Exploring Different Interaction Among Features For CTR Prediction

Leilei Yang
Tianjin University of Technology
Wenguang Zheng (wenguang_zheng@163.com)
Tianjin University of Technology
Yingyuan Xiao
Tianjin University of Technology

Research Article

Keywords: CTR prediction, second-order interaction, explicit feature interactions, attention mechanism

Posted Date: February 10th, 2022

DOI: https://doi.org/10.21203/rs.3.rs-1010709/v1

Soft Computing manuscript No.
(will be inserted by the editor)

Exploring different interaction among features for CTR prediction

Leilei Yang [1] · Wenguang Zheng [2] · Yingyuan Xiao[3]

the date of receipt and acceptance should be inserted later

Abstract Click-through rate (CTR) prediction occupies a very important position in recommendation systems and computational advertising. Accurate prediction of the click-through rate can help advertisers make more detailed advertising and marketing planning and can also help the media in defining their service audience and recommend more popular content to users more clearly. In this paper, we propose a model to make full use of the explicit feature interactions to improve the click-through rate of advertising. In the second-order feature interactions, our model not only considers the fact that the same feature uses different latent vectors when interacting with other features but also adds the latent vectors of each feature to form the embedding vector of the whole feature and then carries out explicit high-order interaction, to learn more nonlinear relations. We, inspired by the Squeeze-and-Excitation Networks (SENet), give different weights to different feature interactions. Extensive experimentation has been done on Avazu and Criteo to show that our model is better than the existing advanced models.

Keywords CTR prediction, second-order interaction, explicit feature interactions, attention mechanism

✉Wenguang Zheng
E-mail: wenguang_zheng@163.com

1
Tianjin Key Laboratory of Intelligence Computing and Novel Software Technology, Tianjin University of Technology, Tianjin, China.

2
Tianjin Key Laboratory of Intelligence Computing and Novel Software Technology, Tianjin University of Technology, Tianjin, China.

3
Tianjin Key Laboratory of Intelligence Computing and Novel Software Technology, Tianjin University of Technology, Tianjin, China.

1 Introduction

With the development of society and the progress of science and technology, the content on the Internet is continuously increasing. It is, sometimes, difficult for people to obtain the information they want and hence the prediction of click-through rate is very important. For the advertising platform, accurate prediction of the click-through rate of advertising can understand the users fully so that the platform can serve the users as well as allocate the resources better. More importantly, a more accurate click-through rate prediction means a higher order fulfillment rate with a direct effect on improving the satisfaction of advertisers and improving the revenue from the advertising platform.

Logistic regression (LR) is one of the proposed methods in the development process of CTR (Juan et al. (2016); Wang et al. (2017); Guo et al. (2017); Zhou et al. (2018b); Feng et al. (2019); Ouyang et al. (2019a); Huang et al. (2019); Zhou et al. (2019); Pi et al. (2019); Ouyang et al. (2019b); Lyu et al. (2020); Xu et al. (2020); Liu et al. (2020)). The LR uses one-hot coding, which converts categorical features into vectors as input. However, the features after one-hot are too sparse and lead to a larger feature space. Moreover, logical regression relies on artificial feature engineering; hence it cannot learn the cross features that do not appear in the training set. This problem is solved by Factorization Machines (FM) (Rendle (2010)) , which achieves the purpose of learning combination feature weights by performing inner product operation on two latent vectors, while the Field-aware Factorization Machines (FFM) adds the concept of the field in FM. Each feature uses different latent vectors for different fields; thus, each feature corresponds to a group of latent vectors that enhance the expression of the model.Since it is very important to learn combined features in the CTR prediction, Product-based Neural Networks (PNN) (Qu et al. (2016)) learn feature interactions in the form of inner prod-

uct, outer product, and inner product plus outer product. However, the PNN does not use different vectors when one feature interacts with other features. The Operation-aware Neural Networks (ONN) (Yang et al. (2020)) performs inner product operation after the operation-aware embedding, solving the problem of using different vectors when the same feature interacts with other features.

In this article, we propose a new model. We know that the correlation between each feature and different features is different. When a feature uses the same vector when interacting with other features, this correlation will be ignored. Therefore, our model considers that a feature uses different hidden vectors when interacting with other features in the second-order feature interaction (Juan et al. (2016)) , which reflects the correlation between features. In addition, during high-order feature interaction, we add up the hidden vectors of each feature to form the embedding vector of the integral feature as an input, which reflects the integrity of the feature. After the feature embedding operation, two layers are introduced in parallel in our model:SENet layer (Hu et al. (2018)) and Cross layer (Wang et al. (2017)). In different scenarios, users pay different attention to different features. If we give the same weight to different cross features, a lot of useful information will be ignored. The SENet layer gives different weights to different interactive features in our model. Compared with the previous deep neural networks, the Cross layer can explicitly carry out high-order interaction and learn more nonlinear relationships.

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

- Our model does not lose the correlation between features when performing second-order interaction after embedding operation, and does not lose the integrity of features when performing high-order feature interaction after embedding operation. Specifically, in the second-order feature interaction, we consider that a feature uses different hidden vectors when interacting with other features, which reflects the correlation between features. In high-order feature interaction, the hidden vectors of each feature are added to form the embedding vector of the integral feature as the input, which reflects the integrity of the feature.
- By introducing the SENet attention module after the second-order feature interactions, different interactive features are given different weights.
- To improve the ability of feature crossing, second-order and explicit high-order interactions are also carried out to explore more nonlinear relationships between features.

The rest of this paper is structured as follows: the second section mentions the related work. In work related to our model, we have mentioned the investigation and research in four different directions. The third section introduces the model in detail, while in the fourth section, we describe the extensive experiments conducted on two open datasets, and we have summarized our work in the fifth section.

2 Related work

2.1 Deep learning evolution of the FM model

The Click-through rate (CTR) is a very important parameter of the computational advertising and recommendation system. In the process of CTR prediction, it is often necessary to combine multiple features. The FM solves the problem of high dimension and highly sparse input feature combination well. Based on the logistic regression, a second-order part is added to the model to obtain the corresponding latent vector for each dimension. The weight is modeled by the latent vector inner product, and the feature combination, which has not appeared in the training set, can also be learned effectively. Due to the limitation of the combinatorial explosion, the model is not easy to be extended to the third-order feature crossing. FFM adds the concept of the feature field to the FM model so that each feature adopts different weights when crossing with features of different fields. Compared with the FM, the ability of feature interactions is further enhanced. From modifying the second-order part of FM, the Neural Factorization Machines (NFM) (He and Chua (2017)) replaces the feature crossing part of FM with a Deep Neural Network (DNN) with a Bi-interaction pooling layer. Bi-interaction pooling can be regarded as the form of element-wise product embedding, with different features. As compared to FM, the NFM has stronger expression and feature crossing abilities. The FM model is adopted for the embedding layer of Factorization Machine supported Neural Network (FNN) (Zhang et al. (2016)) to conduct the dimensional reduction with supervision for sparse features and transform them into dense, continuous features. The convergence speed of FNN becomes faster by using the FM initialization parameter. The Translation-based Factorization Machines (TransFM) (Pasricha and McAuley (2018)) combines the ideas of FM and TransRec and applies them to sequential recommendations. The advantage of this method is it uses a simple model to depict complex interactions while achieving good results. The TransFM changed the inner product calculation method in the FM and used the square Euclidean distance to improve the generalization ability of the model and the transitivity between sample features.

2.2 Combination model

To integrate the advantages of multiple models, combining different models is a common method to build a recommended model. Wide & Deep (Cheng et al. (2016)) combined the Wide Linear Model and DNN for training. The ad-

vantage of the Wide & Deep Model is that the memorization ability of the Wide Model and the generalization ability of the Deep Model are obtained at the same time, and the generalization ability of the Deep Model is pioneered in the construction method of the combined model. This has a significant impact on the subsequent development of the recommended model of deep learning. However, the wide section needs to be screened manually for feature combination. The Deep & Cross Network (DCN) replaces the Wide part in the Wide & Deep Model with Cross Network, which can effectively capture the feature combination of specific order and learn highly nonlinear interaction, without the need of artificial feature engineering. Based on the Wide & Deep Model, the DeepFM replaces the original linear wide part with FM to enhance the feature crossing ability of the wide part. The FM and the Deep Modules share the feature embedding part, accelerating the model training speed. The xDeepFM (Lian et al. (2018)) learns the high-order feature interactions automatically, in both explicit and implicit ways, which makes the feature interactions occur at the vector level, and has the ability of memory and generalization.

2.3 The combination of attention mechanism and recommendation model

The Attention mechanism is inspired by human habits. For example, when people browse the website or Taobao home page casually, they are always attracted by specific areas. Therefore, if the attention mechanism is taken into account in modeling, the accuracy of the recommendation results can be improved greatly. By introducing the attention mechanism, Attentional Factorization Machines (AFM) (Xiao et al. (2017)) is used to assign different importance to different feature combinations where the weights in the network can be learned automatically without introducing any additional field knowledge. Moreover, the Deep Interest Network (DIN) introduced the attention mechanism based on the traditional deep learning recommendation system model and calculated the attention score by using the correlation between the historical items of user behavior and the target advertising items, and thus, according to the different target advertising items, more targeted recommendations are made. However, the user interests are constantly evolving, while the DIN extracts user interests that are independent of each other, without capturing the dynamic evolution of interests. ATRank (Zhou et al. (2018a)) proposed a general user behavior sequence modeling framework, trying to integrate different types of user behaviors and conduct more detailed processing of the user heterogeneous behavior data. Momentas network architecture SENet, the champion of the ImageNet 2017 Challenge, can learn from the importance of different features, thereby weighting the important features and weakening the features that contain little information. The AutoInt (Song et al. (2019)) uses a multi-head Self Attention mechanism to perform automatic feature cross-learning, to improve the accuracy of the CTR prediction task. The explicit learning problem of high-order feature interactions, with good interpretability, is also studied. The user behavior in each session is similar, but the difference between different sessions is significant. Therefore, the Deep Session Interest Network (DSIN) is proposed to model user behavior closely related to the session. The closer the users conversational interest is to the target item, the greater is the weight assigned by DSIN through the attention mechanism. Additionally, the Deep Spatio-Temporal Neural Networks (DSTN) considers both the spatial and temporal field information to estimate the click-through rate of advertising. The DSIN puts forward two Attention models, one is the Self-Attention Model, and the other is the Interactive Attention Model, in which the latter improves the former.

2.4 Changing the way of features cross

The proposed model enriches the way of feature crossing in a deep learning network. The PNN adds a product layer between the embedding layer and the full connection layer to complete a targeted feature crossing. Its product operation combines the features between different feature fields and uses an inner product or outer product to learn the high-order nonlinear features. The Neural Collaborative Filtering (NCF) (He and Chua (2017)) replaces the traditional dot product operation of the user vector, and the item vector in matrix decomposition with the interoperability replaced by a neural network and only the ID features of users and items are used; no other features are added. The cross-network in the DCN uses a multi-layer residual network to fully feature cross each dimension of the feature vectors.

3 Model

The overall framework of our model is shown in Figure 1. In our model, the input is sorted as sparse input and dense input, denoted as x_{sparse} and x_{dense} respectively, where $x_{sparse} = [x_1, x_2, \ldots, x_n]$, where n is the number of features. The sparse input is one-hot coding, but this coding method has the disadvantages of data sparseness and large space occupation, and hence, inspired by the FFM (Juan et al. (2016)) and ONN (Yang et al. (2020)), we embed the sparse features. After the embedding layer, shown on the left side of the model diagram, we perform the inner product operation for learning the second-order interaction between the features. To learn the importance of the feature interactions, we go through the SENet layer (Hu et al. (2018)) after feature interactions. On the right side of the model diagram, we first add the latent vectors generated by each feature and add each

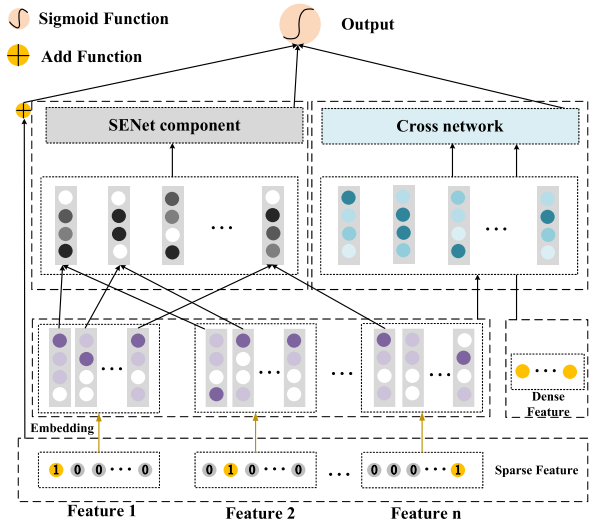

Fig. 1 Overall framework of the model.

Table 1 Notation

Notation	Explanation
x_{sparse}	Sparse input features
x_{dense}	Dense input features
x_i	The ith feature
n	The number of sparse features
e	Embedding vector
v_i	The embedding matrix corresponding to the ith feature
v_i^j	The latent vector of the ith feature interacting with the jth feature
P	The inner product of embedding vector e
S	The result of the average pooling of vector P
A	The result of vector S after excitation
Q	The vector after reweighting P
c	The vector generated by merging embedding vectors and dense features
l	The layers of cross network

feature to an embedded vector. We then interact with the features explicitly. Table 1 defines some symbols used in this paper.

3.1 Embedding layer

In this section, we introduce the process of feature embedding in detail. Suppose we have three features: x_1, x_2, x_3 and v_1, v_2, v_3 are the latent vectors of the three features respectively. When x_1 interacts with x_2 and x_3 , if different representations are not considered for different features interaction, then the weights of x_1 and x_2 and x_1 and x_3 are $w_{x_1,x_2} = v_1 \cdot v_2$ and $w_{x_1,x_3} = v_1 \cdot v_3$, respectively. As far as we know, When x_1 interacts with x_2 and x_3, the importance of each interaction is different. For example, if x_1 represents male users, x_2 represents basketball and x_3 represents lipstick, then in general, male users prefer to buy basketball rather than a lipstick, and therefore, we should consider different representations of different features interactively. Then the weights of x_1 and x_2, x_1 and x_3 are $w_{x_1,x_2} = v_1^2 \cdot v_2^1$ and $w_{x_1,x_3} = v_1^3 \cdot v_3^1$, respectively. In this case, two different vectors are used when x_1 interacts with x_2 and x_3. Figure 2 shows the feature embedding process. It can be

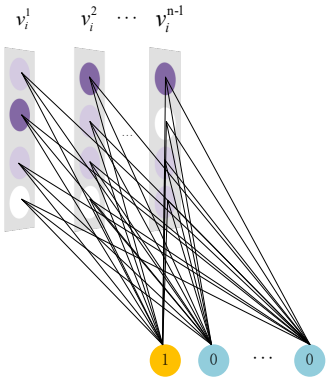

Fig. 2 Embedding process.

seen that a feature generates $n-1$ latent vectors, which are the vectors of one feature interacting with other $n-1$ features. As shown in Figure 3, we combine the $n-1$ vectors to get the embedding vector of the whole feature in which each embedding vector is represented as e_i. Thus, the embedding vector of n features can be expressed as $e = [e_1, e_2, \ldots, e_i, \ldots, e_n]$, where $e_i = v_i \cdot x_i$, v_i is its embedding matrix. For example, v_1 is specifically expressed as $v_1 = [v_1^2, v_1^3, \ldots, v_1^n]$, where v_1^2 represents the latent vector of the interaction between the first and the second feature, v_1^n is the latent vector of interaction between the first feature and the corresponding nth feature, that is, each feature will learn the corresponding latent vector for the other $n-1$ features.

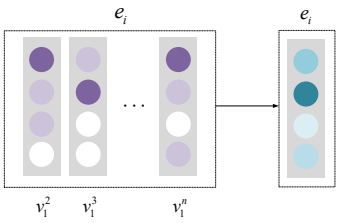

Fig. 3 The vector merging process.

We input e into the cross-network for explicit high-level interaction, the details of which are mentioned in Section 3.4.

3.2 Second-order interaction layer

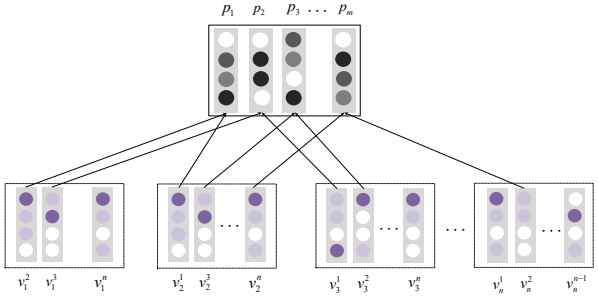

Fig. 4 Second-order interaction feature process.

As shown in Figure 4, each feature learns $n-1$ vectors. Thus, each feature has a unique latent vector corresponding to other features, so Hadamard product operation on the corresponding two latent vectors can be performed, such as v_i^j, $v_j^i, i, j \in [1,\ldots,n]$. For Figure 4, the second-order interaction can be expressed by the following formula:

$$P = \sum_{i=1}^{n} \sum_{j=i+1}^{n} (v_i^j \odot v_j^i) x_i x_j \tag{1}$$

After the second-order interaction, we get m new vectors, which are represented as $P = [p_1, p_2, \ldots, p_m]$, where $m = (n*(n-1))/2$ and vector P is the input of SENet.

3.3 SENet layer

When people usually buy goods, they may prefer to buy one kind of goods. Thus while forecasting these kinds of goods, we should give more weight. As an example, for people who usually like to buy skirts rather than lipstick should be given more weight to skirts than lipstick. To achieve this goal, we introduce the SENet, first applied in the field of the image, which can increase the weight of important features and weaken the weight of relatively unimportant features.

The SENet block is divided into three-step processes: squeeze step, explanation step, and reweight step, as shown in Figure 5:

Lets start with the squeeze step: In this step, the input is the vector $P = [p_1, p_2, \ldots, p_m]$. Then, we compress each of the dimensions of the vector into one dimension and compress the input into vector $S = [s_1, s_2,$

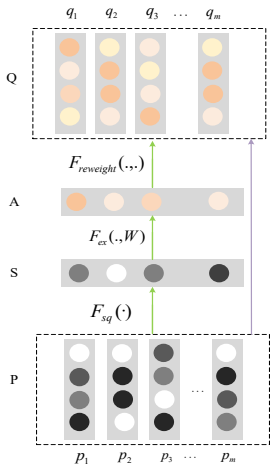

Fig. 5 SENet processes.

$\ldots, s_i, \ldots, s_m]$, where s_i is a scalar, $i \in [1,\ldots,m]$. The calculation of the squeeze is represented as follows:

$$s_i = F_{sq}(p_i) = \frac{1}{k} \sum_{t=1}^{k} p_i^{(t)} \tag{2}$$

Next, we introduce the excitation step: In this step, we use two fully connected layers to learn the weight of vector S. In the first layer, we reduce the dimension with parameter W_1 and then through the activation function σ_1. In the second layer, W_2 is used to restore the dimension before the dimension reduction and then through the activation function σ_2. Then we can learn the new vector $A = [a_1, a_2, \ldots, a_m]$, calculated as:

$$A = F_{ex}(S) = \sigma_2(W_2 \sigma_1(W_1 S)) = [a_1, a_2, a_m] \tag{3}$$

Where $A \in R^m$, $W_1 \in R^{m \times \frac{m}{r}}$, $W_2 \in R^{\frac{m}{r} \times m}$, r is the dimension reduction ratio.

Finally, we introduce the reweight step: In this step, we multiply the vector P with the corresponding position elements of A, i.e., reweighting P to get the final output of SENet: $Q = [q_1, q_2, \ldots, q_m]$. The calculation of Q is given as:

$$Q = F_{ReWeight}(A, P) = [a_1 \cdot p_1, , a_m \cdot p_m] = [q_1, q_2, q_m] \tag{4}$$

Here, $a_i \in R$, $i \in R^d$, $q_i \in R^d$. d is the embedding dimension.

3.4 Explicit high-order interaction layer

In this section, to learn the high-order interaction between features, we introduce the cross-network (Wang et al. (2017)). The inputs of the cross-network are embedding feature and dense feature. Firstly, we need to compress the embedding feature e into a line and splice it with the dense feature to generate a new vector c, while the merging process is shown in Figure 6.

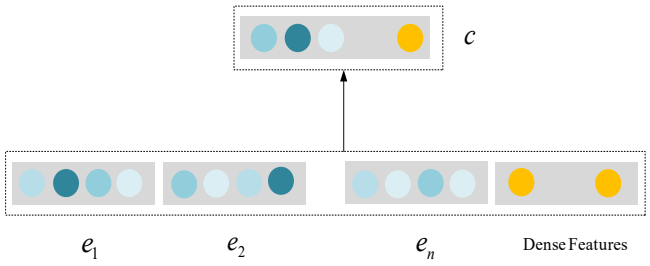

Fig. 6 The merging process of embedding vectors and dense features.

After merging into a vector, we input c into the cross-network. The interaction process of cross-network is shown in Figure 7. From Figure 7, we can see that each layer interacts with the input c, where deeper the layers are, greater is the degree of interaction. The interaction process from the first layer to the lth layer is calculated as follows:

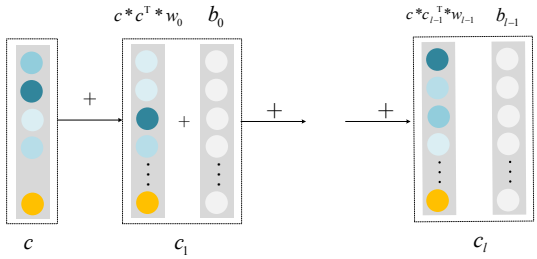

Fig. 7 The process of cross network interaction.

$$\begin{aligned} c_1 &= cc^T W_0 + b_0 + c \\ c_2 &= cc_1^T W_1 + b_1 + c_1 \\ &\cdots \\ c_l &= cc_{l-1}^T W_{l-1} + b_{l-1} + c_{l-1} \end{aligned} \tag{5}$$

Here, W_{l-1} and b_{l-1} are the weights and bias of layer l, respectively.

3.5 Output layer

We combine the linear part, the output of the cross-network, and the output of SENet as the input of the output layer. The formula for the output layer is as follows:

$$\hat{y} = \sigma(W_{linear} x_{linear} + [c_l, Q] W_o + b_o) \tag{6}$$

Here, W_o and b_o are the weights and bias terms of the output layer, respectively. We train the model by minimizing the log loss:

$$L(y, \hat{y}) = -y \log(\hat{y}) - (1-y) \log(1-\hat{y}) \tag{7}$$

Where $y \in \{0,1\}$ represents whether the user has clicked or not.

4 Experiment

4.1 Experimental setup

4.1.1 Datasets

We used two datasets to evaluate our model, as described below.

- Avazu dataset: Avazu is an online advertising company. The dataset comes from the CTR prediction dataset of the Kaggle data mining competition. The purpose is to predict whether users will click on the advertisement, and the dataset can be found on the link- https://www.kaggle.com/c/Avazu-ctr-prediction/data.We have intercepted the first one million data points, and the training set and the test set were 80% and 20% of the intercepted data, respectively. There are 24 attribute columns, namely: ID, click, hour and banner_pos, site_id, site_field, site_category, app_id, app_field, app_category, device_id, device_ip, device_model, device_type, device_conn_type, C1, C14 - C21, where C1 and C14 - C21 are anonymous attribute columns.
- Criteo dataset: Criteo is a marketing technology company with its reach across the globe. The dataset from the Kaggle shows advertising challenge, which aims to predict the click-through rate of ads, and can be found on https://www.kaggle.com/c/Criteo-display-ad-challenge/Download.The label attribute in the data indicates whether the advertisement has been clicked or not. I1-I13 are the numerical features, and C1-C26 are the category features. We have intercepted the first one million data points, and the training and the test sets were 80% and 20% of the intercepted data, respectively.

4.1.2 Evaluation Metrics

We used Log loss and AUC as evaluation indicators, described as:

- Log loss: It is often used in the offline evaluation in which the convergence of the model can be observed. According to this indicator, the smaller the loss value, the better is the model effect.
- AUC: It can be used to evaluate the ranking model of the recommendation system, in which the higher the AUC, the better is the model effect.

4.1.3 Implementation details

In our experiment, the parameters are set as follows: The embedding dimension is 2, the SENet dimension reduction ratio is 3, the optimizer is Adam, and the batch size is 256.

For the Avazu dataset, the number of cross-layers considered is 4. For the Criteo dataset, the number of cross-levels considered is 3.

All the methods mentioned above are implemented on Intel i76700 3.4GHz CPU and 16GB RAM, with Python version of 3.7 and Tensorflow version of 1.14.

4.2 Model comparison

We have compared our model to the following models: FM (Rendle (2010)), FFM (Juan et al. (2016)), DCN (Wang et al. (2017)), AutoInt (Song et al. (2019)), FiBiNET (Huang et al. (2019)), ONN (Yang et al. (2020)), and the results are shown in Table 2, Figure 8 and Figure 9. From the perspective of experimental results, our model performs better in terms of Log loss or AUC.

Table 2 Comparison of model effects

Model	Avazu		Criteo	
	Logloss	AUC	Logloss	AUC
FM	0.38003	0.75707	0.48483	0.75540
FFM	0.37683	0.76293	0.48467	0.75677
DCN	0.38303	0.75943	0.48297	0.75960
AutoInt	0.38477	0.75713	0.48753	0.75100
FiBiNET	0.38080	0.75600	0.48657	0.75217
ONN	0.37930	0.76023	0.48583	0.75407
Our	**0.37617**	**0.76463**	**0.48067**	**0.76147**

Fig. 8 The performance of the Log loss of different models on Avazu and Criteo datasets.

1. **FM:** The FM learns a latent vector for each feature and then carries out the inner product of the latent vector to achieve the second-order interaction.

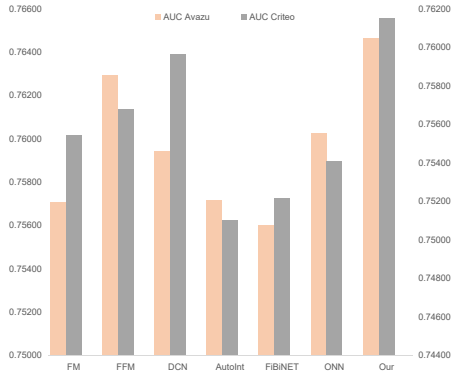

Fig. 9 The performance of the AUC of different models on Avazu and Criteo datasets.

2. **FFM:** The FFM adds the idea of the field based on the FM. It can learn a set of latent vectors for each feature. The latent vector is related not only to the feature but also to the field.
3. **DCN:** The DCN can learn feature combination of the specific order and can carry out feature crossover explicitly.
4. **AutoInt:** The AutoInt uses the multi-head attention mechanism to model feature interactions explicitly.
5. **FiBiNET:** FiBiNET refers to the SENet structure, which highlights the importance of dynamic learning features and the use of a bilinear function to enhance the model cross features.
6. **ONN:** The operation aware embedding method of ONN can learn different representations of different operations instead of sharing an embedding structure among all operations.

4.3 Study of the parameters

In this section, we study the influence of different hyperparameters on the results of the model. We have studied the following hyperparameters on the Avazu dataset. The comparison results are shown in Table 3:

4.3.1 Embedding dimension

We have performed experiments in the range of 212. From Figure 10 and Figure 11, we can see that the model works best when the embedding dimension is 2, in both the Avazu and the Ariteo datasets. Moreover, it is observed that the larger the embedding dimension is, the more complex is the model, which may lead to overfitting.

Table 3 The influence of different hyperparameters on the model results

Parameter		Avazu		Criteo	
		Logloss	AUC	Logloss	AUC
Embedding dimension	2	**0.37617**	**0.76463**	**0.48067**	**0.76147**
	4	0.37663	0.76183	0.48137	0.75960
	8	0.37895	0.76185	0.48240	0.75650
	12	0.37790	0.76125	0.48415	0.75735
Cross layers	2	0.37840	0.76177	0.48210	0.75933
	3	0.37670	0.76290	**0.48067**	**0.76147**
	4	**0.37617**	**0.76463**	0.48170	0.75877
	5	0.37823	0.76170	0.48267	0.75867
Reduction ratio	2	0.37630	0.76177	0.48095	0.75920
	3	**0.37617**	**0.76463**	**0.48067**	**0.76147**
	4	0.37630	0.76420	0.48090	0.76085
	5	0.37660	0.76190	0.48280	0.75898

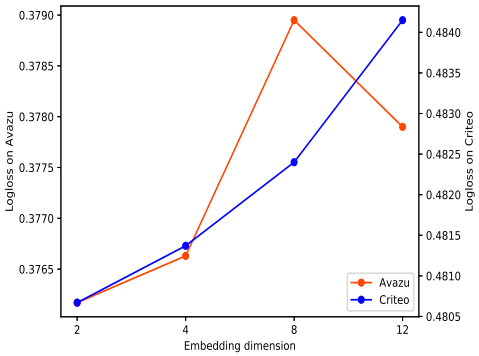

Fig. 10 The Log loss performance in different embedding dimensions.

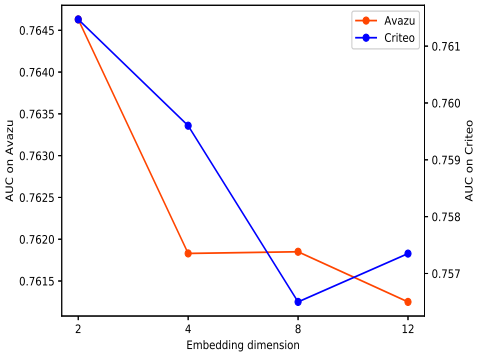

Fig. 11 The AUC performance in different embedding dimensions.

4.3.2 Cross layers

We have performed experiments in the range of 25 cross-layers. From Figure 12 and Figure 13, we can see that in the Avazu dataset, when the number of cross-layers changes from 2 to 4, the effect of the model is improved steadily. In

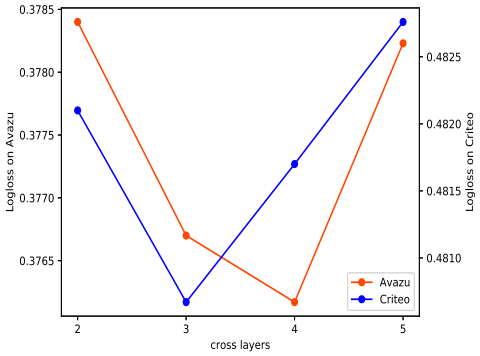

Fig. 12 The Log loss performance in different cross layers.

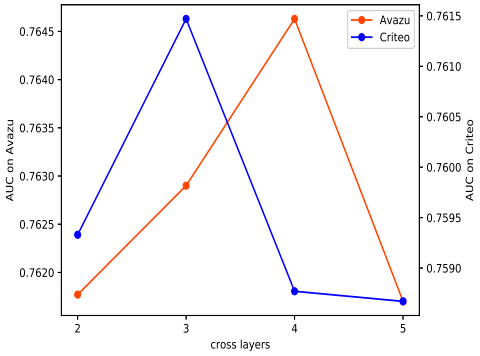

Fig. 13 The AUC performance at different cross layers.

the Criteo dataset, when the number of cross layers changes from 2 to 3, the model shows a significant improvement. However, when the number of layers is too small, the model training is not enough, and the effect cannot be achieved. When the number of cross-layers changes from 4 to 5, the effect will not be improved due to overfitting.

4.3.3 Reduction ratio

We performed experiments with the dimension reduction ratio between 2 to 5. From Figure 14 and Figure 15, we can see that, in the Criteo dataset, the model achieves the best effect when the dimension reduction ratio is 3 and 4, but when the dimension reduction ratio is 5, the model does not perform very well. The reason can be stated as the dimension is reduced too much, and the learning ability of the model is not enough. In the Avazu dataset, the model has a good effect when the dimension reduction ratio is 3 and 4.

After the comparative experiments on the above parameters, it is found that each parameter can find a parameter with the best effect, which requires a lot of experiments to be found. We also found that too small or too large number of parameters could not achieve the best effect because a too small number of parameters may lead to underfitting, and a too large number of parameters will lead to overfitting.

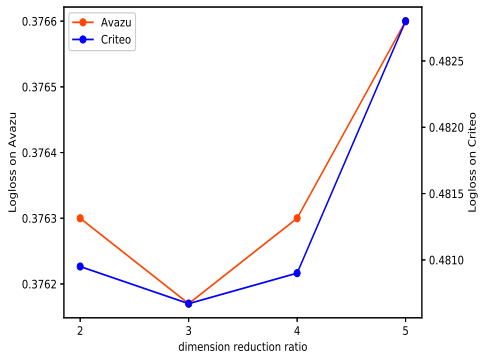

Fig. 14 The influence of different dimension reduction ratios on the Log loss of the model.

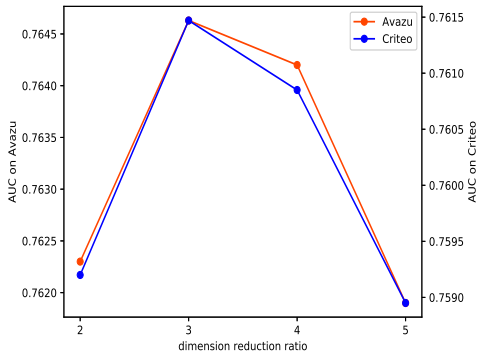

Fig. 15 The influence of different dimension reduction ratios on the AUC of the model.

5 Conclusion

In this paper, we have proposed a novel model to enhance the ability of feature interactions and improve the CTR of advertising. In our model, different latent vectors are used when the same feature interacts with other features, and the latent vectors of each feature are added to form the embedding vector of the whole feature. Then, the explicit high-order interaction is carried out in the proposed way to explore the different interactions between the features. Additionally, we use SENet to give different weights to different feature interactions. The experimental results of the Log loss and the AUC on Avazu and Criteo are found to be better than the existing advanced models.

References

Cheng HT, Koc L, Harmsen J, Shaked T, Chandra T, Aradhye H, Anderson G, Corrado G, Chai W, Ispir M, et al. (2016) Wide & deep learning for recommender systems. In: Proceedings of the 1st workshop on deep learning for recommender systems, pp 7–10

Feng Y, Lv F, Shen W, Wang M, Sun F, Zhu Y, Yang K (2019) Deep session interest network for click-through rate prediction. arXiv preprint arXiv:190506482

Guo H, Tang R, Ye Y, Li Z, He X (2017) Deepfm: a factorization-machine based neural network for ctr prediction. arXiv preprint arXiv:170304247

He X, Chua TS (2017) Neural factorization machines for sparse predictive analytics. In: Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval, pp 355–364

Hu J, Shen L, Sun G (2018) Squeeze-and-excitation networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 7132–7141

Huang T, Zhang Z, Zhang J (2019) Fibinet: combining feature importance and bilinear feature interaction for click-through rate prediction. In: Proceedings of the 13th ACM Conference on Recommender Systems, pp 169–177

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, pp 43–50

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, pp 1754–1763

Liu B, Zhu C, Li G, Zhang W, Lai J, Tang R, He X, Li Z, Yu Y (2020) Autofis: Automatic feature interaction selection in factorization models for click-through rate prediction. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp 2636–2645

Lyu Z, Dong Y, Huo C, Ren W (2020) Deep match to rank model for personalized click-through rate prediction. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol 34, pp 156–163

Ouyang W, Zhang X, Li L, Zou H, Xing X, Liu Z, Du Y (2019a) Deep spatio-temporal neural networks for click-through rate prediction. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp 2078–2086

Ouyang W, Zhang X, Ren S, Li L, Liu Z, Du Y (2019b) Click-through rate prediction with the user memory network. In: Proceedings of the 1st International Workshop on Deep Learning Practice for High-Dimensional Sparse Data, pp 1–4

Pasricha R, McAuley J (2018) Translation-based factorization machines for sequential recommendation. In: Proceedings of the 12th ACM Conference on Recommender Systems, pp 63–71

Pi Q, Bian W, Zhou G, Zhu X, Gai K (2019) Practice on long sequential user behavior modeling for click-through rate prediction. In: Proceedings of the 25th ACM SIGKDD In-

ternational Conference on Knowledge Discovery & Data Mining, pp 2671–2679

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 International Conference on Data Mining (ICDM), IEEE, pp 1149–1154

Rendle S (2010) Factorization machines. In: 2010 IEEE International conference on data mining, IEEE, pp 995–1000

Song W, Shi C, Xiao Z, Duan Z, Xu Y, Zhang M, Tang J (2019) Autoint: Automatic feature interaction learning via self-attentive neural networks. In: Proceedings of the 28th ACM International Conference on Information and Knowledge Management, pp 1161–1170

Wang R, Fu B, Fu G, Wang M (2017) Deep & cross network for ad click predictions. In: Proceedings of the ADKDD'17, pp 1–7

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. arXiv preprint arXiv:170804617

Xu W, He H, Tan M, Li Y, Lang J, Guo D (2020) Deep interest with hierarchical attention network for click-through rate prediction. In: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, pp 1905–1908

Yang Y, Xu B, Shen S, Shen F, Zhao J (2020) Operation-aware neural networks for user response prediction. Neural Networks 121:161–168

Zhang W, Du T, Wang J (2016) Deep learning over multi-field categorical data. In: European conference on information retrieval, Springer, pp 45–57

Zhou C, Bai J, Song J, Liu X, Zhao Z, Chen X, Gao J (2018a) Atrank: An attention-based user behavior modeling framework for recommendation. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol 32

Zhou G, Zhu X, Song C, Fan Y, Zhu H, Ma X, Yan Y, Jin J, Li H, Gai K (2018b) Deep interest network for click-through rate prediction. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp 1059–1068

Zhou G, Mou N, Fan Y, Pi Q, Bian W, Zhou C, Zhu X, Gai K (2019) Deep interest evolution network for click-through rate prediction. In: Proceedings of the AAAI conference on artificial intelligence, vol 33, pp 5941–5948