IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 29, 2019, accepted May 17, 2019, date of publication June 5, 2019, date of current version June 20, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2921026

Field-Aware Neural Factorization Machine for Click-Through Rate Prediction

LI ZHANG, WEICHEN SHEN, JIANHANG HUANG, SHIJIAN LI, AND GANG PAN
Department of Computer Science, Zhejiang University, Hangzhou 310027, China

Corresponding author: Shijian Li (shijianli@zju.edu.cn)

This work was supported by the National Key Research and Development Plan under Grant 2016YFB1001203.

ABSTRACT Recommendation systems and computing advertisements are of great value for commercial applications. Click-through rate (CTR) prediction is a critical issue because the prediction accuracy affects the user experience and the revenue of merchants and platforms. Feature engineering is usually used to improve the click-through rate prediction; however, it heavily relies on user experience. It is difficult to construct a feature combination that can describe the complex patterns implied in the data. This paper combines the traditional feature combination methods and the deep neural networks to automate the feature combinations to improve the accuracy of the click-through rate prediction. We propose a mechanism named Field-aware Neural Factorization Machine (FNFM). This paper can have strong second-order feature interactive learning ability, such as Field-aware Factorization Machine; on this basis, a deep neural network is used for higher order feature combination learning. This experiment shows that the model has stronger expression ability than previous deep learning feature combination models, such as the DeepFM, DCN, and NFM.

INDEX TERMS Big data applications, learning systems, neural networks, pattern analysis, predictive models.

I. INTRODUCTION

A recommendation system was developed to address the demands of users and businesses in the Internet scene. According to report data, recommendation systems brought great sales revenue and browsing traffic to the Amazon, Netflix and Youtube from personalized recommendation traffic [1]. Therefore, building accurate and effective recommendation systems is of great significance for improving user experience and company revenue.

In a recommendation system, a crucial task is to predict the probability of a user clicking on a recommended item. Therefore, click-through rate (CTR) prediction is a core issue for recommendation systems [2], [3]. In many recommendation systems, the goal is to maximize the number of clicks, so recommended items can be ranked by estimated clickthrough rate. In addition, in online advertising systems [4], click-through rate prediction is also very important to improve system revenue, because the ad's sorting strategy can be adjusted by clickthrough rate and bidding.

Features play a central role in the success of many predictive systems. In advertisement recommendation techniques, the features commonly include but are not limited to the ad information, user device information, user behaviors or site information [5]. Different features present specific information from various aspects and dimensions, and cross-combination between features is often very meaningful [6]. Traditional cross-over features have three major drawbacks [7]: First, obtaining high quality features requires high cost. Since effective feature combinations are often generated based on specific task scenarios, engineers need to spend a lot of time manually designing cross-combination features, and artificial feature engineering relies heavily on engineers' prior knowledge and business sensitivity, which has great limitations. Second, in large-scale prediction systems such as recommendation systems, a large number of original features make manual extraction of all cross-features infeasible. Finally, the artificially constructed cross-combination features cannot be overlaid onto the combined patterns in the training data that have occurred. The deep learning technology is a promising way to solve the problems, since it has advantages to handle inner structures inside high-dimensional sparse data scenarios [8], using deep learning technology to improve the feature interaction ability of predictive models is a meaningful research task.

The associate editor coordinating the review of this manuscript and approving it for publication was Haluk Eren.

Several works were done for automatic feature engineering with deep neural networks, like the Deep Factorization Machine (DeepFM) [9], Neural Factorization Machine (NFM) [10] and Deep Cross Networks (DCN) [11]. Inspired by these works, we considered more about further reduction of information loss and confusion in feature combination, and proposed our factorization model named 'Field-aware Neural Factorization Machine' (FNFM). FNFM embeds information in units of 'fields' and use field vector to present the input information, so it can retain more information in second and higher order feature interactive learning. Experiments show FNFM has better information expressive ability compared with previous models.

II. RELATED WORK

Traditional linear models capture second-order feature combinations by means of degree-2 polynomial features. Data sparsity widely exists in actual business scenarios. For some combination features, there are very few cases where the two features are not zero at the same time. When any feature in the combined feature takes a value of 0, then the interaction term coefficients of the combination with this feature cannot be effectively learned.

The factorization machine(FM) [12] uses the idea of matrix decomposition [13] to obtain the matrix of interaction term coefficients between features by the implicit inner product of the feature. This method remedies the shortcomings of the second-order polynomial method that the amount of parameters that need to be learned is too large and cannot effectively process the coefficient data [12].

The FFM(Field-aware Factorization Machine) [14] model is an improvement of the FM model. The FFM model introduces the concept of field, that is, using different hidden vectors presenting different feature groups. When calculating the weight of the interaction term between each pair of features, the traditional FM model is represented by the inner product of the hidden vectors corresponding to the two features.

In the FFM, each feature x_i in feature group i, for features x_j in other feature group j, FFM learns a pair of hidden vectors v_{i,f_j}, v_{j,f_i}. It divides it into multiple fields according to the meaning of the feature, and each feature belongs to a specific field. Each feature has multiple hidden vectors, one for each field. When two features are combined, the inner product of the field corresponding to the two features is used as the inner product, so the model equation of FFM is:

$$\hat{y} = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} < v_{i,f_j}, v_{j,f_i} > x_i x_j \qquad (1)$$

where f_i and f_j are the fields to which the ith and jth features belong, respectively. If there are a total of f fields, then the parameter quantity of the FFM model is $n\ f\ k$, and the computation time complexity is $O(\bar{n}^2 k)$. It is worth noting that in FFM, each hidden vector only needs to learn the effect of interacting with a specific field vector,

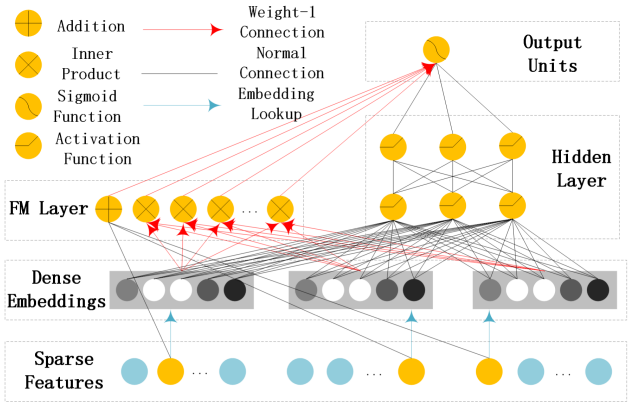

FIGURE 1. DeepFM model sturucure.

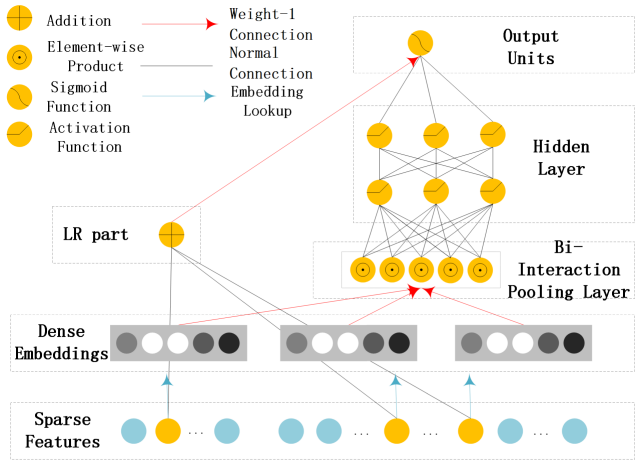

FIGURE 2. NFM model sturucure.

DeepFM [9] is a model that combines FM and Deep Neural Network(DNN) to model low-order feature combinations like FM and model high-order feature combinations like DNN(Figure1). Unlike the "Wide & Deep Learning" method (WDL) [7], DeepFM can perform end-to-end training without any feature engineering because its wide side and deep side share the same input and embedding vectors. The model structure is as follows:

DeepFM consists of two components that share the same input FM component and DNN component. For feature x_i, a scalar w_i is used as its 1-order weight, and a hidden vector V_i is used as an influence factor for its interaction with other features. V_i is input into the FM component to model the 2-order feature interaction, while inputting into the DNN component to model higher-order feature interactions. All parameters are trained by joint prediction models:

$$\hat{y} = sigmoid(y_{FM} + y_{DNN}) \qquad (2)$$

The NFM (Neural FM, Figure2) [10] model uses both FM and neural networks to model sparse data.

$$\hat{y} = w_0 + \sum_{i=1}^{n} w_i x_i + f_{BI}(x) \qquad (3)$$

In the equation, the first and second items are similar to the linear regression items in the FM model. The third item $f_{BI}(x)$ is the core component used by the NFM model to model feature interactions. NFM first incorporates it into the vector input to a second-order interactive pooling layer, which is capable of pooling several embedding vectors into a vector:

$$f_{BI}(V_x) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} x_i v_i \odot x_j v_j \tag{4}$$

where $\odot$ represents the element-wise multiplication between two vectors. The output of the second-order interaction layer is a k-dimensional vector that encodes the second-order interaction between features into the embedding space.

Deep and Cross Network (DCN) [11] adopts a cross-network structure to explicitly calculate the cross-combination between features. The cross network consists of different intersecting layers. This special structure enables the order of the interactive features to increase as the number of layers increases. The highest order (as compared to the original input) that a L layered cross network can capture is $L + 1$.

Deep networks get DCN the ability of higher ordered interaction, but because the interaction between features are based on feature elements which may confuse the information between features [14]. We consider to combine field-based FFM and deep neural network to generate an expressive factorization model.

III. FIELD-AWARE NEURAL FACTORIZATION MACHINE

We propose Field-aware Neural Factorization Machine (FNFM), a click-through prediction model which obtains the advantages of FFM in second order feature interaction and improves NFM's ability on higher order feature interaction. The model of FNFM is described as:

$$\widehat{y}_{FNFM}(x) = w_o + \sum_{1}^{n} w_i x_i + DNN(f_{BI}(V_x)) \tag{5}$$

The first two items is same regression function as FFM, and the last item in equation $DNN(f_{BI}(V_x))$ is the key mechanism to handle feature interactions. The mechanism of FNFM (shown in Figure3) consists of four layers. Lower parts works like FFM, and higher is DNN with second order feature interaction vector input.

A. INPUT LAYER

We first convert the characteristics of users and advertisements into a feature vector that is spliced, from input features under different feature groups (as fields in FFM):

$$x = [x_1; x_2; \ldots; x_f] \tag{6}$$

where f is the number of feature groups. x_i is the feature of t-th feature group. If it is a sparse category feature, then x_i is a one hot encoding vector. If the tth feature group is a dense numerical feature group, then the x_i is a scalar.

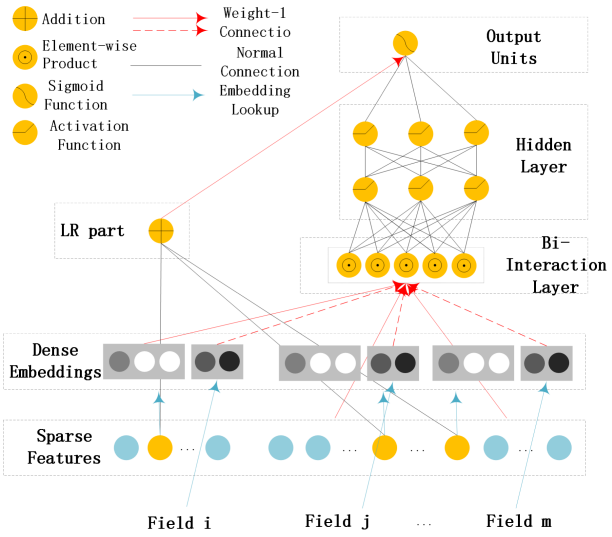

FIGURE 3. FNFM model sturucure.

B. EMBEDDING LAYER

Since the feature representations of category features are generally high-dimensional and sparse, they are usually It is compressed into a low dimensional space [15]. Traditional embedding techniques map feature vectors under each feature set to:

$$e_i = V_i x_i \tag{7}$$

where V_i is the embedding matrix corresponding to feature group t_i, and x_i is a one hot encoding vector. In order to be able to cross and combine dense numerical features with sparse category features, dense numerical features can also be compressed into low-dimensional spaces by embedding techniques. Express numerical features as:

$$e_m = v_m x_m \tag{8}$$

where v_m is an embedding vector corresponding to a feature set m, and x_m is a scalar input value. By projecting the category and numerical features together into a low-dimensional space of the same dimension, it can learn the interaction between features under different feature groups through cross layer.

In our proposed method, the embedding layer has two modes, a regular mode and a space optimized mode.

1) REGULAR MODE

The regular mode of embedding layer uses a feature group embedding method similar to the FFM model to convert feature to a low-dimensional dense representation. For the i-th feature group t_i, let $V^{ij} = [v_1^{ij}, \ldots, v_m^{ij}, \ldots, v_{K_i}^{ij}] \in R^{D \times K_i}$ represent the embedding vector dictionary used by the i-th feature group to interact with features under the j-th feature group, where $v_m^i \in R^D$ is a D dimension embedding vector.

In regular mode, the number of parameters in embedding layer is $n(f-1)k$, where n is the total feature numbers, f is the number of total fields, k is embedding size.

2) SPACE OPTIMIZED MODE

The space optimized mode of embedding layer uses a feature embedding method similar to the FM model and assign a field vector to each feature group. For the i-th feature group t_i, let $V^i = [v_1^i, \ldots, v_m^i, \ldots, v_{K_i}^i] \in R^{D \times K_i}$ represent the embedding vector of features in i-th feature group, where $v_m^{ij} \in R^D$ is a D dimension embedding vector. Let $f_i \in R^D$ represent the field vector of feature group t_i.

In this mode, the number of parameters in embedding layer is $nk + ft$, where n is the total feature numbers, f is the number of total fields, k is embedding size of feature, t is embedding size of field.

C. BI-INTERACTION LAYER

The FNFM model uses the idea of factorization machine to learn the expression of second-order interactive features in the form of hidden vector products. Different from the traditional models such as DeepFM and NFM, the FNFM model uses a second-order feature interaction based on field aware method.

Let two input features from different feature groups f_i, f_j be divided into x_i, x_j. The second-order feature interaction vector calculated by FNFM with regular embedding mode is

$$a_{i,j} = x_i v_{i,f_j} \odot x_j v_{j,f_i} \tag{9}$$

where $\odot$ denotes a vector element-by-element product operation, v_{i,f_j} denotes an implicit vector used when the input x_i interacts with input from the f_jth feature group, and v_{j,f_i} denotes that the input x_j interacts with the input from the f_ith feature group.

The second-order feature interaction vector calculated by FNFM with space optimized embedding mode is

$$a_{i,j} = x_i v_i f_i \odot x_j v_j f_j \tag{10}$$

where $\odot$ denotes a vector element-by-element product operation, v_i denotes the embedding vector of feature i, f_i denotes the embedding vector of field i.

1) BI-INTERACTION CONCATENATION LAYER

For a model with f feature group inputs, we can find $\frac{f*(f-1)}{2}$ second-order cross-product vectors. NFM use sum pooling method to compress the $\frac{f*(f-1)}{2}$ vectors into one with the NFM model. And this vector is the input of deep neural network. Motivated by Yang's Networks on FFM, The FNFM model with bi-interaction concatenation layer concert the cross-product vectors into a vector of $\frac{f*(f-1)}{2} * D$ dimension:

$$f_{BI}(V_x) = a_{1,2} \oplus a_{1,3} \oplus \cdots \oplus a_{f-1,f} \tag{11}$$

where $a_{i,j}$ are the intersection vectors of the features of the feature group t_i and the feature group t_j, and $\oplus$ is the concatenate operator. Compared with the traditional second-order interactive vector pooling layer, the bi-interaction concatenation can retain the maximum information. The information contained in the second-order interaction vector is beneficial to the subsequent deep neural network to extract higher-order combination modes.

2) BI-INTERACTION ATTENTION LAYER

Motivated by the use of attention [16] mechanism used in AFM [17], we found that using the attention-based pooling mechanism can also preserve the second-order cross vector information. We use the attentional pooling mechanism to compress the interaction vectors into one single vector by weighted sum:

$$f_{BI}(V_x) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} w_{ij} a_{ij} \tag{12}$$

where w_{ij} is the attention score for feature interaction a_{ij}, which can be interpreted as its relative importance.

$$w'_{ij} = h^T ReLU(W(v_i \odot v_j)x_i x_j + b) \tag{13}$$

$$w_{ij} = \frac{exp(w'_{ij})}{\sum_{(i,j)\in R_x} exp(w'_{ij})} \tag{14}$$

where $W \in R^{t \times k}$, $b \in R^t$, $h \in R^t$ are model parameters, and t denotes the hidden layer size of the attention network, ReLU [18] is activation function. The attention scores are normalized through the softmax function.

D. NORMALIZATION LAYER

Due to the use of field-aware embedding layer and interactive vector concatenate operations, the numerical statistical distribution of each interaction vector is insensitive to other interaction vectors in the model learning process. The numerical distribution of the output vectors of the second-order interactive concatenation layer in each dimension will have a large difference.

This will reduce the overall convergence speed and performance of the model. We use batch normalization in the second-order interactive concatenation layer to ensure that the statistical distribution of each dimension of the input vector of the deep neural network has small differences. Batch normalization [19] is an adaptive reparameterization method that is mainly used to solve the problem that the gradient of the deep neural network disappears during the training process. Batch normalization can transform the data into a statistical distribution with a mean of 0 variance of 1. If the input data of the network changes too much, using BN can make the input of the network more stable, because it make the learning of one layer independent with other parts of the network.

E. MULTIPLE LAYER PERCEPTRON (MLP)

The MLP is used to extract high-ordered features and for prediction. Its input is concatenated vector after batch normalization. Each layers in the MLP with activation function of RELU on last layer's output, and it uses softmax layer at last to complete the task of probability prediction.

F. LOSS FUNCTION

The negative log-likelihood function is widely used in CTR models, which is usually defined as:

$$L = -\frac{1}{N}\sum_{(x,y)\in\mathbb{S}} (y\, logp(x) + (1-y)\, log(1-p(x))) \quad (15)$$

where S is training data set whose size is N, x is the input of the network, $y \in 0, 1$ represents whether user clicks the item and $p(x)$ is the final output of the network which represents the probability that user will click the item.

The workflow of FNFM is shown in Algorithm1.

Algorithm 1 Field-Aware Neural Factorization Machine

Require: input data D, list of feature vectors
  Initialize *FNFMnetwork*
  D_E = embedding(D) {embedding layer}
  D_A = bianaryInteraction(D_E) {$Eq9, 10$}
  D_BI = concatationAndAttention(D_A) {$Eq11-14$}
  D_BN = batchNormalization(D_BI)
  $\widehat{Y}$ = mlpPredict(D_BN)
  if training **then**
    for *count* in $epoch_num$ **do**
      update network with $Eq15$
    end for
  end if

IV. EXPERIMENT

In this section, we move forward to evaluate the effectiveness of the performance of FNFM.

A. TASK AND DATA

We uses Kaggle Avazu[1] [2] ad click rate prediction data set. Avazu provided 11 days worth of data, including user behavior time, ad delivery site information, ad position information, user device information, user IP address, network connection type, and 9 anonymous category features. The goal is to use the data provided to predict whether a user will click the ad that is shown to him. Avazu contains a total of 40,428,967 samples in 10 days, including 33,563,901 positive samples and 68,865,66 negative samples. Due to the limited performance of the experimental machine, the experiment samples 10% of the sample of the data set, selects the first 9 days as the training data set in the sampled samples, and divides 50% of the samples into the verification data set and the test data set on the last day. The sampled data set has a total of 4042897 samples, including 3620824 training data sets, 211303 verification data sets, and 211037 test data sets.

B. THE EFFECTIVENESS OF THE BI-INTERACTION

All of the models in this section were targeted at minimizing the cross entropy loss function and optimized using the Adam [20] method with a learning rate of 0.001. For the comprehensive consideration of training time and convergence speed, the batch size is chosen to be 4096. The network structure uses 3 hidden layers with 256 neurons per layer. Each feature group has a feature embedding dimension of 4 dimensions, an L2 regularization term with an intensity of 0.00001 for linear weights, and an L2 regularization term of 0.00001 for the embedding vector, and the hidden layer neurons do not use regularization terms. In addition, Batch Normalization technology is used on the output of the Bi-Interaction Layer.

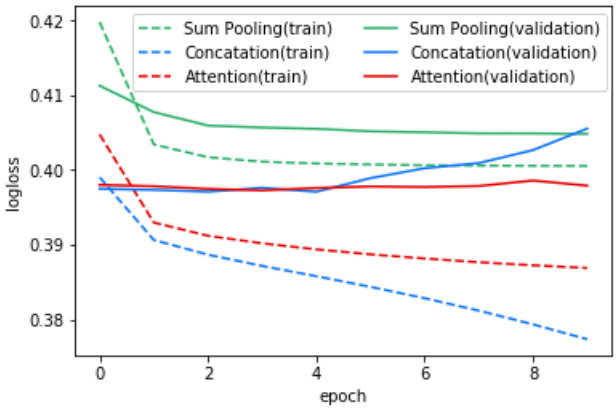

FIGURE 4. Comparison of learning curve between different bi-interaction layer.

The figure shows the training error and validation error for each round of the FNFM model using different Bi-Interaction Layers.

As you can see from Figure 4, the both Bi-Interaction Layer using concatation mode and Bi-Interaction Layer with attention mode can achieve lower training and validation errors, which means that the Bi-Interaction Layer used in the FNFM model compared to the Bi-Interaction Pooling Layer used in NFM can help improve the expressiveness of the model.

C. THE EFFECTIVENESS OF BATCH NORMALIZATION

We found that due to the use of the Bi-Interaction concatation Layer, the distribution of input data to deep neural networks becomes unstable. The following figure shows the standard deviation distribution of neurons input by deep neural networks when using batch normalization (BN for short) and not using BN:

As can be seen from Figure 5, when BN is not used, the standard deviation of the input values of the deep neural network is large, which indicates that the statistical distribution of the input data of the deep neural network is unstable. That will affect the learning process of subsequent network. And after using BN, the standard deviation of the input values is significantly lower, this suggests that BN helps the statistical distribution of deep neural network input values to be more stable. The figure below shows the training and test errors for each round of BN and no BN for the FNFM model:

[1] https://www.kaggle.com/c/avazu-ctr-prediction

[2] the code is released on https://drive.google.com/open?id=10WxN9B6IyL2ioP0lRmGGnHcqsLJprp6G

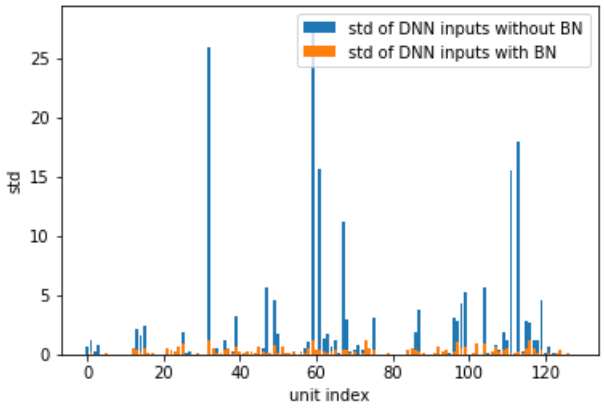

FIGURE 5. Standard deviation distribution of the DNN input, with BN and without BN.

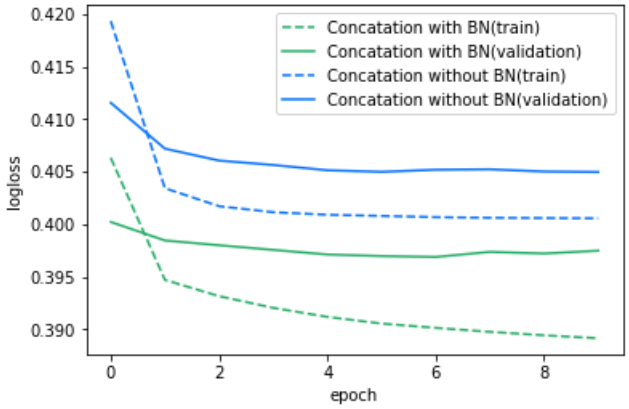

FIGURE 6. Comparison of FNFM learning curves with BN and without BN.

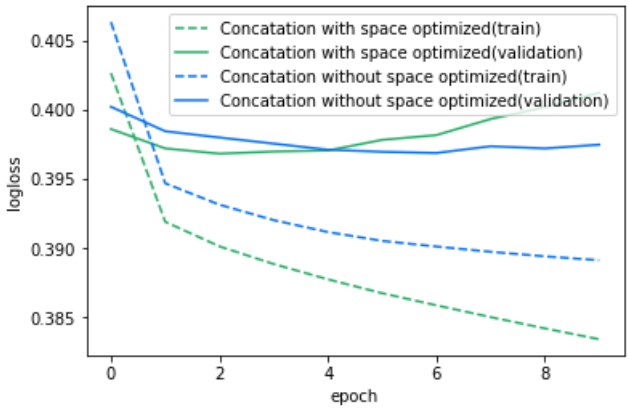

FIGURE 7. Comparison of space optimized embedding mechanism with concatenation layer.

Experiments have shown that the training error after using BN is significantly faster than the speed without using BN. The model using BN can achieve lower cross entropy loss when training same epochs.

D. PERFORMANCE OF SPACE OPTIMIZATION METHODS

In this section, we will compare the effects of space optimized embedding methods on model performance.

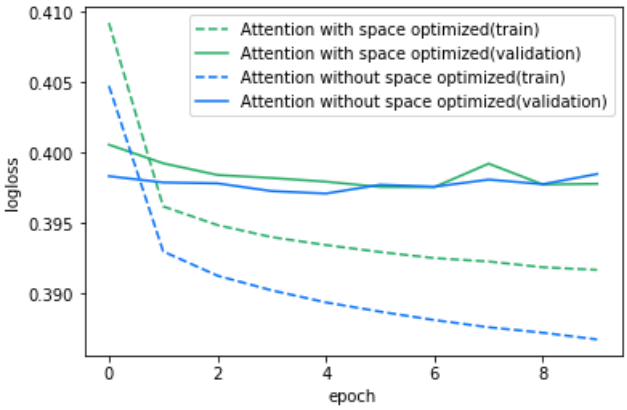

FIGURE 8. Comparison of space optimized embedding mechanism with attention layer.

When using the concatenation layer, the space optimized embedding method converges slower and the logloss is almost close to the original embedding method.

When using the attention layer, the space optimized embedding can achieve lower logloss on both train and validation data.

E. MODEL COMPARISION

1) ACCURACY

This section compares the FNFM model with the LR [21], FM, FFM, PNN [22], WDL, DeepFM, NFM, and DCN models, which include some of the most advanced models currently in the recommended system. In this experiment, because the model is concerned with the automatic learning ability of the feature combination, the features generated by the artificial feature engineering are not added, and all the original features are used.

FNFM combines the FFM and DNN models into an end-to-end model. FFM and DNN use different method when learning high-order combinations of features, one can learn cross feature explicitly by second-order feature combinations, and the other can learn high-order combinations between features implicitly. FNFM aggregate the second-order combination features learned by FFM as input to the DNN module which makes it easier for the DNN module to learn the high-order combination patterns contained in the data.

The hyperparameters for each model are obtained by performing a grid search on the validation set. The best parameter settings are given in the corresponding subsections below. For FM and FFM we use the AdaGrad algorithm with an initial learning rate of 0.1, and for other models we optimize with the Adam algorithm with an initial learning rate of 0.0001. Use L2 regularization of size 0.00001 for hidden vectors. For the FFM and FNFM models, because their parameters are too large, the hidden vectors are fixed in this experiment as 4 dimensions. For other models, search from 4, 8, 16, 32, 64. The structure of the deep neural network is searched from a combination of 2 or 3 layers and 128 or

TABLE 1. Results on the Avazu dataset.

Model	LogLoss	Accuracy	AUC
LR	0.4059	0.8135	0.7296
FM	0.3991	0.8332	0.7433
FFM	0.3980	0.8353	0.7455
PNN	0.3983	0.8353	0.7450
WDL	0.3993	0.8328	0.7425
DeepFM	0.3981	0.8354	0.7451
NFM	0.3988	0.8356	0.7437
DCN	0.3978	0.8354	0.7462
FNFM	**0.3973**	**0.8357**	**0.7470**

256 neurons per layer. Table 1 shows the corresponding test set scores for different models when the validation set achieves the lowest logloss.

By observing the above table, it can be seen that LR is the worst effect in the model, which shows that a series of methods based on factorization is useful for modeling sparse features. Since the artificial feature engineering is not performed in this experiment, the WDL model does not perform well, but it has many performance improvements compared to the LR model, which indicates that the deep neural network plays a role in learning high-order feature combinations. The FNFM model is optimal for both the test set Logloss and AUC metrics, which suggests that it is useful to capture high-order features by introducing the concept of the field during feature interaction (comparing with DCN, the concept of fields can reduce confusion during feature interactions) and using the Bi-Interaction layer (comparing with DeepFM, it same most information in feature interaction) before the DNN network.

2) CROSS-VALIDATION

In order to have insight about how the models will generalize to an independent dataset, we have compared performance of 5 models (PNN, DeepFM, NFM, DCN and FNFM) with a 5-fold cross-validation evaluation. We didn't include LR and WDL models here, because they didn't perform well in accuracy; we also didn't show the results of FM and FFM models here, because DeepFM and FNFM extend their structure respectively and have better performance. For simplicity, we use train the models in 20 epoches and compare their prediction accuracy (hyperparatemeters are the same with the previous section). From Table 2, we can see FNFM has highest average accuracy and second smallest standard deviation. And in the cross validation, FNFM achieves highest results in all 5 folds. The results show FNFM has potentially good stability and generalizability capacity comparing with other state-of-art models. The better stability of FNFM against DeepFM and NFM is an evidence of the advantage of using 'fields' model in click-through prediction tasks.

3) OVERFITTING FIXING AND HYPERPARAMETERS PREFERENCE

In this section, we firstly investigate the valuation accuracy of 5 models (PNN, DeepFM, NFM, DCN and FNFM) when training epoches increase. When we set initial learning rate

TABLE 2. Accuracy on cross validation.

Model	Accuracy	Std.	Fold1	Fold2	Fold3	Fold4	Fold5
PNN	0.8352	**0.000172**	0.8352	0.8355	0.8355	0.8352	0.8353
DeepFM	0.8352	0.000246	0.8352	0.8353	0.8356	0.8352	0.8354
NFM	0.8351	0.000236	0.8353	0.8354	0.8355	0.8353	0.8352
DCN	0.8352	0.000245	0.8351	0.8354	0.8356	0.8350	0.8353
FNFM	**0.8354**	**0.000212**	**0.8354**	**0.8355**	**0.8358**	**0.8356**	**0.8355**

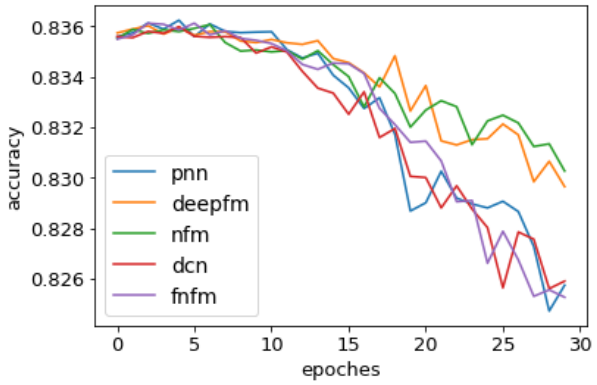

FIGURE 9. Overfitting with learning rate 0.001.

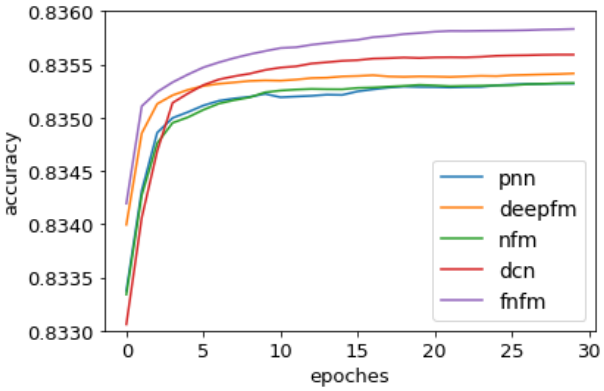

FIGURE 10. Accuracy with learning rate decay from 0.0001.

is 0.001 (all other parameters keep the same with previous sections), we can see the validation accuracy of all the models decrease fast (shown in Figure 9), it means all of the models overfit the training data in small epoches. This result is reasonable because when we consider higher ordered features, we will have large parameter space. So a smaller and decaying learning rate is useful. As shown in Figure 10, when the initial learning rate 0.0001 which decays rate 0.9 for each epoch, we can have converge in 30 epoches with sound result.

Finally we investigate the impact of network structure (i.e. network structure hyperparameters) on performance. We mainly check how changes on 'embedding size' and 'neuron size' will affect performance while other hyperparameters keep the same with previous sections. In Table 3, the column *EMB-n* shows performance of models when we change the feature embedding size to n, and the column *NET-m* shows performance when we change neuron size in neural

TABLE 3. Network structures.

Model	EMB-4	EMB-16	NET-32	NET-128	NET-256
PNN	0.8351	**0.8354**	0.8348	0.8351	**0.8354**
DeepFM	0.8352	**0.8354**	0.8353	**0.8354**	**0.8354**
NFM	0.8351	**0.8353**	0.8310	**0.8353**	0.8350
DCN	0.8352	**0.8354**	0.8353	0.8352	**0.8354**
FNFM	**0.8357**	0.8344	0.8348	0.8356	**0.8357**

network layers to m. We can see all models prefer larger neuron numbers (256) in network, but FNFM prefers smaller embedding size. It might because the larger embedding size will introduce noise and FNFM has more complex process, the two reasons cause more errors.

V. CONCLUSION

This paper mainly introduces how to apply the deep learning technology to the click-rate prediction task and improve the accuracy of the click-through rate prediction model. From the perspective of feature combination learning, this paper proposes a field-aware neural factorization machine for click rate prediction. This model has strong second-order feature interactive learning ability like FFM. Furthermore, our model uses a deep neural network to learn higher-ordered feature combinations. In the experiment on CTR prediction, we first explained that the Bi-Interaction layer used in the FNFM model has better expressiveness than the Bi-Interaction Pooling layer in the NFM model, and then illustrates the solutions of the problems of using the Concatenation Layer. Finally, we made a comparison on the effects of FNFM and other models, which does not only verified the expressive ability of FNFM, but also verified our idea of involving factoritation mechanism and information aggregation in feature extraction.

Now the prediction model doesn't consider some practical issue in business, including the history of user clicking, change of interests, and the attention which can activate the memory. Our next work is to combine such techniques to make a more useful prediction system.

REFERENCES

[1] B. Smith and G. Linden, "Two decades of recommender systems at Amazon.com," *IEEE Internet Comput.*, vol. 21, no. 3, pp. 12–18, May/Jun. 2017.

[2] J. Davidson, B. Liebald, J. Liu, P. Nandy, T. Van Vleet, U. Gargi, and D. Sampath, "The youtube video recommendation system," in *Proc. 4th ACM Conf. Recommender Syst.*, Sep. 2010, pp. 293–296.

[3] J. Beel, B. Gipp, S. Langer, and C. Breitinger, "Paper recommender systems: A literature survey," *Int. J. Digit. Libraries*, vol. 17, no. 4, pp. 305–338, Nov. 2016.

[4] A. Pepelyshev, Y. Staroselskiy, and A. Zhigljavsky, "Adaptive targeting for online advertisement," in *Proc. Int. Workshop Mach. Learn., Optim. Big Data*, Sicily, Italy. Cham, Switzerland: Springer, Jul. 2015, pp. 240–251.

[5] X. He, J. Pan, O. Jin, T. Xu, B. Liu, T. Xu, and J. Q. Candela, "Practical lessons from predicting clicks on ads at Facebook," in *Proc. 8th Int. Workshop Data Mining Online Advertising*, Aug. 2014, pp. 1–9.

[6] P. Gehler and S. Nowozin, "On feature combination for multiclass object classification," in *Proc. IEEE 12th Int. Conf. Comput. Vis.*, Sep. 2009, pp. 221–228.

[7] H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, and R. Anil, "Wide & deep learning for recommender systems," in *Proc. 1st Workshop Deep Learn. Recommender Syst.*, Sep. 2016, pp. 7–10.

[8] O. Chapelle, E. Manavoglu, and R. Rosales, "Simple and scalable response prediction for display advertising," *ACM Trans. Intell. Syst. Technol.*, vol. 5, no. 4, p. 61, Jan. 2015.

[9] H. Guo, R. Tang, Y. Ye, Z. Li, and X. He, "DeepFM: A factorization-machine based neural network for CTR prediction," 2017, *arXiv:1703.04247*. [Online]. Available: https://arxiv.org/abs/1703.04247

[10] X. He and T.-S. Chua, "Neural factorization machines for sparse predictive analytics," in *Proc. 40th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Aug. 2017, pp. 355–364.

[11] R. Wang, B. Fu, G. Fu, and M. Wang, "Deep & cross network for ad click predictions," in *Proc. ADKDD*, Aug. 2017, p. 12.

[12] S. Rendle, "Factorization machines," in *Proc. IEEE Int. Conf. Data Mining*, Dec. 2010, pp. 995–1000.

[13] J. Lee, S. Kim, G. Lebanon, and Y. Singer, "Local low-rank matrix approximation," in *Proc. Int. Conf. Mach. Learn.*, Feb. 2013, pp. 82–90.

[14] Y. Juan, Y. Zhuang, W.-S. Chin, and C.-J. Lin, "Field-aware factorization machines for ctr prediction," in *Proc. 10th ACM Conf. Recommender Syst.*, Sep. 2016, pp. 43–50.

[15] W. Zhang, T. Du, and J. Wang, "Deep learning over multi-field categorical data," in *Proc. Eur. Conf. Inf. Retr.*, Padua, Italy. Cham, Switzerland: Springer, Mar. 2016, pp. 45–57.

[16] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *Proc. Int. Conf. Mach. Learn.*, Jun. 2015, pp. 2048–2057.

[17] J. Xiao, H. Ye, X. He, H. Zhang, F. Wu, and T.-S. Chua, "Attentional factorization machines: Learning the weight of feature interactions via attention networks," 2017, *arXiv:1708.04617*. [Online]. Available: https://arxiv.org/abs/1708.04617

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[19] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015, *arXiv:1502.03167*. [Online]. Available: https://arxiv.org/abs/1502.03167

[20] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*. [Online]. Available: https://arxiv.org/abs/1412.6980

[21] D. Agarwal, R. Agrawal, R. Khanna, and N. Kota, "Estimating rates of rare events with multiple hierarchies through scalable log-linear models," in *Proc. 16th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2010, pp. 213–222.

[22] Y. Qu, H. Cai, K. Ren, W. Zhang, Y. Yu, Y. Wen, and J. Wang, "Product-based neural networks for user response prediction," in *Proc. IEEE 16th Int. Conf. Data Mining (ICDM)*, Dec. 2016, pp. 1149–1154.

LI ZHANG received the B.Eng. and Ph.D. degrees from Zhejiang University, Hangzhou, China, in 2007 and 2013, respectively, where he is currently an Assistant Researcher with the Department of Computer Science. In 2009, he was a Visiting Scholar with The University of Hong Kong, Hong Kong. From 2013 to 2017, he was a Researcher with Works Applications Company, Ltd. He has authored over ten refereed papers and patents. His current research interests include deep learning, game theory, human-machine hybrid computing, and pervasive computing.

WEICHEN SHEN received the B.S. degree in computer science and technology from Hangzhou Dianzi University, Hangzhou, China, in 2016, and the M.S. degree in computer technology from Zhejiang University, Hangzhou, in 2019. His research interests include applications of machine learning and deep learning in recommendation systems.

JIANHANG HUANG received the B.S. degree in computer science and technology from the Huazhong University of Science and Technology, Wuhan, China, in 2018. He is currently pursuing the master's degree with Zhejiang University, Hangzhou, China. His main research interests include machine learning and reinforcement learning.

SHIJIAN LI received the Ph.D. degree from Zhejiang University, Hangzhou, China, in 2006. In 2010, he was a Visiting Scholar with the Institute Telecom SudParis, Évry, France. He is currently with the College of Computer Science and Technology, Zhejiang University. He has published over 40 papers. His research interests include sensor networks, ubiquitous computing, and social computing. He serves as an Editor for the *International Journal of Distributed Sensor Networks* and as a reviewer or a PC member for more than ten conferences.

GANG PAN received the B.Eng. and Ph.D. degrees from Zhejiang University, China, in 1998 and 2004, respectively, where he is currently a Professor with the Department of Computer Science and the Deputy Director of the State Key Laboratory of CAD&CG. From 2007 to 2008, he was a Visiting Scholar with the University of California at Los Angeles, Los Angeles. He has authored over 100 refereed papers. He holds 35 patents granted. His current research interests include artificial intelligence, pervasive computing, brain-inspired computing, and brain–machine interfaces. He received three best paper awards (e.g., ACM UbiComp'16) and three nominations from premier international conferences. He was a recipient of the IEEE TCSC Award for Excellence (Middle Career Researcher), the CCF-IEEE CS Young Computer Scientist Award, and the State Scientific and Technological Progress Award. He serves as an Associate Editor for the IEEE TRANSACTIONS NEURAL NETWORKS AND LEARNING SYSTEMS, the IEEE SYSTEMS JOURNAL, *Pervasive and Mobile Computing*, and *ACM Proceedings of Interactive, Mobile, Wearable and Ubiquitous Technologies* (IMWUT).