J Inf Process Syst, Vol.13, No.5, pp.1052~1070, October 2017
https://doi.org/10.3745/JIPS.02.0069

ISSN 1976-913X (Print)
ISSN 2092-805X (Electronic)

JOURNAL OF INFORMATION PROCESSING SYSTEMS JIPS

A CTR Prediction Approach for Text Advertising Based on the SAE-LR Deep Neural Network

Zilong Jiang*, Shu Gao*, and Wei Dai**

Abstract

For the autoencoder (AE) implemented as a construction component, this paper uses the method of greedy layer-by-layer pre-training without supervision to construct the stacked autoencoder (SAE) to extract the abstract features of the original input data, which is regarded as the input of the logistic regression (LR) model, after which the click-through rate (CTR) of the user to the advertisement under the contextual environment can be obtained. These experiments show that, compared with the usual logistic regression model and support vector regression model used in the field of predicting the advertising CTR in the industry, the SAE-LR model has a relatively large promotion in the AUC value. Based on the improvement of accuracy of advertising CTR prediction, the enterprises can accurately understand and have cognition for the needs of their customers, which promotes the multi-path development with high efficiency and low cost under the condition of internet finance.

Keywords

Deep Neural Network, Machine Learning, Text Advertising, SAE-LR

1. Introduction

Internet text advertising is regarded as a type of more effective advertising communication method due to its strong targeted communication and the ease of user clicking, and it has become an important income resource for many internet companies. Several electronic commerce companies and search engine companies are seeking to explore targeted advertisements to increase their income.

Internet text advertising is usually in the form of text, and the payment method is the cost per click (CPC) [1]. In the CPC model, the click-through rate (CTR) is an important index to measure the effect of advertising placement. In this paper, the CTR prediction for internet text advertising indicates the probability to predict one user clicking a certain text advertisement under the current context environment. Due to the tripartite information of advertisement properties, user properties and context environment, the CTR prediction is very complex.

Currently, CTR prediction for internet advertising has attracted the widespread attention of researchers from industry and academia for its importance in advertisement selection. Cheng and other researchers have adopted the maximum entropy to predict the advertising CTR and match the words

Manuscript received December 15, 2016; first revision February 9, 2017; second revision March 17, 2017; accepted June 4, 2017.
Corresponding Author: Wei Dai (dweisky@163.com)
* School of Computer Science and Technology, Wuhan University of Technology, Wuhan, China (wuhanjzl@163.com, 3297947@qq.com)
** School of Economics and Management, Hubei Polytechnic University, Huangshi, China (dweisky@163.com)

searched by users with advertisement content [2]. Dave and other researchers extracted the similarity features from the advertisement data using the gradient boosting decision tree (GBDT) to predict the advertising CTR [3]. McMahan et al. [4] adopted the logistic regression model to solve the advertising CTR problems for Google. Using multiple aspects of characteristics including user information, search keywords, advertising data, and relative metadata with advertisement as the input of the model, they proposed the online sparse learning algorithm to train the model. Tagami et al. [5] from Yahoo in Japan proposed a kind of algorithm for the advertising CTR prediction based on the learning-to-rank approach in which the regression model is used in the real click logs. Chapelle [6] put forth the CTR prediction framework based on logistic regression in which the features of the advertisers, web publishers, users and time are used as the model input to solve the advertising CTR prediction for Yahoo.

Research on CTR prediction is mainly concentrated in the industry according to the above literature, where traditional models are widely adopted due to their convenience of realization and utility. Classical statistical models for CTR prediction, such as logistic regression [7], regression trees, probit regression [8], etc., are widely used with their precision relying significantly on the design of the features. However, the complex mapping relation, especially for the data with abundant meaning, cannot be efficiently expressed by these surface models and traditional artificial neural networks [9]. Therefore, surface models for CTR prediction are deficient in the feature extraction and prediction precision.

Deep learning, proposed by Geoffrey Hinton et al. [10], is used for the fields of speech recognition and image data processing, and has obtained very good effects. However, there is little relevant literature on deep learning applied to advertising. This paper puts forth a deep neural network model SAE-LR to address the problem of CTR prediction for internet text advertising.

The rest of the paper is organized as follows. In Section 2, the relative theories about stacked autoencoder and its training algorithm are presented. The proposed SAE-LR and its learning algorithm are introduced in Section 3. Then, the outstanding performance of the SAE-LR model is verified by conducting experiments on real datasets in Section 4. Finally, the conclusion and remarks are drawn in Section 5.

2. Relative Theories

2.1 Stacked Autoencoder

The autoencoder (AE) [11] is a kind of neural network model that automatically learns features from data without supervision. It contains three layers, the input layer, hidden layer, and output layer (reconstruction layer). In this neural network, the numbers of nodes in the output layer (reconstruction layer) and the input layer are the same. Its structure is described in Fig. 1.

In Fig. 1, w represents the connection weight of the nodes in the two layers and b represents the bias. In the input layer and hidden layer (encoder), the AE model will translate the input data (**x**) into every node of the hidden layer (**h**). In the hidden layer and reconstruction layer (decoder), the value of the nodes in the hidden layer (**h**) is reconstructed and the output data (**r**) are obtained.

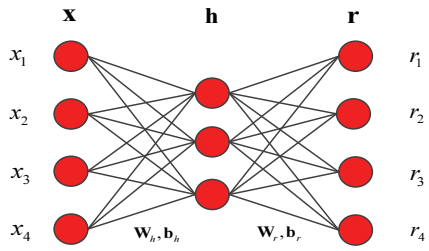

Fig. 1. The structure of autoencoder.

In the process of reconstructing the output data (**r**), only the features of the hidden layer (**h**), as opposed to the input data (**x**), can be used. That is, the information on the features of the reconstruction layer (**r**) is obtained from the features of the hidden layer (**h**). Whether the **h** has sufficient input information can be judged from the reconstruction quality.

Based on this, the process for the autoencoder to extract features can be described by the following steps. A neural network that attempts to learn an identical equation is trained, which forces the output data obtained from the process of reconstruction and input data to remain consistent. This kind of model allows the data to automatically train the network without supervision.

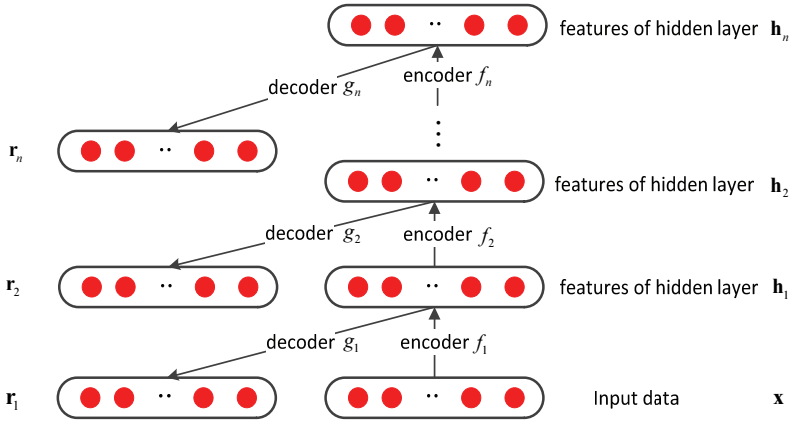

Fig. 2. The structure of the stacked autoencoder.

The stacked autoencoder (SAE) [11,12] is a kind of network that consists of n AE stacks from the bottom to the top, as shown in Fig. 2. The input data of the bottom AE is **x**. When the training of the bottom AE is finished, the feature of the hidden layer is obtained, which can be represented by $\mathbf{h}_1$. Then, $\mathbf{h}_1$ is regarded as the input data of the second AE layer, which is trained and provides the feature of the hidden layer and is represented by $\mathbf{h}_2$. This process is repeated so that when the nth AE is trained, $\mathbf{h}_n$ is obtained, which is also the abstract representation of the highest level of input data.

2.2 The Training of Stacked Autoencoder

2.2.1 Training of the autoencoder

This paper adopts the back-propagation (BP) [13] and gradient descent to train the AE without

supervision [12,14]. The related definition of the jth node in the hidden layer (**h**) 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 *j*th node of hidden layer (**h**) and the *i*th node of input layer (**x**).

b_j^h is the bias of the *j*th 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 *j*th node in the hidden layer (**h**).

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

1) Forward propagation of the input signal

First, an input vector $\mathbf{x} = (x_1, x_2, ..., x_{s_x})$ in the forward propagation calculation is given. When the input vector is encoded, the feature of the hidden layer is obtained and is represented by **h**. The output value of the *j*th node in the hidden layer (**h**) (output value of the activation function) can be represented by formula (1)

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

Therefore, the feature of hidden layer **h** can be obtained from $\mathbf{h} = (o_1^h, o_2^h, ..., o_{s_h}^h)$.

When the feature of the hidden layer **h** is decoded, the feature of the reconstruction layer **r** is obtained. The output value of the jth node in the reconstruction layer **r** (output value of the activation function) can be represented by formula (2)

$$\begin{aligned} o_j^r &= g(net_j^r) = g(b_j^r + \sum_{i=1}^{s_h} w_{ji}^r o_i^h) \\ &= \sigma(b_j^r + \sum_{i=1}^{s_h} w_{j,i}^r (\sigma(b_i^h + \sum_{k=1}^{s_x} w_{ik}^h o_k^x))) \end{aligned} \tag{2}$$

There is an error between the *j*th node in the reconstruction layer **r** and the corresponding *j*th node in the input layer that can be represented by formula (3)

$$e_j = r_j - x_j = o_j^r - o_j^x \tag{3}$$

2) Back propagation of the error signal

The optimized purpose of the AE training is to minimize the error between the reconstruction feature **r** and the input feature **x**. The reconstruction error function $L(\mathbf{x}, \mathbf{r})$ is used to measure the difference between the input feature **x** and the reconstruction feature **r** implemented by the AE. This paper selects the square error as the objective function and adopts the gradient descent to train the parameters, and the objective function can be described by formula (4).

$$J(\mathbf{x},\mathbf{r}) = \frac{1}{2} \| \mathbf{x} - \mathbf{r} \|^2 = \frac{1}{2} \sum_{j=1}^{s_x} (o_j^x - o_j^r)^2 \tag{4}$$

To easily calculate and deduce the formulae, this paper will first define the residual error δ_j^l of the jth node in the lth layer ($l \in \{h,r\}$). Thus, the error is the partial of the objective function $J(W,b)$ with respect to net_j^l .

The residual error δ_j^l of the neuron node of the reconstruction layer can be calculated using formula (5).

$$\begin{aligned} \delta_j^r &= \frac{\partial J}{\partial net_j^r} = e_j \frac{\partial o_j^r}{\partial net_j^r} = e_j \frac{\partial \sigma(net_j^r)}{\partial net_j^r} \\ &= e_j \sigma(net_j^r)(1 - \sigma(net_j^r)) \\ &= e_j o_j^r (1 - o_j^r) \end{aligned} \tag{5}$$

The residual error δ_j^h of the neuron node of the reconstruction layer can be calculated with formula (6) according to chain rule.

$$\begin{aligned} \delta_j^h &= \frac{\partial J}{\partial net_j^h} = \frac{\partial J}{\partial o_j^h} \frac{\partial o_j^h}{\partial net_j^h} \\ &= \sum_{i=1}^{s_r} \left(\frac{\partial J}{\partial net_i^r} \frac{\partial net_i^r}{\partial o_j^h} \right) \cdot \frac{\partial o_j^h}{\partial net_j^h} \\ &= \left(\sum_{i=1}^{s_r} \delta_i^r w_{ji}^r \right) \frac{\partial \sigma(net_j^h)}{\partial net_j^h} \\ &= \left(\sum_{i=1}^{s_r} \delta_i^r w_{ji}^r \right) \sigma(net_j^h)(1 - \sigma(net_j^h)) \\ &= \left(\sum_{i=1}^{s_r} \delta_i^r w_{ji}^r \right) o_j^h (1 - o_j^h) \end{aligned} \tag{6}$$

The gradient of the cost function $J(W,b)$ with respect to the parameters w_{ji}^l and b_j^l can be calculated by formulae (7) and (8).

$$\frac{\partial J}{\partial w_{ji}^l} = \frac{\partial J}{\partial net_j^l} \frac{\partial net_j^l}{\partial w_{ji}^l} = \delta_j^l \cdot o_i^{l-1} \tag{7}$$

$$\frac{\partial J}{\partial b_j^l} = \delta_j^l \tag{8}$$

The updated formula to calculate parameters w_{ji}^l and b_j^l can be described as the following formulae, in which β represents the learning rate and is a hyper-parameter in prior.

$$w_{ji}^l = w_{ji}^l - \beta \frac{\partial J}{\partial w_{ji}^l} = w_{ji}^l - \beta \cdot \delta_j^l \cdot o_i^{l-1} \tag{9}$$

$$b_j^l = b_j^l - \beta \frac{\partial J}{\partial b_j^l} = b_j^l - \beta \cdot \delta_j^l \tag{10}$$

In the above formula (9) and formula (10), the two conditions that need to be satisfied are $l \in \{h, r\}$ and $l = r$, $l-1 = h$; $l = h$, $l-1 = x$.

According to the above analysis, the training algorithm of the AE can be described as Algorithm 1.

Algorithm 1. The training algorithm of the AE

Input: Original input data $\mathbf{x}_d$

Output: Every parameter value of the AE $\theta = \{W, b\}$

1. Use the certain small value to randomly initialize the parameters of the AE $\theta = \{W, b\}$;
2. In the forward propagation of the input signal, use formula (3) to calculate the error between the reconstruction layer and the input layer in terms of the input data $\mathbf{x}_d$;
3. In the back propagation of the error signal, use formula (7) and formula (8) to calculate the gradient of each parameter $\theta = \{W, b\}$;
4. Use formula (9) and formula (10) to update each parameter value of the AE;

2.2.2 Training algorithm of the stacked autoencoder

Based on the above algorithm, the training algorithm of the stacked autoencoder (SAE) can be described as Algorithm 2.

Algorithm 2. The training algorithm of the stacked autoencoder (SAE)

Input: Original input data $\mathbf{x}_d$

Output: Every parameter value of the stacked autoencoder $\theta = \{W, b\}$

1. Use the original input data $\mathbf{x}_d$ as the input and adopt Algorithm 1 to train the bottom AE to obtain the parameter values of the AE network and the feature of the hidden layer $\mathbf{h}_1$;
2. Ensure the parameter and structure of the bottom AE and use the feature of the hidden layer $\mathbf{h}_1$ as the input of the second AE in terms of Algorithm 1 to obtain the parameter value of the second AE and feature of the hidden layer $\mathbf{h}_2$;
3. Repeat this process until the parameter value of the *r*th AE and feature of the hidden layer $\mathbf{h}_r$ are obtained.

3. Study on CTR Prediction for Text Advertising Based on the SAE-LR Deep Neural Network

3.1 CTR Prediction Model for Text Advertising Based on the SAE-LR

3.1.1 The construction of SAE-LR deep neural network

For the autoencoder implemented as a construction component, this paper uses the greedy layer-by-layer pre-training without supervision method to construct the stacked autoencoder [14,15] to extract

the abstract features from the original input data. Then, this abstract features are regarded as the input of the logistic regression model, and the click-through rate of the user to the advertisement under the context environment can be obtained.

The structure of SAE-LR deep neural network is described in Fig. 3, in which the SAE is the generative model section, and LR is the discrimination model section. To construct the SAE-LR deep neural network, the following steps must be completed.

Step 1: Construct the first AE model used to handle the input data of the click blog (real vector);

Step 2: Use the output value of the activation probability of the first AE model as the input and construct the second AE model;

Step 3: Repeat this process until all AE models set in prior are constructed;

Step 4: Stack a generative model using the method for stacking all AE models to obtain a stacked autoencoder (SAE);

Step 5: Add an LR layer to the SAE, and construct a new blend model SAE-LR deep neural network.

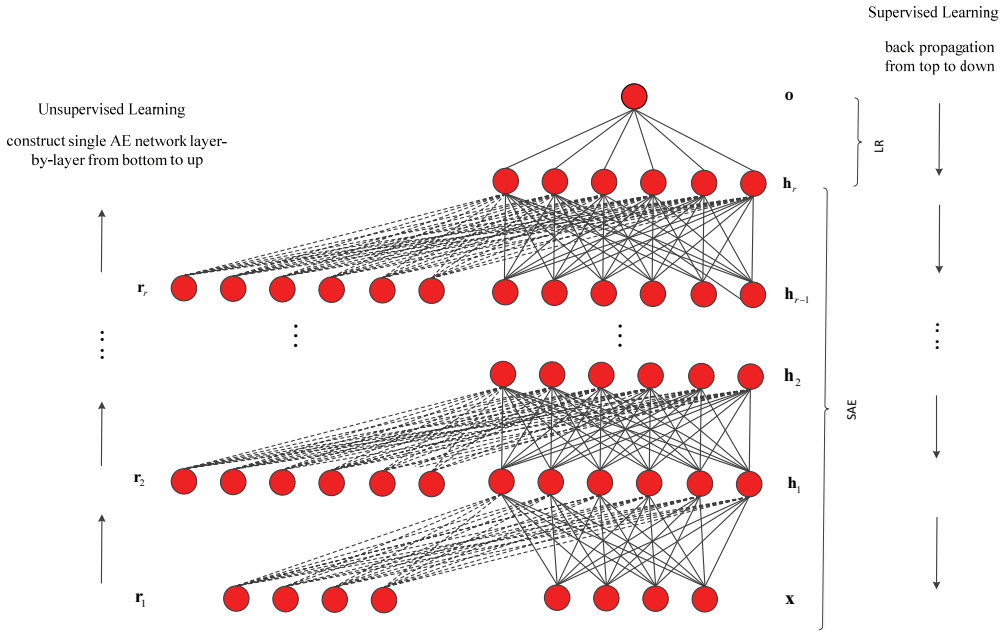

Fig. 3. The structure of the SAE-LR deep neural network.

3.1.2 Extracting abstract features with the SAE

The SAE was adopted to extract the non-linear abstract features of users' click logs and relative fields. The SAE [14,15] is a generative model that is composed of a stack of autoencoders and is trained in a greedy layer-wise unsupervised manner. This method relies on the training algorithm of the autoencoder to initialize the parameters of a stacked autoencoder. The input values of the visible layer of the first autoencoder are directly derived from the original data (real vector). After training the parameters of the first autoencoder with Algorithm 1, the obtained values of the hidden layer nodes can be considered as another compact expression of the input vector. Then, the parameters (weight, bias) of the first autoencoder are fixed, and the values of the hidden nodes ($\mathbf{h}_1$) of the second layer are regarded

as the input vector to obtain the values of the hidden nodes ($\mathbf{h}_2$) of the output layer of the next autoencoder, and so on. Each new layer is stacked on top of the current autoencoder. The lower layer is used to detect simple features and feed them into the upper layer. This process gradually refines the previous learned information and further finds more complex features, thereby generating more abstract information representing the underlying regularities of the input data.

3.1.3 Modeling click-through rate by means of LR

The logistic regression model (LR) [16] is a type of classical model in the field of advertising click rate prediction and is also a kind of probability discrimination model belonging to the set of classification models. The LR model is shown at the top of Fig. 3 with only one node in the output layer that connects all the nodes of the input layer. The activation function of the node in the output layer can be represented as $g(x) = 1/(1+e^{-x})$. The output value of the node (value of the activation function) $h_\theta(\mathbf{x}')$ is the probability value of users' clicking advertisements, and it can also be described as $p(y=1 \mid \mathbf{x}';\boldsymbol{\theta})$. When $h_\theta(\mathbf{x}')$ is larger than a certain threshold value, it can be judged that the user will click the advertisement (click label $y_d = 1$). When $h_\theta(\mathbf{x}')$ is smaller than the threshold value, it can be judged that the user will not click the advertisement (click label $y_d = 0$).

When an arbitrary sample $(\mathbf{x}_d', y_d)$ in the training set $D = \{(\mathbf{x}_1', y_1), (\mathbf{x}_2', y_2), ..., (\mathbf{x}_N', y_N)\}$ is given, the probability value of the advertising click rate prediction can be obtained in terms of the logistic regression, and it can be described as formula (11).

$$p(click \mid a,u,c) = p(y_d = 1 \mid \mathbf{x}_d';\boldsymbol{\theta}) = \frac{1}{1+e^{-\boldsymbol{\theta}^T \mathbf{x}_d'}} \tag{11}$$

Correspondingly, $p(y_d = 0 \mid \mathbf{x}_d';\boldsymbol{\theta}) = 1 - \frac{1}{1+e^{-\boldsymbol{\theta}^T \mathbf{x}_d'}}$.

In the formula, $\boldsymbol{\theta}^T$ is the parameter of the standardized LR model. In addition, a refers to the feature of the advertisement, u is the features of the users, and c represents the features of the context environment. All three features combine into a vector that can be described as $\mathbf{x}_d'$, which is the input feature of the logistic regression model.

Therefore, the advertising click rate prediction is a binary classification. This paper selects the likelihood function as the objective function. For a single sample $(\mathbf{x}_d', y_d)$, the probability of correctly predicting the advertising click rate can be described as $p(y_d \mid \mathbf{x}_d';\theta) = h_\theta(\mathbf{x}_d')^{y_d}(1-h_\theta(\mathbf{x}_d'))^{1-y_d}$. The probability distribution of the entire sample space can be described as $p(Y_D \mid \mathbf{X}_D';\theta) = \prod_{d=1}^{N}(h_\theta(\mathbf{x}_d')^{y_d}(1-h_\theta(\mathbf{x}_d'))^{1-y_d})$.

For easy calculations, the formula can be transferred into formula (12)

$$J(\theta) = -\frac{1}{N}\sum_{d=1}^{N}(y_d \log h_\theta(\mathbf{x}_d') + (1-y_d)\log(1-h_\theta(\mathbf{x}_d'))) \tag{12}$$

In formula (12), the click label y_d is the expected output, and $h_\theta(\mathbf{x}_d')$ is the real output value of the

neuron node in the output layer (output of the activation function), and $h_\theta(\mathbf{x}_d')$ can be described as $h_\theta(\mathbf{x}_d') = g(\mathbf{z}), \mathbf{z} = \sum_i w_i x_i + b = \boldsymbol{\theta}^T \mathbf{x}_d'$.

3.2 Training of CTR Prediction Model for Text Advertising Based on the SAE-LR

Step 1: Use numerous unlabeled data to construct a single AE network layer-by-layer from bottom to top using the greedy training algorithm without supervision to obtain the initial weight and bias parameters of the SAE;

Step 2: Use a few labeled data to obtain the initial weight and bias parameters of the LR model with supervision using the gradient descent;

Step 3: Use a few labeled data to fine-tune the weight and bias parameters of the SAE-LR deep neural network using back-propagation (BP) and gradient descent;

3.2.1 Training of the stacked autoencoder

The training of the SAE is described in Algorithm 2.

3.2.2 Obtaining the parameters of the LR model with supervised learning

A few labeled sample data $(\mathbf{x}_d, y_d)$ are given. In these samples, the input data $\mathbf{x}_d$ is put into the SAE, and a real vector $\mathbf{x}_d'$ constituted by the output value in the final hidden layer of the SAE (output value of the activation function) is obtained. Then, the new samples $(\mathbf{x}_d', y_d)$ are regarded as the data with labels to train the LR model in the top layer.

For formula (12), this paper adopts the gradient descent to train the parameters of the LR model $\theta = \{W, b\}$.

$$\begin{aligned}
\frac{\partial J}{\partial \theta} &= -\frac{1}{N}\sum_{d=1}^{N}\left(y_d \frac{1}{h_\theta(\mathbf{x}_d')}\frac{\partial h_\theta(\mathbf{x}_d')}{\partial \theta} - (1-y_d)\frac{1}{1-h_\theta(\mathbf{x}_d')}\frac{\partial h_\theta(\mathbf{x}_d')}{\partial \theta}\right) \\
&= -\frac{1}{N}\sum_{d=1}^{N}\left(y_d \frac{1}{g(\theta^T\mathbf{x}_d')} - (1-y_d)\frac{1}{1-g(\theta^T\mathbf{x}_d')}\right)\frac{\partial g(\theta^T\mathbf{x}_d')}{\partial \theta} \\
&= -\frac{1}{N}\sum_{d=1}^{N}\left(y_d \frac{1}{g(\theta^T\mathbf{x}_d')} - (1-y_d)\frac{1}{1-g(\theta^T\mathbf{x}_d')}\right) g(\theta^T\mathbf{x}_d')(1-g(\theta^T\mathbf{x}_d'))\frac{\partial \theta^T\mathbf{x}_d'}{\partial \theta} \\
&= -\frac{1}{N}\sum_{d=1}^{N}\left(y_d(1-g(\theta^T\mathbf{x}_d')) - (1-y_d)g(\theta^T\mathbf{x}_d')\right)\mathbf{x}_d' \\
&= -\frac{1}{N}\sum_{d=1}^{N}\left(y_d - g(\theta^T\mathbf{x}_d')\right)\mathbf{x}_d' \\
&= \frac{1}{N}\sum_{d=1}^{N}\left(h_\theta(\mathbf{x}_d') - y_d\right)\mathbf{x}_d' \\
&= C\sum_{d=1}^{N}\left(h_\theta(\mathbf{x}_d') - y_d\right)\mathbf{x}_d'
\end{aligned} \tag{13}$$

As the training progresses, the parameter θ is updated by:

$$\theta_j := \theta_j - \alpha C\sum_{d=1}^{N}\left(h_\theta(\mathbf{x}_d') - y_d\right)\mathbf{x}_d' \tag{14}$$

The above process can be described as Algorithm 3.

Algorithm 3. Training algorithm for the LR
Input: Original labeled sample data $(\mathbf{x}_d, y_d)$
Output: Parameter values of the LR $\theta = \{W, b\}$

1. Put the input data $\mathbf{x}_d$ of the original labeled sample data $(\mathbf{x}_d, y_d)$ into the SAE to obtain a real vector $\mathbf{x}_d'$ constituted by the output value in the final hidden layer of the SAE (output value of the activation function);
2. Set the new samples $(\mathbf{x}_d', y_d)$ as the labeled data to train the LR model in the top layer and adopt formula (13) to calculate the gradient of the parameters θ;
3. Use formula (14) to update the parameters θ;

3.2.3 Fine-tune the parameters of the SAE-LR deep neural network with supervision

This paper uses BP and gradient descent to fine-tune the weight and bias parameters of the SAE-LR deep neural network with supervision.

1) Forward propagation

When the sample data of the click logs are given, and the combined feature vector $\mathbf{x}_d = (x_1, x_2, ..., x_n)$ is given, the *j*th node in the lth layer of the SAE-LR deep neural network can be defined as follows:

L_n is the number of total layers of the entire SAE-LR network (from the input layer to the output layer);

s_l is the number of nodes contained in the lth layer of the SAE-LR network;

$w_{j,i}^l$ is the connection weight between the *j*th node in the lth layer and the *i*th node in the l-1th layer;

b_j^l is the bias of the *j*th node in the lth layer;

$net_j^l = b_j^l + \sum_{i=1}^{s_{n-1}} w_{j,i}^l o_i^{l-1}$ is the weight sum input of the *j*th node in the lth layer;

o_j^l is the output value of the activation function of the *j*th node in the lth layer;

The selected activation function is $g(x) = 1/(1+e^{-x})$.

In the forward propagation, the input signal is transmitted from the input layer to the output layer. For the *j*th node in the second layer of the SAE-LR network (l = 2), the input value can be described as $net_j^2 = b_j^2 + \sum_{i=1}^{s_1} w_{j,i}^2 x_i$. When the input of node *j* is transferred by the activation function, its output value can be described as $o_j^2 = g(net_j^2)$. Therefore, the input value of the *j*th node in the lth layer of SAE-LR network ($l = 3, ..., L_n$) can be described as $net_j^l = \sum_{i=1}^{s_{l-1}} w_{ji}^l o_i^{l-1} + b_j^l$, and its corresponding output value can be described as $o_j^l = g(net_j^l)$.

In particular, the output value for the *j*th node in the L_nth layer of the SAE-LR network can be described as:

$$h_\theta(\mathbf{x}_d) = O_j = O_1 = o_1^{L_n} \tag{15}$$

2) Back propagation

This paper regards the cross entropy as the object function. The local gradient will be refined using the back propagation. The error signal between the output value and the labeled signal (y_d) will be transmitted from the output layer to input layer in the process.

$$\begin{aligned} J(\theta) &= y_d \log h_\theta(\mathbf{x}_d) + (1-y_d)\log(1-h_\theta(\mathbf{x}_d)) \\ &= y_d \log g(net_j^{L_n}) + (1-y_d)\log(1-g(net_j^{L_n})) \end{aligned} \tag{16}$$

In this formula, y_d is the label value of the sample and can be represented as $y_d = 1 \ or \ 0$.

This paper uses the gradient descent to train the model. To simplify the calculation, this paper will first define the residual error δ_j^l of the *j*th node in the lth layer. That is, the residual error is the partial and of the objective function $J(W,b)$ with respect to the weight sum net_j^l of node j. The residual error $\delta_j^{L_n}$ of the jth neuron node of the L_n th layer can be calculated by formula (17).

$$\begin{aligned} \delta_j^{L_n} &= \frac{\partial J}{\partial net_j^{L_n}} = y_d \frac{1}{g(net_j^{L_n})} g'(net_j^{L_n}) + (1-y_d)\frac{-1}{1-g(net_j^{L_n})} g'(net_j^{L_n}) \\ &= y_d \frac{1}{g(net_j^{L_n})} g(net_j^{L_n})(1-g(net_j^{L_n})) + (1-y_d)\frac{-1}{1-g(net_j^{L_n})} g(net_j^{L_n})(1-g(net_j^{L_n})) \\ &= y_d - g(net_j^{L_n}) \\ &= y_d - o_1^{L_n} \end{aligned} \tag{17}$$

The residual error δ_j^l of the *j*th node of lth layer ($l = 2,...L_n - 1$) can be calculated by formula (18) according to chain rule.

$$\begin{aligned} \delta_j^l &= \frac{\partial J}{\partial net_j^l} = \frac{\partial J}{\partial o_j^l}\frac{\partial o_j^l}{\partial net_j^l} \\ &= \sum_{i=1}^{s_{l+1}} \left(\frac{\partial J}{\partial net_i^{l+1}} \frac{\partial net_i^{l+1}}{\partial o_j^l} \right) \cdot \frac{\partial o_j^l}{\partial net_j^l} \\ &= \left(\sum_{i=1}^{s_{l+1}} \delta_i^{l+1} w_{ji}^{l+1} \right) \frac{\partial g(net_j^l)}{\partial net_j^l} \\ &= \left(\sum_{i=1}^{s_{l+1}} \delta_i^{l+1} w_{ji}^{l+1} \right) g(net_j^l)(1-g(net_j^l)) \\ &= \left(\sum_{i=1}^{s_{l+1}} \delta_i^{l+1} w_{ji}^{l+1} \right) o_j^l (1-o_j^l) \end{aligned} \tag{18}$$

In this formula, w_{ji}^{l+1} is the connection weight between the *j*th node of lth layer and the *i*th node of the l+1th layer.

The gradient of the cost function $J(W,b)$ with respect to the parameters w_{ji}^l and b_j^l can be calculated by formulae (19) and (20).

$$\frac{\partial J}{\partial w_{ji}^l} = \frac{\partial J}{\partial net_j^l}\frac{\partial net_j^l}{\partial w_{ji}^l} = \delta_j^l \cdot o_i^{l-1} \tag{19}$$

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

The updated formulae to calculate the parameters w_{ji}^l and b_j^l can be described as the following formulae in which β is the learning rate and is a hyper-parameter in prior.

$$w_{ji}^l = w_{ji}^l - \beta \frac{\partial J}{\partial w_{ji}^l} = w_{ji}^l - \beta \cdot \delta_j^l \cdot o_i^{l-1} \tag{21}$$

$$b_j^l = b_j^l - \beta \frac{\partial J}{\partial b_j^l} = b_j^l - \beta \cdot \delta_j^l \tag{22}$$

The above process can be described as Algorithm 4.

Algorithm 4. The algorithm to fine-tune the parameters of the SAE-LR deep neural network

Input: Original labeled input data

Output: The parameter values of the SAE-LR $\theta = \{W, b\}$

1. In the forward propagation of input signal, when the input data $\mathbf{x}_d$ is put into the input layer, the output value of the output layer (the output of the activation function) can be obtained using formula (15);
2. In the back propagation of the error signal, formula (19) and formula (20) are used to calculate the gradient for every parameter $\theta = \{W, b\}$;
3. Formula (21) and formula (22) update every parameter value of the SAE-LR $\theta = \{W, b\}$;

4. Design of Experiments and Analysis of Results

4.1 Dataset of Training and Test

This paper adopts the KDD Cup 2012 Track2 dataset [17] to implement the experiment. The entire dataset includes the training set and test set in combination with another five additional documents: queryid_tokensid.txt, purchasedkeywordid_tokensid.txt, titleid_tokensid.txt, descriptionid_tokensid.txt, and userid_profile.txt, which are used to expand the information beyond the training set and test set. This paper will randomly extract a half of the training set as the training set of this paper, and a half of the test is set as test set. The 5-fold cross validation method [18] is adopted in the training process. Word embedding is obtained from the words of the title and the description of the advertising property using word2vec [19]. Then, the weighted average pooling method [20] based on the word embedding is adopted to obtain the vector representation of the title and description of the advertising property. And a large number of multi-field categorical features of the data set are used as main features for CTR prediction. These multi-field categorical features, including AdID, AdvertiserID, QueryID, KeywordID, TitleID, DescriptionID, UserID, DisplayURL, are represented as binary features with the one-hot encoding method. Then, the hash strategy (signed hash trick) [21] is adopted to reduce the dimensionality of these sparse binary feature vectors. Finally, the joint vector comprised of the word embedding and one-hot representation is regarded as the input of the machine learning models.

4.2 Hardware Equipment

High performance workstations are used in the experiments. The hardware configuration of the workstations is shown in Table 1.

4.3 Software

Python 2.7 is used to develop the programs, and the Theano Python library [22] is used to build and train the deep architecture.

Table 1. Hardware equipment

Name	Type and parameters
CPU	Intel XEON E5-2687W*2
Memory	32 GB DDR3 ECC REGS 1600 MHz*32
Hard disk	1T SATA

4.4 Baseline Models

This paper selects the usual LR model and support vector regression (SVR) model [23,24] used in industry as baseline models to compare their performance.

4.5 Performance Evaluation Metric of Models

This paper adopts the area under the curve (AUC) [25,26] to evaluate the CTR prediction model. The advertising CTR prediction is a type of classical binary classification by means of the criterion of whether the advertisement is clicked. In the process of predicting the advertising CTR, the rows of the confusion matrix (Table 2) [27] (positive and negative) are used to represent the sample types of the advertising CTR prediction, and the columns (true and false) are used to judge the correct or incorrect advertising CTR prediction. According to the sample numbers clicked by users in the testing data and the result set of the model prediction, four kinds of numbers need to be calculated. These calculations are the number of samples that are clicked by users in the real log and are predicted by the model to be clicked (TP), the number of samples that are clicked by users in the real log and are predicted by the model to be unclicked (TN), the number of samples that are unclicked by users in the real log but predicted by the model to be clicked (FP) and the number of samples that are unclicked by users in the real log and are predicted by the model to be unclicked (FN). Then, formulae (23) and (24) are used to calculate the corresponding TPR value and FPR value, and a dot pair can be obtained. Different dot pairs have their own corresponding coordinates in the ROC [28]. When these values are connected, a saw-toothed curve is obtained, and the AUC is the total of the areas under the saw-toothed curve. The bigger the AUC value becomes, the more accurate the advertising CTR prediction is. Therefore, the advertising placement will be more effective. This paper adopts the interface function provided by the open-source tool sklearn [29] to calculate the AUC value.

$$TPR = TP / (TP + FN) \tag{23}$$

$$FPR = FP / (FP + TN) \tag{24}$$

Table 2. Confusion matrix

	Positive	Negative
True	True positive	True negative
False	False positive	False negative

4.6 Experiment Design and Results Analysis

4.6.1 The training and optimizing of SAE-LR Model

Eventually, the dimensionality of the input vector becomes 4726048. Therefore, the number of nodes of the visible layer (input layer) is also 4726048. Based on past experience, when the number of nodes of the visible layer is large, the nodes of the hidden layer require the process of "dimensionality reduction-dimensionality addition - dimensionality reduction". However, when the number of nodes of the visible layer is small, the nodes of the hidden layer only requires the process "dimensionality addition – dimensionality reduction". In general, the number of hidden nodes will be increased or decreased in multiples. Currently, there is no method to quickly set the number of hidden layers and the number of hidden nodes for every layer. Therefore, many experiments and experiences are evaluated to search for the relative optimal structure, which also provides future research directions. In the process of training the SAE-LR model, the initial control parameters of precision is presented in Table 3.

Table 3 Initial parameters of precision of SAE-LR

Name of parameter	Value
Unsupervised learning rate	0.01
Supervised learning rate	0.1
Number of unsupervised epochs	10
Fine-tuning rate	0.1
Number of fine-tuning	1000
Minibatch	10

To avoid under-fitting and over-fitting of the model, a 5-fold cross-validation [17] is used in the training dataset. However, the computation of the generalization errors in the 5-fold cross-validation is replaced by the computation of the AUC value. After 20 experiments with different configurations, the highest AUC value is obtained when the layers and nodes are shown as follows: there are 3 hidden layers; 50 nodes of the first hidden layer; 500 nodes of the second hidden layer; and 100 nodes of the third hidden layer. This paper chooses this configuration as the final structure. The data from the partial experimental results are shown in Table 4. The AUC value in Table 4 is the average of the 5-fold cross -validation.

4.6.2 Using baseline models to predict CTR

In Section 3, the LR model and its training algorithm were described. Based on this, this paper adopts the method in the literature [30] to implement the logistic regression model with the Python language

in which the Scientific Python libraries include pandas, scipy, numPy, and scikit-learn in the operating system centos 6.5 from the literature [31].

The joint vector of every click log record is regarded as the input data of the LR model. When the input vector is calculated by the logistic regression program, a prediction probability value is obtained. Then, a threshold value is set. If the threshold value is smaller than the probability value, we assume that the advertisement will be clicked, that is, Y is assigned to 1. Otherwise, the advertisement will not be clicked and Y is assigned to 0. Finally, the corresponding interface function of the sklearn library will be used to calculate the AUC value.

Table 4. AUC values of SAE-LR with different layers and nodes

AUC	Layers number	Nodes of h_1	Nodes of h_2	Nodes of h_3	Nodes of h_4	Nodes of h_5
0.7437	3	30	300			
0.7727	4	30	300	100		
0.7823	5	40	400	200	100	
0.7892	4	50	500	100		
0.7874	5	50	500	200	100	
0.7842	6	60	600	400	200	100
0.7865	6	60	600	300	200	100

The SVR [23,24] is an important branch of support vector machines [32]. The SVR algorithm means that when the dimension of the vector increases, the linear decision function is constructed to realize the linear regression in the high-dimensional space. The linear decision function includes the e-insensitivity function and kernel function algorithm. In this paper, the SVR algorithm adopts the kernel function to replace the linear term of the linear equation. Therefore, the original linear algorithm becomes non-linearized and can be used as a non-linearization regression. The detailed theories of the SVR can be found in the literature [23,24]. Based on the above Python libraries [31], this paper adopts the function sklearn.svm.SVR to process the SVR, and the detailed realization process can be seen in the literature [33].

The joint vector of every click log record is regarded as the input data of the SVR. When the input vector is calculated by the SVR program, a prediction probability value is obtained. Then, a threshold value is set. If the threshold value is smaller than the probability value, we assume the advertisement will be clicked, that is, Y is assigned to 1. Otherwise, the advertisement will not be clicked and Y is assigned to 0. Finally, the corresponding interface function of the sklearn library will be used to calculate the AUC value.

4.6.3 The comparison of performance

To compare the three models, this paper distributes the test set into equal 10 portions. A testset1 contains all the data of portion 1 and testset2 contains all the data of portion1 and portion 2. By this analogy, testset10 contains all data from the 10 portions.

According to Fig. 4, when the three different models are applied into test datasets with different scales, the AUC value of the LR model is smaller than that of SVR model when the data are in the small

scale. However, when the data are in the big scale, the AUC value of the LR model is bigger than that of the SVR model. This demonstrates that the SVR model shows a good effect when the data are in the small scale because of its complexity. However, when the data are in the big scale, the effect obtained from the SVR model is worse than the effect obtained from the LR model. These results show that the generalization ability of the SVR model is worse than that of the LR model for its own complexity.

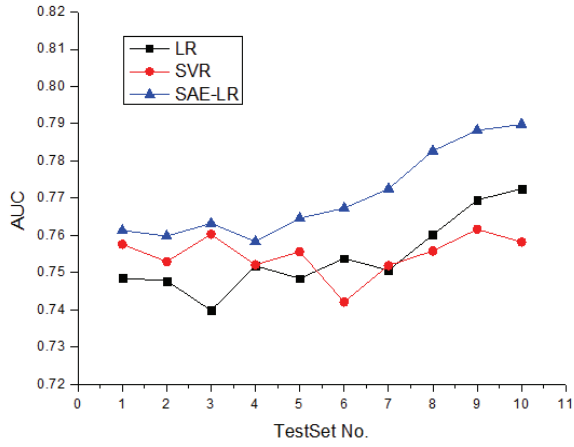

Fig. 4. The AUC values of three models in test sets with different scales.

The AUC value of the SAE-LR model is bigger than those of the LR model and SVR model when the data are in different scales. Therefore, the AUC values obtained by the SAE-LR model are always bigger than the AUC values obtained by the LR model, which shows that the abstract features extracted by the SAE network from the original input data have a better classification effect than the original input features that are only pre-processed. Moreover, the AUC values obtained by the SAE-LR model are also bigger than those obtained by the SVR model, which shows that, in the models used for the advertising CTR prediction, the SAE-LR model is better than the simple surface baseline models.

5. Conclusion and Future Work

This paper proposes a deep neural network SAE-LR to address the problem of advertising CTR prediction. In this model, the features and abstract relation of the advertisements, users' information and context information are adequately learned, which is useful to efficiently improve the precision of the advertising CTR prediction.

In addition, this paper compares the performances of the SAE-LR model with two other surface models (LR and SVR) in test sets with different scales and finds that the AUC value of the SAE-LR model is higher than that of two surface models in large scale test sets.

Meanwhile, our future work will concentrate on the following research. Since there is no method to quickly set the number of hidden layers and the number of hidden nodes of every layer, the future studies should first focus on finding an objective method for addressing this case with fewer experiments [34]. Then, future studies should focus on exploring deeper architectural models to improve the CTR prediction for internet text advertising.

Acknowledgement

This work is supported by the Humanities and Social Science Youth Foundation of the Ministry of Education of China (No. 13YJCZH028) and the National Natural Science Foundation of China (No. 71473074).

References

[1] Y. Hu, J. Shin, and Z. Tang, "Incentive problems in performance-based online advertising pricing: cost per click vs. cost per action," *Management Science*, vol. 62, no. 7, pp. 2022-2038, 2015.

[2] H. Cheng and E. Cantu-Paz, "Personalized click prediction in sponsored search," in *Proceedings of the 3rd ACM International Conference on Web Search and Data Mining*, New York, NY, 2010, pp. 351-360.

[3] K. Dave and V. Varma, "Predicting the click-through rate for rare/new Ads," Centre for Search and Information Extraction Lab International Institute of Information Technology, Hyderabad, *Technical Report IIIT/TR/2010/15*, 2010.

[4] H. B. McMahan, G. Holt, D. Sculley, M, Young, D. Ebner, J. Grady, et al., "Ad click prediction: a view from the trenches," in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Chicago, IL, 2013, pp. 1222-1230.

[5] Y. Tagami, S. Ono, K. Yamamoto, K. Tsukamoto, and A. Tajima, "CTR prediction for contextual advertising: learning-to-rank approach," in *Proceedings of the 7th International Workshop on Data Mining for Online Advertising*, Chicago, IL, 2013.

[6] O. Chapelle, "Modeling delayed feedback in display advertising," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, NY, 2014, pp. 1097-1105.

[7] H. Cheng, R. van Zwol, J. Azimi, E. Manavoglu, R. Zhang, Y. Zhou, and V. Navalpakkam, "Multimedia features for click prediction of new ads in display advertising," in *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Beijing, China, 2012, pp. 777-785.

[8] T. Graepel, J. Q. Candela, T. Borchert, and R. Herbrich, "Web-scale Bayesian click-through rate prediction for sponsored search advertising in Microsofts Bing search engine," in *Proceedings of the 27th International Conference on Machine Learning*, Haifa, Israel, 2010, pp. 13-20.

[9] Y. Bengio, "Learning deep architectures for AI," *Foundations and Trends in Machine Learning*, vol. 2, no. 1, pp. 1-127, 2009.

[10] 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.

[11] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *Advances in Neural Information Processing Systems*, vol. 19, Vancouver, British Columbia: MIT Press, 2006, pp. 153-160.

[12] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P. A. Manzagol, "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.

[13] R. Trivedi, T. N. Singh, and A. K. Raina, "Simultaneous prediction of blast-induced flyrock and fragmentation in opencast limestone mines using back propagation neural network," *International Journal of Mining and Mineral Engineering*, vol. 7, no. 3, pp. 237-252, 2016.

[14] H. I. Suk, S. W. Lee, and D. Shen, "Latent feature representation with stacked auto-encoder for AD/MCI diagnosis," *Brain Structure and Function*, vol. 220, no. 2, pp. 841-859, 2015.

[15] J. Gehring, Y. Miao, F. Metze, and A. Waibel "Extracting deep bottleneck features using stacked auto-encoders," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Vancouver, Canada, 2013, pp. 3377-3381.

[16] T. Osman, P. Divigalpitiya, and T. Arima, "Driving factors of urban sprawl in Giza Governorate of the Greater Cairo metropolitan region using a logistic regression model," *International Journal of Urban Sciences*, vol. 20, no. 2, pp. 206-225, 2016.

[17] KDD Cup 2012, Track 2 [Online]. Available: https://www.kaggle.com/c/kddcup2012-track2/.

[18] K. J. Grimm, G. L. Mazza, and P. Davoudzadeh, "Model selection in finite mixture models: a k-fold cross-validation approach," *Structural Equation Modeling: A Multidisciplinary Journal*, vol. 24, no. 2, pp. 246-252, 2017.

[19] Word2vec [Online]. Available: https://code.google.com/p/word2vec/.2017.

[20] Z. Jiang, S. Gao, and M. Li, "A feature vector representation approach for short text based on RNNLM and pooling computation," *Academic Journal of Manufacturing Engineering*, vol. 15, no. 2, pp. 6-14, 2017.

[21] K. Weinberger, A. Dasgupta, J. Langford, A. Smola, and J. Attenberg, "Feature hashing for large scale multitask learning," in *Proceedings of the 26th Annual International Conference on Machine Learning*, Montreal, Canada, 2009, pp. 1113-1120.

[22] Theano [Online]. Available: https://github.com/Theano/Theano.

[23] G. Santamaria-Bonfil, A. Reyes-Ballesteros, and C. Gershenson, "Wind speed forecasting for wind farms: a method based on support vector regression," *Renewable Energy*, vol. 85, pp. 790-809, 2016.

[24] K. P. Lin and P. F. Pai, "Solar power output forecasting using evolutionary seasonal decomposition least-square support vector regression," *Journal of Cleaner Production*, vol. 134(Part B), pp. 456-462, 2016.

[25] 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," *Methods in Molecular Biology*, vol. 1371, pp. 43-78, 2016.

[26] A. Jimenez-Valverde, "Insights into the area under the receiver operating characteristic curve (AUC) as a discrimination measure in species distribution modelling," *Global Ecology and Biogeography*, vol. 21, no. 4, pp. 498-507, 2012.

[27] X. Deng, Q. Liu, Y. Deng, and S. Mahadevan, "An improved method to construct basic probability assignment based on the confusion matrix for classification problem," *Information Sciences*, vol. 340-341, pp. 250-261, 2016.

[28] A. J. Talabani, B. H. Endreseth, S. Lydersen, and T. H. Edna, "Clinical diagnostic accuracy of acute colonic diverticulitis in patients admitted with acute abdominal pain, a receiver operating characteristic curve analysis," *International Journal of Colorectal Disease*, vol. 32, no. 1, pp. 41-47, 2017.

[29] scikit-learn 0.15.2 [Online]. Available: https://pypi.python.org/pypi/scikit-learn/0.15.2.

[30] Logistic Regression in Python [Online]. Available: http://blog.yhat.com/posts/logistic-regression-and-python.html.

[31] Setting Up Scientific Python [Online]. Available: http://blog.yhat.com/posts/setting-up-scientific-python.html.

[32] P. Rebentrost, M. Mohseni, and S. Lloyd, "Quantum support vector machine for big data classification," *Physical Review Letter*, vol. 113, article no. 130503, 2014.

[33] Python sklearn.svm.SVR Examples [Online]. Available: http://www.programcreek.com/python/example/75189/sklearn.svm.SVR.

[34] Z. Jiang, S. Gao, and W. Dai, "Research on CTR prediction for contextual advertising based on deep architecture model," *Journal of Control Engineering and Applied Informatics*, vol. 18, no. 1, pp. 11-19, 2016.

Zilong Jiang

He has been with the School of Computer Science and Technology from Wuhan University of Technology as a Ph.D. candidate Since September 2012. He received his M.S. degree in computer science from Wuhan University of Technology in 2009, and he has rich practical experience since he worked as an engineer for excellent internet companies over many years. Presently, he is also a researcher at the School of Computer and Information, Qiannan Normal University for Nationalities. His current research interests focus on deep learning, computational advertising, and HPC.

Shu Gao

She is a professor and Ph.D. Supervisor at the School of Computer Science and Technology, Wuhan University of Technology. Her current research interests focus on data visualization and intelligent computation.

Wei Dai

He received his Ph.D. degree from Wuhan University of Technology in 2012. Now he is an associate professor and full researcher at the School of Economics and Management, Hubei Polytechnic University. His current research interests include different aspects of intelligence computation and information systems.