Chapter 14
Graph Neural Networks: Graph Structure Learning

Yu Chen and Lingfei Wu

Abstract Due to the excellent expressive power of Graph Neural Networks (GNNs) on modeling graph-structure data, GNNs have achieved great success in various applications such as Natural Language Processing, Computer Vision, recommender systems, drug discovery and so on. However, the great success of GNNs relies on the quality and availability of graph-structured data which can either be noisy or unavailable. The problem of graph structure learning aims to discover useful graph structures from data, which can help solve the above issue. This chapter attempts to provide a comprehensive introduction of graph structure learning through the lens of both traditional machine learning and GNNs. After reading this chapter, readers will learn how this problem has been tackled from different perspectives, for different purposes, via different techniques, as well as its great potential when combined with GNNs. Readers will also learn promising future directions in this research area.

14.1 Introduction

Recent years have seen a significantly increasing amount of interest in Graph Neural Networks (GNNs) (Kipf and Welling, 2017b; Bronstein et al, 2017; Gilmer et al, 2017; Hamilton et al, 2017b; Li et al, 2016b) with a wide range of applications in Natural Language Processing (Bastings et al, 2017; Chen et al, 2020p), Computer Vision (Norcliffe-Brown et al, 2018), recommender systems (Ying et al, 2018b), drug discovery (You et al, 2018a) and so on. GNN's powerful ability in learning expressive graph representations relies on the quality and availability of graph-structured data. However, this poses some challenges for graph representation

Yu Chen
Facebook AI, e-mail: hugochan2013@gmail.com

Lingfei Wu
JD.COM Silicon Valley Research Center, e-mail: lwu@email.wm.edu

learning with GNNs. On the one hand, in some scenarios where the graph structure is already available, most of the GNN-based approaches assume that the given graph topology is perfect, which does not necessarily hold true because i) the real-word graph topology is often noisy or incomplete due to the inevitably error-prone data measurement or collection; and ii) the intrinsic graph topology might merely represent physical connections (e.g the chemical bonds in molecule), and fail to capture abstract or implicit relationships among vertices which can be beneficial for certain downstream prediction task. On the other hand, in many real-world applications such as those in Natural Language Processing or Computer Vision, the graph representation of the data (e.g., text graph for textual data or scene graph for images) might be unavailable. Early practice of GNNs (Bastings et al, 2017; Xu et al, 2018d) heavily relied on manual graph construction which requires extensive human effort and domain expertise for obtaining a reasonably performant graph topology during the data preprocessing stage.

In order to tackle the above challenges, graph structure learning aims to discover useful graph structures from data for better graph representation learning with GNNs. Recent attempts (Chen et al, 2020m,o; Liu et al, 2021; Franceschi et al, 2019; Ma et al, 2019b; Elinas et al, 2020; Velickovic et al, 2020; Johnson et al, 2020) focus on joint learning of graph structures and representations without resorting to human effort or domain expertise. Different sets of techniques have been developed for learning discrete graph structures and weighted graph structures for GNNs. More broadly speaking, graph structure learning has been widely studied in the literature of traditional machine learning in both unsupervised learning and supervised learning settings (Kalofolias, 2016; Kumar et al, 2019a; Berger et al, 2020; Bojchevski et al, 2017; Zheng et al, 2018b; Yu et al, 2019a; Li et al, 2020a). Besides, graph structure learning is also closely related to important problems such as graph generation (You et al, 2018a; Shi et al, 2019a), graph adversarial defenses (Zhang and Zitnik, 2020; Entezari et al, 2020; Jin et al, 2020a,e) and transformer models (Vaswani et al, 2017).

This chapter is organized as follows. We will first introduce how graph structure learning has been studied in the literature of traditional machine learning, prior to the recent surge of GNNs (section 14.2). We will introduce existing works on both unsupervised graph structure learning (section 14.2.1) and supervised graph structure learning (section 14.2.2). Readers will later see how some of the introduced techniques originally developed for traditional graph structure learning have been revisited and improve graph structure learning for GNNs. Then we will move to our main focus of this chapter which is graph structure learning for GNNs in section 14.3. This part will cover various topics including joint graph structure and representation learning for both unweighted and weighted graphs (section 14.3.1), and the connections to other problems such as graph generation, graph adversarial defenses and transformers (section 14.3.2). We will highlight some future directions in section 24.5 including robust graph structure learning, scalable graph structure learning, graph structure learning for heterogeneous graphs, and transferable graph structure learning. We will summarize this chapter in section 14.5.

14.2 Traditional Graph Structure Learning

Graph structure learning has been widely studied from different perspectives in the literature of traditional machine learning, prior to the recent surge of Graph Neural Networks. Before we move to the recent achievements of graph structure learning in the field of Graph Neural Networks, which is the main focus of this chapter, in this section, we will first examine this challenging problem through the lens of traditional machine learning.

14.2.1 Unsupervised Graph Structure Learning

The task of unsupervised graph structure learning aims to directly learn a graph structure from a set of data points in an unsupervised manner. The learned graph structure may be later consumed by subsequent machine learning methods for various prediction tasks. The most important benefit of this kind of approaches is that they do not require labeled data such as ground-truth graph structures for supervision, which could be expensive to obtain. However, because the graph structure learning process does not consider any particular downstream prediction task on the data, the learned graph structure might be sub-optimal for the downstream task.

14.2.1.1 Graph Structure Learning from Smooth Signals

Graph structure learning has been extensively studied in the literature of Graph Signal Processing (GSP). It is often referred to as the graph learning problem in the literature whose goal is to learn the topological structure from smooth signals defined on the graph in an unsupervised manner. These graph learning techniques (Jebara et al, 2009; Lake and Tenenbaum, 2010; Kalofolias, 2016; Kumar et al, 2019a; Kang et al, 2019; Kumar et al, 2020; Bai et al, 2020a) typically operate by solving an optimization problem with certain prior constraints on the properties (e.g., smoothness, sparsity) of graphs. Here, we introduce some representative prior constraints defined on graphs which have been widely used for solving the graph learning problem.

Before introducing the specific graph learning techniques, we first provide the formal definition of a graph and graph signals. Consider a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ with the vertex set $\mathcal{V}$ of cardinality n and edge set $\mathcal{E}$, its adjacency matrix $A \in \mathbb{R}^{n \times n}$ governs its topological structure where $A_{i,j} > 0$ indicates there is an edge connecting vertex i and j and $A_{i,j}$ is the edge weight. Given an adjacency matrix A, we can further obtain the graph Laplacian matrix $L = D - A$ where $D_{i,i} = \sum_j A_{i,j}$ is the degree matrix whose off-diagonal entries are all zero.

A graph signal is defined as a function that assigns a scalar value to each vertex of a graph. We can further define multi-channel signals $X \in \mathbb{R}^{n \times d}$ on a graph that assigns a d dimensional vector to each vertex, and each column of the feature matrix

X can be considered as a graph signal. Let $X_i \in \mathbb{R}^d$ denote the graph signal defined on the i-th vertex.

Fitness. Early works (Wang and Zhang, 2007; Daitch et al, 2009) on graph learning utilized the neighborhood information of each data point for graph construction by assuming that each data point can be optimally reconstructed using a linear combination of its neighbors. Wang and Zhang (2007) proposed to learn a graph with normalized degrees by minimizing the following objective,

$$\sum_i ||X_i - \sum_j A_{i,j} X_j||^2 \tag{14.1}$$

where $\sum_j A_{i,j} = 1$, $A_{i,j} \geq 0$.

Similarly, Daitch et al (2009) proposed to minimize a measure of fitness that computes a weighted sum of the squared distance from each vertex to the weighted average of its neighbors, formulated as follows:

$$\sum_i ||D_{i,i} X_i - \sum_j A_{i,j} X_j||^2 = ||LX||_F^2 \tag{14.2}$$

where $||M||_F = (\sum_{i,j} M_{i,j}^2)^{1/2}$ is the Frobenius norm.

Smoothness. Smoothness is another widely adopted assumption on natural graph signals. Given a set of graph signals $X \in \mathbb{R}^{n \times d}$ defined on an undirected weighted graph with an adjacency matrix $A \in \mathbb{R}^{n \times n}$, the smoothness of the graph signals is usually measured by the Dirichlet energy (Belkin and Niyogi, 2002),

$$\Omega(A,X) = \frac{1}{2} \sum_{i,j} A_{i,j} ||X_i - X_j||^2 = \mathrm{tr}(X^\top L X) \tag{14.3}$$

where L is the Laplacian matrix and $\mathrm{tr}(\cdot)$ denotes the trace of a matrix. Lake and Tenenbaum (2010); Kalofolias (2016) proposed to learn a graph by minimizing $\Omega(A,X)$ which forces neighboring vertices to have similar features, thus enforcing graph signals to change smoothly on the learned graph. Notably, solely minimizing the above smoothness loss can lead to the trivial solution $A = 0$.

Connectivity and Sparsity. In order to avoid the trivial solution caused by solely minimizing the smoothness loss, Kalofolias (2016) imposed additional constraints on the learned graph,

$$-\alpha \vec{1}^\top \log(A\vec{1}) + \beta ||A||_F^2 \tag{14.4}$$

where the first term penalizes the formation of disconnected graphs via the logarithmic barrier, and the second term controls sparsity by penalizing large degrees due to the first term. Note that $\vec{1}$ denotes the all-ones vector. As a result, this improves the overall connectivity of the graph, without compromising sparsity.

Similarly, Dong et al (2016) proposed to solve the following optimization problem:

$$\begin{aligned} \min_{L\in\mathbb{R}^{n\times n}, Y\in\mathbb{R}^{n\times p}} & ||X-Y||_F^2 + \alpha \operatorname{tr}(Y^\top LY) + \beta ||L||_F^2 \\ \text{s.t.} \quad & \operatorname{tr}(L) = n, \\ & L_{i,j} = L_{j,i} \leq 0, \quad i \neq j, \\ & L \cdot \vec{1} = \vec{0} \end{aligned} \tag{14.5}$$

which is equivalent to finding jointly the graph Laplacian L and Y (i.e., a "noiseless" version of the zero-mean observation X), such that Y is close to X, and in the meantime Y is smooth on the sparse graph. Note that the first constraint acts as a normalization factor and permits to avoid trivial solutions, and the second and third constraints guarantee that the learned L is a valid Laplacian matrix that is positive semidefinite.

Ying et al (2020a) aimed to learn a sparse graph under Laplacian constrained Gaussian graphical model, and proposed a nonconvex penalized maximum likelihood method by solving a sequence of weighted l1-norm regularized sub-problems. Maretic et al (2017) proposed to learn a sparse graph signal model by alternating between a signal sparse coding and a graph update step.

In order to reduce the computational complexity of solving the optimization problem, many approximation techniques (Daitch et al, 2009; Kalofolias and Perraudin, 2019; Berger et al, 2020) have been explored. Dong et al (2019) provided a good literature review on learning graphs from data from a GSP perspective.

14.2.1.2 Spectral Clustering via Graph Structure Learning

Graph structure learning has also been studied in the field of clustering analysis. For example, in order to improve the robustness of spectral clustering methods for noisy input data, Bojchevski et al (2017) assumed that the observed graph A can be decomposed into the corrupted graph A^c and the good (i.e., clean) graph A^g, and it is beneficial to only perform the spectral clustering on the clean graph. They hence proposed to jointly perform the spectral clustering and the decomposition of the observed graph, and adopted a highly efficient block coordinate-descent (alternating) optimization scheme to approximate the objective function. Huang et al (2019b) proposed a multi-view learning model which simultaneously conducts multi-view clustering and learns similarity relationships between data points in kernel spaces.

14.2.2 Supervised Graph Structure Learning

The task of supervised graph structure learning aims to learn a graph structure from data in a supervised manner. They may or may not consider a particular downstream prediction task during the model training phase.

14.2.2.1 Relational Inference for Interacting Systems

Relational inference for interacting systems aims to study how objects in complex systems interact. Early works considered a fixed or fully-connected interaction graph (Battaglia et al, 2016; van Steenkiste et al, 2018) while modeling the interaction dynamics among objects. Sukhbaatar et al (2016) proposed a neural model to learn continuous communication among a dynamically changing set of agents where the communication graph changes over time as agents move, enter and exit the environment. Recent efforts (Kipf et al, 2018; Li et al, 2020a) have been made to simultaneously infer the latent interaction graph and model the interaction dynamics. Kipf et al (2018) proposed a variational autoencoder (VAE) (Kingma and Welling, 2014) based approach which learns to infer the interaction graph structure and model the interaction dynamics among physical objects simultaneously from their observed trajectories in an unsupervised manner. The discrete latent code of VAE represents edge connections of the latent interaction graph, and both the encoder and decoder take the form of a GNN to model the interaction dynamics among objects. Because the latent distribution of VAE is discrete, the authors adopted a continuous relaxation in order to use the reparameterization trick (Kingma et al, 2014). While Kipf et al (2018) focused on inferring a static interaction graph, Li et al (2020a) designed a dynamic mechanism to evolve the latent interaction graph adaptively over time. A Gated Recurrent Unit (GRU) (Cho et al, 2014a) was applied to capture the history information and adjust the prior interaction graph.

14.2.2.2 Structure Learning in Bayesian Networks

A Bayesian network (BN) is a Probabilistic Graphical Model (PGM) which encodes conditional dependencies between random variables via a directed acyclic graph (DAG), where each random variable is represented as a node in DAG. The problem of learning the BN structure is important yet challenging in Bayesian networks research. Most existing works on BN learning focus on score-based learning of DAGs, and aim to find a DAG with the maximal score where a score indicates how well any candidate DAG is supported by the observed data (and any prior knowledge). Early works treat BN learning as a combinatorial optimization problem which is NP-hard due to the intractable search space of DAGs scaling superexponentially with the number of nodes. Some efficient methods have been proposed for exact BN learning via dynamic programming (Koivisto and Sood, 2004; Silander and Myllymäki, 2006) or integer programming (Jaakkola et al, 2010; Cussens, 2011). Recently, Zheng et al (2018b) proposed to formulate the traditional combinatorial optimization problem into a purely continuous optimization problem over real matrices with a smooth equality constraint ensuring acyclicity of the graph. The resulting problem can hence be efficiently solved by standard numerical algorithms. A follow-up work (Yu et al, 2019a) leveraged the expressive power of GNNs, and proposed a variational autoencoder (VAE) based deep generative model with a vari-

ant of the structural constraint to learn the DAG. The VAE was parameterized by a GNN that can naturally handle both discrete and vector-valued random variables.

14.3 Graph Structure Learning for Graph Neural Networks

Graph structure learning has recently been revisited in the field of GNNs so as to handle the scenarios where the graph-structured data is noisy or unavailable. Recent attempts in this line of research mainly focus on joint learning of graph structures and representations without resorting to human effort or domain expertise. fig. 14.1 shows the overview of graph structure learning for GNNs. Besides, we see several important problems being actively studied (including graph generation, graph adversarial defenses and transformer models) in recent years which are closely related to graph structure learning for GNNs. We will discuss their connections and differences in this section.

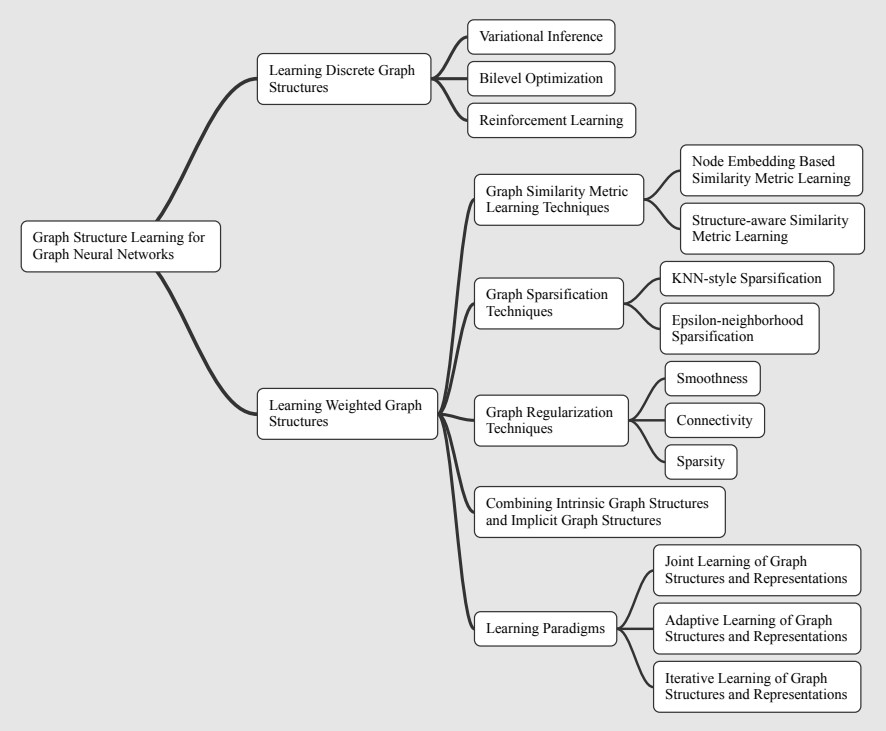

Fig. 14.1: The overview of graph structure learning for GNNs.

14.3.1 Joint Graph Structure and Representation Learning

In recent practice of GNNs, joint graph structure and representation learning has drawn a growing attention. This line of research aims to jointly optimize the graph structure and GNN parameters toward the downstream prediction task in an end-to-end manner, and can be roughly categorized into two groups: learning discrete graph structures and learning weighted adjacency matrices. The first kind of approaches (Chen et al, 2018e; Ma et al, 2019b; Zhang et al, 2019d; Elinas et al, 2020; Pal et al, 2020; Stanic et al, 2021; Franceschi et al, 2019; Kazi et al, 2020) operate by sampling a discrete graph structure (i.e., corresponding to a binary adjacency matrix) from the learned probabilistic adjacency matrix, and then feeding the graph to a subsequent GNN in order to obtain the task prediction. Because the sampling operation breaks the differentiability of the whole learning system, techniques such as variational inference (Hoffman et al, 2013) or Reinforcement Learning (Williams, 1992) are applied to optimize the learning system. Considering that discrete graph structure learning often has the optimization difficulty introduced by the non-differentiable sampling operation and it is hence difficult to learn weights on edges, the other kind of approaches (Chen et al, 2020m; Li et al, 2018c; Chen et al, 2020o; Huang et al, 2020a; Liu et al, 2019b, 2021; Norcliffe-Brown et al, 2018) focuses on learning the weighted (and usually sparse) adjacency matrix associated to a weighted graph which will be later consumed by a subsequent GNN for the prediction task. We will discuss these two types of approaches in great detail next. Before discussing different techniques for joint graph structure and representation learning, let's first formulate the joint graph structure and representation learning problem.

14.3.1.1 Problem Formulation

Let the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be represented as a set of n nodes $v_i \in \mathcal{V}$ with an initial node feature matrix $X \in \mathbb{R}^{d \times n}$, and a set of m edges $(v_i, v_j) \in \mathcal{E}$ (binary or weighted) formulating an initial noisy adjacency matrix $A^{(0)} \in \mathbb{R}^{n \times n}$. Given a noisy graph input $\mathcal{G} := \{A^{(0)}, X\}$ or only a node feature matrix $X \in \mathbb{R}^{d \times n}$, the joint graph structure and representation learning problem we consider aims to produce an optimized graph $\mathcal{G}^* := \{A^{(*)}, X\}$ and its corresponding node embeddings $Z = f(\mathcal{G}^*, \theta) \in \mathbb{R}^{h \times n}$, with respect to certain downstream prediction task. Here, we denote f as a GNN and θ as its model parameters.

14.3.1.2 Learning Discrete Graph Structures

In order to deal with the issue of uncertainty on graphs, many of the existing works on learning discrete graph structures regard the graph structure as a random variable where a discrete graph structure can be sampled from certain probabilistic adjacency matrix. They usually leverage various techniques such as variational infer-

ence (Chen et al, 2018e; Ma et al, 2019b; Zhang et al, 2019d; Elinas et al, 2020; Pal et al, 2020; Stanic et al, 2021), bilevel optimization (Franceschi et al, 2019), and Reinforcement Learning (Kazi et al, 2020) to jointly optimize the graph structure and GNN parameters. Notably, they are often limited to the transductive learning setting where the node features and graph structure are fully observed during both the training and inference stages. In this section, we introduce some representative works on this topic and show how they approach the problem from different perspectives.

Franceschi et al (2019) proposed to jointly learn a discrete probability distribution on the edges of the graph and the parameters of GNNs by treating the task as a bilevel optimization problem Colson et al (2007), formulated as,

$$\begin{aligned} &\min_{\vec{\theta}\in\overline{\mathscr{H}}_N} \mathbb{E}_{A\sim \mathrm{Ber}(\vec{\theta})}[F(w_\theta, A)] \\ &\textit{such that } w_\theta = \mathrm{argmin}_w \, \mathbb{E}_{A\sim \mathrm{Ber}(\vec{\theta})}[L(w, A)] \end{aligned} \tag{14.6}$$

where $\overline{\mathscr{H}}_N$ denotes the convex hull of the set of all adjacency matrices for N nodes, and $L(w,A)$ and $F(w_\theta, A)$ are both task-specific loss functions measuring the difference between GNN predictions and ground-truth labels which are computed on a training set and validation set, respectively. Each edge (i.e., node pair) of the graph is independently modeled as a Bernoulli random variable, and an adjacency matrix $A \sim \mathrm{Ber}(\vec{\theta})$ can thus be sampled from the graph structure distribution parameterized by $\vec{\theta}$. The outer objective (i.e., the first objective) aims to find an optimal discrete graph structure given a GCN and the inner objective (i.e., the second objective) aims to find the optimal parameters w_θ of a GCN given a graph. The authors approximately solved the above challenging bilevel problem with hypergradient descent.

Considering that real-word graphs are often noisy, Ma et al (2019b) viewed the node features, graph structure and node labels as random variables, and modeled the joint distribution of them with a flexible generative model for the graph-based semi-supervised learning problem. Inspired by random graph models from the network science field (Newman, 2010), they assumed that the graph is generated based on node features and labels, and thus factored the joint distribution as the following:

$$p(X,Y,G) = p_{\vec{\theta}}(G|X,Y)p_{\vec{\theta}}(Y|X)p(X) \tag{14.7}$$

where X, Y and G are random variables corresponding to the node features, labels and graph structure, and $\vec{\theta}$ are learnable model parameters. Note that the conditional probabilities $p_{\vec{\theta}(G|X,Y)}$ and $p_{\vec{\theta}(Y|X)}$ can be any flexible parametric families of distributions as long as they are differentiable almost everywhere w.r.t. $\vec{\theta}$. In the paper, $p_{\vec{\theta}(G|X,Y)}$ is instantiated with either latent space model (LSM) (Hoff et al, 2002) or stochastic block models (SBM) (Holland et al, 1983). During the inference stage, in order to infer the missing node labels denoted as Y_{miss}, the authors leveraged the recent advances in scalable variational inference (Kingma and Welling, 2014; Kingma et al, 2014) to approximate the posterior distribution $p_{\vec{\theta}(Y_{\mathrm{miss}}|X,Y_{\mathrm{obs}},G)}$ via a recognition model $q_{\vec{\phi}(Y_{\mathrm{miss}}|X,Y_{\mathrm{obs}},G)}$ parameterized by $\vec{\phi}$ where Y_{obs} denotes the

observed node labels. In the paper, $q_{\vec{\phi}(Y_{\text{miss}}|X,Y_{\text{obs}},G)}$ is instantiated with a GNN. The model parameters $\vec{\theta}$ and $\vec{\phi}$ are jointly optimized by maximizing the Evidence Lower Bound (Bishop, 2006) of the observed data (Y_{obs}, G) conditioned on X.

Elinas et al (2020) aimed to maximize the posterior over the binary adjacency matrix given the observed data (i.e., node features X and observed node labels Y^o), formulated as,

$$p(A|X,Y^o) \propto p_{\vec{\theta}(Y^o|X,A)p(A)} \tag{14.8}$$

where $p_{\vec{\theta}(Y^o|X,A)}$ is a conditional likelihood which can be further factorized following the conditional independence assumption,

$$\begin{aligned} p_{\vec{\theta}(Y^o|X,A)} &= \prod_{y_i \in Y^o} p_{\vec{\theta}(y_i|X,A)} \\ p_{\vec{\theta}(y_i|X,A)} &= \text{Cat}(y_i|\vec{\pi}_i) \end{aligned} \tag{14.9}$$

where $\text{Cat}(y_i|\vec{\pi}_i)$ denotes a categorical distribution, and is the i-th row of a probability matrix $\Pi \in \mathbb{R}^{N\times C}$ modeled by a GCN, namely, $\Pi = \text{GCN}(X,A,\vec{\theta})$. As for the prior distribution over the graph $p(A)$, the authors considered the following form,

$$\begin{aligned} p(A) &= \prod_{i,j} p(A_{i,j}) \\ p(A_{i,j}) &= \text{Bern}(A_{i,j}|\rho^o_{i,j}) \end{aligned} \tag{14.10}$$

where $\text{Bern}(A_{i,j}|\rho^o_{i,j})$ is a Bernoulli distribution over the adjacency matrix $A_{i,j}$ with parameter $\rho^o_{i,j}$. In the paper, $\rho^o_{i,j} = \rho_1 \overline{A}_{i,j} + \rho_2(1-\overline{A}_{i,j})$ was constructed to encode the degree of belief on the absence and presence of observed links with hyperparameters $0 < \rho_1, \rho_2 < 0$. Note that $\overline{A}_{i,j}$ is the observed graph structure which can potentially be perturbed. If there is no input graph available, a KNN graph can be employed. Given the above formulations, the authors developed a stochastic variational inference algorithm by leveraging the reparameterization trick (Kingma et al, 2014) and Concrete distributions techniques (Maddison et al, 2017; Jang et al, 2017) to optimize the graph posterior $p(A|X,Y^o)$ and the GCN parameters $\vec{\theta}$ jointly.

Kazi et al (2020) designed a probabilistic graph generator whose underlying probability distribution is computed based on pair-wise node similarity, formulated as,

$$p_{i,j} = e^{-t||X_i - X_j||} \tag{14.11}$$

where t is a temperature parameter, and X_i is the node embedding of node v_i. Given the above edge probability distribution, they adopted the Gumbel-Top-k trick (Kool et al, 2019) to sample an unweighted KNN graph which would be fed into a GNN-based prediction network. Note that the sampling operation breaks the differentiability of the model, the authors thus exploited Reinforcement Learning to reward edges involved in a correct classification and penalize edges which led to misclassification.

14.3.1.3 Learning Weighted Graph Structures

Unlike the kind of graph structure learning approaches focusing on learning a discrete graph structure (i.e., binary adjacency matrix) for the GNN, there is a class of approaches instead focusing on learning a weighted graph structure (i.e., weighted adjacency matrix). In comparison with learning a discrete graph structure, learning a weighted graph structure has several advantages. Firstly, optimizing a weighted adjacency matrix is much more tractable than optimizing a binary adjacency matrix because the former can be easily achieved by SGD techniques (Bottou, 1998) or even convex optimization techniques (Boyd et al, 2004) while the later often has to resort to more challenging techniques such as variational inference (Hoffman et al, 2013), Reinforcement Learning (Williams, 1992) and combinatorial optimization techniques (Korte et al, 2011) due to its non-differentiability. Secondly, a weighted adjacency matrix is able to encode richer information on edges compared to a binary adjacency matrix, which could benefit the subsequent graph representation learning. For example, the widely used Graph Attention Network (GAT) (Veličković et al, 2018) essentially aims to learn edge weights for the input binary adjacency matrix which benefit the subsequent message passing operations. In this subsection, we will first introduce some common graph similarity metric learning techniques as well as graph sparsification techniques widely used in existing works for learning a sparse weighted graph by considering pair-wise node similarity in the embedding space. Some representative graph regularization techniques will be later introduced for controlling the quality of the learned graph structure. We will then discuss the importance of combining both of the intrinsic graph structures and learned implicit graph structures for better learning performance. Finally, we will cover some important learning paradigms for the joint learning of graph structures and graph representations that have been successfully adopted by existing works.

Graph Similarity Metric Learning Techniques

As introduced in section 14.2.1.1, prior works on unsupervised graph structure learning from smooth signals also aim to learn a weighted adjacency matrix from data. Nevertheless, they are incapable of handling inductive learning setting where there are unseen graphs or nodes in the inference phase. This is because they often learn by directly optimizing an adjacency matrix based on certain prior constraints on the graph properties. Many works on discrete graph structure learning (section 14.3.1.2) have trouble conducting inductive learning as well on account of the similar reason.

Inspired by the success of attention-based techniques (Vaswani et al, 2017; Veličković et al, 2018) for modeling relationships among objects, many recent works in the literature cast graph structure learning as similarity metric learning defined upon the node embedding space assuming that the node attributes more or less contain useful information for inferring the implicit topological structure of the graph. One biggest advantage of this strategy is that the learned similarity metric

function can be later applied to an unseen set of node embeddings to infer a graph structure, thus enabling inductive graph structure learning.

For data deployed in non-Euclidean domains such as graph data, the Euclidean distance is not necessarily the optimal metric for measuring node similarity. Common options for metric learning include cosine similarity (Nguyen and Bai, 2010), radial basis function (RBF) kernel (Yeung and Chang, 2007) and attention mechanisms (Bahdanau et al, 2015; Vaswani et al, 2017). In general, according to the types of raw information sources needed, we group the similarity metric learning functions into two categories: *Node Embedding Based Similarity Metric Learning* and *Structure-aware Similarity Metric Learning*. Next, we will introduce some representative metric learning functions from both categories which have been successfully adopted in prior works on graph structure learning for GNNs.

Node Embedding Based Similarity Metric Learning

Node embedding based similarity metric learning functions are designed to learn a pair-wise node similarity matrix based on node embeddings which ideally encode important semantic meanings of the nodes for graph structure learning.

Attention-based Similarity Metric Functions Most similarity metric functions proposed so far are based on the attention mechanism Bahdanau et al (2015); Vaswani et al (2017). Norcliffe-Brown et al (2018) adopted a simple metric function which computes the dot product between any pair of node embeddings (eq. (14.12)). Given its limited learning capacity, it might have difficulty learning an optimal graph structure.

$$S_{i,j} = \vec{v}_i^\top \vec{v}_j \tag{14.12}$$

where $S \in \mathbb{R}^{n \times n}$ is a node similarity matrix, and $\vec{v}_i$ is the vector representation of node v_i.

To enrich the learning capacity of dot product, Chen et al (2020n) proposed a modified dot product by introducing learnable parameters, formulated as follows:

$$S_{i,j} = (\vec{v}_i \odot \vec{u})^\top \vec{v}_j \tag{14.13}$$

where $\odot$ denotes element-wise multiplication, and $\vec{u}$ is a non-negative trainable weight vector which learns to highlight different dimensions of the node embeddings. Note that the output similarity matrix S is asymmetric.

Chen et al (2020o) proposed a more expressive version of dot product by introducing a weight matrix, formulated as follows:

$$S_{i,j} = \text{ReLU}(W\vec{v}_i)^\top \text{ReLU}(W\vec{v}_j) \tag{14.14}$$

where W is a $d \times d$ weight matrix, and $\text{ReLU}(x) = \max(0,x)$ is a rectified linear unit (ReLU) (Nair and Hinton, 2010) which is used here to enforce the sparsity of the output similarity matrix.

Similar to (Chen et al, 2020o), On et al (2020) introduced a learnable mapping function to node embeddings before computing the dot product, and applied a ReLU

function to enforce sparsity, formulated as follows:

$$S_{i,j} = \text{ReLU}(f(\vec{v}_i)^\top f(\vec{v}_j)) \tag{14.15}$$

where $f : \mathbb{R} \to \mathbb{R}$ is a single-layer feed-forward network without non-linear activation.

Besides using ReLU to enforce sparsity, Yang et al (2018c) applied the square operation to stabilize training, and the row-normalization operation to obtain a normalized similarity matrix, formulated as follows:

$$S_{i,j} = \frac{(\text{ReLU}((W_1\vec{v}_i)^\top W_2\vec{v}_j + b)^2}{\sum_k (\text{ReLU}((W_1\vec{v}_k)^\top W_2\vec{v}_j + b)^2} \tag{14.16}$$

where W_1 and W_2 are $d \times d$ weight matrices, and b is a scalar parameter.

Unlike Chen et al (2020o) that applied the same linear transformation to node embeddings, Huang et al (2020a) applied different linear transformations to the two node embeddings when computing the pair-wise node similarity, formulated as follows:

$$S_{i,j} = \text{softmax}((W_1\vec{v}_i)^\top W_2\vec{v}_j) \tag{14.17}$$

where W_1 and W_2 are $d \times d$ weight matrices, and $\text{softmax}(\vec{z})_i = \frac{e^{z_i}}{\sum_j e^{z_j}}$ is applied to obtain a row-normalized similarity matrix.

Velickovic et al (2020) aimed at graph structure learning in a temporal setting where the implicit graph structure to be learned changes over time. At each time step t, they first computed the pair-wise node similarity $a_{i,j}^{(t)}$ using the same attention mechanism as in (Huang et al, 2020a), and based on that, they further obtained an "aggregated" adjacency matrix $S_{i,j}^{(t)}$ by deriving a new edge for node i by choosing node j with the maximal $\vec{a}_{ij}$. The whole process is formulated as follows:

$$\begin{aligned} a_{i,j}^{(t)} &= \text{softmax}((W_1\vec{v}_i^{(t)})^\top W_2\vec{v}_j^{(t)}) \\ \widetilde{S}_{i,j}^{(t)} &= \mu_i^{(t)}\widetilde{S}_{i,j}^{(t-1)} + (1-\mu_i^{(t)})\mathbb{I}_{j=\text{argmax}_k(a_{i,k}^{(t)})} \\ S_{i,j}^{(t)} &= \widetilde{S}_{i,j}^{(t)} \vee \widetilde{S}_{j,i}^{(t)} \end{aligned} \tag{14.18}$$

where $\mu_i^{(t)}$ is a learnable binary gating mask, $\vee$ denotes logical disjunction between the two operands to enforce symmetry, and W_1 and W_2 are $d \times d$ weight matrices. Because the argmax operation makes the whole learning system non-differentiable, the authors provided the ground-truth graph structures for supervision at each time step.

Cosine-based Similarity Metric Functions Chen et al (2020m) proposed a multi-head weighted cosine similarity function which aims at capturing pair-wise node similarity from multiple perspectives, formulated as follows:

$$
\begin{aligned}
S_{i,j}^p &= \cos(\vec{w}_p \odot \vec{v}_i, \vec{w}_p \odot \vec{v}_j) \\
S_{i,j} &= \frac{1}{m} \sum_{p=1}^{m} S_{ij}^p
\end{aligned}
\tag{14.19}
$$

where $\vec{w}_p$ is a learnable weight vector associated to the p-th perspective, and has the same dimension as the node embeddings. Intuitively, $S_{i,j}^p$ computes the pair-wise cosine similarity for the p-th perspective where each perspective considers one part of the semantics captured in the embeddings. Moreover, as observed in (Vaswani et al, 2017; Veličković et al, 2018), employing multi-head learners is able to stabilize the learning process and increase the learning capacity.

Kernel-based Similarity Metric Functions Besides attention-based and cosine-based similarity metric functions, researchers also explored to apply kernel-based metric functions for graph structure learning. Li et al (2018c) applied a Gaussian kernel to the distance between any pair of node embeddings, formulated as follows:

$$
\begin{aligned}
d(\vec{v}_i, \vec{v}_j) &= \sqrt{(\vec{v}_i - \vec{v}_j)^\top M (\vec{v}_i - \vec{v}_j)} \\
S(\vec{v}_i, \vec{v}_j) &= \frac{-d(\vec{v}_i, \vec{v}_j)}{2\sigma^2}
\end{aligned}
\tag{14.20}
$$

where σ is a scalar hyperparameter which determines the width of the Gaussian kernel, and $d(\vec{v}_i, \vec{v}_j)$ computes the Mahalanobis distance between the two node embeddings $\vec{v}_i$ and $\vec{v}_j$. Notably, M is the covariance matrix of the node embeddings distribution if we assume all the node embeddings of the graph are drawn from the same distribution. If we set $M = I$, the Mahalanobis distance reduces to the Euclidean distance. To make M a symmetric and positive semi-definite matrix, the authors let $M = WW^\top$ where W is a $d \times d$ learnable weight matrix. We can also regard W as the transform basis to the space where we measure the Euclidean distance between two vectors.

Similarly, Henaff et al (2015) first computed the Euclidean distance between any pair of node embeddings, and then applied a Gaussian Kernel or a self-tuning diffusion kernel (Zelnik-Manor and Perona, 2004), formulated as follows:

$$
\begin{aligned}
d(\vec{v}_i, \vec{v}_j) &= \sqrt{(\vec{v}_i - \vec{v}_j)^\top (\vec{v}_i - \vec{v}_j)} \\
S(\vec{v}_i, \vec{v}_j) &= \frac{-d(\vec{v}_i, \vec{v}_j)}{\sigma^2} \\
S_{\text{local}}(\vec{v}_i, \vec{v}_j) &= \frac{-d(\vec{v}_i, \vec{v}_j)}{\sigma_i \sigma_j}
\end{aligned}
\tag{14.21}
$$

where $S_{\text{local}}(\vec{v}_i, \vec{v}_j)$ defines a self-tuning diffusion kernel whose variance is locally adapted around each node. Specifically, σ_i is computed as the distance $d(\vec{v}_i, \vec{v}_{i_k})$ corresponding to the k-th nearest neighbor i_k of node i.

Structure-aware Similarity Metric Learning

When learning implicit graph structures from data, it might be beneficial to utilize the intrinsic graph structures as well if they are available.

Utilizing Intrinsic Edge Embeddings for Similarity Metric Learning Inspired by recent works on structure-aware transformers (Zhu et al, 2019b; Cai and Lam, 2020) which brought the intrinsic graph structure to the self-attention mechanism in the transformer architecture, some works designed structure-aware similarity metric functions which additionally consider the edge embeddings of the intrinsic graph. Liu et al (2019b) introduced a structure-aware attention mechanism as the following:

$$S^l_{i,j} = \text{softmax}(\vec{u}^\top \tanh(W[\vec{h}^l_i, \vec{h}^l_j, \vec{v}_i, \vec{v}_j, \vec{e}_{i,j}])) \tag{14.22}$$

where $\vec{v}_i$ denotes the node attributes for node i, $\vec{e}_{i,j}$ represents the edge attributes between node i and j, $\vec{h}^l_i$ is the vector representation of node i in the l-th GNN layer, and $\vec{u}$ and W are trainable weight vector and weight matrix, respecitively.

Similarly, Liu et al (2021) proposed a structure-aware global attention mechanism for learning pair-wise node similarity, formulated as follows,

$$S_{i,j} = \frac{\text{ReLU}(W^Q \vec{v}_i)^\top (\text{ReLU}(W^K \vec{v}_i) + \text{ReLU}(W^R \vec{e}_{i,j}))}{\sqrt{d}} \tag{14.23}$$

where $\vec{e}_{i,j} \in \mathbb{R}^{d_e}$ is the embedding of the edge connecting node i and j, $W^Q, W^K \in \mathbb{R}^{d \times d_v}$, $W^R \in \mathbb{R}^{d \times d_e}$ are learnable weight matrices, and d, d_v and d_e are the dimensions of hidden vectors, node embeddings and edge embeddings, respectively.

Utilizing Intrinsic Edge Connectivity Information for Similarity Metric Learning In the case where only the edge connectivity information is available in the intrinsic graph, Jiang et al (2019b) proposed a masked attention mechanism for graph structure learning, formulated as follows,

$$S_{i,j} = \frac{A_{i,j} \exp(\text{ReLU}(\vec{u}^\top |\vec{v}_i - \vec{v}_j|))}{\sum_k A_{i,k} \exp(\text{ReLU}(\vec{u}^\top |\vec{v}_i - \vec{v}_k|))} \tag{14.24}$$

where $A_{i,j}$ is the adjacency matrix of the intrinsic graph and $\vec{u}$ is a weight vector with the same dimension as node embeddings $\vec{v}_i$. This idea of using masked attention to incorporate the initial graph topology shares the same spirit with the GAT (Veličković et al, 2018) model.

Graph Sparsification Techniques

The aforementioned similarity metric learning functions all return a weighted adjacency matrix associated to a fully-connected graph. A fully-connected graph is not only computationally expensive but also might introduce noise such as unimportant edges. In real-word applications, most graph structures are much more

sparse. Therefore, it can be beneficial to explicitly enforce sparsity to the learned graph structure. Besides applying the ReLU function in the similarity metric functions (Chen et al, 2020o; On et al, 2020; Yang et al, 2018c; Liu et al, 2021; Jiang et al, 2019b), various graph sparsification techniques have been adopted to enhance the sparsity of the learned graph structure.

Norcliffe-Brown et al (2018); Klicpera et al (2019c); Chen et al (2020o,n); Yu et al (2021a) adopted a KNN style sparsification operation to obtain a sparse adjacency matrix from the node similarity matrix computed by the similarity metric learning function, formulated as follows:

$$A_{i,:} = \text{topk}(S_{i,:}) \tag{14.25}$$

where topk is a KNN-style operation. Specifically, for each node, only the K nearest neighbors (including itself) and the associated similarity scores are kept, and the remaining similarity scores are masked off.

Klicpera et al (2019c); Chen et al (2020m) enforced a sparse adjacency matrix by considering only the ε-neighborhood for each node, formulated as follows:

$$A_{i,j} = \begin{cases} S_{i,j} & S_{i,j} > \varepsilon \\ 0 & \text{otherwise} \end{cases} \tag{14.26}$$

where those elements in S which are smaller than a non-negative threshold ε are all masked off (i.e., set to zero).

Graph Regularization Techniques

As discussed earlier, many works in the field of Graph Signal Processing typically learn the graph structure from data by directly optimizing the adjacency matrix to minimize the constraints defined based on certain graph properties, without considering any downstream tasks. On the contrary, many works on graph structure learning for GNNs aim to optimize a similarity metric learning function (for learning graph structures) toward the downstream prediction task. However, they do not explicitly enforce the learned graph structure to have some common properties (e.g., smoothness) presented in real-word graphs.

Chen et al (2020m) proposed to optimize the graph structures by minimizing a hybrid loss function combining both the task prediction loss and the graph regularization loss. They explored three types of graph regularization losses which pose constrains on the smoothness, connectivity and sparsity of the learned graph.

Smoothness The smoothness property assumes neighboring nodes to have similar features.

$$\Omega(A,X) = \frac{1}{2n^2}\sum_{i,j} A_{i,j}||X_i - X_j||^2 = \frac{1}{n^2}\text{tr}(X^\top LX) \tag{14.27}$$

where $\text{tr}(\cdot)$ denotes the trace of a matrix, $L = D - A$ is the graph Laplacian, and $D_{i,i} = \sum_j A_{i,j}$ is the degree matrix. As can be seen, minimizing $\Omega(A,X)$ forces adjacent

nodes to have similar features, thus enforcing smoothness of the graph signals on the graph associated with A. However, solely minimizing the smoothness loss will result in the trivial solution $A = 0$. We might also want to pose other constraints to the graph.

Connectivity The following equation penalizes the formation of disconnected graphs via the logarithmic barrier.

$$\frac{-1}{n}\vec{1}^{\top}\log(A\vec{1}) \tag{14.28}$$

where n is the number of nodes.

Sparsity The following equation controls sparsity by penalizing large degrees.

$$\frac{1}{n^2}||A||_F^2 \tag{14.29}$$

where $||\cdot||_F$ denotes the Frobenius norm of a matrix.

In practice, solely minimizing one type of graph regularization losses might not be desirable. For instance, solely minimizing the smoothness loss will result in the trivial solution $A = 0$. Therefore, it could be beneficial to balance the trade-off among different types of desired graph properties by computing a linear combination of the various graph regularization losses, formulated as follows:

$$\frac{\alpha}{n^2}\text{tr}(X^{\top}LX) + \frac{-\beta}{n}\vec{1}^{\top}\log(A\vec{1}) + \frac{\gamma}{n^2}||A||_F^2 \tag{14.30}$$

where α, β and γ are all non-negative hyperparameters for controlling the smoothness, connectivity and sparsity of the learned graph.

Besides the above graph regularization techniques, other prior assumptions such as neighboring nodes tend to share the same label (Yang et al, 2019c) and learned implicit adjacency matrix should be close to the intrinsic adjacency matrix (Jiang et al, 2019b) have been adopted in the literature.

Combining Intrinsic Graph Structures and Implicit Graph Structures

Recall that one of the most important motivations for graph structure learning is that the intrinsic graph structure (if it is available) might be error-prone (e.g., noisy or incomplete) and sub-optimal for the downstream prediction task. However, the intrinsic graph typically still carries rich and useful information regarding the optimal graph structure for the downstream task. Hence, it could be harmful to totally discard the intrinsic graph structure.

A few recent works (Li et al, 2018c; Chen et al, 2020m; Liu et al, 2021) proposed to combine the learned implicit graph structure with the intrinsic graph structure for better downstream prediction performance. The rationales are as follows. First of all, they assume that the optimized graph structure is potentially a "shift" (e.g., sub-

structures) from the intrinsic graph structure, and the similarity metric function is intended to learn such a "shift" which is supplementary to the intrinsic graph structure. Secondly, incorporating the intrinsic graph structure can help accelerate the training process and increase the training stability considering there is no prior knowledge on the similarity metric, the trainable parameters are randomly initialized, and thus it may take long to converge.

Different ways for combining intrinsic and implicit graph structures have been proposed. For instance, Li et al (2018c); Chen et al (2020m) proposed to compute a linear combination of the normalized graph Laplacian of the intrinsic graph structure and the normalized adjacency matrix of the implicit graph structure, formulated as follows:

$$\widetilde{A} = \lambda L^{(0)} + (1-\lambda) f(A) \tag{14.31}$$

where $L^{(0)}$ is the normalized graph Laplacian matrix, $f(A)$ is the normalized adjacency matrix associated to the learned implicit graph structure, and λ is a hyperparameter controlling the trade-off between the intrinsic and implicit graph structures. Note that $f : \mathbb{R}^{n\times n} \to \mathbb{R}^{n\times n}$ can be arbitrary normalization operations such as graph Laplacian operation and row-normalization operation. Liu et al (2021) proposed a hybrid message passing mechanism for GNNs which fuses the two aggregated node vectors from the intrinsic graph and the learned implicit graph, respectively, and then feed the fused vector to a GRU (Cho et al, 2014a) to update node embeddings.

Learning Paradigms

Most existing methods for graph structure learning for GNNs consist of two key learning components: graph structure learning (i.e., similarity metric learning) and graph representation learning (i.e., GNN module), and the ultimate goal is to learn the optimized graph structures and representations with respect to certain downstream prediction task. How to optimize the two separate learning components toward the same ultimate goal becomes an important question?

Joint Learning of Graph Structures and Representations

The most straightforward strategy is to jointly optimize the whole learning system in an end-to-end manner toward the downstream prediction task which provides certain form of supervision, as illustrated in fig. 14.2. Jiang et al (2019b); Yang et al (2019c); Chen et al (2020m) designed a hybrid loss function combining both the task prediction loss and the graph regularization loss, namely, $\mathscr{L} = \mathscr{L}_{\text{pred}} + \mathscr{L}_{g}$. The aim of introducing the graph regularization loss is to bring some prior knowledge to the graph properties (e.g., smoothness, sparsity) as we discussed above so as to enforce learning more meaningful graph structures and alleviate the potential overfitting issue.

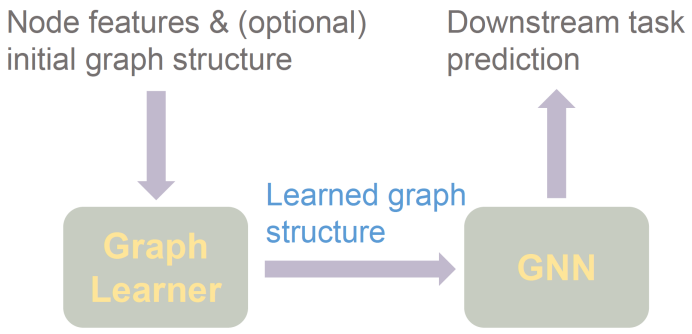

Fig. 14.2: Joint learning paradigm.

Adaptive Learning of Graph Structures and Representations

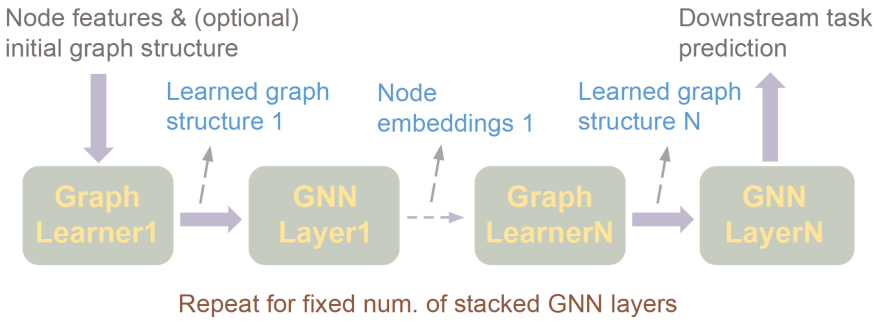

Fig. 14.3: Adaptive learning paradigm.

It is common practice to sequentially stack multiple GNN layers so as to capture long-range dependencies in a graph. As a result, the graph representations updated by one GNN layer will be consumed by the next GNN layer as the initial graph representations. Since input graph representations to each GNN layer are transformed by the previous GNN layer, one may naturally think whether the input graph structure to each GNN layer should be adaptively adjusted to reflect the changes of the graph representations, as illustrated in fig. 14.3. One such example is the GAT (Veličković et al, 2018) model which adatptively reweights the importance of neighboring node embeddings by applying the self-attention mechanism to the previously updated node embeddings when performing neighborhood aggregation at each GAT layer. However, the GAT model does not update the connectivity information of the intrinsic graph. In the literature of graph structure learning for GNNs, some methods (Yang et al, 2018c; Liu et al, 2019b; Huang et al, 2020a; Saire and Ramírez Rivera, 2019) also operate by adaptively learning a graph structure for every GNN layer based on the updated graph representations produced by

the previous GNN layer. And the whole learning system is usually jointly optimized in an end-to-end manner toward the downstream prediction task.

Iterative Learning of Graph Structures and Representations

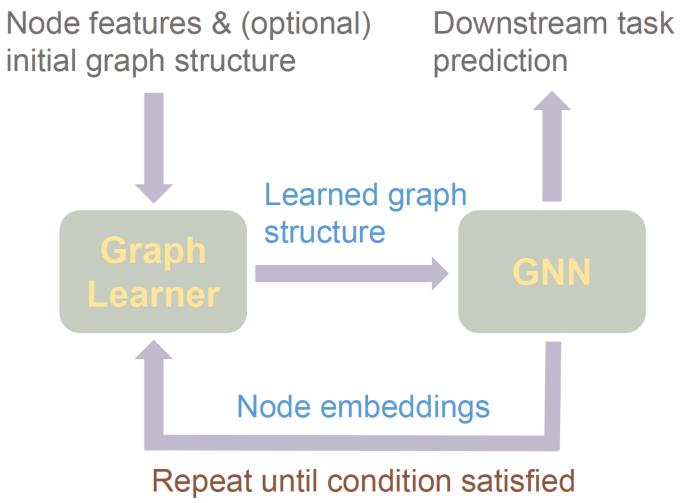

Fig. 14.4: Iterative learning paradigm.

Both of aforementioned joint learning and adaptive learning paradigms aim to learn a graph structure by applying a similarity metric function to the graph representations in a one-shot effort. Even though the adaptive learning paradigm aims to learn a graph structure at each GNN layer based on the updated graph representations, the graph structure learning procedure at each GNN layer is still one-shot. One big limitation of such a one-shot graph structure learning paradigm is that the quality of the learned graph structure heavily relies on the quality of the graph representations. Most existing methods assume that raw node features capture a good amount of information about the graph topology, which unfortunately is not always the case. Thus, it can be challenging to learn good implicit graph structures from the raw node features which do not contain adequate information about the graph topology.

Chen et al (2020m) proposed a novel end-to-end graph learning framework, dubbed as IDGL, for jointly and iteratively learning graph structures and representations. As illustrated in fig. 14.4, the IDGL framework operates by learning a better graph structure based on better graph representations, and in the meantime, learning better graph representations based on a better graph structure in an iterative manner. More specifically, the IDGL framework iteratively searches for an implicit graph structure that augments the intrinsic graph structure (if not available, a KNN graph is used) which is optimized for the downstream prediction task. And this iterative

learning procedure dynamically stops when the learned graph structure approaches close enough to the optimized graph (with respect to the downstream task) according to certain stopping criterion (i.e., the difference between learned adjacency matrices at consecutive iterations are smaller than certain threshold). At each iteration, a hybrid loss combining both the task prediction loss and the graph regularization loss is added to the overall loss. After all iterations, the overall loss is back-propagated through all previous iterations to update model parameters.

This iterative learning paradigm for repeatedly refining the graph structure and graph representations has a few advantages. On the one hand, even when the raw node features do not contain adequate information for learning implicit relationships among nodes, the node embeddings learned by the graph representation learning component could ideally provide useful information for learning a better graph structure because these node embeddings are optimized toward the downstream task. On the other hand, the newly learned graph structure could be a better graph input for the graph representation learning component to learn better node embeddings.

14.3.2 Connections to Other Problems

Graph structure learning for GNNs has interesting connections to a few important problems. Thinking about these connections might spur further research in those areas.

14.3.2.1 Graph Structure Learning as Graph Generation

The task of graph generation focuses on generating realistic and meaningful graphs. The early works of graph generation formalized the problem as a stochastic generation process, and proposed various random graph models for generating a preselected family of graphs such as ER graphs (Erdős and Rényi, 1959), small-world networks (Watts and Strogatz, 1998), and scale-free graphs (Albert and Barabási, 2002). However, these approaches typically make certain simplified and carefully-designed apriori assumptions on graph properties, and thus in general have limited modeling capacity on complex graph structures. Recent attempts focus on building deep generative models for graphs by leveraging RNN You et al (2018b), VAE (Jin et al, 2018a), GAN (Wang et al, 2018a), flow-based techniques (Shi et al, 2019a) and other specially designed models (You et al, 2018a). And GNNs are usually adopted by these models as a powerful graph encoder.

Even though the graph generation task and the graph structure learning task both focus on learning graphs from data, they have essentially different goals and methodologies. Firstly, the graph generation task aims to generate new graphs where both nodes and edges are added to together construct a meaningful graph. However the graph structure learning task aims to learn a graph structure given a set of node

attributes. Secondly, generative models for graphs typically operate by learning the distribution from the observed set of graphs, and generating more realistic graphs by sampling from the learned graph distribution. But graph structure learning methods typically operate by learning the pair-wise relationships among the given set of nodes, and based on that, building the graph topology. It will be an interesting research direction to study how the two tasks can help each other.

14.3.2.2 Graph Structure Learning for Graph Adversarial Defenses

Recent studies (Dai et al, 2018a; Zügner et al, 2018) have shown that GNNs are vulnerable to carefully-crafted perturbations (a.k.a adversarial attacks), e.g., small deliberate perturbations in graph structures and node/edge attributes. Researchers working on building robust GNNs found graph structure learning a powerful tool against topology attacks. Given an initial graph whose topology might become unreliable because of adversarial attacks, they leveraged graph structure learning techniques to recover the intrinsic graph topology from the poisoned graph.

For instance, assuming that adversarial attacks are likely to violate some intrinsic graph properties (e.g., low-rank and sparsity), Jin et al (2020e) proposed to jointly learn the GNN model and the "clean" graph structure from the perturbed graph by optimizing some hybrid loss combining both the task prediction loss and the graph regularization loss. In order to restore the structure of the perturbed graph, Zhang and Zitnik (2020) designed a message-passing scheme that can detect fake edges, block them and then attend to true, unperturbed edges. In order to address the noise brought by the task-irrelevant information on real-life large graphs, Zheng et al (2020b) introduced a supervised graph sparsification technique to remove potentially task-irrelevant edges from input graphs. Chen et al (2020d) proposed a Label-Aware GCN (LAGCN) framework which can refine the graph structure (i.e., filtering distracting neighbors and adding valuable neighbors for each node) before the training of GCN.

There are many connections between graph adversarial defenses and graph structure learning. On the one hand, graph structure learning is partially motivated by improving potentially error-prone (e.g., noisy or incomplete) input graphs for GNNs, which share the similar spirit with graph adversarial defenses. On the other hand, the task of graph adversarial defenses can benefit from graph structure learning techniques as evidenced by some recent works.

However, there is a key difference between their problem settings. The graph adversarial defenses task deals with the setting where the initial graph structure is available, but potentially poisoned by adversarial attacks. And the graph structure learning task aims to handle both the scenarios where the input graph structure is available or unavailable. Even when the input graph structure is available, one can still improve it by "denoising" the graph structure or augmenting the graph structure with an implicit graph structure which captures implicit relationships among nodes.

14.3.2.3 Understanding Transformers from a Graph Learning Perspective

Transformer models (Vaswani et al, 2017) have been widely used as a powerful alternative to Recurrent Neural Networks, especially in the Natural Language Processing field. Recent studies (Choi et al, 2020) have shown the close connection between transformer models and GNNs. By nature, transformer models aim to learn a self-attention matrix between every pair of objects, which can be thought as an adjacency matrix associated with a fully-connected graph containing each object as a node. Therefore, one can claim that transformer models also perform some sort of joint graph structure and representation learning, even though these models typically do not consider any initial graph topology and do not control the quality of the learned fully-connected graph. Recently, many variants of the so-called graph transformers (Zhu et al, 2019b; Yao et al, 2020; Koncel-Kedziorski et al, 2019; Wang et al, 2020k; Cai and Lam, 2020) have been developed to combine the benefits of both GNNs and transformers.

14.4 Future Directions

In this section, we will introduce some advanced topics of graph structure learning for GNNs and highlight some promising future directions.

14.4.1 Robust Graph Structure Learning

Although one of the major motivations of developing graph structure learning techniques for GNNs is to handle noisy or incomplete input graphs, robustness does not lie in the heart of most existing graph structure learning techniques. Most of existing works did not evaluate the robustness of their approaches to noisy initial graphs. Recent works showed that random edge addition or deletion attacks significantly downgraded the downstream task performance (Franceschi et al, 2019; Chen et al, 2020m). Moreover, most existing works admit that the initial graph structure (if provided) might be noisy and thus unreliable for graph representation learning, but they still assume that node features are reliable for graph structure learning, which is often not true in real-world scenarios. Therefore, it is challenging yet rewarding to explore robust graph structure learning techniques for data with noisy initial graph structures and noisy node attributes.

14.4.2 Scalable Graph Structure Learning

Most existing graph structure learning techniques need to model the pair-wise relationships among all the nodes in order to discover the hidden graph structure. Therefore, their time complexity is at least $O(n^2)$ where n is the number of graph nodes. This can be very expensive and even intractable for large-scale graphs (e.g., social networks) in real word. Recently, Chen et al (2020m) proposed a scalable graph structure learning approach by leveraging the anchor-based approximation technique to avoid explicitly computing the pair-wise node similarity, and achieved linear complexity in both computational time and memory consumption with respect to the number of graph nodes. In order to improve the scalability of transformer models, different kinds of approximation techniques have also been developed in recent works (Tsai et al, 2019; Katharopoulos et al, 2020; Choromanski et al, 2021; Peng et al, 2021; Shen et al, 2021; Wang et al, 2020g). Considering the close connections between graph structure learning for GNNs and transformers, we believe there are many opportunities in building scalable graph structure learning techniques for GNNs.

14.4.3 Graph Structure Learning for Heterogeneous Graphs

Most existing graph structure learning works focus on learning homogeneous graph structures from data. In comparison with homogeneous graphs, heterogeneous graphs are able to carry on richer information on node types and edge types, and occur frequently in real-world graph-related applications. Graph structure learning for heterogeneous graphs is supposed to be more challenging because more types of information (e.g., node types, edge types) are expected to be learned from data. Some recent attempts (Yun et al, 2019; Zhao et al, 2021) have been made to learn graph structures from heterogeneous graphs.

14.5 Summary

In this chapter, we explored and discussed graph structure learning from multiple perspectives. We first reviewed the existing works on graph structure learning in the literature of traditional machine learning, including both unsupervised graph structure learning and supervised graph structure learning. As for unsupervised graph structure learning, we mainly looked into some representative techniques developed from the Graph Signal Processing community. We also introduced some recent works on clustering analysis that leveraged graph structure learning techniques. As for supervised graph structure learning, we introduced how this problem was studied in the research on modeling interacting systems and Bayesian Networks. The main focus of this chapter is on introducing recent advances in graph structure learning

for GNNs. We motivated graph structure learning in the GNN field by discussing the scenarios where the graph-structured data is noisy or unavailable. We then moved on to introduce recent research progress in joint graph structure and representation learning, including learning discrete graph structures and learning weighted graph structures. The connections and differences between graph structure learning and other important problems such as graph generation, graph adversarial defenses and transformer models were also discussed. We then highlighted several remaining challenges and future directions in the research of graph structure learning for GNNs.

Editor's Notes: Graph Structure Learning is a fast-emerging research topic and have seen a significant number of interests in recent years. The key idea is to learn an optimized graph structure in order to generate a better node representation (Chapter 4) and a more robust node representation (Chapter 8). Obviously, the graph structure learning could be expensive if the common pair-wise learning approach is adopted and thus the scalability issue could be a real major concern (Chapter 6). Meanwhile, it has tight connection with graph generation (Chapter 11) and self-supervised learning (Chapter 18), since they all consider partially how to modify/leverage graph structure. This chapter can be applicable to a broad range of application domains such as recommendation system (Chapter 19), computer vision (Chapter 20), Natural Language Processing (Chapter 21), Program Analysis (Chapter 22), and so on.