MsFcNET: Multi-scale Feature-Crossing Attention Network for Multi-field Sparse Data

Zhifeng Xie[1,2(✉)], Wenling Zhang[1,2], Huiming Ding[1,2], and Lizhuang Ma[2,3]

[1] Department of Film and Television Engineering, Shanghai University, Shanghai, China
{zhifeng_xie,Wxid7180a,huiming_shu}@shu.edu.cn
[2] Shanghai Engineering Research Center of Motion Picture Special Effects, Shanghai, China
ma-lz@cs.sjtu.edu.cn
[3] Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China

Abstract. Feature engineering usually needs to excavate dense-and-implicit cross features from multi-filed sparse data. Recently, many state-of-the-art models have been proposed to achieve low-order and high-order feature interactions. However, most of them ignore the importance of cross features and fail to suppress the negative impact of useless features. In this paper, a novel multi-scale feature-crossing attention network (MsFcNET) is proposed to extract dense-and-implicit cross features and learn their importance in the different scales. The model adopts the DIA-LSTM units to construct a new attention calibration architecture, which can adaptively adjust the weights of features in the process of feature interactions. On the other hand, it also integrates a multi-scale feature-crossing module to strengthen the representation ability of cross features from multi-field sparse data. The extensive experimental results on three real-world prediction datasets demonstrate that our proposed model yields superior performance compared with the other state-of-the-art models.

Keywords: Feature engineering · Feature interactions · Attention network · Factorization machines

1 Introduction

Feature engineering is the process of transforming the original data into features, which can better describe the potential characteristics of data, so as to further improve the accuracy of the predictive model. It has been considered to be a central task in a variety of machine learning applications, such as recommendation system, computational advertising, search ranking and so on. Unfortunately, multi-field sparse data often influence the effect of feature engineering because

H. W. Lauw et al. (Eds.): PAKDD 2020, LNAI 12084, pp. 142–154, 2020.
https://doi.org/10.1007/978-3-030-47426-3_12

it is very difficult to excavate the dense-and-implicit cross features among the different fields. Therefore, this paper mainly focuses on how to effectively extract and represent the cross features from high-dimensional incomplete data, in order to achieve higher-quality feature engineering and yield higher-accuracy predictive model.

In decade, a number of state-of-the-art models have been proposed to achieve the feature interactions of multi-field sparse data. FM (Factorization Machines) [11] and FFM (Field-aware FM) [6] use matrix factorization to finish low-order feature interactions. But these low-order operations can not produce valuable cross features, so they often fail to obtain higher accuracy for many complex prediction tasks. Recently, with the development of deep learning, Deep Neural Networks (DNN) are successfully applied into feature engineering, such as NFM (Neural FM) [4], Deep Crossing [12], Wide&Deep [1], DeepFM [2], xDeepFM [7], DIN [19], FNFM (Field-aware NFM) [18], AutoInt [13], and so on. But these DNN-based models still lack the powerful ability of extracting higher-dimensional cross features, especially the interaction of useless features may introduce noise and have a negative impact on the predictive model. In brief, for high-order feature interactions, the traditional methods need to be further improved.

Inspired by Attention Mechanism [5,15], we propose multi-scale feature-crossing attention network (MsFcNET) to significantly improve the quality of feature engineering in this paper. As shown in Fig. 1, our new network mainly contains six parts: input layer, embedding layer, multi-scale feature-crossing attention layer, hidden layers, combination layer and output layer. As a core component, our new attention layer can effectively extract dense-and-implicit cross features and dynamically learn their importance in the different scales. In this layer, we design a multi-scale feature-crossing module to better represent the cross features from multi-field sparse data. On the other hand, we also adopt the DIA-LSTM (Dense-and-Implicit Attention-Long Short Term Memory) units to construct a new attention calibration architecture, which can adjust the weights of features before and during feature interaction procedure adaptively. In a word, our MsFcNET model can strengthen the ability of feature interactions while avoiding the negative cross features.

Moreover, we build a new Tobacco dataset which contains static individual information of thousands of tobacco stores, their dynamic order and sales records, and their violation cases in the past four years. The Tobacco dataset has a small amount of data, a large number of feature fields, numerous null values, and a few of anomaly data. Thus it is difficult to predict the violations of tobacco stores. We conduct extensive experiments on the tobacco dataset and two public CTR prediction datasets (Avazu and Criteo). The experimental results demonstrate that our proposed MsFcNET model obtains superior performance compared with other state-of-the-art models.

2 Related Work

In feature engineering, FM (Factorization Machines) [11] is a very successful method, which uses the implicit inner product of features to compute the coefficients matrix of interaction term between features. Since FM considers feature interactions as the factorization problem of high-dimensional sparse matrix, many new cross features and hidden vectors can be efficiently extracted and represented. Later as an improvement of the FM model, FFM (Field-aware Factorization Machines) [6] further introduces the concept of fields to achieve higher-quality feature interactions. However, the feature dimension in many practical applications is very high, so the above models focusing on low-order feature interactions are hard to perfectly capture high-dimensional cross features.

Instead, a number of deep learning techniques have been proposed to effectively handle feature interactions in recent years. NFM (Neural Factorization Machines) [4] constructs deep neural network to improve the second-order feature interactions of FM. Wide&Deep [1] jointly trains linear model and deep neural network to integrate the advantage of memorization and generalization. DeepFM [2] integrates the architectures of FM and deep neural network by sharing the feature embedding, which is an end-to-end model without any manual feature engineering. Deep&Cross [16] replaces the wide component with a novel cross network that learn certain bounded-degree feature interactions. xDeepFM [7] learns explicit and implicit high-order feature interactions and cross features at the vector-wise level. FNFM (Field-Aware Neural Factorization Machine) [18] uses the second-order feature interactions of FFM as the input of deep neural network. In summary, those models based on deep learning can reduce or even get ride of manual feature engineering, and increase the strength of interaction between features. Unfortunately, most of them fail to learn the importance of cross features, and some negative feature interactions are easy to reduce the accuracy of predictive model.

Attention network [15] is motivated by human visual attention and it selectively focuses on the key part of information while ignoring the other perceivable parts. AFM (Attentional FM) [17] first introduces neural attention network to learn the significance of second-order feature interactions of FM. AutoInt [13] models feature interactions in the low-dimensional space by casting features into multiple subspaces and capturing different feature combinations in different subspaces. DIN [19] introduces attention mechanism to adaptively learn the representation of user interests from historical behaviors. Since the attention-based models learn the feature importance and avoid the negative feature interactions, we integrate multi-scale module and attention network into a new multi-scale feature-crossing attention model, which can further improve the quality of feature engineering.

3 Multi-scale Feature-Crossing Attention Network

In this section, we will introduce the neural network architecture of multi-scale feature-crossing attention model as shown in Fig. 1. Our proposed MsFcNET model is composed of the following six parts: input layer, embedding layer, multi-scale feature-crossing attention layer, hidden layers, combination layer and output layer.

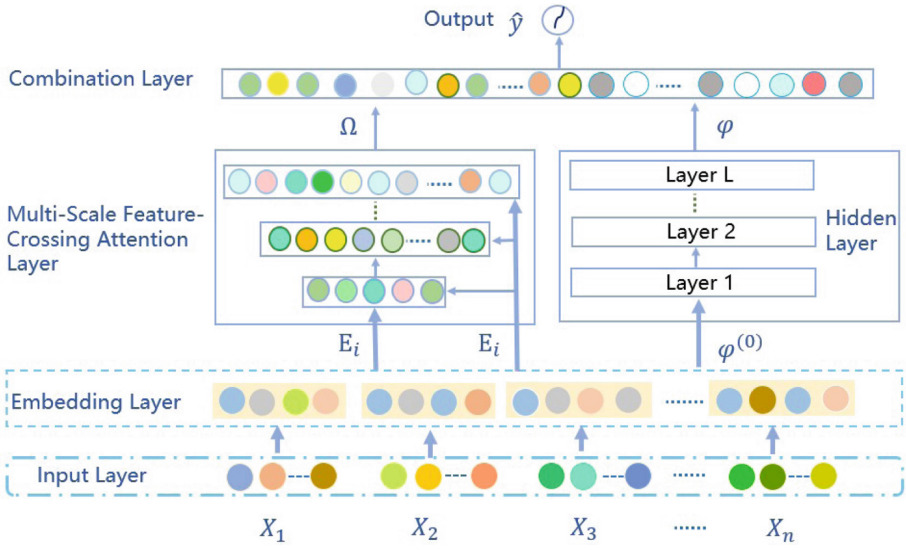

Fig. 1. The neural network architecture of proposed MsFcNET model.

3.1 Input Layer

In this layer, we first convert multi-field sparse data into initial feature vectors, which are composed of the categorical or numerical values in the different fields. The input layer is defined as followed:

$$X = [X_1, X_2, ..., X_i, ..., X_n]^T, X_i = [x_1, x_2, ..., x_t, ..., x_m] \tag{1}$$

where X is the output matrix of the input layer; X_i the feature values of the i-th record; n is the number of total records; m is the number of fields; x_t is the value of the t-th field, which can be computed according to the following rules: if the field is categorical, then x_t is the one-hot encoding value; if the field is numerical, x_t is the normalized value.

3.2 Embedding Layer

The feature vectors of categorical fields are often extreme sparse and high dimensional, which must be compressed into low-dimensional feature space. Besides, we also need to convert the dense features of numerical fields into the same low-dimensional feature space in order to formalize a unified embedding output. Thus the embedding layer can integrate the different features and reduce their dimensions, which is defined as followed:

$$E = [E_1, E_2, ..., E_t, ..., E_m]^T, E_t = V_t Y_t \tag{2}$$

where Y_t is a $n \times n$ feature matrix, which is denoted as $[X^t, X^t, ..., X^t]$; X^t is the feature vector of X in the t-th field, which is an one-hot encoding vector in the categorical field or a normalized vector in the numerical field; V_t is the embedding matrix in the t-th field; E_t is the embedding output of Y_t; E is the concatenation output of the embedding layer.

3.3 Multi-scale Feature-Crossing Attention Layer

In order to improve feature interactions, we get inspiration from DIANet [5] to construct multi-scale feature-crossing attention layer. The goal of the layer is to better extract and calibrate cross features by strengthening useful features and suppressing less useful ones. As illustrated in Fig. 2, it has two branches: attention calibration module and multi-scale feature-crossing module. The first module learns the importance of features and adjusts the feature weights dynamically; the second module extracts dense-and-implicit cross features among fields in different scales.

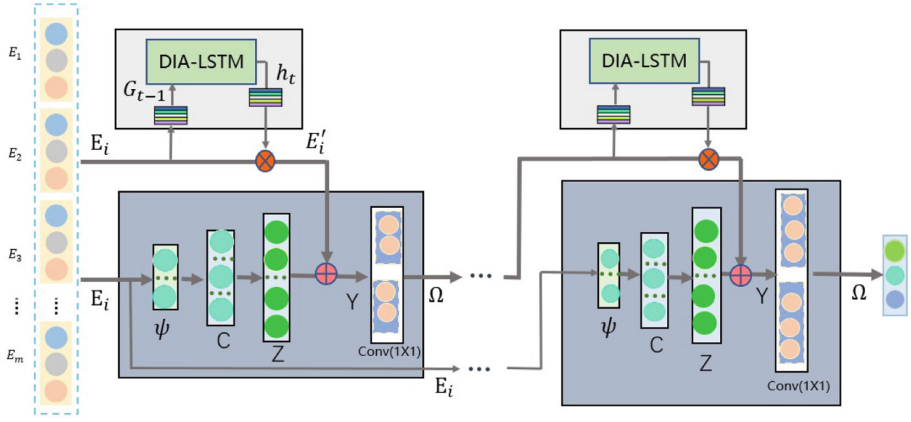

Fig. 2. The structure of multi-scale feature-crossing attention layer.

Attention Calibration Module. Lets $E_i = [e_{11}, e_{12}, .., e_{ti}, .., e_{mk}]^T$ denotes the 2-D $m \times k$ embedding matrix, where E_i represents the i-th embedding sample, $e_{ti} \in R^k$ is the i-th embedding vector of the t-th field , m is the size of fields and k is the embedding size of each embedding vector. First of all, we use mean pooling method to squeeze the embedding vectors of fields into one vector $G = [g_1, ..., g_t, ..., g_m]$ where $t \in [1, m]$. g_t represents the global information of the t-th field, which can be calculated as:

$$g_t = \frac{1}{k} \sum_{q=1}^{k} e_{tq} \tag{3}$$

Then we adopt DIA-LSTM [5] to dynamically capture feature information of different layers, and use the shared Fully Connected module (FC) to reduce the parameters in training process. The DIA-LSTM unit is shown in Fig. 3. The first FC layer is a compression layer with parameters W_1 which compresses the input dimension to the 1/4 original size and uses σ_1 as nonlinear function; The second FC layer expands the dimension to 4 times of the original input with parameter W_2. Given the input vector G_{t-1} and the random initialization of hidden unit h_{t-1}, the output F is denoted as:

$$F = W_2 \sigma_1 (W_1 [G_{t-1}, h_{t-1}]) \tag{4}$$

where σ_1 is LeakyReLU activation function. The outputs of forget gate f_t, input gate i_t, cell state $\widetilde{C_t}$, and output gate O_t are denoted as:

$$f_t = \sigma(F); i_t = \sigma(F); \widetilde{C_t} = \sigma(F); O_t = \sigma(F). \tag{5}$$

where σ denotes sigmoid function. After obtaining these four internal units, we can further get the outputs C_t and h_t of LSTM:

$$C_t = C_{t-1} f_t + i_t \widetilde{C_t} \tag{6}$$

$$h_t = tanh(C_t) O_t \tag{7}$$

where we use tanh [8] function to replace sigmoid function.

Finally, the original embedding matrix E_i is calibrated to the new embedding matrix $E_i^{'}$, which can be defined as:

$$E_i^{'} = h_t \odot E_i \tag{8}$$

where h_t is the weight vector, which represents the importance of features; $\odot$ denotes Hadamard product that is element-wise multiply.

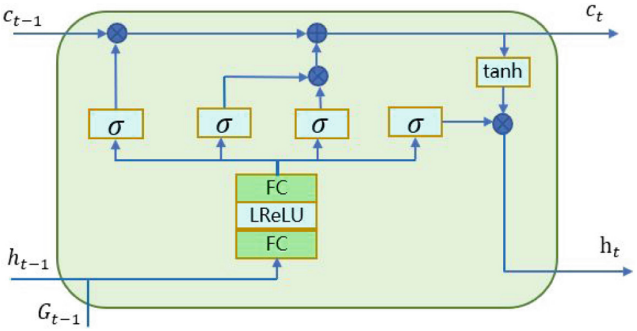

Fig. 3. The structure of DIA-LSTM unit.

Multi-scale Feature-Crossing Module. This module shares the same input with calibration module, and we use two Fully Connected layers (FC) to learn feature interactions from different fields. The first FC layer is a field dimension reduction layer with parameter U_1 and b_1 with reduction ration r that is a hyper-parameter. The second FC layer is a dimensional recovery layer with parameter U_2 and b_2. The cross feature can be calculated as follows:

$$C = U_2\psi + b_2; \psi = U_1 E_i + b_1 \tag{9}$$

where $C \in R^{m\times k}$ is a 2-dimensional vector matrix; $\psi \in R^{k\times \frac{m}{r}}$ is the output of compress cross features; the hyper parameters are U_1, U_2, b_1, b_2; r is the reduction ratio.

Then we use a FC layer to transform dimension, and expand the embedding dimension from low dimension to higher dimension at different scales. The new cross feature is defined as:

$$Z = \tau C + b_3 \tag{10}$$

where $Z \in R^{m\times k^{'}}$ denotes the matrix vectors in different layers; $k^{'}$ equals k in the first layer; τ and b_3 are the weight matrix and bias vectors in different layers.

Finally, in order to reinforce feature interactions, the cross features and calibration module can be fused as:

$$Y_i = [E_{i1}^{'} \oplus Z_{i1}, E_{i2}^{'} \oplus Z_{i2}, ..., E_{im}^{'} \oplus Z_{im}] \tag{11}$$

where Y_i is a 2-dimensional $m \times k^{'}$ matrix vectors; $\oplus$ denotes the element-wise addition; $E_i^{'} \in R^{m\times k^{'}}$ is the matrix vector of the attention calibration module; $Z \in R^{m\times k^{'}}$ is the matrix vector of feature-crossing module.

1×1 convolution [3,14] has been widely applied to achieve dimensionality reduction and dimensionality upgrading. Thus we introduce the 1×1 convolution to upgrade the embedding dimension, which can be calculated as:

$$\Omega_i = conv1D(\omega Y_i) = LeakyReLU(\omega Y_i) \tag{12}$$

where $\Omega_i \in R^{m \times k'}$, k' is the expanded embedding dimension; ω is the convolution weight; the size of convolution kernel is 1×1; the number of filters are k'; the activation function is set to LeakyReLU.

3.4 Hidden Layers

The hidden layers are composed of several fully connected layers with same scale, which capture high-order and implicit feature interactions. Here, the embedding matrix includes different features which have various influence for the models, so we add a matrix parameter to adjust the embedding matrix. The input of this layer $\varphi^{(0)}$ is defined as:

$$\varphi^{(0)} = \zeta \odot E_i \tag{13}$$

where ζ denotes the matrix parameter; E_i is the embedding matrix.

Then, $\varphi^{(0)}$ is entered into the deep network and the forward process is defined as:

$$\varphi^{(l)} = \xi(W^{(l)}\varphi^{(0)} + \beta^{(l)}) \tag{14}$$

where l is the depth; ξ is the LeakyReLU function; $\varphi^{(l)}$ is the output of the l-th hidden layer; $w^{(l)}$ and $\beta^{(l)}$ is the hyper-parameter of the l-th deep layer.

3.5 Combination Layer

The combination layer concatenates interaction vector Ω and hidden vector φ and feeds the concatenated vector into a FC layer. It can be expressed as the following description:

$$\delta = F_{concat}(\Omega, \varphi) = [\Omega_1, ..., \Omega_n, \varphi_1, ..., \varphi_n]^T \tag{15}$$

where δ is the output of the combination layer; n is the number of samples.

3.6 Output Layer

We combine combination layer and linear part to make model stronger by capturing different feature interactions. The output unit is defined as:

$$\hat{y} = \sigma(\Theta\delta + \sum_{i=0}^{n} \beta_i E_i + b) \tag{16}$$

where $\hat{y} \in (0, 1)$ is the predicted result of model; σ is sigmoid function; δ is the output of combination layer; n is the number of samples; E_i is the embedding vectors and β_i is the i-th weight of linear part; Θ and b are the weight and bias.

We introduce the Log loss as the loss function as the objective function, which is expressed as:

$$loss = -\frac{1}{N}\sum_{i=1}^{N}(y_i log(\hat{y}_i) + (1 - y_i) log(1 - \hat{y}_i)) \tag{17}$$

where N is the size of the training samples; y_i is the ground truth of the i-th training instance; $\hat{y}_i$ is the final output of the network.

4 Experiments

In this section, we will further evaluate our proposed model by replying the following questions:

RQ 1 How does our model perform compared with the state-of-the-art methods on multi-field sparse data? Is it efficient for CTR problems?
RQ 2 How do the implicit feature interactions affect performance?
RQ 3 How do the different configurations influence the performance of our model?

4.1 Experiment Setup

Datasets. Besides two public CTR evaluation datasets (Avazu[1] and Criteo[2]), we constuct a new Tobacco dataset with a small amount of data, a large number of feature fields, numerous null values, and a few of anomaly data. Their statistics are illustrated in Table 1. All of datasets are divided into 80% samples for training and 20% remaining ones for testing.

Table 1. Statistics of three evaluation datasets.

Dataset	Instances	Fields	Features (Sparse)
Tobacco	166,626	247	146,103
Avazu	40,428,967	23	1,544,488
Criteo	45,840,617	39	998,960

Evaluation Metrics. We use AUC and Logloss as our evaluation metrics. AUC is not sensitive to whether the samples are balanced and it reflects the sorting ability of the samples. Logloss measures the distance between two distributions and the smaller value indicates the better performance.

Model Comparison. We compare our proposed model with three classes of the traditional models: (i) shallow models, including LR [9], FM [11], AFM [17]. (ii) deep models, including Wide&Deep [1], Deep&Cross [16], DeepFM [2], xDeepFM [7]. (iii) high-order models, including NFM [4], PNN [10], CrossNet [16], CIN [7], AutoInt [13].

[1] Avazu:http://www.kaggle.com/c/avazu-ctr-prediction.
[2] Criteo:https://www.kaggle.com/c/criteo-display-ad-challenge.

Implementation Details. We implement all models with tensorflow. The embedding size is set to 40, 16 for Tobacco and two public datasets respectively. We use Adam as optimization method for all methods with the batch size of 256 for Tobacco and 1024 for other datasets. For Tobacco dataset, the depth of hidden layers is set to 4, the number of neurons per layer is 128, For Avazu and Criteo datasets, we use the same parameters with AutoInt [13] for baseline methods. We use three interaction layers for CrossNet and CIN. The hidden layer size of NFM is set to 200 which is recommended in the paper [4].

4.2 Effectiveness Comparison (RQ1)

Evaluation of Performance. We summarize the performance of different models on three datasets in Table 2. Compared with the shallow models, the deep models have a better performance on all datasets because cross features from deep models yield the higher predictive power. On the other hand, our MsFcNET model achieves an excellent performance over the three datasets. The results indicate that the feature interactions of our model are very effective on multi-field sparse datasets.

Table 2. The overall performance of different models on Tobacco, Avazu and Criteo datasets.

		Tobacco		Avazu		Criteo	
Model Class	Model	AUC	Logloss	AUC	Logloss	AUC	Logloss
Shallow model	LR	0.7240	0.4495	0.7560	0.3964	0.7820	0.4695
	FM	0.7883	0.4116	0.7706	0.3856	0.7836	0.4700
	AFM	0.7901	0.4102	0.7718	0.3854	0.7938	0.4584
deep model	Wide&Deep	0.8565	0.3657	0.7749	0.3824	0.8026	0.4494
	DeepFM	0.8655	0.3611	0.7751	0.3829	0.8066	0.4449
	Deep&Cross	0.8697	0.3588	0.7731	0.3836	0.8067	0.4447
	xDeepFM	0.8840	0.3403	**0.7768**	0.3823	0.8070	0.4447
	MsFcNET	**0.8872**	**0.3330**	0.7766	**0.3819**	**0.8081**	**0.4406**

Evaluation of Efficiency. We compare the runtime of different models on three datasets. Since most of shallow models have the simpler implementation, they are more efficient than the deep models. In the deep models, xDeepFM has an excellent performance, but due to the complexity of cross-layer computing, xDeepFM has much more time consumption. The runtimes for each epoch are 250s, 353100 s, 80145 s on three datasets respectively. MsFcNET's runtimes for each epoch are 171s, 78540 s, 20865 s on three datasets respectively, which has a great improvement in time consumption compared with xDeepFM.

4.3 Performance Comparison Without Deep Module (RQ2)

Table 3. The performance of different high-order models on three datasets.

	Tobacco		Avazu		Criteo	
Model	AUC	Logloss	AUC	Logloss	AUC	Logloss
NFM	0.7923	0.4089	0.7708	0.3864	0.7957	0.4562
CrossNet	0.8624	0.3590	0.7667	0.3868	0.7907	0.4591
PNN	0.8684	0.3520	0.7743	0.3834	0.7973	0.4523
CIN	0.8808	0.3396	0.7758	0.3829	0.8009	0.4517
AutoInt	0.8370	0.3952	0.7752	0.3824	0.8061	0.4455
MsFcNET-	**0.8863**	**0.3352**	**0.7764**	**0.3822**	**0.8068**	**0.4420**

Comparison with High-Order Models. Deep module improves implicit feature interactions and has been widely adopted in predictive models. Here, all of high-order models exclude the deep network layers. For a fair comparison, our MsFcNET model also gets rid of the part of the deep hidden layer, called as MsFcNET-. Table 3 shows the performance of high-order models on three datasets. Our MsFcNET model without deep module still has the outstanding performance on these datasets, which demonstrates the effectiveness of high-order feature interactions in our proposed model.

4.4 Hyper-parameter Analysis (RQ3)

Table 4. The performance of different embedding sizes on three datasets.

	Tobacco		Avazu		Criteo	
Embedding size	AUC	Logloss	AUC	Logloss	AUC	Logloss
10	0.8774	0.3491	0.7761	0.3821	**0.8083**	**0.4403**
20 (16)	0.8790	0.3491	0.7766	0.3819	0.8081	0.4406
30	0.8860	0.3378	0.7767	0.3816	0.8078	0.4412
40	**0.8872**	**0.3330**	**0.7770**	**0.3816**	0.8072	0.4416
50	0.8867	0.3357	0.7765	0.3822	0.8066	0.4424

Embedding Part. We analyze the effects of the embedding size from 10 to 50. As illustrated in Table 4, for Tobacco and Avazu datasets, when the embedding size is set to 40, our model can yield the best performance; for Criteo dataset, there is the best performance when the size is 10. Obviously, the appropriate embedding size can extract more valuable features while avoiding the difficult optimization of too many parameters.

Multi-scale Feature-Crossing Part. We compare the model performance in the number of multi-scale feature-crossing layers from 1 to 5. When the network depth increases from 1 to 3 on Avazu dataset, the AUC increases from 0.7745 to 0.7768 and Logloss decreases from 0.3830 to 0.3823.

However, when a number of layers continues to increase, the performance begins to decrease. Finally, the AUC decreases to 0.7756 and Logloss increases to 0.3828 on Avazu dataset. This is because too complicated models can easily lead to over-fitting. For Avazu dataset, it is more appropriate to set the layer number to 3. Furthermore, the size of reduction ratio can also lead to more complex models. We change the size of reduction ratio from 1 to 5 and get the similar results with the change of layer number. When we set 3 for Avazu dataset as the reduction ratio, our model gets the best performance.

5 Conclusion

In this paper, we propose a novel network named MsFcNET, which can not only dynamically adjust the weights of features, but also efficiently reinforce the extraction of cross features in the different scales. Our proposed model constructs a new attention network based on DIA-LSTM unit, which can learn the importance of features in the process of feature interactions. Moreover, our model also designs a multi-scale feature-crossing module to better extract and represent complex cross features. Experimental results on three real-world datasets demonstrate that our MsFcNET model can yield better performance than the state-of-the-art deep and shallow models.

References

1. Cheng, H.T., Koc, L., Harmsen, J., et al.: Wide & deep learning for recommender systems. In: The 1st Workshop on Recommender Systems, pp. 7–10 (2016)
2. Guo, H., Tang, R., Ye, Y., et al.: DeepFM: a factorization-machine based neural network for CTR prediction. arXiv preprint arXiv:1703.04247 (2017)
3. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the CVPR, pp. 770–778 (2016)
4. He, X., Chua, T.S.: Neural factorization machines for sparse predictive analytics. In: Proceedings of the 40th International ACM SIGIR, pp. 355–364 (2017)
5. Huang, Z., Liang, S., Liang, M., et al.: Dianet: dense-and-implicit attention network. arXiv preprint arXiv:1905.10671 (2019)
6. Juan, Y., Zhuang, Y., Chin, W.S., et al.: Field-aware factorization machines for CTR prediction. In: Proceedings of the 10th ACM. pp. 43–50. ACM (2016)
7. Lian, J., Zhou, X., Zhang, F., et al.: xDeepFM: combining explicit and implicit feature interactions for recommender systems. In: The 24th ACM SIGKDD, pp. 1754–1763. ACM (2018)
8. Malfliet, W., Hereman, W.: The tanh method: I. exact solutions of nonlinear evolution and wave equations. Physica Scripta **54**(6), 563 (1996)
9. McMahan, H.B., Holt, G., Sculley, D., et al.: Ad click prediction: a view from the trenches. In: The 19th ACM SIGKDD, pp. 1222–1230 (2013)

10. Qu, Y., Fang, B., Zhang, W., et al.: Product-based neural networks for user response prediction over multi-field categorical data. ACM TOIS **37**(1), 1–35 (2018)
11. Rendle, S., Gantner, Z., et al.: Fast context-aware recommendations with factorization machines. In: The 34th ACM SIGIR, pp. 635–644. ACM (2011)
12. Shan, Y., Hoens, T.R., Jiao, J., et al.: Deep crossing: Web-scale modeling without manually crafted combinatorial features. In: The 22th ACM SIGKDD, pp. 255–262 (2016)
13. Song, W., Shi, C., Xiao, Z., et al.: Autoint: automatic feature interaction learning via self-attentive neural networks. In: The 28th ACM CIKM, pp. 1161–1170 (2019)
14. Szegedy, C., Liu, W., Jia, Y., et al.: Going deeper with convolutions. In: Proceedings of the CVPR, pp. 1–9 (2015)
15. Wang, F., Jiang, M., Qian, C., et al.: Residual attention network for image classification. In: Proceedings of the IEEE CVPR, pp. 3156–3164 (2017)
16. Wang, R., Fu, B., Fu, G., et al.: Deep & cross network for ad click predictions. In: Proceedings of the ADKDD 2017, p. 12. ACM (2017)
17. Xiao, J., Ye, H., He, X., et al.: Attentional factorization machines: learning the weight of feature interactions via attention networks. arXiv preprint arXiv:1708.04617 (2017)
18. Zhang, L., Shen, W., Huang, J., et al.: Field-aware neural factorization machine for click-through rate prediction. IEEE Access **7**, 75032–75040 (2019)
19. Zhou, G., Zhu, X., Song, C., et al.: Deep interest network for click-through rate prediction. In: Proceedings of the 24th ACM SIGKDD, pp. 1059–1068. ACM (2018)