Attention-over-Attention Field-Aware Factorization Machine

Zhibo Wang,[1] Jinxin Ma,[1] Yongquan Zhang,[1] Qian Wang,[1,4,*] Ju Ren,[2] Peng Sun[3]
[1]Key Lab of Aerospace Information Security and Trusted Computing,
School of Cyber Science and Engineering, Wuhan University
[2]Department of Computer Science and Technology, Tsinghua University
[3]State Key Laboratory of Industrial Control Technology, Zhejiang University
[4]Key Laboratory of Computer Network Technology of Jiangsu Province, Southeast University
{zbwang, jiaoma, dellenzhang, qianwang}@whu.edu.cn, renju@csu.edu.cn, sunpengzju@zju.edu.cn

Abstract

Factorization Machine (FM) has been a popular approach in supervised predictive tasks, such as click-through rate prediction and recommender systems, due to its great performance and efficiency. Recently, several variants of FM have been proposed to improve its performance. However, most of the state-of-the-art prediction algorithms neglected the field information of features, and they also failed to discriminate the importance of feature interactions due to the problem of redundant features. In this paper, we present a novel algorithm called Attention-over-Attention Field-aware Factorization Machine (AoAFFM) for better capturing the characteristics of feature interactions. Specifically, we propose the field-aware embedding layer to exploit the field information of features, and combine it with the attention-over-attention mechanism to learn both feature-level and interaction-level attention to estimate the weight of feature interactions. Experimental results show that the proposed AoAFFM improves FM and FFM with large margin, and outperforms state-of-the-art algorithms on three public benchmark datasets.

Introduction

Predictive analytics is a popular topic in both industry and academics. There are companies, such as Alibaba, Microsoft and Google, who tried to predict users' preferences for click-through rate prediction (Zhou et al. 2017)(Shan et al. 2016)(Cheng et al. 2016), and academic researchers like GroupLens[1], who built the Movielens dataset for rating prediction in recommender systems research. Typically, a predictive task is formulated as estimating a function that maps features to some target, for example real-valued target for regression and categorical target for classification (He and Chua 2017).

Among plenty of prediction algorithms, factorization machine (FM) is one of the most popular ones due to its great performance and efficiency. The key of FM is that it enhances linear regression with feature interactions and estimates the weight of feature interaction with factorization models, which has better generalization ability and can be optimized in linear time. However, there are still some drawbacks that limit the performance of FM.

First, different from continuous features in FM, features in web applications are mostly discrete and categorical. In practice, categorical features will usually be converted into a set of binary features according to their feature values (Cheng et al. 2016)(He and Chua 2017)(Shan et al. 2016). However, this results in the loss of field information, thus failing to fully capture the characteristics of features. Second, in real-world datasets, there are many homogeneous features or redundant features that describe similar meanings in the feature sets of the data. Using FM to learn the feature interaction will cause that embedding vectors learned for these features become similar, and thus wrongly assigns more weights to these feature interactions. For example, in the public context-aware application recommendation dataset Frappe, there are features "weekday=sunday" and "isweekend=weekend" to show the detailed information of the date. Since these two features co-occur often in the dataset, their embedding vectors will be similar, and thereafter will affect the weight of feature interactions. (Xiao et al. 2017) utilized the attention mechanism to discriminate the importance of feature interactions, however, we argue that since they only utilized the element wise product of embedding vectors as the input of their attention network, interactions that include features of similar meanings may have similar attention.

To address these limitations, we propose our Attention-over-Attention Field-aware Factorization Machine (AoAFFM) to better capture the characteristics of features and feature interactions. Specifically, we propose the field-aware embedding layer to exploit the field information of features and combine it with the attention-over-attention mechanism to learn both feature-level and interaction-level attention to estimate the weights of feature interactions. More importantly, AoAFFM characterizes more detailed embedding vectors of features according to their field information, it also learns a feature-level attention to capture more accurate weights of feature interactions from embedding vectors and an interaction-level attention to discriminate the difference between redundant feature interactions. We conduct extensive experiments on three

*Qian Wang is the corresponding author

[1]https://grouplens.org/

public datasets to validate our intuition. Results show that AoAFFM outperforms the state-of-the-art prediction algorithms.

The main contributions of this work are as follows.

- We observe that most of the state-of-the-art prediction algorithms neglect the field information of features and fail to discriminate features of similar meanings.
- We propose a novel algorithm called Attention-over-Attention Field-aware Factorization Machine (AoAFFM). AoAFFM exploits the field information of features, and better captures the characteristics of feature interactions to address the problem of redundant features.
- Extensive experiments show that AoAFFM outperforms the state-of-the-art algorithms on Movielens and Frappe datasets, and also shows promising results on the public click-through rate prediction dataset. We also notice that the field-aware embedding layer works better when each field has more features.

Related Works

Factorization Machine has been a popular approach in supervised predictive tasks, such as click-through rate prediction and recommendation systems, due to its great performance and efficiency. Many variants of FM have been proposed in recent years. Generally, there are two kinds of variants to FM. The first kind is introducing the key of FM, that is feature interactions, into deep neural network architectures. (Cheng et al. 2016) proposed the Wide&Deep algorithm for App recommendation in Google Play, which uses a deep learning component on the concatenation of feature embedding vectors to learn feature interactions. (Shan et al. 2016) proposed DeepCross to automatically learn feature combinations. In (Qu et al. 2016), a novel algorithm named PNN was proposed to utilize multi-layer perceptrons to learn feature interactions with concatenated FM embedding vectors for multi-field categorical data. However, PNN captures little low-order feature interactions. To model both lower-order and high-order feature interactions, (Guo et al. 2017) proposed a new neural network model called DeepFM to model feature interactions of all order. Recently, (Yang et al. 2019) proposed Operation-aware Neural Networks (ONN), which uses FFM to learn different embeddings for different operations, such as convolutional operations and product operations, in forming interactions.

The other kind of FM's variants is to solve the drawbacks of FM. Since (He et al. 2017) pointed out that the inner product of embedding vectors is simply the linear combination of the multiplication of latent factors of embedding vectors, (He and Chua 2017) developed a novel NFM model to deepen FM under the neural network framework for learning higher-order and non-linear feature interactions. (Juan et al. 2016) associated multiple embedding vectors for each feature because they argued that feature interactions with features from different fields should have different embedding vectors. Furthermore, (Xiao et al. 2017) noticed that FM fails to discriminate the importance of feature interactions, thus proposed a novel model called attentional factorization machine (AFM) to utilize attention mechanism to estimate the weight of feature interactions. However, AFM only utilizes the element wise product of embedding vectors as the input of their attention network, so those redundant feature interactions that include features of similar meanings should have similar attention. To more accurately discriminate the importance of feature interactions, we propose to apply feature-level and interaction-level attention by using attention-over-attention mechanism. Besides, (Zhou et al. 2018) recently breaks the bottleneck that FM models generally compress user behaviors into a length-fixed embedding. They proposed Deep Interest Network (DIN), which uses a local activation unit to adaptively learn from sequential user behaviors.

Preliminaries

To begin with, we first describe some notations in our model. Let us assume that the feature matrix $\boldsymbol{X} \in R^{n*m}$, where the i-th row $\mathbf{x}_i \in R^m$ of $\boldsymbol{X}$ describes one instance with m features and the label matrix $\boldsymbol{Y} \in R^{n*1}$ is the prediction target of all n instances. This representation with data matrices and feature vectors is common in machine-learning approaches such as linear regression or support vector machines.

Factorization Machine

The key of Factorization Machine is that it embeds features into a latent space and models feature interactions with the inner product of their embedding vectors (He and Chua 2017). Given a real-valued feature vector $\boldsymbol{x} \in R^m$, FM estimates its target as follows.

$$\hat{y}_{FM}(\mathbf{x}) = \omega_0 + \sum_{i=1}^{m} \omega_i x_i + \sum_{i=1}^{m} \sum_{j=i+1}^{m} \langle \boldsymbol{v_i}, \boldsymbol{v_j} \rangle x_i x_j. \quad (1)$$

where $\langle \cdot, \cdot \rangle$ stands for the inner product of vectors, ω_0 is the global bias, ω_i is the weight of the i-th feature in the linear regression part, x_i is the feature value of the i-th feature, $\{\boldsymbol{v_i}, \boldsymbol{v_j}\} \in R^k (k \ll m)$ are the embedding vectors of the i-th feature and the j-th feature. The inner product of $\boldsymbol{v_i}$ and $\boldsymbol{v_j}$ defines the weight of feature interactions. In this way, FM is able to estimate reliable parameters even in highly sparse data where standard models fail. Note that in FM, features are represented as real-valued variables.

Field-aware Factorization Machine

Field-aware Factorization Machine (FFM) actually originates from tensor factorization (Rendle and Schmidt-Thieme 2010)(Jahrer et al. 2012). In FM, each feature interaction shares the same embedding vectors. However, in practice, features often belong to different fields. Thus, FFM assumes that there should be different embedding vectors for feature interactions in different fields. For example, the feature "userID=u00001" belongs to field "userID", and has feature value "u00001", so it should have different embedding vectors with feature "weather=sunny" and feature "weekday=Monday". To be more formal, we have

$$\hat{y}_{FFM}(\mathbf{x}) = \omega_0 + \sum_{i=1}^{m} \omega_i x_i + \sum_{i=1}^{m} \sum_{j=i+1}^{m} \langle \boldsymbol{v_{i,f_{k_j}}}, \boldsymbol{v_{j,f_{k_i}}} \rangle x_i x_j. \quad (2)$$

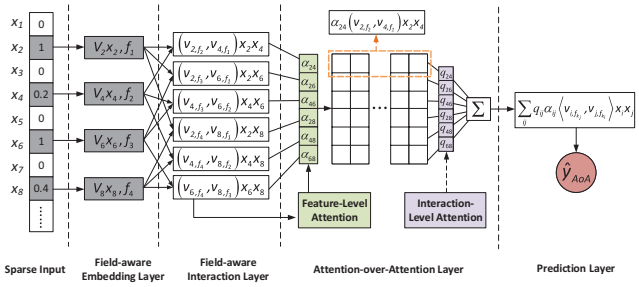


Figure 1: The neural network architecture of Attention-over-Attention Field-aware Factorization Machine

where f_{k_j} is the k_j-th field that the j-th feature belongs to, and $\boldsymbol{v_{i,f_{k_j}}}$ is the embedding vector of the i-th feature for field f_{k_j}. The other notions have the same meaning with FM. Note that usually the size of the embedding vector in FFM is smaller than that in FM because each embedding vector in FFM only needs to learn the effect with a specific field.

Attention-over-Attention Field-aware Factorization Machine

In this section, we will give a detailed introduction to our proposed Attention-over-Attention Field-aware Factorization Machine (AoAFFM). Figure 1 illustrates the neural network architecture of AoAFFM. The input layer is similar with FM, which adopts a sparse representation for input features. Then, the field-aware embedding layer embeds each non-zero feature into a multiple dense vector. Next, the field-aware interaction layer gives the element-wise product of interacted vectors and the attention-over-attention layer estimate the feature-level and interaction-level attention for feature interactions. At last, the prediction layer gives the final estimation of the target.

Field-aware Embedding Layer

First, we propose a new field-aware embedding layer in our neural network modelling. Specifically, the field-aware embedding layer contains multiple fully connected layers that project each feature into multiple dense vector representations, where each dense vector representation belongs to a specific field. Formally, let $\boldsymbol{V_i}$ be the embedding matrix of the i-th feature x_i, where each row is the embedding vector for a field. Then we have a set of field-aware embedding vectors $\boldsymbol{\mathcal{V}_X} = \{\boldsymbol{V}_1 x_1, \boldsymbol{V}_2 x_2, \ldots, \boldsymbol{V}_m x_m\}$, and each feature x_i also belongs to its field f_{k_i}.

Field-aware Interaction Layer

After the field-aware embedding layer, we feed the embedding vectors $\boldsymbol{\mathcal{V}_X}$ into the field-aware interaction layer to model feature interactions. Firstly, we expand m features to $m(m-1)/2$ feature interactions. For each feature interaction, we estimate its weight using element-wise product as in Figure 2. That is, if the first feature belongs to field f_1 and the second feature belongs to field f_2, their element-wise product through field-aware interaction layer

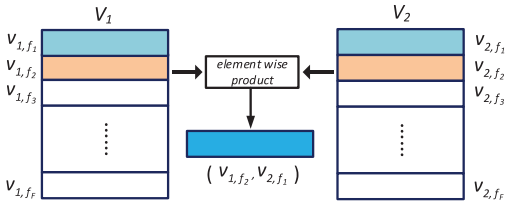


Figure 2: Field-aware Interaction Layer.

is (v_{1,f_2}, v_{2,f_1}). Formally, we can express the output of the field-aware interaction layer as follows:

$$f_{FI}(\boldsymbol{\mathcal{V}_X}) = \{(v_{i,f_{k_j}}, v_{j,f_{k_i}}) x_i x_j\}_{(i,j)\in\mathcal{F}} \tag{3}$$

where $\mathcal{F}$ stands for the set of $m(m-1)/2$ feature interactions. In the following, we write the output in the form of a matrix, denoted as product matrix P.

Attention-over-Attention Layer

After the field-aware interaction layer, we get the element-wise product matrix P of embedding vectors of features. Then we utilize attention-over-attention mechanism to learn a feature-level attention to discriminate the importance of feature interactions and an interaction-level attention to address the redundant features problem. Specifically, in the feature-level attention, we feed the product matrix P into an attention network row-wisely. While in the interaction-level attention, we utilize weight parameters on the output of feature-level attention, to estimate the weight of feature interactions directly on the interaction level.

Feature-level Attention Network To discriminate the importance of feature interactions, we apply the feature-level attention network, which is a feed forward neural network, on the row vector of product matrix P, where each row is the element-wise product of embedding vectors for a feature interaction. We denote α_{ij} as the feature-level attention score for each feature interaction $x_i x_j$. Formally, the feature-level attention network is defined as follows.

$$\begin{aligned} \alpha'_{ij} &= \boldsymbol{h}_{\boldsymbol{\alpha}}^T ReLU(\boldsymbol{W}_{\boldsymbol{\alpha}} P_r + \boldsymbol{b}_{\boldsymbol{\alpha}}) \\ \alpha_{ij} &= \frac{exp(\alpha'_{ij})}{\sum_{(i,j)\in\mathcal{F}} exp(\alpha'_{ij})} \end{aligned} \tag{4}$$

where $\boldsymbol{W}_{\boldsymbol{\alpha}} \in R^{t_1 * K}, \boldsymbol{b}_{\boldsymbol{\alpha}} \in R^{t_1}, \boldsymbol{h}_{\boldsymbol{\alpha}} \in R^{t_1}$ are the parameters of the attention network, and t_1 is the size of the hidden layer. $P_r = (v_{i,f_{k_j}}, v_{j,f_{k_i}}) x_i x_j$, and r stands for the position that feature interaction $x_i x_j$ is in the r-th row of product matrix P. $ReLU(\cdot)$ is the rectified linear unit activation function for non-linearity. The final attention score α_{ij} for feature interaction $x_i x_j$ is given through the softmax function. Note that the output of feature-level attention network, denoted as $\boldsymbol{\alpha}$, is a $\|\mathcal{F}\|$ dimensional vector, where each entry is the attention score for each feature interaction. After we obtain the attention score α_{ij} for each feature interaction from the feature-level attention, we multiply it with the element-wise product of each feature interaction, so we have $\alpha_{ij}(v_{i,f_{k_j}}, v_{j,f_{k_i}}) x_i x_j$ as the output.

Interaction-level Attention Network Although the feature-level attention network learns weights for feature interactions, it cannot address the redundant features problem. Because we use FM or FFM to learn embedding vectors, and it will cause redundant features having similar embedding vectors. Therefore, feature interactions that include embedding vectors of redundant features may have similar attention, which causes too much weights being assigned on one kind of feature interaction. In order to discriminate those similar or redundant feature interactions, we utilize the interaction-level attention layer $\boldsymbol{Q} \in R^{\|\mathcal{F}\|}$. For each output of the feature-level attention network $\alpha_{ij}(v_{i,f_{k_j}}, v_{j,f_{k_i}})x_i x_j$, $\boldsymbol{Q}$ assigns a weight q_{ij} to it. q_{ij} is randomly initialized and jointly trained with the model, and we have $q_{ij}\alpha_{ij}(v_{i,f_{k_j}}, v_{j,f_{k_i}})x_i x_j$ as the final weighted feature interaction, as illustrated in Figure 1. Since feature-level attention is not learned based on embedding vectors, even feature interactions include similar embedding vectors may have different weights, so the model can address the redundant feature problems by training $\boldsymbol{Q}$.

More formally, we estimate our target as follows.

$$\begin{aligned}\hat{y}_{AoA}(\mathbf{x}) =& \omega_0 + \sum_{i=1}^{m}\omega_i x_i + \\ & \boldsymbol{p}^T\{\sum_{i=1}^{m}\sum_{j=i+1}^{m} q_{ij}\alpha_{ij}\langle \boldsymbol{v_{i,f_{k_j}}}, \boldsymbol{v_{j,f_{k_i}}}\rangle x_i x_j\}\end{aligned} \tag{5}$$

where α_{ij} is the feature-level attention for the feature interaction $x_i x_j$, q_{ij} is the interaction-level attention weight for interaction $x_i x_j$, and $\boldsymbol{p} \in R^K$ is the weights for the prediction layer.

Training

To learn the parameters for AoAFFM, we utilized the squared loss as the objective function for training. We also added L_2 regularization on the weight matrix of both interaction-level attention layer and feature-level attention network in the objective function to prevent over-fitting. Hence, our loss function is as follows.

$$\ell = \sum_{\mathbf{x}\in\boldsymbol{X}}(\hat{y}_{AoA}(\mathbf{x}) - y(\mathbf{x}))^2 + \lambda_1\|\boldsymbol{W_\alpha}\|_2 + \lambda_2\|\boldsymbol{Q}\|_2 \tag{6}$$

where λ_1 and λ_2 are the regularization strength parameters for these two attention mechanisms. In addition, we also adopted dropout (Srivastava et al. 2014) on the feature-level attention network and batch normalization (Ioffe and Szegedy 2015) to avoid over-fitting. The idea of dropout is to randomly drop neurons of the neural network during training to avoid co-adaptations of neurons on training data, while batch normalization can normalize layer inputs to a zero-mean unit-variance Gaussian distribution for each mini-batch, which leads to faster convergence and better performance. In addition, since the dropout will be disabled during testing, the whole network can be regarded as a model averaging of multiple neural networks, which can improve the generalization and performance (Srivastava et al. 2014).

Relationship with other Neural Networks

In this section, we compare AoAFFM with existing deep models for CTR prediction.

FFM: FFM (Juan et al. 2016) utilizes the field information of features and deploys multiple feature embedding vectors for each feature according to the field of its feature interactions. After this, it uses the inner product of feature embedding vectors to estimate the weight of feature interactions. Compared with FFM, AoAFFM utilizes the attention-over-attention mechanism to better capture characteristics of feature interactions because of the information loss of the inner product.

NFM: To discriminate the weight of feature interactions, He et al. (He and Chua 2017) utilized a multi-layer perceptron to combine the element-wise product of feature embedding vector as a substitute for the inner product. However, if two feature embedding vectors are similar, their weight are still similar. This will cause the over-weight of redundant features, thus affects the performance of prediction.

AFM: In addition to NFM (He and Chua 2017), AFM (Xiao et al. 2017) adds an attention mechanism to better characterize the weight of feature interactions. AFM takes the element-wise product as input, uses a multi-layer perceptron to learn an attention score for each feature interaction, and eventually combines them with a prediction layer. Although AFM is more powerful to NFM, it may suffer the problem of redundant features as we proposed.

Experiments

To validate our intuition, we conducted experiments to address the following research questions:

RQ 1 How do the key hyper-parameters influence the performance of the proposed model?

RQ 2 Is the attention-over-attention mechanism better than the attention mechanism of AFM?

RQ 3 How does AoAFFM perform compared with the state-of-the-art algorithms?

RQ 4 How does AoAFFM perform on the large-scale click-through rate prediction dataset?

Data sets

We tested the performance of AoAFFM on three real-world benchmarks datasets: Movielens[2], Frappe[3], and Criteo[4].

- **Movielens.** In our experiments, we utilized the tag part of the Movielens dataset (Harper and Konstan 2015). Following (He and Chua 2017), since there are only positive instances in the original dataset, we randomly sampled two negative instances for each positive instance to ensure generalization.

[2]https://grouplens.org/datasets/movielens/

[3]http://baltrunas.info/research-menu/frappe

[4]http://labs.criteo.com/2014/02/kaggle-display-advertising-challenge-dataset/

- **Frappe.** We also utilized Frappe dataset (Baltrunas et al. 2015), which is generally used for context-aware apps recommendation. To ensure model generalization, we also randomly sampled two negative instances for each positive instance.
- **Criteo.** Criteo dataset is a click-through rate prediction dataset in Kaggle competition[5]. Instead of using the original version, we utilized the LIBSVM format of the dataset from (Chang and Lin 2011), because they conducted feature engineering based on a simplified version of the winning solution.

For Movielens and Frappe datasets, we randomly split them into training (70%), validation (20%), and test (10%) sets, respectively. For Criteo dataset, we utilize the train, validation and test sets that it provided. We use the validation set to tune the hyper-parameters and evaluate our attention-over-attention machenism. For final performance comparison, we use the test set. The root mean square error (RMSE) is adopted to evaluate the performance, where a lower score indicates a better performance. The statistics of the three datasets are summarized in Table 1. (Avg. stands for the average feature values that a field has)

Table 1: Statistics of the evaluation datasets.

Datasets	#Instances	#Features	#Fields	Avg.
Movielens	2,006,859	90,445	3	30,148
Frappe	288,909	5,382	10	538
Criteo	51,871,397	662,913	39	16,998

Compared Algorithms

To validate the effectiveness of AoAFFM, we compared it with the following competitive recommendation algorithms.

- **FM** (Rendle 2012). We implemented FM in a tensorflow version to use its embedding as pre-trained embedding.
- **HOFM** (Blondel et al. 2016). HOFM stands for high-order factorization machine, which learns higher-order feature combinations.
- **FFM** (Juan et al. 2016). FFM is the field-aware factorization machine, which assumes that features should have different latent factors when faced with features of different fields.
- **Wide&Deep** (Cheng et al. 2016). Wide&Deep is initially introduced for App recommendation in Google Play. It combines the wide learning component, which can be viewed as a generalized linear model, and deep learning component, which is a multi-layer perceptron, together to improve the performance of recommendation.
- **DeepCross** (Shan et al. 2016). DeepCross is a deep neural network that automatically combines features to produce superior models.
- **AFM** (Xiao et al. 2017). AFM is the attentional factorization machine, which combines attention with factorization machine.

[5]https://www.kaggle.com/c/criteo-display-ad-challenge

Parameter Settings To fairly compare all the models, we learned all models by optimizing the square loss of predictions and labels. The learning rate is searched in $[0.005, 0.01, 0.05, 0.1]$, and the best one is selected for each model. All models are learned using Adagrad in mini-batches. The batch size for Movielens data is set to 4096. For Frappe, the batch size is 128. Without special mention, $t1$, which denotes the size of the hidden layer, is set to 256 for the best performance. We adopted the early stopping strategy based on the performance on the validation set and carefully tuned the dropout ratios and regularization strength values for all models to prevent over-fitting. We also utilized the pretrained feature embedding vectors of FM and FFM to boost the performances of compared algorithms and AoAFFM, respectively.

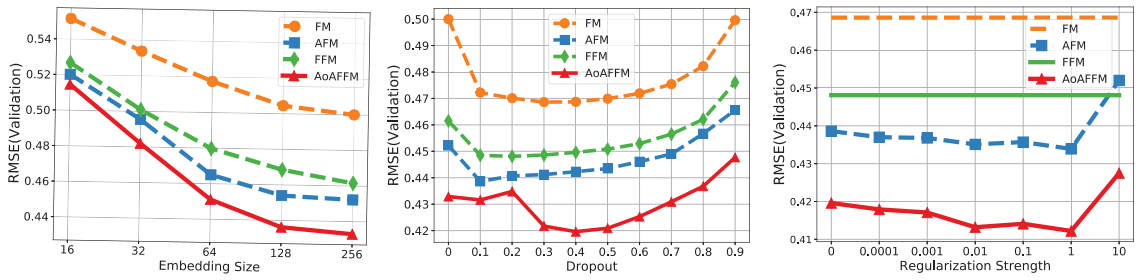
Parameter Analysis

To show the best performance of AoAFFM, we did fine tuning on the Movielens and Frappe datasets on three main parameters: embedding size, dropout ratio, and regularization strength. We first analysed the effect of embedding size, to help select the best parameter. We evaluated FM and FFM on different embedding sizes, and the results of AFM and AoAFFM are based on the pre-training feature embeddings of FM and FFM, respectively. Figure 3(a) and 3(d) illustrate the validation errors of FM, AFM, FFM, and AoAFFM on different embedding sizes. We can observe that for both datasets, the validation errors of the four models decrease as the embedding size increases, and all of them achieve the best performance when the embedding size is 256. This is because a large embedding size brings about better representation ability of models. As a result, we set all embedding sizes to 256 in the following parameter analysis.

We utilized dropout on embedding layer to avoid overfitting. Specifically, the dropout is adopted on the output layer of FM, the field-aware interaction layer for FFM, and the feature-level attention network for AFM and AoAFFM. For fairly comparsion, dropout is not utilized in the per-training feature embeddings of FFM and AoAFFM. Figure 3(b) and 3(e) illustrate the validation RMSE results, and we have the following observations:

- The performance of the four models is significantly improved by setting dropout ratios to proper values. In Movielens dataset, the best performance of AoAFFM and the rest models can be achieved when the dropout ratios are around 0.4 and 0.1, respectively. In Frappe dataset, the best dropout ratio is 0.5 for AFM, 0.2 for AoAFFM, and 0.4 for the rest. Therefore, the benefits of applying dropout can be validated. For the four models, we adopt their best dropout ratios in the following analysis.
- AoAFFM outperforms the other models in a large margin in both of the two datasets. Specifically, in Movielens, AoAFFM consistently outperforms the other models on all dropout ratios. In Frappe, AoAFFM outperforms AFM about 0.56% on the best performance. These observations show the superiority of AoAFFM to other models.

As for regularization strength parameter λ, we first set the same regularization strength for both feature-level and



(a) Movielens (b) Movielens (c) Movielens

(d) Frappe (e) Frappe (f) Frappe

Figure 3: RMSE results on the validation sets of FM, AFM, FFM, and AoAFFM under different parameters.

Table 2: The comparsion between attention-over-attention mechanism and the attentioin mechanism of AFM.

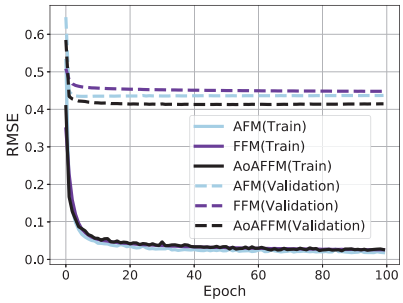
Datasets	Models	#Embedding Size	#Dropout Ratio	#Regularization Strength	**RMSE**
Movielens	AFM	256	0.1	1	0.4339
	AoAFM	256	0.1	1	**0.4248**
	AFFM	256	0.3	0.1	0.4178
	AoAFFM	256	0.4	1	**0.4128**
Frappe	AFM	256	0.6	10	0.3089
	AoAFM	256	0.3	1	**0.3068**
	AFFM	256	0.2	1	0.3130
	AoAFFM	256	0.2	0.1	**0.3116**

interaction-level attention for AoAFFM, then we evaluate AoAFFM and AFM on serveral regularization strength values. As shown in Figure 3(c) and 3(f), in Movielens, the regularization indeed helps to improve the best performance as the regularization strength grows to 1. However, the RMSE results increase when the regularization strength reachs 10, which indicates a larger regularization strength does not always lead to a better performance. As the same, in Frappe, a propoer regularization strength also improves the best performance. Moreover, in both of the two datasets, AoAFFM outperfroms AFM on most regularization values, which illustrates its superiority. Without special mention, the regularization strength is set to the best for the AFM and AoAFFM in the following comparsion.

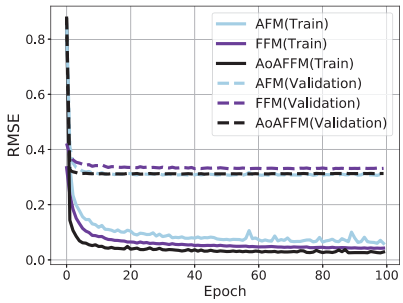
Impact of attention-over-attention layer

To further evaluate the impact of attention-over-attention layer and answer RQ 2, we first employed the attention-over-attention layer on the feature embedding of FM and named the model as AoAFM, then compared AoAFM with AFM on Movielens and Frappe datasets. Furthermore, we employed the attention mechanism of AFM on the feature embedding of FFM and named the model as AFFM, then compared AFFM with AoAFFM on both datasets. We carefully tuned dropout ratio and regularizatioin strength for AoAFM and AFFM as we do in parameter analysis, then we trained each model for ten times and obtained their average validation RMSE as the final results.

Table 2 shows the average validation RMSE results of the four models and the corresponding parameters. We find that AoAFM attains lower RMSE results compared with AFM on both datasets, and AoAFFM also outperforms AFFM on both datasets. Moreover, while the best performance of AFM is better than that of AoAFFM on Frappe in Figure 3(f), we observe that AoAFM finally outperforms AFM on Frappe. The observation confirms that, by addressing the problem of redundant features as we proposed, attention-over-attention mechanism outperforms the attention mech-



(a) Movielens



(b) Frappe

Figure 4: Training and validation error of each epoch.

Table 3: Test RMSE results of all algorithms on embedding size K=256 on Movielens and Frappe dataset.

Model	K = 256			
	Movielens		Frappe	
	#param	RMSE	#param	RMSE
FM	23.24M	0.4735	1.38M	0.3324
HOFM	46.40M	0.4636	2.76M	0.331
Wide&Deep	24.69M	0.4512	4.66M	0.3246
Deep Cross	25.42M	0.5130	8.93M	0.3548
FFM	69.55M	0.4478	13.78M	0.3345
AFM	23.31M	0.4364	1.44M	0.3126
AoAFM	**23.31M**	**0.4234**	**1.46M**	**0.3112**
AoAFFM	**69.61M**	**0.4126**	**13.84M**	**0.3141**

Table 4: Test RMSE results of AoAFFM, AFM, FFM, FM on Criteo dataset.

Model	Criteo	
	parameter settings	RMSE
FM	K=40, bs=4096	0.3971
FFM	K=4, bs=128	0.3851
AFM	K=40, bs=4096	0.3892
AoAFFM	**K=4, bs=128**	**0.3808**

anism of AFM, no matter the based feature embedding is FM or FFM.

Performance Comparison

To answer RQ 3, we compared our results with the state-of-the-art algorithms. Table 3 shows the test RMSE results of all algorithms with embedding size K=256 on Movielens and Frappe datasets, where #param denotes the number of parameters and M stands for "million". As shown in Table 3, AoAFFM achieves the best performance on Movielens dataset, significantly outperforming FM and FFM and better than AFM. This means that the design of feature-level and interaction-level attention on field-aware embedding vectors can indeed better capture the characteristics of features and feature interactions. We also notice that the performance of FFM is better than FM, HOFM, DeepCross. This validates that the field information can be of great use to improve the performance. As for the Frappe datasets, it is worth noting that the performance of AoAFM is better than that of AFM, which validates our intuition that AFM may fail to discriminate redundant feature interactions. However, because of the poor performance of FFM, AoAFFM only achieves the third best performance after AoAFM and AFM. The good performance of HOFM and Wide&Deep proves that higher-order feature interactions can indeed be helpful, but the wide part of Wide&Deep still relies on expertise feature engineering and HOFM almost doubles the amount of parameters yet the improvement is little. Deep Cross performs even worse than FM, which means that deeper learning does not always help for improvement.

Figure 4 shows the training and validation errors of AoAFFM, AFM and FFM of each epoch on Movielens and Frappe datasets. For Movielens, we can see that the validation error of AoAFFM converges faster than both AFM and FFM. For Frappe, both of the train and validation errors of AoAFFM converge faster than the other models. The observations validate that AoAFFM has better representation ability of features. Although AoAFFM shows higher training error than FFM and AFM on Movielens, the lowest test error shows that AoAFFM is not overfitting the data and has the best generalization ability.

Large-scale Dataset Performance

To further demonstrate the effectiveness of our AoAFFM, we also test the performance of AoAFFM, AFM, FFM, and FM on the large-scale click-through rate prediction Criteo dataset. For comparison, we choose the most competitive algorithm AFM and two baselines: FFM and FM. Table 4 shows their test RMSE results on Criteo dataset, where K stands for the size of embedding vectors and bs stands for the batch size. We chose a smaller batch size for FFM and AoAFFM because of the computation power limits. We observe that AoAFFM can still outperform FM and AFM even when the embedding size is only 4. In addition, the performance of FFM is slightly better than that of AFM. This proves that the field information is of great importance since features in web applications are mostly discrete and categorical. Moreover, we observe that AoAFFM outperforms other models most in Movielens than in Criteo and Frappe datasets, as the average number of field features increases, which means the field-aware embedding layer works better when the field has more features.

Conclusion

In this work, we proposed a novel algorithm called attention-over-attention field-aware factorization machine (AoAFFM) to better capture the characteristics of features and feature interactions. Specifically, AoAFFM utilizes field-aware feature embedding vectors to exploit the field information of features, it also adopts feature-level and interaction-level attention to better estimate the weights of feature interactions to address the problem of rebundant features. Our experimental results show that AoAFFM outperforms the state-of-the-art algorithms on the Movielens and Frappe datasets, and also shows promising results on the public click-through rate prediction Criteo dataset. In our experiments, we also notice that the field-aware embedding layer works better when the field has more features.

Acknowledgments

This work was supported by National Natural Science of China (Grants No. 61872274, 61822207, U1636219, and 61702562), Natural Science Foundation of Hubei Province (Grant No. 2017CFB503), the Young Tlite Scientists Sponsorship Program by CAST under Grant No. 2018QNRC001 and the Young Talents Plan of Hunan Province, and Fundamental Research Funds for the Central Universities (Grant No. 2042018gf0043, and 2042019gf0098).

References

Baltrunas, L.; Church, K.; Karatzoglou, A.; and Oliver, N. 2015. Frappe: Understanding the usage and perception of mobile app recommendations in-the-wild. *CoRR*.

Blondel, M.; Fujino, A.; Ueda, N.; and Ishihata, M. 2016. Higher-order factorization machines. In *Proceddings of Advances in Neural Information Processing Systems (NIPS)*, 3351–3359.

Chang, C.-C., and Lin, C.-J. 2011. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology* 2:27:1–27:27.

Cheng, H.-T.; 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*, 7–10.

Guo, H.; Tang, R.; Ye, Y.; Li, Z.; and He, X. 2017. Deepfm: a factorization-machine based neural network for ctr prediction. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*, 1725–1731.

Harper, F. M., and Konstan, J. A. 2015. The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems* 5(4):19:1–19:19.

He, X., and Chua, T.-S. 2017. Neural factorization machines for sparse predictive analyticsn. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, 355–364.

He, X.; Liao, L.; Zhang, H.; Nie, L.; Hu, X.; and Chua, T.-S. 2017. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web (WWW)*, 173–182.

Ioffe, S., and Szegedy, C. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*, 448–456.

Jahrer, M.; Toscher, A.; Lee, J.-Y.; Deng, J.; Zhang, H.; and Spoelstra, J. 2012. Ensemble of collaborative filtering and feature engineered models for click through rate prediction. In *KDDCup Workshop*.

Juan, Y.; Zhuang, Y.; Chin, W.-S.; and Lin, C.-J. 2016. Field-aware factorization machines for ctr prediction. In *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys)*, 43–50.

Qu, Y.; Cai, H.; Ren, K.; Zhang, W.; Yu, Y.; Wen, Y.; and Wang, J. 2016. Product-based neural networks for user response prediction. In *Proceedings of the IEEE 16th International Conference on Data Mining ICDM)*, 1149–1154.

Rendle, S., and Schmidt-Thieme, L. 2010. Pairwise interaction tensor factorization for personalized tag recommendation. In *Proceedings of the third ACM international conference on Web search and data mining (WSDM)*, 81–90.

Rendle, S. 2012. Factorization machines with libfm. *ACM Transactions on Intelligent Systems and Technology* 3(3):57.

Shan, Y.; Hoens, T. R.; Jiao, J.; Wang, H.; Yu, D.; and Mao, J. 2016. Deep crossing: Web-scale modeling without manually crafted combinatorial features. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 255–262.

Srivastava, N.; Hinton, G. E.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research* 15(1):1929–1958.

Xiao, J.; Ye, H.; He, X.; Zhang, H.; Wu, F.; and Chua, T.-S. 2017. Attentional factorization machines: Learning the weight of feature interactions via attention networks. *arXiv preprint arXiv:1708.04617*.

Yang, Y.; Xu, B.; Shen, F.; and Zhao, J. 2019. Operation-aware neural networks for user response prediction. *arXiv preprint arXiv:1904.12579*.

Zhou, G.; Song, C.; Zhu, X.; Ma, X.; Yan, Y.; Dai, X.; Zhu, H.; Jin, J.; Li, H.; and Gai, K. 2017. Deep interest network for click-through rate prediction. *arXiv preprint arXiv:1706.06978*.

Zhou, G.; Zhu, X.; Song, C.; Fan, Y.; Zhu, H.; Ma, X.; Yan, Y.; Jin, J.; Li, H.; and Gai, K. 2018. Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18, 1059–1068. New York, NY, USA: ACM.