Self-gated FM: Revisiting the Weight of Feature Interactions for CTR Prediction

Zhongxue Li[1], Hao Wu[1(✉)], Xin Wang[2], Yiji Zhao[3], and Lei Zhang[4]

[1] School of Information Science and Engineering, Yunnan University, Kunming, China
lzhxue@mail.ynu.edu.cn, haowu@ynu.edu.cn
[2] School of Computer Science, Wuhan University,Wuhan, China
[3] School of Computer Science, Beijing Jiaotong University, Beijing, China
[4] School of Electrical and Automation Engineering, Nanjing Normal University, Nanjing, China

Abstract. With the successful application of factorization machine models in click-through rate prediction, the automatic selection of feature interactions has attracted extensive attention. As the most commonly-used strategies, automatic construction of limited high-order cross-feature and automatic learning of feature interaction weight have made great progress. However, most studies still face challenges of complex training and search process. Therefore, we propose a self-gating mechanism for automatic feature selection of factorization machine models and implement a portable self-gating layer. In the self-gating layer, the weight of feature interaction is revisited through the attention network with different attentive aspects, and then the gate status is dynamically determined according to the attention score to achieve the effect of automatic selection. Our method can be easily ported to FM and DeepFM. The experimental results on two real-world data sets show that our proposed methods are superior to many state-of-the-art methods.

Keywords: Factorization machines · Feature selection · Self-gating · Neural networks · Click-through rate prediction

1 Introduction

With the development of information overload, the problem is becoming more and more serious. Recommender systems came into being and developed into

Supported by the National Natural Science Foundation of China (61962061, 61562090), the Postgraduate Research and Innovation Foundation of Yunnan University (2021Y276), and partially supported by the Yunnan Provincial Foundation for Leaders of Disciplines in Science and Technology (202005AC160005), Yunnan High-Level Talent Training Support Plan: Young Top Talent Special Project (YNWR-QNBJ-2019-188).

H. Gao et al. (Eds.): CollaborateCom 2022, LNICST 460, pp. 415–432, 2022.
https://doi.org/10.1007/978-3-031-24383-7_23

a research hotspot. Click-through rate (CTR) prediction is a basic task of FM-based recommendation systems. The core issue of recommendation algorithms is effectively modeling feature interaction. For example, an 18-year-old female user tends to give feedback (e.g., click, purchase) to skirts. The interaction of gender and age can release a stronger signal to help predict the feedback probability. Making full use of feature interactions is essential for achieving potential benefits for recommender systems.

With the introduction of multiple contextual features into modeling user-item interaction, matrix decomposition methods become powerless regarding the generalization ability. Factorization Machines (FM) [20] can simulate most decomposition models through feature engineering, and thus can combine the universality of feature engineering with the advantages of the decomposition model to estimate the interaction between a wide range of features (e.g. categorical variables and continuous variables). In real life, not all features and feature interactions are equally helpful and predictive [5]. The introduction of noise features may increase the model complexity and damage the prediction performance. Therefore, it is necessary to distinguish useful feature interactions from useless ones to ensure the performance and efficiency of FM models.

Method such as DCN [25], xDeepFM [15], AutoInt [21], AutoCross [17], and FiBiNET [12] focuses on automatically constructing finite high-order cross features and learning the high-order interaction of input features. While these models can only search a small part of all interactive features. Another technical route is to automatically learn weight for feature interactions. Typical examples are AFM [26], FwFM [19], and IFM [9]. A significant disadvantage of such methods is that the noise features are involved in the prediction. The gating mechanism enables the gates to be closed for noise features and kept for the features that are favorable to the model. Based on this principle, AutoFIS [16] and GateNet [11] have been proposed and achieved state-of-the-art performance on feature selection in the FM models.

Although existing solutions have made great progress, they still face some challenges, e.g., the complex training method in AutoFIS, and the complex search process in AutoCross and AutoInt. A simple yet elegant manner is always worth exploring. Guided by these, this paper proposes self-gated factorization machines to address existing challenges. Different from existing solutions, our self-gating mechanism is based on the attention network where we revisit the role of feature interaction weight, and directly take these attention scores as signals to activate the gate status. By this, our method enables the gate status to be determined dynamically in a portable component and to learn the relative importance of the kept feature interaction simultaneously. Therefore, our method does not prescribe any additional search process. Also, our method concentrates on pooling the second-order cross-features which are the most commonly used instance of FMs, and thus is easily applied to most of the factorization machines.

Our contributions in this paper can be summarized as follows: (I)A novel self-gating method for automatic feature selection of cross-features is proposed based on the attention mechanism. Most importantly, the feature-field interaction scheme is proposed and explored to enhance the attention mechanism.

(II) Four variants of self-gating factorization machines (gFM, gFM+, gDeepFM, gDeepFM+) are proposed and demonstrate that our method is portable and easily migrated to the existing FM models to provide a thoroughly end-to-end solution. (III) We conduct extensive experiments on two large-scale datasets and show that migrating self-gating components significantly benefits the existing FM models. Also, our proposed self-gated FM models achieve competitive or superior results compared to the SOTA methods.

The following structure of this paper is organized as follows: Sect. 2 reviews the related work. Section 3 introduces the model of self-gating factorization machines and details the self-gating mechanism. Section 4 is the settings of the experiment. The analysis of experimental results and some influencing factors are carried out in Sect. 5. Section 6 shows the conclusion of this paper.

2 Related Work

2.1 Factorization Machines

Based on naive FM, there are many variants to enhance feature interaction. FFM [13] attributes features of the same properties to the same field and proposes an upgraded version of FM. FmFM [23] models the interactions of field pairs as a matrix. However, these FM models can not model high-order feature interaction until various neural components are employed to stack on naive FM to provide an effective solution for feature interaction to represent more complex learning tasks, such as NFM [8], DeepFM [7], and Wide & Deep [2]. FNN [27] is a forward neural network using FM to pre-train the embedding layer. However, FNN can capture only high-order feature interactions. FMs consider feature interactions among input features by using only polynomial expansion which fails to capture complex nonlinear patterns in data. Moreover, existing FMs do not provide interpretable predictions to users. SEFM [14] is thus proposed to overcome these two limitations by using non-parametric subspace feature mapping. In addition, many variants strengthen the FM model from different facets to solve specific problems.

2.2 Attention Mechanism

In the learning of neural network, more parameters mean stronger model expression ability. However, computing resources are limited, and we hope to use limited computing resources to process more important information. Inspired by human vision research, the attention mechanism is introduced to focus on more target-related information and ignore other irrelevant information, which has been widely used in research fields such as natural language processing and computer vision [1]. Thanks to the attention mechanism, the model focuses on learning input features that have a great impact on the results, so as to extract more critical and important information without bringing more overhead to the model's computation and storage. The self-attention mechanism [24] is that after the model receives the input information, it determines the currently important information according to the input information itself. AFM [26] proposes to apply an

attention mechanism on feature interactions, thereby transforming the computation of interaction vectors from a simple summation to a selective weighted summation. DIN [28] adaptively adjusts the influence of historical information on the prediction of the current model by weighting the historical interest sequence features. Based on the multi-head self-attention mechanism, AutoInt [21] realizes the functions of explicitly constructing cross-features and mining the correlation between features.

2.3 Feature Selection for FMs

Despite the successful application of FM and its many deep learning variants, treating every feature interaction fairly may degrade the performance. Given this, the current feature selection focuses on *Embedded* methods where the selection algorithm is blended as part of the learning algorithm.

One kind of work focuses on automatically constructing finite high-order cross features and learning the high-order interaction of input features. DCN [25] is similar to DeepFM where CrossNet is used to capture explicit high-order cross-features and DNN is to capture implicit intersection features. Following DCN, xDeepFM [15] proposes compressing the interaction layer and AutoInt [21] introduces a new interaction layer based on multi-head self-attention mechanism. AutoCross [17] uses beam search to effectively generate cross features. FiBiNET [12] dynamically learns the feature importance via the Squeeze-Excitation network [10] and fine-grained feature interactions via bilinear function. In essence, these models are still the combination of deep network and FM, and only play the role of feature selection with the help of preset structure. The limitation of these models is that they can only search a small part of all interactive features. Another technical route is to automatically learn weight for feature interactions. AFM [26] learns the importance of each feature interaction through the neural attention network. FwFM [19] learns a weight for the field to which the feature belongs. Similarly, IFM [9] introduces Interaction-Aware Mechanism to learn flexible interactions on feature and field aspects.

The gating mechanism is introduced into the factorization machine models for automatic feature selection. The gates are closed for noise features, so that the model can only learn the features that are beneficial to the model [4]. AutoFIS [16] trains the factorization machine models with a gating mechanism in a phased approach. The first stage mainly removes noise features, and the second stage focuses on retraining beneficial features. GateNet [11] introduces embedding gates and hidden gates to explicitly select potential features and implicitly capture high-order interaction respectively.

3 Methodology

3.1 Self-gated Factorization Machines

Self-gated factorization machines (gFM) can be regarded as a minimum model with self-gating functionality. It contains the *Input* layer, *Embedding* layer, *Feature Interaction* layer, *Self-Gating* layer and *Prediction* layer. On this basis,

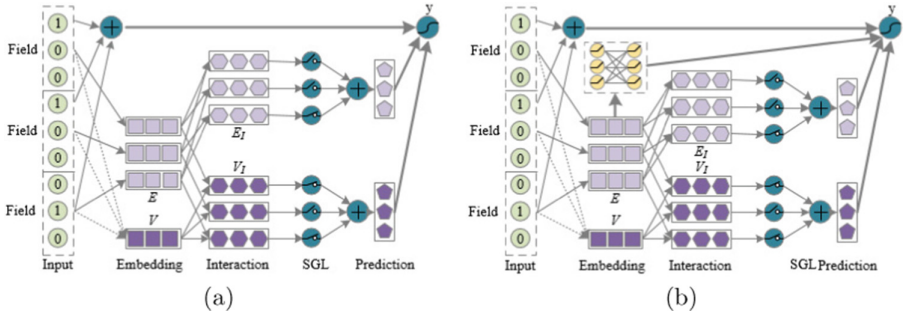


Fig. 1. The architecture of gFM+ and gDeepFM+.

we can demonstrate how to integrate the classical FM models with self-gating functionality (see Fig. 1).

Input Layer: Typical data preprocessing is to convert each data instance into a high-dimensional sparse vector through one-hot or k-hot encoding. The collection of all data instances is $\boldsymbol{X} \in \mathbb{R}^{n \times m}$ where n is the total number of instances and m is the field number of each instance after coding. Each input instance is a sparse vector $\boldsymbol{x} \in \boldsymbol{X}$, and the **feature** value of i-th **field** of $\boldsymbol{x}$ is marked as x_i, i.e.: $\boldsymbol{x} = [x_1, x_2, \ldots, x_m]$.

Embedding Layer: The function of embedding layer is to transform $\boldsymbol{x}$ into a $k \times m$ dense matrix. For each **feature** x_i, it will be projected into a k-dimensional dense vector $\boldsymbol{e}_i$. The embedding matrix $\boldsymbol{E}' = [\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_m]$ consists of m embedding vectors. Further, it is combined with $\boldsymbol{x}$ as $\boldsymbol{E} = [\boldsymbol{e}_1 x_1, \boldsymbol{e}_2 x_2, \ldots, \boldsymbol{e}_m x_m]$. In addition, to enable field-aware interaction, each **field** of the feature is also projected into a k-dimensional dense vector $\boldsymbol{v}_i \in \boldsymbol{V}$ where $i = 1 : m$.

Feature-Feature Interaction Layer: Feature interaction is a standard component in neural factorization machines [7,8,26]. Given m embedding vectors of features, it expands to $m(m-1)/2$ cross-feature vectors, where each cross-feature vector is the *Hadamard* product of two distinct vectors to encode their interaction. Formally, we have:

$$\boldsymbol{E}_I = \{\boldsymbol{e}_i x_i \odot \boldsymbol{e}_j x_j | i = 1 : m-1, j = i+1 : m\}. \tag{1}$$

Feature-Field Interaction Layer: The importance of feature interaction is affected not only by itself but also by the field to which the feature belongs. For example, a male customer (a feature) may pay more attention to the price *field* than the brand *field* when selecting goods. The interaction between the male feature and the price field should be more important than that of the brand field, and vice versa. For this reason, we add a Feature-Field Interaction layer to model this effect. Formally, we have:

$$\boldsymbol{V}_I = \{\boldsymbol{v}_i \odot \boldsymbol{e}_j x_j + \boldsymbol{e}_i x_i \odot \boldsymbol{v}_j | i = 1 : m-1, j = i+1 : m\}. \tag{2}$$

Similar to the Feature-Feature Interaction, each cross-feature is also modeled as the Hadamard product of two distinct vectors to encode their interaction. Considering that each feature is associated with a field and the bilateral symmetry of interaction, we integrate the two interactive operations of the feature to the field to implicitly capture the interaction of associated fields.

Self-gating Layer: We can use various subsequent neural network layers to project $\boldsymbol{E}_I$ and $\boldsymbol{V}_I$ to the prediction score. For example, NFM proposes using a *bilinear interaction pooling* with a fully connected layer; AFM proposes using an *attention-based pooling* and followed by a fully connected layer. However, these methods do not play the role of gating cross-features. Inspired by AutoFIS [16] and GateNet [11] which both attempt to use the gating mechanism for automatic feature selection, we design a novel self-gating layer based on the attention mechanism [18]. Let f_{SG} be a parameterized nonlinear function, it will compress $\boldsymbol{E}_I$ into a dense vector for further processing:

$$f_{SG}(\boldsymbol{E}_I) = \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} g_{ij}(\boldsymbol{e}_i x_i \odot \boldsymbol{e}_j x_j) \tag{3}$$

where g_{ij} indicates the gating status corresponding to the feature interaction of x_i and x_j, and will be learned from the data. In the same way, we can add another self-gating layer for the feature-field interactions as follows:

$$f_{SG}(\boldsymbol{V}_I) = \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} g_{ij}(\boldsymbol{v}_i \odot \boldsymbol{e}_j x_j + \boldsymbol{e}_i x_i \odot \boldsymbol{v}_j) \tag{4}$$

Prediction Layer: The output of the self-gating layer is a k-dimensional vector, which compresses all feature interactions in the embedding space after distinguishing their importance. And then, we use a fully connected layer to project it to the prediction score. At the same time, to preserve the general form of FMs, we give the overall formulation of our model as:

$$y_{gFM+} = w_0 + \sum_{i=1}^{m} w_i x_i + \boldsymbol{p}_e^\intercal f_{SG}(\boldsymbol{E}_I) + \boldsymbol{p}_v^\intercal f_{SG}(\boldsymbol{V}_I) \tag{5}$$

where $\boldsymbol{p}_e, \boldsymbol{p}_v \in \mathbb{R}^k$ denote the weights for the prediction layer.

Noted that our self-gated FM will degrade to a basic version without the feature-field interaction layer. The basic version follows the general form of factorization machines, so we marked the basic version as gFM, and named the full version as gFM+.

3.2 Self-gating Layer

Our self-gating mechanism focuses on pooling the second-order cross-features, which are the most effective and commonly used instances of FMs. The structural diagram of the self-gating layer is shown in Fig. 2. The input of SGL is a set of

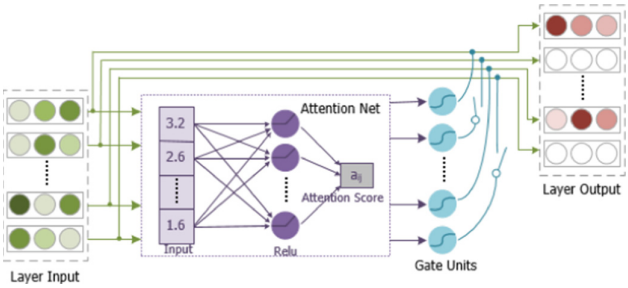


Fig. 2. The structure of the self-gating layer (SGL). The features that enter the SGL first acquire the attention score through the attention net, and then activate the gating status accordingly.

cross-feature vectors. To distinguish the contribution of different cross-features for a prediction task, an attention network [18] is used to assign different weights to each cross-feature automatically. The so-called attention is a mechanism that focuses on local information, such as a certain area of the visual field. The attention is often changed as the task changes. As for the implementation of the attention network, we use a multi-layer perceptron to parameterize the attention score (a.k.a weight of cross-feature) [26]. Given a set of cross-feature vectors (e.g. $\boldsymbol{E}_I$), the weight α_{ij} for every vector is obtained as follows:

$$\begin{aligned} \alpha_{ij} &= \boldsymbol{q}^\intercal \mathbf{relu}(\boldsymbol{W}_e \boldsymbol{E}_I^{ij} + \boldsymbol{W}_v \boldsymbol{V}_I^{ij} + \boldsymbol{b}) \\ &= \boldsymbol{q}^\intercal \mathbf{relu}\left(\boldsymbol{W}_e(\boldsymbol{e}_i x_i \odot \boldsymbol{e}_j x_j) + \boldsymbol{W}_v(\boldsymbol{v}_i \odot \boldsymbol{e}_j x_j + \boldsymbol{e}_i x_i \odot \boldsymbol{v}_j) + \boldsymbol{b}\right) \end{aligned} \tag{6}$$

where $\boldsymbol{q} \in \mathbb{R}^h$, $\boldsymbol{W}_e$, $\boldsymbol{W}_v \in \mathbb{R}^{h \times k}$, $\boldsymbol{b} \in \mathbb{R}^h$ are model parameters in the attention network and h denotes the hidden layer size of the attention network. For simplicity, we always let $h = k$. The activation function is *relu*.

According to the common practices [26], attention score α_{ij} are generally normalized with softmax function: $g_{ij} = \text{softmax}(\alpha_{ij})$ and re-assigned to each cross-feature to achieve feature calibration:

$$f_{Att}(\boldsymbol{E}_I^{ij}) = g_{ij}(\boldsymbol{e}_i x_i \odot \boldsymbol{e}_j x_j). \tag{7}$$

However, this treatment is under a defect. The softmax function in exponential form can enlarge the larger weight and suppress the smaller weight. In extreme cases, it will lead to one-hot activation where only one cross-feature is enabled. When the number of cross-features is counted in hundreds, many useful ones are killed by mistake. To avoid this drawback, it must learn a non-mutually-exclusive relationship, to ensure that more useful features are allowed to enter the subsequent propagation process [10]. For this, we revisit the role of feature interaction weights, and directly take these attention scores as signals to activate the gate status as $g_{ij} = \text{sigmoid}(\alpha_{ij})$. By this, attention scores are projected to [0,1]. It can either let no flow or complete flow of information throughout the gates to play the role of excitation [10]. Here, gate status is only determined by

the corresponding feature weight, so it can be called self-gating. Finally, we get the output of SGL with:

$$f_{SG}(\boldsymbol{E}_I^{ij}) = g_{ij}(\boldsymbol{e}_i x_i \odot \boldsymbol{e}_j x_j). \tag{8}$$

At the same time, we can get another output of SGL with: $f_{SG}(\boldsymbol{V}_I^{ij}) = g_{ij}(\boldsymbol{v}_i \odot \boldsymbol{e}_j x_j + \boldsymbol{e}_i x_i \odot \boldsymbol{v}_j)$ when considering the feature-field interactions.

Different from AutoFIS which requires a separated search stage to estimate g_{ij} and then its mask for deciding whether a cross-feature should be included in the model, our method enables g_{ij} to be determined dynamically in an attention network and to learn the relative importance of the kept feature interaction. Therefore, our methods do not prescribe an additional search process.

The gated recurrent neural network is to introduce a gating mechanism on the classical RNN to control the information transmission in the neural network [3,6]. The gating mechanism can be used to control which information in the memory unit needs to be left and which needs to be discarded. In this way, the gated recurrent neural network can learn the long-term dependence within a relatively long span, and will not lead to gradient explosion or gradient dispersion. Our self-gating layer is also inspired by the gating function in GRU, which also plays a role in controlling information transmission. The difference is that because our feature selection task focuses on suppressing unimportant information rather than learning long-term dependence, our gating layer is not a recurrent structure, and only one nonlinear activation is used to achieve the selection function.

3.3 Exploiting Self-gating Layer in Deep FM Models

SGL can be flexibly added to many existing FM models for feature selection to improve the model performance. For demonstrating this point, we select DeepFM which is a robust variant of deep factorization machines [29], to apply the Self-Gating mechanism. Table 1 summarizes the prediction rules and notations of four self-gated factorization machines: gFM, gFM+, gDeepFM, and gDeepFM+.

3.4 Objective Function and Optimization

Generally, CTR prediction is modeled as a classification task and uses the *binary cross entropy* loss function:

$$\mathcal{L}_{BCE} = -\frac{1}{n}\sum_{i=1}^{n}\left(y_i \log\left(\hat{y}_i\right) + \left(1 - y_i\right)\log\left(1 - \hat{y}_i\right)\right) \tag{9}$$

where n is the number of training instances, y_i and $\hat{y}_i$ indicates the real value and the prediction value respectively.

Over-fitting is a recurring problem in the process of optimizing machine learning models, particularly for the factorization machine models. If the neurons in the pairwise interaction layer are excessively adaptive and interdependent, which inevitably results in over-fitting when training these models. Two widely-used

Table 1. The prediction rules of the self-gated factorization machine models.

Model	Notations and prediction rule
gFM	$y = y_{LR} + y_{SG}$ $y_{SG} = \boldsymbol{p}_e^{\intercal} \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} f_{SG}(\boldsymbol{E}_I^{ij})$ where $y_{LR} = w_0 + \sum_{i=1}^{m} w_i x_i$, $\boldsymbol{p}_e \in \mathbb{R}^k$ is the weight vector of the prediction layer, y_{SG} performs feature selection only on the layer of feature-feature interaction.
gFM+	$y = y_{LR} + y'_{SG}$ $y'_{SG} = \boldsymbol{p}_e^{\intercal} \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} f_{SG}(\boldsymbol{E}_I^{ij}) + \boldsymbol{p}_v^{\intercal} \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} f_{SG}(\boldsymbol{V}_I^{ij})$ where $\boldsymbol{p}_e, \boldsymbol{p}_v \in \mathbb{R}^k$ is the weight vector of the prediction layer, y'_{SG} performs feature selection on both the layers of feature-feature interaction and feature-field interaction
gDeepFM	$y = y_{LR} + y_{SG} + y_{DNN}$ $y_{DNN} = \boldsymbol{p}^{\intercal} \sigma_L (\boldsymbol{W}_L (\dots \sigma_1 (\boldsymbol{W}_1 (\boldsymbol{E}_{con}) + \boldsymbol{b}_1) \dots) + \boldsymbol{b}_L)$ $\boldsymbol{E}_{con} = \boldsymbol{e}_1 x_1 \oplus \boldsymbol{e}_2 x_2 \oplus \dots \oplus \boldsymbol{e}_{m-1} x_{m-1} \oplus \boldsymbol{e}_m x_m$ where σ_L, $\boldsymbol{W}_L$, $\boldsymbol{b}_L$ are the activation function, weight matrix and bias vector of the L-th layer, and $\boldsymbol{p}$ is a weight vector, $\boldsymbol{E}_{con}$ is the concatenation of embedding vectors in $\boldsymbol{E}$, y_{SG} performs feature selection only on the layer of feature-feature interaction
gDeepFM+	$y = y_{LR} + y'_{SG} + y_{DNN}$ where y'_{SG} performs feature selection on both the layers of feature-feature interaction and feature-field interaction

methods to prevent over-fitting in deep learning are *dropout* and L_2 regularization. *Dropout* randomly disables a certain proportion (a.k.a $dr \in [0, 1]$) of the feature detectors to prevent co-adaptation of neurons in each training batch. L_2 regularization imposes large penalties on sparse peaked values, and finally reduces the size of parameter values to reduce model complexity.

4 Experiment Settings

4.1 Datasets

To carry out the experimental evaluation, we use two large-scale datasets from different application fields. Statistics of datasets are summarized in Table 2. Criteo comes from the click logs of 45 million users within a week. There are 26 anonymous categorical features and 13 continuous features in the dataset [1]. Avazu is released in Kaggle's CTR prediction competition. It contains about 38.55M click records of ten days ordered by time [2]. We randomly divided each dataset in the same way: 80% of samples as the training set, 10% of samples as the validation set, and the rest as the test set.

4.2 Evaluation Methods

1. *FM* [20] is the naive factorization machine.

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

Table 2. Statistics of the datasets of Avazu and Criteo.

Dataset	#Instances	#Fields	#Features
Avazu	40,428,967	24	8.37M
Criteo	45,840,617	39	5.55M

2. *FFM* [13] is the field-aware FM.
3. *DeepFM* [7] combines an FM component to capture second-order interaction, and a deep component to simulate high-order interaction.
4. xDeepFM [15] introduces a compressed interaction layer to DeepFM and enables learning high-order feature interaction at the vector level.
5. *NFM* [8] stacks a MLP component over a bilinear interaction pooling layer.
6. *AFM* [26] distinguishes and learns the importance of feature interaction through an attention network.
7. *AutoFIS* [16] is recorded as AutoFM, AutoDeepFM, etc. The models are trained in a phased manner with a search stage and re-train stage.
8. *GateNet* [11] is recorded as GateFM(e), GateFM(h), etc. 'e' and 'h' represents the application embedding gates and hidden gates respectively.
9. *FiBiNet* learns feature importance via the squeeze-excitation network and fine-grained feature interactions via bilinear functions.
10. *Self-Gated FMs* are our proposed methods and marked with a prefix of 'g'.

4.3 Evaluation Criteria

For CTR prediction, two standard indicators AUC (area under ROC) and LogLoss (cross entry) are used. The improvement of AUC and logloss by 0.001 is significant in the performance of the model [2,21,23].

Table 3. The settings of FM models. k: embedding size, lr: learning rate, bs: batch size, λ: L_2 regularization, h: attention units, dr: dropout ratio, rd: reduction ratio, $cross$: cross-layer size, c and μ are parameters in gRDA optimizer.

Model	Avazu	Criteo
General	$k = 32$ $lr = 1e{-}3$ $bs = 4096$ $\lambda = 1e{-}6$ $h = 32$ $dr = 0.2$ mlp = (32,32)	$k = 64$ $lr = 1e{-}3$ $bs = 8192$ $\lambda = 1e{-}6$ $h = 64$ $dr = 0.2$ mlp = (64,64)
NFM	mlp = (64,)	mlp = (64,)
FFM	$k = 16$	$k = 16$
FiBiNet	$rd = 3$	$rd = 3$
xDeepFM	$cross$ = (16,16)	$cross$ = (16,16)
AutoFM	$c = 5e{-}3$, $\mu = 0.8$	$c = 5e{-}4$, $\mu = 0.8$
AutoDeepFM	$c = 5e{-}4$, $\mu = 0.8$	$c = 5e{-}4$, $\mu = 0.8$

4.4 Implementation Details

The gRDA optimizer is used in the AutoFIS model for the search stage. The parameters of other baseline models and our self-gated FMs are optimized by the Adam optimizer. An early-stopping strategy is taken to all models. All models are implemented with PyTorch and all experiments are conducted on a computer with Intel(R) Xeon(R) Bronze 3204 CPU @1.90 GHz, 128 GB memory, and NVIDIA Geforce RTX 3090.

For the evaluation methods, the hyper-parameter setting is carefully searched to follow the existing configuration disclosed in their papers. For model training, both L_2 regularization and *dropout* regularization are adopted for evaluation methods. For two datasets, the parameters of FM models are set as Table 3. For our models, we use the same hyper-parameters as the base models (i.e., FM and DeepFM accordingly) except for extra ones.

5 Experiment Analysis

The experimental results are analyzed for answering the following questions:

RQ 1: Can the self-gating mechanism work well when migrating to different FM models?
RQ 2: How do our methods perform as compared with other counterparts?
RQ 3: How do key parameters and mechanisms affect our methods?
RQ 4: How do our methods perform in terms of computational efficiency?

5.1 Migration to FM Models (RQ 1)

To answer RQ1, we apply the SGL to FM [20] and DeepFM [7] to automatically select cross-features and show the experimental results in Table 4. When comparing gFM with FM, we can find that gFM significantly improves AUC and reduces LogLoss on all datasets and gFM achieves slightly better performance than DeepFM. When comparing gDeepFM with DeepFM, we can still find that gDeepFM outperforms DeepFM on the Avazu and Criteo datasets. It can prove that the self-gating mechanism is effective for the selection of cross features.

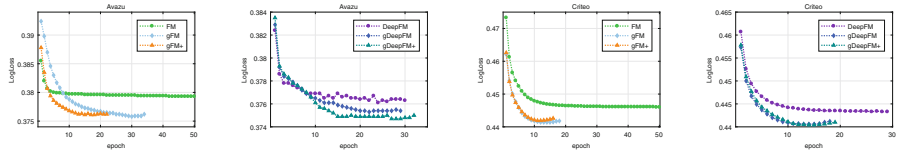
When introducing the feature-field interactions, no significant gain is noted on gFM+, yet surprising benefits are observed on gDeepFM+. On the datasets of Avazu and Criteo, gDeepFM+ increases the AUC of gDeepFM by 0.0016 and 0.0005; at the same time, reduces the LogLoss of gDeeFM by 0.0009 and 0.0010. We speculate the main reason is that it is difficult for shallow FM to learn effective representations for features and fields when the underlying data is sparse (both for the Avazu and Criteo datasets), such as users with specific preferences or niche features with a narrow appeal [2]. In such cases, there should be not enough interactions between most field-feature pairs, but dense embeddings will lead to nonzero predictions for all field-features pairs, and thus can over-generalize and make less relevant predictions [2]. On the contrary, through

multiple combinations of features, the deep component can explore the potential patterns in the data and even the correlation between rare features and the label. It indirectly has the potential to play the role of feature-field interaction patterns and improve the attention effect.

Table 4. Performance comparison of different FM models. The values in the first grade and the second grade are marked in **bold** and *italics*, respectively.

Dataset	Avazu		Criteo	
Model	AUC	LogLoss	AUC	LogLoss
AFM	0.7852	0.3778	0.8083	0.4435
NFM	0.7843	0.3776	0.8102	0.4416
FFM	0.7875	0.3769	0.8073	0.4447
FiBiNet	*0.7876*	0.3765	0.8090	0.4431
xDeepFM	0.7855	0.3787	0.8101	0.4417
AutoFM	0.7850	0.3789	0.8043	0.4478
GateFM(e)	0.7828	0.3792	0.8096	0.4421
AutoDeepFM	0.7861	0.3767	0.8087	0.4436
GateDeepFM(e)	*0.7876*	0.3766	0.8106	*0.4411*
GateDeepFM(h)	0.7848	0.3789	0.8078	0.4443
FM	0.7814	0.3793	0.8057	0.4462
gFM	0.7865	0.3761	0.8102	0.4418
gFM+	0.7865	0.3771	0.8094	0.4427
DeepFM	0.7865	0.3769	0.8085	0.4432
gDeepFM	*0.7876*	*0.3761*	*0.8111*	*0.4411*
gDeepFM+	**0.7892**	**0.3752**	**0.8116**	**0.4401**

Finally, we demonstrate the logloss changes on the validation set with each training epoch. According to Fig. 3, our proposed self-gating FM models converge faster than the naive FM and DeepFM. Since their logloss performance is much lower, it also implies that their generalization ability is better. This is also consistent with the above analysis.



(a) (b) (c) (d)

Fig. 3. Logloss on the validation set with each training epoch. (a–b) Avazu and (c–d) Criteo.

5.2 Performance Comparison (RQ 2)

To answer RQ2, our self-gated factorization machines is compared to seven baselines. The experimental results are also shown in Table 4. gDeepFM+ ranks in the 1st grade, and gDeepFM ranks in the 2nd grade on both datasets.

As above-mentioned that both AFM and NFM focus on pooling the second-order cross-features. gFM shares many components with them. gFM works much better on the Avazu dataset and shows comparable performance to NFM on the Criteo dataset. From the perspective of pooling, it is obvious that our strategy is superior to AFM and NFM. AutoFIS and GateNet are both based on gating mechanisms for the selection of cross-features. However, gFM is superior to AutoFM and GateFM(e) on all two datasets, and correspondingly gDeepFM outperforms AutoDeepFM and GateDeepFM(e) on the Avazu and Criteo datasets. Although GateNet is simpler and also competitive, it cannot be extended to model and exploit the effects of feature-field interactions. On the contrary, our mechanism is more scalable, which can result in more powerful models such as gDeepFM+. In AutoFIS, the search stage and retraining stage must be utilized to avoid co-adaptation of gating status and weight. This trims the parameter adjustment of AutoFIS in different scenarios more complex. Compared with AutoFIS, our models do not need phased training and use only one optimizer which makes the model easier to train and deploy in the context of real applications.

FFM shows its competitive edge as it also considers the effect of fields. However, FFM suffers from large-scale parameters and low efficiency. In contrast, our proposed gFM has advantages both in classification accuracy and computational efficiency (see Table 4). XDeepFM can be said to be a real "deep" factorization machine, but its complexity will be a major bottleneck and it needs to be carefully optimized to achieve a better prediction performance. Compared with SENETLayer used in FiBiNet, there are the following differences with our methods: (1) There is no squeeze stage and no pooling operations. (2) SENETLayer uses two full-connection layers to realize excitation, while SGL uses an attention network, which eliminates the operations of dimension reduction and promotion. (3) Instead of bilinear operation, the hadamard product is used for feature interactions in our method. (4) Most importantly, the feature-field interaction layer is proposed and explored to enhance the attention mechanism. DeepFM, xDeepFM, and FiBiNet are the most competitive variants of FM according to FuxiCTR which is an open benchmark for reproducible research and provides a rigorous comparison of different models for CTR prediction [29]. However, our proposed models especially for gDeepFM+ outperform them significantly.

5.3 Analysis of Influencing Factors (RQ 3)

Impact of Embedding Dimension: We keep the settings of L_2-regularization coefficient λ, dropout ratio, and learning rate unchanged, and observe the impact of embedding dimension (k) on self-gated factorization machines. As shown in Fig. 4, for the same dataset, the optimal embedding dimensions are different with various models. For gFM and gFM+, the optimal setting falls in $k \in [64, 128]$ on both datasets. When the value of k is larger than 64, it incurs overfitting and thus harms the generalization ability of the model. For gDeepFM and gDeepFM+, the optimal setting collapses in $k \in [32, 64]$. As the gDeepFM/gDeepFM+ is of the better generalization ability, increasing k empowers the prediction performance. Generally, the dimension determines the amount of information a

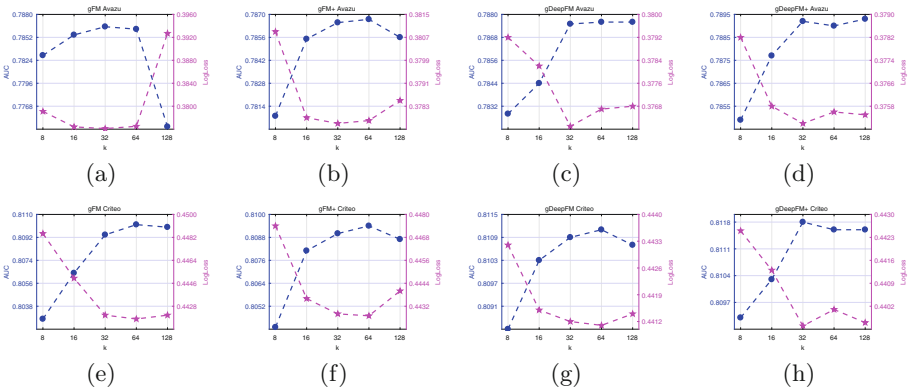


Fig. 4. The impact of embedding dimension (k) on self-gated factorization machines. (a–d) Avazu and (e–h) Criteo.

feature vector can carry. When the dimension is too small, the expression ability is insufficient, and when the dimension is too high, the cost of calculation and memory increases. Because embedding learning also has the risk of under-fitting or over-fitting, $k \in [32, 64]$ is appropriate for our proposed models.

Table 5. Performance comparison of different regularization strategies, where k=32, bs=4096 for Avazu and bs=8192 for Criteo, lr=1e-3.

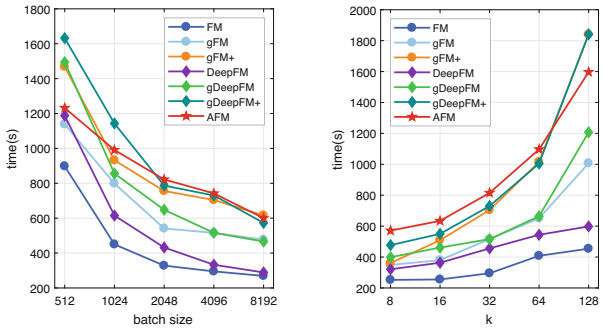
Avazu						
Regularization	$dr = 0,\ \lambda = 0$		$dr = 0.2,\ \lambda = 1e{-}6$		$dr = 0.2,\ \lambda = 0$	
Metric	AUC	LogLoss	AUC	LogLoss	AUC	LogLoss
gFM	0.7841	0.3800	0.7865	0.3761	0.7861	0.3776
gFM+	0.7847	0.3789	0.7865	0.3771	0.7848	0.3787
gDeepFM	0.7866	0.3771	0.7875	0.3761	0.7864	0.3767
gDeepFM+	0.7883	0.3767	0.7892	0.3752	0.7876	0.3770
Criteo						
Regularization	$dr = 0,\ \lambda = 0$		$dr = 0.2,\ \lambda = 1e{-}6$		$dr = 0.2,\ \lambda = 0$	
Metric	AUC	LogLoss	AUC	LogLoss	AUC	LogLoss
gFM	0.8096	0.4423	0.8102	0.4418	0.8109	0.4412
gFM+	0.8102	0.4417	0.8094	0.4427	0.8105	0.4412
gDeepFM	0.8103	0.4417	0.8111	0.4411	0.8108	0.4411
gDeepFM+	0.8112	0.4407	0.8116	0.4401	0.8114	0.4402

Impact of Regularization Strategies: Both strategies of dropout and L_2 regularization can be used in avoiding the risk of the over-fitting problem. Additionally, two regularization strategies can play a complementary effect at the same time [22]. However, according to our experiments, the situation is not always as expected as shown in Table 5. Generally, the prediction performance of self-gated

FM models can be enhanced using dropout regularization. Compared with the case without regularization ($dr = 0$, $\lambda = 0$), enabling dropout ($dr = 0.2$ benefits the prediction accuracy on both the datasets of Avazu and Criteo, except the case of gDeepFM+ on the Avazu dataset. When enabling both L_2 and dropout regularizations ($dr = 0$ and $\lambda = 1e{-}6$), the situation becomes complicated. The performance gains can be seen in gFM, gDeepFM, and gDeepFM+ against the Avazu dataset, and gDeepFM+ against the Criteo dataset. However, it harms the prediction performance of gFM+ on the Avazu dataset, and that of gFM, gFM+, and gDeepFM on the Criteo dataset. The regularization strategies need to be carefully chosen.

Table 6. Model size of different FMs.

Models	Model size
FM	$n(1+k)$
FFM	$n(1+mk)$
AFM	$n(1+k)+(hk+2h+k)$
gFM	$n(1+k)+(hk+2h+k)$
gFM+	$n(1+k)+2(hk+h+k)$
DeepFM	$n(1+k)+(mkh+Lh^2+h)$
gDeepFM	$n(1+k)+(hk+2h+k)+(mkh+Lh^2+h)$
gDeepFM+	$n(1+k)+2(hk+h+k)+(mkh+Lh^2+h)$



(a) k = 32 (b) bachsize = 4096

Fig. 5. The computational overheads with a) batch size and b) embedding dimension, against the Criteo cataset.

5.4 Computational Efficiency (RQ 4)

Regarding the model of FMs, space cost is mainly reflected in the total number of model parameters. Let n be the number of features, m be the number of fields, k be the embedding dimension, L be the total number of layers in MLP

and h be the number of hidden units in each layer, the model size of selected FM models is shown in Table 6. Generally speaking, FM has the least space overhead and FFM has the greatest space overhead. AFM and gFM have the same space complexity. gFM+ introduces $hk+k$ parameters over gFM as it introduces a field-feature interaction layer. DeepFM/gDeepFM /gDeepFM+ add $mkh + Lh^2 + h$ parameters over FM/gFM/gFM+ since that the same MLP is used for modeling deep feature interactions.

Given the same training configuration, Fig. 5(a) and 5(b) show the comparison of the computational overhead of selected methods w.r.t the embedding dimension k and the batch size, respectively. The computational efficiency of various FM variants is linear with the batch size and the embedding dimension. Thanks to the acceleration of GPU, the overhead of our proposed methods is only 2–3 times higher than that of FM or DeepFM, especially compared with AFM. The latter uses the softmax function, which significantly introduces the overhead in the back-propagation process.

6 Conclusions

We have proposed a novel self-gating mechanism for automatic feature selection of cross-features in FM models. Different from previous research, we revisit the role of the attention score of feature interactions and directly take these attention scores as signals to activate the gate status. More importantly, our methods learn that attention score depends not only on the feature-feature interactions but also on the feature-field interactions. By porting the self-gating mechanism to FM and DeepFM, we further confirm that it works better together with neural components on feature selections, and thus shows its potential for migration to other models of FMs.

References

1. Brauwers, G., Frasincar, F.: A general survey on attention mechanisms in deep learning. IEEE Trans. Knowl. Data Eng. 1 (2021)
2. Cheng, H., et al.: Wide & deep learning for recommender systems. In: Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, pp. 7–10 (2016)
3. Cho, K., et al.: Learning phrase representations using RNN encoder-decoder for statistical machine translation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP, pp. 1724–1734. ACL (2014)
4. Dauphin, Y.N., Fan, A., Auli, M., Grangier, D.: Language modeling with gated convolutional networks. In: Proceedings of the 34th International Conference on Machine Learning, ICML 2017, pp. 933–941. PMLR (2017)
5. Du, G., Zhou, L., Yang, Y., Lü, K., Wang, L.: Deep multiple auto-encoder-based multi-view clustering. Data Sci. Eng. **6**(3), 323–338 (2021)
6. Greff, K., Srivastava, R.K., Koutník, J., Steunebrink, B.R., Schmidhuber, J.: LSTM: a search space odyssey. IEEE Trans. Neural Networks Learn. Syst. **28**(10), 2222–2232 (2017)

7. Guo, H., Tang, R., Ye, Y., Li, Z., He, X.: DeepFM: a factorization-machine based neural network for CTR prediction. In: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, pp. 1725–1731 (2017). ijcai.org
8. He, X., Chua, T.: Neural factorization machines for sparse predictive analytics. In: Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 355–364. ACM (2017)
9. Hong, F., Huang, D., Chen, G.: Interaction-aware factorization machines for recommender systems. In: The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, pp. 3804–3811. AAAI Press (2019)
10. Hu, J., Shen, L., Albanie, S., Sun, G., Wu, E.: Squeeze-and-excitation networks. IEEE Trans. Pattern Anal. Mach. Intell. **42**(8), 2011–2023 (2020)
11. Huang, T., She, Q., Wang, Z., Zhang, J.: GateNet: gating-enhanced deep network for click-through rate prediction. CoRR abs/2007.03519 (2020)
12. Huang, T., Zhang, Z., Zhang, J.: FiBiNET: combining feature importance and bilinear feature interaction for click-through rate prediction. In: Proceedings of the 13th ACM Conference on Recommender Systems, 2019, pp. 169–177. ACM (2019)
13. Juan, Y., Zhuang, Y., Chin, W., Lin, C.: Field-aware factorization machines for CTR prediction. In: Proceedings of the 10th ACM Conference on Recommender Systems. pp. 43–50. ACM (2016)
14. Lan, L., Geng, Y.: Accurate and interpretable factorization machines. In: The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, pp. 4139–4146. AAAI Press (2019)
15. Lian, J., Zhou, X., Zhang, F., Chen, Z., Xie, X., Sun, G.: xDeepFM: combining explicit and implicit feature interactions for recommender systems. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD, pp. 1754–1763. ACM (2018)
16. Liu, B., et al.: AutoFIS: automatic feature interaction selection in factorization models for click-through rate prediction. In: Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp. 2636–2645. ACM (2020)
17. Luo, Y., et al.: AutoCross: automatic feature crossing for tabular data in real-world applications. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, pp. 1936–1945. ACM (2019)
18. Luong, T., Pham, H., Manning, C.D.: Effective approaches to attention-based neural machine translation. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, pp. 1412–1421. The Association for Computational Linguistics (2015)
19. Pan, J., et al.: Field-weighted factorization machines for click-through rate prediction in display advertising. In: Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, pp. 1349–1357. ACM (2018)
20. Rendle, S.: Factorization machines. In: ICDM 2010, The 10th IEEE International Conference on Data Mining, pp. 995–1000. IEEE Computer Society (2010)
21. Song, W., et al.: AutoInt: automatic feature interaction learning via self-attentive neural networks. In: Proceedings of the 28th ACM International Conference on Information and Knowledge Management, CIKM, pp. 1161–1170. ACM (2019)
22. Srivastava, N., Hinton, G.E., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**(1), 1929–1958 (2014)

23. Sun, Y., Pan, J., Zhang, A., Flores, A.: FM^2: field-matrixed factorization machines for recommender systems. In: WWW 2021: The Web Conference 2021, pp. 2828–2837. ACM/IW3C2 (2021)
24. Vaswani, A., et al.: Attention is all you need. In: Advances in Neural Information Processing Systems, pp. 5998–6008 (2017)
25. Wang, R., Fu, B., Fu, G., Wang, M.: Deep & cross network for ad click predictions. In: Proceedings of the ADKDD 2017, pp. 12:1–12:7. ACM (2017)
26. Xiao, J., Ye, H., He, X., Zhang, H., Wu, F., Chua, T.: Attentional factorization machines: learning the weight of feature interactions via attention networks. In: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI, pp. 3119–3125 (2017). ijcai.org
27. Zhang, W., Du, T., Wang, J.: Deep learning over multi-field categorical data. In: Ferro, N., et al. (eds.) ECIR 2016. LNCS, vol. 9626, pp. 45–57. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-30671-1_4
28. Zhou, G., et al.: Deep interest network for click-through rate prediction. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD, pp. 1059–1068. ACM (2018)
29. Zhu, J., Liu, J., Yang, S., Zhang, Q., He, X.: FuxiCTR: an open benchmark for click-through rate prediction. arXiv preprint arXiv:2009.05794 (2020)