Hindawi
Computational and Mathematical Methods in Medicine
Volume 2018, Article ID 8056541, 11 pages
https://doi.org/10.1155/2018/8056541

Research Article

A New Approach for Advertising CTR Prediction Based on Deep Neural Network via Attention Mechanism

Qianqian Wang,[1] **Fang'ai Liu**,[1] **Shuning Xing**,[1] **and Xiaohui Zhao**[2]

[1]*School of Information Science and Engineering, Shandong Normal University, Jinan, China*
[2]*School of Mathematical Science, Shandong Normal University, Jinan, China*

Correspondence should be addressed to Fang'ai Liu; lfa@sdnu.edu.cn

Received 29 March 2018; Accepted 1 August 2018; Published 13 September 2018

Academic Editor: Martti Juhola

Click-through rate prediction is critical in Internet advertising and affects web publisher's profits and advertiser's payment. 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, takes advantage of the attention mechanism in neural network modelling, and improves FM to make feature interactions contribute differently to the prediction. We utilize stack autoencoder to explore high-order feature interactions and use improved FM for low-order feature interactions to portray the nonlinear associated relationship of features. The experiment shows that our method improves the effect of CTR prediction and produces economic benefits in Internet advertising.

1. 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 rapid development of the mobile Internet and its wide range of applications, advertising has become one of the most successful business models in the world. Internet text advertising is regarded as a more effective advertising communication method due to its strong targeted communication and convenience of user clicking and has become an important income resource for many Internet companies. Some electronic commerce companies and search engine companies are seeking targeted advertising to increase their revenue.

In general, the display of online advertising can be seen as a three-party game between media, advertisers, and users. How to advertise to specific user groups is a key issue in the field of online advertising. Inappropriate advertising can lead to a decline in user experience. Advertising cannot achieve the desired effect, and the media can also be affected. Internet text advertising is usually in the form of text, and the advertisers get the opportunity to buy media ads through cost-per-click (CPC) [4]. In the CPC model, the click-through rate (CTR) is an important indicator to measure the effectiveness of advertising display and is a key factor in the three-party game. Therefore, the CTR estimation of advertising is a hot research direction in the field of computing advertising. In this paper, the click-through rate prediction of Internet text advertising shows the probability of predicting a user's click on a text under the current context environment. Due to the three-party information of advertising properties, user properties, and context environment, the CTR prediction is very complicated.

At present, the prediction of click-through rate for online advertising has attracted widespread attention from researchers in industry and academia. 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. Typically, a predictive task is formulated as estimating a function that maps predictor variables to some target. To build predictive models with these predictor variables, a common solution is to convert them to a set of binary features (a.k.a. feature vector) via one-hot encoding [5]. McMahan et al. [6] used the logistic regression [7] model to solve the CTR problems of Google Advertising. They adopted user information, advertising data, search keywords, and other features as the input of the model and proposed an online sparse learning algorithm to train the model. Chapelle [8] proposed a machine-learning framework based on the logistic regression in which advertisers, web publishers, users, and time characteristics were used as input to the model to solve the advertising CTR prediction for Yahoo. Dave and Varma [9] used the gradient boosting decision tree (GBDT) to predict the advertising CTR. They extracted similar features from advertising data and discovered implicit relationships between different features. Finally, they found out the nonlinear relationships between the predicted target and features. He et al. [10] introduced a fusion model which combines decision trees with logistic regression for predicting clicks on Facebook ads. The traditional CTR prediction model mainly depends on the design of features. The features of data are artificially selected and processed. The data have a complex mapping relationship, especially for meaningful data, and it is crucial to account for the interactions between features. Many successful solutions in both industry and academia largely rely on manually crafting combinatorial features [11], i.e., constructing new features by combining multiple predictor variables, also known as cross features. However, the power of such features comes at a high cost since it requires heavy engineering efforts and useful domain knowledge to design effective features. Factorization machines (FMs) [12] are a supervised learning approach that embed features into a latent space and model the interactions between features via inner product of their embedding vectors. Models based on degree-2 polynomial mapping and factorization machines are widely used for CTR prediction. The factorization-based prediction method field-aware factorization machines [13] were developed by Juan et al.

In recent years, deep learning [14, 15] has achieved very good results in the fields of speech recognition [16], image data processing [17], and natural language processing [18]. As a powerful approach to learning feature representation, deep neural networks have the potential to learn sophisticated feature interactions. Liu et al. [19] extended CNN for CTR prediction, but CNN-based models are biased towards the interactions between neighboring features. Zhang et al. [20] studied feature representations and proposed factorization machine-supported neural network (FNN). This model pretrained FM before applying DNN and thus limited by the capability of FM. He and Chua [21] proposed a novel neural factorization machine (NFM) for prediction under sparse setting. NFM combines the linearity of FM in modelling second-order feature interactions and the nonlinearity of neural network in modelling higher-order feature interactions. Despite great promise, we argue that FM can be hindered by its modelling of all factorized interactions with the same weight. In real-world applications, different predictor variables often have different predictive power. Not all features contain useful information for predicting the target. Therefore, the interaction of features with less useful information should be assigned a lower weight indicating that they contribute less to the prediction. However, FM lacks the ability to distinguish the importance of feature interactions, which will lead to suboptimal prediction.

Considering the high-dimensional sparsity of advertising data and the highly nonlinear association between features [22], a hybrid model for advertising CTR estimation based on stacked autoencoder, named Attention Stacked Autoencoder (ASAE), is proposed. Our model takes advantage of the attention mechanism in neural network modelling [23, 24] and improves FM to make feature interactions contribute differently to the prediction. More importantly, the importance of feature interactions is automatically learned from the data with any human domain knowledge. We explore data dimension reduction and identify the relationship between features. 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 the factorization machines. In Section 3, the sparse feature learning method for advertising data based on the ASAE model is proposed. In Section 4, we design the experiment and verify the prediction effect 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.

2. Factorization Machines

The factorization machines are originally proposed for learning feature interactions in the recommendation system. Given a real-valued feature vector $X \in \mathbb{R}^n$ where n denotes the number of features, FM estimates the target by modelling all interactions between each pair of features:

$$\widehat{y}\mathrm{FM}(x) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \widehat{w}_{ij} x_i x_j, \tag{1}$$

where w_0 is the global bias, w_i denotes the weight of the ith feature, and w_{ij} denotes the weight of the cross feature $x_i x_j$, which is factorized as $\widehat{w}_{ij} = v_i^T v_j$, where $v_i \in \mathbb{R}^k$ denotes the embedding vector for feature i and k denotes the size of the embedding vector. Besides linear (order-1) interactions among features, FM models pairwise (order-2) feature interactions as inner product of respective feature latent vectors. It can capture order-2 feature interactions much more effectively than previous approaches especially when the dataset is sparse. It is worth noting that FM models all

feature interactions in the same way: first, a latent vector v_i is shared in estimating all feature interactions that the ith feature involves; second, all feature interactions have the same weight of 1. However, it is common that not all features are relevant to the prediction. These interactions of irrelevant features can be considered as noise that does not contribute to the prediction. FM models all features using the same weights for interaction and may have a negative impact on generalization performance.

3. Click-through Rate Estimation Based on Deep Neural Network

One of the necessary steps in the click rate prediction system is to mine features that are highly correlated with the estimated task. To reduce the high sparseness of features and characterize the nonlinear association between features, we propose a sparse feature learning method for advertising data based on deep learning (DLSAE).

3.1. Data Dimensionality Reduction. Click log data contain many types of objects, such as users, queries, and advertisements. The relationship between these objects is very complex. The same objects have similarity, and 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 [25] 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 $A = \{a_1, a_2, \ldots, a_m\}$ and $Q = \{q_1, q_2, \ldots, q_N\}$. Then, we take K samples from the advertising set randomly as the initial point of the cluster center, record as $T = \{t_1, t_2, \ldots, t_k\}$. Next, Equation (2) 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 represented by K_u, K_a, and K_q, respectively. Finally, the number of users, ads, and queries in the dataset is reduced from M_u, M_a, M_q to K_u, K_a, K_q:

$$\mathrm{Dis}(a_i, t_j) = \sqrt{\sum \left(W_{a_i} - W_{q_j} \right)^2}, \tag{2}$$

where W_{a_i} is the weight of a_i, W_{q_j} is the weight of t_j, and $\mathrm{Dis}(a_i, t_j)$ is the distance between a_i and t_j.

There is a ternary relationship between the user-query-ad in the click log data. In this paper, we use the three-dimensional tensor structure model [26, 27] to represent the user, query, and advertisement. Then, 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 the Tucker factorization. Equation (2) is the decomposition formula.

$$\begin{aligned} X = [G; A, B, C] &= G \times_u A \times_q B \times_a C \\ &= \sum_{p=1}^{P} \sum_{m=1}^{M} \sum_{r=1}^{R} g_{pmr} u_p \circ q_m \circ a_r, \end{aligned} \tag{3}$$

where G represents the core tensor of tensor X. 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 1 is a schematic diagram of the Tucker decomposition. The purpose of the Tucker decomposition is to find an approximate tensor $\widehat{X}$ with the original tensor X and to retain the original tensor information and structural information to the greatest extent. The minimization formula is shown below:

$$\min \left\| X - \widehat{X} \right\|, \quad \widehat{X} = G \times_u A \times_q B \times_a C = [G; A, B, C]. \tag{4}$$

Equation (4) is the objective optimization function. According to Equation (3), the expression of the core tensor can be obtained as follows:

$$G = X \times_u A^T \times_q B^T \times_a C^T, \tag{5}$$

and the objective function can be written in a squared form:

$$\begin{aligned} \|X - [G; A, B, C]\|^2 &= \|X\|^2 - 2\langle X \times_u A^T \times_q B^T \times_a C^T, G \rangle \\ &\quad + \|G\|^2, \\ &= \|X\|^2 - 2\langle G, G \rangle + \|G\|^2, \\ &= \|X\|^2 - \|G\|^2, \\ &= \|X\|^2 - \left\| X \times_u A^T \times_q B^T \times_a C^T \right\|^2. \end{aligned} \tag{6}$$

Therefore, the objective function is transformed to

$$\begin{aligned} &\max \left\| X \times_u A^T \times_q B^T \times_a C^T \right\|^2, \\ &\left\| A^T W \right\|, \quad W = X \times_q B^T \times_a C^T, \\ &\left\| B^T W \right\|, \quad W = X \times_u A^T \times_a C^T, \\ &\left\| C^T W \right\|, \quad W = X \times_u A^T \times_q B^T. \end{aligned} \tag{7}$$

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, expand the tensor X into a matrix on the user, query, and advertising dimensions, respectively, as X_1, X_2, X_3 and apply SVD on X_1, X_2, X_3:

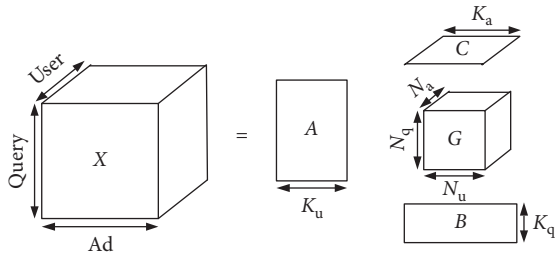

Figure 1: Schematic diagram of the Tucker decomposition.

$$X_1 = A \cdot G_1 \cdot V_1^T,$$
$$X_2 = B \cdot G_2 \cdot V_2^T, \quad (8)$$
$$X_3 = C \cdot G_3 \cdot V_3^T,$$

where G_1, G_2, G_3 are the diagonal singular value matrices obtained using singular value decomposition of the matrices X_1, X_2, X_3. g_1, g_2, g_3 are the dimensions of the singular value matrix A, B, C. The dimensions g_1, g_2, g_3 are obtained by calculating the diagonal singular values of G_1, G_2, G_3 in proportion. In the process of reducing the dimensions, the proportion of excluded singular values is set to 50% in this paper. Therefore, the calculation of the core tensor after dimension reduction is as follows:

$$G' = X \times_u A_{r1}^T \times_q B_{r2}^T \times_a C_{r3}^T,$$
$$X' = G' \times_u A_{r1} \times_q B_{r2} \times_a C_{r3}. \quad (9)$$

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 greatly and improved the efficiency and precision.

3.2. Feature Composition Analysis of the Input Layer. There is a high degree of nonlinear correlation between the features in advertising data. Although the approximate tensor of the original tensor is reduced by the 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. 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 using 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 Characteristics*. The ID class 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, it is necessary to attribute the property class as learning the input layer feature further. Commonly used attribute class features are user's URL, user's gender, user's age, and advertising time to trigger and query keywords.

(3) *Statistical Characteristics*. The statistical feature uses historical data statistics information to provide an estimate for the forecasting model. The statistical characteristics of the text consist of the number of advertising histories, the number of clicks on the advertising history, and the click-through rate after the advertising position normalization, denoted by Shows, Clicks, and COEC In the experiment, the input layer feature of the ASAE model is shown in Figure 2.

3.3. Study on CTR Prediction via Attention Mechanism Based on the Stacked Autoencoder

3.3.1. Attentional Factorization Machines. Since the attention mechanism has been introduced into neural network modelling, it has been widely used in many tasks. On the basis of FM, Figure 3 shows the neural network structure of attentional factorization machines (AFM). The input layer and the embedding layer are the same as the FM; the input features are represented with sparse features, and each nonzero feature item is embedded in the dense vector. Formally, let the set of nonzero features in the feature vector x be χ and the output of the embedding layer be $\lambda = \{v_i x_i\}_{i \in \chi}$. In the interaction layer, we can represent the output as a set of vectors:

$$f_{\text{in}}(\lambda) = \left\{\left(v_i \odot v_j\right) x_i x_j\right\}_{(i,j) \in R_x}, \quad (10)$$

where $\odot$ denotes the element wise product of two vector and $R_x = \{(i, j)\}_{i \in \chi, j \in \chi, j > i}$. By defining the interaction layer, we express FM under the neural network architecture. We compress $f_{\text{in}}(\lambda)$ with a sum pooling. Then use the full connection layer to establish it and get the prediction score:

$$\widehat{y} = p^T \sum_{(i,j) \in R_x} \left(v_i \odot v_j\right) x_i x_j + b, \quad (11)$$

where $p \in \mathbb{R}^k$ denotes the weights and $b \in \mathbb{R}$ denotes the bias for the prediction layer.

The attention mechanism has been widely used in many tasks. The idea is to allow different parts to contribute differently when compressing them to a single

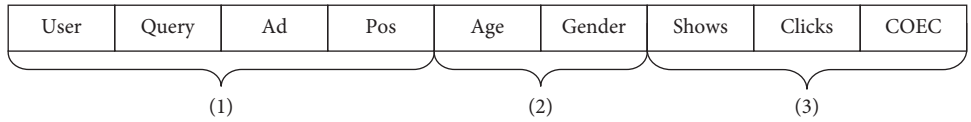

Figure 2: The features of the input layer.

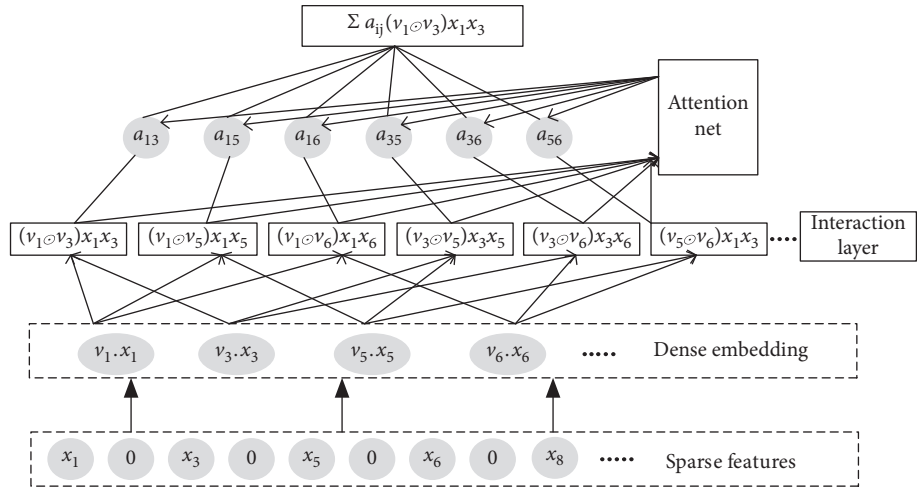

Figure 3: The structure of attentional factorization machines.

representation. We use attention mechanisms for feature interaction:

$$f_{\text{att}}\left(f_{\text{in}}(\lambda)\right) = \sum_{(i,j)\in R_x} a_{ij}\left(v_i \odot v_j\right)x_i x_j, \tag{12}$$

where a_{ij} is the attention score for feature interaction $\widehat{w}_{ij}$, and it can be interpreted as the importance of $\widehat{w}_{ij}$ in predicting goals. a_{ij} can be learned by minimizing the loss function, but the attention scores of interactions that never occur in training data cannot be estimated. In order to solve the generalization problem, we use the multilayer perceptron (MLP) to further parameterize the attention score, which we call the attention network. The input of the attention network is an interaction vector of two features and can encode their interaction information in the embedding space. In general, the attention network is defined as

$$\begin{aligned} a'_{ij} &= h^T \text{Re}\, LU\left(W\left(v_i \odot v_j\right)x_i x_j + b\right), \\ a_{ij} &= \frac{\exp\left(a'_{ij}\right)}{\sum_{(i,j)\in R_x}\exp\left(a'_{ij}\right)}, \end{aligned} \tag{13}$$

where $W \in \mathbb{R}^{s\times k}, b \in \mathbb{R}^s$, and $h \in \mathbb{R}^s$ are the model parameters and s is the hidden layer size of the attention network, which we call the attention factor. Rectifiers are used as the activation function for attention scores and show good performance empirically. The output of the attention layer is a k-dimensional vector that compresses all feature interactions in the embedding space by differentiating their importance. We give the overall formulation of attentional factorization machines as

$$y_{\text{AFM}}(x) = w_0 + \sum_{i=1}^{n} w_i x_i + p^T \sum_{i=1}^{n}\sum_{j=i+1}^{n} a_{ij}\left(v_i \odot v_j\right)x_i x_j, \tag{14}$$

where a_{ij} has been defined in Equation (13).

For the part of the attention network, which is a single-layer MLP, we apply L_2 regularization on the weight matrix W to prevent possible overfitting. In other words, the actual objective function we optimize is

$$L = \sum_{x\in\Upsilon}\left(\widehat{y}_{\text{AFM}}(x) - \widehat{y}(x)\right)^2 + \lambda\|W\|^2, \tag{15}$$

where Υ denotes the set of training instances and λ controls the regularization strength.

3.3.2. Stacked Autoencoder. The autoencoder (AE) [28] is a kind of the neural network model that automatically learns features from data without supervision. 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 4. In Figure 4, w is the connection weight of the two layers and b is the bias. In the input layer and hidden layer, the AE model will convert input data to each node of the hidden layer. In the hidden layer and the reconstruction layer, the value of the nodes in the hidden layer is reconstructed and the output data are obtained.

The stacked autoencoder (SAE) [29] is a kind of network that consists of n AE stacks from the bottom to the top, as shown in Figure 5. The input data of the bottom AE are x. When the training of the bottom AE is finished, the feature

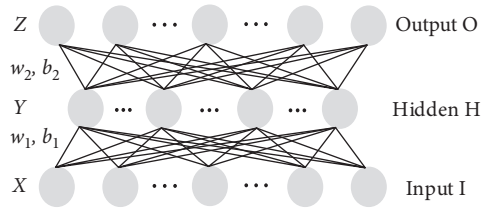

Figure 4: The architecture of the autoencoder.

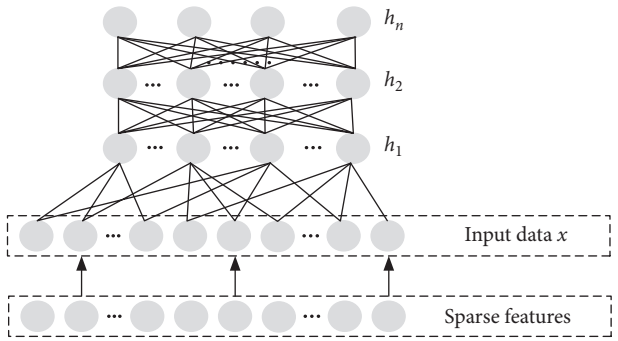

Figure 5: The structure of the stacked autoencoder.

of the hidden layer is obtained and can be represented by h_1. Then, h_1 is regarded as the input data of the second AE layer, which is trained and provides the features of the hidden layer and is represented by h_2. This process is repeated until h_n is obtained.

The related definition of the jth node in the hidden layer of 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). $\text{net}_j^h = b_j^h + \sum_{i=1}^{s_x} w_{ji}^h o_i^x$ 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 the following formula:

$$o_j^h = f\left(\text{net}_j^h\right) = \sigma\left(b_j^h + \sum_{i=1}^{s_x} w_{ji}^h o_i^x\right). \tag{16}$$

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 the following formula:

$$\begin{aligned} o_j^o &= g\left(\text{net}_j^o\right) = g\left(b_j^o + \sum_{i=1}^{s_h} w_{ji}^o o_i^h\right), \\ &= \sigma\left(b_j^o + \sum_{i=1}^{s_h} w_{ji}^o \left(\sigma\left(b_i^h + \sum_{k=1}^{s_o} w_{ik}^h o_k^x\right)\right)\right). \end{aligned} \tag{17}$$

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 the following formula according to the chain rule:

$$\begin{aligned} \delta_j^h &= \frac{\partial J}{\partial \text{net}_j^h} = \frac{\partial J}{\partial o_j^h}\frac{\partial o_j^h}{\partial \text{net}_j^h}, \\ &= \sum_{i=1}^{s_o}\left(\frac{\partial J}{\partial \text{net}_i^o}\frac{\partial \text{net}_i^o}{\partial o_j^h}\right)\cdot\frac{\partial o_j^h}{\partial \text{net}_j^h}, \\ &= \left(\sum_{i=1}^{s_o}\delta_i^o w_{ji}^o\right)\frac{\partial \delta\left(\text{net}_j^h\right)}{\partial \text{net}_j^h}, \\ &= \left(\sum_{i=1}^{s_o}\delta_i^o w_{ji}^o\right)\delta\left(\text{net}_j^h\right)\left(1-\delta\left(\text{net}_j^h\right)\right), \\ &= \left(\sum_{i=1}^{s_o}\delta_i^o w_{ji}^o\right)o_j^h\left(1-o_j^h\right). \end{aligned} \tag{18}$$

The parameters w_{ji}^l and b_j^l can be calculated by formulae (19) and (20):

$$\frac{\partial J}{\partial w_{ji}^l} = \frac{\partial J}{\partial \text{net}_j^l}\frac{\partial \text{net}_j^l}{\partial w_{ji}^l} = \delta_j^l \cdot o_i^{l-1}, \tag{19}$$

$$\frac{\partial J}{\partial b_j^l} = \delta_j^l. \tag{20}$$

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

$$\begin{aligned} w_{ji}^l &= w_{ji}^l - \beta\frac{\partial J}{\partial w_{ji}^l} = w_{ji}^l - \beta\cdot\delta_j^l\cdot o_i^{l-1}, \\ b_j^l &= b_j^l - \beta\frac{\partial J}{\partial b_j^l} = b_j^l - \beta\cdot\delta_j^l. \end{aligned} \tag{21}$$

The SAE is a generative model that is composed of a stack of autoencoders. This method relies on the training algorithm of the autoencoder to initialize the parameters of a stacked autoencoder. Each new layer is stacked on top of the current autoencoder. The process gradually refines the previously learned information and further discovers more complex features. After this, a dense real-value feature vector is generated, which is finally fed into the sigmoid function for CTR prediction:

$$y_{\text{SAE}} = \sigma\left(W^{h+1}\cdot X^h + b^{h+1}\right), \tag{22}$$

where W is the model weight, b is the bias, and h is the number of hidden layer.

This paper selects the square error as the objective function and adopts the gradient descent [30, 31] to train the parameters, and the objective function can be described by the following formula:

$$J(X,O) = \frac{1}{2}\sum_{i=1}^{n}\left\|X^{(i)} - O^{(i)}\right\|^2. \tag{23}$$

3.3.3. ASAE Model. The ASAE model consists of two components, AFM component and SAE component, which share the same input. The graphical model of the ASAE model is shown in Figure 6. It is able to learn feature interactions of all orders in an end-to-end manner, without any feature engineering besides raw features. x_i is fed in AFM component to model order-2 feature interactions and distinguish their importance. x_i is fed in SAE component to model high-order feature interactions, and it can generalize better to unseen feature combinations through low-dimensional dense embedding learned for the sparse features. All parameters are trained jointly for the combined prediction model:

$$\widehat{y} = \text{sigmoid}\left(y_{\text{AFM}} + y_{\text{SAE}}\right), \tag{24}$$

where $\widehat{y} \in (0, 1)$ is the predicted CTR, y_{AFM} is the output of the AFM component, and y_{SAE} is the output of the SAE component.

4. Experiments

4.1. Datasets. We perform experiments with two publicly accessible datasets: Frappe [32] and SIGKDD Cup2012 track2. The Frappe dataset has been used for context-aware recommendation. It contains 96,215 app usage logs of users under different contexts. Each log contains 8 context variables, including app ID, user ID, city, and daytime. We convert each log into a feature vector with one-hot encoding, resulting in 5,479 features in total. We split dataset into the training set and testing set using a random partition method by the ratio of 8 to 1. The target value of 1 indicates that the user has used the application in context.

The KDD2012 CUP track2 corresponding research question is based on the actual click data information to predict the click rate of the advertisement. The training dataset provided by the competition has a total of 149,639,105 records, and the size of 9.8 GB. In addition to the number of click and the number of displays, the test dataset is consistent with the training dataset, a total of 20,257,594 records, 1.28 GB in size. After data cleaning and data preprocessing, a total of 3.5 million samples were randomly selected from the candidate dataset for the experiment. Table 1 summarizes the statistics of the final evaluation datasets.

In the KDD2012 CUP track2 dataset, the samples of seven different scale datasets are 150000, 200000, 300000, 500000, 600000, 750000, and 1 million. The training data are grouped randomly, and the final result is the average of all the experimental results to ensure the reliability of the experimental results.

4.2. Evaluation Index. We use two evaluation metrics in our experiments: AUC (area under ROC) and Logloss (cross entropy). The curve in AUC usually means the receiver operating characteristic (ROC) [33], which is usually used to measure performance of two-class classifier. The CTR prediction is a classic binary classification method based on whether the advertising is clicked. The value of AUC is usually between [0.5, 1). The larger the value of AUC becomes, the more accurate the advertising CTR prediction is.

4.3. Baseline Models. We compare the ASAE model with the following methods that are designed for sparse data prediction:

FM [34]: FM is successfully applied to the recommended system and user response prediction task. FM explores feature interaction, which is effective on sparse data

FNN [20]: FNN is a FM-initialized feedforward neural network. It is able to capture high-order latent patterns of multifield categorical data.

CCPM [19]: convolutional click prediction model (CCPM) is based on convolution neural network. It can extract local-global key features from an input instance with varied elements, which can be implemented for single advertising impression and sequential advertising impression.

Deep cross [11]: it applies a multilayer 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.

Wide and deep [35]: 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.

4.4. Analysis of Experimental Results. This section evaluates the ASAE model from two perspectives: (1) discussing the impact of relevant parameters and (2) comparing the ASAE model with five existing prediction models.

4.4.1. Impact of Parameters. Dropout [36] refers to the probability that a neuron is kept in the network. Dropout is a regularization technique to compromise the precision and the complexity of the neural network. We set the dropout to be 0.1 to 0.8. As shown in Figure 7, the optimal dropout ratio on Frappe is 0.3. The result shows that adding reasonable randomness to model can strengthen model's robustness.

The number of network layers h in the depth learning phase has a direct effect on the final estimate of the model. Therefore, this paper experimented with parameters to select the better combination of parameters. In the Frappe dataset, as presented in Figure 8, increasing number of hidden layers improves the performance of the models at the beginning. However, with the increasing of the number of hidden layers, the model performance is degraded. This phenomenon is because of overfitting. We can see from Figure 8, the highest AUC value is obtained when the number of hidden layers is 4 in Frappe dataset.

The number of iterations (iter) in the training phase have a direct effect on the final estimation of the model. In the KDD dataset, we used a set of sampled data training models

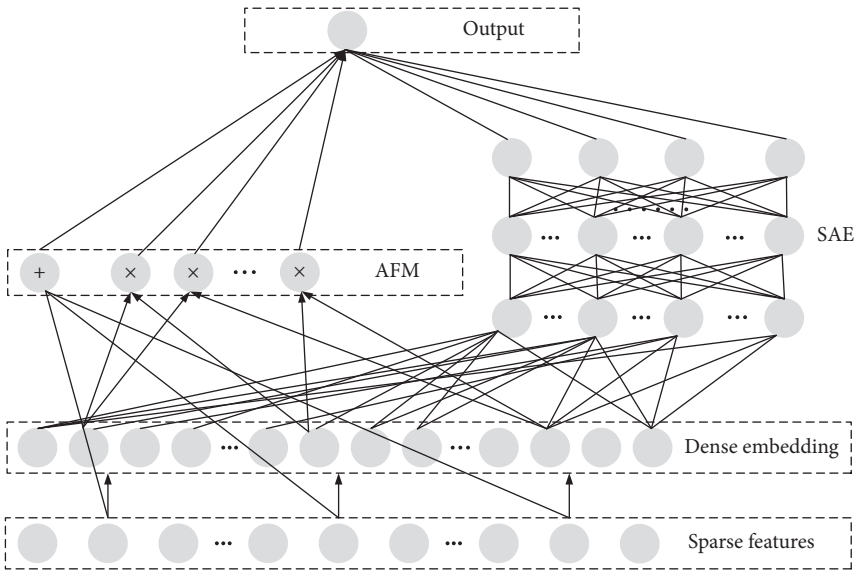

Figure 6: The architecture of ASAE.

Table 1: Statistics of the evaluation datasets.

Dataset	Instance	Feature	User	Item
Frappe	276,672	5,479	1,028	5,183
KDD	3,500,000	192,886	127,385	150,672

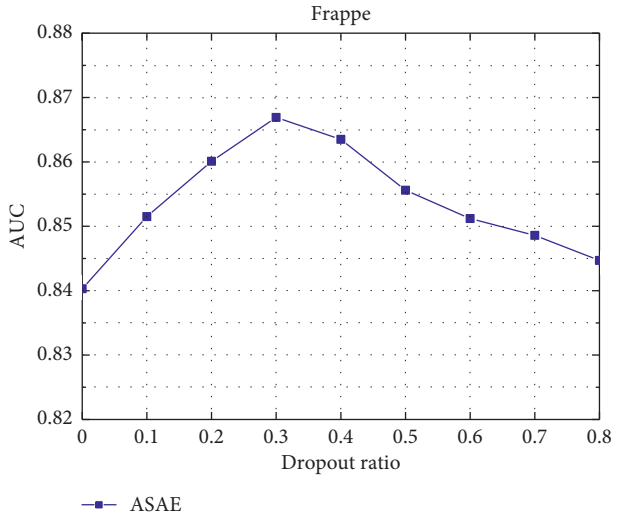

Figure 7: The influence of the dropout ratio.

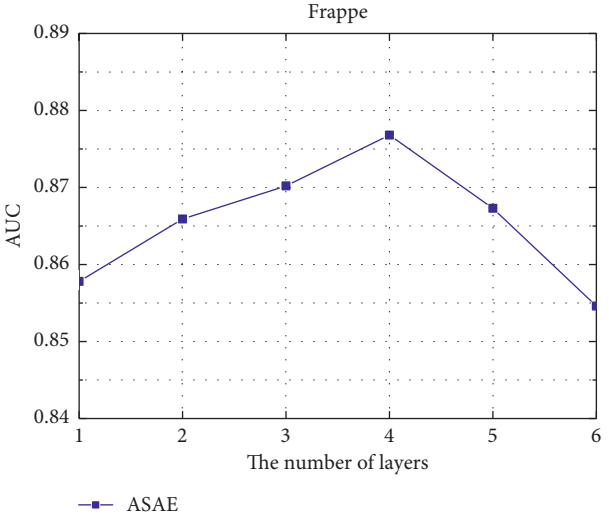

Figure 8: The influence of the number of hidden layers.

with a data size of 500,000 samples, tested on the test set, and we used it to select the best parameters. While fixing the network layer number ($h = 2, 3, 4, 5, 6$) of the model, we can analyze the effect of different iter on the model performance, and the results are shown in Table 2.

In accordance with Table 2, Figure 9 reflects the AUC change for different network hidden layers h and LR model iterations for iter. As seen in Figure 9, when the number of iterations is 90 to 120, the AUC values of several curves stabilized. Therefore, in the comparison experiment, 115 is

Table 2: The relationship between the number of network layers and iter.

iter	AUC				
	$h = 2$	$h = 3$	$h = 4$	$h = 5$	$h = 6$
10	0.6732	0.6717	0.6812	0.6771	0.6761
20	0.6834	0.6735	0.6968	0.6811	0.6814
30	0.6949	0.6812	0.7069	0.6869	0.6933
50	0.7132	0.7073	0.7231	0.7103	0.7058
70	0.7285	0.7191	0.7349	0.7215	0.7245
90	0.7372	0.7311	0.7482	0.7355	0.7401
100	0.7419	0.7429	0.7516	0.7481	0.7464
110	0.7432	0.7441	0.7543	0.7521	0.7508
130	0.7441	0.7452	0.7540	0.7529	0.7486
150	0.7456	0.7463	0.7524	0.7511	0.7502

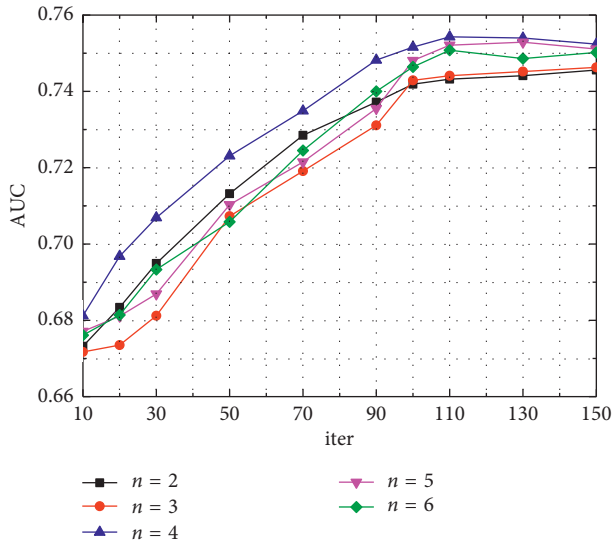

Figure 9: AUC comparison of different iterations and different network layers.

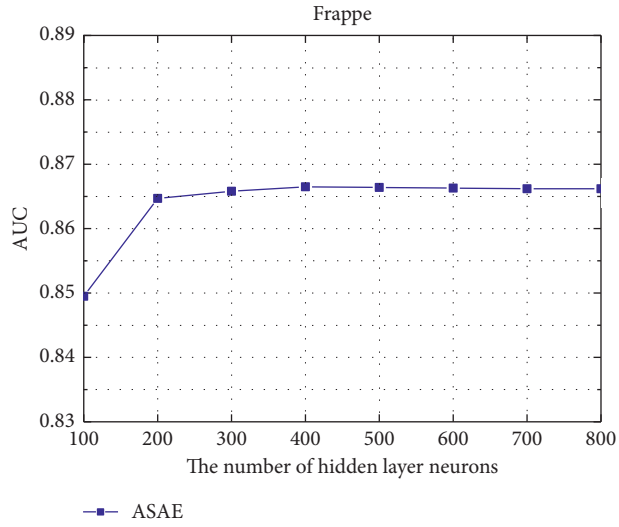

Figure 10: The influence of the number of neurons.

chosen as the number of iterations for training the prediction model. As shown in Figure 9, the curve fluctuates greatly with the change of iterations, and $h = 4$ is relatively stable, so we choose $h = 4$.

When other factors remain constant, the number of hidden layer units in the ASAE has a huge impact on network performance and the direct cause of the problem is extremely important. However, this figure does not have a general parameter adjustment method in theory. Therefore, in this part, we carry out experiments on the effect of the number of hidden layer neurons. As we can observe from Figure 10, increasing the number of neurons per layer does not always bring benefit. For example, when the number of neurons per layer is increased from 400 to 800, the ASAE performs stably. This is because the complicated model is easy to overfit. In our experiment, 200 to 400 neurons per layer is a good choice.

4.4.2. Performance Comparison. We trained the models on the two datasets and evaluated the estimated results on the same test set. Tables 3–5 describe the estimated results for the different methods at different datasets.

Tables 3–5 show the overall performance. Compared with the other five methods, the ASAE model showed a better prediction effect. As the data size increased, the accuracy rose and the logloss declined.

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 Tables 3–5, the performance of this model is worst in all comparison models.

FNN: FNN is a FM-initialized feedforward neural network. The FM pretraining 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 pretraining stage. From Tables 3–5, we can see that the performance of FNN ranked fifth.

CCPM: this model is based on convolution neural network for single and sequential advertising impression. However, this model highly relies on feature alignment and is a lack of interpretation. Thus, as shown in Tables 3–5, the performance of this model ranked fourth.

Deep cross: the deep cross is the deepest method that stacks 10 layers above the embedding layer in all compared methods. From Tables 3–5, we can see that the performance of this model ranked third due to the problems of overfitting.

Table 3: The performance of AUC and Logloss in different models.

	Frappe					
	FM	FNN	CCPM	Deep cross	Wide and deep	ASAE
AUC	0.7935	0.8012	0.8178	0.8266	0.8293	0.8386
Logloss	0.03842	0.03473	0.03208	0.03127	0.03015	0.02813

Table 4: The AUC performance at different data sizes in KDD dataset.

Data size	AUC					
	FM	FNN	CCPM	Deep cross	Wide and deep	ASAE
150,000	0.6849	0.6941	0.6938	0.7135	0.7193	0.7212
200,000	0.6923	0.7012	0.7027	0.7263	0.7275	0.7355
300,000	0.7033	0.7186	0.7134	0.7397	0.7413	0.7482
500,000	0.7119	0.7235	0.7204	0.7434	0.7486	0.7547
600,000	0.7181	0.7398	0.7291	0.7503	0.7578	0.7672
750,000	0.7255	0.7490	0.7467	0.7549	0.7649	0.7793
1,000,000	0.7315	0.7517	0.7528	0.7652	0.7763	0.7981

Table 5: The Logloss performance at different data sizes in KDD dataset.

Data size	Logloss					
	FM	FNN	CCPM	Deep cross	Wide and deep	ASAE
150,000	0.02967	0.02925	0.02941	0.02826	0.02714	0.02733
200,000	0.02898	0.02893	0.02882	0.02793	0.02665	0.02704
300,000	0.02886	0.02872	0.02876	0.02768	0.02657	0.02658
500,000	0.02875	0.02864	0.02872	0.02759	0.02652	0.02639
600,000	0.02783	0.02715	0.02713	0.02674	0.02645	0.02630
750,000	0.02752	0.02739	0.02728	0.02631	0.02627	0.02543
1,000,000	0.02677	0.02642	0.02634	0.02628	0.02606	0.02501

Wide and deep: wide and deep combines a linear model and a deep model. It learns high- and low-order feature interactions. There is a need for expertise feature engineering on the input to the "wide" part. Thus, as shown in Tables 3–5, the performance of this model ranked second.

ASAE: the ASAE model performed best. The reasons are as follows. (1) The input of the model exploits dimension reduction based on decomposition and reduces the sparseness of data. (2) The model takes advantage of the attention mechanism in neural network modelling and improves FM to make feature interactions contribute differently to the prediction. (3) We use improved FM for low-order feature interactions, and stacked autoencoder is used for high-order feature interactions. The model more effectively mines the relationship between features, which can improve the CTR.

5. Conclusions

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 (DLSAE). We used the reduced dimension method to cluster similar advertisements, queries, and users and established a three-dimensional tensor model for the trial after dimension reduction. Then the low-order approximate tensor was obtained using the Tucker decomposition. Aiming at the highly nonlinear relation between the features, we proposed a hybrid model (ASAE) for advertising CTR estimation based on the stacked autoencoder from the perspective of feature learning. The ASAE model trains a deep component and an AFM component jointly. Performance improved based on these advantages. First, this model does not need any pretraining. Second, it learns both high- and low-order feature interactions, introduces a sharing strategy of feature embedding, and more effectively mines the relationship between features. Last but not least, the proposed model distinguishes the importance of features and makes click-through rate predictions more accurate. More importantly, the importance of feature interactions is automatically learned from the data with any human-domain knowledge. We conducted extensive experiments in two datasets to compare the effectiveness of ASAE with other models. There are two interesting directions for future study. One is exploring a convolutional click prediction model based on CNN for single and sequential advertising impression. And another we are interested in exploring the pooling for recurrent neural networks (RNNs) for sequential data modelling.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare no conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (nos. 61572301 and 61772321), the Innovation Foundation of Science and Technology Development Center of Ministry of Education and New H3C Group (2017A15047), CERNET Innovation Project (NGII20170508), and the Open Research Fund from Shandong Provincial Key Laboratory of Computer Network (no. SDKLCN-2016-01).

References

[1] H. T. Cheng, L. Koc, J. Harmsen et al., "Wide & deep learning for recommender systems," in *Proceedings of Workshop on Deep Learning for Recommender Systems*, pp. 7–10, ACM, Boston, MA, USA, September 2016.

[2] O. Chapelle, R. Rosales, and R. Rosales, "Simple and scalable response prediction for display advertising," *ACM Transactions on Intelligent Systems and Technology*, vol. 5, no. 4, pp. 1–34, 2015.

[3] T. Graepel, T. Borchert, and R. Herbrich, "Web-scale Bayesian click-through rate prediction for sponsored search advertising in Microsoft's Bing search engine," in *Proceedings of International Conference on Machine Learning*, pp. 13–20, Omnipress, Haifa, Israel, June 2010.

[4] M. Richardson, E. Dominowska, and R. Ragno, "Predicting clicks: estimating the click-through rate for new ads," in *Proceedings of 16th International Conference on World Wide Web*, Alberta, Canada, May 2007.

[5] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proceedings of 26th International Conference on World Wide Web*, Perth, Australia, April 2017.

[6] H. B. McMahan, D. Golovin, S. Chikkerur et al., "Ad click prediction: a view from the trenches," in *Proceedings of 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Chicago, IL, USA, August 2013.

[7] M. J. Effendi and S. A. Ali, "Click through rate prediction for contextual advertisement using linear regression," *International Journal of Computer Science and Information Security*, vol. 14, no. 11, 2017.

[8] O. Chapelle, "Modeling delayed feedback in display advertising," in *Proceedings of 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, NY, USA, August 2014.

[9] K. Dave and V. Varma, *Predicting the Click-through Rate for Rare/New Ads*, Center for Search and Information Extraction Lab International Institute of Information Technology, Hyderabad, India, 2010.

[10] X. He, S. Bowers, J. Q. Candela et al., "Practical lessons from predicting clicks on ads at Facebook," in *Proceedings of Eighth International Workshop on Data Mining for Online Advertising*, New York, NY, USA, August 2014.

[11] Y. Shan, T. R. Hoens, J. Jiao, H. Wang, D. Yu, and J. Mao, "Deep crossing: web-scale modeling without manually crafted combinatorial features," in *Proceedings of 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 255–262, San Francisco, CA, USA, June 2016.

[12] S. Rendle, "Factorization machines with LIBFM," *ACM Transactions on Intelligent Systems and Technology*, vol. 3, no. 3, p. 57, 2012.

[13] Y. Juan, Y. Zhuang, W.-S. Chin, and C.-J. Lin, "Field-aware factorization machines for CTR prediction," in *Proceedings of 10th ACM Conference on Recommender Systems*, Boston, MA, USA, September 2016.

[14] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[15] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[16] A. Graves, A. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Vancouver, Canada, March 2013.

[17] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–80, 2012.

[18] Y. Shen, X. He, J. Gao, L. Deng, and G. Mesnil, "A latent semantic model with convolutional-pooling structure for information retrieval," in *Proceedings of 23rd ACM International Conference on Conference on Information and Knowledge Management*, Shanghai, China, November 2014.

[19] Q. Liu, F. Yu, S. Wu, and L. Wang, "A convolutional click prediction model," in *Proceedings of 24th ACM International on Conference on Information and Knowledge Management*, Melbourne, Australia, October 2015.

[20] W. Zhang, T. Du, and J. Wang, "Deep learning over multi-field categorical data," in *Proceedings of European Conference on Information Retrieval*, Padua, Italy, March 2016.

[21] X. He and T.-S. Chua, "Neural factorization machines for sparse predictive analytics," in *Proceedings of 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, Tokyo, Japan, August 2017.

[22] M. Genzel and G. Kutyniok, "A mathematical framework for feature selection from real-world data with non-linear observations," http://arxiv.org/abs/1608.08852, 2016.

[23] J. Chen, H. Zhang, X. He, L. Nie, W. Liu, and T.-S. Chua, "Attentive collaborative filtering: multimedia recommendation with item-and component-level attention," in *Proceedings of 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, Tokyo, Japan, August 2017.

[24] L. Chen, H. Zhang, J. Xiao et al., "SCA-CNN: spatial and channel-wise attention in convolutional networks for image captioning," http://arxiv.org/abs/1611.05594, 2016.

[25] J. A. Hartigan and M. A. Wong, "Algorithm AS 136: A k-means clustering algorithm," *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, vol. 28, no. 1, pp. 100–108, 1979.

[26] T. G. Kolda and J. Sun, "Scalable tensor decompositions for multi-aspect data mining," in *Proceedings of Eighth IEEE International Conference on Data Mining*, pp. 363–372, Pisa, Italy, December 2008.

[27] A. Szwabe, P. Misiorek, and M. Ciesielczyk, "Tensor-based modeling of temporal features for big data CTR estimation," in *Proceedings of International Conference: Beyond Databases, Architectures and Structures*, pp. 16–27, Ustron, Poland, April 2017.

[28] J. Chen, B. Sun, H. Li, H. Lu, and X.-S. Hua, "Deep CTR prediction in display advertising," in *Proceedings of 2016 ACM on Multimedia Conference-MM'16*, New York, NY, USA, October 2016.

[29] P. Vincent, H. Larochelle, I. Lajoie et al., "Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion," *Journal of Machine Learning Research*, vol. 11, pp. 3371–3408, 2010.

[30] M. Jahrer, A. T¨oscher, J.-Y. Lee, J. Deng, H. Zhang, and J. Spoelstra, "Ensemble of collaborative filtering and feature engineered model for click through rate prediction," in *Proceedings of KDD Cup 2012 Workshop*, Beijing, China, August 2012.

[31] W. S. Chin, Y. Zhuang, Y. C. Juan et al., "A learning-rate schedule for stochastic gradient methods to matrix factorization," in *Advances in Knowledge Discovery and Data Mining*, Springer International Publishing, Cham, Switzerland, pp. 442–455, 2015.

[32] L. Baltrunas, K. Church, A. Karatzoglou, and N. Oliver, "Frappe: understanding the usage and perception of mobile app recommendations in-the-wild," http://arxiv.org/abs/1505.03014, 2015.

[33] T. Akimova, M. H. Levine, U. H. Beier, and W. W. Hancock, "Standardization, evaluation, and area-under-curve analysis of human and murine treg suppressive function," in *Methods in Molecular Biology*, pp. 43–78, Springer, Berlin, Germany, 2016.

[34] S. Rendle, "Factorization machines," in *Proceedings of International Conference on Data Mining*, pp. 995–1000, Vancouver, Canada, December 2011.

[35] H. Guo, R. Tang, Y. Ye, Z. Li, and X. He, "DeepFM: a factorization-machine based neural network for CTR prediction," in *Proceedings of Twenty-Sixth International Joint Conference on Artificial Intelligence*, pp. 1725–1731, Melbourne, Australia, August 2017.

[36] N. Srivastava, G. Hinton, A. Krizhevsky, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.