Table2Graph: Transforming Tabular Data to Unified Weighted Graph

Kaixiong Zhou[1] , **Zirui Liu**[1] , **Rui Chen**[2] , **Li Li**[2] , **Soo-Hyun Choi**[3*] and **Xia Hu**[1]
[1]Department of Computer Science, Rice University
[2]Samsung Research America
[3]Samsung Electronics
{kaixiong.zhou, zl105, xia.hu}@rice.edu, {rui.chen1, li.li1, sh9.choi}@samsung.com

Abstract

Learning useful interactions between input features is crucial for tabular data modeling. Recent efforts start to explicitly model the feature interactions with graph, where each feature is treated as an individual node. However, the existing graph construction methods either heuristically formulate a fixed feature-interaction graph based on specific domain knowledge, or simply apply attention function to compute the pairwise feature similarities for each sample. While the fixed graph may be sub-optimal to downstream tasks, the sample-wise graph construction is time-consuming during model training and inference. To tackle these issues, we propose a framework named Table2Graph to transform the feature interaction modeling to learning a unified graph. Represented as a probability adjacency matrix, the unified graph learns to model the key feature interactions shared by the diverse samples in the tabular data. To well optimize the unified graph, we employ the reinforcement learning policy to capture the key feature interactions stably. A sparsity constraint is also proposed to regularize the learned graph from being overly-sparse/smooth. The experimental results in a variety of real-world applications demonstrate the effectiveness and efficiency of our Table2Graph, in terms of the prediction accuracy and feature interaction detection.

1 Introduction

Tabular data is ubiquitous in many real-world applications, such as recommender systems [Su and Khoshgoftaar, 2009] and online advertising [Cai *et al.*, 2017]. Typically, each row of the tabular dataset corresponds to one data sample, which consists of multiple individual features from different fields (i.e., columns). Taking online advertising as an example [Li *et al.*, 2019], where each sample is a log representing whether a displayed advertisement is clicked by a user or not. The log contains both user features (e.g., age and region) and the advertised item features (e.g., language and actors of a movie).

Modeling the sophisticated interactions between input features plays a key role in tabular data learning. Recalling the above example of online advertising, a user in United States is more likely to click an advertisement of English movie. In other word, the interaction modeling of feature pairs (region, language) is crucial to predict the user clicking behavior. Lots of efforts have been devoted to modeling the feature interactions, such as logistic regression [Parra *et al.*,], factorization machine [Rendle, 2010] and deep neural networks (DNNs) [Guo *et al.*, 2017]. They either explicitly learn the low-order interactions or implicitly learn the high-order interactions with DNNs, which may be sub-optimal for tabular data. The low-order algorithms have limited capability to capture the sophisticated interactions. For DNNs, the interactions entangled at hidden units lead to extremely complex optimization hyperplanes [Ke *et al.*, 2018], where the raw task objective tends to fall into local optimums. Furthermore, the implicit feature interactions cannot be directly extracted to explain how DNNs make decisions [Seo *et al.*, 2017].

Graph is a structured data modality where the pairwise node relationships are expressed explicitly by the corresponding edge weights. To learn the node interactions, graph neural networks (GNNs) are developed to pass messages along edges and update the node representations [Kipf and Welling, 2016]. Given the expressive graph structure and powerful GNNs, several initial efforts propose to construct the feature-interaction graph for tabular data, where each feature is regarded as an individual node. Some of them formulate fixed graphs heuristically based on their domain knowledge. For example, the co-purchased items are directly connected in the recommender systems [Wang *et al.*, 2020]. The fixed graph may be sub-optimal by missing/introducing the factual/noisy links, and cannot generalizes to other tabular data. The other methods apply self-attention mechanisms to quantify the feature similarities and then construct the weighted graph independently for each sample [Li *et al.*, 2019]. Such sample-wise attention computation is time-consuming during the model training and inference, which limits its practical applications for the time-sensitive scenarios. These problems motivate us to pose the following question: Is there an end-to-end framework to model the feature interactions with a unified graph? This unified graph captures the common and important feature interactions from the whole tabular dataset, and could be inferred efficiently for different samples without

*Corresponding author.

the sample-wise attention computation.

However, effectively learning a unified feature-interaction graph is a non-trivial task due to the following two challenges. First, it is hard to extract the key feature interaction patterns shared by a number of samples in the tabular data. Reusing the movie advertising as an example, we may have two clicking logs accompanied with the feature interaction patterns of [region, language] and [age, actor], respectively. Such varied patterns will lead to unstable learning of the unified graph. Second, it is unclear how to regularize the unified graph. While an overly-sparse graph structure tends to overfit the specific feature interactions, a smoothly-connected graph fails to highlight the key patterns.

To bridge the gaps, we propose Table2Graph to transform the feature interaction modeling in tabular data into a unified graph learning problem. By solving the above two challenges, we make three contributions summarized as follows.

- Motivated from neural architecture search (NAS), whose goal is to optimize the computation graph of neural networks, we propose to employ the reinforcement learning (one of NAS algorithms) to strengthen the key feature interaction connections stably. The reinforcement loss is constructed to train the unified graph based on the reward signal of tabular learning objective.
- We propose a differentiable sparsity constraint to regularize the edge connections. By jointly training with the tabular learning objective, our framework is able to learn the trade-off between sparseness and smoothness.
- We empirically demonstrate the effectiveness and efficiency of Table2Graph in both the real-world and synthetic datasets. Besides delivering the superiority in tabular data learning tasks, our Table2Graph could detect the ground-truth feature interactions. The unified graph modeling is as efficient as the fixed graph construction, without requiring extra training and testing time cost.

2 Previous Work

Our work is related to the following four research topics:

Tabular data learning. Typically, the goal of tabular data classification is to predict the label associated with every data sample, which contains a collection of individual features [Cai *et al.*, 2017; Guo *et al.*, 2017]. Although the traditional tree-based methods are commonly reported to achieve competitive performances [Ke *et al.*, 2018], they are hard to be integrated into an end-to-end framework, and cost large computational memory to store the entire dataset to get global statistics. Besides the low-order methods of logistic regression and factorization machine [Rendle, 2010], by embedding the input features, there have been many DNNs developed to model the high-order feature interactions in the hidden units implicitly [He *et al.*, 2017; Wang *et al.*, 2021].

Graph neural networks. Based on the spatial graph convolutions, GNNs recursively learn a node embedding representation by aggregating its neighbors and combining them with the node itself [Zhou *et al.*, 2020; Zhou *et al.*, 2021a]. The node interaction order is specified by the model depths of GNNs. GNN models have been applied for the real-world applications, such as recommender system [Hamilton *et al.*, 2017] and biochemical analysis [Zhou *et al.*, 2021b].

Graph structure learning. Recently, there have been several initial efforts proposed to construct the hidden graphs of other data modalities. They often rely on the heuristic knowledge of downstream applications, e.g., the co-purchased items are connected in recommender systems [Wang *et al.*, 2020]. Some of them apply the self-attention algorithms to learn the fully-connected weighted graph for each instance, such as the feature correlations of a tabular sample [Li *et al.*, 2019; Song *et al.*, 2019]. The sample-wise graph modeling is too time-consuming to be applied during model inference.

Neural architecture search. NAS finds the optimal neuron connections within the neural networks to maximize the model performance for a given task [Elsken *et al.*, 2019], which is comparable to the feature interaction modeling. Specifically, RL is one of the popular search algorithms [Zoph and Le, 2016]. By sampling the discrete neuron connections to construct the neural networks each time, RL strengthens the neuron connections if the constructed neural networks show superior performance; otherwise it will weaken them. The RL-based NAS frameworks have discovered the well-performing neural architectures for many downstream applications, such as image classification [Jaafra *et al.*, 2019] and graph analytics [Zhou *et al.*, 2019].

3 Problem Statement

Without loss of generality, we present our framework with the example of classification task, such as click-through rate prediction in online advertising [Li *et al.*, 2019]. It could be easily extended to other tasks (e.g., regression fitting) by changing the objective function (e.g., mean squared error).

We consider a tabular dataset containing n data samples (rows) and m feature fields (columns). In particular, the m columns are represented by $\boldsymbol{x} = [x_1, \cdots, x_m]$, where x_j indexes the j-th column. For example, the four columns in Figure 1 are indexed by $[x_1, \cdots, x_4]$. For the i-th sample in the table, it is associated with m feature values $\boldsymbol{x}^{(i)} = [x_1^{(i)}, \cdots, x_m^{(i)}]$ and a discrete label $y^{(i)}$. For example, $\boldsymbol{x}^{(i)}$ in Figure 1 is instantiated by $[0.1, \text{Male}, \text{Green}, 0.5]$. Given the training samples, the goal of tabular data learning is to learn a mapping function $\hat{y}^{(i)} = f(\boldsymbol{x}^{(i)})$ to predict labels in testing set. In this paper, we aim to learn a unified graph to model the feature interactions shared by all the independent samples, and apply GNN to learn function $f(\boldsymbol{x}^{(i)})$. We define the feature-interaction graph and its probability adjacency matrix in the context of tabular data as follows.

Definition 1 (Feature-interaction graph). A graph is denoted by tuple $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ and $\mathcal{E}$ are the sets of nodes and edges. In tabular data, we represent the j-th column with node j, and the feature interactions by edges.

Definition 2 (Probability adjacency matrix). Let $\boldsymbol{A} \in \mathbb{R}^{m \times m}$ denote the probability adjacency matrix of the feature-interaction graph. Each row of matrix $\boldsymbol{A}$ is normalized with sum of 1 to provide an intuitive probability explanation. Element $A_{jk} \geq 0$ is the edge weight between nodes j and k,

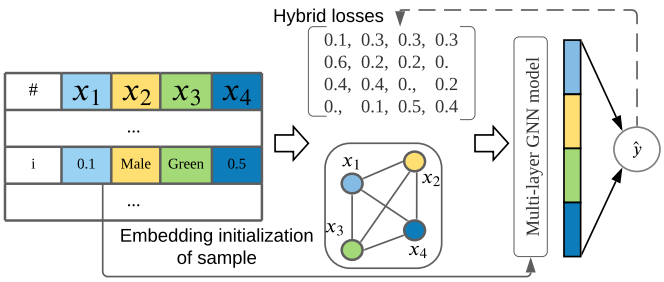

Figure 1: Table2Graph overview. The feature column interactions are represented by the probability adjacency matrix of unified graph. Given a sample, we initialize its feature embeddings and apply GNNs to learn their interactions. The concatenated feature embedding learned from GNNs is used to predict the sample, and the hybrid loss is used to update graph.

and thus represents the feature interaction strength between feature columns x_j and x_k.

4 Our Proposed Framework

Figure 1 illustrates our Table2Graph framework, which models the unified feature-interaction graph by optimizing the probability adjacency. Based on it, we employ GNNs to learn the feature interactions, and generate sample embedding for the downstream task. The details are introduced below.

4.1 Unified Graph Construction

Probability adjacency matrix computation. Given the m columns in tabular data, i.e., $\boldsymbol{x} = [x_1, \cdots, x_m]$, we first represent them with column embeddings $\boldsymbol{E} \in \mathbb{R}^{m \times d}$ to facilitate the unified graph modeling. Column x_j is denoted by the j-th row in $\boldsymbol{E}$. We compute the probability adjacency matrix $\boldsymbol{A}$ with self attention as follows [Shaw *et al.*, 2018]:

$$\boldsymbol{A} = \mathrm{Softmax}(\sigma(\boldsymbol{E}\boldsymbol{W}_l)\sigma(\boldsymbol{E}\boldsymbol{W}_r)^\top) \in \mathbb{R}^{m \times m}. \quad (1)$$

$\boldsymbol{W}_l, \boldsymbol{W}_r \in \mathbb{R}^{d \times d'}$ are trainable matrices; σ is activation function; function $\mathrm{Softmax}$ is applied in row-wise fashion to normalize the link weights of one column to the others.

Feature interaction learning. Considering sample $\boldsymbol{x}^{(i)} = [x_1^{(i)}, \cdots, x_m^{(i)}]$, we transform it into feature embeddings $\boldsymbol{X}_0^{(i)} \in \mathbb{R}^{m \times d}$, where each row is the dense representation of a feature. The series of feature embeddings in $\boldsymbol{X}_0^{(i)}$ is used to initialize nodes in the feature-interaction graph.

We then employ GNNs to learn the sophisticated feature interactions in tabular data. To be specific, GNNs recursively update a node embedding by aggregating and fusing its neighbors' information with the node itself. The graph convolution at the k-th layer is defined as:

$$\boldsymbol{X}_k^{(i)} = \boldsymbol{X}_0^{(i)} + \sigma(\boldsymbol{A}\boldsymbol{X}_{k-1}^{(i)}\boldsymbol{W}_k). \quad (2)$$

$\boldsymbol{X}_k^{(i)} \in \mathbb{R}^{m \times d}$ is the intermediate feature embeddings learned at the k-th layer, and $\boldsymbol{W}_k \in \mathbb{R}^{d \times d}$ is a trainable matrix. We exploit the initial connection of $\boldsymbol{X}_0^{(i)}$ to facilitate the gradient flow to train the initial feature embeddings well [Chen *et al.*, 2022].

Suppose the number of graph convolutional layer is K. Therefore, the feature embedding matrix $\boldsymbol{X}_K^{(i)}$ learned from GNNs aggregates the neighborhood information up to K hops away based on $\boldsymbol{A}$. Considering the given sample $\boldsymbol{x}^{(i)}$, we obtain its representation by concatenating each feature embedding from $\boldsymbol{X}_K^{(i)}$, and finally generate its prediction $\hat{y}^{(i)}$. The cross-entropy loss in binary classification task is:

$$L_{\text{task}} = y^{(i)} \log(\hat{y}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)}). \quad (3)$$

4.2 Unified Graph Training

The naive optimization strategy is to minimize loss L_{task}, and train adjacency matrix $\boldsymbol{A}$ and GNNs with gradient backpropagation. However, such training strategy has two limitations. First, it will learn a dense adjacency matrix to store all the possible feature interactions, and fail to highlight the key interaction patterns shared by the majority of samples. The dense adjacency matrix brings the incorrect prior knowledge to encode the specific feature interactions of a local sample, which leads to poor performance. Second, since the feature interactions vary among the different samples, the training of dense adjacency matrix may be unstable in the batch training.

To overcome these limitations, we propose to employ reinforcement learning (RL) to learn the key feature interactions stably. To be specific, given matrix $\boldsymbol{A}$, we sample the important links by following the learned probability distribution, and then apply RL to reinforce them if they are informative to the whole dataset. The optimization of matrix $\boldsymbol{A}$ focuses on the key feature interactions to make the training more stable. Notably, the unified feature-interaction graph optimization is comparable in spirit to NAS [Elsken *et al.*, 2019], whose goal is to optimize the connections of neural networks' computation graph. It has been shown that RL could converge efficiently [Zhou *et al.*, 2019] to the well-performing neural architectures.We introduce RL details in the context of unified feature-interaction graph learning in the following.

Important link sampling. We sample a number of feature interactions for each row, which is formally given by:

$$\mathcal{I}_i = \mathrm{RowSample}(\boldsymbol{A}[i,:], s) = \{(i, j_1) \cdots, (i, j_s)\}. \quad (4)$$

s is the sample size; $\mathrm{RowSample}$ denotes the random sampling operation based on the multinomial probability distribution; $\mathcal{I} = \bigcup_i \mathcal{I}_i$ denotes the union of sampled interaction pairs. Based on the above sampling, we attend on the key feature interactions with high link weights.

Reward shaping. Given the sampled feature interactions, we exploit REINFORCE rule [Sutton *et al.*, 1999] to shape reward and construct the reinforcement loss. At each epoch, we take the inverse of task loss as a non-differentiable reward signal R, i.e. $R = \frac{1}{L_{\text{task}}}$. The reinforcement loss is:

$$L_{\text{rl}}(\boldsymbol{A}) = -\lambda_1 * \mathbb{E}_{\mathcal{I} \sim \boldsymbol{A}}[\sum_{(i,j) \in \mathcal{I}} (R - R_{\text{avg}}) \log A_{ij}]. \quad (5)$$

λ_1 is loss hyperparameter. R_{avg} denotes the running average of rewards during the batch training. We compare reward R achieved in current batch to reward average R_{avg} to reduce the learning variance of matrix $\boldsymbol{A}$. More importantly, the

sampled feature interaction probabilities A_{ij} are strengthened only if $R > R_{\text{avg}}$; otherwise they are weakened. In other word, we learn the probability adjacency matrix $\boldsymbol{A}$ towards extracting the key feature interactions, which could improve the reward or reduce training loss L_{task}. These feature interactions are commonly preferred by the batch of samples. Following the previous efforts, to simplify the expectation computation over $\mathcal{I} \sim \boldsymbol{A}$, we only sample one $\mathcal{I}$ to approximate reinforcement loss $L_{\text{rl}}(\boldsymbol{A})$.

4.3 Joint Training With Sparsity

Besides updating the feature-interaction graph towards minimizing the task loss, it is also crucial to control the sparsity of adjacency matrix. On one hand, a smoothly-connected graph fails to highlight the key feature interactions, since each row in $\boldsymbol{A}$ is averaged to connect to the remaining features. On the other hand, an overly-sparse graph may overfit in some feature interactions, which is not adaptable to the diverse tabular samples. Therefore, we propose the sparsity loss as follows:

$$L_{\text{sp}}(\boldsymbol{A}) = -\frac{\lambda_2}{m}\mathbf{1}^T \log((\boldsymbol{A} \odot \boldsymbol{A}))\mathbf{1} + \frac{\lambda_3}{m}||\boldsymbol{A}||_F^2. \quad (6)$$

$\mathbf{1} = [1, \cdots, 1]^\top$ denotes all-ones vector of length m; $\odot$ denotes the element-wise multiplication; $||\cdot||_F$ denotes the Frobenius norm; λ_2 and λ_3 are loss hyperparameters. By minimizing the above sparsity loss, the first item is exploited to improve the squared norm of each row, i.e., $\sum_j A_{ij}^2$. Since the probability vector in each row is normalized and has the sum of one, the improvement of squared norm leads to a sparse distribution. In contrast, the second item penalizes a large squared norm to avoid the overly sparsity.

Based on the previous defined losses, we simply and jointly learn the unified graph and GNN model by minimizing the hybrid loss as follows:

$$L = L_{\text{task}} + L_{\text{rl}}(\boldsymbol{A}) + L_{\text{sp}}(\boldsymbol{A}). \quad (7)$$

5 Experiments

In the context of feature-interaction graph modeling, we categorize the tabular data into two cases: one with a small number of feature columns (e.g., most of the Kaggle competition data [Dal Pozzolo *et al.*, 2015]) and the other with a large volume of feature columns (e.g., recommender system accompanied with many items [He *et al.*, 2017]). To validate the general effectiveness of Table2Graph on both the small and large feature-interaction graph modeling from the diverse domains, we carefully design experiments to answer the following four research questions. **Q1:** How does our Table2Graph perform on small graph modeling to improve the downstream tabular data learning and detect the underlying feature interactions? Q2: How effective is Table2Graph to learn the large feature-interaction graph for table with many columns? **Q3:** How does the model hyperparameters affect the performances of Table2Graph? **Q4:** How efficient is Table2Graph comparing with the other graph construction methods?

Data. Considering the small graph modeling for the tabular data with a few feature fields, we adopt two benchmark datasets from the financial fraud detection and online advertising: Creditcard [Dal Pozzolo *et al.*, 2015] and Criteo[1]. The tabular data of Creditcard has $284,807$ transaction samples with 28 numeric anonymous features, while Criteo contains 45 million users' click records with 39 feature fields. The tabular data learning tasks in these two datasets are to predict the discrete labels of samples.

Besides the classification tasks, we detect the meaningful feature interactions upon the small graph modeling. Since there are no ground-truth labels for the feature interactions in most of the real-world datasets, we further synthesize a tabular dataset commonly used in previous efforts [Liu *et al.*, 2020]. To be specific, the synthetic dataset defines the regression task as follows:

$$y = \frac{1}{1 + x_0^2 + x_1^2 + x_2^2} + \sqrt{e^{x_3 + x_4}} + |x_5 + x_6| + x_7 x_8 x_9. \quad (8)$$

The ground-truth feature interactions in Equation (8) are $\{[x_0, x_1, x_2], [x_3, x_4], [x_5, x_6], [x_7, x_8, x_9]\}$.

For the large graph modeling, we adopt MovieLens dataset commonly evaluated in the previous collaborative filtering work [He *et al.*, 2017]. There are total $3,706$ items (columns) in the tabular data of MovieLens, where the classification task is to predict user's personalized preference scores on items and rank the interested items. In this study, our Table2Graph needs to learn the massive item-to-item interactions with 9 million edges in the unified graph. More details of Creditcard, Criteo, synthetic dataset, and MovieLens are in Appendix.

Baselines. We consider the following four baseline categories. More details are in Appendix.

- **First order.** It is represented by linear regression model.
- **Factorization machine based methods.** FM [Rendle, 2010], DeepFM [Guo *et al.*, 2017] and AFM [Xiao *et al.*, 2017] are included to learn the second-order interactions.
- **High order.** The high-order baselines contain the tree-based approaches (including random forest and decision tree) and the deep neural networks (MLP, DeepCrossing [Shan *et al.*, 2016], NFM [He *et al.*, 2017], and CIN [Lian *et al.*, 2018]).
- **Graph learning.** Model Fi-GNN [Li *et al.*, 2019] quantifies the feature similarity of each sample by learning a weighted graph with self-attention, based on which applying GNNs to learn the feature interactions. We further implement baseline Fixed-GNN, where we instead use a fully-connected and fixed graph. Each node is linked to all the remaining nodes with equal probabilities.

Implementations. A three-layer GNN model is used to learn the feature interactions based upon the generated graph. The hyperparameters of λ_1, λ_2 and λ_3 are determined based on the grid search, and their influences are empirically studied in the following experiments.

Small graph modeling. To answer research question **Q1**, we evaluate our method in Creditcard, Criteo, and the synthetic datasets to see whether it could model the small feature-interaction graphs accurately and improve the downstream

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

Types	Creditcard			Criteo			Synthesis	
	Methods	AUC	LogLoss	Methods	AUC	LogLoss	Methods	AUC
First order	LR	0.9104	0.0078	LR	0.7820	0.4695	LR	0.5077
FM based methods	FM	0.7897	0.2198	FM	0.7836	0.4700	FM	0.5119
	DeepFM	0.7374	0.0134	AFM	0.7938	0.4584	DeepFM	0.4839
High order	MLP	0.9686	0.0033	DeepCrossing	0.8009	0.4513	MLP	0.4839
	Random forest	0.9736	0.0029	NFM	0.7957	0.4562	RuleFit	0.7970
	Decision tree	0.8719	0.0064	CIN	0.8009	0.4517	AG	0.6700
Graph learning	Fixed-GNN	0.9727	0.0034	Fixed-GNN	0.8029	0.4483	Fixed-GNN	0.5190
	Fi-GNN	0.9794	0.0029	Fi-GNN	0.8062	0.4453	Fi-GNN	0.4970
	Table2Graph	**0.9810**	**0.0027**	Table2Graph	**0.8089**	**0.4429**	Table2Graph	**0.9714**

Table 1: AUC and/or LogLoss in Creditcard, Criteo, and the synthetic datasets. Due to the space limit, we only report the results of well-performing FM based models and high-order methods for each dataset.

tasks. Following the common evaluation process, we use 10-fold cross validation and judge models with metrics AUC (Area Under the ROC curve) and LogLoss for real-world datasets Creditcard and Criteo. For the feature interaction detection in the synthetic dataset, we only focus on the detection AUC. In Criteo accompanied with millions of samples, we compare with FM, AFM and the high-order methods of DeepCrossing, NFM, and CIN, since they deliver the superior performances in large dataset. In Creditcard with a small quantity of samples, MLP and tree-based methods show the outperforming results comparing with other high-order methods. In the synthetic dataset, we additionally compare with two competitive detection methods, i.e., RuleFit [Friedman *et al.*, 2008] and AG [Sorokina *et al.*, 2008]. Each experiment is run with 3 random trials and report the average results.

The test performance comparisons are listed in Table 1. Specifically, the high-order approaches are potential to achieve better AUCs (or LogLosses) comparing with the first-order and FM based methods, since they have the sufficient capabilities to learn the sophisticated feature interactions popularly existing in the realistic datasets. Due to the distinct advantage of picking informative features, the random forest shows the competitive results and ranks top in the OpenML LeaderBoard[2]. However, the tree-based methods cannot be scaled to the large tabular dataset of Criteo.

Comparing with the above traditional baselines, the graph learning based models generally achieve better performances. Instead of implicitly modeling the interactions, the graph learning based approaches explicitly store the feature interaction strengths over links in the constructed graphs. The powerful GNNs are then applied to pass messages among the correlated features and learn the feature interactions accurately to achieve the desired performances. Notably, our Table2Graph delivers the much superior results comparing with Fixed-GNN and Fi-GNN. To be specific, Fixed-GNN fully connects all the feature pairs to formulate the fixed graphs, which fail to capture the key feature interactions and may introduce noises during the neighbor aggregation in GNNs. Fi-GNN learns an independent feature-interaction graph for each sample with the attention function, which is hard to be trained due to the noises existing in the diverse samples. Fi-GNN is even incapable of summarizing the ground-truth unified graph in the synthetic dataset. Our Table2Graph instead learns a unified feature-interaction graph shared by all the samples to save computation cost. To regularize the unified graph, we propose the reinforcement loss to reduce the graph variation and weight the common feature interactions.

[2]https://www.openml.org/t/145685

Frameworks	Methods	Metrics	
		HR	NDCG
FISM	Plain	0.6730	0.3949
	Fixed-GNN	0.6748	0.3963
	Fi-GNN	0.6755	0.3963
	Table2Graph	**0.6811**	**0.4003**
NAIS	Plain	0.7020	0.4304
	Fixed-GNN	0.7059	0.4305
	Fi-GNN	0.7061	0.4313
	Table2Graph	**0.7111**	**0.4341**

Table 2: Performances of HR and NDCG in Movielens. Plain denotes the original model, i.e, FISM or NAIS.

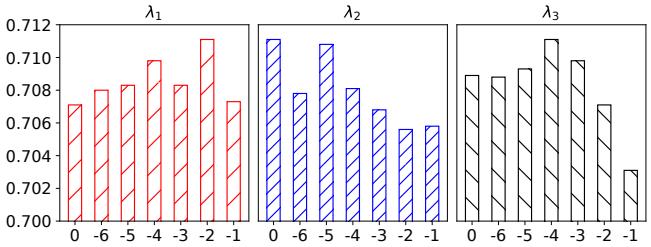

Figure 2: From left to right, the impacts of loss hyperparameters λ_1, λ_2 and λ_3 on Table2Graph trained in MovieLens, respectively. Y-axis is metric HR@10. X-axis is the log of hypervalues ([0, 0.1]).

Large graph modeling. To answer question **Q2**, we conduct the large graph modeling in Movielens. We adopt two popular item-to-item collaborative filtering methods as the underlying recommendation frameworks, namely factored item similarity model (FISM) [Kabbur *et al.*, 2013] and neural attentive item similarity model (NAIS) [He *et al.*, 2018]. The graph learning based methods are implemented over these two frameworks to learn the item-to-item interactions. The traditional feature-interaction modeling methods are removed, since they are not specifically developed to the item based collaborative filtering. Following the previous practice, we judge model performances by hit ratio (HR) and normalized discounted cumulative gain (NDCG) at the position 10. More details about the implementation are in Appendix.

The test results are listed in Table 2. It is observed that our

Methods	Creditcard		Criteo	
	train	test	train	test
Fixed-GNN	79s	2.1s	2076s	45.6s
Fi-GNN	97s	2.3s	2792s	66.7s
Table2Graph	81s	2.1s	2076s	45.2s

Table 3: Training time per epoch and testing time in Creditcard and Criteo. Time is measured in seconds.

model delivers the highest scores of HR and NDCG. Comparing with the plain FISM and NAIS, the graph learning based approaches learn the high-order item interactions to boost the recommendation performances. Comparing with Fixed-GNN and Fi-GNN, our Table2Graph additionally exploits the reinforcement and sparsity constraints to regularize the graph structure. In this way, the common item interactions are weighted to be shared by all the users, while the overly-sparse unified graph will be penalized to avoid trapping in the specific item interaction patterns.

Hyperparameter and ablation study. To understand the hyperparameter impacts and answer research question **Q3**, we conduct experiments with different values of loss hypaparameters λ_1, λ_2 and λ_3. Considering the value range $[0, 0.1]$ and the large graph learning in MovieLens, we present the hyperparameter study in Figure 2.

It is observed that the appropriate choices of these hyperparameters are crucial to construct the good item-to-item interaction graph. Without the reinforcement loss, i.e., $\lambda_1 = 0$, the obtained performance will be much lower than that of $\lambda_1 = 0.01$. That justifies the effectiveness of the proposed reinforcement loss to reduce variation of the learned graph and to model the key feature interactions stably. In the large item graph of MovieLens, a proper sparsity constraint with $\lambda_2 \to 0$ and $\lambda_3 \to 10^{-4}$ could penalize the overly-sparse graph to avoid the overfitting on specific feature interactions.

Efficiency comparison. To answer research question **Q4**, we compare the training time per epoch and the testing time for the graph learning based approaches. The comparison results are listed in Table 3, where our Table2Graph is as efficient as Fixed-GNN. Note that Fixed-GNN uses the fixed graph to represent the constant feature interactions, which does not introduce any extra time cost. Since Fi-GNN is required to compute the feature-interaction graph each time for a sample, it will be extremely time consuming in both the training and testing phases. Once our unified graph is prepared in the training phase, it could be inferred as efficiently as the fixed graph in the testing phase.

Adjacency matrix visualization. To intuitively understand how Table2Graph learns the ground-truth feature interactions, we visualize the adjacency matrix for the synthetic dataset in Figure 3. The synthetic dataset contains 10 features denoted by the corresponding rows and columns in Figure 3. Each elements (i, j) denotes the feature interaction strength. Note that the ground-truth feature interactions in Equation (8) are $\{[x_0, x_1, x_2], [x_3, x_4], [x_5, x_6], [x_7, x_8, x_9]\}$. We observe that each feature (row) has the strongest interactions with the ground-truth neighborhood features. Specifically, for feature pairs $\{3, 4\}$ and $\{5, 6\}$, their interaction strengths are significantly larger than the others in the same row.

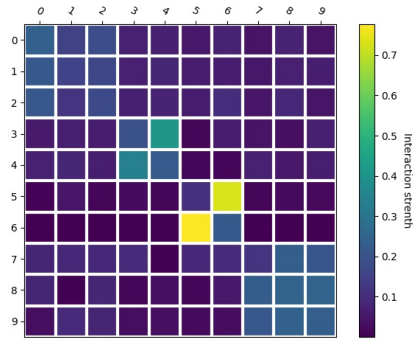

Figure 3: Adjacency matrix visualization in systhetic dataset.

6 Conclusion

In this paper, we propose the Table2Graph framework to learn the feature interactions with a unified graph, where a probability adjacency matrix is modeled to express the key feature interactions shared by diverse samples in a tabular dataset. Specifically, the reinforcement loss and sparsity constraint are proposed to weight the important interaction patterns and regularize the graph connectivity, respectively. The experimental results suggest the effectiveness and efficiency of unified graph modeling in various practical applications, where our Table2Graph consistently delivers the superior prediction performances. In particular, our model is able to accurately detect the ground-truth feature interactions and even outperforms the competitive methods in the synthetic dataset. Once the unified graph is prepared, it could be practical for the realistic applications to model the feature interactions efficiently.

References

[Cai *et al.*, 2017] Han Cai, Kan Ren, Weinan Zhang, Kleanthis Malialis, Jun Wang, Yong Yu, and Defeng Guo. Real-time bidding by reinforcement learning in display advertising. In *Proceedings of the Tenth ACM International Conference on WSDM*, pages 661–670, 2017.

[Chen *et al.*, 2022] Tianlong Chen, Kaixiong Zhou, Keyu Duan, Wenqing Zheng, Peihao Wang, Xia Hu, and Zhangyang Wang. Bag of tricks for training deeper graph neural networks: A comprehensive benchmark study. *IEEE TPAMI*, 2022.

[Dal Pozzolo *et al.*, 2015] Andrea Dal Pozzolo, Olivier Caelen, Reid A Johnson, and Gianluca Bontempi. Calibrating probability with undersampling for unbalanced classification. In *2015 IEEE Symposium Series on Computational Intelligence*, pages 159–166, 2015.

[Elsken *et al.*, 2019] Thomas Elsken, Jan Hendrik Metzen, Frank Hutter, et al. Neural architecture search: A survey. *J. Mach. Learn. Res.*, 20(55):1–21, 2019.

[Friedman *et al.*, 2008] Jerome H Friedman, Bogdan E Popescu, et al. Predictive learning via rule ensembles. *Annals of Applied Statistics*, 2(3):916–954, 2008.

[Guo *et al.*, 2017] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. Deepfm: a

factorization-machine based neural network for ctr prediction. *IJCAI*, 2017.

[Hamilton *et al.*, 2017] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NeuIPS*, pages 1024–1034, 2017.

[He *et al.*, 2017] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th Web Conference*, pages 173–182, 2017.

[He *et al.*, 2018] Xiangnan He, Zhankui He, Jingkuan Song, Zhenguang Liu, Yu-Gang Jiang, and Tat-Seng Chua. Nais: Neural attentive item similarity model for recommendation. *IEEE TKDE*, 30(12), 2018.

[Jaafra *et al.*, 2019] Yesmina Jaafra, Jean Luc Laurent, Aline Deruyver, and Mohamed Saber Naceur. Reinforcement learning for neural architecture search: A review. *Image and Vision Computing*, 89:57–66, 2019.

[Kabbur *et al.*, 2013] Santosh Kabbur, Xia Ning, and George Karypis. Fism: factored item similarity models for top-n recommender systems. In *Proceedings of the 19th ACM SIGKDD*, pages 659–667, 2013.

[Ke *et al.*, 2018] Guolin Ke, Jia Zhang, Zhenhui Xu, Jiang Bian, and Tie-Yan Liu. Tabnn: A universal neural network solution for tabular data. *https://openreview.net/forum?id=r1eJssCqY7*, 2018.

[Kipf and Welling, 2016] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[Li *et al.*, 2019] Zekun Li, Zeyu Cui, Shu Wu, Xiaoyu Zhang, and Liang Wang. Fi-gnn: Modeling feature interactions via graph neural networks for ctr prediction. In *Proceedings of the 28th CIKM*, pages 539–548, 2019.

[Lian *et al.*, 2018] Jianxun Lian, Xiaohuan Zhou, Fuzheng Zhang, Zhongxia Chen, Xing Xie, and Guangzhong Sun. xdeepfm: Combining explicit and implicit feature interactions for recommender systems. In *Proceedings of the 24th ACM SIGKDD*, pages 1754–1763, 2018.

[Liu *et al.*, 2020] Zirui Liu, Qingquan Song, Kaixiong Zhou, Ting-Hsiang Wang, Ying Shan, and Xia Hu. Detecting interactions from neural networks via topological analysis. *NeurIPS*, 33, 2020.

[Parra *et al.*,] Denis Parra, Alexandros Karatzoglou, Xavier Amatriain, and Idil Yavuz. Implicit feedback recommendation via implicit-to-explicit ordinal logistic regression mapping. *Proceedings of the CARS-2011*.

[Rendle, 2010] Steffen Rendle. Factorization machines. In *2010 IEEE ICDM*, pages 995–1000. IEEE, 2010.

[Seo *et al.*, 2017] Sungyong Seo, Jing Huang, Hao Yang, and Yan Liu. Interpretable convolutional neural networks with dual local and global attention for review rating prediction. In *Proceedings of the eleventh ACM conference on recommender systems*, 2017.

[Shan *et al.*, 2016] Ying Shan, T Ryan Hoens, Jian Jiao, Haijing Wang, Dong Yu, and JC Mao. Deep crossing: Web-scale modeling without manually crafted combinatorial features. In *Proceedings of the 22nd ACM SIGKDD*, pages 255–262, 2016.

[Shaw *et al.*, 2018] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. *arXiv preprint arXiv:1803.02155*, 2018.

[Song *et al.*, 2019] Weiping Song, Chence Shi, Zhiping Xiao, Zhijian Duan, Yewen Xu, Ming Zhang, and Jian Tang. Autoint: Automatic feature interaction learning via self-attentive neural networks. In *Proceedings of the 28th ACM CIKM*, pages 1161–1170, 2019.

[Sorokina *et al.*, 2008] Daria Sorokina, Rich Caruana, Mirek Riedewald, and Daniel Fink. Detecting statistical interactions with additive groves of trees. In *Proceedings of the 25th ICML*, pages 1000–1007, 2008.

[Su and Khoshgoftaar, 2009] Xiaoyuan Su and Taghi M Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in artificial intelligence*, 2009, 2009.

[Sutton *et al.*, 1999] Richard S Sutton, David A McAllester, Satinder P Singh, Yishay Mansour, et al. Policy gradient methods for reinforcement learning with function approximation. In *NIPS*. Citeseer, 1999.

[Wang *et al.*, 2020] Shoujin Wang, Liang Hu, Yan Wang, Xiangnan He, Quan Z Sheng, Mehmet Orgun, Longbing Cao, Nan Wang, Francesco Ricci, and Philip S Yu. Graph learning approaches to recommender systems: A review. *arXiv*, 2020.

[Wang *et al.*, 2021] Ruoxi Wang, Rakesh Shivanna, Derek Cheng, Sagar Jain, Dong Lin, Lichan Hong, and Ed Chi. Dcn v2: Improved deep & cross network and practical lessons for web-scale learning to rank systems. In *Proceedings of the Web Conference*, pages 1785–1797, 2021.

[Xiao *et al.*, 2017] Jun Xiao, Hao Ye, Xiangnan He, Hanwang Zhang, Fei Wu, and Tat-Seng Chua. Attentional factorization machines: Learning the weight of feature interactions via attention networks. *arXiv preprint arXiv:1708.04617*, 2017.

[Zhou *et al.*, 2019] Kaixiong Zhou, Qingquan Song, Xiao Huang, and Xia Hu. Auto-gnn: Neural architecture search of graph neural networks. *arXiv:1909.03184*, 2019.

[Zhou *et al.*, 2020] Kaixiong Zhou, Xiao Huang, Yuening Li, Daochen Zha, Rui Chen, and Xia Hu. Towards deeper graph neural networks with differentiable group normalization. *NeurIPS*, 33:4917–4928, 2020.

[Zhou *et al.*, 2021a] Kaixiong Zhou, Xiao Huang, Daochen Zha, Rui Chen, Li Li, Soo-Hyun Choi, and Xia Hu. Dirichlet energy constrained learning for deep graph neural networks. *NeurIPS*, 34, 2021.

[Zhou *et al.*, 2021b] Kaixiong Zhou, Qingquan Song, Xiao Huang, Daochen Zha, Na Zou, and Xia Hu. Multi-channel graph neural networks. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pages 1352–1358, 2021.

[Zoph and Le, 2016] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.