A Novel Integrated Network with LightGBM for Click-Through Rate Prediction

Zhen Xia (isugar3412@gmail.com)
Southwest Petroleum University https://orcid.org/0000-0001-7852-8380
Senlin Mao
Southwest Petroleum University
Jing Bai
Southwest Petroleum University
Xinyu Geng
Southwest Petroleum University https://orcid.org/0000-0002-1101-0058
Liu Yi
Southwest Petroleum University

Research Article

Keywords: LightGBM, Click-through Rate, Integrated Network, Deep & Cross, Product-based Neural Network

Posted Date: October 13th, 2021

DOI: https://doi.org/10.21203/rs.3.rs-872310/v1

Noname manuscript No.
(will be inserted by the editor)

A Novel Integrated Network with LightGBM for Click-through Rate Prediction

Zhen Xia · Senlin Mao · Jing Bai · Xinyu Geng* · Liu Yi

Received: date / Accepted: date

Abstract Click-through Rate (CTR) prediction has become one of the core tasks of the recommendation system and its online advertising with the development of e-commerce. In the CTR prediction field, different features extraction schemes are used to mine the user click behavior to achieve the maximum CTR, which helps the advertisers maximize their profits. At present, achievements have been made in CTR prediction based on Deep Neural Network (DNN), but insufficiently, DNN can only learn high-order features combination. In this paper, Product & Cross supported Stacking Network with LightGBM (PCSNL) is proposed for CTR prediction to solve such problems. Firstly, the L_1 and L_2 regularizations are imposed on Light Gradient Boosting Machine (LightGBM) to prevent overfitting. Secondly, the method of vector-wise feature interactions is added to product layer in product network to learn second-order feature combinations. Lastly, feature information is fully learned through the cross network, product network and stacking network in PCSNL. The online ads CTR prediction datasets released by Huawei and Avazu on the Kaggle platform are involved for experiments. It is shown that the PCSN model and PCSNL have better performance than the traditional CTR prediction models and deep learning models.

Keywords LightGBM · Click-through Rate · Integrated Network · Deep & Cross · Product-based Neural Network

Zhen Xia
isugar3412@gmail.com

Senlin Mao
1285112478@qq.com

Jing Bai
1255258033@qq.com

Xinyu Geng*
gengxy123@126.com

Liu Yi
1050594538@qq.com

Zhen Xia · Senlin Mao · Jing Bai · Xinyu Geng · Liu Yi
School of Computer Science, Southwest Petroleum University, Chengdu, 610500, China

1 Introduction

The Internet and cloud computing make it possible for advertisers to use Internet platforms for precision marketing. Compared with traditional advertising, online advertising boasts the advantages of wide coverage, high flexibility, strong pertinence, and low cost. One of the main goals of online advertising is to maximize the revenue of advertisers with a given budget, such as maximizing clicks or conversions of advertisements,which is defined as CTR prediction problem (Chapelle et al. 2015; Richardson et al. 2007).

There are mainly four billing methods for online advertising: monthly, Cost Per Mille (CPM), Cost Per Click (CPC) and Cost Per Sales (CPS) (Asdemir et al. 2012; Miralles-Pechuán et al. 2017). CPC and CPS are closely related to CTR, for that the order of CTR determines the income of the enterprise. Therefore, advertisers need a high CTR on the advertising platform to help increase the revenue from advertising. The advertising platform utilizes its massive user and consumer resources to achieve precision advertising by tailoring advertisements to different users and contexts. A small increase in CTR can bring greater benefits to the enterprise. Therefore, CTR prediction is becoming more and more important in the recommendation systems (Wen 2021; Vedavathi and Kumar 2021) and online advertising (Koren et al. 2009).

In order to facilitate readers' understanding of this paper, a brief introduction of structure of the paper is presented as follows. Section 2 introduces some related works about CTR prediction model. Section 3 provides some preliminary knowledge for understanding CTR prediction model based on deep learning. In Section 4, details of the integrated network of this paper are explained. Section 5 presents and analyzes the experimental results. Finally, a conclusion is made with potential future works mentioned in Sec 6.

2 Related work

The CTR prediction problem is a typical regression problem. At present, the most commonly used prediction method is Logistic Regression (LR) (Richardson et al. 2007). LR features a simple model, but the linear model can not learn high-order feature combination. Chang et al. (2010) proposed the Poly2 model (Kudo and Matsumoto 2003) in 2010 for CTR prediction. The Poly2 model retains the characteristics of LR learning first-order features, and learns the pairwise combination of second-order features. Due to its high data dimensionality and sparsity, the Poly2 model creates an extra complexity of time and space when learning second-order feature combinations. For the reason, Steffen (2010, 2012) proposed the Factorization Machine (FM) model in 2010. When FM does pairwise feature combinations, the high-dimensional sparse matrix is mapped to the low-dimensional dense matrix. There are also many feature engineering researchers who expand FM to Higher-order Feature Machines (HoFM) (Blondel et al. 2016), Attentional Factorization Machines (AFM) (Xiao et al. 2017) and High-order Attentive Factorization Machine (HoAFM) (Tao et al. 2020). However, FM only engages the combination between two features. Learning feature combinations by FM has different results, because the latent vectors in feature domains may be distributed in different ways. Therefore, Juan et al. (2016) further proposed Field-aware Factorization Machine (FFM) related to feature domains on the basis of FM. The basic concept of FFM is dividing the features into multiple feature domains for feature combinations, and combining the different feature domains of the two features to learn different latent vectors. In addition, researchers of Facebook (He et al. 2014) used Gradient Boost Decision Tree (GBDT) (Friedman 2001) to extract and filter distinguished features and feature combinations, and combined the extracted features and the original ones into a whole as the input to the LR model to solve the problem of feature combinations. This solution is called GBDT+LR.

In recent years, deep learning (Schmidhuber 2015; Yu and Deng 2011; Sulthana et al. 2020) has become successful in the fields of Computer Vision (Chu et al. 2020), Natural Language Processing, Language Recognition(Mohamed et al. 2012a,b) and Cyberspace Security (Yang et al. 2019; Yang and Liu 2020) because of its great power of feature representation learning. The strong learning ability of deep learning is also applied to CTR prediction. Zhang et al. (2016) proposed the Factorization Machine supported Neural Network (FNN) model in 2016. The input of FNN is the dense latent vector obtained by the pre-trained FM model. Qu et al. (2016) proposed the Product-based Neural Network (PNN) model with the product layer integrated into the DNN model, which contains latent vectors, and includes the product operation between feature vectors (inner product and outer product). Models that merely use deep learning can only learn the high-order feature combinations, but the low-order ones are equally important in CTR prediction. In 2016, Cheng et al. (2016) proposed the Wide&Deep model by combining the LR linear model and the deep learning model. Wide&Deep model not only covers low-order feature combinations in wide part (LR model), but also learns high-order feature interactions in deep part (Neural Network). In order to make up for the defect that the Wide&Deep model can only learn first-order features in the linear model part, Guo et al. (2017) proposed to replace the linear part (Wide) of the Wide&Deep model with an FM model, creating a new model called DeepFM.

In addition to the above-mentioned models combining two or more models, Multi-Layer Perceptron (MLP) is used in many studies to directly learn the feature interactions. Shan et al. (2016) proposed Deep Crossing model for CTR prediction. The Deep Crossing model is an MLP composed of an embedding layer, a Stacking layer, a Multiple Residual Unit and a scoring layer. Wang et al. (2017) proposed the Deep&Cross model for CTR prediction. The Deep&Cross model consists of DNN, cross network and stacking network, and the input of stacking network is the combination of the output of first two networks. Zhu et al. (2017) proposed the Deep Embedding Forest (DEF) model. The forest layer in The DEF model is used to learn high-order interactions. However, the same feature information is learned by the multiple residual unit in Deep Crossing. The DEF model can effectively reduce the online prediction time compared with the Deep Crossing model. Lian et al. (2018) proposed eXtreme Deep Factorization Machine (xDeepFM), the Compressed Interaction Network (CIN) in xDeepFM as an explicit network of learning feature interactions.

Some traditional CTR prediction models can only learn low-order feature combinations. Models based on deep learning can learn both low-order and high-order feature interactions, but the feature information learned by most models is not sufficient. Hence, here proposes an integrated network with LightGBM for CTR prediction.

3 Preliminaries

3.1 LightGBM

Compared with eXtreme Gradient Boosting (XGBoost) (Chen and Guestrin 2016), Light Gradient Boosting Machine (LightGBM) (Ke et al. 2017) improves the calculation speed of the algorithm while ensuring smaller memory space and better algorithm performance. First, LightGBM uses Histogram-based Algorithm (HA). Continuous feature in the data is discretized by HA, and represented as a histogram of width k, as is shown in Fig. 1. LightGBM gains many advantages by discretizing continuous feature values, including: reduced memory, convenient storage, faster calculation, etc.

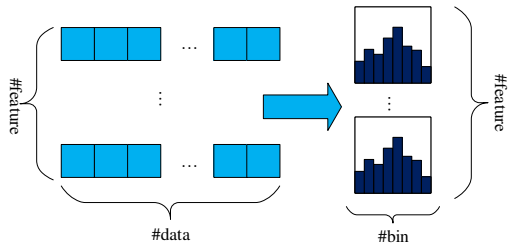


Fig. 1 Histogram-based Algorithm

Another important factor affecting the efficiency of the algorithm is the information gain calculation method of split nodes. Therefore, the leaf-wise tree growth strategy with a maximum depth limit and Gradient-based One-side Sampling (GOSS) (Ke et al. 2017) are used to split nodes.

Compared with other strategies, the leaf-wise one can require fewer split nodes with the same of algorithm performance and error. The weaknesses of the leaf-wise is that it is tendency to create a deeper decision tree, which can lead to overfitting. LightGBM use the leaf-wise tree growth strategy with a maximum depth limit to prevent overfitting while ensuring high efficiency.

GOSS is a sampling algorithm when calculating sample information gain. The main goal is to reduce the impacts of small gradient instances on split points. In order to reduce the complexity of calculating the information gain, GOSS divides the samples into two categories: large gradient ones and small gradient ones. Samples with a large gradient take up a larger proportion, so reducing most of the samples with a small gradient have no great impact on the calculation of the information gain. The samples are sorted in a descending order of the absolute value of their gradients. GOSS firstly selects the first $\alpha*100\%$ instances as large gradient data, where α is the sampling ratio of large gradient data. Then it randomly selects $\beta*100\%$ instances in remaining data as small gradient data, where β is the sampling ratio of large gradient data. Finally, it uses $(\alpha+\beta)\times 100\%$ instances to calculate the information gain as Eq. (1) (Ke et al. 2017).

$$\begin{aligned}\tilde{V}(d) =& \frac{1}{n}\Big(\frac{(\sum_{x_i\in A_l} g_i + \frac{1-\alpha}{\beta}\sum_{x_i\in B_l} g_i)^2}{n_l(d)} \\ &+ \frac{(\sum_{x_i\in A_r} g_i + \frac{1-\alpha}{\beta}\sum_{x_i\in B_r} g_i)^2}{n_r(d)}\Big),\end{aligned} \tag{1}$$

where n is the number of $(\alpha+\beta)*100\%$ instances. $n_l(d)$ and $n_r(d)$ are respectively the number of samples retained by the left node and the right node. A_l and A_r are respectively the large gradient sample sets where the two child nodes (the left one and the right one) are retained. B_l and B_r are respectively the small gradient sample set where the two child nodes are retained. $\frac{1-\alpha}{\beta}$ is used to modify the deviation of the sum of gradients of A^c samples.

Exclusive Feature Bundling (EFB) in LightGBM, a feature dimensionality reduction algorithm, reduces the impacts of high-dimensional data on algorithm performance. The industrial data is high-dimensional sparse data. EFB bundles mutually exclusive into one feature to achieve dimensionality reduction. Finally, the complexity of the algorithm can be reduced from $O(N*D_{feature})$ to $O(N*D_{bundle})$, where D_{bundle} is the number of bundled features.

3.2 Embedding layer

In CTR prediction, most of the features are categorical features, such as "country=China". For most models, the categorical features can not be directly processed and need to be converted into a binary matrix by one-hot (Covington et al. 2016). For example, $[0,1,0,0,0]$ indicates that the feature only has five values, the second of which is China. All categorical features are processed into a high-dimensional sparse binary matrix by one-hot. When it is directly used as the input of a model, it generates exceptionally large computational overhead. Adding the embedding layer before inputting the data into a model can effectively reduce the complexity of the overall model. The Eq.(2) shows the con-

version method for a categorical feature (Wang et al. 2017).

$$\boldsymbol{x}_{\mathrm{e},i} = \boldsymbol{W}_{\mathrm{e},i}\boldsymbol{x}_i, \tag{2}$$

where $\boldsymbol{x}_{\mathrm{e},i}$ is the embedding matrix from i-th sparse feature, $\boldsymbol{x}_i$ is the one-hot matrix of i-th input feature, $\boldsymbol{W}_{\mathrm{e},i} \in \mathbb{R}^{n_r \times n_c}$ is the weight for handling one-hot vectors, n_r and n_c are respectively the dimensions of the embedding matrix and one-hot matrix of features. In the end, we combine the embedding matrices and continuous features $\boldsymbol{x}_{\mathrm{d}}$ in data into one matrix $\boldsymbol{X}_0$ as the input of model (Covington et al. 2016).

$$\boldsymbol{X}_0 = [\boldsymbol{x}_{\mathrm{e},1}^T, \boldsymbol{x}_{\mathrm{e},2}^T, \cdots, \boldsymbol{x}_{\mathrm{e},k}^T, \boldsymbol{x}_{\mathrm{d}}] \tag{3}$$

3.3 Cross Network

Most traditional CTR prediction models do feature engineering before data are inputted. The role of cross network is to reduce the workload of the personnel performing the feature engineering. The iteration method of each layer of cross network is:

$$\boldsymbol{x}_{l+1} = \boldsymbol{x}_0\boldsymbol{x}_l^T\boldsymbol{w}_l + \boldsymbol{b}_l + \boldsymbol{x}_l = f(\boldsymbol{x}_l, \boldsymbol{w}_l, \boldsymbol{b}_l) + \boldsymbol{x}_l, \tag{4}$$

where $\boldsymbol{x}_l, \boldsymbol{x}_{l+1} \in \mathbb{R}^d$ are the outputs of the l-th and the $(l+1)$-th cross layer, and $\boldsymbol{w}_l, \boldsymbol{b}_l$ are the weight and bias parameters of the l-th layer (Wang et al. 2017). The iteration of l-th layer in cross network is presented in Fig. 2 (Wang et al. 2017).

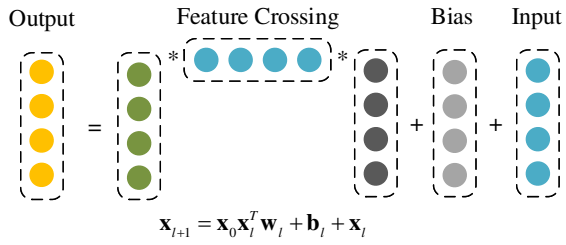


Fig. 2 The iteration of l-th layer in cross network

Cross network has the following advantages in learning cross features through Eq. (4):

1. Limited high-order. The cross multiplication order is determined by the network depth. When the number of hidden layer is l, we can learn the cross features of order $l+1$.
2. Automatic cross product. Cross outputs include all cross product combinations from the first order to the $l+1$ order of the original features, and the model parameters only increase linearly with the input dimension.
3. Parameters sharing. Cross product items have different weights, but not each cross product combination has an independent weight by sharing parameters. Cross network effectively reduces the numbers of parameters, and enables the model to be more generalized and robust.

3.4 Product Network

Product network is an MLP with multiple hidden layers. Unlike traditional MLPs, it adds a product layer to learn second-order feature interactions before the hidden layer. The product layer mainly includes embedding vectors and their vectors product with the formula as Eq. (5) or Eq. (6). Eq. (5) and Eq. (6) are respectively the vectors of inner and outer product. When product network contains only inner product, it is called IPNN. OPNN contains only outer product. PNN contains both the two products (Qu et al. 2016).

$$p_{i,j} = \langle \boldsymbol{v}_i \cdot \boldsymbol{v}_j \rangle = [\, v_i^1 \;\; v_i^2 \;\cdots\; v_i^k \,] \begin{bmatrix} v_j^1 \\ v_j^2 \\ \vdots \\ v_j^k \end{bmatrix} = \sum_{t=1}^{k} v_i^t v_j^t \tag{5}$$

$$\boldsymbol{v}_i \times \boldsymbol{v}_j = \begin{bmatrix} v_i^1 \\ v_i^2 \\ \vdots \\ v_i^k \end{bmatrix} [\, v_j^1 \;\; v_j^2 \;\cdots\; v_j^k \,] = \begin{bmatrix} v_{11} & \cdots & v_{1k} \\ \vdots & \ddots & \vdots \\ v_{k1} & \cdots & v_{kk} \end{bmatrix} \tag{6}$$

$$\Longrightarrow p_{i,j} = \sum_{i=1}^{k}\sum_{j=1}^{k} v_{ij}$$

In the MLP after the product layer, the hidden layers are all fully connected.

3.5 Stacking Network

Stacking network combines the high-order features outputted by the cross network and the product network as the new input of DNN, so that the entire model can be a whole for parameter training. When the hidden layer of stacking network is 0, stacking network is only a simple CTR prediction model. In fact, subsequent experimental results show that adding a small number of hidden layers can improve the performance to a certain extent.

4 Proposed algorithm

4.1 Improved LightGBM

The non-click and click samples in data is usually disproportionate in CTR prediction. It is predicted that the click-through rate of click samples is low, leading to poorer performance of model. The decreasing ability of the model to recognize click samples eventually causes overfitting. We impose L_1 and L_2 regularizations (Zou and Hastie 2005) controlled by hyper-parameter λ_1 and λ_2 on model parameter set ω to avoid overfitting. For the disproportion of non-click and click samples, the click samples are given a higher weight when calculating the loss function.

For the k-th tree of LightGBM, the original loss function is defined as:

$$L_{\text{original}} = -\frac{1}{N}\sum_{i=1}^{N} L(y_i, \hat{y}_i), \tag{7}$$

where y_i is true label, $\hat{y}_i$ is the predicted value by the k-th tree, and $L(\cdot)$ is the loss function. The most used loss function is Logloss (Vovk 2015). The improved loss function is:

$$L_{\text{modified}} = -\frac{1}{N}\sum_{i=1}^{N} \alpha_i L(y_i, \hat{y}_i) + \frac{\lambda_1}{2}\|\boldsymbol{\omega}\|_1 + \frac{\lambda_2}{2}\|\boldsymbol{\omega}\|_2^2, \tag{8}$$

where α_i, the weight of modified samples, is defined as:

$$\alpha_i = \begin{cases} 1, & y_i = 0 \\ \delta, & y_i = 1 \end{cases} \tag{9}$$

When the label of sample is 0, the weight is 1. When the label is 1, the weight is set as a number greater than 1 and then impacts of non-click samples can be further reduced. $\frac{\lambda_1}{2}\|\boldsymbol{\omega}\|_1$ is L_1 regularization, and the λ_1 is the hyper-parameter of L_1. $\frac{\lambda_2}{2}\|\boldsymbol{\omega}\|_2^2$ is L_2 regularization, and the λ_2 is the hyper-parameter of L_2. $\boldsymbol{\omega}$ in L_1 and L_2 is the parameter in the k-th tree.

4.2 Improved PNN

PNN uses vector product to learn second-order feature combination, including two ways of inner product and outer product. In this paper, we add a matrix-based vector-wise interaction algorithm (Lian et al. 2018) to the product layer of the PNN, which is named as VPNN (Vector & Product-based Neural Network). Its architecture is presented in Fig. 3.

The vector-wise method rolls all the embedding vectors into a two-dimensional matrix for learning. Figure 4 is the overview of the vector-wise architecture. The dimension of feature vectors processed by vector-wise n may be inconsistent with the dimension of the embedding vector. The last second-order interaction matrix is calculated via:

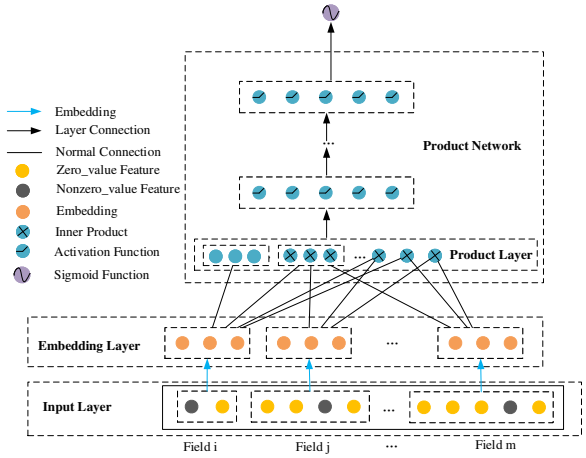


Fig. 3 The architecture of VPNN

$$\boldsymbol{X}_{h,*}^{1} = \sum_{i=1}^{m}\sum_{j=1}^{m} \boldsymbol{W}_{ij}^{1,h}(\boldsymbol{X}_{i,*}^{0} \circ \boldsymbol{X}_{j,*}^{0}), \tag{10}$$

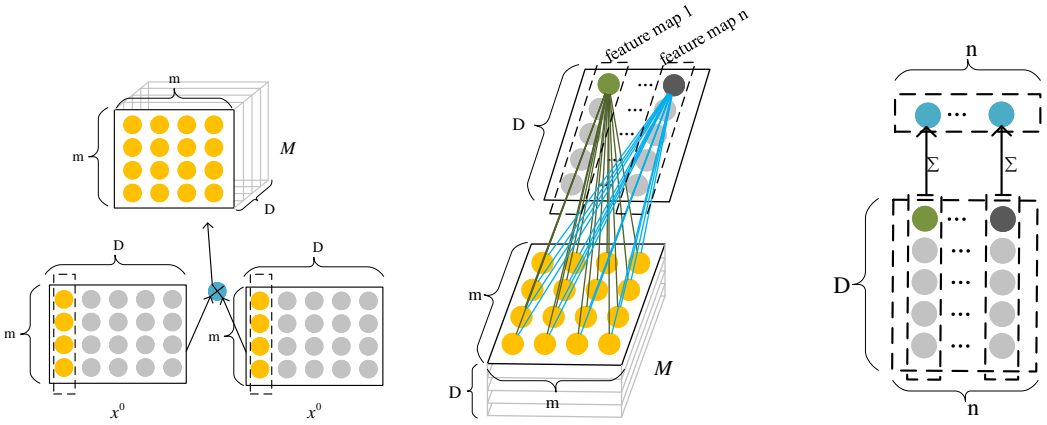
where $1 \leqslant h \leqslant n, \boldsymbol{W}^{1,h} \in \mathbb{R}^{m\times m}$ is the parameter matrix for the h-th feature vector, $\boldsymbol{X}_{i,*}^{0}$ and $\boldsymbol{X}_{j,*}^{0}$ are the i-th and the j-th embedding vectors. $\circ$ denotes the Hadamard product, such as $\langle a_1, b_2, c_3\rangle \circ \langle a_1, b_2, c_3\rangle = \langle a_1 a_1, b_2 b_2, c_3 c_3\rangle$.

The improved PNN can learn the pairwise feature interactions, and combine feature vectors for interactive learning. VPNN learns more features than PNN to ensure that the model improves the CTR prediction performance.

4.3 CTR prediction model

In this section, we introduce the details of PCSNL model. As shown in Fig. 5, the architecture of PCSNL is composed of embedding layer, cross network, product network and stacking network.

The low-order interactions are learned in cross network and product network. And The high-order combinations are learned in product network and stacking network. The input of stacking network combines the outputs of the cross and the product network. The difference between cross network and traditional DNN model is that cross network learns the feature interactions through Eq. (4). Product network learns the low-order feature combinations by adding a vector & product layer in front of the DNN model, which can



(a) Feature interactions with outer product. The matrix M is a temporary result for further learning

(b) Feature mapping. It compresses the temporary matrix M to n embedding vectors

(c) Sum pooling. It compresses n embedding vectors to a vector of dimension n by sum pooling

Fig. 4 Components and architecture of vector-wise

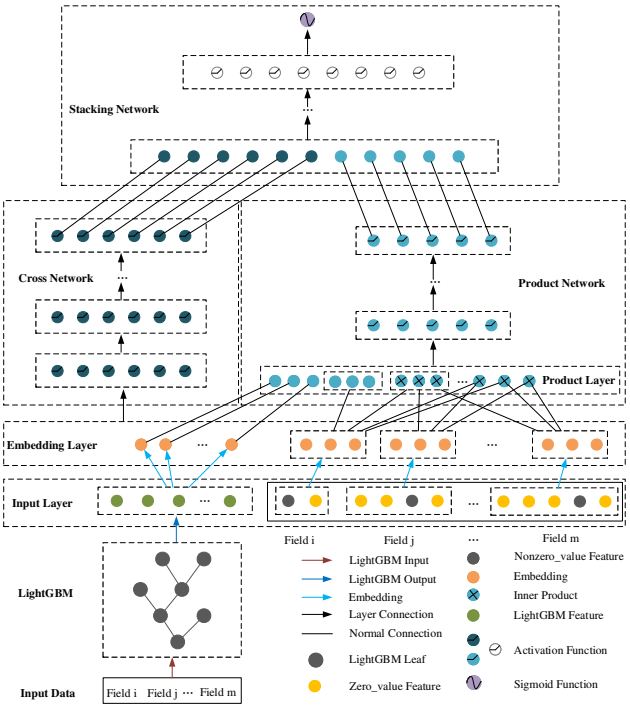


Fig. 5 The architecture of PCSNL

effectively help the model extract feature information. The model input consists of two parts: 1) the feature interactions generated by LightGBM, and 2) the preprocessed dataset. Compared with the existing CTR prediction model, the PCSNL model uses LightGBM to filter and extract feature combinations as the part of input of the PCSNL model, which can reduce the model's learning of low-order feature combinations. At the same time, the model can convert highly sparse data into dense matrix by adding an embedding layer before model training. In order to evaluate the performance of the PCSNL model, datasets of Huawei and Avazu are used to conduct experiments and the results are evaluated with AUC and Logloss in this paper. The hidden layers in the PCSNL model all adopt ReLU (Glorot et al. 2011) as the activation function. It can avoid the problem of either gradient explosion or gradient disappearance.

Adam (Kingma and Ba 2014) optimization algorithm is used to learn the weights and bias parameters in the PCSNL model. The loss function is Logloss (Vovk 2015),

$$L(y, \hat{y}) = -\frac{1}{N}\sum_{i=1}^{N}(y_i \log \hat{y}_i + (1 - y_i)\log(1 - \hat{y}_i)) \quad (11)$$

where y_i is true label, $\hat{y}$ is the click-through rate predicted by the PCSNL model, and N is the total number of inputs.

5 Experiments

5.1 Experimental environment and datasets

5.1.1 Experimental environment

The experiment environment is presented in Table 1. The normal machine learning models are implemented like logistic regression with the library *sklearn*[1] and for

[1] https://scikit-learn.org/stable/

Table 1 Experiment environments

CPU	AMD Ryzen 5600X
GPU	GeForce RTX 3060
Memory	32G
Operating System	Windows 10
Python	3.8
CUDA	11.3
Keras	2.4.3
TensorFlow	2.4.0
Scikit-learn	0.24.1

neural network, we depend on the library *tensorflow*[2] and *keras*[3].

5.1.2 Datasets

The proposed model is evaluated with the following two datasets:

1. **Huawei Dataset.** The Huawei dataset[4], the mobile ad CTR prediction contest released by Huawei on the Kaggle in 2020, provides the advertising data of mobile phone users collected by Huawei in 7 days.
2. **Avazu Dataset.** It is well known as a publicly accessible[5] industrial dataset to predict CTR. The Avazu dataset contains the advertising behavior data of Avazu's website users within 10 days.

The statistics of the two datasets are presented in Table 2. Huawei dataset contains 41,907,133 instances of display advertising. The dataset has 1,445,488 samples with click behavior, accounting for only 3.449% of the total. We negatively sample the dataset. The number of samples afterwards is 5,926,500, of which those with click behavior account for 24.39%. The dataset contains 35 categorical features. The ratio of training set and testing set is 4:1. 4 categorical features with too many values are discarded, and 31 categorical features are retained.

Avazu dataset contains 40,428,967 instances of display advertising and has 6,865,066 samples with click behavior, accounting for only 16.98% of the total. With the same sampling method, we get the last dataset with 20,595,198 samples. The click behavior samples account for 25%. 19 features are retained, which are 31,386 dimensions after being processed with one-hot.

[2] https://tensorflow.org/
[3] https://keras.io/
[4] https://www.kaggle.com/louischen7/2020-digix-advertisement-ctr-prediction
[5] https://www.kaggle.com/c/avazu-ctr-prediction

5.2 Metrics

In the experimentations, AUC (Area Under the ROC curve) (Graepel et al. 2010) and Logloss (cross entropy) (Vovk 2015) are two metrics to evaluate the performance of models. AUC is the under area in Receiver Operating Characteristic (ROC), and is used as one of assessment criteria in CTR prediction field. Some researches have proven its effectiveness. Logloss reflects the gap between the value predicted by the model and the real data. Understanding the loss function is conducive to subsequent optimization analysis model to achieve the optimal prediction performance.

5.3 Models for comparisons

We use the following state-of-the-art methods as baselines in our experiments:

- **LR** (Richardson et al. 2007). LR is the most used model in CTR prediction field for the longest time, which can be used as a benchmark model in experiments.
- **FM** (Steffen 2010, 2012). FM learns second features combination by learning the latent vectors of pairwise features, whose effectiveness has been proven by lots of competitions.
- **FFM** (Juan et al. 2016). FFM uses feature domains to learn feature combinations on basis of FM.
- **GBDT+LR** (He et al. 2014). In GBDT+LR, the input of the LR model is a combination of the features generated by GBDT and the original features.
- **Wide&Deep** (Cheng et al. 2016). Wide&Deep consists of LR model (Wide part) and DNN (Deep part). LR model is responsible for mining low-order feature information, and DNN mines the information on high-order features.
- **DeepFM** (Guo et al. 2017). DeepFM strengthens the ability of learning the low-order features information on basis of Wide&Deep. FM and DNN in DeepFM share the input.
- **DCN** (Wang et al. 2017). DCN uses the explicit feature interactions and neural network for CTR prediction.
- **PNN** (Qu et al. 2016). PNN uses the vector products to obtain the interaction information. The input of DNN in PNN is a combination of vector products and first-order features.

5.4 LightGBM hyper-parameter study

The grid parameter search algorithm is a basic parameter optimization algorithm. Its main idea is to divide

Table 2 The statistics of two datasets

	Huawei Dataset		Avazu Dataset	
	Training Set	Testing Set	Training Set	Testing Set
Sample Number	4,741,200	1,185,300	21,968,211	5,492,053
Click Number	1,155,662	289,826	5,491,121	1,373,945
Click Rate(%)	0.2437	0.2445	0.2499	0.2501
Feature Domain	31	31	19	19
Feature(sparse)	1,213	1,213	31,386	31,386
Embedding layer	341	341	209	209

the parameters that need to be optimized in a fixed range according to the step size grid, and then traverse all the parameter values to calculate the corresponding algorithm results. Lastly, we study the parameters corresponding to the best algorithm results obtained as the optimal parameters. The optimal parameters are presented in Table 3.

We firstly study the two parameters, *max_depth* and *num_leaves*, to ensure that LightGBM has better prediction performance. Because the non-click and click samples of the data are not proportionate, LightGBM may be overfitting. Therefore, we adjust the parameters in Table 3 except *max_depth* and *num_leaves* to prevent the problem.

5.5 Neural network hyper-parameters study

We use Huawei dataset to study the optimal hyper-parameters, and finally use Avazu dataset to verify PCSNL model's prediction performance.

5.5.1 Dimension of embedding vectors

The dimension of embedding vectors has great influences on the complexity of the entire model, especially the inputs of cross network and product network. When we study the optimal dimension, we set that both product network and cross network have 4 hidden layers and that stacking network has 2 hidden layers, so as to ensure that the model learns features in a better way. The structure of product network is {1024-512-256-64}, and stacking network is {512-64}. The dimension of the embedding vector ranges among {5,10,15,20} to avoid high model complexity. The parameters of the model increase from 1,819,287 to 2,143,337, then to 2,467,387, and finally to 2,791,437. Almost 1 million parameters to be learned have been added, so it is necessary to adjust the dimension of the embedding vector. The AUC and Logloss gained by the varying dimension of the embedding vector from 5 to 20 are not significantly improved. The reason is that the embedding layer is only connected to the first layer of the hidden layer, and has little influences on subsequent model training. The results of experiment are shown in Table 4. We set the dimension of embedding vector D as 10 according to Table 4.

5.5.2 Numbers of hidden layers

The product network, cross network and stacking network in the PCSNL model are all MLPs with different structures. Different hidden layers have different effects on the complexity of the model and the performance. In this section, we mainly discuss the impacts of different numbers of hidden layers on the three different network structures and on the performance of the final model.

The hidden layer in product network ranges among {2, 3, 4, 5, 6, 7}, and nodes range among {64, 128, 256, 512, 1024, 1024}. We set the maximum node of each hidden layer as 1024. The hidden layer in cross network ranges among {1, 2, 3, 4, 5, 6}. The hidden layer of stacking network is set from 0 to 5, and the nodes are {0, 64, 128, 256, 512}. The experiment first selects the hidden layer of product network, then cross network, and finally stacking network. As shown in Table 5 and Fig. 6, when the number of hidden layer in product network is 3, cross network 4, and stacking network 2, the performance of PCSN model comes to its best.

5.6 Model performance

In this section, we compare the performance of PCSNL with other CTR prediction models. We set the dimension of embedding vector D in models (FM, FFM, Wide & Deep, DeepFM, DCN, PNN, PCSNL) as 10. ReLU is the activation function in models including DNN, and Sigmoid (Marreiros et al. 2008) is the output node's activation function.

We train PCSNL model with parameters determined by the previous experiments. It can be found that the model has been trained for about 100 times to reach convergence in Fig. 7. But the training results after 100 times jittered many times. The reason may be the disproportion of non-click and click samples in the training

Table 3 The hyper-parameters of LightGBM study

Feature Name	Candidates	optimal parameter
max_depth	{3,4,5,6,7,8}	7
num_leaves	(5,256,5)[1]	105
max_bin	(1,256,5)[1]	145
min_data_in_leaf	(10,200,5)[1]	75
feature_fraction	{0.6,0.7,0.8,0.9,1.0}	0.8
bagging_fraction	{0.6,0.7,0.8,0.9,1.0}	0.7
bagging_freq	(0, 81, 10)[1]	60
lambda_l1	{1e-5,1e-3,0.01,0.03,0.08,0.1,0.2,$\cdots$,0.9}	0.4
lambda_l2	{1e-5,1e-3,0.01,0.03,0.08,0.1,0.2,$\cdots$,0.9}	0.9
min_split_gain	(0,1.0,20)[2]	0.45
min_child_leaf	(0.001,0.01,10)[2]	0.005
α_i	{1,2,3,4,5,6,7,8,9,10}	3
λ_1	{1e-5,1e-3,0.01,0.02,$\cdots$,0.09,0.1,0.2,$\cdots$,0.9}	0.2
λ_2	{1e-5,1e-3,0.01,0.02,$\cdots$,0.09,0.1,0.2,$\cdots$,0.9}	0.1

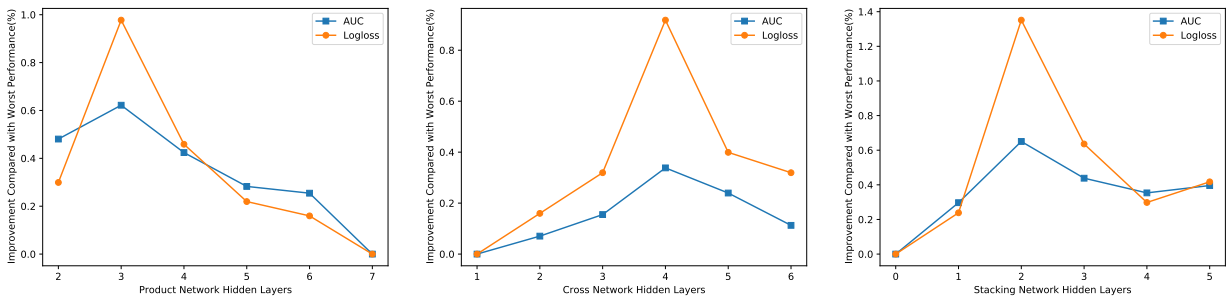
[1] (a,b,c) is an array with the minimum value a, the maximum value b, and the interval of c.
[2] (a,b,c) is an arithmetic array with the minimum value a, the maximum value b, and the number elements of c.

Table 4 Impact of the dimension of the embedding vector on performance

vector dimension	Logloss	AUC
$D = 5$	0.4998	0.7096
$D = 10$	**0.4990**	**0.7105**
$D = 15$	0.5002	0.7093
$D = 20$	0.5006	0.7095

data. Dropout is used in PCSN to prevent overfitting, with p set as 0.5. It means that 50% of the nodes in each layer are randomly discarded during training.

The results of using the Huawei dataset are shown in Table 6 and Fig. 8. It is shown in Fig. 8 that the performance of the LR model is the worst, because the LR model is a linear model and does not involve feature interactions. Compared with the LR model, FM, FFM and GBDT+LR have better performance than LR. Second-order feature interactions can be learned in models (FM, FFM and GBDT+LR), but they do not mine high-order feature interactions, so compared with deep learning models, performance of model is a bit worse. The models based on deep learning have better performance than linear models, because they have more advantages in learning high-order interaction cross features than low-order ones. The four models (Wide&Deep, DeepFM, PNN and DCN) have the ability to learn low-order feature interactions. Compared with other deep learning models, Wide&Deep, which only learns second-order feature interactions, has better performance. The reason may be that the non-click and click samples of the data are disproportionate, so we sample the non-click data and discard some features. The performance of the PCSNL model proposed in this paper has achieved the best improvement on the Huawei dataset. Compared with the LR model, the AUC and Logloss increase by 2.123% and 3.292%, respectively, which indicates that the cross network and product network can obtain more useful information. Hence, both of the networks are better for the model to improve the performance.



(a) Product Network (b) Cross Network (c) Stacking Network

Fig. 6 Impact of number of hidden layers on Logloss and AUC performance

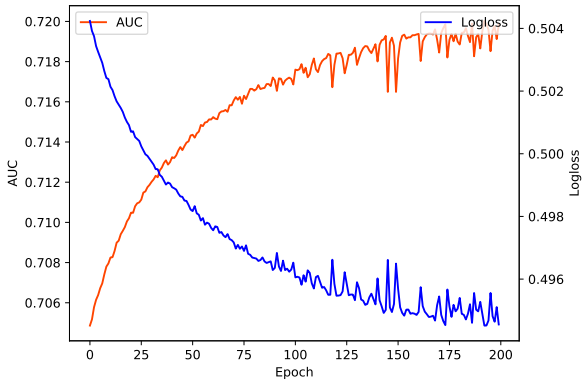


Fig. 7 PCSN model training process on Huawei dataset

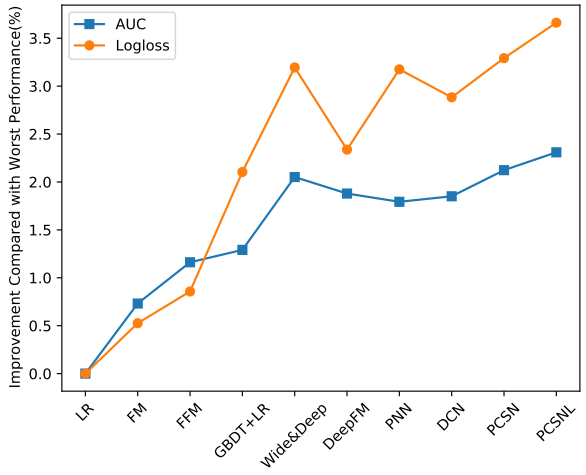


Fig. 8 Model performance on Huawei dataset

PCSNL has better performance than PCSN model in Fig. 8. We combine features generated by LightGBM with the original data as the input of the PCSN model, so that low-order and high-order feature combinations can be learned to ensure that the model has better performance. The experimental results also show that this method can improve the performance of the PCSN model to a certain extent.

After adjusting the hyper-parameters of the model using the Huawei dataset, the Avazu dataset is used to verify the effectiveness of the model. In the experiments in use of the Huawei dataset, it can be found that the performance of LR model is the worst, while the performance of the GBDT+LR model is the worst when using the Avazu dataset. It can be found through comparison that the results of experiments of other models based on both datasets have no obvious disparity. Finally, PCSN and PCSNL proposed in this paper achieve better performance than other models. The experimental results are shown in Table 7 and Fig. 9. Compared with the GBDT+LR model, the AUC and Logloss of the PCSN model increase by 4.21% and 4.617%, respectively, and the PCSNL increases by 4.524% and 5.060%, respectively.

In summary, the PCSNL model can achieve better convergence by analyzing the PCSNL model training process. At the same time, we use the Huawei dataset and the Avazu dataset to compare the performance with the existing traditional CTR prediction models. The experimental results show that the PCSNL model proposed in this paper can obtain better performance. Finally, it is verified in this paper that the PCSNL model has sound effectiveness in CTR prediction through experiments.

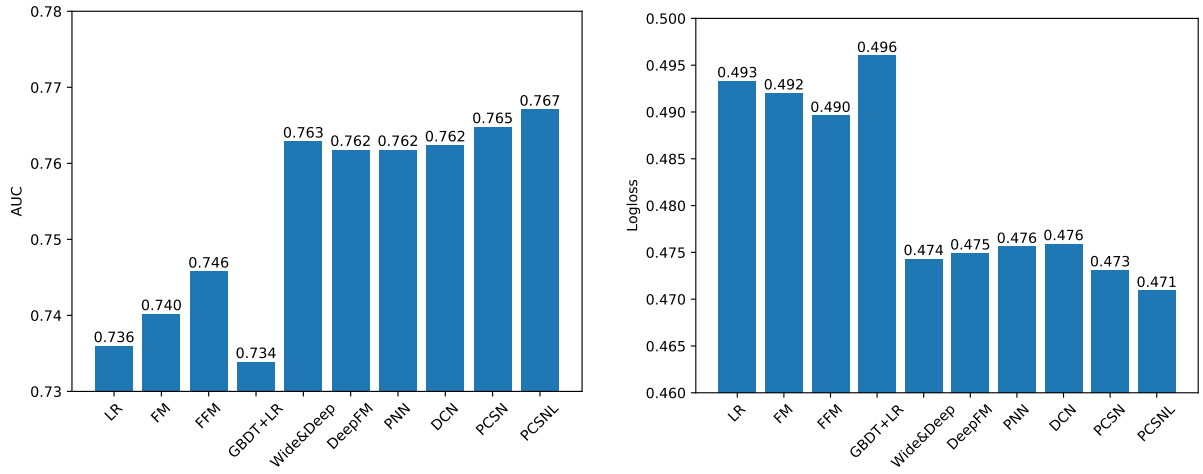
6 Conclusion and future work

The application of deep learning models in CTR prediction has become a new research direction. In this paper, an integrated structure of deep learning model is proposed, which can better learn low-order and high-order interaction cross features. The main contributions of this paper include:

1. The model input includes both first-order feature and second-order interaction cross features by adding LightGBM to the PCSN model.
2. The model combines cross network and improved product network to predict CTR.
3. It is verified that the PCSNL model has good performance in CTR prediction through experiments based on two datasets.

Table 5 Impact of number of network hidden layers

Product Network			Cross Network			Stacking Network		
layers	Logloss	AUC	layers	Logloss	AUC	layers	Logloss	AUC
2	0.4998	0.7109	1	0.5010	0.7095	0	0.5032	0.7073
3	**0.4964**	**0.7119**	2	0.5002	0.71	1	0.5020	0.7094
4	0.4990	0.7105	3	0.4994	0.7106	2	**0.4964**	**0.7119**
5	0.5002	0.7095	4	**0.4964**	**0.7119**	3	0.5000	0.7104
6	0.5005	0.7093	5	0.4990	0.7111	4	0.5017	0.7098
7	0.5013	0.7075	6	0.4999	0.7103	5	0.5011	0.7101



(a) AUC (b) Logloss

Fig. 9 Model performance on Avazu dataset

Table 6 Model performance on Huawei dataset

Model	Logloss	GAP(%)	AUC	GAP(%)
LR	0.5133	0	0.6971	0
FM	0.5106	0.526	0.7022	0.732
FFM	0.5089	0.857	0.7052	1.162
GBDT+LR	0.5025	2.104	0.7061	1.291
Wide&Deep	**0.4969**	**3.195**	**0.7114**	**2.051**
DeepFM	0.5013	2.338	0.7102	1.879
PNN	0.4970	3.176	0.7096	1.793
DCN	0.4985	2.883	0.7100	1.851
PCSN	**0.4964**	**3.292**	**0.7119**	**2.123**
PCSNL	**0.4945**	**3.663**	**0.7132**	**2.310**

Table 7 Model performance on Avazu dataset

Models	Logloss	GAP(%)	AUC	GAP(%)
GBDT+LR	0.4960	0	0.7339	0
LR	0.4933	0.544	0.7360	0.286
FM	0.4920	0.807	0.7402	0.858
FFM	0.4896	1.290	0.7458	1.622
Wide&Deep	0.4743	4.375	0.7629	3.952
DeepFM	0.4749	4.254	0.7618	3.802
DCN	0.4756	4.113	0.7618	3.802
PNN	0.4759	4.052	0.7624	3.883
PCSN	0.4731	4.617	0.7648	4.210
PCSNL	**0.4709**	**5.060**	**0.7671**	**4.524**

Although the PCSNL model proposed in this paper has achieved good performance in CTR prediction, it still has some shortcomings. Therefore, further study shall be conducted from the following aspects:

1. In this paper, *hour*, an attribute in the Avazu data, represents the user's click behavior at different time, which indicates a potential link between the user's changes and time. It would be workable to add an Attention Mechanism to the PCSNL model to learn the distribution of user interests.
2. The currently used CTR prediction datasets have the problem of disproportion between non-click and click samples. It is considered to use the popular Generative Adversarial Networks (GAN) (Goodfellow et al. 2014) or oversampling algorithm Geometric SMOTE (G-SMOTE) (Douzas and Bacao 2019) to generate a small number of click samples.

7 Compliance with Ethical Standards

7.1 Ethical approval

This article does not contain any studies with human participants or animals performed by any of the authors.

7.2 Funding details

This study received no external funding.

7.3 Conflict of interest

All authors declare that they have no conflict with interest.

8 Author Contributions

Z. Xia was involved in conceptualization, data curation and formal analysis, investigation and methodology, validation and visualization, and writing the orig-

inal draft. S. Mao and J. Bai took part in conceptualization, project administration and resources, writing, reviewing and editing. X. Geng took part in conceptualization, project administration and resources, methodology, investigation, writing, reviewing and editing. L. Yi was involved in data curation and formal analysis, validation and visualization, and editing. All authors have read and agreed to the published version of the manuscript.

References

Asdemir K, Kumar N, Jacob VS (2012) Pricing models for online advertising: CPM vs. CPC. Information Systems Research 23(3):804–822

Blondel M, Fujino A, Ueda N, Ishihata M (2016) Higher-order factorization machines. In: Advances in Neural Information Processing Systems, pp 3351–3359

Chang YW, Hsieh CJ, Chang KW, Ringgaard M, Lin CJ (2010) Training and testing low-degree polynomial data mappings via linear svm. The Journal of Machine Learning Research 11:1471–1490

Chapelle O, Manavoglu E, Rosales R (2015) Simple and scalable response prediction for display advertising. ACM Transactions on Intelligent Systems and Technology 5(4):1–34

Chen T, Guestrin C (2016) XGBoost: A scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, California, USA, pp 785–794

Cheng HT, Koc L, Harmsen J, Shaked T, Chandra T, Aradhye H, Anderson G, Corrado G, Chai W, Ispir M, Anil R, Haque Z, Hong L, Jain V, Liu X, Shah H (2016) Wide & Deep learning for recommender systems. In: Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, New York, NY, USA, pp 7–10

Chu X, Zheng A, Zhang X, Sun J (2020) Detection in crowded scenes: One proposal, multiple predictions. In: Proceedings of CVPR'2020, pp 12214–12223

Covington P, Adams J, Sargin E (2016) Deep neural networks for youtube recommendations. In: Proceedings of the 10th ACM Conference on Recommender Systems, Association for Computing Machinery, New York, NY, USA, RecSys '16, pp 191–198

Douzas G, Bacao F (2019) Geometric smote a geometrically enhanced drop-in replacement for smote. Information Sciences 501:118–135

Friedman JH (2001) Greedy function approximation: a gradient boosting machine. The Annals of Statistics 29(5):1189–1232

Glorot X, Bordes A, Bengio Y (2011) Deep sparse rectifier neural networks. In: Proceedings of the 14th International Conference on Artificial Intelligence and Statistics, vol 15, pp 315–323

Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y (2014) Generative adversarial networks. Advances in Neural Information Processing Systems 3:2672–2680

Graepel T, Candela JQ, Borchert T, Herbrich R (2010) Web-scale bayesian click-through rate prediction for sponsored search advertising in Microsoft's bing search engine. In: Proceedings of the 27th International Conference on Machine Learning, Haifa, Israel, pp 13–20

Guo H, Tang R, Ye Y, Li Z, He X (2017) DeepFM: A factorization-machine based neural network for ctr prediction. In: Proceedings of the IJCAI-2017, Melbourne, Australia, pp 1725–1731

He X, Pan J, Jin O, Xu T, Liu B, Xu T, Shi Y, Atallah A, Herbrich R, Bowers S, Quiñonero JC (2014) Practical lessons from predicting clicks on ads at facebook. In: Proceedings of the 8th International Workshop on Data Mining for Online Advertising, New York, NY, USA, pp 1–9

Juan Y, Zhuang Y, Chin WS, Lin CJ (2016) Field-aware factorization machines for ctr prediction. In: Proceedings of the 10th ACM Conference on Recommender Systems, New York, NY, USA, pp 43–50

Ke G, Meng Q, Finley T, Wang T, Chen W, Ma W, Ye Q, Liu TY (2017) LightGBM: a highly efficient gradient boosting decision tree. In: Advances in Neural Information Processing Systems 30, La Jolla, pp 3147–3155

Kingma D, Ba J (2014) Adam: A method for stochastic optimization. Computer Science

Koren Y, Bell R, Volinsky C (2009) Matrix factorization techniques for recommender systems. Computer 42(8):30–37

Kudo T, Matsumoto Y (2003) Fast methods for kernel-based text analysis. In: Proceedings of the 41st Annual Meeting on Association for Computational Linguistics, Sapporo, Japan, pp 24–31

Lian J, Zhou X, Zhang F, Chen Z, Xie X, Sun G (2018) xDeepFM: combining explicit and implicit feature interactions for recommender systems. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Association for Computing Machinery, London, United Kingdom, pp 1754–1763

Marreiros AC, Daunizeau J, Kiebel SJ, Friston KJ (2008) Population dynamics: Variance and the sigmoid activation function. NeuroImage 42(1):147–157

Miralles-Pechuán L, Rosso D, Jiménez F, García JM (2017) A methodology based on deep learning for advert value calculation in cpm, cpc and cpa networks. Soft Computing 21(3):651–665

Mohamed Ar, Dahl GE, Hinton G (2012a) Acoustic modeling using deep belief networks. IEEE transactions on audio, speech, and language processing 20(1):14–22

Mohamed Ar, Hinton G, Penn G (2012b) Understanding how deep belief networks perform acoustic modelling. In: ICASSP'2012, pp 4273–4276

Qu Y, Cai H, Ren K, Zhang W, Yu Y, Wen Y, Wang J (2016) Product-based neural networks for user response prediction. In: 2016 IEEE 16th ICDM, Barcelona, Spain, pp 1149–1154

Richardson M, Dominowska E, Ragno R (2007) Predicting clicks: estimating the click-through rate for new ads. In: Proceedings of the 16th International Conference on World Wide Web, Banff, Alberta, Canada, pp 521–530

Schmidhuber J (2015) Deep learning in neural networks: an overview. Neural Networks 61:85–117

Shan Y, Hoens TR, Jiao J, Wang H, Yu D, Mao JC (2016) Deep crossing: web-scale modeling without manually crafted combinatorial features. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, pp 255–262

Steffen R (2010) Factorization machines. In: 2010 IEEE International Conference on Data Mining, Sydney, NSW, Australia, pp 995–1000

Steffen R (2012) Factorization machines with libfm. ACM Transactions on Intelligent Systems and Technology 3(3)

Sulthana AR, Gupta M, Subramanian S, Mirza S (2020) Improvising the performance of image-based recommendation system using convolution neural networks and deep learning. Soft Computing 24(19):14531–14544

Tao Z, Wang X, He X, Huang X, Chua TS (2020) HoAFM: A high-order attentive factorization machine for ctr prediction. Information Processing & Management 57(6):1–11

Vedavathi N, Kumar KMA (2021) An efficient e-learning recommendation system for user preferences using hybrid optimization algorithm. Soft Computing 25(14):9377–9388

Vovk V (2015) The fundamental nature of the log loss function, Springer International Publishing, Cham, pp 307–318

Wang R, Fu B, Fu G, Wang M (2017) Deep & cross network for sd click predictions. In: Proceedings of the ADKDD'17, New York, NY, USA

Wen X (2021) Using deep learning approach and iot architecture to build the intelligent music recommendation system. Soft Computing 25(4):3087–3096

Xiao J, Ye H, He X, Zhang H, Wu F, Chua TS (2017) Attentional factorization machines: learning the weight of feature interactions via attention networks. In: Proceedings of the IJCAI-2017, AAAI Press, Melbourne, Australia, pp 3119–3125

Yang L, Liu J (2020) Tuningmalconv: Malware detection with not just raw bytes. IEEE Access 8:140915–140922

Yang L, Liu J, Zhang Y (2019) An intelligent security defensive model of scada based on multi-agent in oil and gas fields. International Journal of Pattern Recognition and Artificial Intelligence 34(1):1–13

Yu D, Deng L (2011) Deep learning and its applications to signal and information processing. IEEE Signal Processing Magazine 28(1):145–154

Zhang W, Du T, Wang J (2016) Deep learning over multi-field categorical data: a case study on user response prediction. In: Advances in Information Retrieval, Cham, pp 45–57

Zhu J, Shan Y, Mao JC, Yu D, Rahmanian H, Zhang Y (2017) Deep embedding forest: forest-based serving with deep embedding features. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, pp 1703–1711

Zou H, Hastie T (2005) Regularization and variable selection via the elastic net. Journal of the royal statistical society: series B (statistical methodology) 67(2):301–320