RESEARCH ARTICLE

An improved advertising CTR prediction approach based on the fuzzy deep neural network

Zilong Jiang[1,2], **Shu Gao**[1], **Mingjiang Li**[2]*

1 School of Computer Science and Technology, Wuhan University of Technology, Wuhan, Hubei Province, China, **2** School of Computer and Information, Qiannan Normal University for Nationalities, Duyun, Guizhou Province, China

* mingjiangli1975@gmail.com

OPEN ACCESS

Citation: Jiang Z, Gao S, Li M (2018) An improved advertising CTR prediction approach based on the fuzzy deep neural network. PLoS ONE 13(5): e0190831. https://doi.org/10.1371/journal.pone.0190831

Editor: Yong Deng, Southwest University, CHINA

Received: May 7, 2017

Accepted: December 20, 2017

Published: May 4, 2018

Copyright: © 2018 Jiang et al. This is an open access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

Data Availability Statement: Data are available: kaggle-display-advertising-challenge-dataset (https://doi.org/10.6084/m9.figshare.5732310.v1); iPinYou Global RTB Bidding Algorithm Competition Dataset season2 (https://doi.org/10.6084/m9.figshare.5732328.v1).

Funding: This work is supported by the Natural Science Research Project of Education Department of Guizhou Province of China (No.KY2017350), the Project Funds of the Digital Engineering Research Center for Local National Culture of Universities in Guizhou (No.KY2013227) and the Scientific Research Project of Qiannan Normal University for

Abstract

Combining a deep neural network with fuzzy theory, this paper proposes an advertising click-through rate (CTR) prediction approach based on a fuzzy deep neural network (FDNN). In this approach, fuzzy Gaussian-Bernoulli restricted Boltzmann machine (FGBRBM) is first applied to input raw data from advertising datasets. Next, fuzzy restricted Boltzmann machine (FRBM) is used to construct the fuzzy deep belief network (FDBN) with the unsupervised method layer by layer. Finally, fuzzy logistic regression (FLR) is utilized for modeling the CTR. The experimental results show that the proposed FDNN model outperforms several baseline models in terms of both data representation capability and robustness in advertising click log datasets with noise.

Introduction

Internet advertising is regarded as an effective advertising communication approach due to its strong targeted communication ability. Therefore, increasing numbers of researchers from industry and academia have investigated internet advertising, and it has become an important source of income for internet companies.

The cost per click (CPC) model [1] is one of the most common payment models in internet advertising. Approximately 66% of advertising transactions depend on the CPC because the CPC can more accurately reveal the conversion rate compared with the other models [2]. In the CPC model, the click-through rate (CTR) is a significant index for measuring the effect of advertisement placement.

To address this task, Chapelle O. et al. propose a machine learning framework that uses Maximum Entropy to implement a logistic regression model and can address billions of samples and hundreds of millions of parameters. A two-phase feature selection algorithm is provided to reduce the need for domain expertise: a generalized mutual information method is used to select the feature groups that are used in the model; next, feature hashing is used to regulate the size of the models [3]. McMahan et al. adopted a logistic regression model to solve advertising CTR problems for Google. Using multiple characteristics, including user

Nationalities (No.qnsy2017006). The funder makes contributions on analysis, decision to publish.

Competing interests: The authors have declared that no competing interests exist.

information, search keywords, advertising data, and relative metadata, with advertisements as the input of the model, these researchers proposed an online sparse learning algorithm to the train model [4]. These above methods are recently used to address CTR prediction through the logistic regression (LR) model in industry. These methods can appropriately address a large number of features in real advertising system in industry, and can be efficiently parallelized in a distributed computing system. However, they cannot learn complex mapping relations from data because of their limited data representation capability.

T. A. Anh-Phuong presented an online learning algorithm for click-through rate prediction, known as Follow The Regularized Factorized Leader (FTRFL), which combines the Follow-The-Regularized-Leader Proximal (FTRL-P) algorithm with per-coordinate learning rates to obtain factorization machines. These researchers attempt to obtain both the sparsity provided by FTRL-Proximal and the ability to estimate higher-order features based on FM [5]. Z. Pan et al. proposed a Sparse Factorization Machine (SFM) model to solve the problem of the sparsity of the ad transaction dataset. In this model, the Laplace distribution is introduced to model the parameters because it can better fit the sparse data, which has a high ratio of zero elements. Furthermore, these researchers parallelized the SFM model on the Spark platform to support CTR prediction in a real advertising system [2]. These above methods are recently adopted to address CTR prediction through Factorization Machines (FMs) in academia. These methods can alleviate this problem of feature learning by compressing the sparse features into a dense vector space and using the second-order feature interactions. However, when facing a dataset with complex relations, they cannot sufficiently capture the higher-order non-linear representative features and complex inherent mapping relations among the users, contexts and ads in the advertising datasets.

A. I. Baqapuri and I. Trofimov adopted Artificial Neural Networks (ANNs) to predict the advertising CTR and obtained a better effect than that of the logistic regression model [6]. Dave and other researchers proposed a model that inherits the click information of rare/new ads from frequent ads that are semantically related to calculate the CTR values for newly created ads or rare ads that do not have sufficient historical information. The semantic features are derived from the search ad click-through graphs and advertiser account information. Next, gradient boosted decision trees (GBDTs) is adopted to learn from these similarity features. Experiments demonstrate that this learned model, using these features, obtains good CTR prediction performance for new ads [7]. Y Juan et al. first used trees in GBDT to generate abstract nonlinear features and then combined the original features as input features, which were fed into the FM model for the advertising prediction [8]. These above methods are recently used to address CTR prediction through models ensemble technique, and they give state-of-the-art ideas for industry and academia.

Zhang and other researchers used recurrent neural networks (RNNs) to predict the CTR of sponsored search advertising. Using back-propagation through time (BPTT) to train their model, these researchers obtained more accurate results for advertising CTR prediction than neural networks and logistic regression models [9]. Yu adopted the improved regression neural network to predict advertising CTR by using long short-term memory (LSTM) to modify RNN. This method effectively prevents the explosion or vanishing of the gradient [10]. Jiang et al. adopted the DBNLR model, which integrates a deep belief network (DBN) with logistic regression (LR) to address the problem of CTR prediction. A DBN stacked of RBMs is used to obtain abstract features from original advertisement datasets, and then a regression model is adopted to calculate the CTR prediction value [11]. Junxuan Chen et al. proposed a novel deep neural network that contains convolution layers to extract representative visual features, and fully connected layers learn the complex and effective nonlinear features based on the basic visual features. Finally, these features are fed into a logistic regression model to predict the

CTR value [12]. These above methods are state-of-the-art solutions which are adopted to address CTR prediction task through deep learning technology. In these above methods, a deep architecture model is usually used to automatically extract the features of advertising click log data, and a regression model is subsequently trained to model the advertising CTR. Because a deep architecture is used to learn the features, there is no need for these methods to depend on prior knowledge or human labor; they can effectively fit complex nonlinear mapping relations in advertising datasets. However, when the dataset contains noisy data such as missing values and outliers, these methods may exhibit performance degradation.

A common strategy for dealing with this problem is to identify and delete records with outliers during data preprocessing. However, deleting entire records results in the loss of much valuable and clean information. The other common strategy is to replace missing values and outliers with “0” or “NULL”; however, the records with outliers are preserved in the training set, which also affects the accuracy of CTR prediction.

Uncertainties not only exist in the data themselves but occur at each phase of big data processing. For instance, the collected data may be created by faulty sensors or provided by not fully informed customers; the outputs of specific artificial intelligent algorithms also contain uncertainties. In these cases, fuzzy set techniques could be one of the most efficient tools to handle various types of uncertainties [13]. Because the parameters of these above deep architecture models are constants, and the learning processes of the parameters are constrained in a relatively small space, these methods have insufficient representation capability and relatively weak robustness when the training data have been interrupted by uncertainties. Meanwhile, fuzzy set techniques would be more efficient if they are used associated with other decision making techniques, such as probability, neural networks, etc., because each type of techniques exhibit their own strengths of representing and handling information granularity [13].

Due to the lack of a comprehensive understanding of the advertising datasets with noises, it is often difficult for these methods to ensure that the extracted features capture the optimal information for predicting the CTR. These deficiencies affect the accuracy and fitness of these methods. According to the advantages of fuzzy set techniques, this paper proposes a novel CTR prediction method based on the fuzzy deep neural network (FDNN) to address advertising datasets with noise.

The main contributions of this paper can be summarized as follows:

1. This paper proposes an FDNN model in which FDBN is used to automatically extract abstract and complicated features from raw advertising datasets without any artificial intervention or prior knowledge and subsequently uses fuzzy logistic regression to model the CTR.

2. Fuzzy set techniques are introduced into the Gaussian-Bernoulli Restricted Boltzmann Machine (GBRBM), Restricted Boltzmann Machine (RBM) and logistic regression models to construct basic components of FDNN, and corresponding learning algorithms are presented in detail.

3. We conduct extensive experiments on real-world datasets to demonstrate the effectiveness and efficiency of the proposed FDNN model. The impacts of several baseline models are also discussed. Experiments results show the superiority of the proposed method. On the first dataset, the results show that the proposed method achieves competitive performances compared with the state-of-the-art models in terms of both data representation capability and robustness.

Materials and methods

Relative theories of fuzzy numbers and fuzzy functions

Fuzzy number. Definition 1. Triangular Fuzzy Number

A fuzzy number $\overline{\theta}$ (fuzzy set) on a real domain can be defined as a triangular fuzzy number if its membership function $\mu_{\overline{\theta}}(x) : R \rightarrow [0, 1]$ is equal to [14], and it is shown in S1 Fig.

$$\mu_{\overline{\theta}}(x) = \overline{\theta}(x) = \begin{cases} \frac{x}{m-l} - \frac{l}{m-l}, x \in [l, m] \\ \frac{x}{m-u} - \frac{u}{m-u}, x \in [m, u] \\ 0, otherwise \end{cases}, \tag{1}$$

where $l \leq m \leq u$, l and u stand for the lower and upper values of the support of fuzzy number $\overline{\theta}$ (fuzzy set), respectively; and m is the most likely value (middle value), that is, when $x = m$, x belongs to $\overline{\theta}$. The triangular fuzzy number can be denoted by (l, m, u). The support of $\overline{\theta}$ is the set of elements $\{x \in R | l < x < u\}$. When $l = m = u$, it is a non-fuzzy number.

Definition 2. α-cuts of a fuzzy set

The α-cut of fuzzy set $\overline{\theta}$, represented by $\overline{\theta}[\alpha]$, is defined as $\overline{\theta}[\alpha] = \{x \in \Omega | \overline{\theta} \geq \alpha\} = [\theta_L, \theta_R]$, where $0 < \alpha \leq 1$, θ_L is the lower value of $\overline{\theta}[\alpha]$ and θ_R is the upper value of $\overline{\theta}[\alpha]$ [15, 16].

Fuzzy function. Definition 3. Fuzzy Function

Extended from a real-valued function f: $Y = f(x, \boldsymbol{\theta})$, the fuzzy function $\overline{f}$ can be defined as [16]

$$\overline{Y} = \overline{f}(x, \overline{\boldsymbol{\theta}}), \tag{2}$$

where $\boldsymbol{\theta}$ and $\overline{\boldsymbol{\theta}}$ are parameters of functions f and $\overline{f}$ [16, 17], $\overline{Y}$ is the fuzzy output set.

The membership function $\overline{Y}(y)$ can be expressed as

$$\overline{Y}(y) = sup_{\theta}\{min(\overline{\theta}_1(\theta_1), ..., \overline{\theta}_n(\theta_n)) | f(x, \boldsymbol{\theta}) = y\}, \tag{3}$$

where $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_n)^T$, and $\overline{\boldsymbol{\theta}} = (\overline{\theta}_1, ..., \overline{\theta}_n)^T$

Property 1. Real Domain Interval Arithmetic

For two real domain intervals $[a, b]$ and $[c, d]$, the interval arithmetic can be defined as [16, 18]

$[a, b] + [c, d] = [a + c, b + d]$;

$[a, b] - [c, d] = [a - c, b - d]$;

$[a, b] \times [c, d] = [min(a \times c, a \times d, b \times c, b \times d), max(a \times c, a \times d, b \times c, b \times d)]$;

$[a, b] \div [c, d] = [min(a \div c, a \div d, b \div c, b \div d), max(a \div c, a \div d, b \div c, b \div d)]$.

Definition 4. α-Cuts of a Fuzzy Function

α-Cuts of $\overline{Y}$: For a continuous function f, the α-Cuts of $\overline{Y}$, namely $\overline{Y}[\alpha] = [\overline{Y}_1[\alpha], \overline{Y}_2[\alpha]]$, can be expressed as [16]

$$\begin{cases} \overline{Y}_1[\alpha] = min\{Y(\boldsymbol{\theta}, x) | \boldsymbol{\theta} \in \overline{\boldsymbol{\theta}}[\alpha]\} \\ \overline{Y}_2[\alpha] = max\{Y(\boldsymbol{\theta}, x) | \boldsymbol{\theta} \in \overline{\boldsymbol{\theta}}[\alpha]\} \end{cases}. \tag{4}$$

It is infeasible to calculate the membership function $\overline{Y}(y)$ using formulas (2) and (3) because this requires maximization and minimization of the original function. This paper adopts α-cuts and interval arithmetic to solve this problem.

$$\overline{Y}[\alpha] = f(x, \overline{\boldsymbol{\theta}}[\alpha]) \tag{5}$$

The membership function $\overline{Y}(y)$ of a fuzzy function can be obtained by interval arithmetic because intervals $\overline{\boldsymbol{\theta}}[\alpha]$ are easy to calculate. However, when f is very complex, interval arithmetic will be NP-hard; therefore, this paper introduces a defuzzification approach to handle this problem.

Defuzzification of the fuzzy function. The centroid approximate solution [19] is employed to defuzzify the fuzzy function $\overline{Y} = \overline{f}(x, \overline{\boldsymbol{\theta}})$ in this paper.

The centroid of fuzzy function $\overline{f}(x, \overline{\boldsymbol{\theta}})$ can be denoted by $f_c(x, \overline{\boldsymbol{\theta}})$ [19]:

$$f_c(x, \overline{\boldsymbol{\theta}}) = \frac{\int \boldsymbol{\theta} f(x, \boldsymbol{\theta}) d\boldsymbol{\theta}}{\int f(x, \boldsymbol{\theta}) d\boldsymbol{\theta}}, \boldsymbol{\theta} \in \overline{\boldsymbol{\theta}}. \tag{6}$$

However, directly calculating formula (6) is difficult because it involves integrals. The centroid can be approximated through many α-cuts of the fuzzy function in discrete form. $\overline{\boldsymbol{\theta}}$ is a vector of fuzzy numbers and an α-cut of $\overline{\boldsymbol{\theta}}$ can be denoted as $\overline{\boldsymbol{\theta}}[\alpha] = [\boldsymbol{\theta}_L, \boldsymbol{\theta}_R]$, where $\boldsymbol{\theta}_L$ and $\boldsymbol{\theta}_R$ are lower and upper bounds, respectively, of the interval with respect to α. When x is nonnegative, $\overline{f}(x, \overline{\boldsymbol{\theta}})$ is a monotonically decreasing function with respect to parameters $\overline{\boldsymbol{\theta}}$. Thus, according to the interval arithmetic principle, definition 3 and formula (5), the α-cut of $\overline{f}(x, \overline{\boldsymbol{\theta}})$ is given by

$$\overline{f}(x, \overline{\boldsymbol{\theta}})[\alpha] = f(x, \overline{\boldsymbol{\theta}}[\alpha]) = [f(x, \boldsymbol{\theta}_L), f(x, \boldsymbol{\theta}_R)]. \tag{7}$$

The approximate centroid of fuzzy function $\overline{f}(x, \overline{\boldsymbol{\theta}})$ can be obtained by [20]

$$f_c(x, \overline{\boldsymbol{\theta}}) \approx \frac{\sum_{i=1}^{M} \alpha_i [f(x, \boldsymbol{\theta}_{iL}), f(x, \boldsymbol{\theta}_{iR})]}{2 \sum_{i=1}^{M} \alpha_i}, \tag{8}$$

where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_N)$, $\boldsymbol{\alpha} \in [0, 1]^N$ and $\overline{\boldsymbol{\theta}}[\alpha_i] = [\boldsymbol{\theta}_{iL}, \boldsymbol{\theta}_{iR}]$.

Although all the α-cutsare bounded intervals, this paper takes only the special case where $\alpha_i = 1$ into consideration. Let $\boldsymbol{\theta} = [\boldsymbol{\theta}_L, \boldsymbol{\theta}_R]$. Formula (8) can be expressed as [20]

$$f_c(x, \overline{\boldsymbol{\theta}}) \approx \frac{1}{2}[f(x, \boldsymbol{\theta}_L) + f(x, \boldsymbol{\theta}_R)]. \tag{9}$$

Therefore, fuzzy function $\overline{Y}$ can be expressed as

$$\overline{Y} = \overline{f}(x, \overline{\boldsymbol{\theta}}) = f_c(x, \overline{\boldsymbol{\theta}}) \approx \frac{1}{2}[f(x, \boldsymbol{\theta}_L) + f(x, \boldsymbol{\theta}_R)]. \tag{10}$$

Advertising CTR prediction based on the fuzzy deep neural network

Basic fuzzy components and corresponding learning algorithms. Inspired by the idea of above fuzzy theory and methods [16–20], we first provide a detailed introduction and describe the corresponding learning algorithms of several basic components that have been modified using fuzzy technology.

A. FRBM and its Learning Algorithm

Fuzzy restricted Boltzmann machine (FRBM) is a symmetric neural network with binary nodes that is based on an energy model. It contains a set of visual binary nodes $\boldsymbol{v} \in \{0, 1\}^D$ and another set of hidden binary nodes $\boldsymbol{h} \in \{0, 1\}^F$. There are no connections between different nodes of the same hidden layer or between different nodes of the same visual layer. Fuzzy parameters are employed to govern the FRBM model. The structure of FRBM is described in S2 Fig.

The fuzzy energy function of FRBM can be expressed as [20]

$$\overline{E}(\boldsymbol{x}, \boldsymbol{h}; \overline{\boldsymbol{\theta}}) = -\overline{\boldsymbol{b}}^T \boldsymbol{x} - \overline{\boldsymbol{c}}^T \boldsymbol{h} - \boldsymbol{h}^T \overline{\boldsymbol{W}} \boldsymbol{x}. \tag{11}$$

In this formula, $\overline{\boldsymbol{\theta}} = \{\overline{\boldsymbol{b}}, \overline{\boldsymbol{c}}, \overline{\boldsymbol{W}}\}$, $\overline{\boldsymbol{b}}$ and $\overline{\boldsymbol{c}}$ are the offsets, and $\overline{\boldsymbol{W}}$ is the connection weight between the ith visible node and the jth hidden node.

According to the free energy function $F(\boldsymbol{x}, \boldsymbol{\theta}) = -log\sum_{\tilde{\boldsymbol{h}}} e^{-E(\boldsymbol{x},\tilde{\boldsymbol{h}},\boldsymbol{\theta})}$, the fuzzy free energy function $\overline{F}$ is expressed as

$$\overline{F}(\boldsymbol{x}, \overline{\boldsymbol{\theta}}) = -log\sum_{\tilde{\boldsymbol{h}}} e^{-\overline{E}(\boldsymbol{x},\tilde{\boldsymbol{h}},\overline{\boldsymbol{\theta}})}. \tag{12}$$

If $\overline{F}(\boldsymbol{x}, \overline{\boldsymbol{\theta}})$ is used directly to define the probability, it is difficult to calculate the fuzzy probability, fuzzy maximum likelihood and fuzzy objective function in the optimization. Therefore, $\overline{F}(\boldsymbol{x}, \overline{\boldsymbol{\theta}})$ needs to be defuzzified to transform the optimization into regular maximum-likelihood optimization. This paper adopts a centroid method to defuzzify $\overline{F}(\boldsymbol{x}, \overline{\boldsymbol{\theta}})$.

The centroid of $\overline{F}(\boldsymbol{x}, \overline{\boldsymbol{\theta}})$ is expressed as $F_c(\boldsymbol{x}, \overline{\boldsymbol{\theta}})$, and the probability can be defined as

$$P_c(\boldsymbol{x}, \overline{\boldsymbol{\theta}}) = \frac{e^{-F_c(\boldsymbol{x},\overline{\boldsymbol{\theta}})}}{Z}, Z = \sum_{\tilde{\boldsymbol{x}}} e^{-F_c(\tilde{\boldsymbol{x}},\overline{\boldsymbol{\theta}})}. \tag{13}$$

This paper selects the negative log-likelihood as the objective function of FRBM:

$$L(\overline{\boldsymbol{\theta}}, D) = -\sum_{\boldsymbol{x}\in D} logP_c(\boldsymbol{x}, \overline{\boldsymbol{\theta}}). \tag{14}$$

In the formula, D denotes the training dataset.

The learning algorithm of FRBM finds the parameters $\overline{\boldsymbol{\theta}}$ that minimize the objective function $L(\overline{\boldsymbol{\theta}}, D)$ $(minL_{\overline{\theta}}(\overline{\boldsymbol{\theta}}, D))$. The paper adopts the centroid approximation method to defuzzify $\overline{F}(\boldsymbol{x}, \overline{\boldsymbol{\theta}})$:

$$\overline{F}(\boldsymbol{x}, \overline{\boldsymbol{\theta}}) = F_c(\boldsymbol{x}, \overline{\boldsymbol{\theta}}) \approx \frac{1}{2}[F(\boldsymbol{x}, \boldsymbol{\theta}_L) + F(\boldsymbol{x}, \boldsymbol{\theta}_R)]. \tag{15}$$

The gradients of $L(\overline{\boldsymbol{\theta}}, D)$ with respect to $\boldsymbol{\theta}_L$ can be expressed as

$$-\frac{\partial logP_c(\boldsymbol{x}, \overline{\boldsymbol{\theta}})}{\partial \boldsymbol{\theta}_L} = \frac{\partial F_c(\boldsymbol{x}, \boldsymbol{\theta}_L)}{\partial \boldsymbol{\theta}_L} - E_p\left[\frac{\partial F_c(\boldsymbol{x}, \boldsymbol{\theta}_L)}{\partial \boldsymbol{\theta}_L}\right]. \tag{16}$$

In the formula, $E_p(\cdot)$ is the expectation over the target probability distribution P.

Correspondingly,

$$-\frac{\partial logP_c(\boldsymbol{x}, \overline{\boldsymbol{\theta}})}{\partial \boldsymbol{\theta}_R} = \frac{\partial F_c(\boldsymbol{x}, \boldsymbol{\theta}_R)}{\partial \boldsymbol{\theta}_R} - E_p\left[\frac{\partial F_c(\boldsymbol{x}, \boldsymbol{\theta}_R)}{\partial \boldsymbol{\theta}_R}\right]. \tag{17}$$

Because it is difficult to calculate the expectation $E_p\left[\frac{\partial F_c(\boldsymbol{x},\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\right]$, an approximation approach is employed. After defuzzifying the objective function, Gibbs sampling [21] is adopted to sample from these conditional distributions.

For FRBM, the fuzzy conditional probabilities can be expressed as $\overline{P}(h_j = 1|\boldsymbol{x}) = \sigma(\overline{c}_j + \overline{W}_{\cdot j}\boldsymbol{x})$ and $\overline{P}(x_i = 1|\boldsymbol{h}) = \sigma(\overline{b}_i + \overline{W}_{i\cdot}^T\boldsymbol{h})$. The α-cuts of the fuzzy conditional probabilities are described as $\overline{P}(h_j = 1|\boldsymbol{x})[\alpha] = [P_L(h_j = 1|\boldsymbol{x}), P_R(h_j = 1|\boldsymbol{x})]$ and $\overline{P}(x_i = 1|\boldsymbol{h})[\alpha] = [P_L(x_i = 1|\boldsymbol{h}), P_R(x_i = 1|\boldsymbol{h})]$.

In these formulas, $P_L(h_j|\boldsymbol{x})$, $P_R(h_j|\boldsymbol{x})$, $P_L(x_i|\boldsymbol{h})$ and $P_R(x_i|\boldsymbol{h})$ are the conditional probabilities with respect to the lower bounds and upper bounds of the parameters. Consequently,

$$\begin{aligned} P_L(h_j|\boldsymbol{x}) &= P(h_j|\boldsymbol{x};\boldsymbol{\theta}_L) = \sigma(c_j^L + W_{\cdot j}^L\boldsymbol{x}), \\ P_R(h_j|\boldsymbol{x}) &= P(h_j|\boldsymbol{x};\boldsymbol{\theta}_R) = \sigma(c_j^R + W_{\cdot j}^R\boldsymbol{x}), \end{aligned} \tag{18}$$

and

$$\begin{aligned} P_L(x_i|\boldsymbol{h}) &= P(x_i|\boldsymbol{h};\boldsymbol{\theta}_L) = \sigma(b_i^L + W_{i\cdot}^L\boldsymbol{h}), \\ P_R(x_i|\boldsymbol{h}) &= P(x_i|\boldsymbol{h};\boldsymbol{\theta}_R) = \sigma(b_i^R + W_{i\cdot}^R\boldsymbol{h}). \end{aligned} \tag{19}$$

In these formulas, W_{ij}^L and W_{ij}^R are the lower bound and upper bound of the connection weight, b_i^L and b_i^R are the lower bound and upper bound of the visible bias, and c_j^L and c_j^R are the lower bound and upper bound of the hidden bias [20].

According to (15), (16), (17) and the expectation estimation method of [20, 22], the gradients of $L(\overline{\boldsymbol{\theta}}, D)$ with respect to the fuzzy parameters of FRBM can be obtained as follows:

$$-\frac{\partial logP_c(\boldsymbol{x})}{\partial W_{ij}^L} = E_p[P_L(h_j|\boldsymbol{x})\cdot x_i^L] - P_L(h_j|\boldsymbol{x})\cdot x_i^L,$$

$$-\frac{\partial logP_c(\boldsymbol{x})}{\partial c_j^L} = E_p[P_L(h_j|\boldsymbol{x})] - P_L(h_j|\boldsymbol{x}),$$

$$-\frac{\partial logP_c(\boldsymbol{x})}{\partial b_i^L} = E_p[P_L(x_i|\boldsymbol{h})] - x_i^L,$$

$$-\frac{\partial logP_c(\boldsymbol{x})}{\partial W_{ij}^R} = E_p[P_R(h_j|\boldsymbol{x})\cdot x_i^R] - P_R(h_j|\boldsymbol{x})\cdot x_i^R,$$

$$-\frac{\partial logP_c(\boldsymbol{x})}{\partial c_j^R} = E_p[P_R(h_j|\boldsymbol{x})] - P_R(h_j|\boldsymbol{x}),$$

$$-\frac{\partial logP_c(\boldsymbol{x})}{\partial b_i^R} = E_p[P_R(x_i|\boldsymbol{h})] - x_i^R,$$

where $P_c(\boldsymbol{x})$ is the centroid probability.

The CD1 algorithm [22] is employed to obtain the updating rules for parameters (θ_L and θ_R) to approximate the expectation $E_p\left[\frac{\partial F_c(\boldsymbol{x},\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\right]$ [20, 22]. The above description is summarized as Algorithm 1.

Algorithm 1 The learning algorithm of FRBM

```
Input: $\boldsymbol{x}^{(0)}$ is an input sample of the training set;
    $\varepsilon$ is the learning rate;
    $\boldsymbol{W}^L$ and $\boldsymbol{W}^R$ are the lower and upper bounds of connection weight matrices of the visible and hidden layers;
    $\boldsymbol{b}^L$ and $\boldsymbol{b}^R$ are the lower and upper bounds of bias vectors of the visible nodes;
    $\boldsymbol{c}^L$ and $\boldsymbol{c}^R$ are the lower and upper bounds of bias vectors of the hidden nodes.
Output: The latest parameters of FRBM: $\boldsymbol{W}^L$, $\boldsymbol{W}^R$, $\boldsymbol{b}^L$, $\boldsymbol{b}^R$, $\boldsymbol{c}^L$, $\boldsymbol{c}^R$.
```

```
BEGIN
1. For all hidden nodes j do
   obtain $P_L(h_j^{L(0)} = 1|\boldsymbol{x}^{(0)})$ and $P_R(h_j^{R(0)} = 1|\boldsymbol{x}^{(0)})$ by means of formula (18);
   sample $h_j^{L(0)} \in \{0,1\}$ from $P_L(h_j^{L(0)}|\boldsymbol{x}^{(0)})$;
   sample $h_j^{R(0)} \in \{0,1\}$ from $P_R(h_j^{R(0)}|\boldsymbol{x}^{(0)})$;
   end for
2. For all visible nodes i do
   gain $P_L(x_i^{L(1)} = 1|\boldsymbol{h}^{L(0)})$ and $P_R(x_i^{R(1)} = 1|\boldsymbol{h}^{R(0)})$ by means of formula (19);
   sample $x_i^{L(1)} \in \{0,1\}$ from $P_L(x_i^{L(1)}|\boldsymbol{h}^{L(0)})$;
   sample $x_i^{R(1)} \in \{0,1\}$ from $P_R(x_i^{R(1)}|\boldsymbol{h}^{R(0)})$;
   end for
3. For all hidden nodes j do
   obtain $P_L(h_j^{L(1)} = 1|\boldsymbol{x}^{L(1)})$ and $P_R(h_j^{R(1)} = 1|\boldsymbol{x}^{R(1)})$ by means of formula (18);
   sample $h_j^{L(1)} \in \{0,1\}$ from $P_L(h_j^{L(1)}|\boldsymbol{x}^{L(1)})$;
   sample $h_j^{R(1)} \in \{0,1\}$ from $P_R(h_j^{R(1)}|\boldsymbol{x}^{R(1)})$;
   end for
4. The parameters can be updated according to the following rules:
   $\boldsymbol{W}^L = \boldsymbol{W}^L + \varepsilon(\boldsymbol{x}^{(0)} \cdot P_L(\boldsymbol{h}^{L(0)} = 1|\boldsymbol{x}^{(0)}) - \boldsymbol{x}^{L(1)} \cdot P_L(\boldsymbol{h}^{L(1)} = 1|\boldsymbol{x}^{L(1)}))$;
   $\boldsymbol{b}^L = \boldsymbol{b}^L + \varepsilon(\boldsymbol{x}^{(0)} - \boldsymbol{x}^{L(1)})$
   $\boldsymbol{c}^L = \boldsymbol{c}^L + \varepsilon(P_L(\boldsymbol{h}^{L(0)} = 1|\boldsymbol{x}^{(0)} - P_L(\boldsymbol{h}^{L(1)} = 1|\boldsymbol{x}^{L(1)}))$;
   $\boldsymbol{W}^R = \boldsymbol{W}^R + \varepsilon(\boldsymbol{x}^{(0)} \cdot P_R(\boldsymbol{h}^{R(0)} = 1|\boldsymbol{x}^{(0)}) - \boldsymbol{x}^{R(1)} \cdot P_R(\boldsymbol{h}^{R(1)} = 1|\boldsymbol{x}^{R(1)}))$;
   $\boldsymbol{b}^R = \boldsymbol{b}^R + \varepsilon(\boldsymbol{x}^{(0)} - \boldsymbol{x}^{R(1)})$;
   $\boldsymbol{c}^R = \boldsymbol{c}^R + \varepsilon(P_R(\boldsymbol{h}^{R(0)} = 1|\boldsymbol{x}^{(0)} - P_R(\boldsymbol{h}^{R(1)} = 1|\boldsymbol{x}^{R(1)}))$;
5. return $\boldsymbol{W}^L$, $\boldsymbol{W}^R$, $\boldsymbol{b}^L$, $\boldsymbol{b}^R$, $\boldsymbol{c}^L$, $\boldsymbol{c}^R$.
END
```

B. FGBRBM and Its Learning Algorithm

The nodes of the visible and hidden layers in the fuzzy Gaussian-Bernoulli restricted Boltzmann machine (FGBRBM) model correspond to the Gaussian-Bernoulli nodes. In other words, the nodes of visible layer $\boldsymbol{v} \in R^D$ are Gaussian nodes and the nodes of the first hidden layer $\boldsymbol{h} \in \{0, 1\}^F$ are Bernoulli nodes (binary nodes) [23]. Their structures are described in S3 Fig. The fuzzy energy function of FGBRBM can be defined as

$$\overline{E}(\boldsymbol{v}, \boldsymbol{h}; \overline{\boldsymbol{\theta}}) = -\left(\boldsymbol{v}^T \cdot \frac{1}{\boldsymbol{\sigma}}\right)^T \overline{\boldsymbol{W}}\boldsymbol{h} - \frac{1}{2}\left(\boldsymbol{v}^T \cdot \frac{1}{\boldsymbol{\sigma}} - \overline{\boldsymbol{b}}^T \cdot \frac{1}{\boldsymbol{\sigma}}\right)^2 - \overline{\boldsymbol{b}}^T \boldsymbol{h}, \tag{20}$$

where σ_i denotes the standard deviation of v_i in the ith dimension of the Gaussian visual node.

Correspondingly, as for FGBRBM, the fuzzy conditional probability distribution for obtaining nodes of the hidden layer through nodes of the visual layer can be described as:

$$\overline{P}(h_j = 1|\boldsymbol{v}) = g\left(\overline{c}_j + \overline{W}_{i\cdot}\right) = g\left(\overline{c}_j + \sum_i \frac{v_i}{\sigma_i}\overline{W}_{ij}\right), \tag{21}$$

where $g(\cdot)$ denotes the *sigmoid* function.

The formula for constructing the nodes of the visual layer through nodes of the hidden layer can be expressed as

$$v_i = N(\overline{b}_i + \sigma_i \sum_{j=1}^{F} \overline{W}_{ij} h_j, \sigma_i^2). \tag{22}$$

In this formula, $N(\mu, \sigma^2)$ is a Gaussian probability density function, where μ is the mean value and σ^2 denotes the variance.

The α-cuts of fuzzy conditional probabilities can be expressed as $\overline{P}(h_j = 1|\boldsymbol{v})[\alpha] = [P_L(h_j = 1|\boldsymbol{v}), P_R(h_j = 1|\boldsymbol{v})], \overline{v}_i[\alpha] = N(\overline{b}_i + \sigma_i \sum_{j=1}^{F} \overline{W}_{ij} h_j, \sigma_i^2)[\alpha] = [v_{iL}, v_{iR}]$, where $P_L(h_j|\boldsymbol{v})$ and $P_R(h_j|\boldsymbol{v})$ are the conditional probabilities with respect to the lower and upper bounds of the parameters governing the FGBRBM, and v_{iL} and v_{iR} are the probabilities of the normal distribution. Consequently,

$$\begin{aligned} P_L(h_j|\boldsymbol{v}) &= P(h_j|\boldsymbol{v};\boldsymbol{\theta}_L) = \sigma(c_j^L + W_{\cdot j}^L \boldsymbol{v}), \\ P_R(h_j|\boldsymbol{v}) &= P(h_j|\boldsymbol{v};\boldsymbol{\theta}_R) = \sigma(c_i^R + W_{\cdot j}^R \boldsymbol{v}), \end{aligned} \tag{23}$$

and

$$\begin{aligned} v_{iL} &= N\left(b_i^L + \sigma_i \sum_{j=1}^{F} W_{ij}^L h_j, \sigma_i^2\right), \\ v_{iR} &= N\left(b_i^R + \sigma_i \sum_{j=1}^{F} W_{ij}^R h_j, \sigma_i^2\right), \end{aligned} \tag{24}$$

where W_{ij}^L and W_{ij}^R are the lower and upper bounds of the connection weight, respectively; b_i^L and b_i^R are the lower and upper bounds of visible node bias, respectively; and c_j^L and c_j^R are the lower and upper bounds of the hidden node bias, respectively.

The CD-1 algorithm can also be used for learning parameters of the fuzzy Gaussian-Bernoulli RBM model. In the process of training, the input data of the whole training set should first be normalized so that each dimension of the input data in the FGBRBM model obeys the normal distribution, i.e., the mean value is 0 and the variance is 1. When the formula $P(v_i = 1|\boldsymbol{h}, \overline{\boldsymbol{\theta}})$ is calculated, $\sigma = 1$. In the process of reconstructing the nodes of the visual layer, this paper does not adopt binary data instead of the probabilities of the normal distribution. The above description is summarized as Algorithm 2.

Algorithm 2 The learning algorithm of FGBRBM

```
Input: $\boldsymbol{x}^{(0)}$ is a sample of the training dataset;
    $\varepsilon$ is the learning rate;
    $\boldsymbol{W}^L$ and $\boldsymbol{W}^R$ are the lower and upper bounds of connection weight matrices of the visible and hidden layers;
    $\boldsymbol{b}^L$ and $\boldsymbol{b}^R$ are the lower and upper bounds of bias vectors of the visible nodes;
    $\boldsymbol{c}^L$ and $\boldsymbol{c}^R$ are the lower and upper bounds of bias vectors of the hidden nodes.
Output: The latest parameters of FGBRBM: $\boldsymbol{W}^L$, $\boldsymbol{W}^R$, $\boldsymbol{b}^L$, $\boldsymbol{b}^R$, $\boldsymbol{c}^L$, $\boldsymbol{c}^R$.
  BEGIN
  1. For all hidden nodes j do
     obtain $P_L(h_j^{L(0)} = 1|\boldsymbol{v}^{(0)})$ and $P_R(h_j^{R(0)} = 1|\boldsymbol{v}^{(0)})$ by means of formula (23);
     sample $h_j^{L(0)} \in \{0,1\}$ from $P_L(h_j^{L(0)}|\boldsymbol{v}^{(0)})$;
     sample $h_j^{R(0)} \in \{0,1\}$ from $P_R(h_j^{R(0)}|\boldsymbol{v}^{(0)})$;
     end for
  2. For all visible nodes i do
     obtain $v_{iL}$ and $v_{iR}$ by means of formula (24);
     $v_i^{L(1)} = N(b_i^L + \sum_{j=1}^{F} W_{ij}^L h_j^{L(0)}, 1)$;
     $v_i^{R(1)} = N(b_i^R + \sum_{j=1}^{F} W_{ij}^R h_j^{R(0)}, 1)$;
     end for
  3. For all hidden nodes j do
     obtain $P_L(h_j^{L(1)} = 1|\boldsymbol{v}^{L(1)})$ and $P_R(h_j^{R(1)} = 1|\boldsymbol{v}^{R(1)})$ by means of formula (23);
```

```
        sample $h_j^{L(1)} \in \{0,1\}$ from $P_L(h_j^{L(1)}|\boldsymbol{v}^{L(1)})$;
        sample $h_j^{R(1)} \in \{0,1\}$ from $P_R(h_j^{R(1)}|\boldsymbol{v}^{R(1)})$;
        end for
    4. The parameters can be updated according to the following rules:
        $\boldsymbol{W}^L = \boldsymbol{W}^L + \varepsilon(\boldsymbol{v}^{(0)} \cdot P_L(\boldsymbol{h}^{L(0)} = 1|\boldsymbol{v}^{(0)}) - \boldsymbol{v}^{L(1)} \cdot P_L(\boldsymbol{h}^{L(1)} = 1|\boldsymbol{v}^{L(1)}))$;
        $\boldsymbol{b}^L = \boldsymbol{b}^L + \varepsilon(\boldsymbol{v}^{(0)} - \boldsymbol{v}^{L(1)})$
        $\boldsymbol{c}^L = \boldsymbol{c}^L + \varepsilon(P_L(\boldsymbol{h}^{L(0)} = 1|\boldsymbol{v}^{(0)} - P_L(\boldsymbol{h}^{L(1)} = 1|\boldsymbol{v}^{L(1)}))$;
        $\boldsymbol{W}^R = \boldsymbol{W}^R + \varepsilon(\boldsymbol{v}^{(0)} \cdot P_R(\boldsymbol{h}^{R(0)} = 1|\boldsymbol{v}^{(0)}) - \boldsymbol{v}^{R(1)} \cdot P_R(\boldsymbol{h}^{R(1)} = 1|\boldsymbol{v}^{R(1)}))$;
        $\boldsymbol{b}^R = \boldsymbol{b}^R + \varepsilon(\boldsymbol{v}^{(0)} - \boldsymbol{v}^{R(1)})$
        $\boldsymbol{c}^R = \boldsymbol{c}^R + \varepsilon(P_R(\boldsymbol{h}^{R(0)} = 1|\boldsymbol{v}^{(0)} - P_R(\boldsymbol{h}^{R(1)} = 1|\boldsymbol{v}^{R(1)}))$;
    5. return $\boldsymbol{W}^L$, $\boldsymbol{W}^R$, $\boldsymbol{b}^L$, $\boldsymbol{b}^R$, $\boldsymbol{c}^L$, $\boldsymbol{c}^R$.
    END
```

C. FLR and Its Learning Algorithm Used for Modeling the Click Through Rate

The logistic regression (LR) model [24] is a classic model in the field of advertising click-rate prediction. The LR model is described in S4 Fig. The activation function of the node in the output layer is a sigmoid function [25]. The output value of the node (value of the activation function) $h_\theta(\boldsymbol{x}')$ is the probability that a user will click on the advertisement; it can also be described as $P(y = 1|\boldsymbol{x}'; \boldsymbol{\theta}')$. When a sample $(\boldsymbol{x}'_d, y_d)$ in the training set $D = (\boldsymbol{x}'_1, y_1), (\boldsymbol{x}'_2, y_2), \ldots, (\boldsymbol{x}'_N, y_N)$ is given, the probability value of advertising click rate prediction can be obtained in terms of logistic regression and can be described by formula (25)

$$p(click|\boldsymbol{a}, \boldsymbol{u}, \boldsymbol{c}) = p(y_d = 1|\boldsymbol{x}'_d; \boldsymbol{\theta}') = \frac{1}{1 + e^{-\boldsymbol{\theta}'^T \boldsymbol{x}'_d}}. \tag{25}$$

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

In this formula, $\boldsymbol{\theta}'^T$ denotes the vector of parameters of the standardized logistic regression model, $\boldsymbol{a}$ refers to the features of the advertisement, $\boldsymbol{u}$ denotes the features of users, and $\boldsymbol{c}$ represents the features of the context environment. These three features compose a vector, which can be denoted as $\boldsymbol{x}'_d$. This vector is fed into the logistic regression model for CTR prediction.

For a single sample $(\boldsymbol{x}'_d, y_d)$, the probability of correctly predicting the advertising click-through rate can be described as $P(y_d|\boldsymbol{x}'_d; \boldsymbol{\theta}') = h_{\boldsymbol{\theta}'}(\boldsymbol{x}'_d)^{y_d}(1 - h_{\boldsymbol{\theta}'}(\boldsymbol{x}'_d))^{(1-y_d)}$. This paper selects the likelihood function as the objective function:

$$\begin{aligned} J(\boldsymbol{x}'_d; \boldsymbol{\theta}') &= -\frac{1}{N} log(P(Y_D|\boldsymbol{X}'_D; \boldsymbol{\theta}')) \\ &= -\frac{1}{N} log\left(\prod_{d=1}^{N} (h_{\boldsymbol{\theta}'}(\boldsymbol{x}'_d)^{y_d}(1 - h_{\boldsymbol{\theta}'}(\boldsymbol{x}'_d))^{1-y_d})\right) \\ &= -\frac{1}{N} \sum_{d=1}^{N} (y_d log h_{\boldsymbol{\theta}'}(\boldsymbol{x}'_d) + (1 - y_d) log(1 - h_{\boldsymbol{\theta}'}(\boldsymbol{x}'_d))). \end{aligned} \tag{26}$$

In formula (26), the click label y_d denotes the expected output and $h_{\boldsymbol{\theta}'}(\boldsymbol{x}'_d)$ is the real output value of the neuron node in the output layer (output of the activation function); $h_{\theta'}(\boldsymbol{x}'_d)$ can be described as $h_{\boldsymbol{\theta}'}(\boldsymbol{x}'_d) = g(\boldsymbol{z})$, where $\boldsymbol{z} = \sum_i (w_i x_i + b) = \boldsymbol{\theta}'^T \boldsymbol{x}'_d$ and $g(x)$ denotes the *sigmoid* function.

As for the fuzzy logistic regression model, the objective function can be described as

$$\begin{aligned}\overline{J}(\boldsymbol{x}'_d;\overline{\boldsymbol{\theta}'}) &= -\frac{1}{N}log(\overline{P}(Y_D|\boldsymbol{X}'_D;\overline{\boldsymbol{\theta}'})) \\ &= -\frac{1}{N}log\left(\prod_{d=1}^{N}(\overline{h}_{\overline{\boldsymbol{\theta}'}}(\boldsymbol{x}'_d)^{y_d}(1-\overline{h}_{\overline{\boldsymbol{\theta}'}}(\boldsymbol{x}'_d))^{1-y_d})\right) \\ &= -\frac{1}{N}\sum_{d=1}^{N}(y_d log\overline{h}_{\overline{\boldsymbol{\theta}'}}(\boldsymbol{x}'_d)+(1-y_d)log(1-\overline{h}_{\overline{\boldsymbol{\theta}'}}(\boldsymbol{x}'_d))),\end{aligned} \tag{27}$$

where $\overline{h}_{\overline{\boldsymbol{\theta}'}}(\boldsymbol{x}'_d)=\overline{g}(\overline{z}),\overline{z}=\sum_i(\overline{w_i}x_i+\overline{b})=\overline{\boldsymbol{\theta}'}^T\boldsymbol{x}'_d$ and $g(x)=\frac{1}{1+e^{-x}}$.

According to formulas (5) and (7), the gradient of the objective function of FLR with respect to fuzzy parameter $\overline{\boldsymbol{\theta}'}$ can be expressed as

$$\begin{aligned}\frac{\partial\overline{J}(\boldsymbol{x}'_d;\overline{\boldsymbol{\theta}'})}{\partial\overline{\boldsymbol{\theta}'}_L} &= \frac{\partial J_c(\boldsymbol{x}'_d;\overline{\boldsymbol{\theta}'})}{\partial\overline{\boldsymbol{\theta}'}_L} \\ &= -\frac{1}{N}\sum_{d=1}^{N}(y_d\frac{1}{h_{\boldsymbol{\theta}'_L}(\boldsymbol{x}'_d)}\frac{\partial h_{\boldsymbol{\theta}'_L}(\boldsymbol{x}'_d)}{\partial\boldsymbol{\theta}'_L}-(1-y_d)\frac{1}{1-h_{\boldsymbol{\theta}'_L}(\boldsymbol{x}'_d)}\frac{\partial h_{\boldsymbol{\theta}'_L}(\boldsymbol{x}'_d)}{\partial\boldsymbol{\theta}'_L}) \\ &= \frac{1}{N}\sum_{d=1}^{N}(h_{\boldsymbol{\theta}'_L}(\boldsymbol{x}'_d)-y_d)\boldsymbol{x}'_d \\ &= C\sum_{d=1}^{N}(h_{\boldsymbol{\theta}'_L}(\boldsymbol{x}'_d)-y_d)\boldsymbol{x}'_d,\end{aligned} \tag{28}$$

$$\begin{aligned}\frac{\partial\overline{J}(\boldsymbol{x}'_d;\overline{\boldsymbol{\theta}'})}{\partial\overline{\boldsymbol{\theta}'}_R} &= \frac{\partial J_c(\boldsymbol{x}'_d;\overline{\boldsymbol{\theta}'})}{\partial\overline{\boldsymbol{\theta}'}_R} \\ &= -\frac{1}{N}\sum_{d=1}^{N}(y_d\frac{1}{h_{\boldsymbol{\theta}'_R}(\boldsymbol{x}'_d)}\frac{\partial h_{\boldsymbol{\theta}'_R}(\boldsymbol{x}'_d)}{\partial\boldsymbol{\theta}'_R}-(1-y_d)\frac{1}{1-h_{\boldsymbol{\theta}'_R}(\boldsymbol{x}'_d)}\frac{\partial h_{\boldsymbol{\theta}'_R}(\boldsymbol{x}'_d)}{\partial\boldsymbol{\theta}'_R}) \\ &= \frac{1}{N}\sum_{d=1}^{N}(h_{\boldsymbol{\theta}'_R}(\boldsymbol{x}'_d)-y_d)\boldsymbol{x}'_d \\ &= C\sum_{d=1}^{N}(h_{\boldsymbol{\theta}'_R}(\boldsymbol{x}'_d)-y_d)\boldsymbol{x}'_d.\end{aligned} \tag{29}$$

The above description is summarized as Algorithm 3.

Algorithm 3 The learning algorithm of FLR

```
Input: $(\boldsymbol{x}'_d, y_d)$ is a sample of training sets;
    $\varepsilon$ is the learning rate;
    $\boldsymbol{\theta}'_L=\{\boldsymbol{W}^L,\boldsymbol{c}^L\}$ and $\boldsymbol{\theta}'_R=\{\boldsymbol{W}^R,\boldsymbol{c}^R\}$, where $\boldsymbol{W}^L$ and $\boldsymbol{W}^R$ are the lower and upper
bounds of connection weight matrices of FLR, and $\boldsymbol{c}^L$ and $\boldsymbol{c}^R$ are the lower
and upper bounds of bias vectors of the output node of FLR;
Output: The latest parameters of FLR: $\boldsymbol{\theta}'_L=\{\boldsymbol{W}^L,\boldsymbol{c}^L\}$, $\boldsymbol{\theta}'_R=\{\boldsymbol{W}^R,\boldsymbol{c}^R\}$.
  BEGIN
  1. obtain $\frac{\partial J_c(\overline{\boldsymbol{\theta}'})}{\partial\boldsymbol{\theta}'_L}$ by means of formula (28) based on the input data $\boldsymbol{x}'_d$;
```

```
2. obtain $\frac{\partial J_c(\overline{\theta'})}{\partial \theta'_R}$ by means of formula (29) based on the input data $x'_d$;
3. The parameters can be updated according to the following rules:
   $\theta'_L = \theta'_L + \varepsilon \frac{\partial J_c(\overline{\theta'})}{\partial \theta'_L}$;
   $\theta'_R = \theta'_R + \varepsilon \frac{\partial J_c(\overline{\theta'})}{\partial \theta'_R}$;
4. return $\theta'_L, \theta'_R$.
END
```

Advertising CTR prediction model FDNN. *A. Construction of Advertising CTR Prediction Model FDNN*

This section describes the proposed fuzzy deep neural network (FDNN) model for advertising CTR prediction and its learning algorithm in detail. The architecture of the method and its process of predicting CTR are illustrated in S5 Fig.

First, the fuzzy deep belief network (FDBN) constituted by FRBM and GBRBM is used to automatically learn the discriminative features of raw input data from an advertising dataset, and these features are able to capture internal explanatory information that is hidden in the raw data, amplify the important information for discrimination and weaken irrelevant information [26].

Next, FLR is adopted to model the CTR prediction and map the predicted CTR value to the range between 0 and 1.

The raw input data of advertising CTR prediction are the fields that include the corresponding advertisement features, users' features, and context features, which are extracted from the advertisement click log. These fields contain not only numerical data but also multi-category data; therefore, the raw input data of advertising CTR prediction are composed of vectors with different data types.

Because FRBM only models input with binary variance (0, 1), this paper adopts fuzzy Gaussian-Bernoulli RBM (FGBRBM) to address the input data of advertising CTR prediction.

Then, FRBM can be used to construct FDBN through layer-by-layer stacking; the detailed steps can be found in [22].

In real applications, advertising click rate prediction refers to binary classification or regression; therefore, this paper uses fuzzy logistic regression to model user behaviors in clicking advertisements. After the input data vector $\boldsymbol{x} = (\boldsymbol{a}, \boldsymbol{u}, \boldsymbol{c}) = (x_1, x_2, \ldots, x_n)$ is extracted by FDBN, the vector $\boldsymbol{x}'$, which is composed of the binary values, in the last hidden layer of FDBN is fed into fuzzy the logistic regression model for CTR prediction.

The advantage of the proposed method is that fuzzy set techniques are introduced, and the uncertainties in the relationships between nodes located in adjacent layers is taken into consideration. Nodes in adjacent layers of FDNN often interact in uncertain ways. Because the parameters that represent the relationships between nodes in adjacent layers are fuzzy numbers and the learning process of fuzzy parameters is extended to a relatively wider space, this advantage will be reflected in the fitness of the joint probability distribution. Combined with the merits of deep learning, the proposed method demonstrates superior performance in coping with data with noises.

B. Construction and Training of the Advertising CTR Prediction Model FDNN

The process of constructing FDNN is illustrated in S6 Fig, and the detailed steps of constructing the FDNN are as follows:

Step 1: The training of FRBM layer by layer

First, the vectors that are constituted by related data of textual advertising click logs are regarded as the input of FGBRBM, and the CD1 algorithm is used to train FGBRBM; in this way, the parameter $\boldsymbol{\theta}_1$ of FGBRBM is obtained. Based on the input vectors and the parameter $\boldsymbol{\theta}_1$, the hidden nodes $\boldsymbol{h}_1$ of FGBRBM are obtained.

Second, the parameter $\boldsymbol{\theta}_2$ of the current FRBM can be obtained when $\boldsymbol{h}_1$ is regarded as the input data for the CD1 algorithm. Based on $\boldsymbol{h}_1$ and $\boldsymbol{\theta}_2$, the hidden nodes $\boldsymbol{h}_2$ of the current FRBM can be obtained.

Third, the previous process is repeated until reaching the fixed layer.

Step 2: Constructing FDBN by means of FRBMs

According to the training procedure of FRBM in step 1, the weight values of FRBMs are connected from bottom to top to construct FDBN. In this FDBN, the connection between the top two layers is undirected and the other connection is directed from top to bottom [22].

Step 3: Constructing the FDNN that is used for advertising CTR prediction by means of FDBN

After an FLR model is added onto the top of the FDBN, the probability that the user clicks this advertisement is equal to the output value of the nodes of the input layer. At this point, the network becomes the initially trained fuzzy deep neural network (FDNN).

Step 4: Fine-tuning the parameters of FDNN

After FDNN is initially trained layer by layer, a fine-tuning process needs to be conducted. In contrast to the unsupervised learning approach in the initial training, the fine-tuning is implemented using supervised learning. Based on the weight values of the network that were obtained from the initial training, error backpropagation [27, 29] and stochastic gradient descent are used to update the weight values and bias of the neural network. After fine-tuning, the process of training FDNN is complete. The output of FDNN is in the form of a probability: $p(click|\boldsymbol{a},\boldsymbol{u},\boldsymbol{c}) = p(y_d = 1|\boldsymbol{x}'_d;\boldsymbol{\theta}')$.

The process of fine-tuning the parameters of FDNN contains two phases.

1) Forward calculation of the input signal:

When labeled data $(\boldsymbol{x}_d, y_d)$ of click logs are given, that is, the combined feature vector $\boldsymbol{x} = (\boldsymbol{a}, \boldsymbol{u}, \boldsymbol{c}) = (x_1, x_2, \ldots, x_n)$ is given, the jth node in the lth layer ($l = 2, .., L_n$) of the FDNN deep neural network can be defined as follows:

L_n denotes the total number of layers in the FDNN network;

s_l denotes the number of nodes in the lth layer of the FDNN network;

$\overline{w^l_{ji}}$ denotes the connection weight between the jth node in the lth layer and the ith node in the $l-1$th layer; $\overline{b^l_j}$ denotes the bias of the jth node in the lth layer; $\overline{net^l_j} = \overline{b^l_j} + \sum_{i=1}^{s_{l-1}} \overline{w^l_{ji}}\,\overline{o^{l-1}_i}$ denotes the weighted-sum input of the jth node in the lth layer;

$\overline{o^l_j}$ denotes the output value of the activation function of the jth node in the lth layer;

The selected activation function is sigmoid function.

In the forward calculation, the input signal is transmitted from the input layer to the output layer. The input value of the jth node in the second layer of FDNN ($l = 2$) can be described as $\overline{net^2_j} = \overline{b^2_j} + \sum_{i=1}^{s_1} \overline{w^2_{ji}} x_i$. When the input of node j is transferred by the activation function, its output value can be described as $\overline{o^2_j} = g(\overline{net^2_j})$. Therefore, the input value of the jth node in the lth layer of FDNN ($l = 3, \ldots, L_n$) can be described as $\overline{net^l_j} = \overline{b^l_j} + \sum_{i=1}^{s_{l-1}} \overline{w^l_{ji}} o_i$ and its corresponding output value can be described as $\overline{o^l_j} = g(\overline{net^l_j})$.

In particular, the output value of the jth node in the th layer of FDNN can be described as

$$\overline{h}_{\overline{\theta}}(\boldsymbol{x}_d) = O_j = O_1 = \overline{O^{L_n}_1}. \tag{30}$$

2) Backpropagation of the error signal:

This paper regards the cross-entropy as the objective function [25]. It can be described as formula (31). The local gradient will be refined using backpropagation. The error signal between the output value and labeled signal will be transmitted from the output layer to the input layer in the process.

$$\begin{aligned}\overline{J}(\boldsymbol{x}_d;\overline{\theta}) &= y_d log\overline{h}_{\overline{\theta}}(\boldsymbol{x}_d) + (1-y_d)log(1-\overline{h}_{\overline{\theta}}(\boldsymbol{x}_d)) \\ &= y_d logg(\overline{net_j^{L_n}}) + (1-y_d)log(1-g(\overline{net_j^{L_n}})),\end{aligned} \tag{31}$$

In this formula, y_d is the label value of the sample, which can be represented as $y_d \in [0, 1]$.

The gradient descent method will be adopted to learn the model. This paper will first define the residual error $\overline{\delta_j^l}$ [28] of the jth node in the lth layerlayer as the partialand of objective function $\overline{J}$ with respect to the weighted sum $\overline{net_j^l}$ of node j. The residual error $\overline{\delta_j^{L_n}}$ of the j th neuron node of the L_n th layer can be calculated by formula (32)

$$\begin{aligned}\overline{\delta_j^{L_n}} &= \frac{\partial \overline{J}}{\partial \overline{net_j^{L_n}}} = y_d \frac{1}{g(\overline{net_j^{L_n}})} g'(\overline{net_j^{L_n}}) + (1-y_d)\frac{-1}{1-g(\overline{net_j^{L_n}})} g'(\overline{net_j^{L_n}}) \\ &= y_d - g(\overline{net_j^{L_n}}) \\ &= y_d \overline{O_1^{L_n}}.\end{aligned} \tag{32}$$

The residual error δ_j^l of the jth neuron node of the lth layer ($l = 2, \ldots, L_n - 1$) can be calculated by formula (33) according to the chain rule [29].

$$\begin{aligned}\overline{\delta_j^l} &= \frac{\partial \overline{J}}{\partial \overline{net_j^l}} = \frac{\partial \overline{J}}{\partial \overline{o_j^l}} \cdot \frac{\partial \overline{o_j^l}}{\partial \overline{net_j^l}} \\ &= \sum_{i=1}^{s_{l+1}} \left(\frac{\partial \overline{J}}{\partial \overline{net_i^{l+1}}} \frac{\partial \overline{net_i^{l+1}}}{\partial \overline{o_j^l}} \right) \cdot \frac{\partial \overline{o_j^l}}{\partial \overline{net_j^l}} \\ &= \left(\sum_{i=1}^{s_{l+1}} \overline{\delta_i^{l+1}} \cdot \overline{w_{ji}^{l+1}} \right) \frac{\partial g(\overline{net_j^l})}{\partial \overline{net_j^l}} \\ &= \left(\sum_{i=1}^{s_{l+1}} \overline{\delta_i^{l+1}} \cdot \overline{w_{ji}^{l+1}} \right) g(\overline{net_j^l})(1-g(\overline{net_j^l})) \\ &= \left(\sum_{i=1}^{s_{l+1}} \overline{\delta_i^{l+1}} \cdot \overline{w_{ji}^{l+1}} \right) \overline{o_j^l}(1-\overline{o_j^l}),\end{aligned} \tag{33}$$

where $\overline{w_{ji}^{l+1}}$ denotes the connection weight between the jth node of the lth layer and the ith node of $l+1$th layer.

According to formulas (2) and (10), the gradient of cost function $\overline{J}(\overline{W}, \overline{b})$ with respect to parameters $\overline{w_{ji}^l}$ and $\overline{b_j^l}$ can be calculated by formulas (34), (35), (36) and (37).

$$\frac{\partial \overline{J}}{\partial w_{jiL}^l} = \frac{\partial \overline{J}}{\partial \overline{net_j^l}} \cdot \frac{\partial \overline{net_j^l}}{\partial w_{jiL}^l} = \overline{\delta^l}_{jL} \cdot \overline{o_i^{l-1}}_L, \tag{34}$$

$$\frac{\partial \overline{J}}{\partial b_{jL}^{l}} = \overline{\delta}_{jL}^{l}, \tag{35}$$

$$\frac{\partial \overline{J}}{\partial w_{jiR}^{l}} = \frac{\partial \overline{J}}{\partial \overline{net_{j}^{l}}} \cdot \frac{\partial \overline{net_{j}^{l}}}{\partial w_{jiR}^{l}} = \overline{\delta}_{jR}^{l} \cdot \overline{o_{i}^{l-1}}_{R}, \tag{36}$$

$$\frac{\partial \overline{J}}{\partial b_{jR}^{l}} = \overline{\delta}_{jR}^{l}. \tag{37}$$

The above process is described in Algorithm 4.

Algorithm 4 The algorithm for fine-tuning the parameters of FDNN using BP

```
Input: ($\boldsymbol{x}_d$, $y_d$) is the labeled training data of textual advertising
click logs;
    $\varepsilon$ is the learning rate;
    $w_{jiL}^{l}$ is the lower bound of the connection weight between the $j$th node
in the $l$th layer and the $i$th node in the ($l$ - 1)th layer of FDNN;
    $b_{jL}^{l}$ is the lower bound of the bias of the $j$th node in the $l$th layer
of FDNN;
    $w_{jiR}^{l}$ is the upper bound of the connection weight between the $j$th node
in the $l$th layer and the $i$th node in the ($l$ - 1)th layer of FDNN;
    $b_{jR}^{l}$ is the upper bound of the bias of the $j$th node in the $l$th layer
of FDNN;
Output: The latest parameters: $w_{jiL}^{l}$, $w_{jiR}^{l}$, $b_{jL}^{l}$, $b_{jR}^{l}$
  BEGIN
  1. In the forward-propagation calculation, when the input data vector
$\boldsymbol{x}_d$ is fed into the input layer, the output value of FDNN can be obtained
using formula (30);
  2. In the calculation of error backpropagation,
     obtain $\frac{\partial \overline{J}}{\partial w_{jiL}^{l}}$ by means of formula (34);
     obtain $\frac{\partial \overline{J}}{\partial b_{jL}^{l}}$ by means of formula (35);
     obtain $\frac{\partial \overline{J}}{\partial w_{jiR}^{l}}$ by means of formula (36);
     obtain $\frac{\partial \overline{J}}{\partial b_{jR}^{l}}$ by means of formula (37);
  3. The parameters can be updated according to the following rules:
     $w_{jiL}^{l} = w_{jiL}^{l} + \boldsymbol{\varepsilon} \frac{\partial \overline{J}}{\partial w_{jiL}^{l}}$;
     $b_{jL}^{l} = b_{jL}^{l} + \boldsymbol{\varepsilon} \frac{\partial \overline{J}}{\partial b_{jL}^{l}}$;
     $w_{jiR}^{l} = w_{jiR}^{l} + \boldsymbol{\varepsilon} \frac{\partial \overline{J}}{\partial w_{jiR}^{l}}$;
     $b_{jR}^{l} = b_{jR}^{l} + \boldsymbol{\varepsilon} \frac{\partial \overline{J}}{\partial b_{jR}^{l}}$;
  4. return $w_{jiL}^{l}$, $b_{jL}^{l}$, $w_{jiR}^{l}$, $b_{jR}^{l}$.
  END
```

Experiment datasets and environment

Description of dataset. This paper uses the Criteo Display Advertising Challenge dataset, which is provided by the Criteo company on the famous machine learning website Kaggle for advertising CTR competition [30]. Every advertisement log record contains the set of features of the displayed advertisement, which is composed of 13 numerical features and 26 categorical features. The detailed formats of these features are given in S1 Table. The class Label is used to indicate whether this advertisement is clicked: if the advertisement is clicked, the value of Label is 1; otherwise, the value of Label is 0. Categorical features are desensitized using Harsh mapping; therefore, real meaning can not be obtained from them.

To adequately evaluate the performance of this model and prevent the interference of many factors such as data timeliness, we randomly disrupt the order of the raw samples in the experiment. Meanwhile, to avoid the over-fitting, the 5-fold cross validation method [31] is adopted in the training process of all experiments.

Experiment environment. Six high-performance workstations are used for the experiments. The hardware and software configurations of the experimental environment are shown as S2 Table.

Results and analysis

Baseline models

This paper selects the logistic regression (LR), Factorization Machines (FMs) [32], GBDT+FM [8], and DBNLR [11] models as the baseline models for performance comparison.

Performance evaluation metric of models

AUC. This paper uses the area under the receiver operating characteristic curve (AUC) [33, 34] as the main performance evaluation metric for advertising CTR prediction. The confusion matrix is shown as S3 Table [35, 36]. Formulas $TPR = TP/(TP + FN)$ and $FPR = FP/(FP + TN)$ are used to calculate the corresponding TPR and FPR values of the confusion matrix to obtain a co-ordinate pair (dot pair). The dot pairs make up the receiver operating characteristic (ROC) curve [37]. The AUC is the total area under the ROC. The larger the AUC is, the more accurate the advertising CTR prediction is, and the better the effect achieved by the advertising placement is.

Log-loss. This paper selects the logarithmic loss function 'log-loss' [38, 39] as another auxiliary performance evaluation metric for advertising CTR prediction.

$$logloss = -\frac{1}{L}\sum_{i=1}^{L} y_i log\overline{y}_i + (1 - y_i)log(1 - \overline{y}_i), \tag{38}$$

where L is the number of samples, y_i is the true label (0 or 1), and $\overline{y}_i$ is the predicted probability. The value of log-loss reflects the similarity between the CTR predicted by model and real CTR. The smaller the value of log-loss is, the more accurate the advertising CTR prediction is.

Experiments design and results analysis

Configuration optimization and performance analysis of FDNN and DBNLR. *A. Feature Pre-processing*

FDNN is similar with DBNLR in the architecture. Their difference lies in the components of model. Therefore, this paper selects the DBNLR as a baseline model.

In the experiment on the FDNN and DBNLR models, the numerical features of the raw dataset are directly adopted, and these categorical features are expanded to binary vectors using one-hot representation.

Although the method of one-hot representation is straightforward and effective, it makes the feature vectors sparse and high-dimensional. In the deep learning models, when the dimensionality of the features is large, the scale of the parameters of the model becomes large, which leads to a sharp increase in training time. Moreover, the hardware will be a bottleneck in the training process. Therefore, this paper adopts a feature hashing strategy (signed hashing trick) [40] to reduce the dimensionality of these sparse binary feature vectors. Finally, the joint vector that is composed of numerical features and categorical features and has been processed by the feature hashing strategy is regarded as the input of the two deep architecture models.

Input vectors can be obtained when the data of the Display Advertising Challenge dataset are preprocessed using the feature preprocessing methods described above. In this experiment, after preprocessing, the dimensionality of the input vector is 1746088. Therefore, the number of nodes in the visible layer (input layer) is 1746088.

B. The Parameter Settings for Training FDNN and DBNLR

These two models are developed based on the Theano framework [41] by means of Python. In all experiments, the maximum number of learning epochs in the training process is set to a sufficiently large value (800) and the Early-Stopping strategy is adopted to automatically cut epochs for obtaining the best validation score. Furthermore, this strategy can also avoid over-fitting.

The steps of the Early-Stopping strategy are as follows:

1. Split the dataset into a training set and a validation set;

2. When the training of every epoch has been finished, calculate the validation error (loss) in the validation dataset;

3. When the validation error of the validation dataset no longer decreases, record the number of epochs and the current validation error value. If the validation error of the validation dataset does not increase after continuation, the training process is terminated, the best validation score is obtained, and optimization is complete. Meanwhile, the optimal parameter estimation values are obtained.

In the initial training process of FDBN and DBNLR, the gradient descent method with minibatch is used for training on these samples and each minibatch includes 2000 training samples. For FGBRBM (or GBRBM), the learning rate is 0.01, while for FRBM (or RBM) with the binary variable, the learning rate is 0.1. The decay factor of the weight value is 0.0002.

In the fine-tuning process of FDNN and DBNLR, the gradient descent method with minibatch is used for training on these samples and each minibatch includes 4000 training samples. The decay factor of the weight value is 0 and the learning rate is set as 0.005.

Dropout represents the probability that a node is retained in the neural network [42, 43]. A reasonable sparse dropout strategy can strengthen the robustness of a neural network. Dropout is utilized to reduce over-fitting and the complexity of these two deep learning models. The paper sets the dropout rate to 0.5, 0.6, 0.7, 0.8, and 0.9 to implement the testing experiments. Under the condition of having the same configuration structure, the dropout rates are respectively 0.5 and 0.6 when FDNN and DBNLR use their own optimal settings.

C. Optimization of the number of hidden layers and hidden nodes

Currently, there is no method for quickly setting the number of hidden layers and the number of hidden nodes in each layer. Therefore, many experiments and much experience are utilized to search for the optimal structure. According to past experience, when the number of nodes of the visible layer is large, the nodes of the hidden layer need to perform "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 need to perform "dimensionality addition—dimensionality reduction". In general, the number of hidden nodes will be increased or decreased with multiple. By comparing the performances in many experiments with different configurations (different numbers of nodes of each hidden layer and numbers of hidden layers), the configuration with relatively optimal performance can be obtained.

In S4 Table, information on different configurations in many experiments is given. In S7 Fig, the corresponding AUC values of DBNLR and FDNN with different configurations are given. In S8 Fig, the corresponding log-loss values of DBNLR and FDNN with different configurations are given.

As presented in S7 and S8 Figs, in the experiments for training FDNN, in the beginning, increasing the number of hidden layers can improve the performance of the proposed method; however, when the number of hidden layers is more than 3, the performance is degraded. This is because FDNN does not adequately capture feature interactions when the number of hidden layers is small. However, when the number of hidden layers and the number of nodes in the hidden layers exceed a certain scale, cross-features disturb and degrade the accuracy of CTR prediction. The over-fitting phenomenon is observed and the algorithm spends more time learning the parameters.

After many experiments have been completed, a configuration with relatively optimal performance is obtained: Conf11, the number of hidden layers is 3; the number of nodes of the first hidden layer is 170; the number of nodes of the second hidden layer is 1700; and the number of nodes of the third hidden layer is 17. This paper chooses this configuration as the final structure of FDNN. Corresponding AUC and log-loss on the test dataset can be seen in S9 and S10 Figs.

In the experiments for training DBNLR, relatively optimal performance is obtained when the configuration structure is as follows: Conf7, the number of hidden layers is 3; the number of nodes of the first hidden layer is 150; the number of nodes of the second hidden layer is 1500; and the number of nodes of the third hidden layer is 15. This paper chooses this configuration as the final structure of DBNLR. Corresponding AUC and log-loss on the test dataset can be seen in S9 and S10 Figs.

We observe that the curve of the AUC value of the DBNLR model and that of the FDNN model are indented. However, when they have the same configuration, the overall level of performance of DBNLR is worse than that of FDNN; this is true for the log-loss index. It is because fuzzy set techniques are introduced in the proposed FDNN model. The components of FDNN model are fuzzy RBM, fuzzy GBRBM and fuzzy LR. With the data of many records in the Display Advertising Challenge dataset which contains outliers and missing values implying many uncertainties, the uncertainties can be taken into consideration and efficiently handled by fuzzy set techniques; therefore, the performance of the FDNN model for CTR prediction is better than that of DBNLR in the experiment.

Configuration optimization of other baseline models. *A. LR Solution*

CTR prediction in a real advertising system involves a large number of samples, while logistic regression (LR) model can appropriately address a large number of features and can be trained rapidly, so the LR model is selected as a baseline model in this paper. This paper selects LIBLINEAR [44] to implement L2-regularized logistic regression, and the stochastic gradient method (SG) is adopted to optimize the parameters of LR. In the LR solution, the numerical features are used directly and the categorical features are first represented with one-hot encoding and then used to establish the feature space with MurmurHash 3. After many experiments, this paper adopts the following parameters: $\eta = 0.2, \lambda = 0$.

B. FM Solution

For advertising CTR prediction, Factorization Machines (FMs) combine the advantages of Support Vector Machines with factorization models, and can characterize the basic feature interactions of a large number of advertising data, therefore, this paper selects the FM model as a baseline model [45]. For advertising CTR prediction, Factorization Machines (FMs) is also adopted, which is based on feature engineering and matrix design. This paper adopts LIBFM to implement factorization machines [46, 47] and the stochastic gradient method (SG) is adopted to optimize the parameters of FM. The numerical features are used directly and the categorical features are first represented with one-hot encoding. After many experiments, this paper adopts the following parameters: $\lambda = 40$, $k = 30$.

C. GBDT+FM Solution

Gradient Boost Decision Tree (GBDT) is a nonlinear model that is based on the concept of collective learning boosting. GBDT has strong extracting feature ability and nonlinear fitting ability, and it can flexibly handle various types of data in the advertising dataset including continuous value and discrete value, but no over-fitting occurs. Therefore, this paper selects GBDT+FM solution as a baseline model. Based on the work of Y Juan et al., this paper implements GBDT+FM solutions for performance comparison and analysis. First, in Preprocessing-A, all numerical features are used directly and all categorical features are represented with one-hot encoding. Then, these features are fed into decision trees in GBDT to generate GBDT features. This paper selects Xgboost to implement the GBDT [48]. In Preprocessing-A, the number of decision trees in GBDT is 30 and the depth is 7. Thirty features are generated for each impression. The following process is Preprocessing-B, which is used for generating input features for FM. In Preprocessing-B, numerical features (I1-I13) with values greater than 2 are transformed by $v \leftarrow \lfloor log(v)^2 \rfloor$; categorical features (C1-C26) that appear fewer than 10 times are transformed into a special value; and GBDT features are directly included. Therefore, each record contains 69 features: 13 numerical features, 26 categorical features and 30 GBDT features. Then, these three groups of features are hashed into 1 M dimensions using a hashing trick. Finally, the processed features are fed into the FM model for CTR prediction. The parameters of the FM model that achieve the relatively optimal results are "$\lambda = 40$, $k = 100$".

Experimental results and analysis. According to S9 and S10 Figs, the proposed FDNN method has the best performance and LR has the worst performance. In the process of training models, it can be found that logistic regression model can be trained rapidly, but its effect is in the general level. One reason is that when LR is used to predict CTR, it requires the experts with sophisticated experience to implement feature engineering through the human labor. The other reason is that the simple structure of LR restrains its representation ability. These two disadvantages cause that LR cannot learn complex feature interactions and is insensitive to outliers and missing values. Therefore, LR has the relatively bad performance in the experiment.

According to S9 and S10 Figs, FM is superior in performance to LR. The reason is that FM extends the idea of LR because FM adds to second order fitting on the basis of first order fitting and it can automatically learn cross features of any two features from different dimensions. And cross features are expressed with embedding vector. Compared with LR, FM model has fewer manual intervene but higher efficiency. Therefore, FM has better representation ability than LR in the experiment. However, FM also cannot learn much more complex feature interactions because of its relatively simple structure.

In this experiment, GBDT+FM solution outperform the FM model. The GBDT+FM solution obtains good prediction results, which are slightly inferior to those of the proposed method. The GBDT can learn useful high-order feature interactions because it has strong nonlinear fitting ability and the robustness for hyperparameter, so it can efficiently extracts discriminative features and cross-features of advertising dataset, and shows better performance than FM model in our CTR prediction experiment.

Compared with the GBDT+FM solution, the proposed method achieves more than 0.09% improvement in terms of AUC (0.01% in terms of log-loss) on the dataset. The reasons are as follows. The first is that the quality of data exerts a great influence on effect of model. Because the dataset contains many noisy data, there are bottlenecks in the effects of GBDT+FM solution. Compared with this model, the proposed FDNN model, as a result of being endowed with fuzzy technology, can capture more-complex high-order feature interactions of the advertising dataset, which contains many outliers and missing values. The second is that GBDT+FM solution, belonging to a kind of stacking combined model without backproportion, is not always better than the proposed model with backproportion in effect improvement for CTR prediction from the theoretical perspective. Therefore, it demonstrates relatively good performance in terms of both data representation capability and prediction accuracy. In a real production environment for an advertising system, a small improvement in advertising CTR is likely to result in a significant increase in financial income for the internet company and an improved user experience.

Discussion

This paper proposes a fuzzy deep neural network (FDNN) to address the problem of advertising CTR prediction. The network's performance is compared with those of several baseline models in real advertisement datasets. Due to the fuzziness that is introduced into the proposed model, it can learn the most useful high-order feature interactions and demonstrates good performance in terms of both data representation capability and robustness in advertisement datasets with noise.

Big data bring new perspectives to these fields and challenges in processing and discovering valuable information. The traditional artificial intelligence techniques can no longer effectively extract and organize the discriminative information from raw advertisement data directly [49]. This paper provides a basis for the further application of fuzzy deep neural networks in advertising CTR prediction. Future work will focus on exploring additional unsupervised feature learning and deep architecture models to improve the CTR prediction for internet advertising.

In addition, in big data era, a real production environment for advertising CTR prediction requires that the advertising system return the prediction results in milliseconds or even microseconds. Determining how to implement low-lag and high-efficiency impromptu Ad-Hoc analysis to predict and return results based on a big dataset is a great challenge for big data processing systems. Big data systems with stream computing, such as Spark Streaming, Storm, and Flink, can implement analysis and query in real time. However, because of the limited memory capacity and loss of raw historical data in system, Ad-Hoc analysis and query for a big dataset cannot be implemented. Therefore, exploring CTR prediction solutions based on streaming big data processing technology is another future direction of the work of this paper.

Supporting information

S1 Fig. Triangular fuzzy number.
(ZIP)

S2 Fig. Structure of fuzzy restricted boltzmann machine.
(ZIP)

S3 Fig. Structure of fuzzy gaussian-bernoulli restricted boltzmann machine.
(ZIP)

S4 Fig. Modeling click-through rate prediction by means of LR.
(ZIP)

S5 Fig. Architecture of the FDNN model and its process of predicting CTR.
(ZIP)

S6 Fig. The process of constructing FDNN.
(ZIP)

S7 Fig. AUC values of FDNN and DBNLR with different numbers of layers and nodes.
(ZIP)

S8 Fig. Log-loss values of FDNN and DBNLR with different numbers of layers and nodes.
(ZIP)

S9 Fig. Comparison of AUC among all models on the test dataset.
(ZIP)

S10 Fig. Comparison of log-loss among all models on the test dataset.
(ZIP)

S1 Table. Data format of dataset.
(ZIP)

S2 Table. Experimental environment.
(ZIP)

S3 Table. Confusion matrix.
(ZIP)

S4 Table. Configure information of FDNN and DBNLR with different numbers of layers and nodes.
(ZIP)

Acknowledgments

This work is supported by the Natural Science Research Project of Education Department of Guizhou Province of China (No.KY2017350), the Project Funds of the Digital Engineering Research Center for Local National Culture of Universities in Guizhou (No.KY2013227) and the Scientific Research Project of Qiannan Normal University for Nationalities (No. qnsy2017006). The authors are thankful for the research work in all referenced manuscripts.

Author Contributions

Formal analysis: Zilong Jiang, Shu Gao.

Funding acquisition: Mingjiang Li.

Investigation: Shu Gao, Mingjiang Li.

Methodology: Zilong Jiang.

Project administration: Mingjiang Li.

Software: Zilong Jiang.

Writing – original draft: Zilong Jiang.

Writing – review & editing: Zilong Jiang.

References

1. Sami Najafi-Asadolahi, Fridgeirsdottir Kristin. Cost-Per-Click Pricing for Display Advertising. Journal Manufacturing & Service Operations Management. 2014; 16(4):482–497. https://doi.org/10.1287/msom.2014.0491

2. Z Pan, E Chen, Q Liu, T Xu, H Ma, H Lin. Sparse Factorization Machines for Click-through Rate Prediction. IEEE 16th International Conference on Data Mining (ICDM 2016), Barcelona, Spain, 2016; pp.400–409.
3. Chapelle Olivier, Manavoglu Eren, Rómer Rosales. Simple and scalable response prediction for display advertising [J]. ACM Transactions on Intelligent Systems and Technology, 2014, 5(4):1–34. https://doi.org/10.1145/2532128
4. H. Brendan McMahan, Gary Holt, D. Sculley, Michael Young, Dietmar Ebner, Julian Grady, et al. Ad click prediction: a view from the trenches. Proceedings of the 19th ACM international conference on Knowledge discovery and data mining (ACM SIGKDD 2013), New York, USA, 2013; pp.1222–1230.
5. Anh-Phuong TA. Factorization Machines with Follow-The-Regularized-Leader for CTR prediction inDisplay Advertising. 2015 IEEE International Conference on Big Data (IEEE BigData 2015). Santa Clara, CA, USA, 2015; pp. 2889–2891.
6. Afroze Ibrahim Baqapuri, Ilya Trofimov. Using Neural Networks for Click Prediction of Sponsored Search. arxiv preprint arXiv; 1412.6601, 2014.
7. Kushal Dave, Vasudeva Varma. Predicting the Click-Through Rate for Rare/NewAds. Centre for Search and Information Extraction Lab International Institute of Information Technology Hyderabad, INDIA, April 2010, pp.1–18.
8. Yuchin Juan, Yong Zhuang, Wei-Sheng Chin, Chih-Jen Lin. Field-aware Factorization Machines for CTR Prediction. ACM Conference on Recommender Systems (ACM RecSys 2016), Boston, United States, 2016; pp. 43–50.
9. Y Zhang, H Dai, C Xu, J Feng, T Wang, J Bian, et al. Sequential Click Prediction for Sponsored Search with Recurrent Neural Networks. 28th AAAI Conference on Artificial Intelligence (AAAI 2014), Québec City, Québec, Canada, 2014; pp.1369–1375.
10. Yu Shimin. Prediction of ads'click through rate based on recurrent neural network. Hangzhou: Zhejiang sci-tech university, 2016; pp.12–15.
11. Jiang Z, Gao S and Dai W. Research on CTR Prediction for Contextual Advertising Based on Deep Architecture Model. Control Engineering and Applied Informatics. 2016; 18(1):11–19.
12. Junxuan Chen, Baigui Sun, Hao Li, Hongtao Lu, Xian-Sheng Hua. Deep CTR Prediction in Display Advertising. Proceedings of the 24th ACM on Multimedia Conference (ACM MM 2016), Amsterdam, Netherlands, 2016; pp.811–820.
13. Wang H, Xu Z, Pedrycz W. An overview on the roles of fuzzy set techniques in big data processing: Trends, challenges and opportunities. Knowledge-Based Systems, 2017, 118:15–30. https://doi.org/10.1016/j.knosys.2016.11.008
14. Xu Zeshui. A Method for Priorities of Triangular Fuzzy Number Complementary Judgement Matrices. Fuzzy Systems & Mathematics, 2002; 16(1):47–50.
15. Asghari-Larimi M. Upper and lower (alpha, beta)-intuitionistic fuzzy set. International Mathematical Forum, 2012(9–12); pp.531–538.
16. Hamrawi Hussam, Coupland Simon, John Robert. Type-2 Fuzzy Alpha-Cuts. IEEE Transactions on Fuzzy Systems, 2017; 25(3):682–692. https://doi.org/10.1109/TFUZZ.2016.2574914
17. Dutta Soma and ChakrabortyMihir K. Fuzzy relation and fuzzy function over fuzzy sets: A retrospective. Soft Computing, 2015; 19(1):99–112. https://doi.org/10.1007/s00500-014-1356-z
18. Klimt Gustav, Buckley James J. Fuzzy Probabilities: New Approach and Applications, NewYork: Physica-Verlag, 2003; 20 (3):363–372.
19. Runkler T.A., Glesner Manfred. DECADE—fast centroid approximation defuzzification for real time fuzzy control applications. DBLP, 1994:161–165.
20. Chen C.L. Philip, Zhang Chun-Yang, Chen Long, and Gan Min. Fuzzy Restricted Boltzmann Machine for the Enhancement of Deep Learning. IEEE Transactions on Fuzzy Systems, 2015; 23(6):2163–2173. https://doi.org/10.1109/TFUZZ.2015.2406889
21. Gelfand Alan E. Journal of the American Statistical Association, 2000, vol. 95, no. 452, pp.1300–1304. https://doi.org/10.1080/01621459.2000.10474335
22. Hinton Geoffrey E., Osindero Simon, Teh Yee-Whye. A fast learning algorithm for deep belief nets. Neural Computation, 2006; 18(7):1527–1554. https://doi.org/10.1162/neco.2006.18.7.1527 PMID: 16764513
23. Hyun-Chul Kim, Jong-Hwan Lee. Evaluation of weight sparsity regularizion schemes of deep neural networks applied to functional neuroimaging data. The 42nd IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2017), New Orleans, LA, USA, 2017; pp.6150–6154.

24. Osman Taher, Divigalpitiya Prasanna, Arima Takafumi. Driving factors of urban sprawl in Giza governorate of the Greater Cairo Metropolitan Region using a logistic regression model. International Journal of Urban Sciences, 2016; 58:1–20.

25. Shrestha Ravi, Mohammed Shahed Khan, Hasan Md. Mehedi, Wahid Khabou. Automated Adaptive Brightness in Wireless Capsule Endoscopy Using Image Segmentation and Sigmoid Function. IEEE Transactions on Biomedical Circuits & Systems. 2016; 10(4):884. https://doi.org/10.1109/TBCAS.2016.2546838

26. Yann LeCun, Bengio Yoshua, and Hinton Geoffrey. Deep learning, Nature, 2015; 521(7553): 436–444. https://doi.org/10.1038/nature14539

27. Horikawa SI, Furuhashi T, Uchikawa Y. On fuzzy modeling using fuzzy neural networks with the back-propagation algorithm. IEEE Transactions on Neural Networks, 1992; 3(5):801–806. https://doi.org/10.1109/72.159069

28. Mosca A, Magoulas GD. Advances in Computational Intelligence Systems. New York: Springer Publishing Company, 2017; pp.433–445.

29. Dreyfus SE. Artificial neural networks, back propagation, and the Kelley-Bryson gradient procedure. Journal of Guidance Control & Dynamics, 2012; 13(5):926–928.

30. Kaggle Display Advertising Challenge Dataset.[EB/OL]. http://labs.criteo.com/2014/02/kaggle-display-advertising-challenge-dataset/

31. Triba MN, Le ML, Amathieu R, Goossens C, Bouchemal N. PLS/OPLS models in metabolomics: the impact of permutation of dataset rows on the K-fold cross-validation quality parameters. Molecular Biosystems, 2015; 11(1):13. https://doi.org/10.1039/C4MB00414K

32. Chen C, Hou C, Xiao J, Yuan X. Purchase Behavior Prediction in E-Commerce with Factorization Machines. IEICE Transactions on Information and Systems, 2016, E99.D(1), pp.270–274. https://doi.org/10.1587/transinf.2015EDL8116

33. Akimova Tatiana, Levine Matthew H., Beier Ulf H., Hancock Wayne W. Standardization, Evaluation, and Area-Under-Curve Analysis of Human and Murine Treg Suppressive Function. Suppression and Regulation of Immune Responses. 2016; vol.1371, pp.43–78. https://doi.org/10.1007/978-1-4939-3139-2_4

34. Alberto Jiménez-Valverde. Insights into the area under the receiver operating characteristic curve (AUC) as a discrimination measure in species distribution modeling. Global Ecology and Biogeography. 2012; 21(4):498–507. https://doi.org/10.1111/j.1466-8238.2011.00683.x

35. Deng X, Liu Q, Deng Y, Mahadevan S. An improved method to construct basic probability assignment based on the confusion matrix for classification problem, Information Sciences, 2016; 340: 250–261.

36. Ohsaki M, Wang P, Matsuda K, Katagiri S, Watanabe H, Ralescu A. Confusion-Matrix-Based Kernel Logistic Regression for Imbalanced Data Classification. IEEE Transactions on Knowledge & Data Engineering, 2017; 29(9):1806–1819. https://doi.org/10.1109/TKDE.2017.2682249

37. Talabani A. Jamal, Endreseth B.H., Lydersen S., Edna T.-H. 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. 2017; 32(1):41–47. https://doi.org/10.1007/s00384-016-2644-0

38. sklearn.metrics.log_loss.[EB/OL]. http://scikit-learn.org/stable/modules/generated/sklearn.metrics.log_loss.html.

39. Logarithmic Loss https://www.Kaggle.com/wiki/LogarithmicLoss.

40. sklearn.feature_extraction. FeatureHasher.[EB/OL]. http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.FeatureHasher.html.

41. Theano.[EB/OL]. http://deeplearning.net/software/theano/.

42. Gao W, Zhou Z. Dropout Rademacher complexity of deep neural networks. Science China Information Sciences, 2016; 59(7):072104. https://doi.org/10.1007/s11432-015-5470-z

43. Z Li, B Gong, T Yang. Improved Dropout for Shallow and Deep Learning. The 30th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain, 2016; pp. 1–8.

44. liblinear.[EB/OL]. http://www.csie.ntu.edu.tw/cjlin/liblinear/.

45. R.J. Oentaryo, E.P. Lim, J.W. Low, D. Lo, M. Finegold. Predicting response in mobile advertising with hierarchical importance-aware factorization machine. The Seventh ACM International Conference on Web Search & Data Mining (ACM WSDM 2014), New York, USA, 2014; pp. 123–132.

46. Rendle Steffen. Factorization machines with libfm. ACM Transactions on Intelligent Systems and Technology, 2012; 3(3): Article 57(1–22). https://doi.org/10.1145/2168752.2168771

47. libfm.[EB/OL]. http://www.libfm.org/.

48. Xgboost. Scalable, Portable and Distributed Gradient Boosting (GBDT, GBRT or GBM) Library, for Python, R, Java, Scala, C++ and more.[EB/OL]. https://github.com/dmlc/xgboost.

49. Lei Y, Jia F, Lin J, Xing S, and Ding SX. An Intelligent Fault Diagnosis Method Using Unsupervised Feature Learning Towards Mechanical Big Data. IEEE Transactions on Industrial Electronics, 2016; 65 (5):3137–3147.