.

FNN [34]: FNN is a FM-initialized feed forward neural network. It is able to capture high-order latent patterns of multi-field categorical data.

DBNLR [35]: This model integrates a deep belief network (DBN) with logistic regression to address the problem of CTR prediction. A DBN stacked of RBMs is used to obtain abstract features from original advertisement datasets, and then a regression model is adopted to calculate the CTR prediction value.

3) EVALUATION METRICS

In this paper, the area under the ROC curve (AUC) [3] and log-loss were used as the evaluation criterion of the model. AUC is the size of the area below the ROC curve, which is usually between [0.5, 1). The larger the AUC is, the better the performance of the click-through model.

C. ANALYSIS OF EXPERIMENTAL RESULTS

1) IMPACT OF PARAMETERS

The number of network layers n in the depth learning phase, and the number of iterations ($iter$) in the training phase have a direct effect on the final estimate of the model. Therefore, this paper first experimented with parameters to select the best combination of parameters. We used a set of sampled data training models with a data size of 500,000 samples, tested on the test set. When we fixed the network layer number ($n = 2, 3, 4, 5, 6$) of the model, we analyzed the effect of different $iter$ on the model performance, and the results are shown in table 2.

TABLE 2. The relationship between the number of network layers and *iter*.

iter	AUC				
	n=2	*n*=3	*n*=4	*n*=5	*n*=6
10	0.6757	0.6701	0.6807	0.6767	0.6762
20	0.6870	0.6714	0.6965	0.6854	0.6816
30	0.6916	0.6808	0.7077	0.6862	0.6925
50	0.7112	0.7055	0.7220	0.7106	0.7060
70	0.7292	0.7272	0.7347	0.7244	0.7262
90	0.7369	0.7388	0.7470	0.7361	0.7477
100	0.7406	0.7417	0.7507	0.7501	0.7458
110	0.7428	0.7431	0.7532	0.7517	0.7513
130	0.7434	0.7444	0.7531	0.7534	0.7491
150	0.7447	0.7453	0.7527	0.7508	0.7505

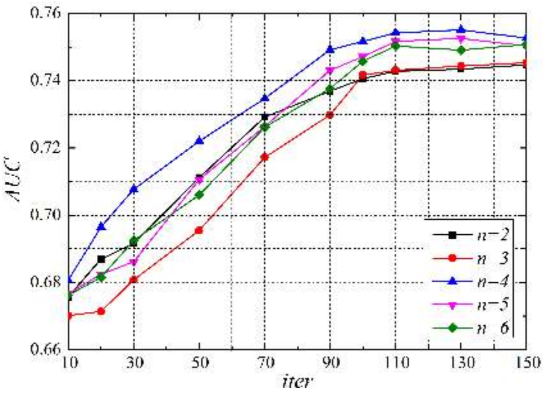
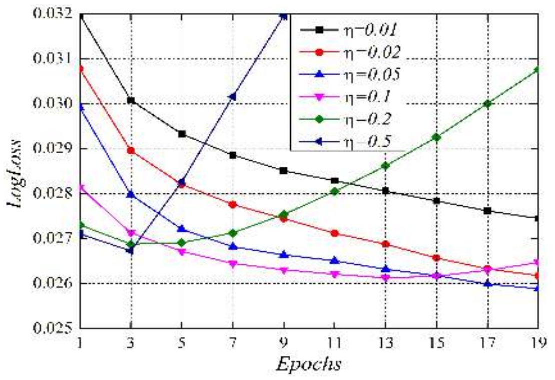


FIGURE 8. AUC comparison of different iter and different network layer.

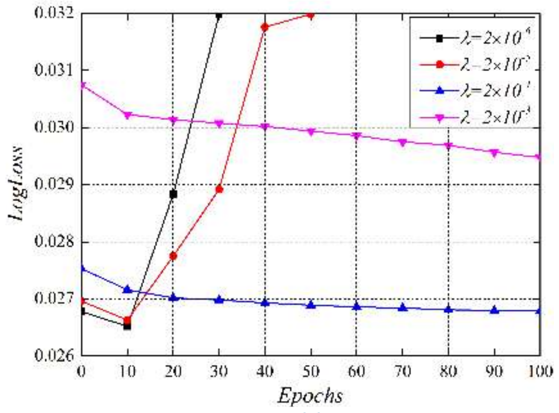
According to table 2 to generate Figure 8, Figure 8 reflects the AUC change for different network hidden layers n and LR model iterations for $iter$. As seen in Figure 8, when the number of iterations is 90 to 120, the AUC values of several curves stabilized. Therefore, in the comparison experiment, 115 was chosen as the number of iterations for training the prediction model. As shown in Figure 8, the curve fluctuated greatly with the change of iterations, and $n = 4$ was relatively stable, so we chose $n = 4$.

TABLE 3. The best log-loss with different values of ***k***.

k	1	2	4	8	10
LogLoss	0.02782	0.02765	0.02749	0.02738	0.02757



(a)



(b)

FIGURE 9. (a). The impact of λ (b). The impact of η.

The AUFM model has three important parameters: 1) The regularization parameter λ, 2) A user-specified parameter k, and 3) A user-specified parameter η. We conducted experiments to investigate the impact of k, λ and η. The results are shown in table 3. The table 3 shows that k did not significantly effect the log-loss. As seen in Figure 9 (a), we found the relationship between λ and log-loss. When λ was too large, the model could not achieve good results. But if λ was small, the model performed better. The training log-loss kept decreasing, and was easy to overfit. Choosing $\lambda = 2 \times 10^{-5}$ as the default value was reasonable. As shown in Figure 9 (b), when we had a small η, the model slowly performed better. However, if η was large, log-loss decreased quickly, but lead to over-fitting. We needed early-stopping so $\eta = 0.2$ was reasonable.

2) RUNTIME COMPARISON

In this paper, the operation time for the three methods with different data sizes was recorded. Table 4 shows the average run time at 150,000, 300,000, 500,000, 750,000 and 1 million data.

TABLE 4. Time comparison with different data sizes.

Time/s	Data Size				
Method	150,000	300,000	500,000	750,000	1000,000
LR	75	164	258	389	502
Human_LR	93	195	302	472	584
HPCM	2734	4328	6256	9024	12742
FM	1926	3858	5572	8661	10893
Wide&Deep	2269	4052	5963	8937	12608
DeepCross	2133	4026	5834	8862	11763
FNN	2207	4134	5873	8946	11971
DBNLR	2125	4098	5892	8902	11879
DLSFL	2075	3950	5749	8743	11547

As seen in table 4, the nine methods differed in the model run time. The LR and Human_LR methods ran for a short time because they only had an estimated model training phase. The HPCM method relates to the complex operation of tensor decomposition and the EM algorithm. FM explores feature interaction, so the running time was longer. The computational overhead of the depth learning algorithm was also large, and the Wide&Deep, DeepCross, FNN, DBNLR and DLSFL methods ran longer. The DLSFL method proposed in this paper had no advantage in runtime. However, the most time-consuming advertising click-through rate estimation model is based on massive data training. The calculation process is carried out in an offline environment. As a result, the runtimes had no effect on the online CTR values.

3) PERFORMANCE COMPARISON

We trained the models on seven different scale datasets and evaluated the estimated results on the same test set. Table 5 and table 6 describe the estimated results for the different methods at different data sizes.

Table 5 and table 6 show the overall performance. Compared with the other seven methods, the DLSFL method showed a better prediction effect. As the data size increased, the accuracy rose and the log-loss declined. That is because more click is conducive to finding the relationship between the features.

LR: Logistic regression is a linear model with simple implementation and fast training speeds. It is widely used in online advertising estimation. However, this model does not consider feature interaction and performed worse than the other models.

Human_LR: This model uses LR as the estimation model and extracts features using artificial methods. Although the relationship between features can be obtained, hidden features cannot be found. Thus, as shown in table 5 and table 6, the performance of this model ranked eighth.

HPCM: Based on the Bayesian network model, this model uses matrix resolution to obtain the ad-query intrinsic association and the user-query related information between the advertisements using tensor decomposition. But the relationship between features cannot be obtained. Thus, as shown in table 5 and table 6, the performance of this model ranked seventh.

FM: This model was successfully applied to the user response prediction task. It explores feature interaction, which is effective on sparse data. However, this model is limited in mining high-order latent patterns or learning quality feature representations. As shown in table 5 and table 6, the performance of this model ranks sixth.

Wide&Deep: Wide&Deep combines a linear model and a deep mode. It learns high- and low-order feature interactions. Thus, as shown in table 5 and table 6, the performance of this model ranked second.

TABLE 5. (a) Overall CTR estimation for AUC performance. (b) Overall CTR estimation for AUC performance.

(a)

Data Size	AUC				
	LR	HPCM	Human_LR	FM	DLSFL
150,000	0.6851	0.6934	0.6931	0.7113	0.7205
200,000	0.6925	0.7003	0.7024	0.7235	0.7328
300,000	0.7034	0.7145	0.7115	0.7386	0.7476
500,000	0.7121	0.7226	0.7197	0.7422	0.7539
600,000	0.7183	0.7393	0.7285	0.7498	0.7663
750,000	0.7267	0.7485	0.7372	0.7537	0.7781
1000,000	0.7302	0.7502	0.7428	0.7649	0.7922

(b)

Data Size	AUC				
	FNN	Wide&Deep	DeepCross	DBNLR	DLSFL
150,000	0.7141	0.7196	0.7155	0.7164	0.7205
200,000	0.7266	0.7288	0.7255	0.7285	0.7328
300,000	0.7382	0.7402	0.7392	0.7403	0.7476
500,000	0.7435	0.7475	0.7447	0.7470	0.7539
600,000	0.7498	0.7563	0.7548	0.7563	0.7663
750,000	0.7560	0.7625	0.7582	0.7607	0.7781
1000,000	0.7617	0.7759	0.7703	0.7769	0.7922

TABLE 6. (a) Overall CTR estimation for log-loss performance. (B) Overall CTR estimation for log-loss performance.

(a)

Data Size	Log-Loss				
	LR	HPCM	Human_LR	FM	DLSFL
150,000	0.02955	0.02926	0.02932	0.02813	0.02773
200,000	0.02893	0.02884	0.02887	0.02782	0.02764
300,000	0.02886	0.02880	0.02882	0.02779	0.02659
500,000	0.02872	0.02869	0.02871	0.02768	0.02647
600,000	0.02764	0.02710	0.02762	0.02660	0.02633
750,000	0.02751	0.02748	0.02749	0.02647	0.02576
1000,000	0.02648	0.02643	0.02646	0.02637	0.02508

(b)

Data Size	Log-Loss				
	FNN	Wide&Deep	DeepCross	DBNLR	DLSFL
150,000	0.02792	0.02701	0.02784	0.02732	0.02773
200,000	0.02764	0.02678	0.02757	0.02708	0.02764
300,000	0.02734	0.02663	0.02689	0.02671	0.02659
500,000	0.02703	0.02656	0.02677	0.02664	0.02647
600,000	0.02655	0.02649	0.02693	0.02668	0.02633
750,000	0.02640	0.02638	0.02676	0.02643	0.02576
1000,000	0.02617	0.02603	0.02638	0.02615	0.02508

DeepCross: DeepCross applies a multi-layer residual network on a feature embedding cascade for learning feature interactions. The performance of this model ranks fourth.

FNN: FNN is a FM-initialized feed forward neural network. The FM pre-training strategy results in some limitations, such as the embedding parameters might be over affected by FM and the efficiency is reduced by the overhead introduced by the pre-training stage. The performance of this model ranks fifth.

DBNLR: DBNLR combines together the powerful data representation and feature extraction capability of Deep Belief Nets, with the advantage of simplicity of traditional Logistic Regression models. The performance of this model ranks third.

DLSFL: The DLSFL method performed best. The reasons are as follow: 1) The method exploits dimension reduction based on decomposition, and reduces the sparseness of the feature. 2) The AUFM model is based on deep learning. It can learn high- and low-order feature interactions simultaneously while sharing the same feature embedding and improving the performance of the CTR prediction model. Most features are categorical, so the AUFM model performance better.

V. CONCLUSION

In click-through rate prediction, the interaction between features is a key factor affecting the prediction rate. In this paper, based on the search advertising click data, we proposed a sparse feature learning method for advertising data from the perspective of feature learning (DLSFL). We used the reduced dimension method to cluster similar advertisements, queries and users and established a three-dimensional tensor model for the triad after dimension reduction. Then the low-order approximate tensor was obtained using Tucker decomposition. Aiming at the highly nonlinear relation between the features, this paper studied the feature learning method based on the depth learning. The AUFM model trained a deep component and an FFM component jointly. Performance improved based on these advantages: It learns both high- and low-order feature interactions and introduces a sharing strategy of feature embedding. We conducted extensive experiments to compare the effectiveness and efficiency of DLSFL with other methods. The method proposed in this paper can better improve the CTR. There are two interesting directions for future study. One is exploring strategies (such as introducing pooling layers) [36] to strengthen learning the most useful high-order feature interactions. The other is to use different dimension reduction methods, such as sparse filtering.

REFERENCES

[1] H.-T. Cheng *et al.*, "Wide & deep learning for recommender systems," in *Proc. 1st Workshop Deep Learn. Recommender Syst.*, 2016, pp. 7–10.

[2] O. Chapelle, E. Manavoglu, and R. Rosales, "Simple and scalable response prediction for display advertising," *Trans. Intell. Syst. Technol.*, vol. 5, no. 4, 2015, Art. no. 61.

[3] T. Graepel, T. Borchert, J. Q. Candela, and R. Herbrich, "Web-scale Bayesian click-through rate prediction for sponsored search advertising in Microsoft's bing search engine," in *Proc. 27th Int. Conf. Mach. Learn.* Madison, WI, USA: Omnipress, 2010, pp. 13–20.

[4] A.-Y. Zhou, M.-Q. Zhou, and X.-Q. Gong, "Computational advertising: A data-centric comprehensive Web application," *Jisuanji Xuebao, Chin. J. Comput.*, vol. 34, no. 10, pp. 1805–1819, 2011.

[5] M. Genzel and G. Kutyniok. (2016). "A mathematical framework for feature selection from real-world data with non-linear observations." [Online]. Available: https://arxiv.org/abs/1608.08852

[6] X. Wang *et al.*, "Click-through rate estimation for rare events in online advertising," in *Online Multimedia Advertising Techniques & Technologies*. IGI Global, 2011, pp. 1–12.

[7] H. B. Mcmahan *et al.*, "Ad click prediction: A view from the trenches," in *Proc. 19th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2013, pp. 1222–1230.

[8] O. Chapelle, "Modeling delayed feedback in display advertising," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2014, pp. 1097–1105.

[9] M. Jahrer, A. Toscher, J.-Y. Lee, J. Deng, H. Zhang, and J. Spoelstra, "Ensemble of collaborative filtering and feature engineered models for click through rate prediction," in *Proc. 18th ACM SIGKDD Conf. Knowl. Discovery Data Mining (KDDCup)*, Beijing, China, 2012, pp. 1222–1230.

[10] X. He, "Practical lessons from predicting clicks on ads at Facebook," in *Proc. 8th Int. Workshop Data Mining Online Advertising*, 2014, pp. 1–9.

[11] S. Shen, B. Hu, W. Chen, and Q. Yang, "Personalized click model through collaborative filtering," in *Proc. Int. Conf. Web Search Data Mining*, 2012, pp. 323–332.

[12] R. Kumar, S. M. Naik, V. D. Naik, S. Shiralli, S. V. G, and M. Husain, "Predicting clicks: CTR estimation of advertisements using Logistic Regression classifier," in *Proc. IEEE Int. Advance Comput. Conf. (IACC)*, Jun. 2015, pp. 1134–1138.

[13] A. I. Baqapuri and I. Trofimov. (2014). "Using neural networks for click prediction of sponsored search." [Online]. Available: https://arxiv.org/abs/1412.6601

[14] Q. Liu, F. Yu, S. Wu, and L. Wang, "A convolutional click prediction model," in *Proc. ACM 24th ACM Int. Conf. Inf. Knowl. Manage.*, 2015, pp. 1743–1746.

[15] Y. Shan, T. R. Hoens, J. Jiao, H. Wang, D. Yu, and J. C. Mao, "Deep crossing: Web-scale modeling without manually crafted combinatorial features," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 255–262.

[16] Y. Qu *et al.*, "Product-based neural networks for user response prediction," in *Proc. IEEE Int. Conf. Data Mining*, Dec. 2017, pp. 1149–1154.

[17] J. A. Hartigan and M. A. Wong, "Algorithm AS 136: A k-means clustering algorithm," *J. Roy. Stat. Soc. C (Appl. Statist.)*, vol. 28, no. 1, pp. 100–108, 2010.

[18] T. G. Kolda and J. Sun, "Scalable tensor decompositions for multi-aspect data mining," in *Proc. IEEE Int. Conf. Data Mining*, Dec. 2008, pp. 363–372.

[19] A. Szwabe, P. Misiorek, and M. Ciesielczyk, "Tensor-based modeling of temporal features for big data CTR estimation," in *Proc. Int. Conf., Beyond Databases, Archit. Struct.* Cham, Switzerland: Springer, 2017, pp. 16–27.

[20] M. Tong, Y. Pan, Z. Li, and W. Lin, "Valid data based normalized cross-correlation (VDNCC) for topography identification," *Neurocomputing*, vol. 308, pp. 184–193, Sep. 2018.

[21] H. Cheng and E. Cantú-Paz, "Personalized click prediction in sponsored search," in *Proc. 3rd ACM Int. Conf. Web Search Data Mining*, 2010, pp. 351–360.

[22] J. Zhu, Y. Song, D. Jiang, and H. Song, "A new deep-Q-learning-based transmission scheduling mechanism for the cognitive Internet of Things," *IEEE Internet Things J.*, vol. 5, no. 4, pp. 2375–2385, Aug. 2017.

[23] J. Chen, B. Sun, H. Li, H. Lu, and X.-S. Hua, "Deep CTR prediction in display advertising," in *Proc. MM*, 2016, pp. 811–820.

[24] S. Rendle and L. Schmidt-Thieme, "Pairwise interaction tensor factorization for personalized tag recommendation," in *Proc. ACM Int. Conf. Web Search Data Mining*, 2010, pp. 81–90.

[25] S. Rendle, "Factorization machines," in *Proc. IEEE Int. Conf. Data Mining*, Dec. 2010, pp. 995–1000.

[26] M. Jahrer, A. Töscher, J.-Y. Lee, J. Deng, H. Zhang, and J. Spoelstra, "Ensemble of collaborative filtering and feature engineered models for click through rate prediction," in *Proc. KDDCup Workshop*, 2012, pp. 1–7.

[27] G. E. Hinton and R. S. Zemel, "Autoencoders, minimum description length and Helmholtz free energy," in *Proc. Int. Conf. Neural Inf. Process. Syst.* Burlington, MA, USA: Morgan Kaufmann, 1993, pp. 3–10.

[28] L. Nie, D. Jiang, L. Guo, S. Yu, and H. Song, "Traffic matrix prediction and estimation based on deep learning for data center networks," *Globecom Workshops (GC Wkshps)*, Dec. 2016, pp. 1–6.

[29] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *J. Mach. Learn. Res.*, vol. 12, no. 7, pp. 257–269, 2011.

[30] W. S. Chin *et al.*, "A learning-rate schedule for stochastic gradient methods to matrix factorization," in *Proc. Pacific–Asia Conf. Knowl. Discovery Data Mining*. Cham, Switzerland: Springer, 2015, pp. 442–455.

[31] M. J. Effendi and S. A. Ali. (2017). "Click through rate prediction for contextual advertisment using linear regression." [Online]. Available: https://arxiv.org/abs/1701.08744

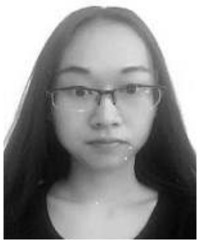
[32] M. Richardson, E. Dominowska, and R. Ragno, "Predicting clicks: Estimating the click-through rate for new ads," in *Proc. Int. Conf. World Wide Web*, 2007, pp. 521–530.

[33] H. Guo, R. Tang, Y. Ye, Z. Li, and X. He. (2017). "DeepFM: A factorization-machine based neural network for CTR prediction," pp. 1725–1731. [Online]. Available: https://arxiv.org/abs/1703.04247

[34] W. Zhang, T. Du, and J. Wang, "Deep learning over multi-field categorical data," in *Proc. European Conf. Inf. Retr.* Cham, Switzerland: Springer, 2016.

[35] J.-H. Chen *et al.*, "A new approach for mobile advertising click-through rate estimation based on deep belief nets," *Comput. Intell. Neurosci.*, vol. 2017, Oct. 2017, Art. no. 7259762.

[36] Y. Shen, X. He, J. Gao, L. Deng, and G. Mesnil, "A latent semantic model with convolutional-pooling structure for information retrieval," in *Proc. 23rd ACM Int. Conf. Inf. Knowl. Manage.*, 2014, pp. 101–110.



QIANQIAN WANG is currently pursuing the Ph.D. degree with the School of Information Science and Engineering, Shandong Normal University, China. Her research interests include the prediction systems, recommendation systems, and data mining.

FANG'AI LIU received the Ph.D. degree in computer science from the China Science College, in 2002. He is a CCF Senior Member. He is currently a Professor and a Doctoral Student Guide Architect with the School of Information Science and Engineering, Shandong Normal University, China. His research interests include the Internet, parallel processing, recommendation systems, prediction systems, and data mining.

SHUNING XING is currently pursuing the Ph.D. degree with the School of Information Science and Engineering, Shandong Normal University, China. Her research interests include recommendation systems and data mining.

XIAOHUI ZHAO received the M.S. degree in computer application from Jilin University, in 2005. She is currently pursuing the Ph.D. degree with the School of Information Science and Engineering, Shandong Normal University, China. Her research interests include social networks and data mining.

TIANLAI LI received the Ph.D. degree in computer science from Shandong Normal University, China, in 2015. He is currently a Lecturer with the School of Information Science and Engineering, Shandong Normal University. His research interests include big data and the networking.