Graph Structure Estimation Neural Networks

Ruijia Wang*
Beijing University of Posts and Telecommunications
Beijing, China
wangruijia@bupt.edu.cn

Shuai Mou
Tencent TEG
Shenzhen, China
harrymou@tencent.com

Xiao Wang*
Beijing University of Posts and Telecommunications
Beijing, China
xiaowang@bupt.edu.cn

Wanpeng Xiao, Qi Ju†
wanpengxiao@tencent.com
damonju@tencent.com
Tencent TEG
Shenzhen, China

Chuan Shi†
Beijing University of Posts and Telecommunications
Beijing, China
shichuan@bupt.edu.cn

Xing Xie
Microsoft Research
Beijing, China
xingx@microsoft.com

ABSTRACT

Graph Neural Networks (GNNs) have drawn considerable attention in recent years and achieved outstanding performance in many tasks. Most empirical studies of GNNs assume that the observed graph represents a complete and accurate picture of node relationship. However, this fundamental assumption cannot always be satisfied, since the real-world graphs from complex systems are error-prone and may not be compatible with the properties of GNNs. Therefore, GNNs solely relying on original graph may cause unsatisfactory results, one typical example of which is that GNNs perform well on graphs with homophily while fail on the disassortative situation. In this paper, we propose graph estimation neural networks GEN, which estimates graph structure for GNNs. Specifically, our GEN presents a structure model to fit the mechanism of GNNs by generating graphs with community structure, and an observation model that injects multifaceted observations into calculating the posterior distribution of graphs and is the first to incorporate multi-order neighborhood information. With above two models, the estimation of graph is implemented based on Bayesian inference to maximize the posterior probability, which attains mutual optimization with GNN parameters in an iterative framework. To comprehensively evaluate the performance of GEN, we perform a set of experiments on several benchmark datasets with different homophily and a synthetic dataset, where the experimental results demonstrate the effectiveness of our GEN and rationality of the estimated graph.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; • **Theory of computation** → **Social networks**.

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

WWW '21, April 19–23, 2021, Ljubljana, Slovenia

ACM ISBN 978-1-4503-8312-7/21/04.
https://doi.org/10.1145/3442381.3449952

KEYWORDS

Graph Neural Networks, Graph Structure Learning, Network Representation Learning

ACM Reference Format:
Ruijia Wang, Shuai Mou, Xiao Wang, Wanpeng Xiao, Qi Ju, Chuan Shi, and Xing Xie. 2021. Graph Structure Estimation Neural Networks. In *Proceedings of the Web Conference 2021 (WWW '21), April 19–23, 2021, Ljubljana, Slovenia.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3442381.3449952

1 INTRODUCTION

Graphs are ubiquitous across application domains ranging from chemo- and bioinformatics to image and social network analysis. With their prevalence, it is particularly important to learn effective representations of graphs and apply them to downstream tasks [13]. Recently, there has been a surge of interest in Graph Neural Networks (GNNs) [14, 21, 25, 41] for representation learning of graphs, which broadly follow a recursive message passing scheme [11] where local neighborhood information is aggregated and passed on to the neighbors. These GNNs have achieved state-of-the-art performance in many analytical tasks such as node classification [47, 48] and recommender systems [44, 49].

Although existing GNNs have been successfully applied in a wide variety of scenarios, they rely on one fundamental assumption that the observed topology is ground-truth information and consistent with the properties of GNNs. But in fact, as graphs are usually extracted from complex interaction systems, such assumption could always be violated. One reason is that these interaction systems usually contain uncertainty or error [27]. For instance, in protein interaction graphs, traditional laboratory experimental error is a primary source of inaccuracy. The another reason is that the issue of missing data is inevitable. As another instance, the graph of Internet is determined by examining either router tables or collections of traceroute paths, both of which give only subsets of the edges. It has been revealed that unreliable error-prone graphs could significantly limit the representation capability of GNNs [10, 37, 51], one typical example of which is that the performance of GNNs can greatly degrade on the disassortative graphs [31] where homophily (*i.e.*, nodes within the same community tend to connect with each other) does not hold. In short, missing, meaningless or even spurious edges are prevalent in real graphs, which results in inconsistency with

the properties of GNNs and casts doubts about the accuracy or correctness of their results. Therefore, it is imperative to explore an optimal graph for GNNs.

Nevertheless, it is technically challenging to effectively learn an optimal graph structure for GNNs. Particularly, two obstacles need to be addressed. (1) The graph generation mechanism should be taken into consideration. It is well established in network science literature [34] that the graph generation is potentially governed by some underlying principles, *e.g.*, the configuration model [33]. Considering these principles fundamentally drives the learned graph to maintain a regular global structure and be more robust to noise in real observations. Unfortunately, the majority of current methods parameterize each edge locally [5, 10, 18] and do not account for the underlying generation of graph, so that the resulted graph has a lower tolerance for noise and sparsity. (2) multifaceted information should be injected to reduce bias. Learning graph structure from one information source inevitably leads to bias and uncertainty. It makes sense that the confidence of an edge would be greater if this edge exists under multiple measurements. Thus, a reliable graph structure ought to make allowance for comprehensive information, although it is complicated to obtain multi-view measurements and depict their relationship with GNNs. Existing approaches [19, 51] mainly utilize the feature similarity, making the learned graph more susceptible to the bias of single view.

To address the aforementioned issues, in this paper, we propose **G**raph structure **E**stimation neural **N**etworks (GEN) to improve the node classification performance through estimating an appropriate graph structure for GNNs. We firstly analyze the properties of GNNs to match proper graph generation mechanism. GNNs, as low-pass filters [1, 23, 45] which smooth neighborhood to make the representations of proximal nodes similar, are suitable to graphs with community structure [12]. Therefore, we attach a structure model to the graph generation, hypothesizing that the estimated graph is drawn from Stochastic Block Model (SBM) [17]. Furthermore, in addition to the observed graph and node feature, we creatively inject multi-order neighborhood information to circumvent bias and present an observation model to jointly treat above multi-view information as observations of the optimal graph. In order to estimate the optimal graph, we construct observations during GNN training, then apply Bayesian inference based on structure and observation models to infer the entire posterior distribution over graph structure. Finally, the estimated graph and the parameters of GNNs achieve mutual, positive reinforcement through elaborately iterative optimization.

In summary, the contributions of this paper are three-fold:

- Concerning graph structure learning for GNNs, we are the first to simultaneously consider the generation of learned graph to fit the mechanism of GNNs, and comprehensively employ the multifaceted information to give a more precise and nuanced picture of the proper graph structure.
- We propose novel graph structure estimation neural networks GEN, which designs a structure model characterizing the underlying graph generation and an observation model injecting multi-order neighborhood information to accurately infer the graph structure based on Bayesian inference.
- We validate the effectiveness of GEN via thorough comparisons with state-of-the-art methods on several challenging benchmarks. Additionally, we also analyze the properties of GEN and verify the rationality of estimated graph on a synthetic dataset.

The rest of the paper is organized as follows. In Section 2, we review some of the related work. In Section 3, we introduce notations and formally explain our proposed GEN. We report experimental results in Section 4 and conclude the work in Section 5.

2 RELATED WORK

In line with the focus of our work, we briefly review the most related work on GNNs and graph structure learning.

2.1 Graph Neural Networks

Over the past few years, graph neural networks have achieved great success in solving machine learning problems on graph-structured data. Most current GNNs can be generally divided into two families, *i.e.*, spectral methods and spatial methods.

Specifically, the first family learns node representation based on graph spectral theory. [2] first proposes a spectral graph-based extension of convolutional networks using the Fourier basis. ChebNet [6] defines graph convolution based on Chebyshev polynomials to remove the computationally expensive Laplacian eigendecomposition. GCN [21] further simplifies ChebNet by using its first-order approximation. In a follow-up work, SGC [45] reduces the graph convolution to a linear model but still achieves competitive performance. The second family of methods directly define graph convolution in the spatial domain as aggregating and transforming local information. GraphSAGE [14] learns aggregators by sampling and aggregating neighbor information. GAT [41] assigns different edge weights based on node features during aggregation. For better efficiency, FastGCN [4] performs importance sampling on each layer to sample a fixed number of nodes and APPNP [22] uses the relationship between GCN and PageRank [36] to derive an improved propagation scheme based on personalized PageRank.

There are many other graph neural models, we please refer the readers to recent surveys [46, 52] for a more comprehensive review. But almost all these GNNs treat the observed graphs, derived from noisy data or modelling assumptions, as ground-truth information, which significantly limits their capability to handle uncertainty in the graph structure.

2.2 Graph Structure Learning

Graph structure learning is not a newly born topic, and there has been a considerable body of previous work in network science [16, 26, 28, 32] dedicated to it. To process raw graph data into more accurate and nuanced estimates of graph quantities, some methods learn graph structure from measurements of the evolutional graphed dynamical systems such as coupled oscillators [43] or spreading processes [24]. There are also work [3, 15, 42] on error correction strategies for missing or extraneous nodes. However, the goal of these work departs from graph representation learning.

As GNNs become the most eye-catching tools for graph representation learning, several efforts have been made to combine graph structure learning and GNNs for boosting performance of downstream tasks. Bayesian GCNN [51] views the observed graph

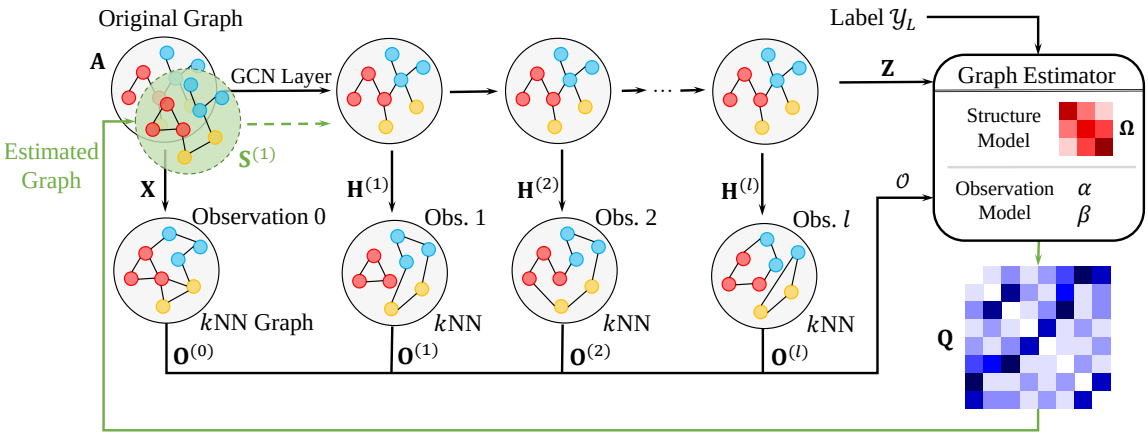


Figure 1: Overall framework of our proposed model GEN, where two rounds of iterative optimization are taken as an example. Note that the probability matrix Ω is the parameter of structure model, while true-positive rate α and false-positive rate β are the parameters of observation model. In the heat maps of matrices Ω and Q, the darker the color, the greater the probability.

as a realization from random graphs and uses its adjacency matrix in conjunction with features to perform joint optimization. LDS [10] presents a new approach for jointly learning the graph and the parameters of GNNs by approximately solving a bilevel programming. Pro-GNN [19] reconstructs the graph by applying low rank, sparsity and feature smoothness regularizers to reduce the negative effects of adversarial structure. Geom-GCN [37] utilizes network geometry to bridge the gap between observed graph and latent continuous space, then redefines local structure to aggregation. However, the graphs learned by these methods are not constrained by generation principles to meet the mechanism of GNNs, and more unreliable since only a part of available information is injected.

3 GEN: THE PROPOSED MODEL

In this section, we elaborate the proposed graph estimation neural networks GEN, a novel graph structure learning framework for GNNs based on Bayesian inference. We first present the problem statement, then begin with the overview of GEN. Subsequently, we zoom into the details of graph estimator with our structure and observation models. Lastly, we illustrate the jointly iterative optimization of graph structure and GNNs parameters.

3.1 Problem Statement

Before we make the problem statement, we first introduce some notations and basic concepts.

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$ be a graph, where $\mathcal{V}$ is the set of N nodes $\{v_1, v_2, \cdots, v_N\}$, $\mathcal{E}$ is the set of edges, $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N] \in \mathbb{R}^{N \times D}$ represents the node feature matrix and $\mathbf{x}_i$ is the feature vector of node v_i. The edges describe the relations between nodes and can be represented by an adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$, where A_{ij} denotes the relation between nodes v_i and v_j. Following the common semi-supervised node classification setting, only a small part of nodes $\mathcal{V}_L = \{v_1, v_2, \cdots, v_l\}$ are associated with corresponding labels $\mathcal{Y}_L = \{y_1, y_2, \cdots, y_l\}$, where y_i is the label of v_i.

Given graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$ and the partial labels $\mathcal{Y}_L$, the goal of graph structure learning for GNNs is to simultaneously learn a optimal adjacency matrix $\mathbf{S} \in \mathcal{S} = [0, 1]^{N \times N}$ and the GNN parameters Θ to improve node classification performance for unlabeled nodes. The objective function can be formulated as

$$\min_{\Theta, \mathbf{S}} \mathcal{L}(\mathbf{A}, \mathbf{X}, \mathcal{Y}_L) = \sum_{v_i \in \mathcal{V}_L} \ell\left(f_\Theta(\mathbf{X}, \mathbf{S})_i, y_i\right), \tag{1}$$

where $f_\Theta : \mathcal{V}_L \to \mathcal{Y}_L$ is the function learned by GNNs, $f_\Theta(\mathbf{X}, \mathbf{S})_i$ is the prediction of node v_i and $\ell(\cdot, \cdot)$ is to measure the difference between prediction and true label, such as cross entropy.

3.2 Overview

Most GNNs process the observed graph as a ground-truth depiction of the relationship between nodes, but often the data we face is imperfect, where spurious edges may be included or other edges with strong relationship may be missing. Even if the observed graph is clean, it is not jointly optimized with GNNs thus may not be compatible with their properties. These defects of the observed graph can cause the performance of GNNs to drop rapidly. Thus, one natural strategy is to optimize graph structure and GNNs simultaneously. However, the existing graph structure learning methods for GNNs do not consider the internal graph generation mechanism and mainly use one-sided feature similarity, resulting in the learned graph vulnerable to noisy and missing data.

In this work, we aim at estimating appropriate graphs for GNNs by constructing a probabilistic method GEN that mainly consists of two modules, *i.e.*, structure and observation models. GEN explicitly constrains the graph generation by the structure model to meet the properties of GNNs and utilizes multifaceted information as observations of the optimal graph structure with observation model where multi-order neighborhood similarity is innovatively injected. The illustration of GEN is shown in Figure 1. Though there exist a number of different GNN methods, we focus on learning graph structure for Graph Convolutional Networks (GCN) [21]. Please

note that it is straightforward to extend the proposed framework to other GNNs [41, 45].

To infer the graph structure for GCN, we have the following two insights. (1) The underlying graph generation that facilitates the node classification performance of GCN tends to maintain homophily among neighbors. (2) The optimal graph might be measured in multiple ways using observed interactions, feature similarity and multi-order neighborhood similarity, which are multifaceted observations that reflect the optimal graph from different views. These observations may be error-prone when viewed separately, but can be ensembled to circumvent bias. Therefore, our GEN firstly utilizes available information to construct an observation set for the optimal graph, then estimates the graph based on these observations with explicitly constraining the underlying structure.

Specifically, we feed the original graph $\mathbf{A}$ and node feature $\mathbf{X}$ into l-layer vanilla GCN. In the light of that node representations in different layers reveal multi-order neighborhood information, we utilize node representations $\mathbf{H}^{(i)}$ of every layer i to construct k-Nearest Neighbor (kNN) graph $\mathbf{O}^{(i)}$ as the observation of optimal graph, as well as the original graph $\mathbf{A}$, to form observation set $\mathcal{O} = \{\mathbf{A}, \mathbf{O}^{(0)}, \mathbf{O}^{(1)}, \cdots, \mathbf{O}^{(l)}\}$. Afterwards, we put these observations $\mathcal{O}$, predictions $\mathbf{Z}$ and labels $\mathcal{Y}_L$ into graph estimator, which is based on our proposed structure and observation models to infer adjacency matrix $\mathbf{Q}$ with underlying community structure, where Q_{ij} denotes the probability that there is an edge between node v_i and node v_j. The entire framework is implemented by iterative optimization between GCN parameters and graph estimation. Firstly, we tune GCN parameters until the cross-entropy loss converges. Secondly, we hold GCN parameters constant to obtain estimated adjacency matrix $\mathbf{Q}$ based on Expectation-Maximization (EM) algorithm, and set a threshold ε on $\mathbf{Q}$ to get estimated graph $\mathbf{S}$. Lastly, we feed the estimated graph $\mathbf{S}$ back to the vanilla GCN to perform the next round of iterative optimization. The better estimated graph would then force GCN to produce more accurate observations, and the process is repeated. During such iterations, the learning of GCN and the inference of graph structure enhance each other.

It is worth noting that the estimated graph is not only generated by the original graph, but also by prior knowledge and multifaceted information. Thus the proposed GEN alleviates the problem of missing or erroneous data from three aspects. (1) Structure model introduces prior knowledge to constrain community structure of the estimated graph, thereby prompting the completion of missing edges inside communities and the deletion of erroneous edges between communities. (2) We construct kNN graphs as observations to infer the optimal graph, and different kNN graphs capture local structures with different orders. Even original graph (first-order structure) may be erroneous, other structures can provide auxiliary information. Based on these multifaceted observations, informative edges may appear multiple times, while erroneous edges are only observed accidentally. (3) With the iterative optimization of GEN, updated node embeddings are learned from the estimated graph. Therefore, the optimization of graph structure and GNN parameters achieve mutual reinforcement, which further resolves the problem.

3.3 Observation Construction

Without loss of generality, we choose representative GCN as backbone. To begin with, we feed the original graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$ into vanilla GCN to construct the initial observation set $\mathcal{O}$ for subsequent graph estimation.

Specifically, GCN follows a neighborhood aggregation strategy, which iteratively updates the presentation of a node by aggregating representations of its neighbors. Formally, the k^{th} layer aggregation rule of GCN is

$$\mathbf{H}^{(k)} = \sigma\left(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{H}^{(k-1)}\mathbf{W}^{(k)}\right). \tag{2}$$

Here, $\tilde{\mathbf{A}}$ is the normalized adjacency matrix and $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$. $\mathbf{W}^{(k)}$ is a layer-wise trainable weight matrix, and σ denotes an activation function. $\mathbf{H}^{(k)} \in \mathbb{R}^{N\times d}$ is the matrix of node representations in the k^{th} layer, and $\mathbf{H}^{(0)} = \mathbf{X}$. In terms of l-layer GCN, the activation function of the last layer l is row-wise softmax and predictions $\mathbf{Z} = \mathbf{H}^{(l)}$. The GCN parameters $\Theta = (\mathbf{W}^{(1)}, \mathbf{W}^{(2)}, \cdots, \mathbf{W}^{(l)})$ can be trained via gradient descent.

The current GCN acts directly on the observed graph $\mathbf{A}$ which is extracted from the real-world complex systems and is usually noisy. To estimate an optimal graph structure for GCN, we need to construct multifaceted observations that could be ensembled to resist bias. Fortunately, after k iterations of aggregation, node representation captures the structural information within its k-order graph neighborhood which provides local to global information. On the other hand, node pairs with similar neighborhood are possibly far away in the graph but apt to the same communities. If these informative node pairs are employed, they could provide useful clues for downstream classification. Therefore, we attempt to connect these distant but similar nodes in our estimated graph to enhance the classification performance of GCN.

Specifically, we fix the GCN parameters Θ and take out the node representations $\mathcal{H} = \{\mathbf{H}^{(0)}, \mathbf{H}^{(1)}, \cdots, \mathbf{H}^{(l)}\}$ to construct kNN graphs $\{\mathbf{O}^{(0)}, \mathbf{O}^{(1)}, \cdots, \mathbf{O}^{(l)}\}$ as observations of the optimal graph, where $\mathbf{O}^{(i)}$ is the adjacency matrix of kNN graph generated by $\mathbf{H}^{(i)}$ and characterizes the similarity of i-order neighborhood. Obviously, the original graph $\mathbf{A}$ is also an important external observation of the optimal graph, thus we combine it with kNN graphs to form the complete observation set $\mathcal{O} = \{\mathbf{A}, \mathbf{O}^{(0)}, \mathbf{O}^{(1)}, \cdots, \mathbf{O}^{(l)}\}$. These observations reflect the optimal graph structure from varied views and can be ensembled to infer a more reliable graph structure.

As preliminary preparations, these observations $\mathcal{O}$, predictions $\mathbf{Z}$ and labels $\mathcal{Y}_L$ will be fed into estimator to accurately infer the posterior distribution of the graph structure. In the following subsection, we will introduce the inference in detail.

3.4 Graph Estimator

Until now, the question we would like to answer is: *given these available observations $\mathcal{O}$, what is the best estimated graph for GCN?* These observations reveal the optimal graph structure from different perspectives, but they may be unreliable or incomplete, and we do not have a priori that how accurate any of the information is. Under this daunting circumstance, it is not straightforward to answer this question directly, but it is relatively easy to answer the reverse question. Imagining that a graph with community structure has been generated, we could calculate the probability of mapping

this graph to these observations. If we can do this, Bayesian inference allows us to invert the calculation and compute the posterior distribution of graph structure, hence our goal is achieved. The procedure is formalized as follows.

3.4.1 Structure Model. Let us denote the optimal graph which we are trying to estimate, as a symmetric adjacency matrix $\mathbf{G}$, and we firstly propose a structure model to represent the underlying structure generation of optimal graph $\mathbf{G}$.

Considering the local smoothing nature of GCN, a good choice is Stochastic Block Model (SBM) which is widely used for community detection and applicable to model a graph that has relatively strong community structure [20, 38]. The values of within- and between-community parameters in the fitted block model can constrain the homophily of the estimate. Although there are also other SBM variants, *e.g.*, degree-corrected SBM [20]. But in this paper, the effectiveness of vanilla SBM as structure model has been verified. We leave more complex structure models as future work and believe it will further improve performance.

Particularly, this process of generating optimal graph $\mathbf{G}$ takes the form of a probability distribution $P(\mathbf{G}|\Omega, \mathbf{Z}, \mathcal{Y}_L)$. Here, Ω represents the parameters of SBM which assumes that the probability of an edge between nodes depends only on their communities. For instance, the probability of an edge between node v_i with community c_i and node v_j with community c_j is $\Omega_{c_i c_j}$. Therefore, Ω indicates the probabilities of within- and between-community connections. Given parameters Ω, predictions $\mathbf{Z}$ and labels $\mathcal{Y}_L$, the probability of generating graph $\mathbf{G}$ is formalized as

$$P(\mathbf{G}|\Omega, \mathbf{Z}, \mathcal{Y}_L) = \prod_{i<j} \Omega_{c_i c_j}^{G_{ij}} (1 - \Omega_{c_i c_j})^{1-G_{ij}}, \tag{3}$$

where

$$c_i = \begin{cases} y_i & \text{if } v_i \in \mathcal{V}_L, \\ z_i & \text{otherwise,} \end{cases} \tag{4}$$

which means that generating an edge between node v_i and v_j in optimal graph $\mathbf{G}$ depends only on the probability $\Omega_{c_i c_j}$ related to community identifications c_i and c_j. Here, in order to obtain more accurate community identifications, we use the label-corrected predictions that replace the community identifications of nodes in training set directly with labels.

3.4.2 Observation Model. Please note that the structure model represents our prior knowledge or constrain about the underlying structure before observing any data. In fact, in what form the optimal graph structure exists is a mystery, and the things that can be done are to combine its external observations to infer it.

Therefore, we introduce an observation model to describe how the optimal graph $\mathbf{G}$ maps onto observations, which assumes that the observations of edges are independent identically distributed Bernoulli random variables, conditional on the presence or absence of an edge in the optimal graph. This assumption is well accepted in literature, *e.g.*, community detection [30] and graph generation [39, 50], which has been proven to be feasible.

$P(\mathcal{O}|\mathbf{G}, \alpha, \beta)$ is the probability of these observations $\mathcal{O}$ given the optimal graph $\mathbf{G}$ and model parameters α and β. Specifically, we parameterize the possible observations by two probabilities: the true-positive rate α, which is the probability of observing an edge where one truly exists in the optimal graph $\mathbf{G}$, and the false-positive rate β, which is the probability of observing an edge where none exists in the optimal graph $\mathbf{G}$. The remaining possibilities of true-negative and false-negative rates are $1-\beta$ and $1-\alpha$, respectively. Let us suppose that out of the M (*i.e.*, $|\mathcal{O}|$) observations, we observe an edge on E_{ij} of them, and no edge on the remaining $M-E_{ij}$. Plugging these definitions, we can write the specific form of $P(\mathcal{O}|\mathbf{G}, \alpha, \beta)$ as

$$P(\mathcal{O}|\mathbf{G}, \alpha, \beta) = \prod_{i<j} \left[\alpha^{E_{ij}}(1-\alpha)^{M-E_{ij}}\right]^{G_{ij}} \times \left[\beta^{E_{ij}}(1-\beta)^{M-E_{ij}}\right]^{1-G_{ij}}. \tag{5}$$

If there is truly an edge in optimal graph $\mathbf{G}$, the probability of observing E_{ij} edges between node v_i and v_j out of total M observations is succinctly written as $\alpha^{E_{ij}}(1-\alpha)^{M-E_{ij}}$. If not, the probability is $\beta^{E_{ij}}(1-\beta)^{M-E_{ij}}$.

3.4.3 Graph Estimation. Having clearly clarified our models for structure and observation, we now present our graph estimation process based on Bayesian inference.

It is difficult to calculate posterior probability $P(\mathbf{G}, \Omega, \alpha, \beta|\mathcal{O}, \mathbf{Z}, \mathcal{Y}_L)$ for optimal graph directly. However, combining our above models and applying Bayes rule, we then have

$$P(\mathbf{G}, \Omega, \alpha, \beta|\mathcal{O}, \mathbf{Z}, \mathcal{Y}_L) = \frac{P(\mathcal{O}|\mathbf{G}, \alpha, \beta) P(\mathbf{G}|\Omega, \mathbf{Z}, \mathcal{Y}_L) P(\Omega) P(\alpha) P(\beta)}{P(\mathcal{O}, \mathbf{Z}, \mathcal{Y}_L)}, \tag{6}$$

where $P(\Omega)$, $P(\alpha)$, $P(\beta)$ and $P(\mathcal{O}, \mathbf{Z}, \mathcal{Y}_L)$ are the probabilities of the parameters and available data, which we assume to be independent.

We get an expression for the posterior probability of the parameters Ω, α and β, by summing all possible values of optimal graph $\mathbf{G}$:

$$P(\Omega, \alpha, \beta|\mathbf{O}, \mathbf{Z}, \mathcal{Y}_L) = \sum_{\mathbf{G}} P(\mathbf{G}, \Omega, \alpha, \beta|\mathbf{O}, \mathbf{Z}, \mathcal{Y}_L). \tag{7}$$

Maximizing this posterior probability *w.r.t.* Ω, α and β will give maximum-a-posteriori (MAP) estimates for these parameters. Based on these MAP estimates, we can calculate the estimated adjacency matrix $\mathbf{Q}$ for optimal graph $\mathbf{G}$

$$Q_{ij} = \sum_{\mathbf{G}} q(\mathbf{G}) G_{ij}, \tag{8}$$

where quantity Q_{ij} is the posterior probability that there is an edge between nodes v_i and v_j, representing our confidence about whether the edge exists.

3.5 Iterative Optimization

Jointly optimizing GCN parameters Θ and estimated adjacency matrix $\mathbf{Q}$ is challenging. And the dependence between them exacerbates the difficulty. In this work, we use an alternating optimization schema to iteratively update Θ and $\mathbf{Q}$.

3.5.1 Update Θ. For semi-supervised node classification, we evaluate the cross-entropy error over all labeled examples $\mathcal{Y}_L$

$$\min_{\Theta} \mathcal{L}(\mathbf{A}, \mathbf{X}, \mathcal{Y}_L) = -\sum_{v_i \in \mathcal{V}_L} \mathbf{y}_i \ln \mathbf{z}_i, \tag{9}$$

which is a typical GCN optimization problem and we can learn parameters Θ via stochastic gradient descent.

3.5.2 Update $\mathbf{Q}$. To update the estimated adjacency matrix $\mathbf{Q}$, we maximize the Eq. (7) with Expectation-Maximization (EM) algorithm [7, 29, 30].

E-step. Since it is convenient to maximize not the probability itself but its logarithm, we apply Jensen's inequality to the log of Eq. (7)

$$\log P(\Omega, \alpha, \beta|\mathbf{O}, \mathbf{Z}, \mathcal{Y}_L) \geq \sum_{\mathbf{G}} q(\mathbf{G}) \log \frac{P(\mathbf{G}, \Omega, \alpha, \beta|\mathbf{O}, \mathbf{Z}, \mathcal{Y}_L)}{q(\mathbf{G})}, \quad (10)$$

where $q(\mathbf{G})$ is any nonnegative function of $\mathbf{G}$ satisfying $\sum_{\mathbf{G}} q(\mathbf{G}) = 1$, which can be seen as a probability distribution over $\mathbf{G}$.

The right-hand side of the inequality Eq. (10) is maximized when the exact equality is achieved

$$q(\mathbf{G}) = \frac{P(\mathbf{G}, \Omega, \alpha, \beta|\mathbf{O}, \mathbf{Z}, \mathcal{Y}_L)}{\sum_{\mathbf{G}} P(\mathbf{G}, \Omega, \alpha, \beta|\mathbf{O}, \mathbf{Z}, \mathcal{Y}_L)}. \quad (11)$$

Substituting Eq. (3) and Eq. (5) into Eq. (11), and eliminating the constants in fraction, we find the following expression for $q(\mathbf{G})$:

$$\begin{aligned} q(\mathbf{G}) &= \frac{\prod_{i<j}\left[\Omega_{c_ic_j}\alpha^{E_{ij}}(1-\alpha)^{M-E_{ij}}\right]^{G_{ij}}\left[(1-\Omega_{c_ic_j})\beta^{E_{ij}}(1-\beta)^{M-E_{ij}}\right]^{1-G_{ij}}}{\sum_{\mathbf{G}}\prod_{i<j}\left[\Omega_{c_ic_j}\alpha^{E_{ij}}(1-\alpha)^{M-E_{ij}}\right]^{G_{ij}}\left[(1-\Omega_{c_ic_j})\beta^{E_{ij}}(1-\beta)^{M-E_{ij}}\right]^{1-G_{ij}}} \\ &= \prod_{i<j}\frac{\left[\Omega_{c_ic_j}\alpha^{E_{ij}}(1-\alpha)^{M-E_{ij}}\right]^{G_{ij}}\left[(1-\Omega_{c_ic_j})\beta^{E_{ij}}(1-\beta)^{M-E_{ij}}\right]^{1-G_{ij}}}{\Omega_{c_ic_j}\alpha^{E_{ij}}(1-\alpha)^{M-E_{ij}} + (1-\Omega_{c_ic_j})\beta^{E_{ij}}(1-\beta)^{M-E_{ij}}}. \end{aligned} \quad (12)$$

Then further maximizing the right-hand side of Eq. (10) will give us the MAP estimates.

M-step. We can find the maximum over the parameters by differentiating. Taking derivatives of the right-hand side of Eq. (10) while holding $q(\mathbf{G})$ constant, and assuming that the priors are uniform, we have

$$\sum_{\mathbf{G}} q(\mathbf{G}) \sum_{i<j}\left[\frac{G_{ij}}{\Omega_{c_ic_j}} - \frac{1-G_{ij}}{1-\Omega_{c_ic_j}}\right] = 0, \quad (13)$$

$$\sum_{\mathbf{G}} q(\mathbf{G}) \sum_{i<j} G_{ij}\left[\frac{E_{ij}}{\alpha} - \frac{M-E_{ij}}{1-\alpha}\right] = 0, \quad (14)$$

$$\sum_{\mathbf{G}} q(\mathbf{G}) \sum_{i<j} (1-G_{ij})\left[\frac{E_{ij}}{\beta} - \frac{M-E_{ij}}{1-\beta}\right] = 0. \quad (15)$$

The solution of these equations gives us MAP estimates for Ω, α and β. Note that Eq. (13) depends only on the SBM and its solution gives the parameter values for structure model. Similarly, Eq. (14) and Eq. (15) depend only on the observation model.

For specific calculations, we swap the order of the summations and find that

$$\Omega_{rs} = \begin{cases} \frac{M_{rs}}{n_r n_s} & \text{if } r \neq s, \\ \frac{2M_{rr}}{n_r(n_r-1)} & \text{otherwise,} \end{cases} \quad (16)$$

where $n_r = \sum_i \delta_{c_i,r}$ and $M_{rs} = \sum_{i<j} Q_{ij}\delta_{c_i,r}\delta_{c_j,s}$. Thus Eq. (16) has the simple interpretation that the probability Ω_{rs} of an edge between community r and s is the average probabilities of the individual edges between all nodes in these two communities. Similar calculations are done for α and β

$$\alpha = \frac{\sum_{i<j} Q_{ij}E_{ij}}{M\sum_{i<j} Q_{ij}}, \quad (17)$$

$$\beta = \frac{\sum_{i<j}(1-Q_{ij})E_{ij}}{M\sum_{i<j}(1-Q_{ij})}. \quad (18)$$

Algorithm 1: Model training for GEN

Input : adjacency matrix $\mathbf{A}$, feature matrix $\mathbf{X}$, labels $\mathcal{Y}_L$, k in kNN, tolerance λ, threshold ε, iterations τ

Output : estimated graph $\mathbf{S}$, GCN parameters Θ

```
Initialize Θ, Ω, α and β;
for i = 1 to τ do
    Update Θ with Eq. (9);
    Construct the observation set O with kNN graph;
    while |α − α^old| > λ or |β − β^old| > λ do
        Ω^old = Ω, α^old = α, β^old = β;
        Calculate Ω, α and β with Eq. (16), (17) and (18);
        Update Q with Eq. (19);
    Extract S^(i) from Q by threshold ε with Eq. (21);
    A = S^(i);
return S^(τ) and Θ;
```

To calculate the value of Q_{ij}, we substitute Eq. (12) into Eq. (8):

$$Q_{ij} = \frac{\Omega_{c_ic_j}\alpha^{E_{ij}}(1-\alpha)^{M-E_{ij}}}{\Omega_{c_ic_j}\alpha^{E_{ij}}(1-\alpha)^{M-E_{ij}} + (1-\Omega_{c_ic_j})\beta^{E_{ij}}(1-\beta)^{M-E_{ij}}}. \quad (19)$$

The posterior distribution $q(\mathbf{G})$ can be conveniently rewritten in terms of Q_{ij} as

$$q(\mathbf{G}) = \prod_{i<j} Q_{ij}^{G_{ij}}\left(1-Q_{ij}\right)^{1-G_{ij}}. \quad (20)$$

To put that another way, the probability distribution over optimal graph is the product of independent Bernoulli distributions of the individual edges, with Bernoulli parameters Q_{ij} which capture both the graph structure itself and the uncertainty in that structure.

This leads to a natural EM algorithm for determining the values of the parameters and posterior distribution over possible graph structures. We perform the E-step by maximizing first over $q(\mathbf{G})$ with the parameters held constant; then go to the M-step over parameters Ω, α and β with $q(\mathbf{G})$ held constant; and repeat these iterations until convergence.

Sparsification. Note that the elements in estimated adjacency matrix $\mathbf{Q}$ range between $[0, 1]$. However, a fully connected graph structure is not only computationally expensive but also makes little sense for most applications. We hence proceed to extract a sparse adjacency matrix $\mathbf{S}$ from $\mathbf{G}$, by masking off those elements smaller than certain non-negative threshold ε:

$$S_{ij} = \begin{cases} Q_{ij} & \text{if } Q_{ij} > \varepsilon, \\ 0 & \text{otherwise.} \end{cases} \quad (21)$$

3.5.3 Training Algorithm. With the aforementioned updating and inference rules, the training algorithm is shown in Algorithm 1. Specifically, we first randomly initialize all the parameters for GEN. Then we update Θ to form observations, and estimate $\mathbf{S}$ based on observations to boost the optimization of Θ in turn. Through such alternative and iterative updates, the more accurate estimated graph $\mathbf{S}$ leads to better optimized parameters Θ, while the better Θ produces more precise observations for estimating $\mathbf{S}$.

Table 1: Description of datasets.

Dataset	Homophily	Nodes	Edges	Classes	Features	Val/Test nodes
Cora	0.83	2, 708	5, 429	7	1, 433	500/1, 000
Citeseer	0.71	3, 327	4, 732	6	3, 703	500/1, 000
Pubmed	0.79	19, 717	44, 338	3	500	500/1, 000
Chameleon	0.25	2, 277	36, 101	5	2, 325	500/1, 000
Squirrel	0.22	5, 201	217, 073	5	2, 089	500/1, 000
Actor	0.24	7, 600	33, 544	5	931	500/1, 000

The complexity of each iteration inside GEN mainly involves the update of Θ, the construction of kNN graphs and the inference of S. We implement an efficient GPU-based two-layer GCN using sparse-dense matrix multiplication. Therefore, the computational complexity for Θ optimization is $O(|\mathcal{E}|Ddc)$, where c represents the number of classes. Regarding kNN graph construction, since both the calculation of all nodes and the construction of different kNN graphs can be paralleled, the complexity is $O(Nd + N \log N)$. It is worth noting that it could be optimized to $O(d \log N)$ using Ball-Tree [35] which we leave for future work. In terms of inference, we implement EM algorithm using array matrix in Numpy library to achieve acceleration, and its complexity is $O(Ni)$, where i denotes the number of steps. In practice, the value of i is always small. In summary, the complexity of τ-iteration GEN is around $O(\tau(|\mathcal{E}|Ddc + N(d + \log N + i)))$, while the complexity of existing graph structure learning methods [10, 19] is $O(\tau N^2)$.

4 EXPERIMENTS

In this section, we evaluate the effectiveness of GEN via extensive experiments. Particularly, we compare GEN with the-state-of-the-art methods on semi-supervised node classification task, and present the change of prediction confidence and iterative optimization process. We further analyze the mechanism and properties of GEN. Lastly, we investigate the hyper-parameter sensitivity.

4.1 Experimental Setup

4.1.1 Datasets. We validate the proposed GEN on six open graph datasets. The statistics of the datasets are summarized in Table 1.

- **Citation networks** [21]. Cora, Citeseer and Pubmed are benchmark citation networks datasets. In these networks, nodes represent papers, and edges denote the citation relationship. Node features are bag-of-words representation of papers, and labels are academic fields.
- **Wikipedia networks** [37]. Chameleon and Squirrel are two page-page networks in Wikipedia with specific topics. In those datasets, nodes are web pages, and edges represent hyper-links. Node features are informative nouns in pages, and labels correspond to the monthly traffic of the pages.
- **Actor co-occurrence network** [37]. This dataset is the actor-only induced subgraph of the fim-director-actor-writer network. Each node represents an actor, and edges denote the collaborations. Node features are keywords in Wikipedia and labels are the types of actors.

Note that according to homophily, these benchmark datasets are divided into assortative graphs (*i.e.*, Cora, Citeseer and Pubmed) and disassortative graphs (*i.e.*, Chameleon, Squirrel and Actor). For challenging disassortative Chameleon, Squirrel and Actor datasets, we transform the supervised setting in original paper [37] into the typical semi-supervised train/val/test split.

4.1.2 Baselines. To evaluate the effectiveness of GEN, we compare it with three categories of representative GNNs, including three spectral-based methods (*i.e.*, SGC [45], GCN [21] and ChebNet [6]), three spatial-based methods (*i.e.*, GAT [41], APPNP [22] and GraphSAGE [14]) and three graph structure learning based methods (*i.e.*, LDS [10], Pro-GNN [19] and Geom-GCN [37]).

4.1.3 Implementation. We implement the proposed GEN with deep learning library PyTorch[1]. All experiments are conducted on a Linux server with GPU (NVIDIA Tesla M40) and CPU (Intel Xeon E5-2680). The Python and PyTorch versions are 3.6.8 and 1.5.0, respectively.

For all datasets, we utilize two-layer GCN as backbone of our model, and train it for 200 epochs using Adam optimizer with an initial learning rate of 0.01 and a weight decay of 5e-4. We set ReLU as the activation function and apply a dropout rate of 0.5 to further prevent overfitting. For the grid search space of hyper-parameters, embedding dimension d is chosen from $\{8, 16, 32, 64, 128\}$, k of kNN is tuned from 3 to 15, tolerance λ is searched in $\{0.1, 0.01, 0.001\}$ and threshold ε is tuned amongst $\{0.1, 0.2, \cdots, 0.9\}$. Note that we fix the optimization iterations τ to 50, and choose the model with highest validation accuracy for test.

We adopt the implementations of SGC, GCN, GAT, APPNP and GraphSAGE from the PyTorch Geometric library [8] in all experiments. For the remaining baselines ChebNet, LDS, Pro-GNN and Geom-GCN, we use the source codes provided by the authors. And we perform a hyper-parameter search for all models on validation set. For fairness, the size of search space for their common hyper-parameters is the same, including embedding dimension, initial learning rate, weight decay and dropout rate. In terms of other hyper-parameters, we follow the settings in their original papers, and carefully tune them to achieve optimal performance.

4.2 Node Classification

4.2.1 Performance Comparison. We evaluate the semi-supervised node classification performance of GEN against state-of-the-art baselines. In addition to the 20 labels per class training explored in previous work, we also evaluate the performance under more

[1] https://pytorch.org/

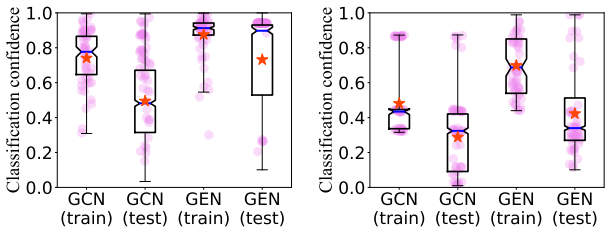
Table 2: Quantitative results (%±σ) on node classification. (L/C: the number of labels per class; bold: best)

Datasets	L/C	SGC	GCN	ChebNet	GAT	APPNP	GraphSAGE	LDS	Pro-GNN	Geom-GCN	GEN
Cora	20	80.9±0.4	81.7±0.8	81.9±0.4	82.3±1.0	83.3±1.0	80.1±0.5	82.5±1.2	80.9±0.9	63.7±0.3	**83.6**±0.4
	10	75.5±0.7	74.6±0.7	72.5±1.0	76.9±0.9	75.6±2.4	72.9±1.1	77.1±3.1	76.9±0.8	47.1±0.6	**77.8**±0.7
	5	72.8±1.0	71.0±0.7	66.6±2.3	75.0±0.7	72.9±2.4	68.4±1.7	75.7±2.9	75.1±0.5	22.5±1.4	**76.2**±1.3
Citeseer	20	71.9±0.8	70.9±0.6	70.0±0.9	72.0±0.9	72.0±0.5	71.8±0.7	72.3±1.1	68.8±0.8	65.6±0.8	**73.8**±0.6
	10	68.6±1.0	66.6±1.0	67.3±1.1	68.4±1.4	70.2±0.8	68.0±1.0	70.4±1.6	69.1±0.6	48.8±0.6	**72.4**±0.5
	5	61.3±1.9	53.5±0.8	51.7±2.3	61.8±1.9	54.2±0.5	55.4±1.0	68.1±0.5	56.6±1.5	28.3±3.6	**70.4**±2.7
Pubmed	20	77.1±2.6	79.4±0.4	78.2±1.0	77.9±0.6	79.6±0.2	73.6±2.2	78.2±1.8	78.0±0.8	77.2±0.3	**80.9**±0.9
	10	71.9±2.3	73.7±0.4	71.5±0.8	71.1±1.4	73.5±0.9	70.6±1.4	74.4±1.5	72.7±0.6	69.3±0.3	**75.6**±1.1
	5	68.7±0.4	73.0±1.4	69.4±1.4	70.2±0.7	73.8±1.1	70.2±1.2	72.8±1.3	70.6±1.7	68.4±0.4	**74.9**±2.0
Chameleon	20	49.1±0.9	49.1±1.1	37.0±0.5	46.4±1.4	46.1±0.7	43.7±2.0	49.4±1.1	50.3±0.6	35.7±0.5	**50.4**±0.9
	10	44.4±1.0	44.2±0.7	32.5±0.8	45.0±2.0	39.4±0.7	41.7±1.9	44.9±1.3	45.5±1.2	31.6±0.4	**45.6**±1.1
	5	39.3±1.2	39.5±0.7	33.2±0.8	39.9±1.8	36.9±1.0	35.9±0.8	40.5±1.5	41.0±1.8	28.5±0.5	**41.4**±2.3
Squirrel	20	34.7±1.5	35.0±0.6	21.2±2.0	27.2±2.9	33.6±1.1	28.3±2.0	30.1±0.4	33.4±2.4	25.4±0.6	**35.5**±1.1
	10	31.8±1.3	33.0±0.4	18.8±1.2	27.1±1.2	31.4±2.2	25.9±2.9	29.4±0.9	32.9±0.4	21.6±0.3	**33.4**±1.1
	5	29.1±0.7	31.3±1.3	18.1±0.7	24.1±2.5	27.0±2.4	24.9±2.9	27.1±1.4	28.2±1.9	22.6±0.3	**32.7**±2.7
Actor	20	22.0±1.3	21.7±1.6	26.7±1.1	23.8±3.6	29.7±0.7	28.9±1.1	27.0±1.4	21.5±1.7	20.7±0.5	**35.3**±0.6
	10	22.0±2.8	20.8±1.0	22.3±1.1	22.7±3.6	28.0±0.8	22.2±2.5	25.7±1.3	22.2±0.7	20.7±0.5	**31.3**±2.2
	5	24.2±2.3	21.8±2.0	21.4±1.0	21.4±2.4	22.4±2.0	23.1±3.6	23.8±0.8	20.9±0.5	24.1±0.4	**30.5**±2.7

severely limited data scenarios where only 10 or 5 labels per class are available. In 5 and 10 labels per class cases, we construct the training set by using the first 5 or 10 labels in the original partition, while keeping the validation and testing sets unchanged. In Table 2, we report the mean and standard deviation results over 10 independent trials with different random seeds.

Based on the results, we make the following observations.

- GEN consistently outperforms other baselines on six datasets, especially under reduced-label and disassortative settings, which demonstrates that our ingeniously designed graph estimation framework can boost node classification performance in a robust manner. We find that as the label rate and homophily decrease, the performance of GNNs drops quickly and the improvement of GEN becomes more pronounced. These phenomena are in line with our expectations that noisy or sparse observed graphs prevent GNNs from aggregating effective information, and our estimated graphs alleviate this issue.
- The overwhelming performance superiority of GEN over backbone GCN implies that GEN is capable of estimating suitable structure, so that the graph structure estimation and the parameter optimization of GCN reinforce each other.
- In comparison with other graph structure learning based methods, our performance improvement illustrates that explicitly constraining the community structure and making full use of multi-faceted information help learn better graph structure and more robust GCN parameters. Note that Geom-GCN does not perform well in most cases. One feasible reason is that it may fit the supervised settings in its original papers, where more supervised information is injected into parameter learning, and may not adapt well to the semi-supervised setting.

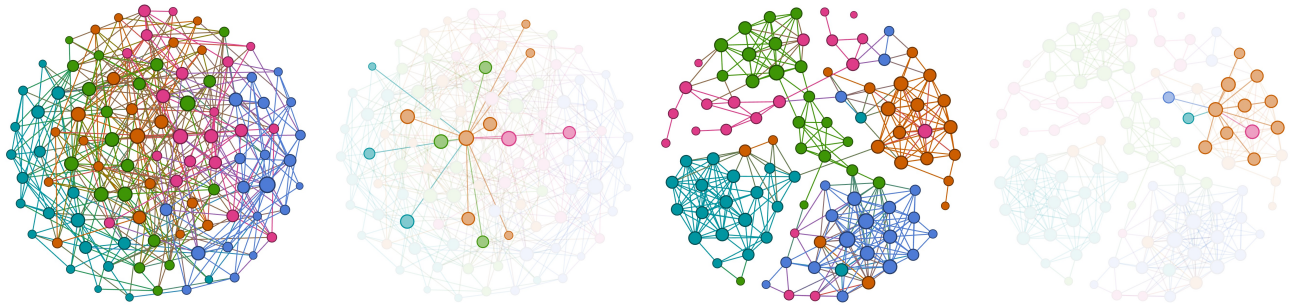


(a) Cora (b) Chameleon

Figure 2: Boxplots of the prediction confidence *w.r.t* nodes in training and test set for GCN and GEN on (a) Cora and (b) Chameleon datasets. The box indicates 25-75 percentiles; the median is shown by a blue horizontal line; the star represents the mean value. The whiskers extend to the smallest and largest data points that are not outliers. To explain more intuitively, for each box we visualize the prediction confidence of 50 nodes with pink circles.

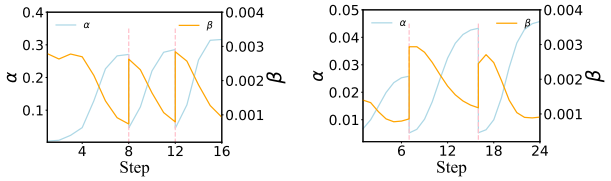
4.2.2 Prediction Confidence. To provide further insight to the performance improvement of GEN over backbone GCN, we analyze the change of prediction confidence. Specifically, for every training and test set node v_i in Cora and Chameleon datasets, we select the values with the true class y_i from the final prediction vectors of GCN and GEN, and plot them as boxes in Figure 2.

From plots, we find that for Cora or more challenging Chameleon datasets, the prediction confidence obtained on the estimated graphs is much higher than original graphs. We conjecture that injecting multi-order neighborhood similarity into graph structure learning makes the learned graph portray more informative node pairs, so



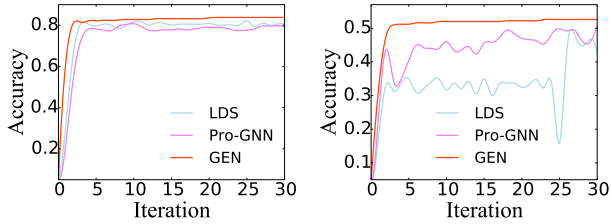
(a) Original graph (b) Neighbors in original graph (c) Estimated graph (d) Neighbors in estimated graph

Figure 3: Visualization of the graph structures for (a) original graph and (c) estimated graph, where color indicates the communities of nodes and the node diameter is proportional to the PageRank score. We fade the remaining nodes and edges to emphasize the neighbors of selected node in (b) original graph and (d) estimated graph.



(a) Cora (b) Chameleon

Figure 4: Curves of parameter change during estimation on (a) Cora and (b) Chameleon datasets. The coordinate of X-axis represents the alternate step of expectation and maximization in EM algorithm, and the pink dashed lines separate the different iterations of GEN.



(a) Cora (b) Chameleon

Figure 5: Curves of validation accuracy *w.r.t.* the number of iterations on (a) Cora and (b) Chameleon datasets.

that the classification distribution becomes more peaked and arm the vanilla GCN with the ability to correct misclassified nodes.

4.2.3 Optimization Analysis. To understand the iterative estimation process of GEN, we present the value change curves of true-positive rate α and false-positive rate β in Eq. (5) on Cora and Chameleon datasets. Particularly, we fix the tolerance λ to 0.001 and show the value change of several successive iterations in Figure 4. We observe that the deduced EM algorithm always meets the convergence condition within 6 steps, illustrating an extremely efficient optimization. In addition, during the iterative process, the true-positive rate α gradually increases, implying that the observations built by backbone GCN become more and more accurate.

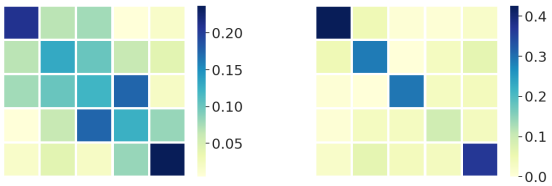
We further compare the convergence speed of GEN and other graph structure learning baselines, as shown in Figure 5. Please note that Geom-GCN is outside the scope of comparison, since it does not follow an iterative framework. It can be observed that GEN has faster convergence speed and better validation accuracy on both datasets, demonstrating the efficiency and effectiveness of our proposed GEN. Moreover, for disassortative Chameleon dataset, the validation accuracy of LDS and Pro-GNN fluctuates greatly while GEN improves accuracy steadily, which once again confirms the robustness of considering the graph generation principles and multifaceted information.

4.3 Analysis of Estimated Graph

In the previous subsection, we have demonstrated the effectiveness of the proposed framework on real-world datasets. In this section, we aim at understanding the mechanism of GEN and the properties of estimated graph.

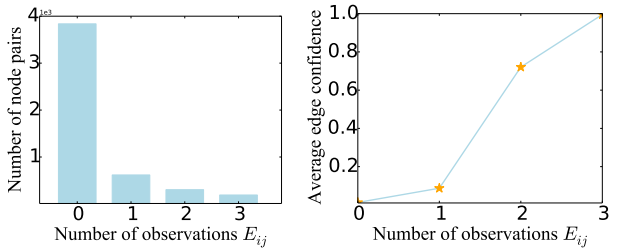
To better explore the mechanism of our GEN, we perform studies on a synthetic graph using an attributed stochastic block model, which has been used extensively to benchmark graph clustering methods [9, 40]. For better visualization and analysis, in our version of SBM there are 5 communities and each with 20 nodes. We randomly initialize the symmetric probability matrix to generate edges, and the diagonal element is the largest in corresponding row under most circumstances to ensure a certain degree of homophily. The 8-dimensional features of nodes are generated using a multivariate normal distribution, where nodes in different communities share the random covariance matrix but have different means according to their own communities. In terms of train/val/test split, we utilize 5 nodes per class for training, 5 nodes for validation and the remaining as test.

4.3.1 Graph Visualization. To examine the graph structure changes brought by GEN intuitively, we visualize the original graph and



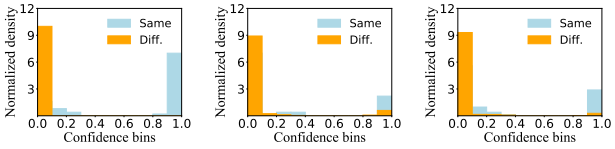
(a) Original graph (b) Estimated graph

Figure 6: Heat maps for the probability matrices of the (a) original graph and (b) estimated graph. Note that the color shades of two maps represent different scales, as shown by bars on the right.



(a) Statistics of observations (b) Relationship between edge confidence and observations

Figure 7: The bar chart (a) shows the number of node pairs with each possible value of E_{ij}, and the polyline (b) represents the average edge confidence for every value of E_{ij}.



(a) Training set (b) Validation set (c) Test set

Figure 8: Normalized histograms of edge confidence for (a) training, (b) validation and (c) test set nodes *w.r.t.* the same/different communities.

estimated graph using Gephi tool in Figure 3 (a) and (c). Meanwhile, we zoom into the local details of above graphs, and select one specific node to highlight the changes of its neighborhood in Figure 3 (b) and (d). As shown in Figure 3, the original graph is a bit chaotic, and there exist lots of between-community connections which can be seen more clearly by the neighborhood of selected node. In this case, the node classification accuracy of GCN is only 60%. After applying our proposed graph estimation framework, the community structure of estimated graph is crisp, where many spurious edges are removed and strong relationships are preserved. And the classification accuracy of GEN increases to 84%.

In order to quantify the transformation of community structure before and after estimation, we calculate the probability matrix between communities, and draw them as heat maps in Figure 6. We observe that in Figure 6 (a), the off-diagonal color blocks are also dark and even darker than the diagonal ones, indicating that the original graph does not maintain a good homophily and high probabilities of between-community connections could bring difficulties to the GCN optimization. But for the probability matrix of estimated graph, GEN widens the gaps between the diagonal and off-diagonal elements, where the former is usually significantly larger than the latter, interpreting the reason for the performance improvement of classification.

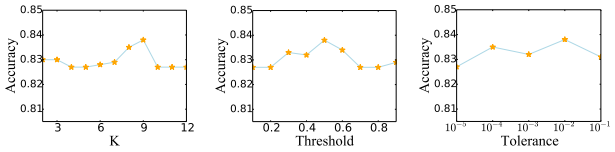
4.3.2 Edge Confidence. Recall that the estimated adjacency matrix $\mathbf{Q}$ in GEN is the posterior probability of graph structure, where Q_{ij} represents our confidence in the existence of that edge. To investigate the meaning of estimated edge, we present the relationship between edge confidence Q_{ij} and number of observations E_{ij}. In terms of the synthetic dataset, there are three observations in total, thus the range of E_{ij} is 0 to 3. For each possible value of E_{ij}, we accumulate the corresponding number of nodes pairs and calculate average edge confidence for these nodes pairs in Figure 7. We find that most node pairs are in the "zero observation" bar, since the graph is sparse and a large majority of node pairs never meet. And from the relationship between the number of observation E_{ij} and average edge confidence, it can be found that an edge observed only zero or one time implies a low Q_{ij} (less than 0.1), so a single observation is probably a false alarm. But there is a relatively sharp transition between $E_{ij} = 1$ and $E_{ij} = 2$, indicating that two or more observations of the same edge result in a much larger Q_{ij} which is a strong inference that the edge exists in the optimal graph.

To show the distribution of edge confidence, we divide the edges in two groups: edges between nodes of the same communities and different communities. And we report the normalized histograms of these optimized edge confidence on training, validation and test set in Figure 8, respectively. We observe that the confidence of edges between same communities is concentrated on the last bin (more than 0.9), while the confidence of edges between different communities is more skewed towards the first bin (less than 0.1). This phenomenon is extremely obvious on the training set, and also prominent on validation and test set, illustrating that GEN captures useful edge confidence, *i.e.*, higher confidence for edges between the same communities.

4.4 Hyper-parameter Sensitivity

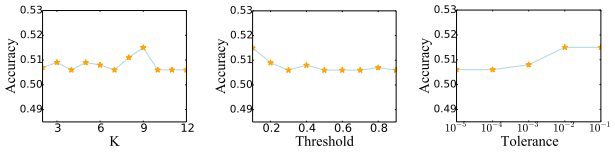
In this subsection, we explore the sensitivity of hyper-parameters: k in kNN, threshold ε in Eq. (21) and tolerance λ in EM algorithm. More specifically, we alter the value of k, ε and λ to see how they affect the performance of our model. We vary k from 2 to 12, ε from 0.1 to 0.9, and λ from 1e-5 to 0.1 in a log scale of base 10. We only report the node classification results on Cora and Chameleon datasets in Figure 9 and 10, since similar observations are made in other datasets.

***k* in *k*NN graph.** The change trend of accuracy *w.r.t.* k is roughly increases first then starts to decrease. It is probably because the sparsity of kNN graphs reduces the number of observed edges



(a) k in kNN (b) Threshold ε (c) Tolerance λ

Figure 9: Impact of hyper-parameters on Cora.



(a) k in kNN (b) Threshold ε (c) Tolerance λ

Figure 10: Impact of hyper-parameters on Chameleon.

which results in information loss and ineffective inference, while too large k would introduce more noise.

Threshold ε. As for threshold ε, if we set ε so high that there are few edges above the threshold in estimated graph, GEN is unable to aggregate enough information, while too low ε makes estimated graph including lots of spurious edges. We note that the accuracy curves of ε are very different on two datasets, and the optimal value on Cora is much larger than that on Chameleon. This is also consistent with the actual situations of these two datasets, where noisy Chameleon dataset leads to large difference between observations thus the edge confidences are generally lower.

Tolerance λ. Tolerance λ controls the convergence speed of our proposed EM algorithm. GEN achieves optimal performance at around $\lambda = 0.01$, and too small or large values harm the model.

5 CONCLUSION

Graph Neural Networks rely heavily on a reasonable graph structure, while an incompatible graph will hurt their performance seriously. In this paper, we propose a novel graph estimation neural networks GEN that estimates the graph structure for GNNs to boost their performance. Specifically, we introduce a structure model to consider underlying community structure in graph generation, and present an observation model that novelly treats multi-view information, *e.g.*, multi-order neighborhood similarity, as observations of optimal graph. Based on these models, we finally attain the estimated graph in Bayesian inference framework. Extensive experimental results verify the effectiveness of GEN and its ability to estimate meaningful graph structure.

An interesting direction for future work is to extend GEN into dynamic graphs. From an intuitive point of view, the observation set could be constructed at varied time slices. However, the observation set cannot reflect the time sequence, and additional nodes during graph evolution would require retraining the entire model from scratch. Therefore, these considerations motivate a more sophisticated inference strategy to adaption.

ACKNOWLEDGMENTS

We thank anonymous reviewers for their time and effort in reviewing this paper. This work is supported in part by the National Natural Science Foundation of China (No. U20B2045, U1936104, 61972442, 61772082, 62002029), the National Key Research and Development Program of China (2018YFB1402600) and the CCF-Tencent Open Fund.

REFERENCES

[1] Muhammet Balcilar, Guillaume Renton, Pierre Héroux, Benoit Gaüzère, Sébastien Adam, and Paul Honeine. 2020. Bridging the Gap Between Spectral and Spatial Domains in Graph Neural Networks. *CoRR* abs/2003.11702 (2020).
[2] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2014. Spectral Networks and Locally Connected Networks on Graphs. In *ICLR*.
[3] Giona Casiraghi, Vahan Nanumyan, Ingo Scholtes, and Frank Schweitzer. 2017. From Relational Data to Graphs: Inferring Significant Links Using Generalized Hypergeometric Ensembles. In *SocInfo*, Vol. 10540. Springer, 111–120.
[4] Jie Chen, Tengfei Ma, and Cao Xiao. 2018. FastGCN: Fast Learning with Graph Convolutional Networks via Importance Sampling. In *ICLR*.
[5] Yu Chen, Lingfei Wu, and Mohammed J. Zaki. 2019. Deep Iterative and Adaptive Learning for Graph Neural Networks. *CoRR* abs/1912.07832 (2019).
[6] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In *NIPS*. 3837–3845.
[7] Arthur P Dempster, Nan M Laird, and Donald B Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)* 39, 1 (1977), 1–22.
[8] Matthias Fey and Jan Eric Lenssen. 2019. Fast Graph Representation Learning with PyTorch Geometric. *CoRR* abs/1903.02428 (2019).
[9] Santo Fortunato and Darko Hric. 2016. Community detection in networks: A user guide. *Physics reports* 659 (2016), 1–44.
[10] Luca Franceschi, Mathias Niepert, Massimiliano Pontil, and Xiao He. 2019. Learning Discrete Structures for Graph Neural Networks. In *ICML*, Vol. 97. 1972–1982.
[11] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. 2017. Neural Message Passing for Quantum Chemistry. In *ICML*, Vol. 70. 1263–1272.
[12] Michelle Girvan and Mark EJ Newman. 2002. Community structure in social and biological networks. *Proceedings of the national academy of sciences* 99, 12 (2002), 7821–7826.
[13] William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Representation Learning on Graphs: Methods and Applications. *IEEE Data Eng. Bull.* 40, 3 (2017), 52–74.
[14] William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *NIPS*. 1024–1034.
[15] Xiao Han, Zhesi Shen, Wen-Xu Wang, and Zengru Di. 2015. Robust reconstruction of complex networks from sparse data. *Physical review letters* 114, 2 (2015), 028701.
[16] Mark S Handcock and Krista J Gile. 2010. Modeling social networks from sampled data. *The Annals of Applied Statistics* 4, 1 (2010), 5.
[17] Paul W Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. 1983. Stochastic blockmodels: First steps. *Social networks* 5, 2 (1983), 109–137.
[18] Bo Jiang, Ziyan Zhang, Doudou Lin, Jin Tang, and Bin Luo. 2019. Semi-Supervised Learning With Graph Learning-Convolutional Networks. In *CVPR*. 11313–11320.
[19] Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiliang Tang. 2020. Graph Structure Learning for Robust Graph Neural Networks. In *KDD*. 66–74.
[20] Brian Karrer and Mark EJ Newman. 2011. Stochastic blockmodels and community structure in networks. *Physical review E* 83, 1 (2011), 016107.
[21] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.
[22] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. 2019. Predict then Propagate: Graph Neural Networks meet Personalized PageRank. In *ICLR*.
[23] Qimai Li, Zhichao Han, and Xiao-Ming Wu. 2018. Deeper Insights Into Graph Convolutional Networks for Semi-Supervised Learning. In *AAAI*. 3538–3545.
[24] Xun Li and Xiang Li. 2017. Reconstruction of stochastic temporal networks through diffusive arrival times. *Nature communications* 8, 1 (2017), 1–10.
[25] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard S. Zemel. 2016. Gated Graph Sequence Neural Networks. In *ICLR*.
[26] Dean Lusher, Johan Koskinen, and Garry Robins. 2013. *Exponential random graph models for social networks: Theory, methods, and applications*. Cambridge University Press.
[27] Peter V Marsden. 1990. Network data and measurement. *Annual review of sociology* (1990), 435–463.
[28] Travis Martin, Brian Ball, and Mark EJ Newman. 2016. Structural inference for uncertain networks. *Physical Review E* 93, 1 (2016), 012306.

[29] Geoffrey J McLachlan and Thriyambakam Krishnan. 2007. *The EM algorithm and extensions*. Vol. 382. John Wiley & Sons.

[30] MEJ Newman. 2018. Network structure from rich but noisy data. *Nature Physics* 14, 6 (2018), 542–545.

[31] Mark EJ Newman. 2002. Assortative mixing in networks. *Physical review letters* 89, 20 (2002), 208701.

[32] Mark EJ Newman. 2018. Estimating network structure from unreliable measurements. *Physical Review E* 98, 6 (2018), 062321.

[33] Mark EJ Newman, Steven H Strogatz, and Duncan J Watts. 2001. Random graphs with arbitrary degree distributions and their applications. *Physical review E* 64, 2 (2001), 026118.

[34] Mark E. J. Newman. 2010. *Networks: An Introduction*. Oxford University Press.

[35] Stephen M Omohundro. 1989. *Five balltree construction algorithms*. International Computer Science Institute Berkeley.

[36] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. *The PageRank citation ranking: Bringing order to the web*. Technical Report. Stanford InfoLab.

[37] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. 2020. Geom-GCN: Geometric Graph Convolutional Networks. In *ICLR*.

[38] Maria A Riolo and MEJ Newman. 2020. Consistency of community structure in complex networks. *Physical Review E* 101, 5 (2020), 052306.

[39] Martin Simonovsky and Nikos Komodakis. 2018. GraphVAE: Towards Generation of Small Graphs Using Variational Autoencoders. In *ICANN*, Vol. 11139. Springer, 412–422.

[40] Anton Tsitsulin, John Palowitch, Bryan Perozzi, and Emmanuel Müller. 2020. Graph Clustering with Graph Neural Networks. *CoRR* abs/2006.16904 (2020).

[41] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *ICLR*.

[42] Dan J. Wang, Xiaolin Shi, Daniel A. McFarland, and Jure Leskovec. 2012. Measurement error in network data: A re-classification. *Soc. Networks* 34, 4 (2012), 396–409.

[43] Wen-Xu Wang, Ying-Cheng Lai, and Celso Grebogi. 2016. Data based identification and prediction of nonlinear and complex dynamical systems. *Physics Reports* 644 (2016), 1–76.

[44] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural Graph Collaborative Filtering. In *SIGIR*. 165–174.

[45] Felix Wu, Amauri H. Souza Jr., Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Q. Weinberger. 2019. Simplifying Graph Convolutional Networks. In *ICML*, Vol. 97. 6861–6871.

[46] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. 2019. A Comprehensive Survey on Graph Neural Networks. *CoRR* abs/1901.00596 (2019).

[47] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks?. In *ICLR*.

[48] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. 2018. Representation Learning on Graphs with Jumping Knowledge Networks. In *ICML (Proceedings of Machine Learning Research, Vol. 80)*. 5449–5458.

[49] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. 2018. Graph Convolutional Neural Networks for Web-Scale Recommender Systems. In *KDD*. 974–983.

[50] Jiaxuan You, Rex Ying, Xiang Ren, William L. Hamilton, and Jure Leskovec. 2018. GraphRNN: Generating Realistic Graphs with Deep Auto-regressive Models. In *ICML*, Vol. 80. 5694–5703.

[51] Yingxue Zhang, Soumyasundar Pal, Mark Coates, and Deniz Üstebay. 2019. Bayesian Graph Convolutional Neural Networks for Semi-Supervised Classification. In *AAAI*. 5829–5836.

[52] Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, and Maosong Sun. 2018. Graph Neural Networks: A Review of Methods and Applications. *CoRR* abs/1812.08434 (2018).