

# GraphFM: Graph Factorization Machines for Feature Interaction Modeling

Shu Wu<sup>1</sup>, Zekun Li<sup>2</sup>, Yunyue Su<sup>1</sup>, Zeyu Cui<sup>3</sup>, Xiaoyu Zhang<sup>4</sup>  
and Liang Wang<sup>1</sup>

<sup>1</sup>Institute of Automation, Chinese Academy of Science, Beijing 100190, China.

<sup>2</sup>University of California, Santa Barbara 93106, USA.

<sup>3</sup>Alibaba Group, DAMO Institute, Beijing 695571, China.

<sup>4</sup>Institute of Information Engineering, Chinese Academy of Science, Beijing 695571, China.

Contributing authors: [shu.wu@nlpr.ia.ac.cn](mailto:shu.wu@nlpr.ia.ac.cn); [zekunli@cs.ucsb.edu](mailto:zekunli@cs.ucsb.edu);  
[yunyue.su@ia.ac.cn](mailto:yunyue.su@ia.ac.cn); [cuizeyu15@gmail.com](mailto:cuizeyu15@gmail.com); [zhxy333@gmail.com](mailto:zhxy333@gmail.com);  
[liang.wang@nlpr.ia.ac.cn](mailto:liang.wang@nlpr.ia.ac.cn);

## Abstract

Factorization machine (FM) is a prevalent approach to modelling pairwise (second-order) feature interactions when dealing with high-dimensional sparse data. However, on the one hand, FMs fail to capture higher-order feature interactions suffering from combinatorial expansion. On the other hand, taking into account interactions between every pair of features may introduce noise and degrade the prediction accuracy. To solve these problems, we propose a novel approach, the graph factorization machine (GraphFM), which naturally represents features in the graph structure. In particular, we design a mechanism to select beneficial feature interactions and formulate them as edges between features. Then the proposed model, which integrates the interaction function of the FM into the feature aggregation strategy of the graph neural network (GNN), can model arbitrary-order feature interactions on graph-structured features by stacking layers. Experimental results on several real-world datasets demonstrate the rationality and effectiveness of our proposed approach. The code and data are available at <https://github.com/CRIPAC-DIG/GraphCTR>.

**Keywords:** Feature interaction, Factorization machines, Graph neural network, Recommender system, Deep learning.

## 1 Introduction

Predictive analytics is a fundamental task in machine learning (ML) and data mining (DM), which involves using input features to predict an output target, such as a real value for regression or categorical labels for classification. This is particularly important for web applications, such as online advertising and recommender systems [1–3]. Distinct from continuous features which can be naturally found in images and audio, the features for web applications are mostly sparse and categorical. To accurately perform predictive analytics on these types of features, it is important to consider the interactions between them. As an example, consider a scenario in which we want to predict users' preferences for movies based on five categorical variables: (1) *Language* = {*English*, *Chinese*, *Japanese*, ...}, (2) *Genre* = {*action*, *fiction*, ...}, (3) *Director* = {*Ang Lee*, *Christopher Nolan*, ...}, (4) *Stars* = {*Bruce Lee*, *Leonardo DiCaprio*, ...} and (5) *Release Date* = {1991, 1992, ...}. To capture the impact of these feature interactions, a model might consider a 3rd-order cross feature such as (*Genre* = *fiction*, *Director* = *Christopher Nolan*, *Starring* = *Leonardo DiCaprio*) or (*Language* = *Chinese*, *Genre* = *action*, *Starring* = *Bruce Lee*) as potentially indicating higher user preferences.

FM [4, 5] is a popular and effective method for modelling feature interactions, that involves learning a latent vector for each one-hot encoded feature and modelling the pairwise (second-order) interactions between them through the inner product of their respective vectors. FM has been widely used in the field of recommender systems and click-through rate predictions due to its simplicity and effectiveness. However, because FM considers all feature interactions, it has two main drawbacks.

One of the main limitations of FM is that it is not able to capture higher-order feature interactions, which are interactions between three or more features. While higher-order FM (HOFM) has been proposed [4, 5] as a way to address this issue, it suffers from high complexity due to the combination expansion of higher-order interactions. This makes HOFM difficult to use in practice. To address the limitations of FM in capturing higher-order feature interactions, several variants have been proposed that utilize deep neural networks (DNNs) [2, 6–8]. Factorisation machine-supported neural networks (FNNs) [6] apply DNNs on top of pretrained factorization machines to model high-order interactions. Neural factorization machines (NFM) [2] design a bi-interaction layer to learn pairwise feature interactions and apply DNNs to learn higher-order interactions. Wide&Deep [7] introduces a hybrid architecture containing both shallow and deep components to jointly learn low-order and high-order feature interactions. DeepFM [8] similarly combines a shallow component with a deep component to learn both types of interactions. While

these DNN-based models can effectively learn high-order feature interactions, in an implicit, bitwise manner. Consequently, they may lack the ability to provide persuasive rationales for their outputs.

In addition to not being able to effectively capture higher-order feature interactions, FM is also suboptimal because it considers the interactions between every pair of features, even if some of these interactions may not be beneficial for prediction [9, 10]. These unhelpful feature interactions can introduce noise and lead to overfitting, as they do not provide useful information but make it harder to train the model. For example, in the context of predicting movie preferences, the feature interactions between *Language* and *Release Date* might not be relevant and, therefore, might not provide useful information for prediction. Ignoring these irrelevant feature interactions can improve model training. To solve this problem, the Attentional Decomposition Machine (AFM) [11] distinguishes the importance of the factorized interaction by reweighing each cross-feature using the attentional score [12], i.e. the influence of useless cross-features is reduced by assigning lower weights. However, it requires a predefined maximum order, which limits the potential of the model to find discriminative crossing features. Therefore, the Adaptive Factorization Network (AFN) [13] used a logarithmic neural transformation layer composed of multiple vector-wise logarithmic neurons to automatically learn the powers (i.e. the order) of features in a potentially useful combination, thereby adaptively learning cross-features and their weights from the data.

Currently, Graph Neural Networks (GNN) [14–16] have recently emerged as an effective class of models for capturing high-order relationships between nodes in a graph and have achieved state-of-the-art results on a variety of tasks such as computer vision [17], neural language processing [18, 19], and recommender systems [20, 21]. At their core, GNNs learn node embeddings by iteratively aggregating features from neighboring nodes, layer by layer. This allows them to explicitly encode high-order relationships between nodes in the embeddings. GNNs have shown great potential for modelling high-order feature interactions for click-through rate prediction. Fi-GNN [22] proposed connecting each pair of features and treating the multi-field features as a fully-connected graph, using a gated graph neural network (GGNN) [23] to model feature interactions on the graph. Graph factorizer machine (GFM) [24] utilizes FM to aggregate second-order neighbour messages, and utilizes the superposition of multiple GFM layers to aggregate higher-order neighbour messages to achieve multi-order interactions from neighborhoods for recommendation. Graph-Convolved Factorization Machines (GCFM) [25] developed the graph-convolved feature crossing (GCFC) layer to traverse all features for each input example and leveraged the features of each sample to compute the corresponding multi-feature interaction graph and propagated its influence on other features. KD-DAGFM [26] proposes a directed acyclic graph based model, that can be aligned with the DP [27] algorithm to improve the knowledge distillation (KD)[28] capability. However, not all feature interactions are beneficial, and

GNNs rely on the assumption that neighboring nodes share similar features, which may not always hold in the context of feature interaction modelling.

In summary, when dealing with feature interactions, FM suffers intrinsic drawbacks. We thus propose a novel model graph factorization machine (GraphFM), which takes advantage of the GNN to overcome the problems of FM for feature interaction modelling. By treating features as nodes and feature interactions as the edges between them, the selected beneficial feature interactions can be viewed as a graph. We thus devise a novel technique to select beneficial feature interactions, which also involves inferring the graph structure. Then, we adopt an attentional aggregation strategy to aggregate these selected beneficial interactions to update the feature representations. Specifically, to accommodate the polysemy of feature interactions in different semantic spaces, we utilize a multi-head attention mechanism [16, 29]. Each layer of our proposed model produces higher-order interactions based on the existing layers; thus, the highest-order interactions are determined by layer depth. Since our proposed approach selects the beneficial feature interactions and models them in an explicit manner, it has high efficiency in analysing high-order feature interactions and thus provides rationales for the model outcome. Through extensive experiments conducted on the CTR benchmark and recommender system datasets, we verify the rationality, effectiveness, and interpretability of our proposed approach.

Overall, the main contributions of this work are threefold: (1) We analyse the shortcomings and strengths of FM and GNN in modelling feature interactions. To solve their problems and leverage strengths, we propose a novel model GraphFM for feature interaction modelling. (2) By treating features as nodes and their pairwise feature interactions as edges, we bridge the gap between the GNN and FM, and make it feasible to leverage the strength of the GNN to solve the FM problem. (3) Extensive experiments are conducted on the CTR benchmark and recommender system datasets to evaluate the effectiveness and interpretability of our proposed method. We show that GraphFM can provide persuasive rationales for feature interaction modelling and prediction-making processes.

## 2 Related Work

In this work, we proposed a GNN-based approach for modelling feature interactions. We design a feature interaction selection mechanism, which can be seen as learning the graph structure by viewing the feature interactions as edges between features. In this section, we review three lines of research that are relevant to this work: 1) techniques for learning feature interactions, 2) GNNs, and 3) graph structure learning methods.

### 2.1 Feature Interaction Modelling

Modelling feature interactions is a crucial aspect of predictive analytics and has been widely studied in the literature. FM [4] is a popular method that

learns pairwise feature interactions through vector inner products. Since its introduction, several variants of FM have been proposed, including field-aware factorization machine (FFM) [30] which takes into account field information and introduces field-aware embeddings, and AFM [11], which considers the weight of different second-order feature interactions. FmFM [31] modelled the interactions of field pairs as a matrix and utilized a kernel product to capture field interactions. However, these approaches are limited to modelling second-order interactions, which may not be sufficient in some cases.

As deep neural networks (DNNs) have proven successful in a variety of fields, researchers have begun using them to learn high-order feature interactions due to their deeper structures and nonlinear activation functions. The general approach is to concatenate the representations of different feature fields and feed them into a DNN to learn the high-order feature interactions. Factorization-machine supported Neural Networks (FNNs) [6] used pretrained factorization machines to create field embeddings before applying a DNN, while product-based neural networks (PNNs) [32] model both second-order and high-order interactions through the use of a product layer between the field embedding layer and the DNN layer. Like PNNs, neural factorization machines (NFM) [2] also use a separate layer to model second-order interactions, but they use a Bi-Interaction Pooling layer instead of a product layer and follow it with summation rather than concatenation. Other approaches to modelling second-order and high-order interactions jointly use hybrid architectures. The Wide&Deep [7] and DeepFM [8] involve modelling low-order interactions and deep modelling high-order interactions. However, similar to other DNN-based approaches, these models learn high-order feature interactions in an implicit, bit-wise manner and may lack transparency in their feature interaction modelling process and model outputs. As a result, some studies have attempted to learn feature interactions in an explicit fashion through the use of specifically designed networks. Deep&Cross [33] introduces a CrossNet that takes the outer product of features at the bit level, while xDeepFM [34] uses a compressed interaction network(CIN) to take the outer product at the vector level and then compresses the resulting feature maps to update the feature representations. However, xDeepFM has been found to have issues with generalizability and scalability, and it has relatively high complexity due to its consideration of all pairwise bit-level interactions. DCNV2 [35] similarly used CIN to learn efficient explicit and implicit feature intersections, but it additionally leverages low-rank techniques to approximate feature crosses in subspaces for better performance and latency trade-offs.

More recently, several studies have attempted to use attention mechanisms to model feature interactions in a more interpretable way. HoAFM [36] updates feature representations by attentively aggregating the representations of co-occurring features, while AutoInt [37] uses a multi-head self-attention mechanism to explicitly model feature interactions. InterHAt [38] is another model that uses an attentional aggregation strategy with residual connections to learn feature representations and model feature interactions. However, even

with the use of attention mechanisms to account for the weight of each pair of feature interactions, aggregating all interactions together can still introduce noise and degrade the prediction accuracy. To address these issues, some recent studies have attempted to identify beneficial feature interactions automatically. AutoFIS [39] is a two-stage algorithm that uses a gate operation to search and model beneficial feature interactions, but there is a loss of information between the stages, and the modelling process is not interpretable. AFN [13] used a logarithmic neural network to adaptively learn high-order feature interactions, and the SIGN [10] utilized mutual information to detect beneficial feature interactions and a linear aggregation strategy to model them. However, these approaches may not be expressive or interpretable enough.

## 2.2 Graph Neural Networks

A graph is a kind of data structure that reflects a set of entities (nodes) and their relationships (edges). Graph neural networks (GNNs), as deep learning architectures for graph-structured data, have attracted increasing attention. The concept of GNNs was first proposed by [40], and further elaborated in [41]. Currently, most of the prevailing GNN models follow the neighbourhood aggregation strategy, that is, to learn the latent node representations by aggregating the features of neighbourhoods layer by layer. The high-order relations between nodes can be modelled explicitly by stacking layers. Gated graph neural networks (GGNN) [23] use GRUs [42] to update node representations based on aggregated neighborhood feature information. However based on graph spectral theory [43], the learning process of graph convolutional networks (GCNs) [14] can also be considered a mean-pooling neighborhood aggregation. GraphSAGE [15] concatenates node features and introduces three mean-/max/LSTM aggregators to pool neighborhood information. Graph attention network (GAT) [16] incorporates an attention mechanism to measure the weights of neighbors when aggregating neighborhood information of a node.

Due to its strength in modelling relations on graph-structured data, the GNN has been widely applied to various applications, such as neural machine translation [44], semantic segmentation [45], image classification [46], situation recognition [17], recommendation [20, 21, 47], script event prediction [48], and fashion analysis [49]. The Fi-GNN [22] is the first attempt to exploit the GNN for feature interaction modelling. It first proposes connecting all the feature fields; thus, the multi-field features can be treated as a fully-connected graph. Then the GGNN [23] is utilized to model high-order feature interactions on the feature graph. KD-DAGFM [26] uses knowledge distillation and proposes a lightweight student model, namely, directed acyclic graph FM, for learning arbitrarily explicit high-order feature interactions from teacher networks. Other graph-based work, like GFM [24] utilized the popular factorization machine to effectively aggregate multi-order interactions in the GNN. In addition, GCFM [25] uses the multifilter graph-convolved feature crossing (GCFC) layer to learn the neighbor feature interactions.

Nevertheless, the GNN was originally designed for graph classification tasks, and is based on the assumption that neighbors share similar features. As a result, the GNN inherits unnecessary and unsuitable operations for feature interaction modelling. Our proposed model GraphFM introduces the interaction function of FM into the neighborhood aggregation strategy of the GNN to effectively capture the beneficial factorized interaction.

### 3 Preliminaries

In this section, we first introduce the background of feature embeddings, which are fundamental for most feature interaction models based on deep learning. To help understand our proposed model GraphFM, which is based on FMs and GNNs, we then describe these two lines of work.

#### 3.1 Feature Embeddings

In many real-world predictive tasks, such as CTR prediction, input instances consist of both sparse categorical and numerical features. Traditionally, we represent each input instance as a sparse vector:

$$\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n], \quad (1)$$

where  $n$  is the number of *feature fields* and  $\mathbf{x}_i$  is the representation of the  $i$ -th feature field (aka *feature*). Since categorical features are very sparse and high-dimensional, a common way is to map them into a low-dimensional latent space. Specifically, a categorical feature  $\mathbf{x}_i$  is mapped to dense embedding  $\mathbf{e}_i \in \mathbb{R}^d$  as:

$$\mathbf{e}_i = \mathbf{V}_i \mathbf{x}_i, \quad (2)$$

where  $\mathbf{V}_i$  denotes the embedding matrix of field  $i$ .

For a numerical feature  $\mathbf{x}_j$  which is a scalar  $x_j$ , we also represent it in the  $d$ -dimensional embedding space:

$$\mathbf{e}_j = \mathbf{v}_j x_j \quad (3)$$

where  $\mathbf{v}_j$  is the embedding vector for the numerical field  $j$ . Therefore, we can obtain a feature embedding matrix consisting of these feature embeddings:

$$\mathbf{E} = [\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_n]^\top. \quad (4)$$

#### 3.2 Factorization Machines

The factorization machine (FM) was originally proposed for collaborative recommendation [4, 5]. It estimates the target by modelling all interactions

between each pair of features:

$$\hat{y}_{\text{FM}} = \langle \mathbf{w}, \mathbf{x} \rangle + \sum_{i_2 > i_1}^n \langle \mathbf{e}_{i_1}, \mathbf{e}_{i_2} \rangle, \quad (5)$$

where  $\langle \cdot, \cdot \rangle$  denotes the inner product operation. Intuitively, the first term  $\langle \mathbf{w}, \mathbf{x} \rangle$  is the linear regression of raw features, and the second term is the sum of all pairwise interactions, i.e., inner products of feature embeddings.

In principle, FMs can be extended to higher-order feature combinations [4, 5]. Let  $k \in \{2, \dots, K\}$  denote the order or degree of feature interactions considered and  $\mathbf{e}_i^{(k)}$  denote the embedding of feature  $i$  for order  $k$ , the  $K$ -order higher-order FM (HOFM) can be defined as

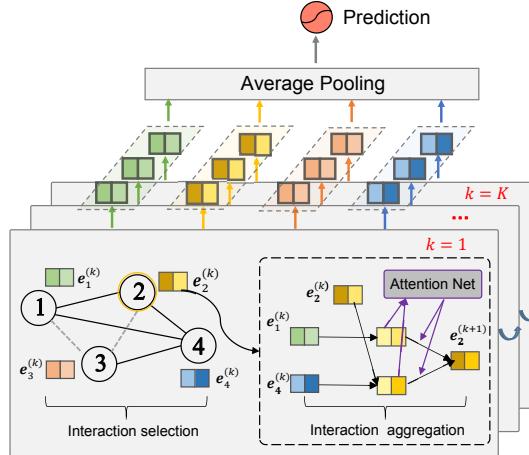
$$\begin{aligned} \hat{y}_{\text{HOFM}} = & \langle \mathbf{w}, \mathbf{x} \rangle + \sum_{i_2 > i_1}^n \langle \mathbf{e}_{i_1}^{(2)}, \mathbf{e}_{i_2}^{(2)} \rangle + \sum_{i_3 > i_2 > i_1}^n \langle \mathbf{e}_{i_1}^{(3)}, \mathbf{e}_{i_2}^{(3)}, \mathbf{e}_{i_3}^{(3)} \rangle \\ & + \dots + \sum_{i_K > \dots > i_1}^n \langle \mathbf{e}_{i_1}^{(K)}, \dots, \mathbf{e}_{i_K}^{(K)} \rangle, \end{aligned} \quad (6)$$

where  $\langle \mathbf{e}_{i_1}^{(K)}, \dots, \mathbf{e}_{i_K}^{(K)} \rangle = \text{sum}(\mathbf{e}_{i_1}^{(K)} \odot \dots \odot \mathbf{e}_{i_K}^{(K)})$  (sum of element-wise products). Since all the feature interactions of order up to  $K$  are included, its time complexity increases exponentially, resulting in high computational complexity. Considering that not all feature interactions are beneficial, FMs have trouble modelling higher-order feature interactions in terms of both efficiency and effectiveness.

Although several recent studies have enhanced FMs with DNNs to model higher-order feature interactions, like NFM [2] and DeepFM [8], they model higher-order feature interactions in an implicit manner, and lack persuasive rationales for model outcomes.

### 3.3 Graph Neural Networks

Given that a graph  $G = \{V, E\}$  denotes a graph, GNNs learn the representation vectors of nodes by exploring their correlations with neighboring nodes. The modified GNNs follow a neighborhood aggregation strategy, where we iteratively update the representation of a node by aggregating the features of its neighbors [50]. After  $k$  iterations of aggregation, a node's representation encodes its interaction with neighbors within  $k$  hops. The choice of aggregation strategy for GNNs is crucial. A number of architectures have been proposed. In this work, we adopt the attentional aggregation strategy in [16], which will be further elaborated upon in Section 4.



**Fig. 1** Overview of GraphFM. The input features are modelled as a graph, where nodes are feature fields, and edges are interactions. At each layer of GraphFM, the edges (beneficial interactions) are first selected by the *interaction selection* component. Then these selected feature interactions are aggregated via the attention network to update the feature embeddings in the *interaction aggregation* component. The learned feature embeddings at every layer are used for the final prediction jointly.

## 4 Graph Factorization Machine

### 4.1 Model Overview

An overview of GraphFM is shown in Fig. 1. Each input instance (multi-field feature) is represented as a graph where the nodes are feature fields, and the edges are interactions [10, 22]. Note that we use nodes and features, edges, and interactions interchangeably in this paper. GraphFM updates feature representations layer by layer. The feature embeddings described in Section 3.1 are taken as the initial feature embeddings of GraphFM, i.e.,  $\mathbf{e}_i^{(1)} = \mathbf{e}_i$ , where  $\mathbf{e}_i^{(k)}$  represents the updated feature embeddings at the  $k$ -th layer. Since no edge information is given, we need to select the edges (beneficial interactions) by the interaction selection component first. Then, we aggregate these selected feature interactions to update the feature embeddings in the neighborhood aggregation component. Within each  $k$ -th layer, we are able to select and model only the beneficial  $k$ -th order feature interactions and encode these factorized interactions into feature representations. Finally, these learned feature embedding encoded interactions of order up to  $K$  are concatenated to make the final prediction.

There are two main components in each layer of GraphFM. Next, we will introduce them in detail. As we focus on describing the detailed mechanism at every single layer, we omit the layer index  $k$  if not necessary.

### 4.2 Interaction Selection

To select beneficial pairwise feature interactions, we devised an interaction selection mechanism. This can also be viewed as inferring the graph structure,

which predicts the links between nodes. However, the graph structure  $G = \{V, E\}$  is discrete, where an edge  $(v_i, v_j) \in E$  linking two nodes is either present or absent. This makes the process non-differentiable; therefore, it cannot be directly optimized with gradient-descent-based optimization techniques.

To overcome this limitation, we replace the edge set  $E$  with weighted adjacency  $\mathbf{P}$ , where  $p_{ij}$  is interpreted as the probability of  $(v_i, v_j) \in E$ , which also reflects how beneficial their interaction is. Notably, we learn different graph structures  $\mathbf{P}^{(k)}$  at each  $k$ -th layer. Compared with using a fixed graph at each layer, we have more efficiency and flexibility in enumerating the beneficial higher-order feature interaction by this means. More specifically, by using a fixed graph structure at each layer, we can only obtain a fixed set of feature interactions. However, with the adaptive learned graph structure at each layer, the model is capable of modelling any potential feature interactions.

#### 4.2.1 Metric Function

We aim to design a metric function between each pair of feature interactions to measure whether they are beneficial. Formally, the weight  $p_{ij}$  of an edge  $(v_i, v_j)$  is computed via a metric function  $f_s(\mathbf{e}_i, \mathbf{e}_j)$ . Here, we adopt a neural matrix factorization (NMF) [1] based function to estimate the edge weight. Formally, a multi-layer perception (MLP) with one hidden layer is used to transform the element-wise product of these feature vectors to a scalar:

$$f_s(\mathbf{e}_i, \mathbf{e}_j) = \sigma(\mathbf{W}_2^s \delta(\mathbf{W}_1^s(\mathbf{e}_i \odot \mathbf{e}_j) + \mathbf{b}_1^s) + \mathbf{b}_2^s), \quad (7)$$

where  $\mathbf{W}_1^s$ ,  $\mathbf{W}_2^s$ ,  $\mathbf{b}_1^s$ , and  $\mathbf{b}_2^s$  are the parameters of the MLP.  $\delta(\cdot)$  and  $\sigma(\cdot)$  are ReLU and sigmoid activation functions, respectively. It should be noted that  $f_s$  is invariant to the order of its input, i.e.,  $f_s(\mathbf{e}_i, \mathbf{e}_j) = f_s(\mathbf{e}_j, \mathbf{e}_i)$ . Therefore, the estimated edge weights are identical for the same pair of nodes. Such continuous modelling of the graph structure enables backpropagation of the gradients. Since we do not have a ground-truth graph structure, the gradients come from the errors between the model output with and the target.

Intuitively, we treat the element-wise product of each pair of feature embeddings as a term and estimate its weight using the MLP. One can also choose the Euclidean distance [51] or other distance metrics.

#### 4.2.2 Graph Sampling

From the estimated edge weighted matrix  $\mathbf{P}^{(k)}$  at each layer, we then sample the beneficial feature interactions to sample the neighborhood for each feature field. In this work, we uniformly sample a fixed-size set of neighbors. For each feature node  $v_i$  at the  $k$ -th layer, we select  $m_k$  edges according to the first  $m_k$

elements of  $\mathbf{P}^{(k)}[i, :]$ , which can be illustrated as follows:

$$\begin{aligned} & \text{for } i = 1, 2, \dots, n \\ & \text{idx}_i = \text{argtop}_{m_k}(\mathbf{P}^{(k)}[i, :]) \\ & \mathbf{P}^{(k)}[i, -\text{idx}] = 0, \end{aligned} \quad (8)$$

where  $\mathbf{P}^{(k)}[i, :]$  denotes the  $i$ -th column of matrix  $\mathbf{P}^{(k)}$  at the  $k$ -th layer, and  $\mathbf{P}^{(k)}[i, -\text{idx}_i]$  contains a subset of columns of  $\mathbf{P}^{(k)}$  that are not indexed by  $\text{idx}_i$ .  $\text{argtop}_{m_k}$  is an operator that selects the  $m_k$ -most important nodes for query node  $i$ . We only retain these  $m_k$  feature nodes, and the others are masked. Thus the neighborhood set of node  $v_j$  is defined as:

$$\mathcal{N}_i^{(k)} = \left\{ v_j \mid p_{ij}^{(k)} > 0, j = 1, 2, \dots, n \right\}. \quad (9)$$

Practically speaking, we found that our approach could achieve high performance when  $k = 3$ , and  $m_1$  equals the number of feature fields, which means that in the first layer, we model all pairs of feature interactions.

It is worth mentioning that we also tried to set a threshold to select the edges in the graph, i.e., setting a minimum value for the edge probability of cutting edges off. However, the performance is not as good as that of using a fixed-degree graph. This is reasonable because the edge weights of different nodes' neighbors are at different scales. Setting a single threshold on all the nodes will lead to the situation in which the numbers of nodes' neighbors vary greatly. Some nodes will have barely any adjacent nodes after cutting off, while some may still have many.

### 4.3 Interaction Aggregation

Since we have selected the beneficial feature interactions, or in other words, learned the graph structure, we perform the interaction (neighborhood) aggregation operation to update the feature representations.

For a target feature node  $v_i$ , when aggregating its beneficial interactions with neighbors, we also measure the attention coefficients of each interaction. To measure the attention coefficients, we use a learnable projection vector  $\mathbf{a}$  and apply a LeakyReLU non-linear activation function. Formally, the coefficients are computed as:

$$c_{ij} = \text{LeakyReLU}(\mathbf{a}^\top (\mathbf{e}_i \odot \mathbf{e}_j)). \quad (10)$$

This indicates the importance of the interactions between feature  $v_i$  and feature  $v_j$ .

Note that we only compute  $c_{ij}$  for nodes  $j \in \mathcal{N}_i$ , where  $\mathcal{N}_i$  denotes the neighborhood of node  $v_i$ , which is also the set of features whose interactions with  $v_i$  are beneficial. To make coefficients easily comparable across different

feature nodes, we normalized them across all choices of  $j$  using a softmax function:

$$\alpha_{ij} = \frac{\exp(c_{ij})}{\sum_{j' \in \mathcal{N}_i} \exp(c_{ij'})}. \quad (11)$$

Once the normalized attention coefficients are obtained, we compute the linear combination of these feature interactions with nonlinearity as the updated feature representations:

$$\mathbf{e}'_i = \sigma \left( \sum_{j \in \mathcal{N}_i} \alpha_{ij} p_{ij} \mathbf{W}_a (\mathbf{e}_i \odot \mathbf{e}_j) \right), \quad (12)$$

where  $\alpha_{ij}$  measures the attention coefficients of each feature interaction between feature  $i$  and  $j$ , while  $p_{ij}$  represents the probability of this feature interaction being beneficial. The attention coefficient  $\alpha_{ij}$  is calculated by the soft attention mechanism, while  $p_{ij}$  is calculated by the hard attention mechanism. By multiplying them together, we control the information of selected feature interactions and make the parameters in the interaction selection component trainable with gradient back-propagation.

To capture the diverse polysemy of feature interactions in different semantic subspaces [38] and stabilize the learning process [16, 29], we extend our mechanism to employ multi-head attention. Specifically,  $H$  independent attention mechanisms update Equation 12, and then these features are concatenated, resulting in the following output feature representation:

$$\mathbf{e}'_i = \parallel_{h=1}^H \sigma \left( \sum_{j \in \mathcal{N}_i} \alpha_{ij}^h p_{ij} \mathbf{W}_a^h (\mathbf{e}_i \odot \mathbf{e}_j) \right), \quad (13)$$

where  $\parallel$  denotes the concatenation,  $\alpha_{ij}^h$  is the normalized attention coefficient computed by the  $h$ -th attention mechanism, and  $\mathbf{W}_a^h$  is the corresponding linear transformation matrix. One can also choose to employ average pooling to update the feature representations:

$$\mathbf{e}'_i = \sigma \left( \frac{1}{H} \sum_{h=1}^H \sum_{j \in \mathcal{N}_i} \alpha_{ij}^h p_{ij} \mathbf{W}_a^h (\mathbf{e}_i \odot \mathbf{e}_j) \right). \quad (14)$$

#### 4.4 Prediction and Optimization

The output of each  $k$ -th layer, is a set of  $n$  feature representation vectors encoding feature interactions of order up to  $k$ , namely  $\{\mathbf{e}_1^{(k)}, \mathbf{e}_2^{(k)}, \dots, \mathbf{e}_n^{(k)}\}$ . Since the representations obtained in different layers encode the interactions of different orders, they have different contributions to the final prediction. As such, we concatenate them to constitute the final representation of each

**Table 1** Statistics of the evaluation datasets.

Dataset	#Instances	#Fields	#Features (sparse)
Criteo	45,840,617	39	998,960
Avazu	40,428,967	23	1,544,488
MovieLens-1M	739,012	7	3,529

feature [20]:

$$\mathbf{e}_i^* = \mathbf{e}_i^{(1)} \parallel \cdots \parallel \mathbf{e}_i^{(K)}, \quad (15)$$

Finally, we employ average pooling on the vectors of all features to obtain a graph-level output and use a projection vector  $\mathbf{p}$  to make the final prediction:

$$\mathbf{e}^* = \frac{1}{n} \sum_{i=1}^n \mathbf{e}_i^*, \quad (16)$$

$$\hat{y} = \mathbf{p}^\top \mathbf{e}^*. \quad (17)$$

GraphFM can be applied to various prediction tasks, including regression, classification, and ranking. In this work, we conduct experiments on CTR prediction, a binary classification task. We thus use log loss as the loss function:

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^N y_i \log \sigma(\hat{y}_i) + (1 - y_i) \log(1 - \sigma(\hat{y}_i)), \quad (18)$$

where  $N$  is the total number of training instances, and  $\sigma$  denotes the sigmoid function.  $y_i$  and  $\hat{y}_i$  denote the label of instance  $i$  and the prediction of GraphFM, respectively. The model parameters are updated using Adam[52].

## 5 Experiments

This section presents an empirical investigation of the performance of GraphFM on two CTR benchmark datasets and a recommender system dataset. The experimental settings are described, followed by comparisons with other state-of-the-art methods. An ablation study is also conducted to verify the importance of each component of the model and evaluate its performance under different hyperparameter settings. Finally, the question of whether GraphFM can provide interpretable explanations for its predictions is examined.

### 5.1 Experimental Settings

Our experiments are conducted on three real-world datasets, two CTR benchmark datasets, and one recommender system dataset. Details of these datasets are listed in Table 1. The data preparation follows the strategy in [26]. We randomly split all the instances at 8:1:1 for training, validation, and testing. We adopt the two most popular metrics, **AUC** and **Logloss** to measure the probability that one prediction diverges from the ground truth.

### 5.1.1 Baselines

We compare GraphFM with four classes of state-of-the-art methods: (A) the linear approach that only uses individual features; (B) FM-based methods that consider second-order feature interactions; (C) DNN-based methods that model high-order feature interactions; and (D) aggregation-based methods that update features' representation and model their high-order feature interactions via an aggregation strategy.

The models associated with their respective classes are listed as follows:

- **LR** (A) refers to linear/logistics regression, which can only model linear interactions.
- **FM** [5] (B) is the official FM implementation, which can only model second-order interactions.
- **AFM** [11] (B) is an extension of FM that considers the weights of different second-order feature interactions by using attention mechanisms.
- **AFN** [13] (B) learns arbitrary-order feature interactions adaptively from data, instead of explicitly modelling all the cross features within a fixed maximum order.
- **FmFM** [31] (B) uses a field matrix between two feature vectors to model their interactions and learns the matrix separately for each field pair.
- **NFM** [2] (C) devises a bi-interaction layer to model second-order interactions and uses a DNN to introduce nonlinearity and model high-order interactions.
- **HOFM** (C) [53] is the implementation of the higher-order FM [53]. It is a linear model.
- **DeepCrossing** [54] (C) utilizes a DNN with residual connections to model non-linear feature interactions in an implicit manner.
- **CrossNet** [33] (C) is the core of the Deep&Cross model, which models feature interactions explicitly by taking the outer product of the concatenated feature vector at the bit-wise level.
- **xDeepFM** [34] (C) takes the outer product of the stacked feature matrix at a vector-wise level to explicitly model feature interactions. ANNs can also be combined with DNNs, which model implicit and explicit interactions simultaneously.
- **DCNV2** [35] (C) utilizes a cross network from the DCN to learn explicit and bounded-degree cross features.
- **AutoInt** [37] (D) uses a self-attention network to learn high-order feature interactions explicitly. It can also be seen as performing the multi-head attention mechanism [29] on a fully-connected graph.
- **Fi-GNN** [22] (D) models the features as a fully-connected graph and utilizes a gated graph neural network to model feature interactions.
- **InterHAt** [38] (D) utilizes an attention mechanism to aggregate features, which are then multiplied by the original features to produce higher-order feature interactions.

**Table 2** Performance comparison of different methods on three datasets. The four model classes (A, B, C, D) are defined in Section 5.1.1. The last two columns are average improvements of our proposed model GraphFM compared with corresponding base models (“+”: increase, “-”: decrease). We highlight the best performances on each dataset. Further analysis is provided in Section 5.2.

Model	Criteo		Avazu		MovieLens-1M	
	AUC	LogLoss	AUC	LogLoss	AUC	LogLoss
LR	0.7820	0.4695	0.7560	0.3964	0.7716	0.4424
FM [4]	0.7836	0.4700	0.7706	0.3856	0.8252	0.3998
AFM[11]	0.7938	0.4584	0.7718	0.3854	0.8227	0.4048
AFN [13]	0.8079	0.4433	0.7786	0.3799	0.8771	0.4721
FmFM [31]	0.8083	0.4434	0.7746	0.3859	0.8821	0.3279
NFM [2]	0.7957	0.4562	0.7708	0.3864	0.8357	0.3883
HOFM [53]	0.8005	0.4508	0.7701	0.3854	0.8304	0.4013
DeepCrossing [54]	0.8009	0.4513	0.7643	0.3889	0.8448	0.3814
CrossNet [33]	0.7907	0.4591	0.7667	0.3868	0.7968	0.4266
xDeepFM [34]	0.8009	0.4517	0.7758	0.3829	0.8286	0.4108
DCNV2 [35]	0.8074	0.4436	0.7666	0.3865	0.8833	0.4885
AutoInt [37]	0.8084	0.4427	0.7781	0.3795	0.8823	0.3463
Fi-GNN [22]	0.8077	0.4413	0.7778	0.3811	0.8792	0.3537
InterHAt [38]	0.8076	0.4446	0.7758	0.3860	0.8769	0.3591
GraphFM (ours)	<b>0.8091</b>	<b>0.4399</b>	<b>0.7798</b>	<b>0.3781</b>	<b>0.8902</b>	<b>0.3259</b>

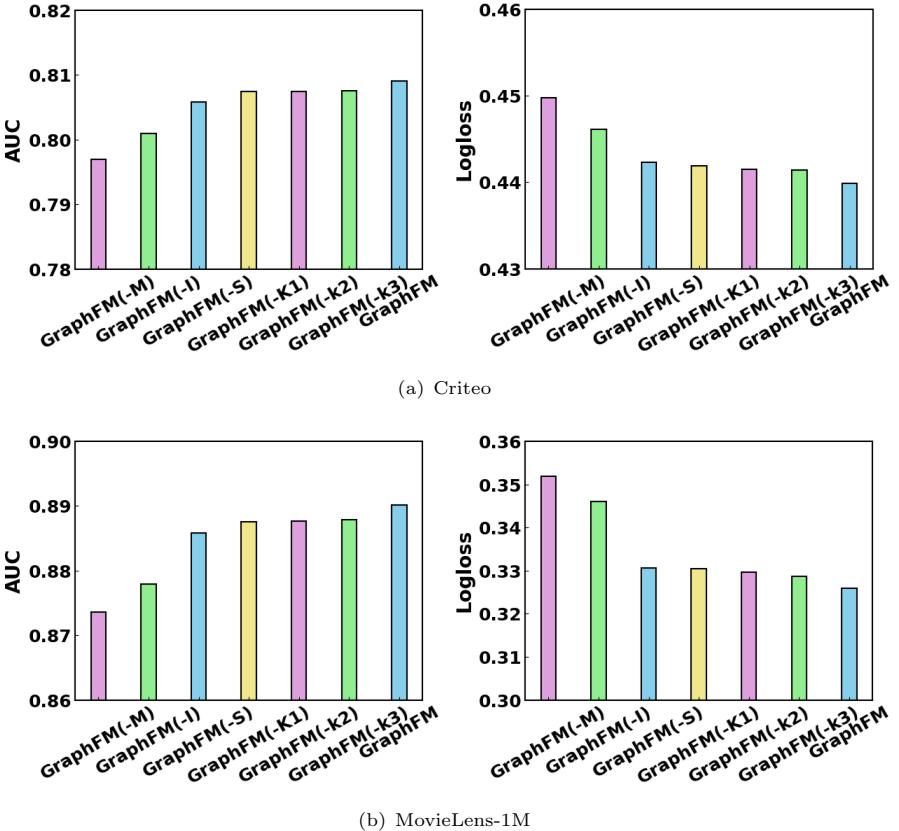
### 5.1.2 Implementation Details

We implement the method using TensorFlow [55] and Pytorch [56]. The feature embedding size is set as 16 for all methods. For a fair comparison, we set three layers for AutoInt, FiGNN, and GraphFM. There are two attention heads in AutoInt and GraphFM. The implementation of the other compared baselines follows [37] and [26]. The optimal hyper-parameters are found via a grid search.  $[m_1, m_2, m_3]$  are set as [39, 20, 5], [23, 10, 2], and [7, 4, 2] for the Criteo, Avazu and MovieLens-1M datasets respectively. We use Adam [52] to optimize all these models. The experiments were conducted over a server equipped with 8 NVIDIA Titan X GPUs.

## 5.2 Model Comparison

### 5.2.1 Evaluation of Effectiveness

The performance comparison of these methods on three datasets is presented in Table 2, from which we have the following observations. Our proposed GraphFM achieves the best performance among all four classes of methods on three datasets. The performance improvement of GraphFM compared with the three classes of methods (A, B, C) is especially significant, above the **0.01**-level.



**Fig. 2** Performance comparison of GraphFM with different components on the Criteo and MovieLens-1M datasets. Further analysis is provided in Section 5.3.

The aggregation-based methods including InterHAt, AutoInt, Fi-GNN and our GraphFM consistently outperform the other three classes of models, which demonstrates the strength of the aggregation strategy in capturing high-order relations. Compared with the strong aggregation-based baselines AutoInt and Fi-GNN, GraphFM still achieves large improvements in performance, especially on the MovieLens-1M dataset. The performance improvement on the other two datasets is also at the **0.001**-level, which can be regarded as significant for the CTR prediction task [3, 7, 8, 37, 57, 58]. This improvement can be attributed to its combination with FM, which introduces feature interaction operations, and the interaction selection mechanism, which selects and models only the beneficial feature interactions. GraphFM outperforms the compared baselines by the largest margin on the MovieLens-1M dataset, whose feature size is the smallest among the three datasets. This is likely because the feature embedding size is not large enough for the other two datasets.

### 5.3 Ablation Studies

To validate the effectiveness of each component in GraphFM, we conduct ablation studies and compare several variants of it:

- **GraphFM(-S):** *interaction selection* is the first component in each layer of GraphFM, which selects only the beneficial feature interactions and treats them as edges. As a consequence, we can model only these beneficial interactions with the next *interaction aggregation* component. To check the necessity of this component, we remove these components, so that all pairs of feature interactions are modelled as a fully-connected graph.
- **GraphFM(-I):** In the *interaction aggregation* component, we aggregate the feature interactions instead of the neighbors' features, as in the standard GNNs. To check its rationality, we test the performance of directly aggregating neighborhood features instead of the feature interactions with them.
- **GraphFM(-M):** In the *interaction aggregation* component, we use a multi-head attention mechanism to learn the diversified polysemy of feature interactions in different semantic subspaces. To check its rationality, we use only one attention head when aggregating.
- **GraphFM(- $K_i$ ):** Before obtaining the final representation of the feature, we concatenate and average the feature representation  $e_k$  output from layer  $k$ . To study the degree of contribution of the features learned at different layers to the results, we use the feature representation of the  $k$ -th layer for direct prediction, and there are  $K = 3$  layers.

The performances of GraphFM and these four variants are shown in Fig. 2. We observe that GraphFM outperforms all the ablative methods, which proves the necessity of all these components in our model. The performance of GraphFM(-M) suffers from a sharp decrease compared with that of GraphFM, proving that it is necessary to transform and aggregate the feature interactions in multiple semantic subspaces to accommodate polysemy. Note that although we did not present the statistics here, we also tested the influence of the number of attention heads  $H$ . The performance of using only one head, i.e., GraphFM(-M), is worse than that of using two, and more attention heads do not lead to improvement in performance but introduce much greater time and space complexity. GraphFM(-I) does not perform well either. This is reasonable, as without the interaction between features, neighborhood aggregation operation will only make neighboring features similar. As a consequence, no feature interactions are guaranteed to be captured. This interaction is also the most significant difference between GraphFM and GNN, and the resulting difference in terms of performance indicates that GraphFM is able to leverage the strength of FM to overcome the drawbacks of GNN in modelling feature interactions. GraphFM(-S) achieves slightly worse performance than GraphFM, demonstrating that selecting and modelling only the beneficial interactions instead of all of them can avoid noise and make it easier to train the model.

The GraphFM(- $K_i$ ) results show that each layer can learn features that benefit the results. Although different features can have a positive effect on task prediction, the difference in effect is not large, and the prediction results are worse than those obtained by combining all features, demonstrating the need to merge the features of the  $k$ -th layers.

## 5.4 Study of Neighborhood Sampled Size

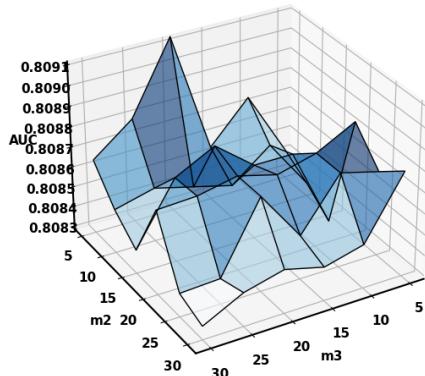
The number of selected neighbors for each feature node is an important hyper-parameter that controls the number of features with which each feature interacts. We thus investigate how the neighborhood sample size affects the model performance. As the total search space is too large, we only show the performance of our model with  $K = 3, m_1 = n$  and varying values of  $m_2$  and  $m_3$ . The results on three datasets are summarized in Fig. 3.

On the Criteo dataset, there are a total of 39 feature fields. We found that our model achieves the best performance with  $m_1 = 39$ , and  $m_2 \times m_3 = 100$ . The performances vary in the range of [0.8084, 0.8091], which proves that our model is quite robust, and not very sensitive to the size of the neighborhood sampled. On the Avazu dataset, the model performance peaks with  $m_1 = 23, m_2 = 10, m_3 = 2$  or  $m_2 = 15, m_3 = 4$ . On the MovieLens-1M dataset, the model performance peaks when  $m_2 \times m_3$  is approximately 9. We also found a diminishing trend for sampling larger or smaller neighborhoods. In other words, the optimal neighborhood sample size depends on the dataset size.

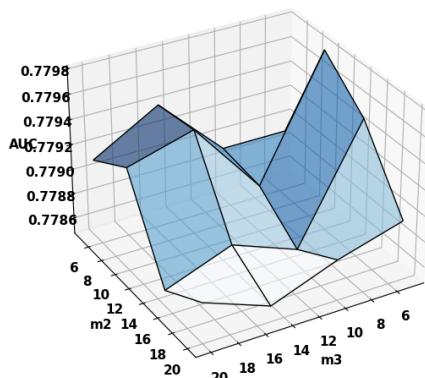
## 5.5 Visualization of the Interaction Graph

Since the features along with selected beneficial feature interactions are treated as a graph, they can provide human readable interpretations of the prediction. Here, we visualize heat maps of the estimated edge weights of two cherry-pick instances on the MovieLens-1M dataset in Fig. 4. We show the measured edge weights of each instance in the three layers, which select the order-2, order-3, and order-4 feature interactions. The positive edge weights indicate how beneficial feature interactions are. We set  $S_1 = 7, S_2 = 4$ , and  $S_3 = 2$ , which means that we only retain 7, 4, and 2 pairs of beneficial order-2, order-3, and order-4 feature interactions respectively. Therefore, there are only 7, 4, and 2 interaction feature fields for each feature field in each row for heat maps of order-2, order-3, and order-4, respectively. The axes represent feature fields (*Gender*, *Age*, *Occupation*, *Zipcode*, *ReleaseTime*, *WatchTime*, *Genre*). *Gender*, *Age*, *Occupation* and *Zipcode* are users' demographic information. Note that *Zipcode* indicates the users' place of residence. *textitReleaseTime* and *Gender* are the movie information. *WatchTime* (*Timestamp*) represents the time when users watched the movies.

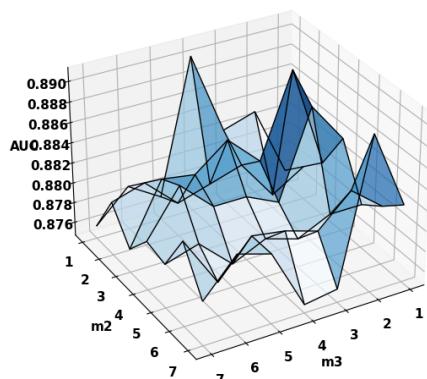
From the two instances in Fig. 4, we can obtain the following interesting observations. We find that in the first layer, which models the second order feature interactions, these feature fields are difficult to distinguish when selecting the beneficial interactions. This suggests that almost all the second-order



(a) Criteo

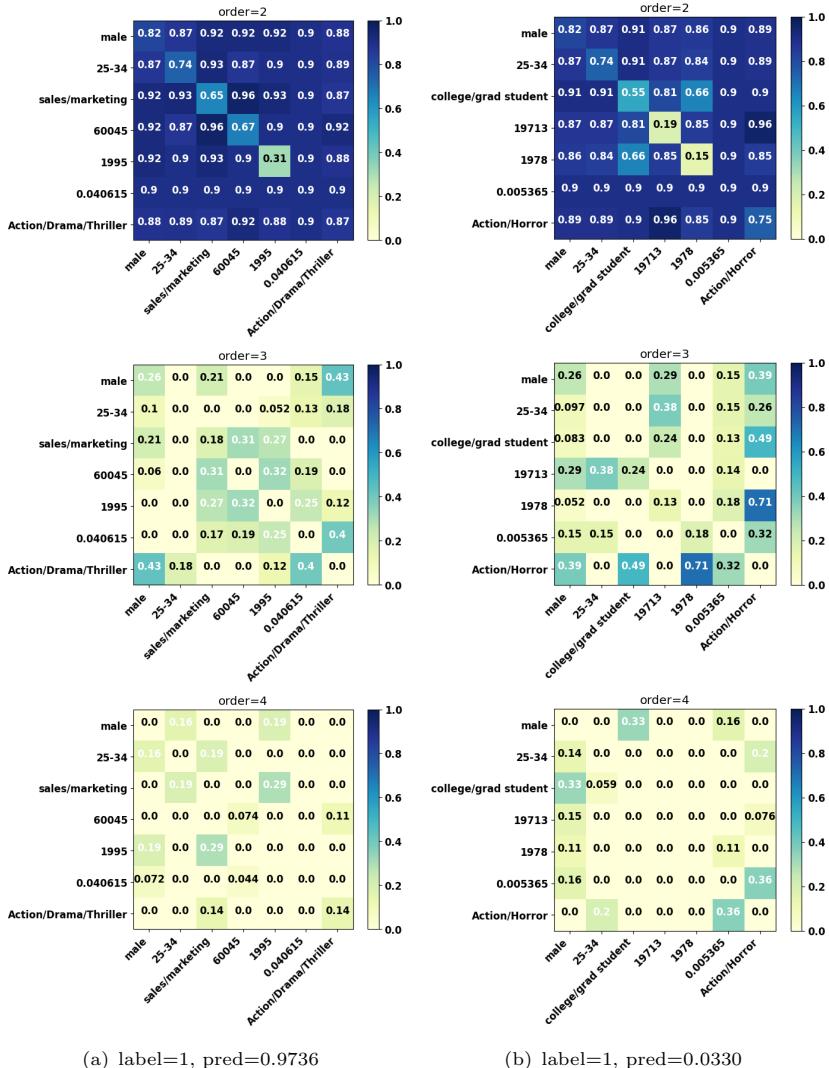


(b) Avazu



(c) MovieLens

**Fig. 3** Model performance with respect to the size of the sampled neighborhood, where the “neighborhood sample size” refers to the number of neighbors sampled at each depth for  $K = 3$  with  $m_1 = n$ , and  $m_2, m_3$  with varying values.



**Fig. 4** Heat maps of the estimated edge weights of correctly predicted instances (a) and incorrectly predicted instances (b) on the MovieLens-1M dataset, where positive edge weights indicate beneficial feature interactions. The axes represent feature fields (*Gender, Age, Occupation, Zipcode, ReleaseTime, WatchTime, Genre*).

feature interactions are useful, which is why we sample all of them in the first layer, i.e.,  $m_1 = n$ , except that the diagonal elements have the smallest values, which suggests that our designed interaction selection mechanism can classify the redundant self-interacting feature interactions, even though we keep and model all pairs of feature interactions. The selected feature interactions of order-3 and order-4 mostly do not overlap in the correctly predicted instance

(a). In instance (a), our model selects relevant feature fields (*Gender*, *Age*, *ReleaseTime*, and *WatchTime*) for *Genre* in order-3 but selects the other two feature fields (*Occupation* and *Gender*) in order-4. However, in the wrongly predicted instances (b), the feature interactions of order-3 and order-4 mostly do not overlap.

This proves that our model can indeed select meaningful feature combinations and model feature interactions of increasing orders with multiple layers in most cases, rather than selecting the redundant feature combinations of the same feature fields. We can also find some meaningful feature combinations in common cases. For example, *Gender* is usually relevant to the feature fields *Age*, *occupation*, and *WatchTime*, while *Age* is usually relevant to the feature fields *Gender*, *WatchTime*, and *Genre*. This provides some rationale for the model prediction.

## 6 Conclusion and Future Work

In this work, we disclose the relationship between FMs and GNNs, and seamlessly combine them to propose a novel model GraphFM for feature interaction learning. The proposed model leverages the strengths of FMs and GNNs and solves their respective drawbacks. At each layer of GraphFM, we select the beneficial feature interactions and treat them as edges in a graph. Then, we utilize a neighborhood/interaction aggregation operation to encode the interactions into feature representations. By design, the highest order of feature interaction increases at each layer and is determined by layer depth; thus, the feature interactions of order up to the highest can be learned. GraphFM models high-order feature interactions in an explicit manner, and can generate human readable explanations of outcomes. The experimental results show that GraphFM outperforms the state-of-the-art baselines by a large margin. In addition, we conduct extensive experiments to analyse how the highest order of feature interactions and the number of modelled feature interactions influence model performance, which can help us gain deeper insight into feature interaction modelling. In the future, we aim to investigate whether the proposed method can also benefit graph representation learning, and graph/node classification tasks.

**Acknowledgements.** We would like to thank the anonymous reviewers for their valuable comments and suggestions, allowing us to improve the quality of this paper. This work is sponsored by the National Science Foundation of China (62141608).

## References

[1] He, X., Liao, L., Zhang, H., Nie, L., Hu, X., Chua, T.-S.: Neural collaborative filtering. In: WWW (2017). International World Wide Web Conferences Steering Committee

- [2] He, X., Chua, T.-S.: Neural factorization machines for sparse predictive analytics. In: SIGIR (2017)
- [3] Zhang, S., Zheng, N., Wang, D.-L.: A novel attention-based global and local information fusion neural network for group recommendation. *Machine Intelligence Research* **19**(4), 331–346 (2022)
- [4] Rendle, S.: Factorization machines. In: ICDM (2010)
- [5] Rendle, S.: Factorization machines with libfm. ACM TIST (2012)
- [6] Zhang, W., Du, T., Wang, J.: Deep learning over multi-field categorical data: A case study on user response prediction. arXiv preprint arXiv:1601.02376 (2016)
- [7] Cheng, H.-T., Koc, L., Harmsen, J., Shaked, T., Chandra, T., Aradhye, H., Anderson, G., Corrado, G., Chai, W., Ispir, M., *et al.*: Wide & deep learning for recommender systems. In: Proceedings of the 1st Workshop on Deep Learning for Recommender Systems (2016). ACM
- [8] Guo, H., Tang, R., Ye, Y., Li, Z., He, X.: Deepfm: a factorization-machine based neural network for ctr prediction. In: IJCAI (2017)
- [9] Zhang, C., Bengio, S., Hardt, M., Recht, B., Vinyals, O.: Understanding deep learning requires rethinking generalization. arXiv preprint arXiv:1611.03530 (2016)
- [10] Su, Y., Zhang, R., Erfani, S., Xu, Z.: Detecting beneficial feature interactions for recommender systems via graph neural networks. arXiv e-prints, 2008 (2020)
- [11] Xiao, J., Ye, H., He, X., Zhang, H., Wu, F., Chua, T.-S.: Attentional factorization machines: Learning the weight of feature interactions via attention networks. arXiv preprint arXiv:1708.04617 (2017)
- [12] Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473 (2014)
- [13] Cheng, W., Shen, Y., Huang, L.: Adaptive factorization network: Learning adaptive-order feature interactions. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, pp. 3609–3616 (2020)
- [14] Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: ICLR (2017)
- [15] Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In: NIPS (2017)

- [16] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y.: Graph attention networks. In: ICLR (2018)
- [17] Li, R., Tapaswi, M., Liao, R., Jia, J., Urtasun, R., Fidler, S.: Situation recognition with graph neural networks. In: ICCV (2017)
- [18] Marcheggiani, D., Titov, I.: Encoding sentences with graph convolutional networks for semantic role labeling. arXiv preprint arXiv:1703.04826 (2017)
- [19] Yao, L., Mao, C., Luo, Y.: Graph convolutional networks for text classification. In: AAAI (2019)
- [20] Wang, X., He, X., Wang, M., Feng, F., Chua, T.-S.: Neural graph collaborative filtering. In: SIGIR (2019)
- [21] Wu, S., Tang, Y., Zhu, Y., Wang, L., Xie, X., Tan, T.: Session-based recommendation with graph neural networks. In: AAAI (2019)
- [22] Li, Z., Cui, Z., Wu, S., Zhang, X., Wang, L.: Fi-gnn: Modeling feature interactions via graph neural networks for ctr prediction. In: CIKM (2019). ACM
- [23] Li, Y., Tarlow, D., Brockschmidt, M., Zemel, R.: Gated graph sequence neural networks. arXiv preprint arXiv:1511.05493 (2015)
- [24] Xi, D., Zhuang, F., Zhu, Y., Zhao, P., Zhang, X., He, Q.: Graph factorization machines for cross-domain recommendation. arXiv preprint arXiv:2007.05911 (2020)
- [25] Zheng, Y., Wei, P., Chen, Z., Cao, Y., Lin, L.: Graph-convolved factorization machines for personalized recommendation. IEEE Transactions on Knowledge and Data Engineering (2021)
- [26] Tian, Z., Bai, T., Zhang, Z., Xu, Z., Lin, K., Wen, J.-R., Zhao, W.X.: Directed acyclic graph factorization machines for ctr prediction via knowledge distillation. In: Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining, pp. 715–723 (2023)
- [27] Dudzik, A.J., Veličković, P.: Graph neural networks are dynamic programmers. Advances in Neural Information Processing Systems **35**, 20635–20647 (2022)
- [28] Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531 (2015)
- [29] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: Advances

in Neural Information Processing Systems, pp. 5998–6008 (2017)

[30] Juan, Y., Zhuang, Y., Chin, W.-S., Lin, C.-J.: Field-aware factorization machines for ctr prediction. In: RecSys (2016). ACM

[31] Sun, Y., Pan, J., Zhang, A., Flores, A.: Fm2: Field-matrixed factorization machines for recommender systems. In: Proceedings of the Web Conference 2021, pp. 2828–2837 (2021)

[32] Qu, Y., Cai, H., Ren, K., Zhang, W., Yu, Y., Wen, Y., Wang, J.: Product-based neural networks for user response prediction. In: ICDM (2016)

[33] Wang, R., Fu, B., Fu, G., Wang, M.: Deep & cross network for ad click predictions. In: ADKDD (2017)

[34] Lian, J., Zhou, X., Zhang, F., Chen, Z., Xie, X., Sun, G.: Xdeepfm: Combining explicit and implicit feature interactions for recommender systems. In: SIGKDD (2018)

[35] Wang, R., Shivanna, R., Cheng, D., Jain, S., Lin, D., Hong, L., Chi, E.: Dcn v2: Improved deep & cross network and practical lessons for web-scale learning to rank systems. In: Proceedings of the Web Conference 2021, pp. 1785–1797 (2021)

[36] Tao, Z., Wang, X., He, X., Huang, X., Chua, T.-S.: Hoafm: A high-order attentive factorization machine for ctr prediction. Information Processing & Management, 102076 (2019)

[37] Song, W., Shi, C., Xiao, Z., Duan, Z., Xu, Y., Zhang, M., Tang, J.: Autoint: Automatic feature interaction learning via self-attentive neural networks. In: CIKM (2019)

[38] Li, Z., Cheng, W., Chen, Y., Chen, H., Wang, W.: Interpretable click-through rate prediction through hierarchical attention. In: Proceedings of the 13th International Conference on Web Search and Data Mining, pp. 313–321 (2020)

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

[40] Gori, M., Monfardini, G., Scarselli, F.: A new model for learning in graph domains. In: IJCNN (2005)

[41] Scarselli, F., Gori, M., Tsoi, A.C., Hagenbuchner, M., Monfardini, G.:

The graph neural network model. *IEEE Transactions on Neural Networks* (2009)

- [42] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014)
- [43] Bruna, J., Zaremba, W., Szlam, A., LeCun, Y.: Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203* (2013)
- [44] Beck, D., Haffari, G., Cohn, T.: Graph-to-sequence learning using gated graph neural networks. *arXiv preprint arXiv:1806.09835* (2018)
- [45] Qi, X., Liao, R., Jia, J., Fidler, S., Urtasun, R.: 3d graph neural networks for rgbd semantic segmentation. In: *ICCV* (2017)
- [46] Marino, K., Salakhutdinov, R., Gupta, A.: The more you know: Using knowledge graphs for image classification. In: *CVPR* (2017)
- [47] Chen, L., Wu, L., Hong, R., Zhang, K., Wang, M.: Revisiting graph based collaborative filtering: A linear residual graph convolutional network approach. *arXiv preprint arXiv:2001.10167* (2020)
- [48] Li, Z., Ding, X., Liu, T.: Constructing narrative event evolutionary graph for script event prediction. *arXiv preprint arXiv:1805.05081* (2018)
- [49] Cui, Z., Li, Z., Wu, S., Zhang, X.-Y., Wang, L.: Dressing as a whole: Outfit compatibility learning based on node-wise graph neural networks. In: *WWW*, pp. 307–317 (2019)
- [50] Xu, K., Hu, W., Leskovec, J., Jegelka, S.: How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826* (2018)
- [51] Kazi, A., Cosmo, L., Navab, N., Bronstein, M.: Differentiable graph module (dgm) graph convolutional networks. *arXiv preprint arXiv:2002.04999* (2020)
- [52] Kingma, D., Ba, J.: Adam: A method for stochastic optimization. *Computer Science* (2014)
- [53] Blondel, M., Fujino, A., Ueda, N., Ishihata, M.: Higher-order factorization machines. In: *NIPS* (2016)
- [54] Shan, Y., Hoens, T.R., Jiao, J., Wang, H., Yu, D., Mao, J.: Deep crossing: Web-scale modeling without manually crafted combinatorial features. In: *SIGKDD* (2016)

- [55] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., *et al.*: Tensorflow: A system for large-scale machine learning. In: OSDI (2016)
- [56] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., *et al.*: Pytorch: An imperative style, high-performance deep learning library. Advances in neural information processing systems **32** (2019)
- [57] Wu, S., Yu, F., Yu, X., Liu, Q., Wang, L., Tan, T., Shao, J., Huang, F.: Tfnet: Multi-semantic feature interaction for ctr prediction. In: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 1885–1888 (2020)
- [58] Xu, Y., Zhu, Y., Yu, F., Liu, Q., Wu, S.: Disentangled self-attentive neural networks for click-through rate prediction. In: Proceedings of the 30th ACM International Conference on Information & Knowledge Management, pp. 3553–3557 (2021)



**Shu Wu** received his B.S. degree from Hunan University, China, in 2004, his M.S. degree from Xiamen University, China, in 2007, and his Ph.D. degree from the University of Sherbrooke, Quebec, Canada. He is an Associate Professor at the NLPR, CASIA.

His research interests include data mining and pattern recognition.

E-mail: shu.wu@nlpr.ia.ac.cn

ORCID iD: 0000-0003-2164-3577



**Zekun Li** is a Ph.D. at the University of California, Santa Barbara. He obtained a master's degree from the University of Chinese Academy of Sciences, Beijing, China, in 2021, and the B.Eng degree from Shandong University, China, was obtained in 2018.

His research interests include data mining, recommender systems, and natural language processing.

E-mail: zekunli@cs.ucsb.edu



**Yunyue Su** is with the Institute of Automation, Chinese Academy of Sciences. She received her B.S degree from the Computer Network Information Center, University of Chinese Academy of Sciences, China, in 2023. Her research interests include data mining, machine learning, and recommender systems.  
E-mail: yunyue.su@ia.ac.cn



**Zeyu Cui** is with the Alibaba Group, DAMO Institute. He obtained his Ph.D. degree from the School of Artificial Intelligence, University of the Chinese Academy of Sciences, Beijing, China. He received the B.S degree from North China Electric Power University, China, in 2016. His research interests include data mining, machine learning, and recommender systems.  
E-mail: cuizeyu15@gmail.com



**Xiaoyu Zhang** received a Ph.D. degree in pattern recognition and intelligent systems from the Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 2010. He is currently a Professor with the Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China. His research interests include artificial intelligence, data mining, computer vision, etc.  
E-mail: zhxy333@gmail.com  
ORCID iD: 0000-0001-5224-8647



**Liang Wang** received a Ph.D. degree from the National Laboratory of Pattern Recognition (NLPR), Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 2004. He is currently a Professor at NLPR, CASIA. His major research interests include computer vision, pattern recognition, machine learning, and data mining.  
E-mail: liang.wang@nlpr.ia.ac.cn