Detecting Beneficial Feature Interactions for Recommender Systems via Graph Neural Networks

Yixin Su [1], Rui Zhang [1], Sarah Erfani [1], Zhenghua Xu [2]
[1]University of Melbourne, [2]Hebei University of Technology
yixins1@student.unimelb.edu.au, rui.zhang@unimelb.edu.au, sarah.erfani@unimelb.edu.au, zhenghua.xu@hebut.edu.cn

Abstract

Feature interactions are essential for achieving high accuracy in recommender systems. Many studies take into account the interaction between every pair of features. However, this is suboptimal because some feature interactions may not be that relevant to the recommendation result and taking them into account may introduce noise and decrease recommendation accuracy. To make the best out of feature interactions, we propose a graph neural network approach to effectively model them, together with a novel technique to automatically detect those feature interactions that are beneficial in terms of recommendation accuracy. The automatic feature interaction detection is achieved via edge prediction with an L_0 activation regularization. Our proposed model is proved to be effective through the information bottleneck principle and statistical interaction theory. Experimental results show that our model (i) outperforms existing baselines in terms of accuracy, and (ii) automatically identifies beneficial feature interactions.

1 Introduction

Recommender systems play a central role in addressing information overload issues in many Web applications, such as e-commerce, social media platforms and lifestyle apps. The core of recommender systems is to predict how likely a user will interact with an item (e.g., purchase, click). An important technique is to discover the effects of features (e,g., contexts, user/item attributes) on the target prediction outcomes for fine-grained analysis (Shi, Larson, and Hanjalic 2014). Some features are correlated to each other, and the joint effects of these correlated features (i.e., *feature interactions*) are crucial for recommender systems to get high accuracy (Blondel et al. 2016; He and Chua 2017). For example, it is reasonable to recommend a user to use *Uber* on a *rainy* day at off-work hours (e.g., during *5-6pm*). In this situation, considering the feature interaction $<$ *5-6pm*, *rainy* $>$ is more effective than considering the two features separately. Therefore, in recent years, many research efforts have been put in modeling the feature interactions, such as NFM (He and Chua 2017), xDeepFM (Lian et al. 2018) and AutoInt (Song et al. 2019). These models take into account the interaction between every pair of features. However, in practice, not all feature interactions are relevant to the recommendation result (Langley et al. 1994; Siegmund et al. 2012). Modeling the feature interactions that provide little useful information may introduce noise and cause overfitting, and hence decrease the prediction accuracy (Zhang et al. 2017; Louizos, Welling, and Kingma 2018). For example, a user may use *Gmail* on a workday no matter what weather it is. However, if the interaction of workday and specific weather is taking into account in the model and due to the bias in the training set (e.g., the weather of the days when the Gmail usage data are collected happens to be cloudy), the interaction $<$ *Monday*, *Cloudy* $>$ might be picked up by the model and make less accurate recommendation on Gmail usage.

In this paper, we first define the beneficial "pairwise feature interactions" (or simply "feature interactions" since we focus on pairwise feature interactions in this paper, and we leave high-order feature interactions in future work) by the performance gain, i.e., a set of pairwise feature interactions is more beneficial than another set if modeling it can produce higher prediction accuracy. Then, to overcome this problem, we propose a novel graph neural network (GNN)-based recommendation model, L_0-SIGN, that detects beneficial feature interactions, and utilizes only the beneficial ones for recommendation, where each data sample is treated as a graph, features as nodes and feature interactions as edges. Specifically, our model consists of two components. One component is an L_0 edge prediction model, which detects beneficial feature interactions by predicting the existence of edges between nodes. To ensure the success of the detection, an L_0 activation regularization is proposed to encourage unbeneficial edges (i.e. feature interactions) to have the value of 0, which means that edge does not exist. Another component is a graph classification model, called **S**tatistical **I**nteraction **G**raph neural **N**etwork (SIGN). SIGN takes nodes (i.e., features) and detected edges (i.e., beneficial feature interactions) as the input graph, and outputs recommendation predictions by effectively modeling and aggregating the node pairs that are linked by an edge. The accurate interaction modeling in SIGN provides useful feedback (founded by statistical interaction) to the L_0 edge prediction model for better detection from the errors between recommendation predictions and target outcomes.

Theoretical analyses are further conducted to verify the effectiveness of our model. First, the beneficial feature interactions are guaranteed to be detected in L_0-SIGN. This is proved by showing that the empirical risk minimization procedure of L_0-SIGN is a variational approximation of the Information Bottleneck (IB) principle, which is a golden criterion to find the most relevant information correlating to target outcomes from inputs (Tishby, Pereira, and Bialek 2000). Specifically, only the most beneficial feature interactions will be retained in L_0-SIGN. It is because, in the training stage, our model simultaneously minimizes the number of detected feature interactions by the L_0 activation regularization, and maximizes the recommendation accuracy with the detected feature interactions. Second, we further show that the modeling of the detected feature interactions in SIGN is very effective. By accurately leveraging the relational reasoning ability of GNN, *iff* a feature interaction is detected to be beneficial in the L_0 edge prediction component, it will be modeled in SIGN as a statistical interaction (an interaction is called statistical interaction if the joint effects of variables are modeled correctly).

We summarize our contributions as follows:

- We focus on the shortcoming of the existing feature-interaction-based recommender systems, and propose a GNN-based model, L_0-SIGN, to detect and utilize the beneficial feature interactions to achieve more accurate recommendations.
- We theoretically prove the effectiveness of L_0-SIGN through the information bottleneck principle and statistical interaction theory.
- We have conducted extensive experimental studies. The results show that (i) L_0-SIGN outperforms existing baselines in terms of accuracy; (ii) the superior performance of L_0-SIGN comes from the correct detecting and modeling of the beneficial feature interactions.

2 Related Work

Feature Interaction based Recommender Systems Feature interaction modeling has attracted attention in recommender systems nowadays. Factorization machine (FM) (Rendle 2010) is one of the most popular algorithms in modeling feature interactions. FM represents features as embeddings and models every feature interactions by an inner product function. Due to the progress of deep learning, modern FM-based models focus on modeling feature interactions using deep learning methods and gain state-of-the-art performances (Xiao et al. 2017; He and Chua 2017; Guo et al. 2017). However, in these models, all feature interactions are considered, while our model, L_0-SIGN, detects and models only beneficial feature interactions.

Graph neural network (GNN) GNN is a framework that can facilitate learning about entities and their relations (Battaglia et al. 2018). Much existing work leverage GNN to perform relational reasoning in various domains. For example, Duvenaud et al. (2015) and Gilmer et al. (2017) leverage GNNs to predict molecules' property by learning their features from molecular graphs. Chang et al. (2016) use GNN to learn object relations in dynamic physical systems. Besides, some relational reasoning models in computer vision such as (Santoro et al. 2017; Wang et al. 2018) have been shown to be variations of GNN (Battaglia et al. 2018). Our model innovatively connects the beneficial feature interactions in recommender systems to the existence of edges in graphs and leverages the relational reasoning ability of GNN to model beneficial feature interactions.

L_0 regularization L_0 regularization sparsifies models by penalizing non-zero parameters. Due to the problem of non-differentiable, it does not attract attention previously in deep learning domains until Louizos, Welling, and Kingma (2018) solve this problem by proposing a hard concrete distribution in L_0 regularization. Then, L_0 regularization has been commonly utilized to compress neural networks (Tsang et al. 2018; Shi, Glocker, and Castro 2019; Yang et al. 2017). We explore to utilize L_0 regularization to limit the number of detected edges in feature graphs for beneficial feature interaction detection.

3 Problem Formulation and Definitions

Consider a dataset with input-output pairs: $D = \{(X_n, y_n)\}_{1 \leq n \leq N}$, where $y_n \in \mathbb{R}/\mathbb{Z}$, $X_n = \{c_k : x_k\}_{k \in J_n}$ is a set of categorical features (c) with their values (x), $J_n \subseteq J$ and J is an index set of all features in D. For example, in app recommendation, X_n may consist of a user ID, an app ID and a set of context features (e.g., *Cloudy*, *Monday*) with values to be 1 (i.e., recorded in this data sample), and y_n is a binary value to indicate whether the user will use this app. Our goal is to design a predictive model $F(X_n)$ that detects beneficial feature interactions and utilizes only the detected feature interactions to predict the true output y_n.

Beneficial Pairwise Feature Interactions Inspired by the definition of relevant feature by usefulness (Langley et al. 1994; Blum and Langley 1997), we formally define the beneficial feature interactions in Definition 1.

Definition 1. ***(Beneficial Pairwise Feature Interactions)*** *Given a data sample $X = \{x_i\}_{1 \leq i \leq k}$ of k features whose corresponding full pairwise feature interaction set is $A = \{(x_i, x_j)\}_{1 \leq i,j \leq k}$, a set of pairwise feature interactions $I_1 \subseteq A$ is more beneficial than another set of pairwise feature interactions $I_2 \subseteq A$ to a model with X as input if the accuracy of the predictions that the model produces using I_1 is higher than the accuracy achieved using I_2.*

The above definition formalizes our detection goal: find and retain only a part of feature interactions that can produce the highest prediction accuracy by our model.

Statistical Interaction Statistical interaction, or non-additive interaction, ensures a joint influence of several variables on an output variable is not additive (Tsang et al. 2018). Sorokina et al. (2008) formally define the pairwise statistical interaction:

Definition 2. ***(Pairwise Statistical Interaction)*** *Function $F(X)$, where $X = \{x_i\}_{1 \leq i \leq k}$ has k variables, shows* ***no*** *pairwise statistical interaction between variables x_i and x_j if $F(X)$ can be expressed as the sum of two functions $f_{\backslash i}$ and $f_{\backslash j}$, where $f_{\backslash i}$ does not depend on x_i and $f_{\backslash j}$ does not* *depend on x_j:*

$$F(X) = f_{\setminus i}(x_1, ..., x_{i-1}, x_{i+1}, ..., x_k) + f_{\setminus j}(x_1, ..., x_{j-1}, x_{j+1}, ..., x_k). \quad (1)$$

More generally, if using $\boldsymbol{v}_i \in \mathbb{R}^d$ to describe the i-th variable with d factors (Rendle 2010), e.g., variable embedding, each variable can be described in a vector form $\boldsymbol{u}_i = x_i \boldsymbol{v}_i$. Then, we define the pairwise statistical interaction in variable factor form by changing the Equation 1 into:

$$F(X) = f_{\setminus i}(\boldsymbol{u}_1, ..., \boldsymbol{u}_{i-1}, \boldsymbol{u}_{i+1}, ..., \boldsymbol{u}_k) + f_{\setminus j}(\boldsymbol{u}_1, ..., \boldsymbol{u}_{j-1}, \boldsymbol{u}_{j+1}, ..., \boldsymbol{u}_k). \quad (2)$$

The definition indicates that the interaction information of variables (features) x_i and x_j will not be captured by $F(X)$ if it can be expressed as the above equations. Therefore, to correctly capture the interaction information, our model should not be expressed as the above equations. In this paper, we theoretically prove that the interaction modeling in our model strictly follow the definition of pairwise statistical interaction, which ensures our model to correctly capture interaction information.

4 Our Model

In this section, we formally describe L_0-SIGN's overview structure and the two components in detail. Then we illustrate the empirical risk minimization function for training. Finally, we give theoretical analyses.

L_0-SIGN

Model Overview Each input of L_0-SIGN is represented as a graph (without edge information), where its features are nodes and their interactions are edges. More specifically, a data sample n is a graph $G_n(X_n, E_n)$, and $E_n = \{(e_n)_{ij}\}_{i,j \in X_n}$ is a set of edge/interaction values [1], where $(e_n)_{ij} \in \{1, 0\}$, 1 indicates that there is an edge (beneficial feature interaction) between nodes i and j, and 0 otherwise. Since no edge information are required, $E_n = \emptyset$.

While predicting, the L_0 edge prediction component, $F_{ep}(X_n; \boldsymbol{\omega})$, analyzes the existence of edges on each pair of nodes, where $\boldsymbol{\omega}$ are parameters of F_{ep}, and outputs the predicted edge set $E_n^{'}$. Then, the graph classification component, SIGN, performs predictions based on $G(X_n, E_n^{'})$. Specifically, SIGN firstly conducts interaction modeling on each pair of initial nodes that are linked by an edge. Then, each node representation is updated by aggregating all of the corresponding modeling results. Finally, all updated node representations are aggregated to get the final prediction. The general form of SIGN prediction function is $y_n^{'} = f_S(G_n(X_n, E_n^{'}); \boldsymbol{\theta})$, where $\boldsymbol{\theta}$ is the parameters of SIGN and the predicted outcome $y_n^{'}$ is the graph classification result. Therefore, the L_0-SIGN prediction function f_{LS} is:

$$f_{LS}(G_n(X_n, \emptyset); \boldsymbol{\theta}, \boldsymbol{\omega}) = f_S(G_n(X_n, F_{ep}(X_n; \boldsymbol{\omega})); \boldsymbol{\theta}). \quad (3)$$

[1] In this paper, nodes and features are used interchangeably, and the same as edges and feature interactions.

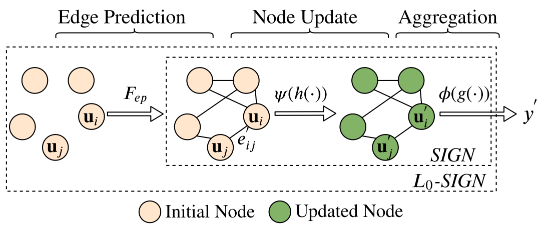


Figure 1: An overview of L_0-SIGN.

Figure 1 shows the structure of L_0-SIGN[2]. Next, we will show the two components in detail. When describing the two components, we focus on one input-output pair, so we omit the index "n" for simplicity.

L_0 Edge Prediction Model A matrix factorization (MF) based model is used for edge prediction. MF is simple and efficient in modeling relations between node pairs by factorizing the adjacency matrix of a graph into node dense embeddings (Menon and Elkan 2011). In L_0-SIGN, since we do not have the ground truth adjacency matrix, the gradients for optimizing this component come from the errors between the outputs of SIGN and the target outcomes.

More specifically, the edge value, $e_{ij}^{'} \in E^{'}$, is predicted by a MF-based edge prediction function $f_{ep}(\boldsymbol{v}_i^e, \boldsymbol{v}_j^e) : \mathbb{R}^{2 \times b} \rightarrow \mathbb{Z}_2$, which takes a pair of node embeddings for edge prediction with dimension b as input, and output a binary value to indicate whether the two nodes are connected by an edge. $\boldsymbol{v}_i^e = \boldsymbol{o}_i \boldsymbol{W}^e$ is the embedding of node i for edge prediction, where $\boldsymbol{W}^e \in \mathbb{R}^{|X| \times b}$ are parameters and $\boldsymbol{o}_i$ is the one-hot embedding of node i with size $|X|$. To ensure that the detection results are identical to the same pair of nodes, f_{ep} should be invariant to the order of its input, i.e., $f_{ep}(\boldsymbol{v}_i^e, \boldsymbol{v}_j^e) = f_{ep}(\boldsymbol{v}_j^e, \boldsymbol{v}_i^e)$. Note that $e_{ii}^{'} = 1$ can be regarded as the feature i being beneficial.

While training, an L_0 activation regularization is performed on $E^{'}$ to minimize the number of detected edges, which will be described later in the section about the empirical risk minimization function of L_0-SIGN.

SIGN In SIGN, each node i is first represented as an initial node embedding $\boldsymbol{v}_i$ of d dimensions. Then, pairwise analysis is performed on each node pair (i, j) that $e_{ij}^{'} = 1$, by a non-additive function $h(\boldsymbol{u}_i, \boldsymbol{u}_j) : \mathbb{R}^{2 \times d} \rightarrow \mathbb{R}^d$ (e.g., a multilayer neural network), where $\boldsymbol{u}_i = x_i \boldsymbol{v}_i$. The result is denoted as $\boldsymbol{z}_{ij}$. Similar to f_{ep}, h should also be invariant to the order of its input. The above procedure can be reformulated as $\boldsymbol{s}_{ij} = e_{ij}^{'} \boldsymbol{z}_{ij}$, where $\boldsymbol{s}_{ij} \in \mathbb{R}^d$ is called the statistical interaction analysis result of (i, j).

Next, each node is updated by aggregating all of the analysis results between the node and its neighbors using a linear aggregation function ψ: $\boldsymbol{v}_i^{'} = \psi(\varsigma_i)$, where $\boldsymbol{v}_i^{'} \in \mathbb{R}^d$

[2] Section *D* of Appendix lists the pseudocodes of our model and the training procedures.

is the updated embedding of node i, ς_i is a set of statistical interaction analysis results between node i and its neighbors. Note that ψ should be invariant to input permutations, and be able to take inputs with variant number of elements (e.g., element-wise summation/mean). While training, node embedding will be updated iteratively by replacing initial embedding with last updated embedding: $(\boldsymbol{v}_i)_t \leftarrow (\boldsymbol{v}_i^{'})_{t-1}$, where t is the iteration index while training.

Finally, each updated node embedding will be transformed into a scalar value by a linear function $g : \mathbb{R}^d \to \mathbb{R}$, and all scalar values are linearly aggregated as the output of SIGN. That is: $y^{'} = \phi(\nu)$, where $\nu = \{g(\boldsymbol{u}_i^{'}) \mid i \in X\}$, $\boldsymbol{u}_i^{'} = x_i \boldsymbol{v}_i^{'}$ and $\phi : \mathbb{R}^{|\nu| \times 1} \to \mathbb{R}$ is an aggregation function having similar properties to ψ. Therefore, the prediction function of SIGN is:

$$f_S(G; \boldsymbol{\theta}) = \phi(\{g(\psi(\{e_{ij}^{'} h(\boldsymbol{u}_i, \boldsymbol{u}_j)\}_{j \in X}))\}_{i \in X}). \tag{4}$$

In summary, we formulate the L_0-SIGN prediction function of Equation 3 with the two components in detail:

$$f_{LS}(G; \boldsymbol{\omega}, \boldsymbol{\theta}) = \phi(\{g(\psi(\{f_{ep}(\boldsymbol{v}_i^e, \boldsymbol{v}_j^e) h(\boldsymbol{u}_i, \boldsymbol{u}_j)\}_{j \in X}))\}_{i \in X}). \tag{5}$$

Note that $F_{ep}(X, \boldsymbol{\omega})$ in Equation 3 is the set of all f_{ep} procedures in a graph G.

Empirical Risk Minimization Function

The empirical risk minimization function of L_0-SIGN minimizes a loss function, a reparameterized L_0 activation regularization[3] on the predicted edge values in $E_n^{'}$, and an L_2 activation regularization on the interaction modeling results $\boldsymbol{z}_n$. Instead of regularizing parameters, activation regularization regularizes the output of models (Merity, McCann, and Socher 2017). We leverage activation regularization to link our model with the IB principle to ensures the success of the interaction detection, which will be discussed in the theoretical analyses. Formally, the function is:

$$\begin{aligned} \mathcal{R}(\boldsymbol{\theta}, \boldsymbol{\omega}) = & \frac{1}{N} \sum_{n=1}^{N} (\mathcal{L}(F_{LS}(G_n; \boldsymbol{\omega}, \boldsymbol{\theta}), y_n) \\ & + \lambda_1 \sum_{i,j \in X_n} (\pi_n)_{ij} + \lambda_2 \left\| \boldsymbol{z}_n \right\|_2), \\ \boldsymbol{\theta}^*, \boldsymbol{\omega}^* = & \arg\min_{\boldsymbol{\theta}, \boldsymbol{\omega}} \mathcal{R}(\boldsymbol{\theta}, \boldsymbol{\omega}), \end{aligned} \tag{6}$$

where $(\pi_n)_{ij}$ is the probability of $(e_n^{'})_{ij}$ being 1 (i.e., $(e_n^{'})_{ij} = Bern((\pi_n)_{ij})$), $G_n = G_n(X_n, \emptyset)$, λ_1 and λ_2 are weight factors for the regularizations, $\mathcal{L}(\cdot)$ corresponds to a loss function and $\boldsymbol{\theta}^*$, $\boldsymbol{\omega}^*$ are final parameters.

A practical difficulty of performing L_0 activation regularization is that it is non-differentiable. Inspired by (Louizos, Welling, and Kingma 2018), we smooth the L_0 activation regularization by approximating the Bernoulli distribution with a hard concrete distribution. Then, $e_{ij}^{'}$ follows a hard concrete distribution, which is differentiable [4].

[3]Section E of Appendix gives detailed description about L_0 regularization and its reparameterization trick.

[4]Section F of Appendix gives details about the approximation.

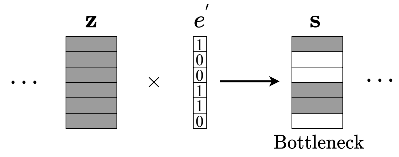


Figure 2: The interaction analysis results $\boldsymbol{s}$ from L_0-SIGN are like the relevant part (bottleneck) in the IB principle.

Theoretical Analyses

We first show how L_0-SIGN satisfies the IB principle to guarantee the success of beneficial feature interaction detection. We also illustrate the relation of the statistical interaction analysis results with the spike-and-slab distribution, which is the golden standard in sparsity. Then, we demonstrate how SIGN and L_0-SIGN strictly follow the statistical interaction theory for effective interaction modeling.

Satisfaction of Information Bottleneck (IB) Principle IB principle (Tishby, Pereira, and Bialek 2000) aims to extract the most relevant information that input random variables $\boldsymbol{X}$ contains about output variables $\boldsymbol{Y}$ by considering a trade-off between the accuracy and complexity of the process. The relevant part of $\boldsymbol{X}$ over $\boldsymbol{Y}$ denotes $\boldsymbol{S}$. The IB principle can be mathematically represented as:

$$\min \quad (I(\boldsymbol{X}; \boldsymbol{S}) - \beta I(\boldsymbol{S}; \boldsymbol{Y})), \tag{7}$$

where $I(\cdot)$ denotes mutual information between two variables and $\beta > 0$ is a weight factor.

The empirical risk minimization function of L_0-SIGN in Equation 6 can be approximately derived from Equation 7 (Section A of Appendix gives a detailed derivation):

$$\min \mathcal{R}(\boldsymbol{\theta}, \boldsymbol{\omega}) \approx \min(I(\boldsymbol{X}; \boldsymbol{S}) - \beta I(\boldsymbol{S}; \boldsymbol{Y})). \tag{8}$$

Intuitively, the L_0 regularization in Equation 6 minimizes a Kullback–Leibler divergence between every $e_{ij}^{'}$ and a Bernoulli distribution $Bern(0)$, and the L_2 regularization minimizes a KullbackLeibler divergence between every $\boldsymbol{z}_{ij}$ and a multivariate standard distribution $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. As illustrated in Figure 2, the statistical interaction analysis results, $\boldsymbol{s}$, can be approximated as the relevant part $\boldsymbol{S}$ in Equation 7.

Therefore, through training L_0-SIGN, $\boldsymbol{s}$ is the most compact representation of the interactions. This provides a theoretical guarantee that the predicted edges contain the most beneficial interaction information to the target outcome.

Relation to spike-and-slab distribution The spike-and-slab distribution (Mitchell and Beauchamp 1988) is the golden standard in sparsity. It is defined as a mixture of a delta spike at zero and a continuous distribution over the real line (e.g., a standard normal):

$$\begin{aligned} p(a) = Bern(\pi), \qquad & p(\theta \mid a = 0) = \delta(\theta), \\ & p(\theta \mid a = 1) = \mathcal{N}(\theta \mid 0, 1). \end{aligned} \tag{9}$$

We can regard the spike-and-slab distribution as the product of a continuous distribution and a Bernoulli distribution. In L_0-SIGN, the predicted edge value vector $\boldsymbol{e}^{'}$ (the vector form of $E^{'}$) is a multivariate Bernoulli distribution and can be regarded as $p(a)$ in Equation 9. The pairwise modeling result $\boldsymbol{z}$ is a multivariate normal distribution and can be regarded as $p(\theta)$ in Equation 9. Therefore, L_0-SIGN's statistical interaction analysis results, $\boldsymbol{s}$, is a multivariate spike-and-slab distribution that performs edge sparsification by discarding unbeneficial feature interactions. The retained edges in the spike-and-slab distribution are sufficient for L_0-SIGN to provide accurate predictions.

Statistical Interaction in SIGN The feature interaction modeling in SIGN strictly follows the definition of statistical interaction, which is formally described in Theorem 1 (Section *B* of Appendix gives the proof):

Theorem 1. *(Statistical Interaction in SIGN) Consider a graph $G(X, E)$, where X is the node set and $E = \{e_{ij}\}_{i,j \in X}$ is the edge set that $e_{ij} \in \{0, 1\}, e_{ij} = e_{ji}$. Let $G(X, E)$ be the input of SIGN function $f_S(G)$ in Equation 4, then the function flags pairwise statistical interaction between node i and node j if and only if they are linked by an edge in $G(X, E)$, i.e., $e_{ij} = 1$.*

Theorem 1 guarantees that SIGN will capture the interaction information from node pairs *iff* they are linked by an edge. This ensures SIGN to accurately leverage the detected beneficial feature interactions for both inferring the target outcome and meanwhile providing useful feedback to the L_0 edge prediction component for better detection.

Statistical Interaction in L_0-SIGN L_0-SIGN provides the same feature interaction modeling ability as SIGN, since we can simply extend Theorem 1 to Corollary 1.1 (Section *C* of Appendix gives the proof):

Corollary 1.1. *(Statistical Interaction in L_0-SIGN) Consider a graph G that the edge set is unknown. Let G be the input of L_0-SIGN function $F_{LS}(G)$ in Equation 5, the function flags pairwise statistical interaction between node i and node j if and only if they are predicted to be linked by an edge in G by L_0-SIGN, i.e., $e'_{ij} = 1$.*

5 Experiments

This section focuses on answering three questions: (i) how L_0-SIGN performs compared to baselines and whether SIGN helps detect more beneficial interactions for better performance? (ii) How is the detection ability of L_0-SIGN? (iii) Can the statistical interaction analysis results provide potential explanations for the recommendations predictions?

Experimental Protocol

Datasets We study two real-word datasets for recommender systems to evaluate our model.

Frappe (Baltrunas et al. 2015). It is a context-aware recommendation dataset. It records app usage logs from different users with eight types of contexts (e,g, weather, daytime). We treat each log as a graph (without edges), and nodes are either user ID, app ID, or the contexts.

MovieLens-tag (He and Chua 2017). It focuses on the movie tag recommendation (e.g., "sci-fi", "must-see"). Each data instance is regarded as a graph, with nodes as user ID, movie ID, and a tag that the user gives to the movie.

Table 1: Dataset statistics. All datasets are denoted in graph form. Each node represents one feature. Twitter and DBLP datasets are used for answering question (ii).

Dataset	#Features	#Graphs	#Nodes/Graph
Frappe	5,382	288,609	10
MovieLens	90,445	2,006,859	3
Twitter	1,323	144,033	4.03
DBLP	41,324	19,456	10.48

To evaluate the question (ii), we further study two datasets for graph classification, which will be discussed later. The statistics of the datasets are summarized in Table 1.

Baselines We compare our model with recommender system baselines that model all feature interactions:

FM (Koren 2008): It is one of the most popular recommendation algorithms that models every feature interactions by dot product. **AFM** (Xiao et al. 2017): Comparing to FM, it additionally calculates an attention value for each feature interaction. **NFM** (He and Chua 2017): It replaces the dot product procedure of FM by a multilayer neural network (MLP). We set the MLP structure in NFM the same as the pairwise analysis function h in SIGN for fair comparison. **DeepFM** (Guo et al. 2017): It combines interaction analysis results from using MLP and FM together for prediction. We set the MLP structure the same as the function h in SIGN. **xDeepFM** (Lian et al. 2018): It is an extension of DeepFM that models feature interactions in both explicit and implicit way. We use the same setting as DeepFM. **AutoInt** (Song et al. 2019): It explicitly models all feature interactions using a multi-head self-attentive neural network. We use the same neural network settings as our model.

Experimental set-up L_0-SIGN is a general model so that the implementation is flexible. In the experiments, we use element-wise mean as both linear aggregation functions $\psi(\cdot)$ and $\phi(\cdot)$. The linear function $g(\cdot)$ is a weighted sum function (i.e., $g(\boldsymbol{u}'_i) = \boldsymbol{w}_g^T \boldsymbol{u}'_i$, where $\boldsymbol{w}_g \in \mathbb{R}^{d\times 1}$ are the weight parameters). For the pairwise modeling function $h(\cdot)$, we use a MLP with one hidden layer after element-wise product: $h(\boldsymbol{u}_i, \boldsymbol{u}_j) = \boldsymbol{W}_2^h \sigma(\boldsymbol{W}_1^h(\boldsymbol{u}_i \odot \boldsymbol{u}_j) + \boldsymbol{b}_1^h) + \boldsymbol{b}_2^h$, where $\boldsymbol{W}_1^h, \boldsymbol{W}_2^h, \boldsymbol{b}_1^h, \boldsymbol{b}_2^h$ are parameters of MLP and $\sigma(\cdot)$ is a Relu activation function. We implement the edge prediction model based on the neural collaborative filtering framework (He and Chua 2017), which has a similar form to $h(\cdot)$: $f_{ep}(\boldsymbol{v}_i^e, \boldsymbol{v}_j^e) = \boldsymbol{W}_2^e \sigma(\boldsymbol{W}_1^e(\boldsymbol{v}_i^e \odot \boldsymbol{v}_j^e) + \boldsymbol{b}_1^e) + \boldsymbol{b}_2^e$. We set node embedding sizes for both pairwise modeling and edge prediction to 8 (i.e., $b, d = 8$) and the sizes of hidden layer for both h and f_{ep} to 32 (i.e., $\boldsymbol{W}_1^h, \boldsymbol{W}_1^e \in \mathbb{R}^{32\times 8}$). We choose the weighting factors λ_1 and λ_2 from $[0.001, 0.1]$ that produce the best performance in each dataset.

Each dataset is randomly split into training, validation, and test datasets with a proportion of 70%, 15%, and 15%. We choose the model parameters that produce the best results in validation set when the number of predicted edges being steady. We use accuracy (ACC) and the area under a curve with Riemann sums (AUC) as evaluation metrics.

Table 2: Summary of results in comparison with baselines.

	Frappe		MovieLens	
	AUC	ACC	AUC	ACC
FM	0.9263	0.8729	0.9190	0.8694
AFM	0.9361	0.8882	0.9205	0.8711
NFM	0.9413	0.8928	0.9342	0.8903
DeepFM	0.9422	0.8931	0.9339	0.8895
xDeepFM	0.9435	0.8950	0.9347	0.8906
AutoInt	0.9432	0.8947	0.9351	0.8912
SIGN	0.9448	0.8974	0.9354	0.8921
L_0-SIGN	**0.9580**	**0.9174**	**0.9407**	**0.8970**

Model Performance

We compare our model with recommendation baselines that model all feature interactions, and the results are in Table 2, with the best results for each dataset in bold. The results of SIGN come from using all feature interactions as input without detection, i.e., the input is a complete graph.

Through the table, we observe that: (i) L_0-SIGN outperforms all baselines. It shows L_0-SIGN's ability in providing accurate recommendations. Meanwhile, SIGN solely gains comparable results to competitive baselines, which shows the effectiveness of SIGN in modeling feature interactions for recommendation. (ii) L_0-SIGN gains significant improvement from SIGN. It shows that more accurate predictions can be delivered by retaining only beneficial feature interactions and effectively modeling them. (iii) FM and AFM (purely based on dot product to model interactions) gain lower accuracy than other models, which shows the necessity of using sophisticated methods (e.g., MLP) to model feature interactions for better predictions. (iv) The models that explicitly model feature interactions (xDeepFM, AutoInt, L_0-SIGN) outperform those that implicitly model them (NFM, DeepFM). It shows that explicit feature interaction analysis is promising in delivering accurate predictions.

Comparing SIGN with other GNNs in Our Model

To evaluate whether our L_0 edge prediction technique can be used on other GNNs and whether SIGN is more suitable than other GNNs in our model, we replace SIGN with existing GNNs: GCN (Kipf and Welling 2017), Chebyshev filter based GCN (Cheby) (Defferrard, Bresson, and Vandergheynst 2016) and GIN (Xu et al. 2019). We run on two datasets for graph classification since they contain heuristic edges (used to compare with the predicted edges):

Twitter (Pan, Wu, and Zhu 2015). It is extracted from twitter sentiment classification. Each tweet is regarded as a graph with nodes being word tokens. The edges are the co-occurrence relationship between two words in each tweet.

DBLP (Pan et al. 2013). It consists of papers with labels indicating either they are from DBDM or CVPR field. Each paper is a graph with nodes being paper ID or keywords. The edges are the citation relationship between papers or keyword relations in the title.

Table 3 shows the accuracies of each GNN that runs both on using the given (heuristic) edges (without "L_0" in name) and on predicting edges with our L_0 edge prediction model

Table 3: The results in comparison with existing GNNs. The model names without "L_0-" use **heuristic** edges, and those with "L_0-" automatically **detect** edges via our L_0 edge prediction technique.

	Twitter		DBLP	
	AUC	ACC	AUC	ACC
GCN	0.7049	0.6537	0.9719	0.9289
L_0-GCN	0.7053	0.6543	0.9731	0.9301
CHEBY	0.7076	0.6522	0.9717	0.9291
L_0-CHEBY	0.7079	0.6519	0.9719	0.9297
GIN	0.7149	0.6559	0.9764	0.9319
L_0-GIN	0.7159	0.6572	0.9787	0.9328
SIGN	0.7201	0.6615	0.9761	0.9316
L_0-SIGN	**0.7231**	**0.6670**	**0.9836**	**0.9427**

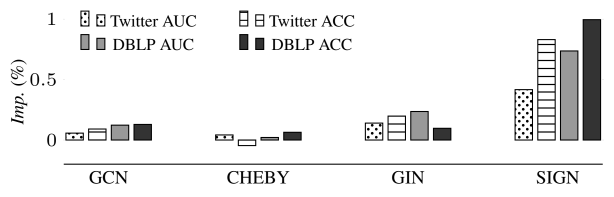


Figure 3: The comparison of accuracy improvement via the L_0 edge prediction technique to using heuristic edges in Table 3.

(with "L_0" in name). We can see that for all GNNs, L_0-GNNs gain competitive results comparing to corresponding GNNs. It shows that our model framework lifts the requirement of domain knowledge in defining edges in order to use GNNs in some situations. Also, L_0-SIGN outperforms other GNNs and L_0-GNNs, which shows L_0-SIGN's ability in detecting beneficial feature interactions and leveraging them to perform accurate predictions.

Figure 3 shows each GNN's improvement from the results of using given edges to predicting edges (from "GNN" to "L_0-GNN") in Table 3. It shows that among all the different GNN based models, SIGN gains the largest improvement from the L_0 edge prediction technique v.s. SIGN without L_0 edge prediction. It shows that SIGN can help the L_0 edge prediction model to better detect beneficial feature interactions for more accurate predictions.

Evaluation of Interaction Detection

We then evaluate the effectiveness of beneficial feature interaction detection in L_0-SIGN.

Prediction Accuracy vs Numbers of Edges Figure 4 shows the changes in prediction accuracy and the number of edges included while training. The accuracy first increases while the number of included edges decreases dramatically. Then, the number of edges becomes steady, and the accuracy reaches a peak at similar epochs. It shows that our model can recognize unbeneficial feature interactions and remove them for better prediction accuracy.

Using a different number of edges We evaluate how *predicted edges* (the retained edges) and *reversed edges* (the excluded edges) influence the performance. Specifically, we generate 5 edge sets with a different number of edges by

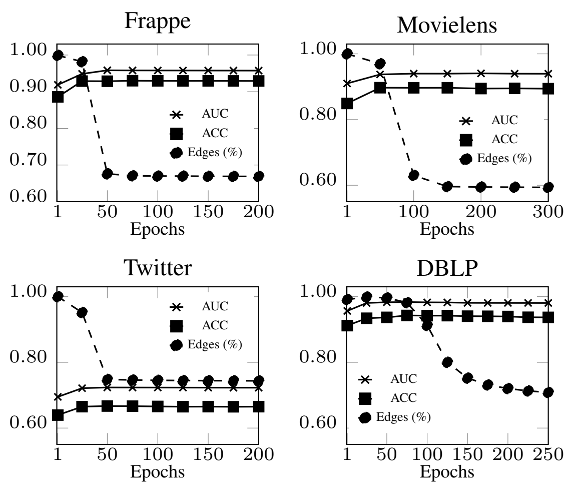


Figure 4: The changes in prediction accuracy and number of edges (in percentage) while training.

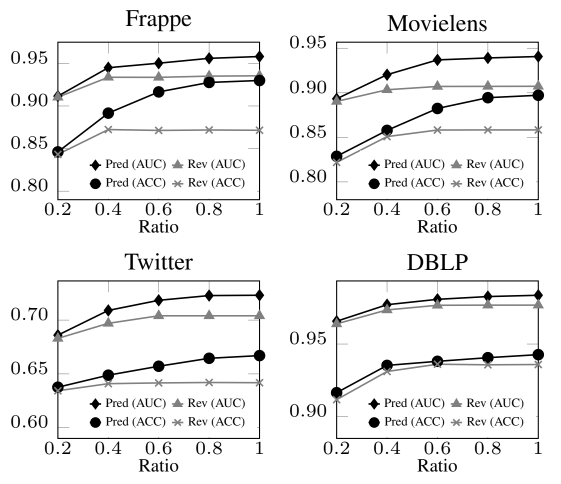
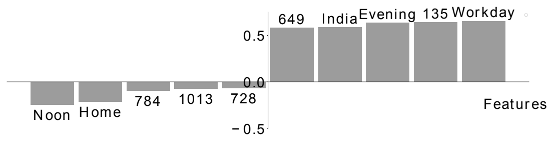
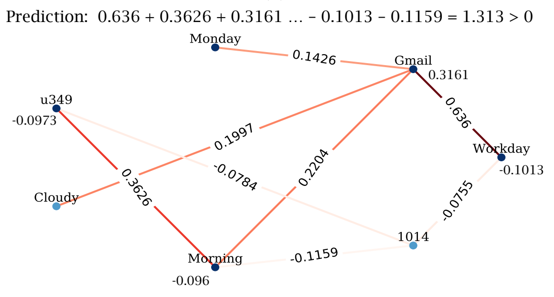


Figure 5: Evaluating different number of edges. “Pred” is the predicted edges and “Rev” is the reversed edges.

randomly selecting from 20% predicted edges (ratio 0.2) to 100% predicted edges (ratio 1.0). We generate another 5 edge sets similarly from reversed edges. Then, we run SIGN on each edge set for 5 times, and show the results in average in Figure 5. It shows that using predicted edges always gets higher accuracy than using reversed edges. Meanwhile, the accuracy stops increasing when using reversed edges since the ratio 0.6, while it continually improves when using more predicted edges. According to Definition 1, the detected edges are proved beneficial since considering more of them can provide further performance gain and the performance is the best when using all predicted edges, while considering the reversed edges cannot. Note that the increment of reversed edges from 0.2 to 0.6 may come from covering more nodes since features solely can provide some useful information for prediction.



(a)



(b)

Figure 6: (a) Highest beneficial interaction values for each orientation with *Gmail*. The number features are city indexes (e.g., 1014). (b) The prediction that the user *u349* will use *Gmail* as the prediction value is $1.313 > 0$. Darker edges are higher interaction values.

Case Study

Another advantage of detecting beneficial feature interactions is that potential explanations about recommendation predictions may be automatically discovered. Therefore, we conduct case studies on the Frappe dataset to show how L_0-SIGN illustrates the explanations.

We first show the beneficial interactions that have the highest statistical interaction analysis values with *Gmail* in Figure 6a. We can see that *Workday* has a high positive value, while *Home* has a high negative value. It may show that Gmail is very likely to be used in workdays since people need to use Gmail while working. However, Gmail is not likely to be used at home since people may not want to dealing with email while resting. We then show how L_0-SIGN provides potential explanations for the prediction of each data sample. Figure 6b visualizes a prediction result that a user (*u349*) may use *Gmail*. The graph shows useful information for explanations. Despite the beneficial interactions such as $< Gmail, Workday >$ that have discussed above, we can also see that *Cloudy* and *Morning* have no beneficial interaction, which means that whether it is a cloudy morning does not contribute to decide the user’s will of using Gmail.

6 Conclusion and Future Work

We propose L_0-SIGN, a GNN-based model that detects beneficial pairwise feature interactions via edge prediction, and leverages the beneficial feature interactions to perform recommendations via graph classification. Theoretical analyses and extensive experiments show the ability of L_0-SIGN in detecting and modeling beneficial feature interactions for accurate recommendations. In future work, we will extend our models to high-order feature interactions with theoretical foundations.

References

Alemi, A. A.; Fischer, I.; Dillon, J. V.; and Murphy, K. 2017. Deep Variational Information Bottleneck. In *ICLR*, 1–16.

Baltrunas, L.; Church, K.; Karatzoglou, A.; and Oliver, N. 2015. Frappe: Understanding the usage and perception of mobile app recommendations in-the-wild. *arXiv preprint arXiv:1505.03014* .

Battaglia, P. W.; Hamrick, J. B.; Bapst, V.; Sanchez-Gonzalez, A.; Zambaldi, V.; Malinowski, M.; Tacchetti, A.; Raposo, D.; Santoro, A.; and Faulkner, R. 2018. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261* .

Blondel, M.; Ishihata, M.; Fujino, A.; and Ueda, N. 2016. Polynomial Networks and Factorization Machines: New Insights and Efficient Training Algorithms. In *ICML*, 850–858.

Blum, A. L.; and Langley, P. 1997. Selection of Relevant Features and Examples in Machine Learning. *AI* 97(1-2): 245–271.

Chang, M. B.; Ullman, T.; Torralba, A.; and Tenenbaum, J. B. 2016. A compositional object-based approach to learning physical dynamics. In *ICLR*, 1–14.

Defferrard, M.; Bresson, X.; and Vandergheynst, P. 2016. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In *NeurIPS*, 3844–3852.

Duvenaud, D. K.; Maclaurin, D.; Iparraguirre, J.; Bombarell, R.; Hirzel, T.; Aspuru-Guzik, A.; and Adams, R. P. 2015. Convolutional networks on graphs for learning molecular fingerprints. In *NeurIPS*, 2224–2232.

Gilmer, J.; Schoenholz, S. S.; Riley, P. F.; Vinyals, O.; and Dahl, G. E. 2017. Neural message passing for quantum chemistry. In *ICML*, 1263–1272.

Guo, H.; Tang, R.; Ye, Y.; Li, Z.; and He, X. 2017. Deepfm: a factorization-machine based neural network for ctr prediction. In *IJCAI*, 1725–1731.

He, X.; and Chua, T.-S. 2017. Neural factorization machines for sparse predictive analytics. In *SIGIR*, 355–364.

Kipf, T. N.; and Welling, M. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*, 1–14.

Koren, Y. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *SIGKDD*, 426–434.

Langley, P.; et al. 1994. Selection of Relevant Features in Machine Learning. In *AAAI*, volume 184, 245–271.

Lian, J.; Zhou, X.; Zhang, F.; Chen, Z.; Xie, X.; and Sun, G. 2018. xdeepfm: Combining Explicit and Implicit Feature Interactions for Recommender Systems. In *SIGKDD*, 1754–1763.

Louizos, C.; Welling, M.; and Kingma, D. P. 2018. Learning Sparse Neural Networks through L_0 Regularization. In *ICLR*, 1–11.

MacKay, D. J. 2003. *Information Theory, Inference and Learning Algorithms*. Cambridge university press.

Maddison, C. J.; Mnih, A.; and Teh, Y. W. 2017. The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables. In *ICLR*, 1–12.

Menon, A. K.; and Elkan, C. 2011. Link Prediction via Matrix Factorization. In *ECML PKDD*, 437–452.

Merity, S.; McCann, B.; and Socher, R. 2017. Revisiting Activation Regularization for Language RNNs. In *ICML Workshop*.

Mitchell, T. J.; and Beauchamp, J. J. 1988. Bayesian variable selection in linear regression. *ASA* 83(404): 1023–1032.

Pan, S.; Wu, J.; and Zhu, X. 2015. CogBoost: Boosting for Fast Cost-sensitive Graph Classification. In *TKDE*, 2933–2946.

Pan, S.; Zhu, X.; Zhang, C.; and Philip, S. Y. 2013. Graph Stream Classification Using Labeled and Unlabeled Graphs. In *ICDE*, 398–409.

Rendle, S. 2010. Factorization Machines. In *ICDM*, 995–1000.

Santoro, A.; Raposo, D.; Barrett, D. G.; Malinowski, M.; Pascanu, R.; Battaglia, P.; and Lillicrap, T. 2017. A simple neural network module for relational reasoning. In *NeurIPS*, 4967–4976.

Shi, C.; Glocker, B.; and Castro, D. C. 2019. PVAE: Learning Disentangled Representations with Intrinsic Dimension via Approximated L0 Regularization. In *PMLR*, 1–6.

Shi, Y.; Larson, M.; and Hanjalic, A. 2014. Collaborative Filtering Beyond the User-item Matrix: A Survey of the State of the Art and Future Challenges. *CSUR* 47(1): 1–45.

Siegmund, N.; Kolesnikov, S. S.; Kästner, C.; Apel, S.; Batory, D.; Rosenmüller, M.; and Saake, G. 2012. Predicting Performance via Automated Feature-interaction Detection. In *ICSE*, 167–177.

Song, W.; Shi, C.; Xiao, Z.; Duan, Z.; Xu, Y.; Zhang, M.; and Tang, J. 2019. Autoint: Automatic Feature Interaction Learning via Self-attentive Neural Networks. In *CIKM*, 1161–1170.

Sorokina, D.; Caruana, R.; Riedewald, M.; and Fink, D. 2008. Detecting Statistical Interactions with Additive Groves of Trees. In *ICML*, 1000–1007.

Tishby, N.; Pereira, F. C.; and Bialek, W. 2000. The Information Bottleneck Method. *arXiv preprint physics/0004057* .

Tsang, M.; Liu, H.; Purushotham, S.; Murali, P.; and Liu, Y. 2018. Neural Interaction Transparency (NIT): Disentangling Learned Interactions for Improved Interpretability. In *NeurIPS*, 5804–5813.

Wang, X.; Girshick, R.; Gupta, A.; and He, K. 2018. Non-local neural networks. In *CVPR*, 7794–7803.

Xiao, J.; Ye, H.; He, X.; Zhang, H.; Wu, F.; and Chua, T.-S. 2017. Attentional factorization machines: Learning the weight of feature interactions via attention networks. In *IJCAI*, 3119–3125.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2019. How Powerful are Graph Neural Networks? *ICLR* 1–13.

Yang, C.; Bai, L.; Zhang, C.; Yuan, Q.; and Han, J. 2017. Bridging collaborative filtering and semi-supervised learning: a neural approach for poi recommendation. In *SIGKDD*, 1245–1254.

Zhang, C.; Bengio, S.; Hardt, M.; Recht, B.; and Vinyals, O. 2017. Understanding Deep Learning Requires Rethinking Generalization. In *ICLR*, 1–11.

A Derivation from IB to L_0-SIGN

Recall that the empirical risk minimization procedure of L_0-SIGN in Equation 6 is:

$$\begin{aligned}\mathcal{R}(\boldsymbol{\theta}, \boldsymbol{\omega}) =& \frac{1}{N}\sum_{n=1}^{N}(\mathcal{L}(F_{LS}(G_n; \boldsymbol{\omega}, \boldsymbol{\theta}), y_n) \\ &+ \lambda_1 \sum_{i,j\in X_n} (\pi_n)_{ij} + \lambda_2 \left\|\boldsymbol{z}_n\right\|_2),\end{aligned}$$

Deep variational information bottleneck method (Alemi et al. 2017) performs a variational approximation to the Information Bottleneck principle (Equation 7). Specifically, the function can be approximated by maximizing a lower bound L:

$$\begin{aligned}L \approx& \frac{1}{N}\sum_{n=1}^{N}[\int d\tilde{s}_n\, p(\tilde{s}_n \mid \tilde{x}_n) \log q(\tilde{y}_n \mid \tilde{s}_n) \\ &- \beta p(\tilde{s}_n \mid \tilde{x}_n) \log \frac{p(\tilde{s}_n \mid \tilde{x}_n)}{r(\tilde{s}_n)}],\end{aligned} \tag{10}$$

where $\tilde{x}_n, \tilde{y}_n, \tilde{s}_n$ are input, output and the middle state respectively, $r(\tilde{s}_n)$ is the variational approximation to the marginal $p(\tilde{s}_n)$.

Then, maximizing the lower bound L equals to minimizing the J_{IB}:

$$\begin{aligned}J_{IB} = \frac{1}{N}\sum_{n=1}^{N}&(\mathbb{E}_{\tilde{s}_n\sim p(\tilde{s}_n|\tilde{x}_n)}[-\log q(\tilde{y}_n \mid \tilde{s}_n)] \\ &+ \beta KL[p(\tilde{s}_n \mid \tilde{x}_n), r(\tilde{s}_n)]),\end{aligned} \tag{11}$$

where $KL[p(\tilde{s}_n \mid \tilde{x}_n), r(\tilde{s}_n)]$ is the Kullback–Leibler divergence between $p(\tilde{s}_n \mid \tilde{x}_n)$ and $r(\tilde{s}_n)$.

In L_0-SIGN, the middle state part between input and output is the statistical interaction analysis result $\boldsymbol{s}_n$, which is the multiplication of predicted edge values $\boldsymbol{e}_n^{'}$ (the vector form of $E_n^{'}$) and pairwise modeling results $\boldsymbol{z}_n$. $(e_n^{'})_{ij} = Bern((\pi_n)_{ij})$ so that $\boldsymbol{e}_n^{'}$ is a multivariate Bernoulli distribution, denoted as $p(\boldsymbol{e}_n^{'} \mid X_n)$. Similarly, $(\boldsymbol{z}_n)_{ij}$ is a multivariate normal distribution $\mathcal{N}((\boldsymbol{z}_n)_{ij}, \Sigma_{ij})$ so that $\boldsymbol{z}_n$ is a multivariate normal distribution, denoted as $p(\boldsymbol{z}_n \mid X_n)$. Therefore, the distribution of $\boldsymbol{s}_n$ (denoted as $p(\boldsymbol{s}_n \mid X_n)$) is represented as:

$$\begin{aligned}p(\boldsymbol{s}_n \mid X_n) &= p(\boldsymbol{e}_n^{'} \mid X_n)p(\boldsymbol{z}_n \mid X_n) \\ &= \|_{i,j\in X_n}[Bern(\pi_{ij})\mathcal{N}((\boldsymbol{z}_n)_{ij}, \Sigma_{ij})],\end{aligned} \tag{12}$$

where Σ_{ij} is a covariance matrix and $\|$ is concatenation.

Meanwhile, we set the variational approximation of the $\boldsymbol{s}_n$ being a concatenated multiplication of normal distributions with mean of 0 and variance of 1, and Bernoulli distributions that the probability of being 0 is 1. Then, the variational approximation in the vector form is:

$$r(\boldsymbol{s}_n) = Bern(\boldsymbol{0})\mathcal{N}(\boldsymbol{0}, \boldsymbol{I}), \tag{13}$$

where I is an identity matrix.

Combining Equation 12 and Equation 13 into Equation 11, the minimization function correlating to L_0-SIGN becomes:

$$\begin{aligned}J_{IB} = \frac{1}{N}\sum_{n=1}^{N}&(\mathbb{E}_{\boldsymbol{s}_n\sim p(\boldsymbol{s}_n|X_n)}[-\log q(y_n \mid \boldsymbol{s}_n)] \\ &+ \beta KL[p(\boldsymbol{s}_n \mid X_n), r(\boldsymbol{s}_n)]).\end{aligned} \tag{14}$$

Next, we use the forward KullbackLeibler divergence $KL[r(\boldsymbol{s}_n), p(\boldsymbol{s}_n \mid X_n)]$ to approximate the reverse KullbackLeibler divergence in Equation 14 to ensure the KullbackLeibler divergence can be properly derived into the L_0 and L_2 activation regularization (will be illustrated in Equation 17 and Equation 18). We can perform this approximation because when the variational approximation $r(\tilde{z}_n)$ only contains one mode (e.g., Bernoulli distribution, normal distribution), both forward and reverse KullbackLeibler divergence force $p(\tilde{z}_n \mid \tilde{x}_n)$ to cover the only mode and will have the same effect (MacKay 2003). Then Equation 14 becomes:

$$\begin{aligned}&J_{IB} \\ \approx& \frac{1}{N}\sum_{n=1}^{N}(\mathbb{E}_{\boldsymbol{s}_n\sim p(\boldsymbol{s}_n|X_n)}[-\log q(y_n \mid \boldsymbol{s}_n)] \\ &+ \beta KL[r(\boldsymbol{s}_n), p(\boldsymbol{s}_n \mid X_n)]) \\ =& \frac{1}{N}\sum_{n=1}^{N}(\mathbb{E}_{\boldsymbol{s}_n\sim p(\boldsymbol{s}_n|X_n)}[-\log q(y_n \mid \boldsymbol{s}_n)] \\ &+ \beta KL[Bern(\boldsymbol{0})\mathcal{N}(\boldsymbol{0}, \boldsymbol{I}), p(\boldsymbol{e}_n^{'} \mid X_n)p(\boldsymbol{z}_n \mid X_n)]) \\ =& \frac{1}{N}\sum_{n=1}^{N}(\mathbb{E}_{\boldsymbol{s}_n\sim p(\boldsymbol{s}_n|X_n)}[-\log q(y_n \mid \boldsymbol{s}_n)] \\ &+ \beta(dKL[Bern(\boldsymbol{0}), p(\boldsymbol{e}_n^{'} \mid X_n)] \\ &\quad + KL[\mathcal{N}(\boldsymbol{0}, \boldsymbol{I}), p(\boldsymbol{z}_n \mid X_n)])),\end{aligned} \tag{15}$$

where d is the dimention of each vector $(\boldsymbol{z}_n)_{ij}$.

Minimizing $\mathbb{E}_{\boldsymbol{s}_n\sim p(\boldsymbol{s}_n|X_n)}[-\log q(y_n \mid \boldsymbol{s}_n)]$ in Equation 15 is equivalent to minimizing $\mathcal{L}(f_{LS}(G_n; \boldsymbol{\omega}, \boldsymbol{\theta}), y_n)$ in Equation 6: its the negative log likelihood of the prediction as the loss function of L_0-SIGN, with $p(\boldsymbol{s}_n \mid X_n)$ being the edge prediction procedure and pairwise analysis procedure, and $q(y_n \mid \boldsymbol{s}_n)$ being the aggregation procedure from the statistical interaction analysis result to the outcome y_n.

For the part $KL[Bern(\boldsymbol{0}), p(\boldsymbol{e}_n^{'} \mid X_n)]$ in Equation 15, $p(\boldsymbol{e}_n^{'} \mid X_n) = Bern(\boldsymbol{\pi}_n)$ is a multivariate Bernoulli distribution, so the KL divergence is:

$$\begin{aligned}&KL[Bern(\boldsymbol{0}), p(\boldsymbol{e}_n^{'} \mid X_n)] = KL[Bern(\boldsymbol{0}), Bern(\boldsymbol{\pi}_n)] \\ =& \sum_{i,j\in X_n}(0 \log \frac{0}{(\pi_n)_{ij}} + (1-0)\log\frac{1-0}{1-(\pi_n)_{ij}}) \\ =& \sum_{i,j\in X_n} \log \frac{1}{1-(\pi_n)_{ij}}.\end{aligned} \tag{16}$$

Next, we use a linear function $\gamma(\pi_n)_{ij}$ to approximate $\log \frac{1}{1-(\pi_n)_{ij}}$ in Equation 16, where $\gamma > 0$ is a scalar

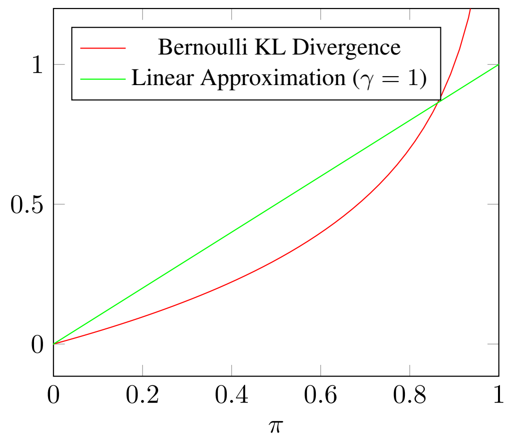


Figure 7: Linear Approximation vs. Bernoulli KL Divergence on different π values.

constant. Figure 7 shows the values (penalization) of the Bernoulli KL divergence and its linear approximation on different π values. It can be seen that both Bernoulli KL divergence and its approximations are monotonically increasing. In the empirical risk minimization procedure, they will have similar effects on penalizing those $\pi > 0$. In addition, the approximation is more suitable for our model because: (i) it penalizes more than the Bernoulli KL divergence when π is approaching 0 (take more effort on removing unbeneficial feature interactions); and (ii) it gives reasonable (finite) penalization when π is approaching 1 (retrain beneficial feature interactions), while the Bernoulli KL divergence produces infinite penalization when $\pi = 1$.

Then the KL divergence of the multivariate Bernoulli distribution can be approximately calculated by:

$$\begin{aligned} KL[Bern(\mathbf{0}), p(\boldsymbol{e}_n^{'} \mid X_n)] &= \sum_{i,j \in X_n} \log \frac{1}{1-(\pi_n)_{ij}} \\ &\approx \sum_{i,j \in X_n} \gamma(\pi_n)_{ij}. \end{aligned} \tag{17}$$

For the part $KL[\mathcal{N}(\mathbf{0}, \boldsymbol{I}), p(\boldsymbol{z}_n \mid X_n)]$, the distribution $p(\boldsymbol{z}_n \mid X_n)$ is a multivariate normal distribution and is denoted as $\mathcal{N}(\boldsymbol{z}_n, \Sigma)$. If we assume all normal distributions in $p(\boldsymbol{z}_n \mid X_n)$ are i.i.d, and have the same variance (i.e., $\Sigma = diag(\sigma^2, \sigma^2, \dots, \sigma^2)$ where σ is a constant), we can reformulate the KL divergence:

$$\begin{aligned} & KL[\mathcal{N}(\mathbf{0}, \boldsymbol{I}), p(\boldsymbol{z}_n \mid X_n)] = KL[\mathcal{N}(\mathbf{0}, \boldsymbol{I}), \mathcal{N}(\boldsymbol{z}_n, \Sigma_n)] \\ =& \frac{1}{2}(\mathrm{Tr}(\Sigma_n^{-1}\boldsymbol{I}) + (\boldsymbol{z}_n - \mathbf{0})^T \Sigma_1^{-1}(\boldsymbol{z}_n - \mathbf{0}) + \ln \frac{\det \Sigma_n}{\det \boldsymbol{I}} - d|X_n|) \\ =& \frac{1}{2} \sum_{i,j \in X_n} \sum_{k=1}^{d} (\frac{1}{\sigma^2} + \frac{(z_n)_{ijk}^2}{\sigma^2} + \ln \sigma^2 - 1) \\ =& \frac{1}{2\sigma^2} \sum_{i,j \in X_n} \sum_{k=1}^{d} ((z_n)_{ijk}^2 + C_2) \\ \propto& \frac{1}{2\sigma^2} \sum_{i,j \in X_n} \sum_{k=1}^{d} (z_n)_{ijk}^2, \end{aligned} \tag{18}$$

where $C_2 = 1 + \sigma^2 \ln \sigma^2 - \sigma^2$ is a constant and $(z_n)_{ijk}$ is the kth dimension of $(\boldsymbol{z}_n)_{ij}$.

Relating to Equation 6, Equation 17 is exactly the L_0 activation regularization part and Equation 18 is the L_2 activation regularization part in the empirical risk minimization procedure of L_0-SIGN, with $\lambda_1 = d\beta\gamma$ and $\lambda_2 = \frac{\beta}{2\sigma^2}$. Therefore, the empirical risk minimization procedure of L_0-SIGN is proved to be a variational approximation of minimizing the object function of IB (J_{IB}):

$$\min \mathcal{R}(\boldsymbol{\theta}, \boldsymbol{\omega}) \approx \min J_{IB}. \tag{19}$$

B Prove of Theorem 1

Theorem 1. ***(Statistical Interaction in SIGN)*** *Consider a graph $G(X, E)$, where X is the node set and $E = \{e_{ij}\}_{i,j \in X}$ is the edge set that $e_{ij} \in \{0, 1\}, e_{ij} = e_{ji}$. Let $G(X, E)$ be the input of SIGN function $f_S(G)$ in Equation 4, then the function flags pairwise statistical interaction between node i and node j if and only if they are linked by an edge in $G(X, E)$, i.e., $e_{ij} = 1$.*

Proof. We prove Theorem 1 by proving two lemmas:

Lemma 1.1. *Under the condition of Theorem 1, for a graph $G(X, E)$, if $f_S(G)$ shows pairwise statistical interaction between node i and node j, where $i, j \in X$, then the two nodes are linked by an edge in $G(X, E)$, i.e., $e_{ij} = 1$.*

Proof. We prove this lemma by contradiction. Assume that the SIGN function f_S with $G(X, E_{\setminus e_{ij}})$ as input shows pairwise statistical interaction between node i and node j, where $G(X, E_{\setminus e_{ij}})$ is a graph with $E_{\setminus e_{ij}}$ being a set of edges that $e_{ij} = 0$.

Recall that the SIGN function in Equation 4. Without losing generality, we set both the aggregation functions ϕ and ψ being element-wise average. That is:

$$f_S(G) = \frac{1}{|X|} \sum_{i \in X} (\frac{1}{\rho(i)} \sum_{j \in X} (e_{ij} h(\boldsymbol{u}_i, \boldsymbol{u}_j))), \tag{20}$$

where $\rho(i)$ is the degree of node i.

From Equation 20, we know that the SIGN function can be regarded as the linear aggregation of non-additive statistical interaction modeling procedures $h(\boldsymbol{u}_k, \boldsymbol{u}_m)$ for all nodes pairs (k, m) that $k, m \in X$ and $e_{km} = 1$. Since $E_{\setminus e_{ij}}$ does not contain an edge between i and j (i.e., $e_{ij} = 0$), the SIGN function does not perform interaction modeling between the two nodes into final predictions.

According to Definition 2, since i and j have statistical interaction, we cannot find a replacement form of SIGN function like:

$$\begin{aligned} f_S(G) =& q_{\setminus i}(\boldsymbol{u}_1, ..., \boldsymbol{u}_{i-1}, \boldsymbol{u}_{i+1}, ..., \boldsymbol{u}_{|X|}) \\ &+ q_{\setminus j}(\boldsymbol{u}_1, ..., \boldsymbol{u}_{j-1}, \boldsymbol{u}_{j+1}, ..., \boldsymbol{u}_{|X|}), \end{aligned} \tag{21}$$

where $q_{\setminus i}$ and $q_{\setminus j}$ are functions without node i and node j as input, respectively.

However, from our assumption, since there is no edge between node i and node j, there is no interaction modeling function that performs between them in $f_S(G)$. Therefore, we can easily find many such $q_{\setminus i}$ and $q_{\setminus j}$ that satisfy Equation 21. For example:

$$q_{\setminus i}(\boldsymbol{u}_1, ..., \boldsymbol{u}_{i-1}, \boldsymbol{u}_{i+1}, ..., \boldsymbol{u}_{|X|}) = \frac{1}{|X|} \sum_{k \in X \setminus \{i\}} (\frac{1}{\rho(k)} \sum_{m \in X \setminus \{i\}} (e_{km} h(\boldsymbol{u}_k, \boldsymbol{u}_m))), \tag{22}$$

and

$$\begin{aligned} q_{\setminus j}(\boldsymbol{u}_1, ..., \boldsymbol{u}_{j-1}, \boldsymbol{u}_{j+1}, ..., \boldsymbol{u}_{|X|}) = \\ \frac{1}{|X|} \sum_{m \in X \setminus \{j\}} (\frac{1}{\rho(i)} e_{im} h(\boldsymbol{u}_i, \boldsymbol{u}_m)) \\ + \frac{1}{|X|} \sum_{k \in X \setminus \{j\}} (\frac{1}{\rho(k)} e_{ki} h(\boldsymbol{u}_k, \boldsymbol{u}_i)). \end{aligned} \tag{23}$$

Therefore, it contradicts our assumption. Lemma 1.1 is proved. □

Lemma 1.2. *Under the condition of Theorem 1, for a graph $G(X, E)$, if an edge links node i and node j in G (i.e., $i, j \in X$ and $e_{ij} = 1$), then $f_S(G)$ shows pairwise statistical interaction between node i and node j.*

Proof. We prove this lemma by contradiction as well. Assume there is a graph $G(X, E)$ with a pair of nodes (i, j) that $e_{ij} = 1$, but shows no pairwise statistical interaction between this node pair in $f_S(G)$.

Since $e_{ij} = 1$, we can rewrite SIGN function as:

$$\begin{aligned} f_S(G) = & \frac{1}{|X|} \sum_{k \in X} (\frac{1}{\rho(k)} \sum_{m \in X} (e_{km} h(\boldsymbol{u}_k, \boldsymbol{u}_m))) \\ & + \frac{\rho(i) + \rho(j)}{|X| \rho(i) \rho(j)} (h(\boldsymbol{u}_i, \boldsymbol{u}_j)), \end{aligned} \tag{24}$$

where $(k, m) \notin \{(i, j), (j, i)\}$.

In our assumption, $f_S(G)$ shows no pairwise statistical interaction between node i and node j. That is, we can write $f_S(G)$ in the form of Equation 21 according to Definition 2. For the first component in the RHS of Equation 24, we can easily construct functions $q_{\setminus i}$ and $q_{\setminus j}$ in a similar way of Equation 22 and Equation 23 respectively. However, for the second component in the RHS of Equation 24, the non-additive function $h(\boldsymbol{u}_i, \boldsymbol{u}_j)$ operates on node i and node j. Through the definition of non-additive function, we cannot represent a non-additive function h as a form like $h(\boldsymbol{u}_i, \boldsymbol{u}_j) = f_1(\boldsymbol{u}_i) + f_2(\boldsymbol{u}_j)$, where f_1 and f_2 are functions. That is to say, we cannot merge the second component in the RHS into either $q_{\setminus i}$ or $q_{\setminus j}$.

Therefore, Equation 24 cannot be represented as the form of Equation 21, and the node pair (i, j) shows pairwise statistical interaction in $f_S(G)$, which contradicts our assumption. Lemma 1.2 is proved. □

Combing Lemma 1.1 and Lemma 1.2, Theorem 1 is proved. □

C Proof of Corollary 1.1

Corollary 1.1. ***(Statistical Interaction in L_0-SIGN)*** *Consider a graph G that the edge set is unknown. Let G be the input of L_0-SIGN function $F_{LS}(G)$ in Equation 5, the function flags pairwise statistical interaction between node i and node j if and only if they are predicted to be linked by an edge in G by L_0-SIGN, i.e., $e'_{ij} = 1$.*

Proof. In Equation 5, we can perform the prediction procedure by first predicting edge values on all potential node pairs. Then we perform node pair modeling and aggregating the results to get the predictions (as illustrated in Figure 1). Specifically, we can regard the edge prediction procedure in L_0-SIGN as being prior to the following SIGN procedure. The edge values in an input graph $G(X, \emptyset)$ can be first predicted by function F_{ep}. Then, we have the graph $G(X, E')$, where E' is a predicted edge set. Therefore, the following procedure is the same as the SIGN model with $G(X, E')$ as the input graph, which satisfies Theorem 1. □

D Algorithms

In this section, we provide the pseudocode of SIGN and L_0-SIGN prediction algorithms in Algorithm 1 and Algorithm 2, respectively. Meanwhile, we provide the pseudocode of SIGN and L_0-SIGN training algorithm in Algorithm 3 and Algorithm 4, respectively.

Algorithm 1 SIGN prediction function f_S

```
Input: data $G(X, E)$
for each pair of feature $(i, j)$ do
    if $e_{ij} = 1$ then
        $\boldsymbol{z_{ij}} = h(x_i \boldsymbol{v}_i, x_j \boldsymbol{v}_j)$
        $\boldsymbol{s}_{ij} = \boldsymbol{z}_{ij}$
    else
        $\boldsymbol{s}_{ij} = \boldsymbol{0}$
    end if
end for
for each feature $i$ do
    $\boldsymbol{v}'_i = \psi(\varsigma_i)$
    $\boldsymbol{u}'_i = x_i \boldsymbol{v}'_i$
    $\nu_i = g(\boldsymbol{u}'_i)$
end for
$y' = \phi(\nu)$
Return: $y'$
```

Algorithm 2 L_0-SIGN prediction function f_{LS}

```
**Input:** data $G(X, \emptyset)$
**for** each pair of feature $(i, j)$ **do**
    $e_{ij}^{'} = HardConcrete(f_{ep}(\boldsymbol{v}_i^e, \boldsymbol{v}_j^e))$
    $\boldsymbol{z_{ij}} = h(x_i\boldsymbol{v}_i, x_j\boldsymbol{v}_j)$
    $\boldsymbol{s}_{ij} = e_{ij}^{'}\boldsymbol{z}_{ij}$
**end for**
**for** each feature $i$ **do**
    $\boldsymbol{v}_i^{'} = \psi(\varsigma_i)$
    $\boldsymbol{u}_i^{'} = x_i\boldsymbol{v}_i^{'}$
    $\nu_i = g(\boldsymbol{u}_i^{'})$
**end for**
$y^{'} = \phi(\nu)$
**Return:** $y^{'}$
```

Algorithm 3 Training procedure of SIGN

```
Randomly initialize $\boldsymbol{\theta}$
**repeat**
    **for** each input-output pair $(G_n(X_n, E_n), y_n)$ **do**
        get $y_n^{'}, \boldsymbol{v}_n^{'}$ from $f_S(G_n; \boldsymbol{\theta})$
        $\boldsymbol{v}_n = \boldsymbol{v}_n^{'}$
    **end for**
    $\mathcal{R}(\boldsymbol{\theta}) = \frac{1}{N}\sum_{n=1}^{N}(\mathcal{L}(F_{LS}(y_n^{'}, y_n))$
    update $\boldsymbol{\theta}$ (exclude $\boldsymbol{v}$) by $\min \mathcal{R}(\boldsymbol{\theta})$
**until** reach the stop conditions
```

Algorithm 4 Training procedure of L_0-SIGN

```
Randomly initialize $\boldsymbol{\theta}, \boldsymbol{\omega}$
**repeat**
    **for** each input-output pair $(G_n(X_n, \emptyset), y_n)$ **do**
        get $y_n^{'}, \boldsymbol{v}_n^{'}$ from $f_{LS}(G_n; \boldsymbol{\theta}, \boldsymbol{\omega})$
        $\boldsymbol{v}_n = \boldsymbol{v}_n^{'}$
    **end for**
    calculate $\mathcal{R}(\boldsymbol{\theta}, \boldsymbol{\omega})$ through Equation 6
    update $\boldsymbol{\omega}, \boldsymbol{\theta}$(exclude $\boldsymbol{v}$) by $\min \mathcal{R}(\boldsymbol{\theta}, \boldsymbol{\omega})$
**until** reach the stop conditions
```

E L_0 Regularization

L_0 regularization encourages the regularized parameters $\boldsymbol{\theta}$ to be exactly zero by setting an L_0 term:

$$||\boldsymbol{\theta}||_0 = \sum_{j=1}^{|\boldsymbol{\theta}|} \mathbb{I}[\theta_j \neq 0], \tag{25}$$

where $|\boldsymbol{\theta}|$ is the dimensionality of the parameters and $\mathbb{I}$ is 1 if $\theta_j \neq 0$, and 0 otherwise.

For a dataset D, an empirical risk minimization procedure is used with L_0 regularization on the parameters $\boldsymbol{\theta}$ of a hypothesis $\mathcal{H}(\cdot; \boldsymbol{\theta})$, which can be any objective function involving parameters, such as neural networks. Then, using reparameterization of $\boldsymbol{\theta}$, we set $\theta_j = \tilde{\theta}_j z_j$, where $\tilde{\theta}_j \neq 0$ and z_j is a binary gate with Bernoulli distribution $Bern(\pi_j)$ (Louizos, Welling, and Kingma 2018). The procedure is represented as:

$$\mathcal{R}(\tilde{\boldsymbol{\theta}}, \boldsymbol{\pi}) = \mathbb{E}_{p(\boldsymbol{z}|\boldsymbol{\pi})}\frac{1}{N}(\sum_{n=1}^{N}\mathcal{L}(\mathcal{H}(X_n; \tilde{\boldsymbol{\theta}} \odot \boldsymbol{z}), y_n)) + \lambda\sum_{j=1}^{|\boldsymbol{\theta}|}\pi_j,$$
$$\tilde{\boldsymbol{\theta}}^*, \boldsymbol{\pi}^* = \arg\min_{\tilde{\boldsymbol{\theta}}, \boldsymbol{\pi}} \mathcal{R}(\tilde{\boldsymbol{\theta}}, \boldsymbol{\pi}), \tag{26}$$

where $p(z_j|\pi_j) = Bern(\pi_j)$, N is the number of samples in D, $\odot$ is element-wise production, $\mathcal{L}(\cdot)$ is a loss function and λ is the weighting factor of the L_0 regularization.

F Approximate L_0 Regularization with Hard Concrete Distribution.

A practical difficulty of performing L_0 regularization is that it is non-differentiable. Inspired by (Louizos, Welling, and Kingma 2018), we smooth the L_0 regularization by approximating the binary edge value with a hard concrete distribution. Specifically, let f_{ep} now output continuous values. Then

$$\begin{aligned} &u \sim \mathcal{U}(0, 1),\\ &s = Sigmoid((\log u - \log(1-u) + \log(\alpha_{ij}))/\beta),\\ &\bar{s} = s(\delta - \gamma) + \gamma,\\ &e_{ij}^{'} = min(1, max(0, \bar{s})), \end{aligned} \tag{27}$$

where $u \sim \mathcal{U}(0, 1)$ is a uniform distribution, Sig is the Sigmoid function, $\alpha_{ij} \in \mathbb{R}^+$ is the output of $f_{ep}(\boldsymbol{v}_i^e, \boldsymbol{v}_j^e)$, β is the temperature and (γ, δ) is an interval with $\gamma < 0$, $\delta > 0$.

Therefore, the L_0 activation regularization is changed to: $\sum_{i,j\in X_n}(\pi_n)_{ij} = \sum_{i,j\in X_n} Sig(\log\alpha_{ij} - \beta\log\frac{-\gamma}{\delta})$.

As a result, $e_{ij}^{'}$ follows a hard concrete distribution through the above approximation, which is differentiable and approximates a binary distribution. Following the recommendations from (Maddison, Mnih, and Teh 2017), we set $\gamma = -0.1, \delta = 1.1$ and $\beta = 2/3$ throughout our experiments. We refer interested readers to (Louizos, Welling, and Kingma 2018) for details about hard concrete distributions.