Heterogeneous Graph Structure Learning for Graph Neural Networks

Jianan Zhao,[1,2*] Xiao Wang,[1*] Chuan Shi,[1†] Binbin Hu,[3] Guojie Song,[4] Yanfang Ye[2]

[1] School of CS, Beijing University of Posts and Telecommunications, Beijing, China
[2] Department of CDS, Case Western Reserve University, OH, USA
[3] Ant Group
[4] Key Laboratory of Machine Perception, Ministry of Education, Peking University
jxz1244@case.edu, {xiaowang, shichuan}@bupt.edu.cn,
bin.hbb@antfin.com, gjsong@pku.edu.cn, yanfang.ye@case.edu

Abstract

Heterogeneous Graph Neural Networks (HGNNs) have drawn increasing attention in recent years and achieved outstanding performance in many tasks. The success of the existing HGNNs relies on one fundamental assumption, i.e., the original heterogeneous graph structure is reliable. However, this assumption is usually unrealistic, since the heterogeneous graph in reality is inevitably noisy or incomplete. Therefore, it is vital to learn the heterogeneous graph structure for HGNNs rather than rely only on the raw graph structure. In light of this, we make the first attempt towards learning an optimal heterogeneous graph structure for HGNNs and propose a novel framework HGSL, which jointly performs Heterogeneous Graph Structure Learning and GNN parameter learning for classification. Different from traditional homogeneous graph structure learning, considering the heterogeneity of different relations in heterogeneous graph, HGSL generates each relation subgraph separately. Specifically, in each generated relation subgraph, HGSL not only considers the feature similarity by generating feature similarity graph, but also considers the complex heterogeneous interactions in features and semantics by generating feature propagation graph and semantic graph. Then, these graphs are fused to a learned heterogeneous graph and optimized together with a GNN towards classification objective. Extensive experiments on real-world graphs demonstrate that the proposed framework significantly outperforms the state-of-the-art methods.

Introduction

Many real-world data can be viewed as graphs, such as bibliographic graphs, citation graphs, and social media graphs. Graph Neural Network (GNN), as a powerful deep representation learning tool to deal with graph data, has drawn increasing attention and is widely applied to node classification (Kipf and Welling 2017; Velickovic et al. 2018; Xu et al. 2019), graph classification (Duvenaud et al. 2015; Lee, Lee, and Kang 2019), and recommendation (Ying et al. 2018; Fan et al. 2019b; Wang et al. 2019a). Recently, with the proliferation of real-world applications on heterogeneous graphs, which consist of multiple types of nodes and links (Shi et al. 2017), Heterogeneous Graph Neural Networks (HGNNs) are brought forward and have achieved remarkable improvements on series of applications (Wang et al. 2020a; Hu et al. 2019b; Li et al. 2019; Hu et al. 2019a; Fan et al. 2019a; Wang et al. 2020b).

Most HGNNs follow a message-passing scheme where the node embedding is learned by aggregating and transforming the embeddings of its original neighbors (Zhang et al. 2019a; Zhao et al. 2020a; Hong et al. 2020) or metapath-based neighbors (Wang et al. 2019b; Yun et al. 2019; Hu et al. 2020; Fu et al. 2020). These methods rely on one fundamental assumption, i.e. the raw heterogeneous graph structure is good. However, as heterogeneous graphs are usually extracted from complex interaction systems by some pre-defined rules, such assumption cannot be always satisfied. One reason is that, these interaction systems inevitably contain some uncertain information or mistakes. Taking a user-item graph built from recommendation as an example, it is well accepted that users may misclick some unwanted items, bringing noisy information to the graph. The other reason is that, the heterogeneous graphs are often extracted with data cleaning, feature extraction and feature transformation by some pre-defined rules, which are usually independent to the downstream tasks and lead to the gap between the extracted graph and the optimal graph structure for the downstream tasks. Therefore, learning an optimal heterogeneous graph for GNN is a fundamental problem.

Recently, to adaptively learn graph structures for GNNs, graph structure learning (GSL) methods (Franceschi et al. 2019; Jiang et al. 2019; Chen, Wu, and Zaki 2019; Jin et al. 2020) are proposed, most of which parameterize the adjacency matrix and optimize it along with the GNN parameters toward downstream tasks. However, these methods are all designed for homogeneous graphs, which can not be directly applied to heterogeneous graphs with the following challenges: (1) *The heterogeneity in heterogeneous graphs* When learning a homogeneous graph with only one type of relation, we usually only need to parameterize one adjacency matrix. However, a heterogeneous graph consists of multiple relations, each of which reflects one aspect of the heterogeneous graph. Since treating these heterogeneous relations uniformly will inevitably restrict the capability of graph structure learning. How to deal with this heterogeneity is a challenging problem. (2) *The complex interactions in heterogeneous graphs.* Different relations and node fea-

*Both authors contributed equally to this research.
†Corresponding author.

Copyright © 2021, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

tures have complex interactions, which drives the formation of different kinds of underlying graph structure (Zhang, Swami, and Chawla 2019). Moreover, the combination of different relations further forms a large number of high-order relationships with diverse semantics, which also implies distinct ways of graph generation. The heterogeneous graph structure will be affected by all these factors, therefore, these complex interactions must be thoroughly considered in heterogeneous graph structure learning.

In this paper, we make the first attempt to investigate **H**eterogeneous **G**raph **S**tructure **L**earning for graph neural networks, and propose a novel framework HGSL. In HGSL, the heterogeneous graph and the GNN parameters are jointly learned towards better node classification performance. Particularly, in the graph learning part, aiming to capture the heterogeneous metric of different relation generation, each relation subgraph is separately learned. Specifically, for each relation, three types of candidate graphs, i.e. the feature similarity graph, feature propagation graphs and semantic graphs, are generated by mining the complex correlations from heterogeneous node features and graph structures. The learned graphs are further fused to a heterogeneous graph and fed to a GNN. The graph learning parameters and the GNN parameters are jointly optimized towards classification objective. Our major contributions are highlighted as follows:

- A suitable heterogeneous graph structure is one basic guarantee of a successful HGNN. To our best knowledge, we make the first attempt to study how to learn an optimal heterogeneous graph structure for GNN towards downstream task.
- We propose a novel heterogeneous graph neural network with heterogeneous graph structure learning, where three kinds of graph structures (feature similarity graph, feature propagation graph, semantic graph) are generated, so as to comprehensively fuse an optimal heterogeneous graph for GNN.
- We conduct extensive experiments on three real-world datasets to validate the effectiveness of HGSL against the state-of-the-art methods.

Related Work

Graph Neural Network (GNN)

Most of current GNNs can be generally divided into two categories: spectral GNNs and spatial GNNs (Wu et al. 2019b). Specifically, spectral GNNs learn node representation based on graph spectral theory. For example, (Bruna et al. 2014) designs the graph convolution operation in Fourier domain by the graph Laplacian. Then, ChebNet (Defferrard, Bresson, and Vandergheynst 2016) utilizes Chebyshev polynomials as the convolution filter to improve the efficiency. GCN (Kipf and Welling 2017) simplifies ChebNet by using its first-order approximation. Further, SGC (Wu et al. 2019a) reduces the graph convolution to a linear model and still achieves competitive performance. Spatial GNNs define convolution operations directly on the graph, utilizing spatially close neighbors. For instance, GAT (Velickovic et al. 2018) aggregates neighborhood representations with attention mechanism. GraphSAGE (Hamilton, Ying, and Leskovec 2017) performs inductive graph convolution by aggregating information from sampled neighbors. For better efficiency, FastGCN (Chen, Ma, and Xiao 2018) performs importance sampling on each convolution layer. Readers may refer to these elaborate surveys (Zhang, Cui, and Zhu 2018; Wu et al. 2019b) for a thorough review.

Heterogeneous Graph Neural Network (HGNN)

HGNNs are proposed to deal with the ubiquitous heterogeneous data. Some HGNNs perform graph convolution directly on the original heterogeneous graphs. HGAT (Hu et al. 2019b) aggregates in node and type level information with attention mechanism for short-text classification. HetGNN (Zhang et al. 2019a) samples heterogeneous neighbors by random walk and then aggregates node and type information. To solve the metapath selection conundrum, HetSANN (Hong et al. 2020) aggregates multi-relational information of projected nodes by means of attention mechanism. NSHE (Zhao et al. 2020a) preserves the pairwise and network schema structure. HGT (Hu et al. 2020) adopts the meta-relation based mutual attention to perform message passing on heterogeneous graphs and learns the implicit meta paths. Other HGNN methods use metapaths to generate graphs and apply GNN afterwards. GraphInception (Zhang et al. 2018) applies graph convolution on metapaths based homogeneous graphs to perform collective classification. HAN (Wang et al. 2019b) applies node-level and semantic-level attention on metapath-based graphs. GTN (Yun et al. 2019) performs metapath generation via stacking multiple graph transformer layers and perform graph convolution afterwards. MAGNN (Fu et al. 2020) applies intra-metapath and inter-metapath aggregation on metapath instances.

Graph Structure Learning (GSL)

To alleviate the limitation that GNNs rely on the good quality of raw graph structure, several efforts have been made (Zhang et al. 2019b; Zheng et al. 2020; Ye and Ji 2019; Zhao et al. 2020b; Wang et al. 2020c; Pei et al. 2020). Besides these efforts, very recently, graph structure learning was proposed. GSL methods aim to learn the graph structure and GNN parameters jointly. To illustrate, LDS (Franceschi et al. 2019) models each edge inside the adjacency matrix as a parameter and learns them along with GNN parameters in a bi-level fashion. GLCN (Jiang et al. 2019) generates similarity-based graph structure from node features. IDGL (Chen, Wu, and Zaki 2019) iteratively learns the metrics to generate graph structure from node features and GNN embeddings. ProGNN (Jin et al. 2020) jointly learns GNN parameters and a robust graph structure with graph properties. However, these aforementioned GSL methods are all designed for homogeneous graphs.

Preliminaries

In this section, we introduce some basic concepts and formalize the problem of heterogeneous graph structure learning.

Definition 1. Heterogeneous Graph A Heterogeneous graph $G = (V; E; \mathcal{F})$ is composed of a node set V, an edge set E, and a feature set $\mathcal{F}$, along with the node type mapping function $: V \to \mathcal{T}$, and the edge type mapping function $: E \to \mathcal{R}$, where $\mathcal{T}$ and $\mathcal{R}$ denotes the node and edge types, $|\mathcal{T}| + |\mathcal{R}| > 2$. Let V denote the node set of type $\in \mathcal{T}$, the feature set $\mathcal{F}$ is composed of $|\mathcal{T}|$ feature matrix, $\mathcal{F} = \{\mathbf{F} \; ; \; \in \mathcal{T}\}$, $\mathbf{F} \in \mathbb{R}^{|V| \; | \; d}$, where V stands for all the node with type, d stands for the feature dimension of nodes.

Definition 2. Metapath A metapath P is defined as a path in the form of $v_1 \xrightarrow{r_1} v_2 \xrightarrow{r_2} \cdots \xrightarrow{r_l} v_{l+1}$, which describes a composite relation $r_1 \circ r_2 \circ \cdots \circ r_l$ between two nodes v_1 and v_{l+1}, where $\circ$ denotes the composition operator on relations.

Definition 3. Node Relation Triple A node relation triple $\langle v_i; r; v_j \rangle$, describes that two nodes v_i (head node) and v_j (tail node) are connected by relation $r \in \mathcal{R}$. We further define the type mapping functions $_h; \; _t : \mathcal{R} \to \mathcal{T}$ that map the relation to its head node type and tail node type respectively.
Example: In a user-item heterogeneous graph, say $r = \backslash UI"$ (a user buys an item), then we have $_h(r) = \backslash User"$ and $_t(r) = \backslash Item"$.

Definition 4. Relation Subgraph Given a heterogeneous graph $G = (V; E; \mathcal{F})$, a relation subgraph G_r is a subgraph of G that contains all node-relation triples with relation r. The adjacency matrix of G_r is $\mathbf{A}_r \in \mathbb{R}^{|V| \; _h(r)| \; |V| \; _t(r)|}$, where $\mathbf{A}_r[i; j] = 1$ if $\langle v_i; r; v_j \rangle$ exists in G_r, otherwise $\mathbf{A}_r[i; j] = 0$. $\mathcal{A}$ denotes the relation subgraph set of all the relation subgraphs in G, i.e. $\mathcal{A} = \{\mathbf{A}_r; r \in \mathcal{R}\}$.
Definition 5. Heterogeneous Graph Structure Learning (HGSL). Given a heterogeneous graph G, the task of heterogeneous graph structure learning is to jointly learn a heterogeneous graph structure, i.e. a new relational subgraph set $\mathcal{A}^0$, and the parameters of GNN for downstream tasks.

The Proposed Method

Model Framework

Figure 1 (a) illustrates the framework of the proposed HGSL. As we can see, given a heterogeneous graph, HGSL firstly constructs the semantic embedding matrices $\mathcal{Z}$ by metapath-based node embeddings from M metapaths. Afterwards, the heterogeneous graph structure and GNN parameters are trained jointly. For the graph learning part, HGSL takes the information from the original relation subgraph, the node features, and the semantic embeddings as input and generates relation subgraphs separately. Specifically, taking relation r_1 as an example, HGSL learns a feature graph $\mathbf{S}_{r_1}^{Feat}$ and a semantic graph $\mathbf{S}_{r_1}^{Sem}$ and fuse them with the original graph $\mathbf{A}_{r_1}$ to obtain the learned relation subgraph $\mathbf{A}_{r_1}^0$. Then, the learned subgraphs are fed into a GNN and a regularizer to perform node classification with regularization. By minimizing the regularized classification loss, HGSL optimizes the graph structure and the GNN parameters jointly.

Figure 1: Overview of the HGSL framework. (a) Model framework. (b) Feature graph generator. (c) Semantic graph generator.

Feature Graph Generator

Since the original graph may not be optimal for downstream task, a natural idea would be to augment the original graph structure via fully utilizing the rich information inside heterogeneous node features. Usually, there are two factors that affect the formation of graph structure based on features. One is the similarity between node features, and the other is the relationship between node feature and relation in HIN (Wang et al. 2020c). As shown in Figure 1 (b), we first propose to generate a feature similarity graph that captures the potential relationship generated by node features via heterogeneous feature projection and metric learning. Then we propose to generate the feature propagation graph, by propagating feature similarity matrices through topology structure. Finally, the generated feature similarity graph and feature propagation graph are aggregated to a final feature graph through a channel attention layer.

Feature Similarity Graph The feature similarity graph $\mathbf{S}_r^{FS}$ determines the possibility of an edge with type $r \in \mathcal{R}$ between two nodes based on node features. Specifically, for each node v_i of type (v_i) with feature vector $\mathbf{f}_i \in \mathbb{R}^{1 \; d_{(v_i)}}$, we adopt a type-specific mapping layer to project the feature $\mathbf{f}_i$ to a d_c-dimensional common feature $\mathbf{f}_i^0 \in \mathbb{R}^{1 \; d_c}$:

$$\mathbf{f}_i^0 = \quad \mathbf{f}_i \cdot \mathbf{W}_{(v_i)} + \mathbf{b}_{(v_i)} \; ; \tag{1}$$

where $(\cdot)$ denotes a non-linear activation function, $\mathbf{W} \in \mathbb{R}^{d_{(v_i)} \; d_c}$ and $\mathbf{b} \in \mathbb{R}^{1 \; d_c}$ denote the mapping matrix and the bias vector of type (v_i), respectively. Then, for a relation r, we perform metric learning on the common features and obtain the learned feature similarity graph $\mathbf{S}_r^{FS} \in \mathbb{R}^{|V| \; _h(r)| \; |V| \; _t(r)|}$, where the edge between nodes v_i and v_j is obtained by:

$$\mathbf{S}_r^{FS}[i; j] = \begin{cases} {}_r^{FS}(\mathbf{f}_i^0; \mathbf{f}_j^0) & {}_r^{FS}(\mathbf{f}_i^0; \mathbf{f}_j^0) \geq {}^{FS} \\ 0 & \text{otherwise,} \end{cases} \tag{2}$$

where ${}^{FS} \in [0; 1]$ is the threshold that controls the sparsity of feature similarity graph, and larger ^{FS} implies a more