Towards Trustworthy Machine Learning on Graph Data

by

Jiaqi Ma

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Information)
in The University of Michigan
2022

Doctoral Committee:

Professor Qiaozhu Mei, Chair
Associate Professor Danai Koutra
Associate Professor Daniel Romero
Professor Ji Zhu

Jiaqi Ma
jiaqima@umich.edu
ORCID iD: 0000-0001-8292-5901

ACKNOWLEDGEMENTS

First, I would like to express my deepest gratitude to my advisor, Qiaozhu Mei, for his incredibly generous and thoughtful guidance and support. Throughout my Ph.D. journey, Qiaozhu has always been inspiring. I learned a great deal from his out-of-box thinking, rigorous logic, and ability to make synergy among different disciplines. Qiaozhu has also always been supportive and encouraging. His encouragement has been the fuel of my confidence, especially when I was at an early exploring stage. In retrospect, I think such encouragements are the key reason that I have luckily lived through a Ph.D. life with little dark time. Beyond research, Qiaozhu also carefully guided me in all matters of academic life, such as writing and presentation skills, workshop organization, grant application, and job search, etc. I feel extremely fortunate to have Qiaozhu as my Ph.D. advisor.

I would also like to sincerely thank my committee members, Ji Zhu, Daniel Romero, and Danai Koutra. I am very fortunate to have had Ji as a long-term collaborator. Both my research and thinking benefited a lot from his statistical insights. Daniel and Danai have been on my first-year project and dissertation committee. I am very grateful for their valuable feedback, which helped better shape my work for these important milestones of my Ph.D. study.

My Ph.D. life would not be complete without the many friends and colleagues at Michigan. I want to thank all the past and current Foreseer group members, Jian Tang, Zhe Zhao, Cheng Li, Yue Wang, Wei Ai, Sam Carton, Shiyan Yan, Teng Ye, Xuan Lu, Huoran Li, Xuedong Li, Cristina Garbacea, Zhuofeng Wu, Yutong Xie,

Yachuan Liu, and Xingjian Zhang. I enjoyed all the discussions, collaborations, and life moments shared by us. I also owe thanks to my other wonderful collaborators at Michigan, Joyce Chai, Junwei Deng, Paramveer Dhillon, Shuangrui Ding, Lingyun Guo, Xiaoxiao Guo, Ziqiao Ma, Akbar K. Waljee, Ziqiao Xu, Jiaxin Ying, and Xuefei Zhang. I am grateful to all my friends at UMSI and across University of Michigan. I also want to thank the faculty and staff at UMSI for maintaining an inspiring and inclusive academic environment.

Outside Michigan, I am very grateful to my undergrad research at Tsinghua University, and my early internships at Google, which were tremendously helpful for my Ph.D. study. I would like to thank my undergrad advisor, Jie Tang, and my mentors and managers at Google, Zhe Zhao, Xinyang Yi, Lichan Hong, and Ed H. CHi, for their incredibly patient and careful instructions. I also want to thank my collaborators at Google, Bo Chang, Jilin Chen, Minmin Chen, Ang Li, Jiaxi Tang, and Ji Yang, as well as many people at the SIR team. I learned about how to conduct research with real-world impact by working with them and enjoyed the happy summer time at the Bay area.

Finally, I want to thank my parents for their unconditional love and support, without which I could not reach anything close to where I am. I am deeply grateful to Weijing Tang for always being on my side to support me and for being my source of inspiration and happiness.

TABLE OF CONTENTS

LIST OF TABLES

Table

LIST OF FIGURES

Figure

LIST OF APPENDICES

Appendix

ABSTRACT

Machine learning has been applied to more and more socially-relevant scenarios that influence our daily lives, ranging from social media and e-commerce to self-driving cars and criminal justice. It is therefore crucial to develop trustworthy machine learning methods that perform reliably, in order to avoid negative impacts on individuals and society. In this dissertation, we focus on understanding and improving the trustworthiness of *graph machine learning*, which poses unique challenges due to the complex relational structure of the graph data.

In particular, we view the trustworthiness of a machine learning model as being reliable under exceptional conditions. For example, the performance of a machine learning model should not degrade seriously under adversarial attacks or on a subpopulation, which respectively corresponds to the problems of adversarial robustness or fairness. The unique challenges for trustworthy graph machine learning are that there are many more complicated and sometimes implicit exceptional conditions in the context of graph data. This dissertation identifies under-explored exceptional conditions, understands the expected model behavior under the identified exceptional conditions, and improves the existing models under such exceptional conditions.

Specifically, we focus on *graph neural networks* (GNNs), a family of popular graph machine learning models that leverage recent advances in deep learning. In this dissertation, we identify three exceptional conditions of GNNs. First, we study the adversarial robustness of GNNs with a new and practical threat model inspired by social network application scenarios, and investigate when and why GNNs may suffer

from the adversarial attacks. Second, we find that existing GNNs can be misspecified for many real-world graph data and develop a novel framework to improve existing models. Finally, we discover a type of unfairness of GNN predictions among subpopulations of test nodes that is relevant to the structural positions of the nodes. We also propose an active learning framework to mitigate the unfairness problem.

CHAPTER I

Introduction

Artificial Intelligence (AI), especially Machine Learning (ML), has been integrated into human society as a general-purpose technology[1], with the promise of reshaping our daily lives in numerous aspects, ranging from social media and e-commerce to self-driving cars and criminal justice. However, despite the enormous empirical success and commercial value brought by AI and ML, more widespread deployment of these techniques requires better understandings of ML models' impacts on society. As a result, *Trustworthy ML* arises as an increasingly popular research direction.

Trustworthy ML is an umbrella concept that includes various topics regarding the reliability and transparency of ML, such as fairness, robustness, explainability, etc.

For example, ML models may perform systematically worse on specific subpopulations, which leads to fairness concerns. As a result, there has been a rapid increase in research interests in ML fairness. There have also been real-world ML applications that demonstrate biases and unfairness: Amazon's AI recruiting tool was found to have gender biases [37]; a once widely-used criminal prediction tool, Correctional Offender Management Profiling for Alternative Sanctions (COMPAS), was found to be racially biased [4].

As another example, ML models have been shown to be sensitive to small adversarial perturbations added to the data, and thus vulnerable to adversarial attacks [136].

[1] `https://en.wikipedia.org/wiki/General-purpose_technology`

For instance, state-of-the-art computer vision models can be misled to recognize a Stop sign as a Speed Limit sign through a seemingly random graffiti on the Stop sign [45].

Due to the variety of relevant topics and the bibliographical history of our scientific understanding of trustworthy ML, the community has naturally grown a set of relatively well-accepted conceptual categories of trustworthiness issues, which include but are not limited to *fairness*, *robustness*, *safety*, *privacy*, *accountability*, *explainability*, and *causality*. While this conceptual classification, like any classification system, helps to ease the understanding of the field, it can also sometimes be misleading[2].

First, this classification can make the different issues of trustworthy ML perceived as isolated topics. However, these different trustworthiness issues can be conflicting or relevant to each other. For example, there are inherent conflicts between certain notions of privacy and fairness [32, 24]. On the other hand, fairness can also be relevant to out-of-domain generalization [99]. Furthermore, explainable ML [41] and causal inference [113] could be candidate solutions to some of the fairness or robustness problems. A flat taxonomy of the conceptual categories fails to capture the rich interrelationships among the different topics.

Second, this classification tends to drive efforts in searching for overly generic solutions for each topic, which may not be the best way to approach the problem of trustworthy ML. Because of the conceptual nature of the topics, there are often various intuitively sound ways to formalize a trustworthiness concept (e.g., fairness or robustness) into quantitative notions, while achieving trustworthiness in terms of all notions simultaneously is unrealistic. For example, Kleinberg et al. [78] proved that it is generally impossible to have a single algorithm that simultaneously meets three common fairness standards. Therefore, there is no generic solution that is a panacea for all applications. In addition, the importance and the proper formulation

[2]See Bowker and Star [15] for a comprehensive investigation on the impacts that classification systems can have.

of different trustworthiness issues are highly application-specific. In terms of the importance of different aspects of trustworthiness, for example, a self-driving car is likely to suffer from adversarial attacks as it takes an input of data in the wild [45]; in contrast, performing adversarial attacks on Electronic Health Records (EHR) data is practically much more difficult, as the data are generated by authorized medical experts and circulated in closed systems. On the other hand, the privacy standard for EHR data is much higher than that for driving data. In terms of the proper formulation of trustworthiness, it has been shown that the choice of the formulation should leverage the perception of the stakeholders in the specific application [28]. Overall, trustworthiness should be investigated as properties of an ML technique *situated in a specific type of application scenario*, as opposed to properties of a *generic* ML technique.

In a sense, the aforementioned categories are both too narrow and too broad. They are too narrow because this classification misses many interactions among the different categories. They are also too broad because the concept of each category can have many different (and even conflicting) formulations.

In light of the limitations of the conventional and conceptual definition and classification of trustworthiness, this dissertation follows an alternative operational procedure to organize and approach the study of trustworthy ML. In particular, we use the following application-centric procedure to identify trustworthiness issues.

1. We are given a specific application scenario and certain metrics that measure the behaviors of an ML model (e.g., the predictive performance).

2. We assume that there is a *normal condition* where the ML model is behaving well.

3. Finally, we identify trustworthiness issues by revealing certain *exceptional conditions* where the ML model has unexpected behaviors or degraded performance.

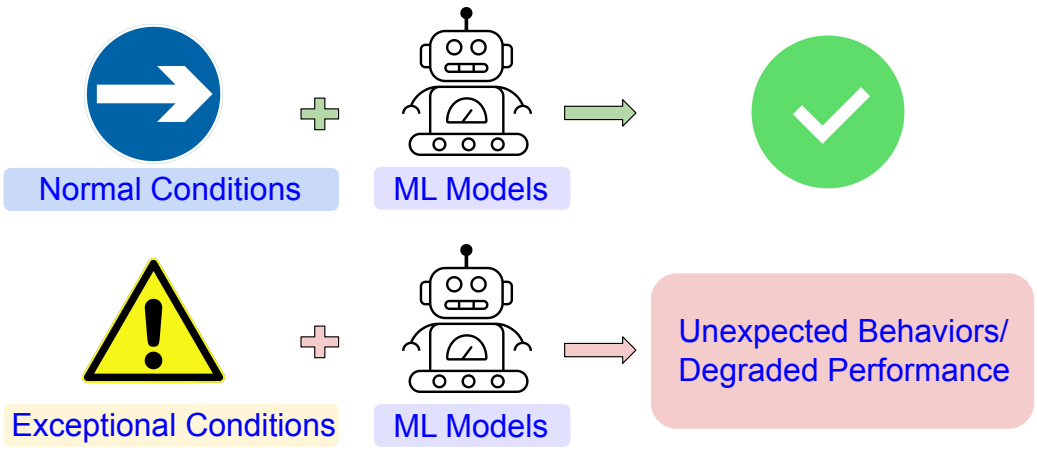


Figure I.1: A trustworthiness issue can be identified as the discrepancy of the model behaviors between an exceptional condition and the normal condition.

As shown in Figure I.1, a trustworthiness issue can be identified as *the discrepancy of the model behaviors between an exceptional condition and the normal condition.*

Many existing trustworthiness notions can be reformulated following this procedure. For example, the unfairness problems of ML models are often due to their degraded performance on specific minority sub-populations, compared to their performance on majority sub-populations. The adversarial vulnerability of ML refers to their degraded performance under adversarial attacks, compared to their performance on clean data. On the other hand, some other trustworthiness concepts, such as explainability or causality, cannot be directly formulated through the procedure above. To some extent, a model being unfair or non-robust will have direct consequences, while explainability or causality can be viewed as candidate solutions that mitigate the problems (e.g., unfairness or non-robustness). The procedure above focuses on the trustworthiness concepts that are problems rather than solutions. This procedure also emphasizes grounding the trustworthiness issues with the application scenarios.

To constrain the scope in a particular type of application scenarios, this dissertation focuses on the trustworthiness of *Graph Machine Learning* (GML). Real-world data are rich in relational structures, often represented in the form of graphs. For example, users on social media or sensors in an Internet-of-Things system are inter-

connected through a graph structure. Such relational graph structures provide significant predictive power when exploited properly in prediction tasks. GML is a popular family of machine learning techniques that leverage graph structures into predictive models. Recently, GML has demonstrated great performance in many applications that influence everyone's daily life. For common examples, GML has been playing a major role in industrial recommender systems at Uber Eats [65], Amazon [162], and Pinterest [157]; GML is also broadly used to model geographical data in tasks such as ETA prediction in Google Map [38] or real estate price estimation [114]. Moreover, due to the ubiquity of relational structures, GML methods either have been applied to or are ready to be applied to high-stake decision-making problems such as social justice. Examples include crime prediction and data-driven prosecution [68, 156], police misconduct prediction [22], risk assessment of parole decisions [132], surveillance for public safety [95], and many other social justice and security problems in general [111].

Given the numerous socially-relevant application scenarios of GML, the trustworthiness issues of such ML systems become crucially important. Moreover, compared to conventional ML, there are unique challenges in understanding and improving the trustworthiness issues of GML due to the complex relational structures. In particular, there are many more complicated and sometimes implicit exceptional conditions in the context of GML. Taking adversarial attack as an example, in conventional ML setups, the attacker mostly conducts attacks by adding adversarial perturbations to the input features. For GML, there are more complicated threats in practice: the attacker can perturb not only the node attributes but also the graph structures; the attacker can also conduct an indirect attack that influences the prediction on one node by perturbing its neighbors. In terms of ML fairness among subpopulations, most conventional literature studies subpopulations concerning certain sensitive attributes, such as gender or ethnicity. In graph data, one can investigate subpopulations based on the

graph structures, such as node centrality [12, 13] or community structures [51, 47]. The structural traits of people in a social network are often relevant to their socioeconomic status suggested by social science theories [53, 16]. The unique adversarial threats and structure-based subpopulations in graph data present exceptional conditions that are not well-explored in conventional ML literature, making trustworthy GML more challenging.

This dissertation aims to address such unique challenges toward understanding and improving the trustworthiness of GML. Specifically, this dissertation aims to answer the following three types of research questions and demonstrate the methodology under three application scenarios.

1. What are potential exceptional conditions a GML model may encounter in a real-world application scenario?

2. What are the expected behaviors of the GML model under the identified exceptional conditions?

3. How to mitigate the performance disparity of the GML model under the identified exceptional conditions?

Among the GML methods, we focus on *Graph Neural Networks* (GNNs) [52, 124, 77], which are a large family of trending GML models that leverage the recent advancements of *deep learning* [83] into GML, and have shown superior performance on many real-world applications.

The overall structure is organized as follows. Chapter I provides an introduction and a summary of the contributions. Chapter II provides preliminary background concepts for the rest of the dissertation and a brief literature review on trustworthy GML. Chapter III to Chapter VI introduce the main technical contributions of this dissertation. Finally, Chapter VII provides a conclusion and a discussion on future directions. The technical contributions are summarized in the following Section 1.1.

1.1 Introduction to Individual Technical Contributions

In this dissertation, we investigate three exceptional conditions for GNNs. First, Chapter III [97] investigates a practical adversarial attack condition for GNNs inspired by social network application scenarios, and demonstrates that existing GNNs are vulnerable in such an adversarial condition. Second, Chapter IV [98] discovers a condition that, for many real-world graph data, most existing GNNs can be misspecified and thus have suboptimal performance. We also introduce a novel framework to improve existing GNNs. Third, Chapter V [99] reveals that GNNs have lower prediction accuracy systematically on specific subgroups of nodes relevant to their structural characteristics, leading to a unique type of unfairness concerns for GNNs. Chapter VI [100] follows up on the problem revealed in Chapter V and proposes an active learning approach to improve the generalization and fairness of existing GNNs. Most of the technical contributions introduced in this dissertation have been published or preprinted, as suggested by the citations in this paragraph.

1.1.1 Practical Adversarial Attacks on Graph Neural Networks (Chapter III)

Like deep learning models on images or text, GNNs are also shown to be vulnerable to adversarial attacks [169]. Most early studies about adversarial attacks on GNNs assume the attacker has at least access to the GNN predictions on all nodes, following common adversarial attack setups for image classifiers. However, such setups, i.e., the attacker having access to global information of the GNN, tend to be unrealistic for many GNN applications such as social networks. For instance, an attacker usually only has access to information of a limited set of nodes on a social network. In Chapter III, we study adversarial attacks on GNNs under a highly restricted black-box setup that is practical to social network applications. Under this setup, we derive a principled attack strategy and show that an effective attack on GNNs is still possible.

The key insight is that the graph structure of GNNs leaks considerable information about the models. In particular, a few nodes are significantly more influential on the model than others, analogous to the fragility of scale-free networks under targeted attack. The attacker can therefore poison the model without knowing any information about the model parameters or predictions for most nodes, to the degree that is not likely for deep learning models on non-graph-structured data.

1.1.2 The Representational and Correlational Roles of Graphs in Graph Neural Networks (Chapter IV)

Graph-structured data are ubiquitous. However, graphs encode diverse types of information and thus play different roles in data representation. In Chapter IV, we distinguish the *representational* and the *correlational* roles played by the graphs in different real-world graph data, and we investigate how GNN models can effectively leverage both types of information. Conceptually, the representational information provides guidance for the model to construct better node features; while the correlational information indicates the correlation between node outcomes conditional on node features. Through a simulation study, we find that many popular GNN models have suboptimal performance when the graph plays a significant role in providing correlational information. We also propose a general framework to improve existing GNNs by leveraging the idea of the copula, a principled way to describe the dependence among multivariate random variables. The proposed framework, Copula Graph Neural Network (CopulaGNN), can take a wide range of GNN models as base models and utilize both representational and correlational information stored in the graphs. Experimental results on two types of regression tasks verify the effectiveness of the proposed method.

1.1.3 Generalization and Fairness of Graph Neural Networks (Chapter V)

Establishing theoretical guarantees for the generalization and fairness of machine learning models is a key step towards developing trustworthy machine learning models. While a large body of literature has studied the generalization and fairness of machine learning under the supervised learning setup with *independent and identically distributed* (IID) data, little work has been done for GNNs under the semi-supervised learning setup, where non-IID graph-structured data are involved. In Chapter V, we provided one of the first learning-theoretic generalization bounds for GNNs under the semi-supervised learning setup by applying a PAC-Bayesian analysis. With the theoretical tools developed above, we further investigated how the structural characteristics of different nodes affect the GNN predictions on them. We found that there is a systematic accuracy disparity among test nodes. In particular, we showed that test nodes that are "farther away" from the training nodes tend to suffer from a more significant generalization error, suggesting an unfairness unique to machine learning on graph-structured data.

1.1.4 Partition-Based Active Learning for Graph Neural Networks (Chapter VI)

Motivated by the analysis in Chapter V, we propose a novel active learning approach for GNNs. The proposed method first splits the graph into disjoint partitions and then selects representative nodes within each partition as training data. The training data is selected in a balanced way such that distances from test nodes to training nodes are significantly reduced, which in turn reduces the generalization error for each subgroup. Extensive experiments on multiple benchmark datasets demonstrate that the proposed method improves both the generalization and fairness of GNNs. In addition, the proposed method does not introduce additional hyperparameters, which is crucial for model training, especially in the active learning setting

where a labeled validation set may not be available.

CHAPTER II

Preliminaries

All of the technical chapters of this dissertation focus on *graph neural networks* (GNNs) in the context of *graph-based semi-supervised learning* problem. This chapter formally introduces these two concepts to familiarize readers with the preliminary background. Some of the introductions here may be repeated (sometimes with slightly different notations) in the following technical chapters as reminders for the readers and to ease the notations in each chapter. This chapter also provides a brief literature review on trustworthy GML.

2.1 Graph-Based Semi-Supervised Learning

This dissertation focuses on one of the most popular types of GML tasks, graph-based semi-supervised learning. Traditional machine learning methods typically treat data samples as independent and approximate a mapping function from the features to the outcome of each individual sample. However, many real-world data, such as social media or scientific articles, often come with richer relational information among the individual samples. Graph-based semi-supervised learning is a family of problems in such scenarios where the relational information is stored in a graph structure with the data samples as nodes, and the learning task is to predict the outcomes of unlabeled nodes based on the node features, the graph structure, as well as the labels of a subset

of nodes.

Formally, given a set of data samples $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$, $x_i \in \mathbb{R}^D$ and $y_i \in \mathbb{R}^K$ are the feature and outcome vectors of sample i respectively. We further denote $X \in \mathbb{R}^{N\times D}$ and $Y \in \mathbb{R}^{N\times K}$ as the matrices formed by feature and outcome vectors. The dataset also comes with a graph $G = (V, E)$ with the data samples as nodes, where $V = \{1, 2, \cdots, N\}$ is the set of nodes and $E \subseteq V \times V$ is the set of edges. In the semi-supervised learning setting, only $0 < M < N$ samples have observed their outcome labels, and the outcome labels of other samples are missing. Without loss of generality, we assume the outcomes of the samples $1, 2, \cdots, M$ are observed, and that of $M+1, \cdots, N$ are missing. Therefore we can partition the outcome matrix as

$$Y = \begin{bmatrix} Y_{\text{obs}} \\ Y_{\text{miss}} \end{bmatrix}.$$

The goal of graph-based semi-supervised learning is to infer Y_{miss} based on $(X, \boldsymbol{Y}_{\text{obs}}, G)$. This is usually done by learning a prediction model $y = f(x; X, G)$ using empirical risk minimization, optionally with regularizations:

$$\hat{f} = \arg\min_f \frac{1}{M} \sum_{i=1}^{M} \mathcal{L}(y_i, f(x_i; X, G)) + \lambda \mathcal{R}(f; G),$$

where $\mathcal{L}(\cdot, \cdot)$ is a loss function, $\mathcal{R}(\cdot; G)$ is a graph-based regularization term, and λ is a hyper-parameter controlling the strength of the regularization. Then $\hat{f}$ is used to predict Y_{miss}. While $\hat{f}$ can take any general form, we are particularly interested in the form of GNNs (see Section 2.2).

There are two specific learning settings, namely *transductive learning* and *inductive learning*, which are common in graph-based semi-supervised learning. Transductive learning assumes that X and G are fully observed during both the learning and inference stages. In contrast, inductive learning assumes $X_{M+1:N}$ and the nodes of

$M+1, \cdots, N$ in G are missing during the learning stage but available during the inference stage. In this dissertation, we will mainly focus on the transductive learning setting.

2.2 Graph Neural Networks

This dissertation also focuses on a popular family of GML methods, GNNs. Given the graph G, a GNN model is a function $f_G : \mathbb{R}^{N \times D} \to \mathbb{R}^{N \times K}$ that maps the node features X to output logits of each node. We denote the output logits of all nodes as a matrix $H \in \mathbb{R}^{N \times K}$ and $H = f_G(X)$. A GNN f_G is usually built by stacking a certain number (L) of layers, with the l-th layer, $1 \leq l \leq L$, taking the following form:

$$H_i^{(l)} = \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij} \boldsymbol{W}_l H_j^{(l-1)}\right), \tag{2.1}$$

where $H^{(l)} \in \mathbb{R}^{N \times D_l}$ is the hidden representation of nodes with D_l dimensions, output by the l-th layer; $\boldsymbol{W}_l$ is a learnable linear transformation matrix; σ is an element-wise nonlinear activation function; and different GNNs have different normalization terms α_{ij}. For instance, $\alpha_{ij} = 1/\sqrt{d_i d_j}$ or $\alpha_{ij} = 1/d_i$ in Graph Convolutional Networks (GCN) [77]. In addition, $H^{(0)} = X$ and $H = H^{(L)}$.

2.3 Trustworthy Graph Machine Learning

This section provides a preliminary literature review on a few aspects of trustworthy GML. This literature review is by no means comprehensive, and only serves the purpose of guiding the readers toward the technical chapters. While we have discussed the limitations of the conceptual classification of trustworthiness in Chapter I, this is still the most common way of summarizing literature in survey papers. We therefore have followed this convention in this literature review section.

2.3.1 Adversarial Attacks

The study of adversarial attacks on GNNs has surged recently. A taxonomy of existing work has been summarized by Jin et al. [70], and we give a brief introduction here. First, there are two types of machine learning tasks on graphs that are commonly studied, node-level classification and graph-level classification. Next, there are a couple of choices of the attack form. For example, the attack can happen either during model training (poisoning) or during model testing (evasion); the attacker may aim to mislead the prediction on specific nodes (targeted attack) [169] or damage the overall task performance (untargeted attack) [168]; the adversarial perturbation can be done by modifying node features, adding or deleting edges, or injecting new nodes [135]. For the adversarial perturbation, most existing works of untargeted attacks apply global constraints on the proportion of node features or the number of edges to be altered. Our work in Chapter III belongs to untargeted evasion attacks on node-level classification tasks. We set a novel local constraint on node access, which is more realistic in practice: perturbation on top (e.g., celebrity) nodes is prohibited and only a small number of nodes can be perturbed. Finally, depending on the attacker's knowledge about the GNN model, existing work can be split into three categories: white-box attacks [154, 26, 149] have access to full information about the model, including model parameters, input data, and labels; grey-box attacks [168, 169, 135] have partial information about the model and the exact setups vary in a range; in the most challenging setting, black-box attacks [35, 10, 25] can only access the input data and sometimes the black-box predictions of the model. In Chapter III, we consider an even more strict black-box attack setup, where model predictions are invisible to the attackers. As far as we know, the only prior works that conduct untargeted black-box attacks without access to model predictions are those by Bojchevski and Günnemann [10] and Chang et al. [25]. However, both of them require access to embeddings of nodes, which are prohibited as well in our setup.

2.3.2 Generalization

The majority of existing literature that aims to develop theoretical understandings of GNNs has focused on the expressive power of GNNs (see Sato [123] for a survey along this line). At the same time, the number of studies trying to understand the generalization of GNNs is rather limited. Among them, some [42, 49, 89] focus on graph-level tasks, the analyses of which cannot be easily applied to node-level tasks. As far as we know, Scarselli et al. [125], Verma and Zhang [142], and Baranwal et al. [6] are the only existing studies investigating the generalization of GNNs on node-level tasks, even though node-level tasks are more common in reality. Scarselli et al. [125] present an upper bound of the VC-dimension of GNNs; Verma and Zhang [142] derive a stability-based generalization bound for a single-layer GCN [77] model. Yet, both Scarselli et al. [125] and Verma and Zhang [142] (implicitly) assume that the training nodes are IID samples from a certain distribution, which does not align with the common practice of node-level semi-supervised learning. Baranwal et al. [6] investigate the generalization of graph convolution under a specific data generating mechanism (i.e., the contextual stochastic block model [39]). Our work in Chapter V presents the first generalization analysis of GNNs for non-IID node-level tasks without strong assumptions on the data generating mechanism.

2.3.3 Fairness

The fairness issues of machine learning on graphs start to receive research attention recently. Following conventional machine learning fairness literature, most previous work along this line [1, 14, 18, 34, 80, 118, 134, 159] concerns fairness with respect to a given sensitive attribute, such as gender or race, which defines protected groups. In practice, the fairness issues of learning on graphs are much more complicated due to the asymmetric nature of the graph-structured data. However, only a few studies [75] investigate the unfairness caused by the graph structure without knowing

a sensitive feature. Moreover, in a node-level semi-supervised learning task, the non-IID sampling of training nodes brings additional uncertainty to the fairness of the learned models. The work in Chapter V is the first to present a learning theoretic analysis under this setup, which in turn suggests how the graph structure and the selection of training nodes may influence the fairness of machine learning on graphs.

CHAPTER III

Practical Adversarial Attacks on Graph Neural Networks

3.1 Introduction

Graph neural networks (GNNs) [151], the family of deep learning models on graphs, have shown promising empirical performance on various applications of machine learning to graph data, such as recommender systems [157], social network analysis [85], and drug discovery [131]. Like other deep learning models, GNNs have also been shown to be vulnerable under adversarial attacks [169], which has recently attracted increasing research interest [70]. Indeed, adversarial attacks have been an efficient tool to analyze both the theoretical properties as well as the practical accountability of graph neural networks. As graph data have more complex structures than image or text data, researchers have come up with diverse adversarial attack setups. For example, there are different tasks (node classification and graph classification), assumptions of attacker's knowledge (white-box, grey-box, and black-box), strategies (node feature modification and graph structure modification), and corresponding budget or other constraints (norm of feature changes or number of edge changes).

Despite these research efforts, there is still a considerable gap between the existing

attack setups and the reality. It is unreasonable to assume that an attacker can alter the input of a large proportion of nodes, and even if there is a budget limit, it is unreasonable to assume that they can attack any node as they wish. For example, in a real-world social network, the attackers usually only have access to a few bot accounts, and they are unlikely to be among the top nodes in the network; it is difficult for the attackers to hack and alter the properties of celebrity accounts. Moreover, an attacker usually has limited knowledge about the underling machine learning model used by the platform (e.g., they may roughly know what types of models are used but have no access to the model parameters or training labels). Motivated by the real-world scenario of attacks, in this chapter we study a new type of black-box adversarial attack for node classification tasks, which is more restricted and more realistic, assuming that the attacker has no access to the model parameters or predictions. Our setup differs from existing work with a novel constraint on node access, where attackers only have access to a subset of nodes in the graph, and they can only manipulate a small number of them.

The proposed black-box adversarial attack requires a two-step procedure: 1) selecting a small subset of nodes to attack under the limits of node access; 2) altering the node attributes or edges under a per-node budget. In this chapter, we focus on the first step and study the node selection strategy. The key insight of the proposed strategy lies in the observation that, with no access to the GNN parameters or predictions, the strong *structural inductive biases* of the GNN models can be exploited as an effective information source of attacks. The structural inductive biases encoded by various neural architectures (e.g., the convolution kernel in convolutional neural networks) play important roles in the success of deep learning models. GNNs have even more explicit structural inductive biases due to the graph structure and their heavy weight sharing design. Theoretical analyses have shown that the understanding of structural inductive biases could lead to better designs of GNN models [153, 79].

From a new perspective, our work demonstrates that such structural inductive biases can turn into security concerns in a black-box attack, as the graph structure is usually exposed to the attackers.

Following this insight, we derive a node selection strategy with a formal analysis of the proposed black-box attack setup. By exploiting the connection between the backward propagation of GNNs and random walks, we first generalize the gradient-norm in a white-box attack into a model-independent importance score similar to the PageRank. In practice, attacking the nodes with high importance scores increases the classification loss significantly but does not generate the same effect on the mis-classification rate. Our theoretical and empirical analyses suggest that such discrepancy is due to the diminishing-return effect of the mis-classification rate. We further propose a greedy correction procedure for calculating the importance scores. Experiments on three real-world benchmark datasets and popular GNN models show that the proposed attack strategy significantly outperforms baseline methods. We summarize our main contributions as follows:

1. We propose a novel setup of black-box attacks for GNNs with a constraint of limited node access, which is by far the most restricted and realistic compared to existing work.

2. We demonstrate that the structural inductive biases of GNNs can be exploited as an effective information source of black-box adversarial attacks.

3. We analyze the discrepancy between classification loss and mis-classification rate and propose a practical greedy method of adversarial attacks for node classification tasks.

4. We empirically verify the effectiveness of the proposed method on three benchmark datasets with popular GNN models.

3.2 Principled Black-Box Attack Strategies with Limited Node Access

In this section, we derive principled strategies to attack GNNs under the novel black-box setup with limited node access. We first analyze the corresponding *white-box* attack problem in Section 3.2.2 and then adapt the theoretical insights from the white-box setup to the black-box setup and propose a black-box attack strategy in Section 3.2.3. Finally, in Section 3.2.4, we correct the proposed strategy by taking into account of the diminishing-return effect for the mis-classification rate.

3.2.1 Preliminary Notations

We first introduce necessary notations. We denote a graph as $G = (V, E)$, where $V = \{1, 2, \ldots, N\}$ is the set of N nodes, and $E \subseteq V \times V$ is the set of edges. For a node classification problem, the nodes of the graph are collectively associated with node features $X \in \mathbb{R}^{N \times D}$ and labels $y \in \{1, 2, \ldots, K\}^N$, where D is the dimensionality of the feature vectors and K is the number of classes. Each node i's local neighborhood including itself is denoted as $\mathcal{N}_i = \{j \in V \mid (i, j) \in E\} \cup \{i\}$, and its degree as $d_i = |\mathcal{N}_i|$. To ease the notation, for any matrix $A \in \mathbb{R}^{D_1 \times D_2}$ in this chapter, we refer A_j to the transpose of the j-th row of the matrix, i.e., $A_j \in \mathbb{R}^{D_2}$.

GNN models. Given the graph G, a GNN model is a function $f_G : \mathbb{R}^{N \times D} \to \mathbb{R}^{N \times K}$ that maps the node features X to output logits of each node. We denote the output logits of all nodes as a matrix $H \in \mathbb{R}^{N \times K}$ and $H = f_G(X)$. A GNN f_G is usually built by stacking a certain number (L) of layers, with the l-th layer, $1 \leq l \leq L$, taking the following form:

$$H_i^{(l)} = \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij} \boldsymbol{W}_l H_j^{(l-1)}\right), \tag{3.1}$$

where $H^{(l)} \in \mathbb{R}^{N \times D_l}$ is the hidden representation of nodes with D_l dimensions, output by the l-th layer; $\boldsymbol{W}_l$ is a learnable linear transformation matrix; σ is an element-wise

nonlinear activation function; and different GNNs have different normalization terms α_{ij}. For instance, $\alpha_{ij} = 1/\sqrt{d_i d_j}$ or $\alpha_{ij} = 1/d_i$ in Graph Convolutional Networks (GCN) [77]. In addition, $H^{(0)} = X$ and $H = H^{(L)}$.

Random walks. A random walk [94] on G is specified by the matrix of transition probabilities, $M \in \mathbb{R}^{N \times N}$, where

$$M_{ij} = \begin{cases} 1/d_i, & \text{if } (i,j) \in E \text{ or } j = i, \\ 0, & \text{otherwise.} \end{cases}$$

Each M_{ij} represents the probability of transiting from i to j at any given step of the random walk. And powering the transition matrix by t gives us the t-step transition matrix M^t.

3.2.2 White-Box Adversarial Attacks with Limited Node Access

Problem formulation. Given a classification loss $\mathcal{L} : \mathbb{R}^{N \times K} \times \{1, \ldots, K\}^N \to \mathbb{R}$, the problem of white-box attack with limited node access can be formulated as an optimization problem as follows:

$$\begin{aligned} \max_{S \subseteq V} \quad & \mathcal{L}(H, y) \\ \text{subject to} \quad & |S| \leq r, d_i \leq m, \forall i \in S \\ & H = f(\tau(X, S)), \end{aligned} \tag{3.2}$$

where $r, m \in \mathbb{Z}^+$ respectively specify the maximum number of nodes and the maximum degree of nodes that can be attacked. Intuitively, we treat high-degree nodes as a proxy of celebrity accounts in a social network. For simplicity, we have omitted the subscript G of the learned GNN classifier f_G. The function $\tau : \mathbb{R}^{N \times D} \times 2^V \to \mathbb{R}^{N \times D}$ perturbs the feature matrix X based on the selected node set S (i.e., *attack set*). Under the white-box setup, theoretically τ can also be optimized to maximize the

loss. However, as our goal is to study the node selection strategy under the black-box setup, we set τ as a pre-determined function. In particular, we define the j-th row of the output of τ as $\tau(X, S)_j = X_j + \mathbb{1}[j \in S]\epsilon$, where $\epsilon \in \mathbb{R}^D$ is a small constant noise vector constructed by attackers' domain knowledge about the features. In other words, the same small noise vector is added to the features of every attacked node.

We use the Carlili-Wagner loss for our analysis, a close approximation of cross-entropy loss and has been used in the analysis of adversarial attacks on image classifiers [21]:

$$\mathcal{L}(H, y) \triangleq \sum_{j=1}^{N} \mathcal{L}_j(H_j, y_j) \triangleq \sum_{j=1}^{N} \max_{k \in \{1,\dots,K\}} H_{jk} - H_{jy_j}. \tag{3.3}$$

The change of loss under perturbation. Next we investigate how the overall loss changes when we select and perturb different nodes. We define the change of loss when perturbing the node i as a function of the perturbed feature vector x:

$$\Delta_i(x) = \mathcal{L}(f(X'), y) - \mathcal{L}(f(X), y), \text{ where } X'_i = x \text{ and } X'_j = X_j, \forall j \neq i.$$

To concretize the analysis, we consider the GCN model with $\alpha_{ij} = \frac{1}{d_i}$ in our following derivations. Suppose f is an L-layer GCN. With the connection between GCN and random walk [153] and Assumption 3.2.1 on the label distribution, we can show that, in expectation, the first-order Taylor approximation $\widetilde{\Delta}_i(x) \triangleq \Delta_i(X_i) + (\nabla_x \Delta_i(X_i))^T(x - X_i)$ is related to the sum of the i-th column of the L-step random walk transition matrix M^L. We formally summarize this finding in Proposition 1.

Assumption 3.2.1 (Label Distribution)**.** *Assume the distribution of the labels of all nodes follows the same constant categorical distribution, i.e.,*

$$\Pr[y_j = k] = q_k, \forall j = 1, 2, \dots, N,$$

where $0 < q_k < 1$ *for* $k = 1, 2, \ldots, K$ *and* $\sum_{k=1}^{K} q_k = 1$. *Moreover, since the classifier* f *has been well-trained and fixed, the prediction of* f *should capture certain relationships among the* K *classes. Specifically, we assume the chance for* f *predicting any node* j *as any class* $k \in \{1, \ldots, K\}$, *conditioned on the node label* $y_j = l \in \{1, \ldots, K\}$, *confines to a certain distribution* $p(k \mid l)$, *i.e.,*

$$\Pr\left[\left(\underset{c\in\{1,\ldots,K\}}{\arg\max} H_{jc}\right) = k \mid y_j = l\right] = p(k \mid l).$$

Proposition 1. For an L-layer GCN model, if Assumption 3.2.1 and a technical assumption about the GCN[1] hold, then

$$\delta_i \triangleq \mathbb{E}\left[\widetilde{\Delta}_i(x)\,|_{x=\tau(X,\{i\})_i}\right] = C\sum_{j=1}^{N}[M^L]_{ji},$$

where C is a constant independent of i.

3.2.3 Adaptation from the White-Box Setup to the Black-Box Setup

Now we turn to the *black-box* setup where we have no access to the model parameters or predictions. This means we are no longer able to evaluate the objective function $\mathcal{L}(H, y)$ of the optimization problem (3.2). Proposition 1 shows that the relative ratio of δ_i/δ_j between different nodes $i \neq j$ only depends on the random walk transition matrix, which we can easily calculate based on the graph G. This implies that we can still approximately optimize the problem (3.2) in the black-box setup.

Node selection with importance scores. Consider the change of loss under the perturbation of a set of nodes S. If we write the change of loss as a function of the perturbed features and take the first order Taylor expansion, which we denote as δ, we have $\delta = \sum_{i\in S}\delta_i$. Therefore δ is maximized by the set of r nodes with degrees less

[1]This is an assumption made by Xu et al. [153], which we list as Assumption A.1.1 in Appendix A.1.

than m and the largest possible δ_i, where m, r are the limits of node access defined in the problem (3.2). Therefore, we can define an *importance score* for each node i as the sum of the i-th column of M^L, i.e., $I_i = \sum_{j=1}^{N} [M^L]_{ji}$, and simply select the nodes with the highest importance scores to attack. We denote this strategy as **RWCS** (Random Walk Column Sum). We note that RWCS is similar to PageRank. The difference between RWCS and PageRank is that the latter uses the stationary transition matrix M^∞ for a random walk with restart.

Empirically, RWCS indeed significantly increases the classification loss (as shown in Section 3.3.2). The nonlinear loss actually increases linearly w.r.t. the perturbation strength (the norm of the perturbation noise ϵ) for a wide range, which indicates that $\widetilde{\Delta}_i$ is a good approximation of Δ_i. Surprisingly, RWCS fails to continue to increase the mis-classification rate (which matters more in real applications) when the perturbation strength becomes larger. Details of this empirical finding are shown in Figure III.1 in Section 3.3.2. We conduct additional formal analyses on the mis-classification rate in the following section and find a diminishing-return effect of adding more nodes to the attack set when the perturbation strength is adequate.

3.2.4 Diminishing-Return of Mis-classification Rate and its Correction

Analysis of the diminishing-return effect. Our analysis is based on the investigation that each target node $i \in V$ will be mis-classified as we increase the attack set.

To assist the analysis, we first define the concepts of *vulnerable function* and *vulnerable set* below.

Definition 3.2.2 (Vulnerable Function)**.** We define the vulnerable function $g_i : 2^V \rightarrow$

$\{0, 1\}$ of a target node $i \in V$ as, for a given attack set $S \subseteq V$,

$$g_i(S) = \begin{cases} 1, & \text{if } i \text{ is mis-classified when attacking } S, \\ 0, & \text{if } i \text{ is correctly-classified when attacking } S. \end{cases}$$

Definition 3.2.3 (Vulnerable Set)**.** We define the vulnerable set of a target node $i \in V$ as a set of all attack sets that could lead i to being mis-classified:

$$A_i \triangleq \{S \subseteq V \mid g_i(S) = 1\}.$$

We also make the following assumption about the vulnerable function.

Assumption 3.2.4. *g_i is non-decreasing for all $i \in V$, i.e., if $T \subseteq S \subseteq V$, then $g_i(T) \leq g_i(S)$.*

With the definitions above, the mis-classification rate can be written as the average of the vulnerable functions: $h(S) = \frac{1}{N} \sum_{i=1}^{N} g_i(S)$. By Assumption 3.2.4, h is also clearly non-decreasing.

We further define the *basic vulnerable set* to characterize the minimal attack sets that can lead a target node to being mis-classified.

Definition 3.2.5 (Basic Vulnerable Set)**.** $\forall i \in V$, we call $B_i \subseteq A_i$ a basic vulnerable set of i if,

1) $\emptyset \notin B_i$; if $\emptyset \in A_i, B_i = \emptyset$;

2) if $\emptyset \notin A_i$, for any nonempty $S \in A_i$, there exists a $T \in B_i$ s.t. $T \subseteq S$;

3) for any distinct $S, T \in B_i$, $|S \cap T| < \min(|S|, |T|)$.

And the existence of such a basic vulnerable set is guaranteed by Proposition 2.

Proposition 2. For any $i \in V$, there exists a unique B_i.

The distribution of the sizes of the element sets of B_i is closely related to the perturbation strength on the features. When the perturbation is small, we may have to perturb multiple nodes before the target node is mis-classified, and thus the element sets of B_i will be large. When perturbation is relatively large, we may be able to turn a target node to be mis-classified by perturbing a single node, if chosen wisely. In this case B_i will have a lot of singleton sets.

Our following analysis (Proposition 3) shows that h has a diminishing-return effect if the vulnerable sets of nodes on the graph present *homophily* (Assumption 3.2.6), which is common in real-world networks, and the perturbation on features becomes considerably large (Assumption 3.2.7).

Assumption 3.2.6 (Homophily)**.** $\forall S \in \cup_{i=1}^{N} A_i$ *and* $|S| > 1$*, there are* $b(S) \geq 1$ *nodes s.t., for any node* j *among these nodes,* $S \in A_j$*.*

Intuitively, the vulnerable sets present strong homophily if $b(S)$'s are large.

Assumption 3.2.7 (Considerable Perturbation)**.** $\forall S \in \cup_{i=1}^{N} A_i$ *and if* $|S| > 1$*, then there are* $\lceil p(S) \cdot b(S) \rceil$ *nodes s.t., for any node* j *among these nodes, there exists a set* $T \subseteq S$*,* $|T| = 1$*, and* $T \in A_j$*. And* $\frac{r}{r+1} < p(S) \leq 1$*.*

Proposition 3. If Assumptions 3.2.6 and 3.2.7 hold, h is γ-approximately submodular for some $0 < \gamma < \frac{1}{r}$, i.e., there exists a non-decreasing submodular function $\tilde{h} : 2^V \to \mathbb{R}_+$, s.t. $\forall S \subseteq V$,

$$(1 - \gamma)\tilde{h}(S) \leq h(S) \leq (1 + \gamma)\tilde{h}(S).$$

As greedy methods are guaranteed to enjoy a constant approximation ratio for such approximately submodular functions [60], Proposition 3 motivates us to develop a greedy correction procedure to compensate the diminishing-return effect when calculating the importance scores.

The greedy correction procedure. We propose an iterative node selection procedure and apply two greedy correction steps on top of the RWCS strategy, motivated by Assumption 3.2.6 and 3.2.7.

To accommodate Assumption 3.2.6, after each node is selected into the attack set, we exclude a k-hop neighborhood of the selected node for next iteration, for a given constant integer k. The intuition is that nodes in a local neighborhood may contribute to similar target nodes due to homophily. To accommodate Assumption 3.2.7, we adopt an adaptive version of RWCS scores. First, we binarize the L-step random walk transition matrix M^L as $\widetilde{M}$, i.e.,

$$\left[\widetilde{M}\right]_{ij} = \begin{cases} 1, & \text{if } [M^L]_{ij} \text{ is among Top-}l \text{ of } [M^L]_i \text{ and } [M^L]_{ij} \neq 0, \\ 0, & \text{otherwise,} \end{cases} \tag{3.4}$$

where l is a given constant integer. Next, we define a new adaptive influence score as a function of a matrix Q: $\widetilde{I}_i(Q) = \sum_{j=1}^{N} [Q]_{ji}$. In the iterative node selection procedure, we initialize Q as $\widetilde{M}$. We select the node with highest score $\widetilde{I}_i(Q)$ subsequently. After each iteration, suppose we have selected the node i in this iteration, we will update Q by setting to zero for all the rows where the elements of the i-th column are 1. The underlying assumption of this operation is that, adding i to the selected set is likely to mis-classify all the target nodes corresponding to the aforementioned rows, which complies Assumption 3.2.7. We name this iterative procedure as the **GC-RWCS** (Greedily Corrected RWCS) strategy, and summarize it in Algorithm A.1 in Appendix A.3.

Finally, we want to mention that, while the derivation of RWCS and GC-RWCS requires the knowledge of the number of layers L for GCN, we find that the empirical performance of the proposed attack strategies are not sensitive w.r.t. the choice of L. Therefore, the proposed methods are applicable to the black-box setup where we do not know the exact L of the model.

3.3 Experiments

3.3.1 Experiment Setup

GNN models. We evaluate the proposed attack strategies on two common GNN models, GCN [77] and JK-Net [153]. For JK-Net, we test on its two variants, JKNet-Concat and JKNetMaxpool, which apply concatenation and element-wise max at last layer respectively. We set the number of layers for GCN as 2 and the number of layers for both JK-Concat and JK-Maxpool as 7. The hidden size of each layer is 32. For the training, we closely follow the hyper-parameter setup in Xu et al. [153].

Datasets. We adopt three citation networks, Citeseer, Cora, and Pubmed, which are standard node classification benchmark datasets [155]. Following the setup of JK-Net [153], we randomly split each dataset by 60%, 20%, and 20% for training, validation, and testing. And we draw 40 random splits.

Baseline methods for comparison. As we summarized in the preliminary chapter, our proposed black-box adversarial attack setup is by far the most restricted, and none of existing attack strategies for GNN can be applied. We compare the proposed attack strategies with baseline strategies by selecting nodes with top centrality metrics. We compare with three well-known network metrics capturing different aspects of node centrality: **Degree**, **Betweenness**, and **PageRank** and name the attack strategies correspondingly. In classical network analysis literature [107], real-world networks are shown to be fragile under attacks to high-centrality nodes. Therefore we believe these centrality metrics serve as reasonable baselines under our restricted black-box setup. For the purpose of sanity check, we also include a trivial baseline **Random**, which randomly selects the nodes to be attacked.

Hyper-parameters for GC-RWCS. For the proposed GC-RWCS strategy, we fix the number of step $L = 4$, the neighbor-hop parameter $k = 1$ and the parameter $l = 30$ for the binarized $\widetilde{M}$ in Eq. (3.4) for all models on all datasets. Note that $L = 4$

is different from the number of layers of both GCN and JK-Nets in our experiments. But we achieve effective attack performance. We also conduct a sensitivity analysis in Appendix A.5 and demonstrate the proposed method is not sensitive w.r.t. L.

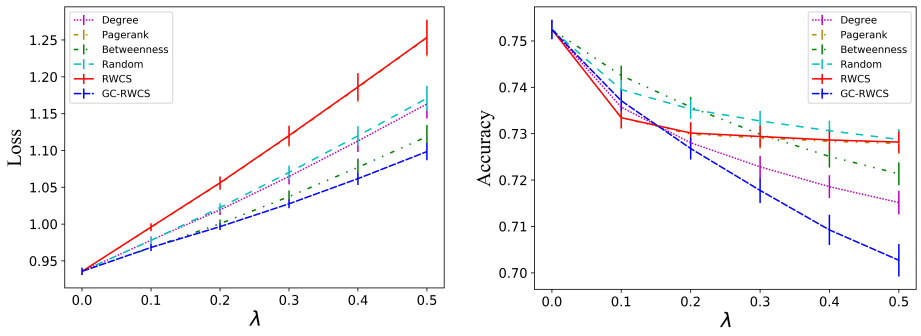
Nuisance parameters of the attack procedure. For each dataset, we fix the limit on the number of nodes to attack, r, as 1% of the graph size. After the node selection step, we also need to specify how to perturb the node features, i.e., the design of ϵ in τ function in the optimization problem (3.2). In a real-world scenario, ϵ should be designed with domain knowledge about the classification task, without access to the GNN models. In our experiments, we have to simulate the domain knowledge due to the lack of semantic meaning of each individual feature in the benchmark datasets. Formally, we construct the constant perturbation $\epsilon \in \mathbb{R}^D$ as follows, for $j = 1, 2, \ldots, D$,

$$\epsilon_j = \begin{cases} \lambda \cdot \text{sign}(\sum_{i=1}^{N} \frac{\partial \mathcal{L}(H,y)}{\partial X_{ij}}), & \text{if } j \in \arg\text{ top-}J \left(\left[\left| \sum_{i=1}^{N} \frac{\partial \mathcal{L}(H,y)}{\partial X_{il}} \right| \right]_{l=1,2,\ldots,D} \right), \\ 0, & \text{otherwise,} \end{cases} \tag{3.5}$$

where λ is the magnitude of modification. We fix $J = \lfloor 0.02D \rfloor$ for all datasets. While gradients of the model are involved, we emphasize that we only use extremely limited information of the gradients: determining a few number of important features and the binary direction to perturb for each selected feature, only at the *global level* by averaging gradients on all nodes. We believe such coarse information is usually available from domain knowledge about the classification task. The perturbation magnitude for each feature is fixed as a constant λ and is irrelevant to the model. In addition, the *same* perturbation vector is added to the features of *all* the selected nodes. The construction of the perturbation is totally independent of the selected nodes.

3.3.2 Experiment Results

Verifying the discrepancy between the loss and the mis-classification rate. We first provide empirical evidence for the discrepancy between classification loss (cross-entropy) and mis-classification rate. We compare the RWCS strategy to baseline strategies with varying perturbation strength as measured by λ in Eq. (3.5). The results shown in Figure III.1 are obtained by attacking GCN on Citeseer. First, we observe that RWCS increases the classification loss almost linearly as λ increases, indicating our approximation of the loss by first-order Taylor expansion actually works pretty well in practice. Not surprisingly, RWCS performs very similarly as PageRank. And RWCS performs much better than other centrality metrics in increasing the classification loss, showing the effectiveness of Proposition 1. However, we see the decrease of classification accuracy when attacked by RWCS (and PageRank) quickly saturates as λ increases. The GC-RWCS strategy that is proposed to correct the importance scores is able to decreases the classification accuracy the most as λ becomes larger, although it increases the classification loss the least.



(a) Loss on Test Set (b) Accuracy on Test Set

Figure III.1: Experiments of attacking GCN on Citeseer with increasing perturbation strength λ. Results are averaged over 40 random trials and error bars indicate standard error of mean.

Full experiment results. We then provide the full experiment results of attacking GCN, JKNetConcat, and JKNetMaxpool on all three datasets in Table 3.1. The perturbation strength is set as $\lambda = 1$. The thresholds 10% and 30% indicate that we

set the limit on the maximum degree m as the lowest degree of the top 10% and 30% nodes respectively.

The results clearly demonstrate the effectiveness of the proposed GC-RWCS strategy. GC-RWCS achieves the best attack performance on almost all experiment settings, and the difference to the second-best strategy is significant in almost all cases. It is also worth noting that the proposed GC-RWCS strategy is able to decrease the node classification accuracy by up to 33.5%, and GC-RWCS achieves a 70% larger decrease of the accuracy than the Random baseline in most cases (see Table A.3 in Appendix A.5). And this is achieved by merely adding the same constant perturbation vector to the features of 1% of the nodes in the graph. This verifies that the explicit structural inductive biases of GNN models make them vulnerable even in the extremely restricted black-box attack setup.

Table 3.1: Summary of the attack performance. The lower the accuracy (in %) the better the attacks. The **bold** marker denotes the best performance. The asterisk (*) means the difference between the best strategy and the second-best strategy is statistically significant by a t-test at significance level 0.05. The error bar (±) denotes the standard error of the mean by 40 independent trials.

	Cora			Citeseer			Pubmed		
Method	GCN	JKNetConcat	JKNetMaxpool	GCN	JKNetConcat	JKNetMaxpool	GCN	JKNetConcat	JKNetMaxpool
None	85.6 ± 0.3	86.2 ± 0.2	85.8 ± 0.3	75.1 ± 0.2	72.9 ± 0.3	73.2 ± 0.3	85.7 ± 0.1	85.8 ± 0.1	85.7 ± 0.1
					Threshold 10%				
Random	81.3 ± 0.3	68.8 ± 0.8	68.8 ± 1.3	71.3 ± 0.3	60.8 ± 0.8	61.7 ± 0.9	82.0 ± 0.3	75.9 ± 0.7	75.4 ± 0.7
Degree	**78.2** ± 0.4	60.7 ± 1.0	59.9 ± 1.5	67.5 ± 0.4	52.5 ± 0.8	53.7 ± 1.0	78.9 ± 0.5	63.4 ± 1.0	63.3 ± 1.2
Pagerank	79.4 ± 0.4	71.6 ± 0.6	70.0 ± 1.0	70.1 ± 0.3	61.5 ± 0.5	62.6 ± 0.6	80.3 ± 0.3	71.3 ± 0.8	71.2 ± 0.8
Betweenness	79.7 ± 0.4	60.5 ± 0.9	60.3 ± 1.6	68.9 ± 0.3	53.5 ± 0.8	55.1 ± 1.0	78.5 ± 0.6	67.1 ± 1.1	66.2 ± 1.1
RWCS	79.5 ± 0.3	71.2 ± 0.5	69.9 ± 1.0	69.9 ± 0.3	60.8 ± 0.6	62.2 ± 0.7	79.8 ± 0.3	70.7 ± 0.8	70.7 ± 0.8
GC-RWCS	78.5 ± 0.5	**52.7** ± 1.0*	**53.3** ± 1.9*	**65.1** ± 0.5*	**46.6** ± 0.8*	**48.2** ± 1.1*	**77.3** ± 0.7	**62.1** ± 1.2	**60.6** ± 1.4*
					Threshold 30%				
Random	82.6 ± 0.4	70.7 ± 1.1	71.8 ± 1.1	72.6 ± 0.3	62.7 ± 0.8	63.9 ± 0.8	82.6 ± 0.2	77.3 ± 0.4	77.4 ± 0.5
Degree	**80.7** ± 0.4	64.9 ± 1.4	67.0 ± 1.5	70.4 ± 0.4	56.9 ± 0.8	58.7 ± 0.9	81.5 ± 0.4	72.4 ± 0.7	72.3 ± 0.7
Pagerank	82.6 ± 0.3	79.6 ± 0.4	79.7 ± 0.4	72.9 ± 0.2	70.2 ± 0.3	70.3 ± 0.3	83.0 ± 0.2	79.3 ± 0.3	79.6 ± 0.3
Betweenness	81.8 ± 0.4	64.1 ± 1.3	65.9 ± 1.4	70.7 ± 0.3	56.3 ± 0.8	58.3 ± 0.9	81.3 ± 0.3	74.1 ± 0.5	74.6 ± 0.5
RWCS	82.8 ± 0.3	79.3 ± 0.5	79.5 ± 0.4	72.9 ± 0.2	69.8 ± 0.3	70.1 ± 0.3	82.1 ± 0.2	77.8 ± 0.3	78.4 ± 0.3
GC-RWCS	**80.7** ± 0.5	**59.1** ± 1.6*	**61.1** ± 1.6*	**67.8** ± 0.5*	**49.0** ± 0.9*	**50.7** ± 1.1*	**80.3** ± 0.5*	**69.2** ± 0.7*	**70.0** ± 0.7*

3.4 Conclusion

In this chapter, we propose a novel black-box adversarial attack setup for GNN models with constraint of limited node access, which we believe is by far the most restricted and realistic black-box attack setup. Nonetheless, through both theoretical analyses and empirical experiments, we demonstrate that the strong and explicit

structural inductive biases of GNN models make them still vulnerable to this type of adversarial attacks. We also propose a principled attack strategy, GC-RWCS, based on our theoretical analyses on the connection between the GCN model and random walk, which corrects the diminishing-return effect of the mis-classification rate. Our experimental results show that the proposed strategy significantly outperforms competing attack strategies under the same setup.

CHAPTER IV

The Representational and Correlational Roles of Graphs in Graph Neural Networks

4.1 Introduction

Graphs, as flexible data representations that store rich relational information, have been commonly used in data science tasks. Machine learning methods on graphs [23], especially Graph Neural Networks (GNNs), have attracted increasing interest in the research community. They are widely applied to real-world problems such as recommender systems [157], social network analysis [85], and transportation forecasting [158]. Among the heterogeneous types of graph-structured data, it is worth noting that graphs usually play diverse roles in different contexts, different datasets, and different tasks. Some of the roles are relational, as a graph may indicate certain statistical relationships of connected observations; some are representational, as the topological structure of a graph may encode important features/patterns of the data; some are even causal, as a graph may reflect causal relationships specified by domain experts.

It is crucial to recognize the distinct roles of a graph in order to correctly utilize the signals in the graph-structured data. In this chapter, we distinguish the *representational role* and the *correlational role* of graphs in the context of node-level

(semi-)supervised learning, and we investigate how to design better GNNs that take advantage of both roles.

In a node-level prediction task, the observed graph in the data may relate to the outcomes of interest (e.g., node labels) in multiple ways. Conceptually, we call that the graph plays a representational role if one can leverage it to construct better feature representations. For example, in social network analysis, aggregating user features from one's friends is usually helpful (thanks to the well-known homophily phenomenon [104]). In addition, the structural properties of a user's local network, e.g., structural diversity [140] and structural holes [17, 93], often provide useful information for making predictions about certain outcomes of that user. On the other hand, sometimes a graph directly encodes correlations between the outcomes of connected nodes, and we call it playing a *correlational role*. For example, hyper-linked Webpages are likely to be visited together even if they have dissimilar content. In spatiotemporal predictions, the outcome of nearby locations, conditional on all the features, may still be correlated.

While both the representational and the correlational roles are common in graph-structured data, we find that, through a simulation study, many existing GNN models are incapable of utilizing the correlational information encoded in a graph. Specifically, we design a synthetic dataset for the node-level regression. The node-level outcomes are drawn from a multivariate normal distribution, with the mean and the covariance as functions of the graph to reflect the representational and correlation roles respectively. We find that when the graph only provides correlational information of the node outcomes, many popular GNN models underperform a multi-layer perceptron which does not consider the graph at all.

To mitigate this deficiency of GNNs, we propose a principled solution, the Copula Graph Neural Network (CopulaGNN), which can take a wide range of GNNs as the base model and improve their capabilities of modeling the correlational graph infor-

mation. The key insight of the proposed method is that, by decomposing the joint distribution of node outcomes into the product of marginal densities and a copula density, the representational information and correlational information can be separately modeled. The former is modeled by the marginal densities through a base GNN while the latter is modeled by a Gaussian copula. The proposed method also enjoys the benefit of easy extension to various types of node outcome variables including continuous variables, discrete count variables, or even mixed-type variables. We instantiate CopulaGNN with normal and Poisson marginal distributions for continuous and count regression tasks respectively. We also implement two types of copula parameterizations combined with two types of base GNNs.

We evaluate the proposed method on both synthetic and real-world data with both continuous and count regression tasks. The experimental results show that CopulaGNNs significantly outperform their base GNN counterparts when the graph in the data exhibits both correlational and representational roles. We summarize our main contributions as follows:

1. We raise the question of distinguishing the two roles played by the graph and demonstrate that many existing GNNs are incapable of utilizing the graph information when it plays a pure correlational role.
2. We propose a principled solution, the CopulaGNN, to integrate the representational and correlational roles of the graph.
3. We empirically demonstrate the effectiveness of CopulaGNN compared to base GNNs on semi-supervised regression tasks.

4.2 Related Work

There have been extensive existing works that model either the representational role or the correlational role of the graph in node-level (semi-)supervised learning tasks. However, there are fewer methods that try to model both sides simultaneously,

especially with a GNN.

Methods focusing on the representational role. As we mentioned in Section 4.1, the graph can help construct better node feature representations by both providing extra topological information and guiding node feature aggregation. There have been vast existing studies on both directions, and among them we can only list a couple of examples. Various methods have been proposed to leverage the topological information of graph-structured data in machine learning tasks, such as graph kernels [143], node embeddings [115, 138, 54], and GNNs [152]. Aggregating node features on an attributed graph has also been widely studied, e.g., through feature smoothing [105] or GNNs [77, 57]. In this work, we restrict our focus on the GNN models, which have been the state-of-the-art graph representation learning method on various tasks.

Methods focusing on the correlational role. On the other hand, there has also been extensive literature on modeling the dependence of variables on connected nodes in a graph. One group of methods is called the graph-based regularization [166, 87], where it is assumed that the variables associated with linked objects change smoothly and pose an explicit similarity regularization among them. The correlational role of the graph is also closely related to undirected graphical models [82, 72, 144]. In graphical models, the edges in a graph provide a representation of the conditional (in)dependence structure among a set of random variables, which are represented by the node set of the graph. Finally, there has been a line of research that combines graphical models with copulas and leads to more flexible model families [44, 40, 91, 7]. Our proposed method integrates the benefits of copulas and GNNs to capture both the representational and correlational roles.

Methods improving GNNs by leveraging the correlational graph information. A few methods explicitly leverage the correlational graph information to improve the GNN training, but most of them focus on a classification setting [117, 96].

A recent study [67] that we have been aware of only lately shares a similar motivation to ours, yet our methodology differs significantly. In particular, Jia and Benson [67] apply a multivariate normal distribution to model the correlation of node outcomes, which can be viewed as a special case of our proposed CopulaGNN when a Gaussian copula with normal marginals is used. Our method not only generalizes to other marginals (we show the effectiveness of some of them), but also has a more flexible parameterization on the correlation matrix of the copula distribution. In addition, we differ with these previous works by explicitly distinguishing the two roles of the graph in the data.

Spatial autoregressive models for network data. There is another line of literature in statistics that explicitly describes the correlation among the responses associated with nodes in a network [84, 167, 64]. They applied the spatial autoregressive (SAR) models [30] to network data. In particular, the response of each node is linearly associated with the responses of its neighbor nodes and exogenous covariates. Therefore, given the node covariates and the network, the correlation among responses is captured by a network autocorrelation coefficient, which is often assumed to the same for all nodes. Recently, [102] proposed a semiparametric autoregressive method that allows the spatial dependence to vary across nodes. The SAR models can be thought as alternatives to the copula model used in this work. It is an interesting future direction to consider leveraging SAR models to improve GNNs similarly as the CopulaGNN framework proposed in this work.

4.3 Simulating the Two Roles of the Graph

In this section, we investigate, through a simulation study, the representational and correlational roles of the graph in the context of node-level semi-supervised learning.

4.3.1 Node-Level Semi-Supervised Learning

We start by formally introducing the problem of node-level semi-supervised learning. A graph is a tuple: $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, 2, \ldots, n\}$ is the set of n nodes; $\mathcal{E} \in \mathcal{V} \times \mathcal{V}$ is the set of edges. The graph is also associated with $\boldsymbol{X} \in \mathbb{R}^{n \times d}$ and $\boldsymbol{y} \in \mathbb{R}^n$, which are the node features and outcome labels. In the semi-supervised learning setting, we only observe the labels of $0 < m < n$ nodes. Without loss of generality, we assume the labels of nodes $\{1, 2, \ldots, m\}$ are observed and those of $\{m+1, \ldots, n\}$ are missing. Therefore, the label vector $\boldsymbol{y}$ can be partitioned as $\boldsymbol{y} = (\boldsymbol{y}_{\text{obs}}^T, \boldsymbol{y}_{\text{miss}}^T)^T$. The goal of a node-level semi-supervised learning task is to infer $\boldsymbol{y}_{\text{miss}}$ based on $(\boldsymbol{y}_{\text{obs}}, \boldsymbol{X}, \mathcal{G})$.

4.3.2 Synthetic Data

To simulate the representational and correlational roles of the graph, we first design a synthetic dataset by specifying the joint distribution of $\boldsymbol{y}$ conditional on $\boldsymbol{X}$ and $\mathcal{G}$. In particular, we let the joint distribution of the node outcomes take the form of $\boldsymbol{y}|\boldsymbol{X}, \mathcal{G} \sim \mathcal{N}(\boldsymbol{\mu}(\boldsymbol{X}, \mathcal{G}), \boldsymbol{\Sigma}(\mathcal{G}))$, for some $\boldsymbol{\mu}, \boldsymbol{\Sigma}$ to be specified. In this way, the graph $\mathcal{G}$ plays a representational role through $\boldsymbol{\mu}(\boldsymbol{X}, \mathcal{G})$ and a correlational role through $\boldsymbol{\Sigma}(\mathcal{G})$.

Specifically, we generate synthetic node-level regression data on a graph with n nodes and m edges (see Appendix B.1.1 for the whole procedure). We first randomly generate a feature matrix $\boldsymbol{X} \in \mathbb{R}^{n \times d_0}$. Assume $\boldsymbol{A}$ is the adjacency matrix of the graph, $\boldsymbol{D}$ is the degree matrix, and $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}$ is the graph Laplacian. Let $\tilde{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{I}$ and $\tilde{\boldsymbol{D}} = \boldsymbol{D} + \boldsymbol{I}$. Given parameters $\boldsymbol{w}_y \in \mathbb{R}^{d_0}$, we generate the node label vector $\boldsymbol{y} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where, for some $\gamma > 0, \tau > 0$, and $\sigma^2 > 0$,

(a) $\boldsymbol{\mu} = \tilde{\boldsymbol{D}}^{-1}\tilde{\boldsymbol{A}}\boldsymbol{X}\boldsymbol{w}_y$, $\boldsymbol{\Sigma} = \sigma^2\boldsymbol{I}$;

(b) $\boldsymbol{\mu} = \boldsymbol{X}\boldsymbol{w}_y$, $\boldsymbol{\Sigma} = \tau(\boldsymbol{L} + \gamma\boldsymbol{I})^{-1}$;

(c) $\boldsymbol{\mu} = \tilde{\boldsymbol{D}}^{-1}\tilde{\boldsymbol{A}}\boldsymbol{X}\boldsymbol{w}_y$, $\boldsymbol{\Sigma} = \tau(\boldsymbol{L} + \gamma\boldsymbol{I})^{-1}$.

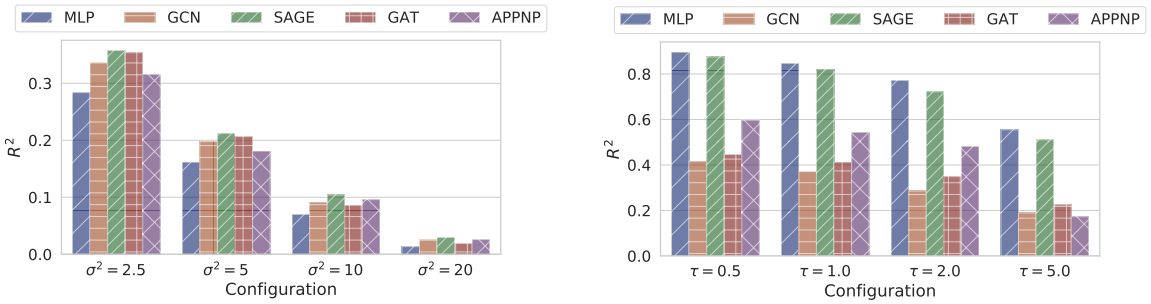
Depending on how $(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ are configured, we get three types of synthetic data settings: (a), (b), and (c). Intuitively, the graph plays a pure representational role in setting (a) since the label of a node depends on the aggregated features of its local neighborhood and the node labels are independent conditional on the node features. In setting (b), the graph plays a pure correlational role; while the means of node labels only depend on their own node features, the node labels are still correlated conditional on the features, and the correlation is determined by the graph structure. Finally, setting (c) is a combination of (a) and (b) where the graph plays both representational and correlational roles.

In the rest of this section, we test the performance of a few widely used GNNs under setting (a) and (b) to examine their capabilities of utilizing the representational and correlational information. We defer the experimental results under setting (c) to Section 4.5.2 for ease of reading.

4.3.3 Simulation Study

Simulation Setup. We set $n = 300, m = 5000$, and $d_0 = 10$. Elements of both $\boldsymbol{W}_g$ and $\boldsymbol{w}_y$ are generated from i.i.d. standard normal distribution. For setting (a), we vary $\sigma^2 \in \{2.5, 5, 10, 20\}$. For settings (b) and (c), we set $\gamma = 0.1$ and vary $\tau \in \{0.5, 1, 2, 5\}$. We test 4 common GNN models, GCN [77], GraphSAGE [57] (denoted as SAGE), GAT [141], and APPNP [79], as well as the multi-layer perceptron (MLP).

Simulation Results. First, we observe that all 4 types of GNNs outperform MLP under setting (a) (Figure IV.1a), where the graph plays a pure representational role. This is not surprising as the architectures of the GNNs encode a similar feature aggregation structure as the data. However, under setting (b) (Figure IV.1b) where the graph plays a pure correlational role, all 4 types of GNNs underperform MLP. This suggests that a majority of popular GNN models might be incapable of fully



(a) $\boldsymbol{\mu} = \tilde{\boldsymbol{D}}^{-1}\tilde{\boldsymbol{A}}\boldsymbol{X}\boldsymbol{w}_y$, $\boldsymbol{\Sigma} = \sigma^2\boldsymbol{I}$. (b) $\boldsymbol{\mu} = \boldsymbol{X}\boldsymbol{w}_y$, $\boldsymbol{\Sigma} = \tau(\boldsymbol{L} + \gamma\boldsymbol{I})^{-1}$.

Figure IV.1: The coefficient of determination R^2 (the higher the better) of GNNs and MLP when the graph plays (a) the representational role or (b) the correlational role. For each configuration, the results are aggregated from 100 trials. In (a), all GNNs outperform MLP; in (b), all GNNs underperform MLP.

utilizing the correlational graph information.

Motivated by our findings in the simulation study, in the following section, we seek for methods that augment existing GNN models in order to better utilize both representational and correlational information in the graph.

4.4 Copula Graph Neural Network

In this section, we propose a principled solution called the Copula Graph Neural Network (CopulaGNN). At the core of our method is the application of copulas, which are widely used for modeling multivariate dependence. In the rest of this section, we first provide a brief introduction to copulas (more detailed expositions can be found in the monographs by Joe [71] and Czado [33]), then present the proposed CopulaGNN and its parameterization, learning, and inference.

4.4.1 Introduction to Copulas

General formulation. Sklar's theorem [133] states that any multivariate joint distribution F of a random vector $\boldsymbol{Y} = (Y_1, \ldots, Y_n)$ can be written in terms of one-dimensional marginal distributions $F_i(y) = \mathbb{P}(Y_i \leq y)$ and a copula $C : [0,1]^n \to [0,1]$ that describes the dependence structures among variables: $F(y_1, \ldots, y_n) =$

$C(F_1(y_1), \ldots, F_p(y_n))$. In other words, one can decompose a joint distribution into two components: the marginals and the copula. Furthermore, a copula C can also be regarded as the Cumulative Distribution Function (CDF) of a corresponding distribution on the unit hypercube $[0,1]^n$. Its copula density is denoted by $c(u_1, \ldots, u_n) := \partial^n C(u_1, u_2, \ldots, u_n)/\partial u_1 \cdots \partial u_n$. The Probability Density Function (PDF) of a random vector can be represented by its corresponding copula density. If the random vector $\boldsymbol{Y}$ is continuous, its PDF can be written as

$$f(\boldsymbol{y}) = c(u_1, \ldots, u_n) \prod_{i=1}^{n} f_i(y_i), \tag{4.1}$$

where f_i is the PDF of Y_i, $u_i = F_i(y_i)$, and c is the copula density. For discrete random vectors, the form of the Probability Mass Function (PMF) is more complex. See Appendix B.2.2 for details.

Gaussian copula. One of the most popular copula family is the *Gaussian copula*. When the joint distribution F is multivariate normal with a mean of $\mathbf{0}$ and a covariance matrix of $\boldsymbol{\Sigma}$, the corresponding copula is the Gaussian copula:

$$C(u_1, u_2, \cdots, u_n; \boldsymbol{\Sigma}) = \Phi_n(\Phi^{-1}(u_1), \cdots, \Phi^{-1}(u_n); \mathbf{0}, \boldsymbol{R}),$$

where $\Phi_n(\cdot; \mathbf{0}, \boldsymbol{R})$ is the multivariate normal CDF, $\boldsymbol{R}$ is the correlation matrix of $\boldsymbol{\Sigma}$, and $\Phi^{-1}(\cdot)$ is the quantile function of the univariate standard normal distribution. Its copula density is

$$c(u_1, u_2, \ldots, u_n; \boldsymbol{\Sigma}) = (\det \boldsymbol{R})^{-1/2} \exp\left(-\frac{1}{2}\Phi^{-1}(\boldsymbol{u})^T(\boldsymbol{R}^{-1} - \boldsymbol{I}_n)\Phi^{-1}(\boldsymbol{u})\right),$$

where $\boldsymbol{I}_n$ is the identity matrix of size n and $\boldsymbol{u} = (u_1, u_2, \ldots, u_n)$.

4.4.2 The Proposed Model

Recall that our goal is to model both representational and correlational graph information in the conditional joint distribution of the node outcomes,

$$f(\boldsymbol{y}; \boldsymbol{X}, \mathcal{G}) = c(u_1, \ldots, u_n; \boldsymbol{X}, \mathcal{G}) \prod_{i=1}^{n} f_i(y_i; \boldsymbol{X}, \mathcal{G}), \tag{4.2}$$

which can be decomposed into the copula density and marginal densities. In this formulation, the representational information and correlational information are naturally separated into the marginal densities f_i, for $i = 1, \ldots, n$, and the copula density c, respectively. Note that both the marginal densities and the copula density are conditional on the node features $\boldsymbol{X}$ and the graph $\mathcal{G}$. Next, we need to choose a proper distribution family for each of these densities and parameterize the distribution parameters as functions of $\boldsymbol{X}$ and $\mathcal{G}$.

Choice of distribution family and parameterization for the copula density. For the distribution family, we choose the Gaussian copula as the copula family, $c(u_1, \ldots, u_n; \boldsymbol{\Sigma}(\boldsymbol{X}, \mathcal{G}; \boldsymbol{\theta}))$, where the form of $\boldsymbol{\Sigma}(\cdot; \boldsymbol{\theta})$ and the learnable parameters $\boldsymbol{\theta}$ remain to be specified. To leverage the correlational graph information, we draw a connection between the graph structure $\mathcal{G}$ and the covariance matrix $\boldsymbol{\Sigma}$ in the Gaussian copula density. Let $\boldsymbol{K} = \boldsymbol{\Sigma}^{-1}$ be the precision matrix; if two nodes i and j are not linked in the graph, we constrain the corresponding (i, j)-th entry in $\boldsymbol{K}$ to be 0. In other words, the absence of an edge between nodes i and j leads to their outcome variables y_i and y_j being conditionally independent given all other variables. The motivation of parameterizing the precision matrix $\boldsymbol{K}$ instead of the covariance matrix $\boldsymbol{\Sigma}$ is closely related to undirected graphical models [82, 72, 144], where the conditional dependence structure among a set of variables is fully represented by edges in an underlying graph. In our use case, we could view our assumption on $\boldsymbol{K}$ as a graphical model among random variables $(y_1, \ldots, y_n)$, where the underlying

graph structure is known.

The conditional independence assumption has significantly reduced the number of non-zero entries in $\boldsymbol{K}$ to be estimated. However, without any further constraints, there are still $|\mathcal{E}|$ free parameters growing with the graph size, which can hardly be estimated accurately given only one observation of $(y_1, \ldots, y_m)$. In practice, we consider two ways of parametrizing $\boldsymbol{K}$ with fewer parameters.

Two-parameter parametrization. A rather strong but simple constraint is to assume the non-zero off-diagonal entries of $\boldsymbol{K}$ have the same value or they are proportional to the corresponding entries in the normalized adjacency matrix, and introduce two global parameters controlling the overall strength of correlation. For example, we could have $\boldsymbol{K} = \tau^{-1}(\boldsymbol{L} + \gamma \boldsymbol{I})$ as what we did in the simulation study in Section 4.3, or $\boldsymbol{K} = \beta(\boldsymbol{I}_n - \alpha \boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2})$ as used in Jia and Benson [67], where (τ, γ) or (α, β) are learnable parameters.

Regression-based parametrization. We further propose a more flexible parameterization that allows the non-zero entries in $\boldsymbol{K}$ to be estimated by a regressor taking node features as inputs. In pariticular, for any (i, j)-pair corresponding to a non-zero entry of $\boldsymbol{K}$, we set $\widehat{\boldsymbol{A}}_{i,j} = \text{softplus}(h(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{\theta}))$, where h is a two-layer MLP that takes the concatenation of $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ as input and outputs a scalar. Let $\widehat{\boldsymbol{D}}$ be the degree matrix if we treat $\widehat{\boldsymbol{A}}$ as a weighted adjacency matrix, and we set the precision matrix $\boldsymbol{K} = \boldsymbol{I}_n + \widehat{\boldsymbol{D}} - \widehat{\boldsymbol{A}}$. This parameterization improves the flexibility on estimating $\boldsymbol{K}$ while keeping the number of learnable parameters $\boldsymbol{\theta}$ independent of the graph size n. It also ensures that $\boldsymbol{K}$ is positive-definite and thus invertible.

Choice of distribution families and parameterization for the marginal densities. One benefit of the copula framework is the flexibility on the choice of distribution families for the marginal densities. In this work, we choose the marginal densities to be normal distributions if the labels $\boldsymbol{y}$ are continuous variables, and we choose them to be Poisson distributions if the $\boldsymbol{y}$ are discrete. We denote the

i-th marginal density function by $f_i(y_i; \boldsymbol{\eta}_i(\boldsymbol{X}, \mathcal{G}; \boldsymbol{\theta}))$ and the corresponding CDF by $F_i(y_i; \boldsymbol{\eta}_i(\boldsymbol{X}, \mathcal{G}; \boldsymbol{\theta}))$, where $\boldsymbol{\eta}_i(\cdot; \boldsymbol{\theta})$ denotes the distribution parameters to be specified.

If the i-th marginal distribution takes the form of a normal distribution $\mathcal{N}(\mu_i, \sigma_i^2)$, then $\boldsymbol{\eta}_i(\boldsymbol{X}, \mathcal{G}; \boldsymbol{\theta}) = (\mu_i(\boldsymbol{X}, \mathcal{G}; \boldsymbol{\theta}), \sigma_i^2(\boldsymbol{X}, \mathcal{G}; \boldsymbol{\theta}))$. We define $\mu_i(\boldsymbol{X}, \mathcal{G}; \boldsymbol{\theta})$ as the output of a base GNN model for node i, and $\sigma_i^2(\boldsymbol{X}, \mathcal{G}; \boldsymbol{\theta})$ as the i-th diagnoal element of the covariance matrix $\boldsymbol{\Sigma}(\boldsymbol{X}, \mathcal{G}; \boldsymbol{\theta})$ as we specified in the Gaussian copula. If the i-th marginal distribution takes the form of a Poisson distribution $\text{Pois}(\lambda_i)$, then $\boldsymbol{\eta}_i(\boldsymbol{X}, \mathcal{G}; \boldsymbol{\theta}) = \lambda_i(\boldsymbol{X}, \mathcal{G}; \boldsymbol{\theta})$, and we define $\lambda_i(\boldsymbol{X}, \mathcal{G}; \boldsymbol{\theta})$ as the output of a base GNN model for node i. Either way, the representational role of the graph is reflected in the location parameters (μ_i or λ_i) computed by a base GNN model. In practice, we can also choose other distribution families such as the log-normal or negative binomial, depending on our belief on the true distributions of the node outcomes. One can even choose different distribution families for different nodes simultaneously if necessary.

4.4.3 Model Learning and Inference

For simplicity of notation, we write $\boldsymbol{\eta}_i(\boldsymbol{X}, \mathcal{G}; \boldsymbol{\theta})$ and $\boldsymbol{\Sigma}(\boldsymbol{X}, \mathcal{G}; \boldsymbol{\theta})$ as $\boldsymbol{\eta}_i$ and $\boldsymbol{\Sigma}$ throughout this section.

Model learning. The model parameters $\boldsymbol{\theta}$ are learned by maximizing the log-likelihood on the observed node labels. Given the partition of $\boldsymbol{y}$, we can further partition the covariance matrix $\boldsymbol{\Sigma}$ accordingly:

$$\boldsymbol{y} = \begin{pmatrix} \boldsymbol{y}_{\text{obs}} \\ \boldsymbol{y}_{\text{miss}} \end{pmatrix} \text{ and } \boldsymbol{\Sigma} = \begin{pmatrix} \boldsymbol{\Sigma}_{00} & \boldsymbol{\Sigma}_{01} \\ \boldsymbol{\Sigma}_{10} & \boldsymbol{\Sigma}_{11} \end{pmatrix},$$

where $\boldsymbol{y}_{\text{obs}} = (y_1, \ldots, y_m)$ and $\boldsymbol{y}_{\text{miss}} = (y_{m+1}, \ldots, y_n)$. In other words, $\boldsymbol{\Sigma}_{00}$ and $\boldsymbol{\Sigma}_{11}$ are the covariance matrices of the observed and missing nodes. We further denote $u_i = F_i(y_i; \boldsymbol{\eta}_i)$ for $i = 1, \ldots, n$, $\boldsymbol{u}_{\text{obs}} = (u_1, \ldots, u_m)$, and $\boldsymbol{u}_{\text{miss}} = (u_{m+1}, \ldots, u_n)$; that is, u_i is the probability integral transform of the i-th label y_i. According to

Equation 4.1, the joint density can be written as the product of the copula density and the marginal densities. Therefore, the loss function is

$$\mathcal{L}(\boldsymbol{\theta}) = \log f(\boldsymbol{y}_{\text{obs}}; \boldsymbol{X}, \mathcal{G}) = \log c(\boldsymbol{u}_{\text{obs}}; \boldsymbol{\Sigma}_{00}) + \sum_{i=1}^{m} \log f_i(y_i; \boldsymbol{\eta}_i).$$

The parameters $\boldsymbol{\theta}$ are learned end-to-end using standard optimization algorithms such as Adam [76].

Model inference. At inference time, we are interested in the conditional distribution $f(\boldsymbol{y}_{\text{miss}}|\boldsymbol{y}_{\text{obs}}; \boldsymbol{X}, \mathcal{G})$. The inference of the conditional distribution can be done via sampling. Since $f(\boldsymbol{y}; \boldsymbol{X}, \mathcal{G})$ is modeled by the Gaussian copula, we have

$$\begin{pmatrix} \Phi^{-1}(\boldsymbol{u}_{\text{obs}}) \\ \Phi^{-1}(\boldsymbol{u}_{\text{miss}}) \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \mathbf{0} \\ \mathbf{0} \end{pmatrix}, \begin{pmatrix} \boldsymbol{R}_{00} & \boldsymbol{R}_{01} \\ \boldsymbol{R}_{10} & \boldsymbol{R}_{11} \end{pmatrix} \right),$$

where $\boldsymbol{R}$ is the correlation matrix corresponding to the covariance matrix $\boldsymbol{\Sigma}$. By the property of the multivatiate normal distribution, the conditional distribution of $\Phi^{-1}(\boldsymbol{u}_{\text{miss}})|\Phi^{-1}(\boldsymbol{u}_{\text{obs}})$ is also multivatiate normal:

$$\Phi^{-1}(\boldsymbol{u}_{\text{miss}})|\Phi^{-1}(\boldsymbol{u}_{\text{obs}}) \sim \mathcal{N}\left(\boldsymbol{R}_{10}\boldsymbol{R}_{00}^{-1}\Phi^{-1}(\boldsymbol{u}_{\text{obs}}), \boldsymbol{R}_{11} - \boldsymbol{R}_{10}\boldsymbol{R}_{00}^{-1}\boldsymbol{R}_{01}\right). \tag{4.3}$$

This provides a way to draw samples from $f(\boldsymbol{y}_{\text{miss}}|\boldsymbol{y}_{\text{obs}}; \boldsymbol{X}, \mathcal{G})$ as follows: (i) compute $\boldsymbol{u}_{\text{obs}}$ and $\boldsymbol{R}$ from $(\boldsymbol{y}_{\text{obs}}, \boldsymbol{X}, \mathcal{G})$, as described in Section 4.4.2; (ii) draw random samples $\Phi^{-1}(\boldsymbol{u}_{\text{miss}})$ from the conditional distribution in Equation 4.3; (iii) apply $\Phi(\cdot)$ elementwise to get $\boldsymbol{u}_{\text{miss}}$; and (iv) apply $F_i^{-1}(\cdot; \boldsymbol{\eta}_i)$ elementwise to get $\boldsymbol{y}_{\text{miss}}$. The average of the samples can be used as the point estimate.

4.5 Experiments

4.5.1 General Setup

We instantiate CopulaGNN with either GCN or GraphSAGE as the base GNN models, and implement both the two-parameter parameterization (the (α, β) parameterization, denoted by "$\alpha\beta$-C-", where "C" stands for copula) and the regression-based parameterization (denoted by "R-C-"). In combination, we have four variants of CopulaGNN: $\alpha\beta$-**C-GCN**, **R-C-GCN**, $\alpha\beta$-**C-SAGE**, and **R-C-SAGE**. When the outcome is a continuous variable, the normal margin is used; and when the outcome is a count variable, the Poisson margin is used. In particular, in the former case, the $\alpha\beta$-C-GNN degenerates to the Correlation GNN proposed by Jia and Benson [67]. We compare different variants of CopulaGNN with their base GNN counterparts, as well as an MLP model, on two types of regression tasks: continuous outcome variables and count outcome variables. More experiment details can be found in Appendix C.3.

4.5.2 Regression with Continuous Outcome Variables

We use two groups of datasets with continuous outcome variables. The first group is the synthetic data of setting (c) as described in Sections 4.3.2 and 4.3.3, where a graph provides both representational and correlational information. The second group includes four regression tasks constructed from the U.S. Election data [67]. We use the coefficient of determination R^2 to measure the model performance.

Results. For the synthetic datasets (Table 4.1), we vary the value of τ, which controls the overall magnitude of the label covariance. Unsurprisingly, as τ increases, the labels become noisier and the test R^2 of all models decreases. In all configurations, R-C-GCN and R-C-SAGE respectively outperform their base model counterparts, GCN and SAGE, by significant margins. This verifies the effectiveness of the proposed method when the graph provides both representational and correlational information.

Table 4.1: Experiment results on the synthetic data under setting (c) as described in Sections 4.3.2 and 4.3.3. The average test R^2 from 100 trials is reported (the larger the better). The asterisk markers, *, **, and ***, indicate the difference between a variant of CopulaGNN and its GNN base model is statistically significant by a pairwise t-test at significance levels of 0.1, 0.05, and 0.01, respectively. The ($\pm$) error bar denotes the standard error of the mean.

	$\tau = 0.5$	$\tau = 1.0$	$\tau = 2.0$	$\tau = 5.0$
MLP	0.624 $\pm$ 0.011	0.549 $\pm$ 0.014	0.437 $\pm$ 0.018	0.193 $\pm$ 0.020
GCN	0.673 $\pm$ 0.022	0.563 $\pm$ 0.034	0.384 $\pm$ 0.055	0.174 $\pm$ 0.032
$\alpha\beta$-C-GCN	0.669 $\pm$ 0.024	0.568 $\pm$ 0.033	0.408 $\pm$ 0.050*	0.200 $\pm$ 0.035
R-C-GCN	0.706 $\pm$ 0.017***	0.617 $\pm$ 0.023***	0.489 $\pm$ 0.034***	0.217 $\pm$ 0.029*
SAGE	0.733 $\pm$ 0.013	0.644 $\pm$ 0.020	0.507 $\pm$ 0.030	0.262 $\pm$ 0.025
$\alpha\beta$-C-SAGE	0.741 $\pm$ 0.013**	0.650 $\pm$ 0.019	0.518 $\pm$ 0.029*	0.281 $\pm$ 0.024**
R-C-SAGE	0.754 $\pm$ 0.010***	0.665 $\pm$ 0.017**	0.540 $\pm$ 0.024**	0.290 $\pm$ 0.022*

Another interesting observation is that GCN outperforms MLP when τ is small (0.5 and 1.0), but underperforms MLP when τ becomes large (2.0 and 5.0), whereas R-C-GCN consistently outperforms MLP. Note that τ can also be viewed as the tradeoff between the representational role and the correlational role served by the graph. The correlational role of the graph will have more influence on the outcome variables when τ becomes larger. This explains the intriguing observation: GCN fails to utilize the correlational information and its advantages on the representational information diminish as τ increases.

For the U.S. Election dataset (Table 4.2), we observe that all variants of CopulaGNN significantly outperform their base GNN counterparts. It is interesting that the simpler two-parameter parameterization outperforms the regression-based parameterization in most setups on this dataset. One possible explanation is that the outcome variables that are connected in the graph tend to have strong correlations, since adjacent counties usually have similar statistics. This is indeed suggested by Jia and Benson [67]. The Unemployment task in particular, where the two-parameter parameterization appears to have the largest advantage, is shown to have the strongest correlation.

Table 4.2: Experiment results on regression tasks of the U.S. Election dataset (with continuous outcome variables). The average test R^2 from 10 trials is reported (the larger the better). The asterisk markers and the ($\pm$) error bar indicate the same meaning as in Table 4.1.

	Education	Election	Income	Unemployment
MLP	0.660 $\pm$ 0.004	0.400 $\pm$ 0.003	0.597 $\pm$ 0.006	0.400 $\pm$ 0.004
GCN	0.418 $\pm$ 0.004	0.472 $\pm$ 0.002	0.607 $\pm$ 0.003	0.572 $\pm$ 0.010
$\alpha\beta$-C-GCN	0.452 $\pm$ 0.001***	0.558 $\pm$ 0.002***	0.635 $\pm$ 0.001***	0.750 $\pm$ 0.004***
R-C-GCN	0.454 $\pm$ 0.002***	0.578 $\pm$ 0.005***	0.661 $\pm$ 0.001***	0.654 $\pm$ 0.009***
SAGE	0.677 $\pm$ 0.004	0.565 $\pm$ 0.005	0.714 $\pm$ 0.006	0.628 $\pm$ 0.005
$\alpha\beta$-C-SAGE	0.709 $\pm$ 0.003***	0.700 $\pm$ 0.003***	0.775 $\pm$ 0.003***	0.813 $\pm$ 0.004***
R-C-SAGE	0.707 $\pm$ 0.003***	0.692 $\pm$ 0.005***	0.763 $\pm$ 0.003***	0.695 $\pm$ 0.003***

4.5.3 Regression with Count Outcome Variables

We use two groups of datasets with count outcome variables. The first group consists of two Wikipedia datasets: Wiki-Chameleon and Wiki-Squirrel [122]; both are page-page networks of Wikipedia pages with the visiting traffic as node labels. The second group is a co-citation network of papers at the EMNLP conferences. The goal is to predict the overall number of citations of each paper (including citations from outside EMNLP). We use the R^2-deviance, an R^2 measure for count data [20], to measure the model performance.

Results. The results of the count regression tasks are shown in Table 4.3. Intuitively, hyper-linked web pages or co-cited papers are more likely to be visited or cited together, therefore leading to correlated outcome variables captured by the graph. Indeed, we observe that the different variants of CopulaGNN outperform their base model counterparts in almost all setups. However, as the correlation may not be as strong as in the U.S. Election dataset, we observe that the regression-based parameterization (R-C-GCN and R-C-SAGE) has a greater advantage.

Table 4.3: Experiment results on regression tasks with count outcome variables. The average test R^2 (deviance) from 50 trials is reported (the larger the better). The asterisk markers and the ($\pm$) error bar indicate the same meaning as in Table 4.1.

	EMNLP	Wiki-Chameleon	Wiki-Squirrel
MLP	0.125 $\pm$ 0.006	0.347 $\pm$ 0.008	0.439 $\pm$ 0.004
GCN	0.609 $\pm$ 0.002	0.408 $\pm$ 0.008	0.470 $\pm$ 0.006
$\alpha\beta$-C-GCN	0.630 $\pm$ 0.001***	0.405 $\pm$ 0.007	0.476 $\pm$ 0.006***
R-C-GCN	0.657 $\pm$ 0.001***	0.424 $\pm$ 0.007**	0.490 $\pm$ 0.006***
SAGE	0.711 $\pm$ 0.002	0.343 $\pm$ 0.009	0.539 $\pm$ 0.004
$\alpha\beta$-C-SAGE	0.721 $\pm$ 0.002***	0.352 $\pm$ 0.009**	0.548 $\pm$ 0.004***
R-C-SAGE	0.734 $\pm$ 0.002***	0.360 $\pm$ 0.008***	0.551 $\pm$ 0.004***

4.6 Conclusion

In this work, we explicitly distinguish the representational and correlational roles of the graph representation of data. We demonstrate through a simulation study that many popular GNN models are incapable of fully utilizing the correlational graph information. Furthermore, we propose CopulaGNN, a principled method that improves upon a wide range of GNNs to achieve better prediction performance when the graph plays both representational and correlational roles. Compared with the corresponding base GNN models, multiple variants of CopulaGNN yield consistently superior results on both synthetic and real-world datasets for continuous and count regression tasks.

CHAPTER V

Generalization and Fairness of Graph Neural Networks

5.1 Introduction

Graph Neural Networks (GNNs) [52, 124, 77] are a family of machine learning models that can be used to model non-Euclidean data as well as inter-related samples in a flexible way. In recent years, there have been enormous successful applications of GNNs in various areas, such as drug discovery [69], computer vision [106], transportation forecasting [158], recommender systems [157], etc. Depending on the type of prediction target, the application tasks can be roughly categorized into node-level, edge-level, subgraph-level, and graph-level tasks [151].

In contrast to the marked empirical success, theoretical understanding of the generalization ability of GNNs has been rather limited. Among the existing literature, some studies [42, 49, 89] focus on the analysis of graph-level tasks where each sample is an entire graph and the samples of graphs are IID. A very limited number of studies [125, 142] explore GNN generalization for node-level tasks but they assume the nodes (and their associated neighborhoods) are IID samples, which does not align with the commonly seen graph-based semi-supervised learning setups. Baranwal et al. [6] investigate GNN generalization without IID assumptions but under a specific data

generating mechanism.

In this chapter, our first contribution is to provide a novel PAC-Bayesian analysis for the generalization ability of GNNs on node-level tasks with non-IID assumptions. In particular, we assume the node features are fixed and the node labels are independently sampled from distributions conditioned on the node features. We also assume the training set and the test set can be chosen as arbitrary subsets of nodes on the graph. We first prove two general PAC-Bayesian generalization bounds (Theorem 5.4.2 and Theorem 5.4.3) under this non-IID setup. Subsequently, we derive a generalization bound for GNN (Theorem 5.4.10) in terms of characteristics of the GNN models and the node features.

Notably, the generalization bound for GNN is influenced by the distance between the test nodes and the training nodes in terms of their aggregated node features. This suggests that, given a fixed training set, test nodes that are "far away" from all the training nodes may suffer from larger generalization errors. Based on this analysis, our second contribution is the discovering of a type of unfairness that arises from theoretically predictable accuracy disparity across some subgroups of test nodes. We further conduct a empirical study that investigates the prediction accuracy of four popular GNN models on different subgroups of test nodes. The results on multiple benchmark datasets indicate that there is indeed a significant disparity in test accuracy among these subgroups.

We summarize the contributions of this chapter as follows:

(1) We establish a novel PAC-Bayesian analysis for graph-based semi-supervised learning with non-IID assumptions.

(2) Under this setup, we derive a generalization bound for GNNs that can be applied to an arbitrary subgroup of test nodes.

(3) As an implication of the generalization bound, we predict that there would be

an unfairness of GNN predictions that arises from accuracy disparity across subgroups of test nodes.

(4) We empirically verify the existence of accuracy disparity of popular GNN models on multiple benchmark datasets, as predicted by our theoretical analysis.

5.2 Related Work

In Chapter 2.3, we have introduced prior work investigating generalization and fairness of GNNs. In this section, we further give a very brief literature review on PAC-Bayesian analysis.

5.2.1 PAC-Bayesian Analysis

PAC-Bayesian analysis [103] has become one of the most powerful theoretical framework to analyze the generalization ability of machine learning models. We will briefly introduce the background in Section 5.3.2, and refer the readers to a recent tutorial [56] for a systematic overview of PAC-Bayesian analysis. We note that Liao et al. [89] recently present a PAC-Bayesian generalization bound for GNNs on IID graph-level tasks. Both Liao et al. [89] and this work utilize results from Neyshabur et al. [109], a PAC-Bayesian analysis for ReLU-activated neural networks, in part of our proofs. Compared to Neyshabur et al. [109], the key contribution of Liao et al. [89] is the derivation of perturbation bounds of two types of GNN architectures; while the key contribution of this work is the novel analysis under the setup of non-IID node-level tasks. There is also an existing work of PAC-Bayesian analysis for transductive semi-supervised learning [8]. But it is different from our problem setup and, in particular, it cannot be used to analyze the generalization on subgroups.

5.3 Preliminaries

In this section, we first formulate the problem of node-level semi-supervised learning. We also provide a brief introduction of the PAC-Bayesian framework.

5.3.1 The Problem Formulation and Notations

Semi-supervised node classification. Let $G = (V, E) \in \mathcal{G}_N$ be an undirected graph, with $V = \{1, 2, \ldots, N\}$ being the set of N nodes and $E \subseteq V \times V$ being the set of edges. And $\mathcal{G}_N$ is the space of all undirected graphs with N nodes. The nodes are associated with node features $X \in \mathbb{R}^{N \times D}$ and node labels $y \in \{1, 2, \ldots, K\}^N$.

In this work, we focus on the transductive node classification setting [155], where the node features X and the graph G are observed prior to learning, and every quantity of interest in the analysis will be conditioned on X and G. Without loss of generality, we treat X and G as fixed throughout our analysis, and the randomness comes from the labels y. In particular, we assume that for each node $i \in V$, its label y_i is generated from an unknown conditional distribution $\Pr(y_i \mid Z_i)$, where $Z = g(X, G)$ and $g : \mathbb{R}^{N \times D} \times \mathcal{G}_N \to \mathbb{R}^{N \times D'}$ is an aggregation function that aggregates the features over (multi-hop) local neighborhoods[1]. We also assume that the node labels are generated independently conditional on their respective aggregated features Z_i's.

Given a small set of the labeled nodes, $V_0 \subseteq V$, the task of node-level semi-supervised learning is to learn a classifier $h : \mathbb{R}^{N \times D} \times \mathcal{G}_N \to \mathbb{R}^{N \times K}$ from a function family $\mathcal{H}$ and perform it on the remaining unlabeled nodes. Given a classifier h, the classification for a node i is obtained by

$$\hat{y}_i = \underset{k \in \{1, \ldots, K\}}{\arg\max}\, h_i(X, G)[k],$$

[1] A simple example is $g_i(X, G) = \frac{1}{|\mathcal{N}(i)|+1} \left(X_i + \sum_{j \in \mathcal{N}(i)} X_j \right)$, where $g_i(X, G)$ is the i-th row of $g(X, G)$ and $\mathcal{N}(i) := \{j \mid (i, j) \in E\}$ is the set of 1-hop neighbors of node i.

where $h_i(X, G)$ is the i-th row of $h(X, G)$ and $h_i(X, G)[k]$ refers to the k-th element of $h_i(X, G)$.

Subgroups. In Section 6.4.1, we will present an analysis of the GNN generalization performance on any subgroup of the set of unlabeled nodes, $V \setminus V_0$. Note that the analysis on any subgroup is a stronger result than that on the entire unlabeled set, as any set is a subset of itself. Later we will show that the analysis on subgroups (rather than on the entire set) further allows us to investigate the accuracy disparity across subgroups. We denote a collection of subgroups of interest as $V_1, V_2, \ldots, V_M \subseteq V \setminus V_0$. In practice, a subgroup can be defined based on an attribute of the nodes (e.g., a gender group), certain graph-based properties, or an arbitrary partition of the nodes. We also define the size of each subgroup as $N_m := |V_m|, m = 0, \ldots, M$.

Margin loss on each subgroup. Now we can define the *empirical* and *expected margin loss* of any classifier $h \in \mathcal{H}$ on each subgroup $V_m, m = 0, 1, \ldots, M$. Given a sample of observed node labels y_i's, the empirical margin loss of h on V_m for a margin $\gamma \geq 0$ is defined as

$$\widehat{\mathcal{L}}_m^{\gamma}(h) := \frac{1}{N_m} \sum_{i \in V_m} \mathbb{1} h_i(X, G)[y_i] \leq \gamma + \max_{k \neq y_i} h_i(X, G)[k], \tag{5.1}$$

where $\mathbb{1}\cdot$ is the indicator function. The expected margin loss is the expectation of Eq. (5.1), i.e.,

$$\mathcal{L}_m^{\gamma}(h) := \mathbb{E}_{y_i \sim \Pr(\mathrm{y}|Z_i), i \in V_m} \widehat{\mathcal{L}}_m^{\gamma}(h). \tag{5.2}$$

To simplify the notation, we define $y^m := \{y_i\}_{i \in V_m}, \forall m = 0, \ldots, M$, so that Eq. (5.2) can be written as $\mathcal{L}_m^{\gamma}(h) = \mathbb{E}_{y^m} \widehat{\mathcal{L}}_m^{\gamma}(h)$. We note that the classification *risk* and *empirical risk* of h on V_m are respectively equal to $\mathcal{L}_m^0(h)$ and $\widehat{\mathcal{L}}_m^0(h)$.

5.3.2 The PAC-Bayesian Framework

The PAC-Bayesian framework [103] is an approach to analyze the generalization ability of a stochastic predictor drawn from a distribution Q over the predictor family $\mathcal{H}$ that is learned from the training data. For any stochastic classifier distribution Q and $m = 0, \ldots, M$, slightly overloading the notation, we denote the empirical margin loss of Q on V_m as $\widehat{\mathcal{L}}_m^\gamma(Q)$, and the corresponding expected margin loss as $\mathcal{L}_m^\gamma(Q)$. And they are given by

$$\widehat{\mathcal{L}}_m^\gamma(Q) := \mathbb{E}_{h\sim Q}\widehat{\mathcal{L}}_m^\gamma(h), \quad \mathcal{L}_m^\gamma(Q) := \mathbb{E}_{h\sim Q}\mathcal{L}_m^\gamma(h).$$

In general, a PAC-Bayesian analysis aims to bound the generalization gap between $\mathcal{L}_m^\gamma(Q)$ and $\widehat{\mathcal{L}}_m^\gamma(Q)$. The analysis is usually done by first proving that, for any "prior" distribution[2] P over $\mathcal{H}$ that is independent of the training data, the generalization gap can be controlled by the discrepancy between P and Q; the analysis is then followed by careful choices of P to get concrete upper bounds of the generalization gap. While the PAC-Bayesian framework is built on top of stochastic predictors, there exist standard techniques [81] that can be used to derive generalization bounds for deterministic predictors from PAC-Bayesian bounds.

Finally, we denote the *Kullback-Leibler (KL) divergence* as $D_{\mathrm{KL}}(Q\|P) := \int \ln \frac{dQ}{dP} dQ$, which will be used in the following analysis.

5.4 The Generalization Bound and Its Implications for Fairness

As we mentioned in Section 5.2.1, existing PAC-Bayesian analyses cannot be directly applied to the non-IID semi-supervised learning setup where we care about the

[2]The distribution is called "prior" in the sense that it doesn't depend on training data. "Prior" and "posterior" in PAC-Bayesian are different with those in conventional Bayesian statistics. See Guedj [56] for details.

generalization (and its disparity) across different subgroups of the unlabeled samples. In this section, we first present general PAC-Bayesian theorems for subgroup generalization under our problem setup; then we derive a generalization bound for GNNs and discuss fairness implications of the bound.

5.4.1 General PAC-Bayesian Theorems for Subgroup Generalization

Stochastic classifier bound. We first present the general PAC-Bayesian theorem (Theorem 5.4.2) for subgroup generalization of stochastic classifiers. The generalization bound depends on a notion of *expected loss discrepancy* between two subgroups as defined below.

Definition 5.4.1 (Expected Loss Discrepancy)**.** Given a distribution P over $\mathcal{H}$, for any $\lambda > 0$ and $\gamma \geq 0$, for any two subgroups V_m and $V_{m'}$ $(0 \leq m, m' \leq M)$, define the expected loss discrepancy between V_m and $V_{m'}$ with respect to (P, γ, λ) as

$$D^{\gamma}_{m,m'}(P;\lambda) := \ln \mathbb{E}_{h\sim P} e^{\lambda\left(\mathcal{L}^{\gamma/2}_{m}(h) - \mathcal{L}^{\gamma}_{m'}(h)\right)},$$

where $\mathcal{L}^{\gamma/2}_{m}(h)$ and $\mathcal{L}^{\gamma}_{m'}(h)$ follow the definition of Eq. (5.2).

Intuitively, $D^{\gamma}_{m,m'}(P;\lambda)$ captures the difference of the expected loss between V_m and $V_{m'}$ in an average sense (over P). Note that $D^{\gamma}_{m,m'}(P;\lambda)$ is asymmetric in terms of V_m and $V_{m'}$, and can be negative if the loss on V_m is mostly smaller than that on $V_{m'}$.

For stochastic classifiers, we have the following Theorem 5.4.2. Proof can be found in Appendix C.1.1.

Theorem 5.4.2 (Subgroup Generalization of Stochastic Classifiers)**.** *For any $0 < m \leq M$, for any $\lambda > 0$ and $\gamma \geq 0$, for any "prior" distribution P on $\mathcal{H}$ that is independent of the training data on V_0, with probability at least $1-\delta$ over the sample*

of y^0, for any Q on $\mathcal{H}$, we have[3]

$$\mathcal{L}_m^{\gamma/2}(Q) \leq \widehat{\mathcal{L}}_0^{\gamma}(Q) + \frac{1}{\lambda}\left(D_{\mathrm{KL}}(Q\|P) + \ln\frac{1}{\delta} + \frac{\lambda^2}{4N_0} + D_{m,0}^{\gamma}(P;\lambda)\right). \tag{5.3}$$

Theorem 5.4.2 can be viewed as an adaptation of a result by Alquier et al. [3] from the IID supervised setting to our non-IID semi-supervised setting. The terms $D_{\mathrm{KL}}(Q\|P), \ln\frac{2}{\delta}$, and $\frac{\lambda^2}{4N_0}$ are commonly seen in PAC-Bayesian analysis for IID supervised setting. In particular, when setting $\lambda = \Theta(\sqrt{N_0})$, $\frac{1}{\lambda}\left(\ln\frac{2}{\delta} + \frac{\lambda^2}{4N_0}\right)$ vanishes as the training size N_0 grows. The divergence between Q and P, $D_{\mathrm{KL}}(Q\|P)$, is usually considered as a measurement of the model complexity [56]. And there will be a trade-off between the training loss, $\widehat{\mathcal{L}}_0^{\gamma}(Q)$, and the complexity (how far can the learned "posterior" Q go from the "prior" P).

Uniquely for the non-IID semi-supervised setting, there is an extra term $D_{m,0}^{\gamma}(P;\lambda)$, which is the expected loss discrepancy between the target test subgroup V_m and the training set V_0. Note that this quantity is independent of the training labels y^0. Not surprisingly, it is difficult to give generalization guarantees if the expected loss on V_m is much larger than that on V_0 for any stochastic classifier P independent of training data. We have to make some assumptions about the relationship between V_m and V_0 to obtain a meaningful bound on $\frac{1}{\lambda}D_{m,0}^{\gamma}(P;\lambda)$, which we will discuss in details in Section 5.4.2.

Deterministic classifier bound. Utilizing standard techniques in PAC-Bayesian analysis [81, 103, 109], we can convert the bound for stochastic classifiers in Theorem 5.4.2 to a bound for deterministic classifiers as stated in Theorem 5.4.3 below (see Appendix C.1.2 for the proof).

Theorem 5.4.3 (Subgroup Generalization of Deterministic Classifiers)**.** *Let $\tilde{h}$ be any classifier in $\mathcal{H}$. For any $0 < m \leq M$, for any $\lambda > 0$ and $\gamma \geq 0$, for any "prior" distri-*

[3]Theorem 5.4.2 also holds when we substitute $\mathcal{L}_m^{\gamma/2}(h)$ and $\mathcal{L}_m^{\gamma/2}(Q)$ as $\mathcal{L}_m^{\gamma}(h)$ and $\mathcal{L}_m^{\gamma}(Q)$ respectively. But we state Theorem 5.4.2 in this form to ease the presentation of the later analysis.

bution P on $\mathcal{H}$ that is independent of the training data on V_0, with probability at least $1-\delta$ over the sample of y^0, for any Q on $\mathcal{H}$ such that $\Pr_{h\sim Q}\left(\max_{i\in V_0\cup V_m}\|h_i(X,G)-\tilde{h}_i(X,G)\|_\infty<\frac{\gamma}{2}\right)\geq\frac{1}{2}$*, we have*

$$\mathcal{L}_m^0(\tilde{h}) \leq \widehat{\mathcal{L}}_0^\gamma(\tilde{h}) + \frac{1}{\lambda}\left(2(D_{\mathrm{KL}}(Q\|P)+1) + \ln\frac{1}{\delta} + \frac{\lambda^2}{4N_0} + D_{m,0}^{\gamma/2}(P;\lambda)\right). \tag{5.4}$$

Theorem 5.4.2 and 5.4.3 are not specific to GNNs and hold for any (respectively stochastic and deterministic) classifier under the semi-supervised setup. In Section 5.4.2, we will apply Theorem 5.4.3 to obtain a subgroup generalization bound that explicitly depends on the characteristics of GNNs and the data.

5.4.2 Subgroup Generalization Bound for Graph Neural Networks

The GNN model. We consider GNNs where the node feature aggregation step and the prediction step are separate. In particular, we assume the GNN classifier takes the form of $h_i(X,G) = f(g_i(X,G);W_1,W_2,\ldots,W_L)$, where g is an aggregation function as we described in Section 5.3.1 and f is a ReLU-activated L-layer Multi-Layer Perceptron (MLP) with $W_1,\ldots,W_L$ as parameters for each layer[4]. Denote the largest width of all the hidden layers as b.

Remark V.1. *There is a technical restriction on the possible choice of the aggregation function g. For the following derivation of the generalization bound (5.6) to be valid, we need the condition that the node labels y_i's are independent conditional on their aggregated features $g_i(X,G)$'s, as introduced in the problem formulation in Section 5.3.1. However, we also note that this condition tends to hold when the aggregated features $g_i(X,G)$'s contain rich information about the labels.*

Upper-bounding $D_{m,0}^\gamma(P;\lambda)$. To derive the generalization guarantee, we need to upper-bound the expected loss discrepancy $D_{m,0}^\gamma(P;\lambda)$. It turns out that we have to

[4]SGC [148] and APPNP [79] are special cases of GNNs in this form.

make some assumptions on the data in order to get a meaningful upper bound.

So far we have not had any restrictions on the conditional label distributions $\Pr(y_i = k \mid g_i(X, G))$. If the label distributions on $V \setminus V_0$ can be arbitrarily different from those on V_0, the generalization can be arbitrarily poor. We therefore assume that the label distributions conditional on aggregated features are smooth (Assumption 5.4.4).

Assumption 5.4.4 (Smoothness of Data Distribution)**.** *Assume there exist c-Lipschitz continuous functions* $\eta_1, \eta_2, \ldots, \eta_K : \mathbb{R}^{D'} \to [0, 1]$*, such that, for any node* $i \in V$*,*

$$\Pr(y_i = k \mid g_i(X, G)) = \eta_k(g_i(X, G)), \forall k = 1, \ldots, K.$$

We also need to characterize the relationship between a target test subgroup V_m and the training set V_0. For this purpose, we define the distance from V_m to V_0 and the concept of *near set* below.

Definition 5.4.5 (Distance To Training Set and Near Set)**.** For each $0 < m \leq M$, define the distance from the subgroup V_m to the training set V_0 as

$$\epsilon_m := \max_{j \in V_m} \min_{i \in V_0} \|g_i(X, G) - g_j(X, G)\|_2.$$

Further, for each $i \in V_0$, define the near set of i with respect to V_m as

$$V_m^{(i)} := \{j \in V_m \mid \|g_i(X, G) - g_j(X, G)\|_2 \leq \epsilon_m\}.$$

Clearly,

$$V_m = \cup_{i \in V_0} V_m^{(i)}.$$

Then, with the Assumption 5.4.6 and 5.4.7 below, we can bound the expected loss discrepancy $D_{m,0}^{\gamma}(P; \lambda)$ with the following Lemma 5.4.8 (see the proof in Ap-

pendix C.1.3).

Assumption 5.4.6 (Equal-Sized and Disjoint Near Sets)**.** *For any $0 < m \leq M$, assume the near sets of each $i \in V_0$ with respect to V_m are disjoint and have the same size $s_m \in \mathbb{N}^+$.*

Assumption 5.4.7 (Concentrated Expected Loss Difference)**.** *Let P be a distribution on $\mathcal{H}$, defined by sampling the vectorized MLP parameters from $\mathcal{N}(0, \sigma^2 I)$ for some $\sigma^2 \leq \frac{(\gamma/8\epsilon_m)^{2/L}}{2b(\lambda N_0^{-\alpha} + \ln 2bL)}$. For any L-layer GNN classifier $h \in \mathcal{H}$ with model parameters $W_1^h, \ldots, W_L^h$, define $T_h := \max_{l=1,\ldots,L} \|W_l^h\|_2$. Assume that there exists some $0 < \alpha < \frac{1}{4}$ satisfying*

$$\Pr_{h \sim P}\left(\mathcal{L}_m^{\gamma/4}(h) - \mathcal{L}_0^{\gamma/2}(h) > N_0^{-\alpha} + cK\epsilon_m \mid T_h^L \epsilon_m > \frac{\gamma}{8}\right) \leq e^{-N_0^{2\alpha}}.$$

Lemma 5.4.8 (Bound for $D_{m,0}^{\gamma}(P;\lambda)$)**.** *Under Assumptions 5.4.4, 5.4.6 and 5.4.7, for any $0 < m \leq M$, any $0 < \lambda \leq N_0^{2\alpha}$ and $\gamma \geq 0$, assume the "prior" P on $\mathcal{H}$ is defined by sampling the vectorized MLP parameters from $\mathcal{N}(0, \sigma^2 I)$ for some $\sigma^2 \leq \frac{(\gamma/8\epsilon_m)^{2/L}}{2b(\lambda N_0^{-\alpha} + \ln 2bL)}$. We have*

$$D_{m,0}^{\gamma/2}(P;\lambda) \leq \ln 3 + \lambda cK\epsilon_m. \tag{5.5}$$

Intuitively, what we need to bound $D_{m,0}^{\gamma}(P;\lambda)$ is that the training set V_0 is "representative" for V_m. This is reasonable in practice as it is natural to select the training samples according to the distribution of the population. Specifically, Assumption 5.4.6 assumes that V_m can be split into equal-sized partitions indexed by the training samples. The elements of each partition $V_m^{(i)}$ are close to the corresponding training sample i but not so close to training samples other than i. This assumption is stronger than needed to obtain a meaningful bound on $D_{m,0}^{\gamma}(P;\lambda)$, and we can relax it by only assuming that most samples in V_m have proportional "close

representatives" in V_0. But we keep Assumption 5.4.6 in this work, as it is intuitively clear and significantly eases the analysis and notations. Assumption 5.4.7 essentially assumes that the expected margin loss on V_m is not much larger than that on V_0 when the number of samples becomes large. We first note that this assumption becomes trivially true in the degenerate case that all samples in V_m and V_0 are IID. In this case, $\mathcal{L}_m^{\gamma/4}(h) = \mathcal{L}_0^{\gamma/4}(h) < \mathcal{L}_0^{\gamma/2}(h) \leq 0$ for any classifier h. In Appendix C.1.5, we further provide a simple non-IID example where Assumption 5.4.7 holds.

The bound (5.5) suggests that the closer between V_m and V_0 (smaller ϵ_m), the smaller the expected loss discrepancy.

Bound for GNNs. Finally, with an additional technical assumption (Assumption 5.4.9) that the maximum L2 norm of aggregated node features does not grow in terms of the number of training samples, we obtain a subgroup generalization bound for GNNs in Theorem 5.4.10. The proof of Theorem 5.4.10 can be found in Appendix C.1.4.

Assumption 5.4.9. *Define* $B_m := \max_{i \in V_0 \cup V_m} \|g_i(X, G)\|_2$. *For any classifier* $\tilde{h} \in \mathcal{H}$ *with parameters* $\{\widetilde{W}_l\}_{l=1}^L$, *assume* $\|\widetilde{W}_l\|_F \leq C$ *for* $l = 1, \ldots, L$. *Assume* B_m, C *are constants with respect to* N_0.

Theorem 5.4.10 (Subgroup Generalization Bound for GNNs)**.** *Let* $\tilde{h}$ *be any classifier in* $\mathcal{H}$ *with parameters* $\{\widetilde{W}_l\}_{l=1}^L$. *Under Assumptions 5.4.4, 5.4.6, 5.4.7, and 5.4.9, for any* $0 < m \leq M$, $\gamma \geq 0$, *and large enough* N_0, *with probability at least* $1 - \delta$ *over the sample of* y^0, *we have*

$$\mathcal{L}_m^0(\tilde{h}) \leq \widehat{\mathcal{L}}_0^\gamma(\tilde{h}) + O\left(cK\epsilon_m + \frac{b\sum_{l=1}^L \|\widetilde{W}_l\|_F^2}{(\gamma/8)^{2/L} N_0^\alpha} (\epsilon_m)^{2/L} + \frac{1}{N_0^{1-2\alpha}} + \frac{1}{N_0^{2\alpha}} \ln \frac{LC(2B_m)^{1/L}}{\gamma^{1/L}\delta} \right). \tag{5.6}$$

Next, we investigate the qualitative implications of our theoretical results.

5.4.3 Implications for Fairness of Graph Neural Networks

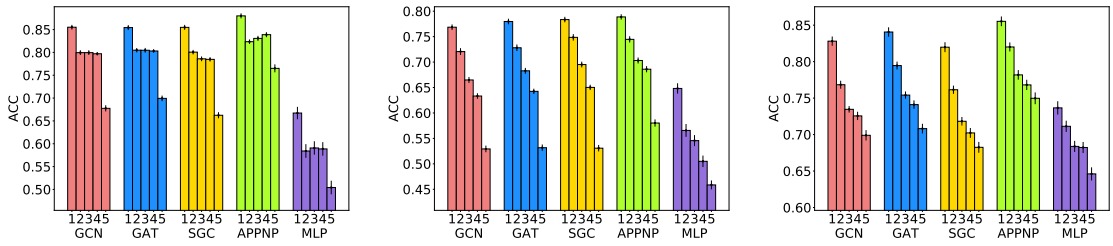
Theoretically predictable accuracy disparity. One merit of our analysis is that we can apply Theorem 5.4.10 on different subgroups of the unlabeled nodes and compare the subgroup generalization bounds. This allows us to study the accuracy disparity across subgroups from a theoretical perspective.

A major factor that affects the generalization bound (5.6) is ϵ_m, the aggregated-feature distance (in terms of $g(X, G)$) from the target test subgroup V_m to the training set V_0. The generalization bound (5.6) suggests that there is a better generalization guarantee for subgroups that are closer to the training set. In other words, it is unfair for subgroups that are far away from the training set. While our theoretical analysis can only tell the difference among *upper bounds* of generalization errors, we empirically verify that, in the following Section 6.5, the aggregated-feature distance ϵ_m is indeed a strong predictor for the test accuracy of each subgroup V_m. More specifically, the test accuracy decreases as the distance increases, which is consistent with the theoretical prediction given by the bound (5.6).

Impact of the structural positions of nodes. We further investigate if the aggregated-feature distance can be related to simpler and more interpretable graph characteristics, in order to obtain a more intuitive understanding of how the structural positions of nodes influence the prediction accuracy on them. We find that the *geodesic distance* (length of the shortest path) between two nodes is positively related to the distance between their aggregated features in some scenarios[5], such as when the node features exhibit homophily [104]. Empirically, we also observe that test nodes with larger geodesic distance to the training set tend to suffer from lower accuracy (see Figure V.2).

In contrast, we find that common node centrality metrics (e.g., degree and PageRank) have less influence on the test accuracy (see Figure V.3). These centrality metrics

[5]A more detailed discussion on such scenarios is provided in Appendix C.4.1.



(a) Cora. (b) Citeseer. (c) Pubmed.

Figure V.1: Test accuracy disparity across subgroups by aggregated-feature distance. Each figure corresponds to a dataset, and each bar cluster corresponds to a model. Bars labeled 1 to 5 represent subgroups with increasing distance to training set. Results are averaged over 40 independent trials with different random splits of the data, and the error bar represents the standard error of the mean.

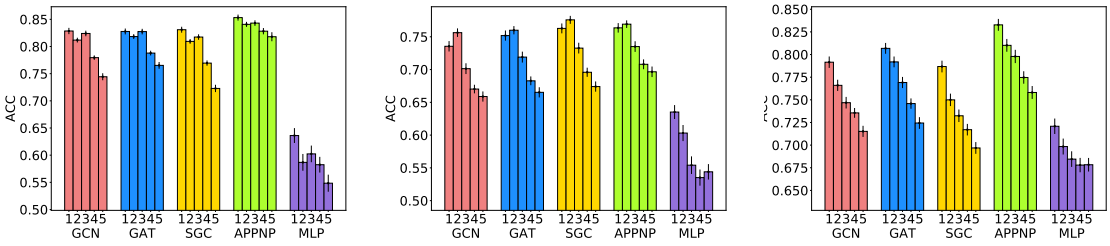
only capture the graph characteristics of the test nodes alone, but do not take their relationship to the training set into account, which is a key factor suggested by our theoretical analysis.

Impact of training data selection. Another implication of the theoretical results is that the selection of the training set plays an important role in the fairness of the learned GNN models. First, if the training set is selected unevenly on the graph, leaving part of the test nodes far away, there will likely be a large accuracy disparity. Second, a key ingredient in the proof of Lemma 5.4.8 is that the GNN predictions on two nodes tend to be more similar if they are closer in terms of the aggregated node features. This suggests that, if an individual training node is close to many test nodes, it may bias the predictions of the learned GNN on the test nodes towards the class it belongs to.

5.5 Experiments

In this section, we empirically verify the fairness implications suggested by our theoretical analysis.

General setup. We experiment on 4 popular GNN models, GCN [77], GAT [141],



(a) Cora. (b) Citeseer. (c) Pubmed.

Figure V.2: Test accuracy disparity across subgroups by geodesic distance. The experiment and plot settings are the same as Figure V.1, except for the bars labeled from 1 to 5 here represent subgroups with increasing shortest-path distance to training set.

SGC [148], and APPNP [79], as well as an MLP model for reference. For all models, we use the implementations by Deep Graph Library [145]. For each experiment setting, 40 independent trials are carried out.

5.5.1 Accuracy Disparity Across Subgroups

Subgroups. We examine the accuracy disparity with three types of subgroups as described below.

Subgroup by aggregated-feature distance. In order to directly investigate the effect of ϵ_m on the generalization bound (5.6), we first split the test nodes into subgroups by their distance to the training set in terms of the aggregated features. We use the two-step aggregated features to calculate the distance. In particular, denote the adjacency matrix of the graph G as $A \in \{0,1\}^{N\times N}$ and the corresponding degree matrix as D, where D is an $N \times N$ diagonal matrix with $D_{ii} = \sum_{j=1}^{N} A_{ij}, \forall i = 1, \ldots, N$. Given the feature matrix $X \in \mathbb{R}^{N\times D}$, the two-step aggregated features Z are obtained by $Z = (D+I)^{-1}(A+I)(D+I)^{-1}(A+I)X$. For each test node i, we calculate its aggregated-feature distance to the training set V_0 as $d_i = \min_{j\in V_0} \|Z_i - Z_j\|_2$. Then we sort the test nodes according to this distance and split them into 5 equal-sized subgroups.

Strictly speaking, our theory does not directly apply to GCN and GAT as they are not in the form as we defined in Section 5.4.2. Moreover, the two-step aggregated feature does not match exactly to the feature aggregation function of SGC and APPNP. Nevertheless, we find that even with such approximations, we are still able to observe the expected descending trend of test accuracy with respect to increasing distance in terms of the two-step aggregated features, on all four GNN models.

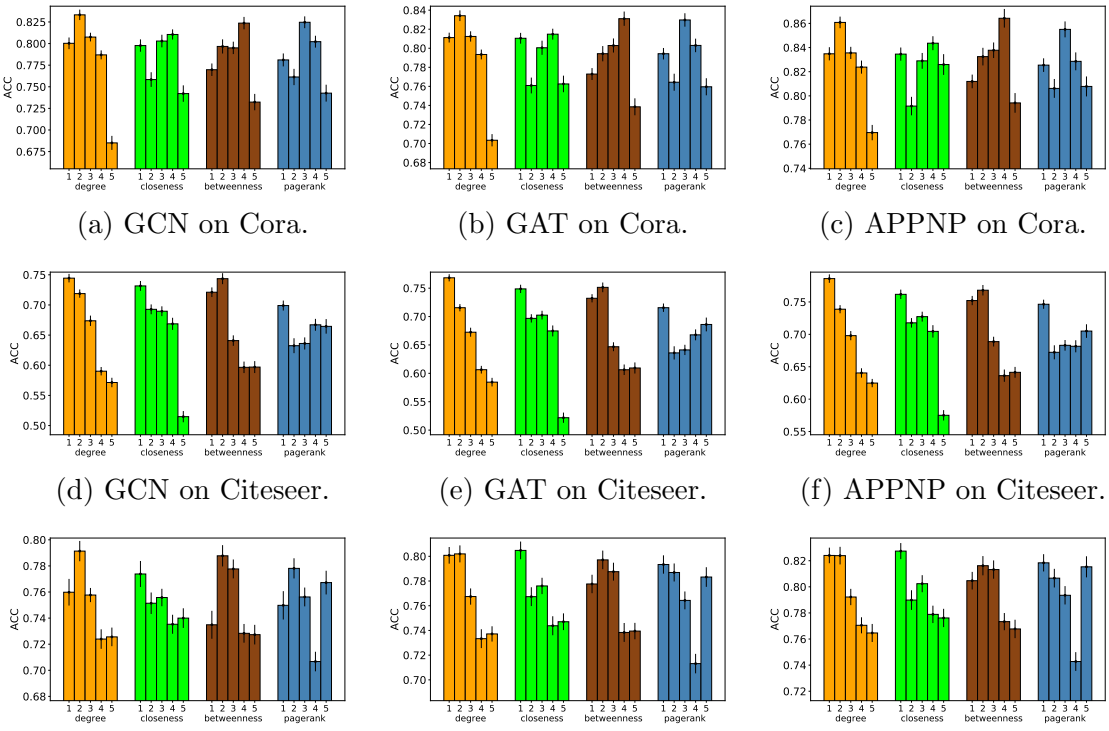
Subgroup by geodesic distance. As we discussed in Section 5.4.3, geodesic distance on the graph may well relate to the aggregated-feature distance. So we also define subgroups based on geodesic distance. We split the subgroups by replacing the aggregated-feature distance d_i of each test node i with the minimum of the geodesic distances from i to each training node on the graph.

Subgroup by node centrality. Lastly, we define subgroups based on 4 types of common node centrality metrics (degree, closeness, betweenness, and PageRank) of the test nodes. We split the subgroups by replacing the aggregated-feature distance d_i of each test node i with the centrality score of i. The purpose of this setup is to show that the common node centrality metrics are not sufficient to capture the monotonic trend of test accuracy.

Experiment setup. Following common GNN experiment setup [130], we randomly select 20 nodes in each class for training, 500 nodes for validation, and 1,000 nodes for testing. Once training is done, we report the test accuracy on subgroups defined by aggregated-feature distance, geodesic distance, and node centrality in Figure V.1, V.2, and V.3 respectively[6].

Experiment results. First, as shown in Figure V.1, there is a clear trend that the accuracy of a test subgroup decreases as the aggregated-feature distance between the test subgroup and the training set increases. And the trend is consistent for all 4

[6]The main paper reports the results on a small set of datasets (Cora, Citeseer, and Pubmed [126, 155]). Results on more datasets, including large-scale datasets from Open Graph Benchmark [63], are shown in Appendix C.3.



(a) GCN on Cora. (b) GAT on Cora. (c) APPNP on Cora.

(d) GCN on Citeseer. (e) GAT on Citeseer. (f) APPNP on Citeseer.

(g) GCN on Pubmed. (h) GAT on Pubmed. (i) APPNP on Pubmed.

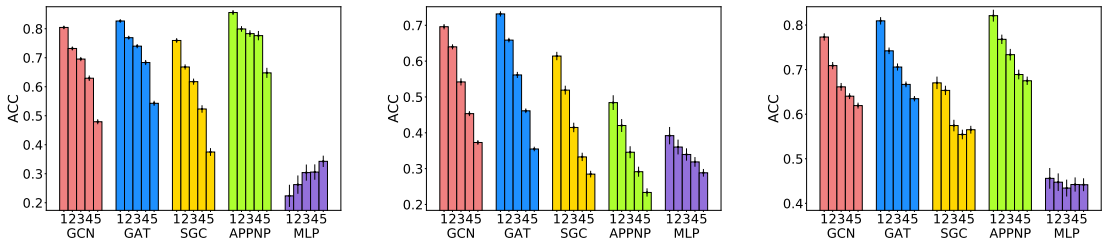
Figure V.3: Test accuracy disparity across subgroups by node centrality. Each figure corresponds to the results of a pair of model and dataset, and each bar cluster corresponds to the subgroups defined by a certain centrality metric. In each cluster, the bars labeled from 1 to 5 represent subgroups with decreasing node centrality. Other settings are the same as Figure V.1.

GNN models on all the datasets we test on (except for APPNP on Cora). This result verifies the existence of accuracy disparity suggested by Theorem 5.4.10.

Second, we observe in Figure V.2 that there is a similar trend when we split subgroups by the geodesic distance. This suggests that the geodesic distance on the graph can sometimes be used as a simpler indicator in practice for machine learning fairness on graph-structured data. Using such a classical graph metric as an indicator also helps us connect graph-based machine learning to network theory, especially to understandings about social networks, to better analyze fairness issues of machine learning on social networks, where high-stake decisions related to human subjects may be involved.

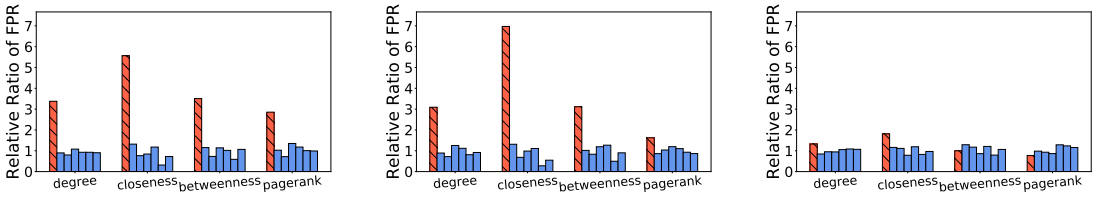
Furthermore, as shown in Figure V.3, there is no clear monotonic trend for test accuracy when we split subgroups by node centrality, except for some particular combinations of GNN model and dataset. Empirically, the common node centrality metrics are not as good as the geodesic distance in terms of capturing the accuracy disparity. This contrast highlights the importance of the insight provided by our analysis: the "distance" to the training set, rather than some graph characteristics of the test nodes alone, is the key predictor of test accuracy.

Finally, it is intriguing that, in both Figure V.1 and V.2, the test accuracy of MLP (which does not use the graph structure) also decreases as the distance of a subgroup to the training set increases. This result is perhaps not surprising if the subgroups were defined by distance on the original node features, as MLP can be viewed as a special GNN where the feature aggregation function is an identity mapping, so the "aggregated features" for MLP essentially equal to the original features. Our theoretical analysis can then be similarly applied to MLP. The question is why there is also an accuracy disparity w.r.t. the aggregated-feature distance and the geodesic distance. We suspect this is because these datasets present homophily, i.e., original (non-aggregated) features of geodesically closer nodes tend to be more simi-



(a) Cora. (b) Citeseer. (c) Pubmed.

Figure V.4: Test accuracy disparity across subgroups by aggregated-feature distance, experimented with noisy features. The experiment and plot settings are the same as Figure V.1, except for the node features are perturbed by independent noises such that they are less homophilous.



(a) GCN on Cora. (b) GAT on Cora. (c) MLP on Cora.

Figure V.5: Relative ratio between the FPR under biased training node selection and the FPR under uniform training node selection. Each bar in each cluster corresponds to a class (there are 7 classes in total). The red shaded bar indicates the class with high centrality training nodes under the biased setup. Each cluster corresponds to a centrality metric being used for the biased node selection.

lar. As a result, a subgroup with smaller geodesic distance may also have closer node features to the training set. To verify this hypothesis, we repeat the experiments in Figure V.1, but with independent noises added to node features such that they become less homophilous. As in Figure V.4, the decreasing pattern of test accuracy across subgroups remains for the 4 GNNs on all datasets; while for MLP, the pattern disappears on Cora and Pubmed and becomes less sharp on Citeseer.

5.5.2 Impact of Biased Training Node Selection

In all the previous experiments, we follow the standard GNN training setup where 20 training nodes are uniformly sampled for each class. Next we investigate the impact

if the selection of training nodes is biased, verifying our discussions in Section 5.4.3. We will demonstrate that the node centrality scores of the training nodes play an important role in the learned GNN model.

We choose a "dominant class" and construct a manipulated training set. For each class, we still sample 20 training nodes but in a biased way. For the dominant class, the sample is biased towards nodes of high centrality; while for other classes, the sample is biased towards nodes of low centrality. We evaluate the relative ratio of False Positive Rate (FPR) for each class between the setup using the manipulated training set and the setup using a uniformly sampled training set.

As shown in Figure V.5, compared to MLP, the GNN models have significantly worse FPR for the dominant class when the training nodes are biased. This is because, after feature aggregation, there will be a larger proportion of test nodes that are closer to the training nodes of higher centrality. And the learned GNN model will be heavily biased towards the training labels of these nodes.

5.6 Conclusion

We present a novel PAC-Bayesian analysis for the generalization ability of GNNs on node-level semi-supervised learning tasks. As far as we know, this is the first generalization bound for GNNs for non-IID node-level tasks without strong assumptions on the data generating mechanism. One advantage of our analysis is that it can be applied to arbitrary subgroups of the test nodes, which allows us to investigate an accuracy-disparity style of fairness for GNNs. Both the theoretical and empirical results suggest that there is an accuracy disparity across subgroups of test nodes that have varying distance to the training set, and nodes with larger distance to the training nodes suffer from a lower classification accuracy.

CHAPTER VI

Partition-Based Active Learning for Graph Neural Networks

6.1 Introduction

In the previous chapter, we have seen that the selection of training nodes plays a critical role in the generalization and fairness of GNNs. In this chapter, we investigate the problem of Graph-based Semi-Supervised Learning (GSSL) with GNNs in an *active learning* setup [128], where one is allowed to actively query node labels on the graph given a limited annotation budget. Our goal is to design effective active learning strategy that can improve the generalization and fairness of GNNs.

The active learning setup is also particularly interesting in the context of GSSL as we usually have access to abundant unlabeled samples prior to learning and, in many cases (e.g., on a social network), we have the flexibility to query the labels for a small portion of the samples. Furthermore, since a key advantage of GNN is the ability to utilize the relational information among the inter-connected samples, properly selecting nodes to annotate may further enhance the GNN performance.

However, directly adapting conventional active learning methods to GSSL may be sub-optimal due to the special structure of the problem and the GNN models. Indeed, utilizing proper smoothness properties of the data has been a key ingredient for the

success of many active learning methods. For example, a commonly used assumption (which we call *feature smoothness*) is that samples with similar features have higher chances to fall into the same class. In addition to feature smoothness, real-world GSSL tasks often leverage multiple types of smoothness properties over the graph, spanning the spectrum between *local smoothness* and *global smoothness* [163]. While there have been existing graph-based active learning methods utilizing some of these smoothness properties [36, 19, 150], methods that fully utilize feature and structural smoothness at the proper level are rare.

In this chapter, we propose *GraphPart*, a Graph-Partition-based active learning method for GNNs. The method is largely motivated by the community structures that are commonly present in real-world graphs. Node and structural properties often exhibit homogeneity within a community and heterogeneity across communities. We formalize this observation with proper smoothness assumptions of the graph-structured data at community level (represented by partitions of the graph) and conduct a novel analysis of the GNN classification error under such assumptions. The analysis further motivates the graph-partitioning step in the proposed method, GraphPart. In particular, GraphPart first splits the graph into several partitions based on modularity [29] and then selects the most representative nodes within each partition to query. An important merit of the proposed method is that it does not introduce additional hyperparameters, which is crucial for active learning setups as labeled validation data are often absent. Through extensive experiments, we demonstrate that the proposed method outperforms existing active learning methods for GNNs on multiple benchmark datasets for a wide range of annotation budgets. In addition, the proposed active learning method is able to mitigate the accuracy disparity phenomenon of GNNs that we have seen in Chapter V.

6.2 Related Work

Active learning is a subfield of machine learning that primarily concerns about the bottleneck in expensive annotation and attempts to achieve high accuracy by querying as few labeled samples as possible. Early efforts in this field prior to the emergence of deep learning has been comprehensively summarized by Settles [128].

Active Learning Setups The classic active learning algorithms query one sample at a time and label it. Such a setting is inefficient for a deep learning model as it frequently retrains but updates little, and it is prone to overfitting [120]. Therefore, in deep active learning, the batch-mode setting, where a diverse set of instances are sampled and queried, is more often considered. In recent years, the *optimal experimental design* principle [116, 2] motivates the machine learning community to minimize the use of training resources and avoid tuning on a validation set. Combining the settings of one-shot learning and batch-mode active learning, some recent studies [31, 150] adopt a one-step batch-mode active learning setting. In each run, the algorithm use up the predefined budget to select a batch of nodes to label. The querying process is done once and for all in order to minimize retraining. In this work, we focus on such one-step batch-mode setup as we concern active learning for graph neural networks.

Active Learning on Graphs Early works in active learning on graphs [55, 66, 36] are designed specifically for non-deep-learning models and/or fail to take the node features into consideration. As deep geometric learning and GNNs become popular, iterative node selection criteria were designed upon the expressiveness of GNNs. In particular, AGE [19] evaluates the informativeness of nodes by linearly combining centrality, density and uncertainty, Gao et al. [48] further proposed ANRMAB, which extends this framework by dynamically learning the weights with a multi-armed bandit

mechanism and maximizing the surrogate reward. Similarly, Chen et al. [27] follows previous works and proposed ActiveHNE, which extends AL on non-i.i.d data and heterogeneous networks. The above frameworks neglect interaction between nodes and suffer from sub-optimality due to short-term surrogate criterion. Upon this observation, Hu et al. [62] propose GPA, which formalizes AL task as a Markov decision process and learns the optimal query strategy with reinforcement learning techniques. More recent works approached the problem by incremental clustering [92], adversarial learning [88] and influence maximization [160, 161]. Yet, all of these models are based on an iterative setting. Given some efforts to avoid validation [119], iterative querying and retraining is still required.

Another line of optimization-based approaches develop active learning algorithms by investigating upper bounds of the classification loss. FeatProp [150] is one of the state-of-the-art active learning approaches for GNNs, which derives an upper bound of the node classification error under smoothness assumptions over the label distributions and GNN models. The theoretical analysis by Wu et al. [150] follows prior work, including ANDA [9] which proves a finite sample bound on the expected loss of KNNs in the covariate shift setting, and Coreset [127] which conducts an analysis on Convolution Neural Networks. A common pattern of the active learning algorithms along this line is that the algorithms seek to select a set of nodes that achieves a good coverage of the space of sample features or hidden representations, and this is typically done via certain clustering algorithms. Our work falls into this category. Compared to existing methods, our method utilizes both the local and global smoothness properties of graph-structured data, where the latter was largely missing in the literature of active learning for GNNs so far.

6.3 Preliminaries

6.3.1 Notations

We start by introducing useful notations to characterize node classification with GNNs.

Attributed Graph The task of node classification is defined on an attributed graph. Define $[n] = \{1, 2, \cdots, n\}$. We denote a graph of size n as $G = (V, A)$, where $V = [n]$ is the set of nodes and $A \in \{0, 1\}^{n \times n}$ is the adjacency matrix. Each node $i \in V$ is associated with a d-dimensional feature vector $\mathbf{x}_i \in \mathcal{X} \subseteq \mathbb{R}^d$, and a label $y_i \in \mathcal{Y} = [C]$, where C is the number of classes. Denote $X \in \mathbb{R}^{n \times d}$ as the feature matrix stacking each node feature. Given the original adjacency matrix A, the degree matrix $D \in \mathbb{R}^{n \times n}$ is defined as $D_{ii} = \sum_j A_{ij}, \forall i \in V$, and $D_{ij} = 0, \forall i, j \in V$, $i \neq j$. We denote S as the normalized adjacency matrix with added self-loops: $S = (I + D)^{-\frac{1}{2}}(A + I)(I + D)^{-\frac{1}{2}}$.

Graph Neural Networks GNNs are a family of neural networks modeling graph-structured data. A GNN model is typically defined with two types of operations on the node representations: aggregation and transformation. For example, an L-layer Graph Convolution Network (GCN) [77] can be recursively represented by a series of aggregation operations $g^{(l)}_{\text{GCN}}$ and transformation operations $h^{(l)}_{\text{GCN}}$, for $l = 1, \ldots, L$, defined as follows,

$$\widetilde{H}^{(l)} = g^{(l)}_{\text{GCN}}(H^{(l-1)}, G) := SH^{(l-1)}, \quad \text{and}$$

$$H^{(l)} = h^{(l)}_{\text{GCN}}(\widetilde{H}^{(l)}) := \text{ReLU}(\widetilde{H}^{(l)}W^{(l)})$$

where $H^{(0)} = X$ is the feature matrix, $\text{ReLU}(\cdot)$ is the element-wise rectified-linear unit activation function, and $W^{(l)}$ is the parameter matrix for the transformation

operation $h^{(l)}_{\text{GCN}}$.

Noticing that, in most GNNs, the prediction for each node only depends on a local neighborhood of that node, we can define a general GNN as a function that maps a local neighborhood of a node to its classification predictions. Define $\mathcal{N}^{(L)}(i)$ as the set of all neighboring nodes of node i that can be reached within L-hop on the graph (including itself), and $G_i^{(L)}$ as the subgraph of G restricted on $\mathcal{N}^{(L)}(i)$. For each node i, define $v_i^{(L)} := \left(\{x_j\}_{j\in\mathcal{N}^{(L)}(i)}, G_i^{(L)}\right)$, which is a tuple including all the node features and the subgraph in node i's local neighborhood. Let $\mathcal{V}^{(L)}$ be the space of such possible $v_i^{(L)}$. Then we can define a GNN as a function

$$f : \mathcal{V}^{(L)} \to \mathbb{R}^C.$$

In our later analysis in Section 6.4.1, we will consider a type of abstract GNNs in the form of $f = h \circ g$, where the aggregation operation $g : \mathcal{V}^{(L)} \to \mathbb{R}^{d'}$ and the transformation operation $h : \mathbb{R}^{d'} \to \mathbb{R}^C$ are separated. In particular, g can represent any types of feature aggregation on the L-hop ego-network of a node. And h is a transformation function with learnable parameters, which is often modeled by a Multi-Layer Perceptron (MLP). Some recently developed GNN models, such as SGC [148] and APPNP [79], are special cases of this form.

In the rest of this chapter, we omit the superscription (L) in notations for simplicity.

Margin Loss Given some $\gamma \geq 0$, the *margin loss* [109] on a GNN classifier $f : \mathcal{V} \to \mathbb{R}^C$ for a given labeled sample (v_i, y_i) is defined as follows,

$$\mathcal{L}_\gamma(f(v_i), y_i) := \mathbb{1}\big[f(v_i)[y_i] \leq \gamma + \max_{c\neq y_i} f(v_i)[c]\big], \tag{6.1}$$

where $\mathbb{1}(\cdot)$ is the indicator function. When we set $\gamma = 0$, $\mathcal{L}_0$ corresponds to the 0-1 classification error.

6.3.2 Active Learning for Graph Neural Networks

In this work, we focus on the one-step batch-mode active learning setup [31, 150] for node classification using GNNs. Suppose there is an attributed graph where the graph structure and the node features are known but the node labels are unobserved. We assume that for each node i, the node label y_i is a random variable following a conditional distribution $\Pr[y_i \mid v_i]$.

At the beginning of learning, we are given a small set $\mathbf{s}_0 \subseteq V$ of nodes being labeled, which we call a *seed set*. Given all the node samples $\{v_i\}_{i \in V}$ and labels on the seed set $\{y_i\}_{i \in \mathbf{s}_0}$, for a fixed annotation budget $b > 0$, an active learning algorithm aims to carefully select a set of nodes $\mathbf{s}_1 \subseteq V \setminus \mathbf{s}_0$ and $|\mathbf{s}_1| \leq b$, query the labels on $\mathbf{s}_1$, and train a GNN model $\hat{f}$ based on the labeled data $\{(v_i, y_i)\}_{i \in \mathbf{s}_0 \cup \mathbf{s}_1}$, such that the expected classification error on the remaining unlabeled set is small.

6.4 A Graph-Partition-Based Active Learning Framework

The key motivation of the proposed method is that many real-world graphs present community structures, which induces a proper level of smoothness properties in the graph-structured data. In this section, we first formalize this motivation through an analysis of the GNN performance given proper smoothness assumptions, and then introduce the proposed graph-partition-based active learning approach.

6.4.1 An Analysis of Expected Classification Error Under Smoothness Assumptions

The main result of our analysis is an upper bound (Proposition 4) on the quantity $\mathbb{E}_{y_i} \mathcal{L}_0(f(v_i), y_i)$, the expected classification error for a node i and a fixed GNN model

f.

Assumptions We first state the set of assumptions in order to establish the upper bound. Denote a *K-partition* of the graph as $\mathcal{T}_K = \{T_1, T_2, \cdots, T_K\}$, where $T_1, \ldots, T_K \subseteq V$ are disjoint partitions satisfying $\bigcup_{k=1}^{K} T_k = V$. Recall that we consider GNNs in the form of $f = h \circ g$, where $g : \mathcal{V} \to \mathbb{R}^{d'}$ is a feature aggregation function and $h : \mathbb{R}^{d'} \to \mathbb{R}^C$ is an MLP that takes the aggregated features as input and output the classification logits. We make the following two assumptions on the smoothness of the label distribution and the model.

Assumption 6.4.1 (Label Smoothness)**.** *Assume that $\forall c \in [C]$, there exists a function $\eta_c : \mathcal{V} \to [0, 1]$ such that $\Pr[y_i = c \mid v_i] = \eta_c(v_i)$ for any $i \in V$. Moreover, $\forall k \in [K]$, $\forall i, j \in T_k$, assume that there exists a constant $\delta_\eta < \infty$, such that*

$$|\eta_c(v_i) - \eta_c(v_j)| \leq \delta_\eta ||g(v_i) - g(v_j)||_2.$$

Assumption 6.4.2 (Model Smoothness)**.** *Assume that $\forall e, e' \in \mathbb{R}^{d'}$, the MLP h satisfies $||h(e) - h(e')||_\infty \leq \delta_h ||e - e'||_2$ for some constant $\delta_h < \infty$.*

The Main Result We use $T(i)$ to denote the partition where the node i belongs to, and denote for convenience the training set $S_{tr} := \mathbf{s}_0 \cup \mathbf{s}_1$ and the test set $S_{te} := V \setminus S_{tr}$. We have the following result.

Proposition 4. For any fixed GNN model f, under Assumptions 6.4.1 and 6.4.2, for any $i \in S_{te}$, if $S_{tr} \cap T(i) \neq \emptyset$, letting $\tau(i) := \arg\min_{l \in S_{tr} \cap T(i)} \|g(v_i) - g(v_l)\|_2$, $\epsilon_i := \|g(v_i) - g(v_{\tau(i)})\|_2$, and $\gamma_i := 2\delta_h \varepsilon_i$, then we have

$$\mathbb{E}_{y_i}[\mathcal{L}_0(f(v_i), y_i)] \leq C\delta_\eta \varepsilon_i + \mathbb{E}_{y_{\tau(i)}}[\mathcal{L}_{\gamma_i}(f(v_{\tau(i)}), y_{\tau(i)})]. \tag{6.2}$$

Proposition 4 provides an upper bound of the expected classification loss $\mathbb{E}_{y_i}[\mathcal{L}_0(f(v_i), y_i)]$

for each node i in the test set S_{te}. This upper bound is primarily dependent on the training node $\tau(i)$ and their distance ε_i on the aggregated feature space, where $\tau(i)$ is the closest training node to i among the ones reside in the same graph partition as i. Specifically, the first term is linearly proportional to ε_i; and the second term (the expected margin loss of $\tau(i)$) is increasing with respect to ε_i. This upper bound motivates an active learning algorithm that selects the training set by minimizing $\sum_{i \in S_{te}} \varepsilon_i$, which we will introduce in details in Section 6.4.2.

Remarks We make a few remarks on the bound (6.2) in comparison to relevant previous work. Our novel technical contributions in the analysis include that 1) we have a weaker assumption on label smoothness; and 2) our proof removes an unrealistic implicit assumption in previous work.

First, our analysis can be viewed as an extension of the results by [127] and [150]. One key difference between our analysis and theirs lies in the assumption on label smoothness (Assumption 6.4.1). Adapting the label smoothness assumption by Sener and Savarese [127] and Wu et al. [150] into our notations gives the following Assumption 6.4.3.

Assumption 6.4.3 (Label Smoothness by Sener and Savarese [127] and Wu et al. [150])**.** *Assume that* $\forall c \in [C]$*, there exists a function* $\eta_c : \mathcal{V} \to [0, 1]$ *such that* $\Pr[y_i = c \mid v_i] = \eta_c(v_i)$ *for any* $i \in V$*. Moreover,* $\forall i, j \in V$*, assume that there exists a constant* $\delta'_\eta < \infty$*, such that*

$$|\eta_c(v_i) - \eta_c(v_j)| \leq \delta'_\eta ||g(v_i) - g(v_j)||_2.$$

The following obvious fact indicates that Assumption 6.4.1 is weaker than Assumption 6.4.3.

Lemma 6.4.4. *If Assumption 6.4.3 holds, then there exists a constant* $\delta_\eta \leq \delta'_\eta$ *satis-*

fying Assumption 6.4.1.

In addition, the δ_η in Assumption 6.4.1 could be much smaller than the δ'_η Assumption 6.4.3 if the label distribution presents *global smoothness* over the graph [163], i.e., the distributions of node labels tend to be closer for nodes within a partition/community of the graph than for nodes reside in different partitions/communities.

Second, while the upper bounds on classification error given by Sener and Savarese [127] and Wu et al. [150] seem to be tighter than our bound (6.2), we remark that they made an implicit assumption in the proofs that is counter-intuitive. In particular, the implicit assumption is that, the label distributions of the selected training samples are always concentrated on one class. This assumption is counter-intuitive as the selection of training samples alters the label distributions of those samples. We avoid such an assumption in our analysis at the expense of introducing an extra margin loss term.

Finally, we remark that while GCN cannot be written in the form of $f = h \circ g$ as we assumed in our analysis, it has been shown by Wu et al. [150] that the difference between outputs of an L-layer GCN on two nodes $i, j \in V$ can be upper bounded by $\delta_{GCN}\|(S^L X)_i - (S^L X)_j\|_2$ for some constant $0 < \delta_{GCN} < \infty$. So the analysis in this section can be similarly applied to GCN.

6.4.2 The Proposed Graph Partition-Based Active Learning Framework

The General Framework Motivated by Proposition 4, we propose an active learning framework that selects the set of nodes to be labeled, $\mathbf{s}_1$, by solving the following optimization problem: for ε_i as defined in Proposition 4,

$$\min_{\mathbf{s}_1:|\mathbf{s}_1|\leq b} \sum_{i\in S_{te}} \epsilon_i = \min_{\mathbf{s}_1:|\mathbf{s}_1|\leq b} \sum_{i\in V} \min_{j\in T(i)\cap(\mathbf{s}_0\cup\mathbf{s}_1)} ||g(v_i) - g(v_j)||_2. \tag{6.3}$$

For any partition $T_k \in \mathcal{T}_K$, if we further specify a budget b_k for the number of nodes to be selected from this partition, such that $\sum_{k=1}^{K} = b_k = b$, then we can approximately[1] re-write the optimization problem (6.3) as K separate optimization problems as follows: for $k = 1, \ldots, K$,

$$\min_{\mathbf{s}_1^{(k)} \in T_k : |\mathbf{s}_1^{(k)}| \leq b_k,} \sum_{i \in T_k} \min_{j \in \mathbf{s}_1^{(k)}} ||g(v_i) - g(v_j)||_2, \tag{6.4}$$

where the individual optimization problem (6.4) is equivalent to minimizing the objective of a K-Medoids problem with b_k medoids on the partition T_k.

Given a K-partition of the graph and a feature aggregation function $g : \mathcal{V} \to \mathbb{R}^{d'}$, we summarize the proposed general framework, which we call it **GraphPart** (Graph-Partition-based query), in Algorithm VI.1. For the K-Medoids algorithm, in practice, we apply an efficient approximation by Park and Jun [112] by selecting nodes closest to K-Means centers. The aggregation function g should be chosen according to the GNN architecture that we will be training. For example, for a 2-Layer GCN model, we set $g(v_i) = (S^2X)_i$ for any $i \in V$, since this type of aggregation function effectively reflects the output difference of GCN as we remarked at the end of Section 6.4.1.

The Graph-Partition Method We use a modularity-based graph partition method, which is one of the most popular class of methods for community detection [108, 29]. Specifically, we obtain a K-partition of a graph using the Clauset-Newman-Moore greedy modularity maximization [29] method. This method is a bottom-up algorithm, which begins with communities containing each single node, and iteratively merges the pair of communities that increases modularity the most, until K communities are left. If no pairs of communities can be merged to increase modularity when more than K communities are left, we further use agglomerative hierarchical clustering [73] to iteratively merge small outlying communities until only K communities

[1] Omitting the seed set $\mathbf{s_0}$, which is 0 in one-step setting.

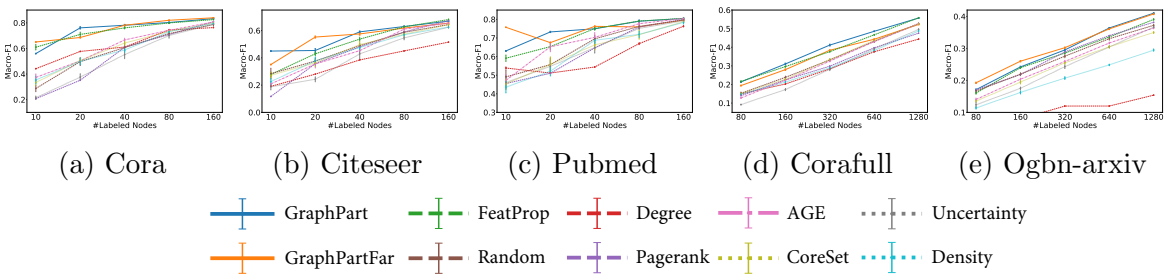
Algorithm VI.1: Graph-Partition-Based Query

Input: A K-partition $\mathcal{T}_K$ of the graph, budget b
Output: A subset of unlabelled nodes $\mathbf{s}_1$ of size b: $\mathbf{s}_1 \subseteq V \setminus \mathbf{s}_0$ and $|\mathbf{s}_1| = b$

Set $\mathbf{s}_1 = \emptyset$.
for $T_k \in \mathcal{T}_K$ **do**
  $b_k \leftarrow b//K$.
  $T_k \leftarrow T_k \setminus \{\mathbf{s}_0 \cup \mathbf{s}_1\}$.
  $E_k \leftarrow \{g(v_i)\}_{i \in T_k}$.
  $\mathbf{s} \leftarrow b_k$-`Medoids`$(E_k)$.
  `//Perform K-Medoids clustering on the set of data points` E_k `with` b_k `medoids returned as s.`
  $\mathbf{s}_1 = \mathbf{s}_1 \cup \mathbf{s}$.
end for
return $\mathbf{s}_1$

are left. Notably, we also propose a simple elbow method to automatically determine the number of partitions K without the need of node labels. Therefore the whole active node selection process is hyperparameter-free. Please see more details of the graph-partition method in Appendix D.3.

Compensating for the Interference across Partitions One potential shortcoming of the proposed GraphPart method shown in Algorithm VI.1 is that it ignores the interference across partitions as it optimizes separate K-Medoids problems independently on each partition. The medoids selected on two different partitions may be close to each other, which compromises the overall covering of the unlabeled nodes. To compensate for this problem, we come up with a greedy correction that when selecting the medoids in the partition T_k, we penalize the nodes that is close to the medoids already selected in the partitions $T_1, T_2, \ldots, T_{k-1}$. Instead of selecting nodes closest to K-Means centers, the distance function to minimize is penalized by the minimum distance to any selected node. We name this corrected variant of GraphPart as **GraphPartFar**, which makes sure that all the nodes returned to $\mathbf{s}_1$ are not too close and similar to each other, increasing the diversity of the pool.



(a) Cora (b) Citeseer (c) Pubmed (d) Corafull (e) Ogbn-arxiv

Figure VI.1: Performance of GCN on citation networks with training set selected by each active learning baseline in one shot, averaged from 10 different runs. Our methods are **embolden**. In each sub-figure, the Macro-F1 score is plotted against budget in log scale. The error bar stands for the standard error of the mean.

6.5 Experiments

6.5.1 Experiment Setup

We experiment on 7 public benchmark datasets, and compare the proposed methods against several state-of-the-art baseline active learning approaches on training 3 different GNN models. Following Wu et al. [150], we evaluate each baseline with a series of label budgets and report the Macro-F1 performance for node classification over the full graph. Each experiment setting is repeated with 10 random seeds.

Dataset We experiment on citation networks Citeseer, Cora, and Pubmed [126], three standard node classification benchmarks. We also experiment on Corafull [11] and Ogbn-Arxiv [63], for performance on denser network with more classes, and on co-authorship networks [130] for diversity. The summary statistics of the datasets are provided in Table D.1 in Appendix D.2.1.

GNN Models We perform experiments over different popular GNN models, including GCN [77], GraphSAGE [57], and GAT [141]. Details are described in Appendix D.2.2.

Baselines We compare active learning methods that can be applied to one-step setting, divided into two categories: 1) general-purpose methods that is agnostic to the graph structure (Random, Density, Uncertainty, and CoreSet); and 2) methods tailored for graph-structured data (Centrality, AGE, FeatProp, GraphPart, and GraphPartFar), where GraphPart and GraphPartFar are our proposed methods.

- **Random**: The Random Sampling method chooses nodes uniformly at random.
- **Density** [19]: The Density method first performs clustering on the hidden representations of the nodes, and then chooses nodes with maximum density score, which is (approximately) inversely proportional to the L_2-distance between each node and its cluster center.
- **Uncertainty** [129]: The Uncertainty method chooses the nodes with maximum entropy on the predicted class distribution.
- **CoreSet** [127]: The CoreSet method performs K-Center clustering over the hidden representations of nodes. Since the MIP optimized version is not scalable to large datasets, we use the time-efficient greedy approximation described in the original work.
- **Centrality**: The Centrality method chooses nodes with the largest graph centrality metric value, which only considers only the graph structure, but not the node features. As is empirically validated by Cai et al. [19], **Degree** centrality and **PageRank** centrality outperform other metrics.
- **AGE** [19]: AGE defines the informativeness of nodes by linearly combining three metrics: centrality, density and uncertainty. It further chooses nodes with the highest scores.
- **FeatProp** [150]: The FeatProp method first performs K-Means on the aggregated node features, and then choose the nodes closest to the K-Means centers.

- **GraphPart** and **GraphPartFar**: These are two variants of the proposed method as described in Section 6.4.2.

Active Learning Setup In our one-step setting, we vanish the seed set s_0 to zero. Note that some baseline methods (Density, Uncertainty, CoreSet, and AGE) are intended for iterative settings and require an initial model trained with the seed set $\mathbf{s}_0$, as these methods rely on the hidden representations of nodes or the predicted class distribution returned by the initial model. For these baselines, we choose one third of the budget as random initialization, and let the method select the other two thirds. On smaller datasets with sparser networks and less classes, we test each active learning approach with the series of budgets chosen as $2^{\{0,1,2,3,4\}} \times 10$. On the large datasets where the network is denser and the number of classes is drastically larger, the budgets are chosen as $2^{\{3,4,5,6,7\}} \times 10$.

6.5.2 Experiment Results

We provide the active learning results of GCN on 5 datasets in Figure VI.1. We further provide the exact average Macro-F1 scores and their standard errors for a sub-sample of budget sizes in Table 6.1. Due to page limit, the results of other model-dataset combinations are attached in Appendix D.4.

Overall, we observe in Figure VI.1 and Table 6.1 that the proposed methods, GraphPart and GraphPartFar, outperform baseline methods in a wide range of budgets before the performance saturates. On smaller datasets where the number of classes is relatively small, the proposed methods outperform baseline methods by a large margin under a moderate budget size. On Corafull and Arxiv where the number of classes is much larger, the proposed methods demonstrate advantages over baseline methods (in particular, FeatProp) in a slightly later stage. This may be due to that some classes were not yet fully visited so the model has not been learning reliably

Table 6.1: Summary of the performance of GCN on citation networks with 40/320 budget nodes queried. The numerical values represent the average Macro-F1 score of 10 independent trials and the error bar denotes the standard error of the mean (all in %). The **bold** marker denotes the best performance and the underlined marker denotes the second best performance. Asterisk (*) means the difference between our strategy and the best baseline strategy is statistically significant by a pairwise t-test at significance level 0.05.

Baselines Budget	Cora 40	Citeseer 40	Pubmed 40	Corafull 320	Ogbn-Arxiv 320
Random	60.7 ± 8.7	48.9 ± 5.8	69.5 ± 6.2	33.2 ± 1.8	27.6 ± 1.5
Uncertainty	55.2 ± 10.0	42.8 ± 12.5	64.8 ± 9.2	28.1 ± 2.1	24.3 ± 1.8
Density	58.7 ± 5.5	47.5 ± 6.4	68.7 ± 5.3	29.0 ± 0.9	20.7 ± 1.4
CoreSet	64.1 ± 5.3	49.6 ± 7.5	66.3 ± 8.6	33.4 ± 1.1	25.5 ± 1.2
Degree	61.1 ± 1.3	38.5 ± 0.3	54.4 ± 0.7	28.4 ± 5.9	12.0 ± 0.4
Pagerank	60.2 ± 1.2	45.6 ± 0.9	66.4 ± 0.1	29.9 ± 0.7	28.9 ± 0.1
AGE	66.7 ± 4.1	45.2 ± 7.7	70.3 ± 8.0	32.6 ± 1.1	25.9 ± 0.1
FeatProp	76.1 ± 2.5	53.7 ± 4.5	75.1 ± 2.8	37.6 ± 0.8	28.5 ± 0.6
GraphPart	78.1 ± 1.5	**59.0*** ± 2.0	74.9 ± 1.3	**41.2*** ± 1.4	29.5* ± 0.8
GraphPartFar	**78.1** ± 2.1	57.5* ± 2.9	**76.2** ± 0.9	38.4 ± 0.6	**30.2*** ± 0.6

when the budget size is small. Indeed, as shown in Figure VI.1d, VI.1e, the difference between our methods and FeatProp are not statistically significant until the budget size is at least 160.

The superior performance of the proposed methods, especially against the state-of-the-art baseline FeatProp, demonstrates that selecting training nodes within proper partitions of the graph significantly helps the active learning performance.

6.5.3 More Analysis

We further conducted two sets of more detailed analysis to better understand the proposed methods.

Mitigating Accuracy Disparity of GNNs The GNN models are reported to be less accurate on nodes that are further away from the training nodes, which leads to potential fairness concerns [99]. As the goal of proposed active learning methods is

to have the selected training nodes evenly distributed on the graph, we conduct an analysis on the accuracy disparity of the actively learned GNN models, following the study by Ma et al. [99]. As can be seen in Figure VI.2, we split the test nodes into subgroups according to their aggregated-feature distance to the training nodes and report the test accuracy on each subgroup. While the GNNs learned by GraphPart and GraphPartFar are still less accurate for test nodes that are further away, the accuracy disparity is mitigated in most cases compared to that by Random. The results on other datasets are provided in Appendix D.6.1.

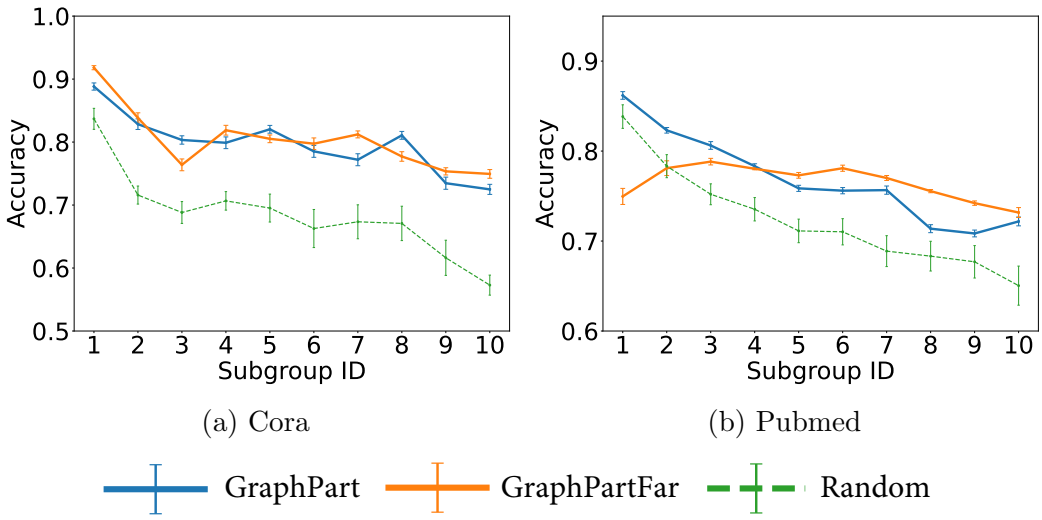


Figure VI.2: Accuracy disparity across 10 subgroups. Increasing subgroup indices represent increasing distance to selected training set.

Sensitivity Analysis on Graph Partitioning We further conduct an sensitivity analysis for the proposed method. In particular, we are interested in the effectiveness of the partition approach when combined with distance metrics on different node representations. Specifically, (1) **Aggregation**: aggregated node features S^2X; (2) **Embedding**: the last hidden layer of GCN trained on one third of the budget; and (3) **Feature**: the original node features. As is shown in Table 6.2, the graph-partition step is robustly effective when combined with various types of distance metrics. Due to the page limit, the complete results are attached in Appendix D.6.2.

Table 6.2: Performance of different combinations of distance metric on Cora datasets with or without using graph partition.

Node Rep.	With Partition?	Cora 20	Cora 40	Cora 80
Aggregation	yes	**76.1*** ± 2.7	**78.1** ± 1.5	**80.3** ± 1.6
	no	71.0 ± 5.7	76.1 ± 2.5	79.9 ± 0.9
Embedding	yes	**61.3*** ± 4.8	**69.4*** ± 3.9	**76.7** ± 3.4
	no	54.5 ± 4.7	62.6 ± 5.7	74.1 ± 3.7
Feature	yes	**65.6*** ± 2.6	**71.0*** ± 2.1	77.2 ± 1.3
	no	53.2 ± 5.2	64.0 ± 5.1	**77.4** ± 1.7

6.6 Conclusion and Discussion

In this work, we investigate the problem of active learning for GNNs. Inspired by the commonly seen local and global smoothness properties on graph-structured data, we propose a graph-partition-based active learning framework for GNNs, with two variants of concrete algorithms. The proposed framework can be seen as an active training node selection algorithm that approximately optimizes an upper bound of the expected classification error of unlabeled nodes. Through extensive experiments, we show that the proposed methods significantly outperform existing state-of-the-art baseline active learning methods. Furthermore, the proposed active learning methods are able to mitigate the accuracy disparity phenomenon commonly seen in GNN models.

CHAPTER VII

Conclusion and Future Directions

At the beginning of this dissertation, we have a discussion on the limitations of the conceptual categories of trustworthiness in the trustworthy ML literature: (1) the concept of each category can have multiple (even conflicting) quantitative formulations (e.g., different notions of fairness); and (2) some formulations of the concepts among different categories can be relevant or have tensions with each other (e.g., privacy versus fairness). This discussion motivates us to follow an operational procedure that takes a holistic but application-centric view to organize and approach the study of trustworthy ML. In particular, we identify trustworthiness issues of an ML model as the discrepancy of the model behaviors between a normal condition and an exceptional condition that the model may run into. Following this approach, this dissertation aims to investigate the trustworthiness issues of ML on graph data, where the complex relational structures lead to many complicated and sometimes implicit exceptional conditions in the context of GML, leading to unique challenges for trustworthy GML. From Chapter III to Chapter VI, we have identified three types of trustworthiness issues of GNNs, a popular family of ML models on graph data.

- In Chapter III, we investigated the exceptional conditions for GNNs in social media application scenarios when there are malicious hackers that aim to degrade the model performance by hacking into user accounts. We identified a

more practical adversarial attack setup for GNNs in such scenarios and demonstrated that GNNs are vulnerable in the proposed setup.

- In Chapter IV, we revealed that most prior GNNs have significantly degraded performance in an exceptional condition where the graph mainly plays a correlational role. We proposed a CopulaGNN framework to improve most GNNs for this scenario.

- In Chapter V and Chapter VI, we investigated the non-IID generalization ability of GNNs on nodes in different subgroups. Both theoretical and empirical results suggest that GNNs may suffer from degraded performance for nodes farther away from the training nodes, leading to fairness concerns. We also proposed an active learning approach to mitigate this problem.

Many graph data are highly socially-relevant. For instance, social networks and financial networks are networks of people; spatio-temporal data may be used to estimate crime rates or real estate prices. Non-robustness or unfairness of GML systems in these domains may lead to high-stakes consequences for individuals and society. The studies presented in this dissertation can be potentially applied to mitigate some of the real-world problems. For example, the adversarial vulnerability revealed in Chapter III could be practically relevant to real-world GNN applications on social networks and financial networks, as we have made realistic assumptions directly motivated by these scenarios. The correlational role of the graph emphasized in Chapter IV is prevalent in social network and spatio-temporal data. And the studies in Chapter V and Chapter VI may help us mitigate unfairness on social media applications without knowing protected sensitive attributes.

While the concrete case studies in this dissertation focus on the domain of GML, we believe the general procedure to systematically identify, understand, and mitigate the (un)trustworthiness issues could be helpful beyond GML. Finally, we briefly dis-

cuss a few future directions motivated by this dissertation in the following Section 7.1.

7.1 Future Directions

7.1.1 Evaluation of Trustworthy ML

As an empirical discipline, the development of ML models heavily relies on benchmark evaluations. Nevertheless, evaluating the trustworthiness of ML models appears to be tricky due to the complication of trustworthiness definitions. Motivated by the general procedure of identifying trustworthiness issues proposed in this dissertation, we foresee the following directions to advance the evaluation benchmarks for trustworthy ML.

Evaluation driven by exceptional conditions. The operational nature of the trustworthiness definitions induced by the aforementioned procedure provides a new and natural way to establish evaluation benchmarks for trustworthy ML methods. In particular, the procedure involves a step that finds normal and exceptional conditions. Once one can define or simulate these conditions, then we can establish evaluation criteria by contrasting the model performance between the two types of conditions.

Infrastructures for crowdsourcing benchmarks. As the trustworthiness issues are often highly application-dependent, it is therefore necessary to crowdsource benchmarks through a bottom-up approach. In order to leverage the community effort as much as possible, an important future direction is to develop infrastructures that can ease the contribution of evaluation benchmarks by individual researchers and practitioners. Following this line, we have initiated and successfully held a workshop, Graph Learning Benchmarks (GLB)[1], that calls to crowdsource machine learning benchmarks on graph data. We plan to continue this effort and further implement a

[1]We have successfully held the GLB workshop twice, respectively collocated with The Web Conference 2021 and 2022. Workshop website: `https://graph-learning-benchmarks.github.io/`.

long-term and evolving platform for GML benchmark dataset submission and curation.

7.1.2 Human-Machine Collaboration

Ultimately, trustworthy ML concerns how we can make ML systems trusted by humans, which influences how these ML systems can be deployed in reality. In more and more ML applications, the deployment of ML is not a static decision. Instead, the ML systems need to interact and co-operate with humans. For example, search and recommendation systems serve the purpose of helping users efficiently acquire relevant information; healthcare AI systems are mainly used to assist medical experts in making clinical decisions. As a result, it is essential to investigate and improve the trustworthiness of ML in the context of such dynamic human-machine collaboration systems, rather than the trustworthiness of ML models alone.

An interesting future direction would be extending the proposed trustworthiness definition in the aforementioned human-machine collaboration context. In particular, a few canonical paradigms of human-machine collaboration systems are commonly seen. One paradigm is *machine recommendation and human decision*, such as AI-assisted medical diagnostics or social media content moderation. Another paradigm is *machine augmented innovation*, such as the use of ML in scientific discovery or art creation. A straightforward extension of the trustworthiness definition might be a contrast between the utility of the human-machine collaboration system and the utility of human or ML alone. If the ML is designed to be trustworthy by humans, then we may expect improved utility for the collaboration system.

APPENDICES

APPENDIX A

Appendix of Chapter III

A.1 Proof of Proposition 1

We first remind the reader for some notations, a GCN model is denoted as a function f, the feature matrix is $X \in \mathbb{R}^{N\times D}$, and the output logits $H = f(X) \in \mathbb{R}^{N\times K}$. The L-step random walk transition matrix is M^L. More details can be found in in Section 6.3

We give in Lemma A.1.2 the connection between GCN models and random walks. Lemma A.1.2 relies on a technical assumption about the GCN model (Assumption A.1.1) and the proof can be found in Xu et al. [153].

Assumption A.1.1 (Xu et al. [153])**.** *All paths in the computation graph of the given GCN model are independently activated with the same probability of success ρ.*

Lemma A.1.2. *(Xu et al. [153].) Given an L-layer GCN with averaging as $\alpha_{i,j} = 1/d_i$ in Eq. 3.1, assume that all path in the computation graph of the model are activated with the same probability of success ρ (Assumption A.1.1). Then, for any node $i, j \in V$,*

$$\mathbb{E}\left[\frac{\partial H_j}{\partial X_i}\right] = \rho \cdot \prod_{l=L}^{1} \boldsymbol{W}_l [M^L]_{ji}, \tag{A.1}$$

where $\boldsymbol{W}_l$ is the learnable parameter at l-th layer.

Then we are able to prove Proposition 1 below.

Proof. First, we derive the gradient of the loss $\mathcal{L}(H, y)$ w.r.t. the feature X_i of node i,

$$\begin{aligned}\nabla_{X_i}\mathcal{L}(H,y) &= \nabla_{X_i}\left(\sum_{j=1}^{N}\mathcal{L}_j(H_j,y_j)\right)\\ &= \sum_{j=1}^{N}\nabla_{X_i}\mathcal{L}_j(H_j,y_j)\\ &= \sum_{j=1}^{N}\left(\frac{\partial H_j}{\partial X_i}\right)^T\frac{\partial\mathcal{L}_j(H_j,y_j)}{\partial H_j},\end{aligned} \tag{A.2}$$

where H_j is the jth row of H but being transposed as column vectors and y_j is the true label of node j. Note that $\frac{\partial\mathcal{L}_j(H_j,y_j)}{\partial H_j} \in \mathbb{R}^K$, and $\frac{\partial H_j}{\partial X_i} \in \mathbb{R}^{K\times D}$.

Next, we plug Eq. A.2 into $\widetilde{\Delta}_i(x)\,|_{x=\tau(X,\{i\})_i}$. For simplicity, We write $\widetilde{\Delta}_i(x)\,|_{x=\tau(X,\{i\})_i}$ as $\widetilde{\Delta}_i$ in the rest of the proof.

$$\begin{aligned}\widetilde{\Delta}_i &= \left(\nabla_{X_i}\mathcal{L}(H,y)\right)^T\epsilon\\ &= \sum_{j=1}^{N}\left(\frac{\partial\mathcal{L}_j(H_j,y_j)}{\partial H_j}\right)^T\frac{\partial H_j}{\partial X_i}\epsilon.\end{aligned} \tag{A.3}$$

Denote $a^j \triangleq \frac{\partial\mathcal{L}_j(H_j,y_j)}{\partial H_j} \in \mathbb{R}^K$. From the definition of loss

$$\mathcal{L}_j(H_j,y_j) = \sum_{j=1}^{N}\max_{k\in\{1,\ldots,K\}}H_{jk} - H_{jy_j},$$

we have

$$a_k^j = \begin{cases} -1, & \text{if } k = y_j \text{ and } y_j \neq \arg\max_{c\in\{1,\ldots,K\}} H_{jc},\\ 1, & \text{if } k \neq y_j \text{ and } k = \arg\max_{c\in\{1,\ldots,K\}} H_{jc},\\ 0, & \text{otherwise,}\end{cases}$$

for $k = 1, 2, \ldots, K$. Under Assumption 3.2.1, the expectation of each element of a^j is

$$\mathbb{E}[a_k^j] = -q_k(1 - p(k \mid k)) + \sum_{w=1, w\neq k}^{K} p(k \mid w) q_w, k = 1, 2, \ldots, K$$

which is a constant independent of H_j and y_j. Therefore, we can write

$$\mathbb{E}[a^j] = c, \forall j = 1, 2, \ldots, N,$$

where $c \in \mathbb{R}^K$ is a constant vector independent of j.

Taking expectation of Eq. (A.3) and plug in the result of Lemma A.1.2,

$$\begin{aligned}
\mathbb{E}\left[\widetilde{\Delta}_i\right] &\approx \mathbb{E}\left[\sum_{j=1}^{N} \left(\frac{\partial \mathcal{L}_j(H_j, y_j)}{\partial H_j}\right)^T \frac{\partial H_j}{\partial X_i} \epsilon\right] \\
&= \sum_{j=1}^{N} \mathbb{E}[a^j]^T \left(\rho \prod_{l=L}^{1} W_l [M^L]_{ji}\right) \epsilon \\
&= \left(\rho c^T \prod_{l=L}^{1} W_l \epsilon\right) \sum_{j=1}^{N} [M^L]_{ji} \\
&= C \sum_{j=1}^{N} [M^L]_{ji},
\end{aligned}$$

where $C = \rho c^T \prod_{l=L}^{1} W_l \epsilon$ is a constant scalar independent of i. □

A.2 Proofs for Propositions in Section 3.2.4

Proof of Proposition 2.

Proof. If $A_i = \emptyset$, $B_i \subseteq A_i$ so $B_i = \emptyset$. The three conditions of Definition 3.2.5 are also trivially true. Below we investigate the case $A_i \neq \emptyset$.

The existence can be given by a constructive proof. We check the nonempty elements in A_i one by one with any order. If this element is a super set of any other element in A_i, we skip it. Otherwise, we put it into B_i. Then we verify that the

resulted B_i is a basic vulnerable set for i. $B_i \subseteq A_i$. For condition 1), clearly, $\emptyset \notin B_i$ and if $\emptyset \in A_i$, all nonempty elements in A_i are skipped so $B_i = \emptyset$. For condition 2), given $\emptyset \notin A_i$, for any nonempty $S \in A_i$, if $S \in B_i$, the condition holds. If $S \notin B_i$, by construction, there exists a nonempty strict subset $S_1 \subset S$ and $S_1 \in A_i$. If $S_1 \in B_i$, the condition holds. If $S_1 \notin B_i$, we can similarly find a nonempty strict subset $S_2 \subset S$ and $S_2 \in A_i$. Recursively, we can get a series $S \supset S_1 \supset S_2 \supset \cdots$. As S is finite, we will have a set S_k that no longer has strict subset so $S_k \in B_i$. Therefore the condition holds. Condition 3) means any set in B_i is not a subset of another set in B_i. This condition holds by construction.

Now we prove the uniqueness. Suppose there are two distinct basic vulnerable sets $B_i \neq C_i$. Without loss of generality, we assume $S \in B_i$ but $S \notin C_i$. $B_i \neq \emptyset$ so $\emptyset \notin A_i$. Further $S \in A_i$, hence $C_i \neq \emptyset$. As $S \in B_i \subseteq A_i$, $S \neq \emptyset$, and C_i satisfies condition 2), there will be a nonempty $T \in C_i$ s.t. $T \subset S$. If $T \in B_i$, then condition 3) is violated for B_i. If $T \notin B_i$, there will be a nonempty $T' \in B_i$ s.t. $T' \subset T$. But $T' \subset S$ also violates condition 3). By contradiction we prove the uniqueness. □

In order to prove Proposition 3, we first would like to construct a submodular function that is close to h, with the help of Lemma A.2.1 below.

Lemma A.2.1. *If $\forall i \in V$, B_i is either empty or only contains singleton sets, then h is submodular.*

Proof. We first prove the case when $\forall i \in V, A_i \neq \emptyset$.

First, we show that $\forall i \in V$, if $A_i \neq \emptyset$, for any nonempty $S \subseteq V, g_i(S) = 1$ if and only if $B_i = \emptyset$ or $\exists T \in B_i, T \subseteq S$. On one hand, if $g_i(S) = 1$, then $S \in A_i$. If $\emptyset \in A_i, B_i = \emptyset$. If $\emptyset \notin A_i$, by condition 2) of the basic vulnerable set, $\exists T \in B_i, T \subseteq S$. On the other hand, if $\exists T \in B_i, T \subseteq S$, $g_i(T) = 1$, by Assumption 3.2.4, $g_i(S) \geq g_i(T)$, so $g_i(S) = 1$. If $B_i = \emptyset$, as $A_i \neq \emptyset$, if $\emptyset \notin A_i$, the condition 2) of Definition 3.2.5 will be violated. Therefore $\emptyset \in A_i$ so $g_i(\emptyset) = 1$. Still by Assumption 3.2.4, $g_i(S) \geq g_i(\emptyset)$,

so $g_i(S) = 1$.

Define a function $e : V \to 2^V$ s.t. for any node $i \in V$,

$$e(i) = \{j \in V \mid \{i\} \in B_j\}.$$

Given B_i is either empty or only contains singleton sets for any $i \in V$, for any nonempty $S \subseteq V$

$$\begin{aligned} h(S) &= \frac{1}{N} \sum_{i=1}^{N} g_i(S) \\ &= \frac{1}{N} \left|\{j \in V \mid B_j = \emptyset \text{ or } \exists T \in B_j, T \subseteq S\}\right| \\ &= \frac{1}{N} \left|\{j \in V \mid B_j = \emptyset \text{ or } \exists \{i\} \in B_j, i \in S\}\right| \\ &= \frac{1}{N} \left|\{j \in V \mid B_j = \emptyset \text{ or } \exists i \in S, \{i\} \in B_j\}\right| \\ &= \frac{1}{N} \left(\left|\cup_{i \in S} e(i)\right| + \left|\{j \in V \mid B_j = \emptyset\}\right|\right). \end{aligned} \tag{A.4}$$

$|\{j \in V \mid B_j = \emptyset\}|$ is a constant independent of S. Therefore, maximizing $h(S)$ over S with $|S| \leq r$ is equivalent to maximizing $|\cup_{i \in S} e(i)|$ over S with $|S| \leq r$, which is a maximum coverage problem. Therefore h is submodular.

The case of allowing some nodes to have empty vulnerable sets can be easily proved by removing such nodes in Eq. (A.4) as their corresponding vulnerable functions always equal to zero. □

Proof of Proposition 3. For simplicity, we assume $A_i \neq \emptyset$ for any $i \in V$. The proof below can be easily adapted to the general case without this assumption, by removing the nodes with empty vulnerable sets similarly as the proof for Lemma A.2.1.

Proof. $\forall i \in V$, define $\widetilde{B}_i \triangleq \{S \in B_i \mid |S| = 1\}$. We can then define a new group of

vulnerable sets $\widetilde{A}_i$ on V for $i \in V$. Let

$$\widetilde{A}_i = \begin{cases} 2^V, & \text{if } B_i = \emptyset, \\ \emptyset, & B_i \neq \emptyset \text{ but } \widetilde{B}_i = \emptyset, \\ \{S \subseteq V \mid \exists T \in \widetilde{B}_i, T \subseteq S\}, & \text{otherwise.} \end{cases}$$

Then it is clear that $\widetilde{B}_i$ is a valid basic vulnerable set corresponding to $\widetilde{A}_i$, for $i \in V$. If we define $\tilde{g}_i : 2^V \to \{0, 1\}$ as

$$\tilde{g}_i(S) = \begin{cases} 1, & \text{if } B_i = \emptyset \text{ or } \exists T \in \widetilde{B}_i, T \subseteq S, \\ 0, & \text{otherwise,} \end{cases}$$

we can easily verify that $\tilde{g}_i$ is a valid vulnerable function corresponding to $\widetilde{A}_i$, for $i \in V$. Further let $\tilde{h} : 2^V \to \mathbb{R}_+$ as

$$\tilde{h}(S) = \frac{1}{N} \sum_{i=1}^{N} \tilde{g}_i(S).$$

By Lemma A.2.1, as $\forall i \in V, \widetilde{B}_i$ is either empty or only contains singleton sets, we know $\tilde{h}$ is submodular.

Next we investigate the difference between h and $\tilde{h}$. First, for any $S \subseteq V$, if $S \notin \cup_{i=1}^{N} A_i$, clearly $h(S) = \tilde{h}(S) = 0$; if $|S| \leq 1$, it's easy to show $h(S) = \tilde{h}(S)$. Second, for any $S \in \cup_{i=1}^{N} A_i$ and $|S| > 1$, by Assumption 3.2.6, there are exactly b (omitting the S in $b(S)$) nodes whose vulnerable set contains S. Without loss of generality, let us assume the indexes of b nodes are $1, 2, \ldots, b$. Then, for any node $i > b$, $g_i(S) = 0, \tilde{g}_i(S) = 0$. For node $i = 1, 2, \ldots, b$, $g_i(S) = 1$, and

$$\tilde{g}_i(S) = \begin{cases} 1, & \text{if } B_i = \emptyset \text{ or } \exists T \subseteq S, |T| = 1 \text{ and } T \in \widetilde{B_i}, \\ 0, & \text{otherwise.} \end{cases}$$

By Assumption 3.2.7, there are at least $\lceil pb \rceil$ (omitting the S in $p(S)$) nodes like j s.t. $\tilde{g}_j(S) = 1$. Therefore, $h(S) = \frac{b}{N}$ and $\frac{\lceil pb \rceil}{N} \leq \tilde{h}(S) \leq \frac{b}{N}$. Hence $1 - \frac{1}{r} < 1 \leq \frac{h(S)}{\tilde{h}(S)} \leq \frac{1}{p} < 1 + \frac{1}{r}$. □

A.3 Algorithm Details of GC-RWCS

We summarize the GC-RWCS strategy in Algorithm A.1.

Algorithm A.1: The GC-RWCS Strategy for Node Selection.

Input: number of nodes limit r; maximum degree limit m; neighbor hops k; binarized transition matrix $\widetilde{M}$; the adaptive influence score function $\widetilde{I}_i, \forall i \in V$.

Output: the set S to be attacked.

Initialize the candidate set $P = \{i \in V \mid d_i \leq m\}$, and the score matrix $Q = \widetilde{M}$;

Initialize $S = \emptyset$;

for $t = 1, 2, \ldots, r$ **do**

  $z \leftarrow \arg\max_{i \in P} \widetilde{I}_i(Q)$;

  $S \leftarrow S \cup \{z\}$;

  $P \leftarrow P \setminus \{i \in P \mid \text{shortest-path}(i, z) \leq k\}$;

  $q \leftarrow Q_{\cdot,z}$;

  for $i \in V$ **do**

    if q_i *is* 1 **then**

      $Q_i \leftarrow \mathbf{0}$;

    end

  end

end

return S;

A.4 Additional Experiment Details

Datasets. We adopt the Deep Graph Library [145] version of Cora, Citeseer, and Pubmed in our experiments. The summary statistics of the datasets are summarized in Table A.1. The number of edges does not include self-loops.

Table A.1: Summary statistics of datasets.

Dataset	Nodes	Edges	Classes	Features
Citeseer	3,327	4,552	6	3,703
Cora	2,708	5,278	7	1,433
Pubmed	19,717	44,324	3	500

A.5 Additional Experiment Results

In this section, we provide results of more experiment setups and conduct a sensitivity analysis of the hyper-parameter L in GC-RWCS in Table A.2. We provide a setup of 20% threshold in addition to the 10% and 30% thresholds shown in Section 3.3.2, to give a better resolution of the results. And the results of threshold 20% are consistent with other setups. We also show the results of GC-RWCS with $L = 3, 4, 5, 6, 7$. Note that GCN has 2 layers and the JK-Nets have 7 layers. The variations of GC-RWCS results with the provided range of L are typically within 2%, indicating that the proposed GC-RWCS strategy does not rely on the exact knowledge of number of layers in the GNN models to be effective.

Further, we also compare the relative decrease of accuracy between the proposed GC-RWCS strategy ($L = 4$) and the Random strategy in Table A.3. GC-RWCS is able to decrease the node classification accuracy by up to 33.5%, and achieves a 70% larger decrease of the accuracy than the Random baseline in most cases. As the GC-RWCS and Random use exactly the same feature perturbation and the node selection step of Random does not include any information of the graph structure, this relative comparison can be roughly viewed as an indicator of the attack effectiveness

Table A.2: Summary of the accuracy (in %) when $L = \{3, 4, 5, 6, 7\}$. The **bold number** and the asterisk (*) denotes the same meaning as Table 3.1. The underline marker denotes the values of GC-RWCS outperforms all the baseline.

	Cora			Citeseer			Pubmed		
Method	GCN	JKNetConcat	JKNetMaxpool	GCN	JKNetConcat	JKNetMaxpool	GCN	JKNetConcat	JKNetMaxpool
None	85.6 ± 0.3	86.2 ± 0.2	85.8 ± 0.3	75.1 ± 0.2	72.9 ± 0.3	73.2 ± 0.3	85.7 ± 0.1	85.8 ± 0.1	85.7 ± 0.1
Threshold 10%									
Random	81.3 ± 0.3	68.8 ± 0.8	68.8 ± 1.3	71.3 ± 0.3	60.8 ± 0.8	61.7 ± 0.9	82.0 ± 0.3	75.9 ± 0.7	75.2 ± 0.7
Degree	78.2 ± 0.4	60.7 ± 1.0	59.9 ± 1.5	67.5 ± 0.4	52.5 ± 0.8	53.7 ± 1.0	78.9 ± 0.5	63.4 ± 1.0	63.2 ± 1.2
Pagerank	79.4 ± 0.4	71.6 ± 0.6	70.0 ± 1.0	70.1 ± 0.3	61.5 ± 0.5	62.6 ± 0.6	80.3 ± 0.3	71.3 ± 0.8	71.2 ± 0.8
Betweenness	79.7 ± 0.4	60.5 ± 0.9	60.3 ± 1.6	68.9 ± 0.3	53.5 ± 0.8	55.1 ± 1.0	78.5 ± 0.6	67.1 ± 1.1	66.1 ± 1.1
RWCS	79.4 ± 0.4	71.7 ± 0.5	70.3 ± 0.9	69.9 ± 0.3	62.4 ± 0.4	63.1 ± 0.6	79.8 ± 0.3	70.7 ± 0.8	70.7 ± 0.8
GC-RWCS-3	78.6 ± 0.5	**52.1** ± 1.1*	**53.0** ± 1.9*	**64.8** ± 0.5*	**46.4** ± 0.8*	**48.2** ± 1.0*	78.1 ± 0.6	62.3 ± 1.2	61.6 ± 1.5
GC-RWCS-4	78.5 ± 0.5	52.7 ± 1.0*	53.3 ± 1.9*	65.1 ± 0.5*	46.6 ± 0.8*	48.2 ± 1.1*	**77.3** ± 0.7	**62.1** ± 1.2	**60.6** ± 1.4*
GC-RWCS-5	78.9 ± 0.5	53.5 ± 1.1*	54.2 ± 1.9*	65.3 ± 0.5*	46.6 ± 0.8*	48.4 ± 1.0*	78.4 ± 0.5	64.2 ± 1.2	62.5 ± 1.4
GC-RWCS-6	78.5 ± 0.5	54.3 ± 1.1*	54.9 ± 1.9*	65.5 ± 0.5*	47.1 ± 0.8	48.9 ± 1.1*	78.0 ± 0.6	63.7 ± 1.1	62.6 ± 1.4
GC-RWCS-7	**78.1** ± 0.5	54.2 ± 1.1*	54.8 ± 1.9*	66.1 ± 0.4*	47.5 ± 0.8	49.3 ± 1.1*	78.7 ± 0.5	64.9 ± 1.2	63.3 ± 1.3
Threshold 20%									
Random	82.3 ± 0.3	71.7 ± 1.1	69.8 ± 1.1	72.1 ± 0.3	62.1 ± 0.7	62.6 ± 0.9	82.6 ± 0.2	77.9 ± 0.5	77.5 ± 0.5
Degree	**79.3** ± 0.4	64.2 ± 1.2	61.6 ± 1.3	69.2 ± 0.4	56.0 ± 0.8	56.4 ± 1.0	80.6 ± 0.4	69.5 ± 0.8	69.4 ± 1.0
Pagerank	80.8 ± 0.3	74.5 ± 0.8	73.0 ± 0.8	72.1 ± 0.3	68.3 ± 0.3	68.2 ± 0.4	82.2 ± 0.2	77.7 ± 0.4	77.8 ± 0.4
Betweenness	80.7 ± 0.4	62.2 ± 1.4	60.1 ± 1.4	70.1 ± 0.4	54.8 ± 0.8	55.8 ± 1.1	80.2 ± 0.4	72.4 ± 0.8	72.0 ± 0.7
RWCS	81.4 ± 0.3	76.8 ± 0.6	76.0 ± 0.6	72.4 ± 0.3	68.9 ± 0.3	69.0 ± 0.4	81.3 ± 0.2	76.0 ± 0.4	76.5 ± 0.4
GC-RWCS-3	79.4 ± 0.5	**57.5** ± 1.6*	**53.1** ± 1.5*	**67.1** ± 0.4*	48.4 ± 0.9*	49.3 ± 1.2*	**79.0** ± 0.5*	**67.4** ± 0.9*	**66.3** ± 1.0*
GC-RWCS-4	79.4 ± 0.5	57.5 ± 1.7*	53.2 ± 1.4*	67.3 ± 0.5*	**47.9** ± 0.9*	**48.8** ± 1.3*	**79.0** ± 0.5*	67.4 ± 1.0*	**66.3** ± 1.0*
GC-RWCS-5	79.4 ± 0.5	59.0 ± 1.7*	54.5 ± 1.4*	67.3 ± 0.4*	48.4 ± 0.9*	49.4 ± 1.3*	79.2 ± 0.5*	68.5 ± 0.9	68.1 ± 0.9
GC-RWCS-6	79.5 ± 0.5	59.3 ± 1.7	54.9 ± 1.5*	68.1 ± 0.4*	49.2 ± 0.9*	50.2 ± 1.3*	79.1 ± 0.5*	68.4 ± 0.9	68.5 ± 1.0
GC-RWCS-7	79.4 ± 0.5	59.3 ± 1.6	55.3 ± 1.5*	68.1 ± 0.4*	50.0 ± 0.9*	50.8 ± 1.3*	79.2 ± 0.5*	68.7 ± 0.9	68.2 ± 0.8
Threshold 30%									
Random	82.6 ± 0.4	70.7 ± 1.1	71.8 ± 1.1	72.6 ± 0.3	62.7 ± 0.8	63.9 ± 0.8	82.6 ± 0.2	77.3 ± 0.4	77.3 ± 0.5
Degree	80.7 ± 0.4	64.9 ± 1.4	67.0 ± 1.5	70.4 ± 0.4	56.9 ± 0.8	58.7 ± 0.9	81.5 ± 0.4	72.4 ± 0.7	72.1 ± 0.8
Pagerank	82.6 ± 0.3	79.6 ± 0.4	79.7 ± 0.4	72.9 ± 0.2	70.2 ± 0.3	70.3 ± 0.3	83.0 ± 0.2	79.3 ± 0.3	79.5 ± 0.3
Betweenness	81.8 ± 0.4	64.1 ± 1.3	65.9 ± 1.4	70.7 ± 0.3	56.3 ± 0.8	58.3 ± 0.9	81.3 ± 0.3	74.1 ± 0.5	74.5 ± 0.5
RWCS	82.9 ± 0.3	79.7 ± 0.4	80.0 ± 0.4	72.9 ± 0.2	70.2 ± 0.3	70.4 ± 0.3	82.1 ± 0.2	77.8 ± 0.3	78.4 ± 0.3
GC-RWCS-3	**80.2** ± 0.6	**57.3** ± 1.7*	**59.0** ± 1.6*	67.9 ± 0.5*	49.1 ± 0.9*	50.8 ± 1.1*	80.3 ± 0.5*	**69.0** ± 0.7*	**69.8** ± 0.7*
GC-RWCS-4	80.7 ± 0.5	59.1 ± 1.6*	61.1 ± 1.6*	**67.8** ± 0.5*	**49.0** ± 0.9*	**50.7** ± 1.1*	80.3 ± 0.5*	69.2 ± 0.7*	70.0 ± 0.7*
GC-RWCS-5	80.8 ± 0.5	59.8 ± 1.6*	61.5 ± 1.6*	68.4 ± 0.5*	49.2 ± 0.9*	51.2 ± 1.1*	**80.2** ± 0.5*	70.4 ± 0.6*	71.5 ± 0.6
GC-RWCS-6	80.7 ± 0.5	59.8 ± 1.5*	61.4 ± 1.5*	68.5 ± 0.5*	50.5 ± 0.9*	52.2 ± 1.1*	**80.2** ± 0.5*	70.5 ± 0.5*	71.6 ± 0.6
GC-RWCS-7	80.7 ± 0.5	60.2 ± 1.5*	61.9 ± 1.5*	68.7 ± 0.5*	50.7 ± 0.9*	52.6 ± 1.1*	80.3 ± 0.4*	70.9 ± 0.5*	71.9 ± 0.6

attributed to the structural inductive biases of the GNN models.

Table A.3: Accuracy decrease (in %) comparison with clean dataset

	Cora			Citeseer			Pubmed		
Method	GCN	JKNetConcat	JKNetMaxpool	GCN	JKNetConcat	JKNetMaxpool	GCN	JKNetConcat	JKNetMaxpool
				Threshold 10%					
Random	4.3	17.4	17	3.8	12.1	11.5	3.7	9.9	10.3
GC-RWCS	7.1	33.5	32.5	10.0	26.3	25.0	8.4	23.7	25.1
GC-RWCS/Random	165.12%	192.53%	191.18%	263.16%	217.36%	217.39%	227.03%	239.39%	243.69%
				Threshold 30%					
Random	3.0	15.5	14	2.5	10.2	9.3	3.1	8.5	8.3
GC-RWCS	4.9	27.1	24.7	7.3	23.9	22.5	5.4	16.6	15.7
GC-RWCS/Random	163.33%	174.84%	176.43%	292.00%	234.31%	241.94%	174.19%	195.29%	189.16%

APPENDIX B

Appendix of Chapter IV

B.1 Experiment Details

B.1.1 Details of Synthetic Data Generation

We generate the synthetic data with the following procedure.

1. Sample a node feature matrix $\boldsymbol{X} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_{d_0})$, where $\boldsymbol{X} \in \mathbb{R}^{n \times d_0}$ and d_0 is the feature dimension.
2. Generate the graph in a way similarly to a latent space model [59]. First compute the latent variables $\boldsymbol{Z} = \boldsymbol{X}\boldsymbol{W}_g$, where $\boldsymbol{W}_g \in \mathbb{R}^{d_0 \times d_1}$ is a given weight matrix. Then calculate the latent distance $\|\boldsymbol{z}_i - \boldsymbol{z}_j\|_2$ for each node pair $(i, j), 1 \leq i < j \leq n$. Finally assign edges between the m pairs of nodes with the shortest latent distances to form a graph with n nodes and m edges.
3. Assume $\boldsymbol{A}$ is the adjacency matrix of the graph, $\boldsymbol{D}$ is the degree matrix, and $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}$ is the graph Laplacian. Let $\tilde{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{I}$ and $\tilde{\boldsymbol{D}} = \boldsymbol{D} + \boldsymbol{I}$. Given parameters $\boldsymbol{w}_y \in \mathbb{R}^{d_0}$, generate the node label vector $\boldsymbol{y} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where, for some $\gamma > 0, \tau > 0$, and $\sigma^2 > 0$,
   (a) $\boldsymbol{\mu} = \tilde{\boldsymbol{D}}^{-1}\tilde{\boldsymbol{A}}\boldsymbol{X}\boldsymbol{w}_y$, $\boldsymbol{\Sigma} = \sigma^2\boldsymbol{I}$;
   (b) $\boldsymbol{\mu} = \boldsymbol{X}\boldsymbol{w}_y$, $\boldsymbol{\Sigma} = \tau(\boldsymbol{L} + \gamma\boldsymbol{I})^{-1}$;
   (c) $\boldsymbol{\mu} = \tilde{\boldsymbol{D}}^{-1}\tilde{\boldsymbol{A}}\boldsymbol{X}\boldsymbol{w}_y$, $\boldsymbol{\Sigma} = \tau(\boldsymbol{L} + \gamma\boldsymbol{I})^{-1}$.

B.1.2 Simulation Details for Section 4.3

For each configuration we randomly generate 100 datasets with different seeds. For each dataset, we randomly split the nodes into training, validation, and test sets equally to form a semi-supervised learning task.

We set the number of layers as 2 and the total hidden units as 16 for all models. We use the Adam optimizer [76] with an initial learning rate of 0.01 to train all models by minimizing the MSE loss, with early stopping on the validation set. Finally, we report the R^2 score on the test set.

B.1.3 Experiment Details for Section 4.5

Training details. For all neural networks involved in the experiments, we set the number of layers as 2 and the number of hidden units as 16. We use the Adam optimizer to train all the models and apply early stopping on a validation set. For the real-world datasets, the initial learning rate is chosen from $\{0.01, 0.001\}$ on the validation set.

The U.S. Election dataset. The nodes in the election data are U.S. counties and edges connect adjacent counties on the map. Each county is associated with demographic and election statistics. In each of the regression tasks, one statistic is selected as the node outcome and the remaining statistics are used as the node features. The four regression tasks are named by the outcome statistics: Education, Election, Income, Unemployment. We randomly split the data into training, validation, and test sets with ratio 6:2:2 following Jia and Benson [67], and we refer to their work for more details of the datasets.

The Wikipedia datasets. Each of the two Wikipedia datasets consists of a graph on Wikipedia, where each node is a Wikipedia page related to the animal of the page title and edges reflect mutual hyper-links between the pages. The node features are principal components of binary indicators for the presence of certain nouns. The count

outcome variable of each node label is the monthly traffic in the unit of thousands.

The EMNLP dataset. This dataset is constructed from the DBLP citation data provided by AMiner [137]. We first extract a set of papers published on the EMNLP conference and treat each paper as a node. Then we construct a graph where two papers have an edge if they are *cited simultaneously* by at least two EMNLP papers. The node features are principal components of the bag-of-words of paper titles and abstracts as well as the year of publication. The node label is the number of citations of each paper *from outside* EMNLP. For both types of datasets, we randomly split the data into training, validation, and test sets with ratio 1:1:1.

B.2 More Details about Copulas

B.2.1 Two-Dimensional Examples

Figure B.1 shows PDFs of two-dimensional distributions constructed using different parametric copulas; the marginal distributions are all standard normal.

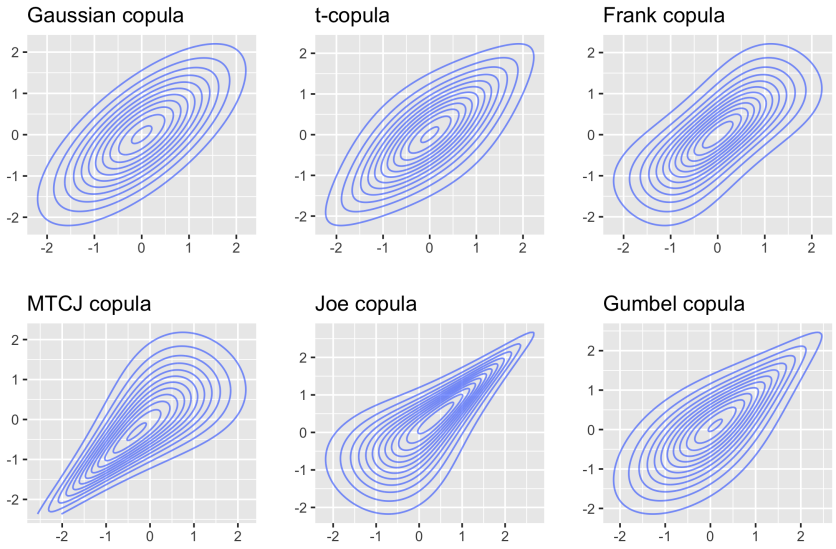


Figure B.1: Density functions of two-dimensional distributions constructed using copulas. The marginal distributions are all standard normal. See Joe [71] for the definitions of these parametric copulas.

B.2.2 Approximating the Copulas for Discrete Random Variables

If the random vector $\boldsymbol{Y}$ is discrete, the copula representation of its PMF is more complex. Take $n = 2$ as an example, the PMF of $\boldsymbol{Y} = (Y_1, Y_2)$ is

$$\begin{aligned} f(\boldsymbol{y}) &= \mathbb{P}(Y_1 = y_1, Y_2 = y_2) \\ &= \mathbb{P}(Y_1 \leq y_1, Y_2 \leq y_2) - \mathbb{P}(Y_1 < y_1, Y_2 \leq y_2) - \mathbb{P}(Y_1 \leq y_1, Y_2 < y_2) + \mathbb{P}(Y_1 < y_1, Y_2 < y_2) \\ &= C(u_{12}, u_{22}) - C(u_{11}, u_{22}) - C(u_{12}, u_{21}) + C(u_{11}, u_{21}), \end{aligned}$$

where $u_{i1} = \lim_{x \to y_i^-} F_i(x) = F_i(y_i^-)$ and $u_{i2} = F_i(y_i)$. In general, the PMF has the following form:

$$f(\boldsymbol{y}) = \sum_{j_1=1}^{2} \cdots \sum_{j_n=1}^{2} (-1)^{j_1 + \cdots + j_n} C(u_{1j_1}, \ldots, u_{nj_n}), \tag{B.1}$$

which is computationally intractable because there are 2^n summands. Kazianka and Pilz [74] propose an approximation of the above PMF based on the generalized quantile transform. It smooths the CDF of ordinal discrete variables from a step function to a piece-wise linear one:

$$f(\boldsymbol{y}) \approx c(v_1, \ldots, v_n) \prod_{i=1}^{n} f_i(y_i), \tag{B.2}$$

where $v_i = (u_{i1} + u_{i2})/2$. It has been shown that the approximation works well as long as the marginal variance is not too small. We apply this method to approximate PMF to handle discrete random variables.

APPENDIX C

Appendix of Chapter V

C.1 Proofs

C.1.1 Proof of Theorem 5.4.2

We first introduce three lemmas whose proofs can be found in the referred literature.

Lemma C.1.1 (Hoeffding's Inequality for Bounded Random Variables [58]). *Suppose $X_1, X_2, \ldots, X_n$ are independent random variables with $a_i \leq X_i \leq b_i, \forall i = 1, 2 \ldots, n$. Let $\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$. Then, for any $t > 0$,*

$$\Pr\big(|\bar{X} - \mathbb{E}\bar{X}| > t\big) \leq 2e^{-\frac{n^2t^2}{\sum_{i=1}^{n}(b_i-a_i)^2}}.$$

Lemma C.1.2 (Sub-Gaussianity). *If X is a centered random variable, i.e., $\mathbb{E}X = 0$, and if $\exists \nu > 0$, for any $t > 0$,*

$$\Pr(|X| > t) \leq 2e^{-\nu t^2}.$$

Then, for any $\lambda > 0$,

$$\mathbb{E}e^{\lambda X} \leq e^{\frac{\lambda^2}{2\nu}}.$$

See Rivasplata [121] (Theorem 3.1) for the proof of Lemma C.1.2.

Lemma C.1.3 (Change-of-Measure Inequality, Lemma 17 in Germain et al. [50])**.** *For any two distributions P and Q defined on $\mathcal{H}$, and any function $\psi : \mathcal{H} \to \mathbb{R}$,*

$$\mathbb{E}_{h\sim Q}[\psi(h)] \leq D_{\mathrm{KL}}(Q\|P) + \ln \mathbb{E}_{h\sim P}[e^{\psi(h)}].$$

Then we can prove Theorem 5.4.2. For convenience, we re-state it as Theorem C.1.4 below.

Theorem C.1.4 (Subgroup Generalization of Stochastic Classifiers)**.** *For any $0 < m \leq M$, for any $\lambda > 0$ and $\gamma \geq 0$, for any "prior" distribution P on $\mathcal{H}$ that is independent of the training data on V_0, with probability at least $1-\delta$ over the sample of y^0, for any Q on $\mathcal{H}$, we have*[1]

$$\mathcal{L}_m^{\gamma/2}(Q) \leq \widehat{\mathcal{L}}_0^{\gamma}(Q) + \frac{1}{\lambda}\left(D_{\mathrm{KL}}(Q\|P) + \ln\frac{1}{\delta} + \frac{\lambda^2}{4N_0} + D_{m,0}^{\gamma}(P;\lambda)\right). \tag{C.1}$$

Proof. We prove the result by upper-bounding the quantity $\lambda(\mathcal{L}_m^{\gamma/2}(Q) - \widehat{\mathcal{L}}_0^{\gamma}(Q))$. First, we have

$$\begin{aligned}
&\lambda(\mathcal{L}_m^{\gamma/2}(Q) - \widehat{\mathcal{L}}_0^{\gamma}(Q)) \\
\leq &\mathbb{E}_{h\sim Q}\lambda(\mathcal{L}_m^{\gamma/2}(h) - \widehat{\mathcal{L}}_0^{\gamma}(h)) \\
\leq &D_{\mathrm{KL}}(Q\|P) + \ln \mathbb{E}_{h\sim P} e^{\lambda\left(\mathcal{L}_m^{\gamma/2}(h) - \widehat{\mathcal{L}}_0^{\gamma}(h)\right)},
\end{aligned} \tag{C.2}$$

where the first inequality is due to Jensen's inequality, and the last inequality is due to Lemma C.1.3.

Next we would like to upper-bound the second term in the RHS of (C.2). Note that the quantity $U := \mathbb{E}_{h\sim P} e^{\lambda\left(\mathcal{L}_m^{\gamma/2}(h) - \widehat{\mathcal{L}}_0^{\gamma}(h)\right)}$ is a random variable with the randomness

[1]Theorem C.1.4 also holds when we substitute $\mathcal{L}_m^{\gamma/2}(h)$ and $\mathcal{L}_m^{\gamma/2}(Q)$ as $\mathcal{L}_m^{\gamma}(h)$ and $\mathcal{L}_m^{\gamma}(Q)$ respectively. But we state the theorem in this form to ease the development of the later analysis.

coming from the sample of node labels y^0 for V_0. Also note that P is independent of y^0. Applying Markov's inequality to U, we have for any $\delta > 0$, with probability at least $1 - \delta$ over the sample of y^0,

$$U \leq \frac{1}{\delta}\mathbb{E}_{y^0}U,$$

and hence,

$$\ln U \leq \ln \frac{1}{\delta}\mathbb{E}_{y^0}U = \ln \frac{1}{\delta} + \ln \mathbb{E}_{y^0}U.$$

Then we need to upper-bound the quantity $\ln \mathbb{E}_{y^0}U$. We can re-write it as

$$\ln \mathbb{E}_{y^0}U = \ln \mathbb{E}_{y^0}\mathbb{E}_{h\sim P}e^{\lambda\left(\mathcal{L}_m^{\gamma/2}(h)-\widehat{\mathcal{L}}_0^{\gamma}(h)\right)} = \ln \mathbb{E}_{h\sim P}\mathbb{E}_{y^0}e^{\lambda\left(\mathcal{L}_m^{\gamma/2}(h)-\widehat{\mathcal{L}}_0^{\gamma}(h)\right)}. \tag{C.3}$$

For a fixed model h,

$$\begin{aligned}
&\mathbb{E}_{y^0}e^{\lambda\left(\mathcal{L}_m^{\gamma/2}(h)-\widehat{\mathcal{L}}_0^{\gamma}(h)\right)}\\
=&\mathbb{E}_{y^0}e^{\lambda\left(\mathcal{L}_m^{\gamma/2}(h)-\mathcal{L}_0^{\gamma}(h)+\mathcal{L}_0^{\gamma}(h)-\widehat{\mathcal{L}}_0^{\gamma}(h)\right)}\\
=&\mathbb{E}_{y^0}e^{\lambda\left(\mathcal{L}_m^{\gamma/2}(h)-\mathcal{L}_0^{\gamma}(h)\right)}e^{\lambda\left(\mathcal{L}_0^{\gamma}(h)-\widehat{\mathcal{L}}_0^{\gamma}(h)\right)}\\
=&e^{\lambda\left(\mathcal{L}_m^{\gamma/2}(h)-\mathcal{L}_0^{\gamma}(h)\right)}\mathbb{E}_{y^0}e^{\lambda\left(\mathcal{L}_0^{\gamma}(h)-\widehat{\mathcal{L}}_0^{\gamma}(h)\right)}.
\end{aligned} \tag{C.4}$$

In the following we will give an upper bound on $\mathbb{E}_{y^0}e^{\lambda\left(\mathcal{L}_0^{\gamma}(h)-\widehat{\mathcal{L}}_0^{\gamma}(h)\right)}$ that is independent of h. Recall that

$$\widehat{\mathcal{L}}_0^{\gamma}(h) = \frac{1}{N_0}\sum_{i\in V_0}\mathbb{1}h_i(X,G)[y_i] \leq \gamma + \max_{k\neq y_i}h_i(X,G)[k],$$

where the node labels are independently sampled (though not from the identical distribution), so $\widehat{\mathcal{L}}_0^{\gamma}(h)$ is the empirical mean of N_0 independent Bernoulli random

variables and $\mathcal{L}_0^\gamma(h)$ is the expectation of $\widehat{\mathcal{L}}_0^\gamma(h)$. By Lemma C.1.1, for any $t > 0$,

$$\Pr\Big(|\mathcal{L}_0^\gamma(h) - \widehat{\mathcal{L}}_0^\gamma(h)| \geq t\Big) \leq 2e^{-2N_0 t^2},$$

and hence $\mathcal{L}_0^\gamma(h) - \widehat{\mathcal{L}}_0^\gamma(h)$ is sub-Gaussian. Further by Lemma C.1.2, we have

$$\mathbb{E}_{y^0} e^{\lambda\left(\mathcal{L}_0^\gamma(h) - \widehat{\mathcal{L}}_0^\gamma(h)\right)} \leq e^{\frac{\lambda^2}{4N_0}},$$

which implies that

$$\mathbb{E}_{y^0} e^{\lambda\left(\mathcal{L}_0^\gamma(h) - \widehat{\mathcal{L}}_0^\gamma(h)\right)} \leq e^{\frac{\lambda^2}{4N_0}}, \tag{C.5}$$

Therefore, plugging (C.4) and (C.5) back into (C.3), we have

$$\begin{aligned}
&\ln \mathbb{E}_{y^0} U \\
\leq &\ln \mathbb{E}_{h\sim P} e^{\lambda\left(\mathcal{L}_m^{\gamma/2}(h) - \mathcal{L}_0^\gamma(h)\right)} e^{\frac{\lambda^2}{4N_0}} \\
= &D_{m,0}^\gamma(P;\lambda) + \frac{\lambda^2}{4N_0}.
\end{aligned}$$

Finally, plugging everything back into (C.2), we get

$$\begin{aligned}
&\lambda(\mathcal{L}_m^{\gamma/2}(Q) - \widehat{\mathcal{L}}_0^\gamma(Q)) \\
\leq &D_{\mathrm{KL}}(Q\|P) + \ln \mathbb{E}_{h\sim P} e^{\lambda\left(\mathcal{L}_m^{\gamma/2}(h) - \widehat{\mathcal{L}}_0^\gamma(h)\right)} \\
\leq &D_{\mathrm{KL}}(Q\|P) + \ln\frac{1}{\delta} + \frac{\lambda^2}{4N_0} + D_{m,0}^\gamma(P;\lambda).
\end{aligned}$$

Rearranging the terms gives us the final result

$$\mathcal{L}_m^{\gamma/2}(Q) \leq \widehat{\mathcal{L}}_0^\gamma(Q) + \frac{1}{\lambda}\left(D_{\mathrm{KL}}(Q\|P) + \ln\frac{1}{\delta} + \frac{\lambda^2}{4N_0} + D_{m,0}^\gamma(P;\lambda)\right).$$

□

C.1.2 Proof of Theorem 5.4.3

We re-state Theorem 5.4.3 as Theorem C.1.5 below.

Theorem C.1.5 (Subgroup Generalization of Deterministic Classifiers)**.** *Let $\tilde{h}$ be any classifier in $\mathcal{H}$. For any $0 < m \leq M$, for any $\lambda > 0$ and $\gamma \geq 0$, for any "prior" distribution P on $\mathcal{H}$ that is independent of the training data on V_0, with probability at least $1-\delta$ over the sample of y^0, for any Q on $\mathcal{H}$ such that $\Pr_{h\sim Q}\left(\max_{i\in V_0\cup V_m} \|h_i(X,G) - \tilde{h}_i(X,G)\|_\infty < \frac{\gamma}{8}\right) \geq \frac{1}{2}$, we have*

$$\mathcal{L}_m^0(\tilde{h}) \leq \widehat{\mathcal{L}}_0^\gamma(\tilde{h}) + \frac{1}{\lambda}\left(2(D_{\mathrm{KL}}(Q\|P)+1) + \ln\frac{1}{\delta} + \frac{\lambda^2}{4N_0} + D_{m,0}^{\gamma/2}(P;\lambda)\right). \tag{C.6}$$

Proof. For simplicity, we write $h_i(X,G)$ and $\tilde{h}_i(X,G)$ as h_i and $\tilde{h}_i$ in this proof. We first construct a distribution Q' by restricting Q on $\mathcal{H}_{\tilde{h}} \subseteq \mathcal{H}$, where

$$\mathcal{H}_{\tilde{h}} := \{h \in \mathcal{H} \mid \max_{i\in V_0\cup V_m} \|h_i - \tilde{h}_i\|_\infty < \frac{\gamma}{8}\}.$$

And Q' is defined as

$$Q'(h) = \begin{cases} \frac{1}{Z_{Q'}}Q(h), & \text{if } h \in \mathcal{H}_{\tilde{h}} \\ 0, & \text{otherwise} \end{cases},$$

where $Z_{Q'} = \Pr_{h\sim Q}(h \in \mathcal{H}_{\tilde{h}}) \geq \frac{1}{2}$ by the condition of the theorem.

For any $h \in \mathcal{H}_{\tilde{h}}$ and any sample $i \in V_0 \cup V_m$, by definition of $\mathcal{H}_{\tilde{h}}$, we have

$$\max_{k,k'\in\{1,\ldots,K\}} |(\tilde{h}_i[k] - \tilde{h}_i[k']) - (h_i[k] - h_i[k'])| < \frac{\gamma}{4},$$

which implies the following relationships:

$$\mathcal{L}_m^0(\tilde{h}) \leq \mathcal{L}_m^{\gamma/4}(h), \quad \widehat{\mathcal{L}}_0^{\gamma/2}(h) \leq \widehat{\mathcal{L}}_0^\gamma(\tilde{h}).$$

Therefore, with probability at least $1-\delta$ over the sample of y^m,

$$\begin{aligned}
&\mathcal{L}_m^0(\tilde{h}) \\
\leq &\mathbb{E}_{h\sim Q'}\mathcal{L}_m^{\gamma/4}(h) \\
\leq &\mathbb{E}_{h\sim Q'}\widehat{\mathcal{L}}_0^{\gamma/2}(h) + \frac{1}{\lambda}\left(D_{\mathrm{KL}}(Q'\|P) + \ln\frac{1}{\delta} + \frac{\lambda^2}{4N_0} + D_{m,0}^{\gamma/2}(P;\lambda)\right) \\
\leq &\widehat{\mathcal{L}}_0^{\gamma}(\tilde{h}) + \frac{1}{\lambda}\left(D_{\mathrm{KL}}(Q'\|P) + \ln\frac{1}{\delta} + \frac{\lambda^2}{4N_0} + D_{m,0}^{\gamma/2}(P;\lambda)\right),
\end{aligned}$$

where the second inequality is due to the application of Theorem 5.4.2 by substituting γ as $\gamma/2$ and Q as Q'.

Finally, to complete the proof, we only need to show

$$D_{\mathrm{KL}}(Q'\|P) \leq 2(D_{\mathrm{KL}}(Q\|P)+1).$$

Denote $\mathcal{H}_{\tilde{h}}^c$ as the complement of $\mathcal{H}_{\tilde{h}}$ and define Q'^c as the distribution restricted to $\mathcal{H}_{\tilde{h}}^c$ similarly as Q'. Define $H(x) := -x\ln x - (1-x)\ln(1-x)$, which is the binary entropy function and we know $H(Z)\leq 1$. Then

$$\begin{aligned}
D_{\mathrm{KL}}(Q\|P) &= \int_{\mathcal{H}_{\tilde{h}}} \ln\frac{dQ}{dP}dQ + \int_{\mathcal{H}_{\tilde{h}}^c} \ln\frac{dQ}{dP}dQ \\
&= Z_{Q'}\int_{\mathcal{H}} \ln\frac{dQ'}{dP}dQ' + (1-Z'_Q)\int_{\mathcal{H}} \ln\frac{dQ'^c}{dP}dQ'^c - H(Z_{Q'}) \\
&= Z_{Q'}D_{\mathrm{KL}}(Q'\|P) + (1-Z'_Q)D_{\mathrm{KL}}(Q'^c\|P) - H(Z_{Q'}).
\end{aligned}$$

So

$$D_{\mathrm{KL}}(Q'\|P) = \frac{1}{Z_{Q'}}\left(D_{\mathrm{KL}}(Q\|P) + H(Z_{Q'}) - (1-Z'_Q)D_{\mathrm{KL}}(Q'^c\|P)\right) \leq 2(D_{\mathrm{KL}}(Q\|P)+1),$$

since $D_{\mathrm{KL}}(Q'^c\|P)\geq 0$. □

C.1.3 Proof of Lemma 5.4.8

We first present the following Lemma C.1.6 that bounds the difference between the margin loss on V_m and that on V_0 for a fixed GNN.

Lemma C.1.6. *Suppose an L-layer GNN classifier h is associated with model parameters $W_1, \ldots, W_L$. Define $T_h := \max_{l=1,\ldots,L} \|W_l\|_2$. Under Assumption 5.4.4 and 5.4.6, for any $0 < m \leq M$ and $\gamma \geq 0$, if $\epsilon_m T_h^L \leq \frac{\gamma}{4}$, then*

$$\mathcal{L}_m^{\gamma/2}(h) - \mathcal{L}_0^{\gamma}(h) \leq cK\epsilon_m.$$

Proof. For simplicity in this proof, for any $i \in V_0 \cup V_m$ and $k = 1, \ldots, K$, we use h_i to denote $h_i(X, G)$ and use $\eta_k(i)$ to denote $\Pr(y_i = k \mid g_i(X, G))$. And define $\mathcal{L}^{\gamma}(h_i, y_i) := \mathbb{1}h_i[y_i] \leq \gamma + \max_{k \neq y_i} h_i[k]$. Then we can write

$$\begin{aligned}
&\mathcal{L}_m^{\gamma/2}(h) - \mathcal{L}_0^{\gamma}(h) \\
=&\mathbb{E}_{y^m}\left[\frac{1}{N_m}\sum_{j\in V_m}\mathcal{L}^{\gamma/2}(h_j, y_j)\right] - \mathbb{E}_{y^0}\left[\frac{1}{N_0}\sum_{i\in V_0}\mathcal{L}^{\gamma}(h_i, y_i)\right] \\
=&\frac{1}{N_0}\mathbb{E}_{y^0,y^m}\sum_{i\in V_0}\frac{1}{s_m}\left(\sum_{j\in V_m^{(i)}}\mathcal{L}^{\gamma/2}(h_j, y_j)\right) - \mathcal{L}^{\gamma}(h_i, y_i)
\end{aligned}$$

where in the last step we have used Assumption 5.4.6. Therefore,

$$
\begin{aligned}
&\mathcal{L}_m^{\gamma/2}(h) - \mathcal{L}_0^{\gamma}(h) \\
=&\frac{1}{N_0}\sum_{i\in V_0}\frac{1}{s_m}\left(\sum_{j\in V_m^{(i)}}\mathbb{E}_{y_j}\mathcal{L}^{\gamma/2}(h_j,y_j)\right) - \mathbb{E}_{y_i}\mathcal{L}^{\gamma}(h_i,y_i) \\
=&\frac{1}{N_0}\sum_{i\in V_0}\frac{1}{s_m}\left(\sum_{j\in V_m^{(i)}}\sum_{k=1}^{K}\eta_k(j)\mathcal{L}^{\gamma/2}(h_j,k)\right) - \sum_{k=1}^{K}\Pr(y_i=k)\mathcal{L}^{\gamma}(h_i,k) \\
=&\frac{1}{N_0}\sum_{i\in V_0}\frac{1}{s_m}\sum_{j\in V_m^{(i)}}\sum_{k=1}^{K}\left(\eta_k(j)\mathcal{L}^{\gamma/2}(h_j,k) - \eta_k(i)\mathcal{L}^{\gamma}(h_i,k)\right) \\
=&\frac{1}{N_0}\sum_{i\in V_0}\frac{1}{s_m}\sum_{j\in V_m^{(i)}}\sum_{k=1}^{K}\left(\eta_k(j)\left(\mathcal{L}^{\gamma/2}(h_j,k) - \mathcal{L}^{\gamma}(h_i,k)\right) + \left(\eta_k(j)-\eta_k(i)\right)\mathcal{L}^{\gamma}(h_i,k)\right) \qquad \text{(C.7)} \\
\leq&\frac{1}{N_0}\sum_{i\in V_0}\frac{1}{s_m}\sum_{j\in V_m^{(i)}}\sum_{k=1}^{K}\left(1\cdot\left(\mathcal{L}^{\gamma/2}(h_j,k) - \mathcal{L}^{\gamma}(h_i,k)\right) + \left(\eta_k(j)-\eta_k(i)\right)\cdot 1\right), \qquad \text{(C.8)}
\end{aligned}
$$

where the last inequality utilizes the facts that both $\eta_k(j)$ and $\mathcal{L}^{\gamma}(h_i,k)$ are upper-bounded by 1. According to Assumption 5.4.4 and 5.4.6,

$$\eta_k(j) - \eta_k(i) \leq c\|g_j(X,G) - g_i(X,G)\|_2 \leq c\epsilon_m.$$

Further, as $h_i = f(g_i(X,G); W_1, \ldots, W_L)$ where f is a ReLU-activated MLP, so

$$\|h_i - h_j\|_\infty \leq \|g_i(X,G) - g_j(X,G)\|_2 \prod_{l=1}^{L}\|W_l\|_2 \leq \epsilon_m T_h^L \leq \frac{\gamma}{4}.$$

This implies that, for any $k = 1, \ldots, K$,

$$\mathcal{L}^{\gamma/2}(h_j,k) \leq \mathcal{L}^{\gamma}(h_i,k).$$

So we have

$$\begin{aligned}
&\mathcal{L}_m^{\gamma/2}(h) - \mathcal{L}_0^{\gamma}(h) \\
\leq &\frac{1}{N_0} \sum_{i \in V_0} \frac{1}{s_m} \sum_{j \in V_m^{(i)}} \sum_{k=1}^{K} 0 + c\epsilon_m \\
= &cK\epsilon_m.
\end{aligned}$$

□

Then we can prove Lemma 5.4.8, which we re-state as Lemma C.1.7 below.

Lemma C.1.7 (Bound for $D_{m,0}^{\gamma}(P;\lambda)$)**.** *Under Assumption 5.4.4, 5.4.6 and 5.4.7, for any $0 < m \leq M$, any $0 < \lambda \leq N_0^{2\alpha}$ and $\gamma \geq 0$, assume the "prior" P on $\mathcal{H}$ is defined by sampling the vectorized MLP parameters from $\mathcal{N}(0, \sigma^2 I)$ for some $\sigma^2 \leq \frac{(\gamma/8\epsilon_m)^{2/L}}{2b(\lambda N_0^{-\alpha} + \ln 2bL)}$. We have*

$$D_{m,0}^{\gamma/2}(P;\lambda) \leq \ln 3 + \lambda c K \epsilon_m. \tag{C.9}$$

Proof. Recall that $D_{m,0}^{\gamma/2}(P;\lambda) = \ln \mathbb{E}_{h\sim P} e^{\lambda\left(\mathcal{L}_m^{\gamma/4}(h) - \mathcal{L}_0^{\gamma/2}(h)\right)}$. We prove the upper bound of $D_{m,0}^{\gamma/2}(P;\lambda)$ by decomposing the space $\mathcal{H}$ into the two regimes: a regime with bounded spectral norms of the model parameters required by Lemma C.1.6, and its complement. Following Lemma C.1.6, for any classifier h with parameters $W_1, \ldots, W_L$, we define $T_h := \max_{l=1,\ldots,L} \|W_l\|_2$.

We first prove an upper bound on the probability $\Pr\left(T_h^L \epsilon_m > \frac{\gamma}{8}\right)$ over the drawing of $h \sim P$. For any h, as its vectorized MLP parameters $\mathrm{vec}(W_l)$, for each $l = 1, \ldots, L$, is sampled from $\mathcal{N}(0, \sigma^2 I)$, we have the following spectral norm bound [139], for any $t > 0$,

$$\Pr(\|W_l\|_2 > t) \leq 2be^{-\frac{t^2}{2b\sigma^2}},$$

where b is the maximum width of all hidden layers of the MLP. Setting $t = \left(\frac{\gamma}{8\epsilon_m}\right)^{1/L}$

and applying a union bound, we have that

$$\Pr\Big(T_h^L \epsilon_m > \frac{\gamma}{8}\Big) = \Pr\left(T_h > \left(\frac{\gamma}{8\epsilon_m}\right)^{1/L}\right) \leq 2bLe^{-\frac{(\gamma/8\epsilon_m)^{2/L}}{2b\sigma^2}} \leq e^{-\lambda N_0^{-\alpha}},$$

where the last inequality utilizes the condition $\sigma^2 \leq \frac{(\gamma/8\epsilon_m)^{2/L}}{2b(\lambda N_0^{-\alpha} + \ln 2bL)}$.

For any h satisfying $T_h^L \epsilon_m \leq \frac{\gamma}{8}$, by Lemma C.1.6, we know that $e^{\lambda\left(\mathcal{L}_m^{\gamma/4}(h) - \mathcal{L}_0^{\gamma/2}(h)\right)} \leq e^{\lambda c K \epsilon_m}$. For all h such that $T_h^L \epsilon_m > \frac{\gamma}{8}$, by Assumption 5.4.7, with probability at least $1 - e^{-N_0^{2\alpha}}$,

$$e^{\lambda\left(\mathcal{L}_m^{\gamma/4}(h) - \mathcal{L}_0^{\gamma/2}(h)\right)} \leq e^{\lambda N_0^{-\alpha} + \lambda c K \epsilon_m}.$$

Also note that $\mathcal{L}_m^{\gamma/4}(h) - \mathcal{L}_0^{\gamma/2}(h) \leq 1$ trivially holds for any h. Therefore we have

$$\begin{aligned}
&D_{m,0}^{\gamma/2}(P;\lambda) \\
&= \ln \mathbb{E}_{h\sim P} e^{\lambda\left(\mathcal{L}_m^{\gamma/4}(h) - \mathcal{L}_0^{\gamma/2}(h)\right)} \\
&\leq \ln\left(\Pr\Big(T_h^L \epsilon_m > \frac{\gamma}{8}\Big)\left(e^{-N_0^{2\alpha}} \cdot e^{\lambda} + (1 - e^{-N_0^{2\alpha}}) \cdot e^{\lambda N_0^{-\alpha} + \lambda c K \epsilon_m}\right) + \Pr\Big(T_h^L \epsilon_m \leq \frac{\gamma}{8}\Big) e^{\lambda c K \epsilon_m}\right) \\
&\leq \ln\left(e^{\lambda - N_0^{2\alpha}} + \Pr\Big(T_h^L \epsilon_m > \frac{\gamma}{8}\Big) e^{\lambda N_0^{-\alpha}} e^{\lambda c K \epsilon_m} + e^{\lambda c K \epsilon_m}\right) \\
&\leq \ln\left(1 + e^{-\lambda N_0^{-\alpha}} e^{\lambda N_0^{-\alpha}} e^{\lambda c K \epsilon_m} + e^{\lambda c K \epsilon_m}\right) \\
&= \ln\left(1 + 2e^{\lambda c K \epsilon_m}\right) \\
&\leq \ln 3 + \lambda c K \epsilon_m,
\end{aligned}$$

since $1 + 2e^{\lambda c K \epsilon_m} \leq 3e^{\lambda c K \epsilon_m}$.

□

C.1.4 Proof of Theorem 5.4.10

The proof of Theorem 5.4.10 relies on the combination of Theorem 5.4.3, Lemma 5.4.8, and an intermediate result of the Theorem 1 in Neyshabur et al. [109] (which we state as Lemma C.1.8 below).

Lemma C.1.8 (Neyshabur et al. [109]). *Let $\tilde{h}$ be any classifier in $\mathcal{H}$ with parameters $\{\widetilde{W}_l\}_{l=1}^L$. Define $\tilde{\beta} = \left(\prod_{l=1}^L \|\widetilde{W}_l\|_2\right)^{1/L}$. Let $\{U_l\}_{l=1}^L$ be the random perturbation to be added to $\{\widetilde{W}_l\}_{l=1}^L$ and $vec(\{U_l\}_{l=1}^L) \sim \mathcal{N}(0, \sigma^2 I)$. Define $B_m := \max_{i \in V_0 \cup V_m} \|g_i(X, G)\|_2$. If*

$$\sigma \leq \frac{\gamma}{84 L B_m \beta^{L-1} \sqrt{b \ln(4bL)}},$$

and β is any constant satisfying $|\tilde{\beta} - \beta| \leq \frac{\tilde{\beta}}{L}$, then with respect to the random draw of $\{U_l\}_{l=1}^L$,

$$\Pr\left(\max_{i \in V_0 \cup V_m} \|f(g_i(X, G); \{\widetilde{W}_l\}_{l=1}^L) - f(g_i(X, G); \{\widetilde{W}_l + U_l\}_{l=1}^L)\|_\infty < \frac{\gamma}{8}\right) > \frac{1}{2}.$$

Then we prove Theorem 5.4.10 (re-stated as Theorem C.1.9 below).

Theorem C.1.9 (Subgroup Generalization Bound for GNNs). *Let $\tilde{h}$ be any classifier in $\mathcal{H}$ with parameters $\{\widetilde{W}_l\}_{l=1}^L$. Under Assumptions 5.4.4, 5.4.6, 5.4.7, and 5.4.9, for any $0 < m \leq M$, $\gamma \geq 0$, and large enough N_0, with probability at least $1 - \delta$ over the sample of y^0, we have*

$$\mathcal{L}_m^0(\tilde{h}) \leq \widehat{\mathcal{L}}_0^\gamma(\tilde{h}) + O\left(cK\epsilon_m + \frac{b\sum_{l=1}^L \|\widetilde{W}_l\|_F^2}{(\gamma/8)^{2/L} N_0^\alpha} (\epsilon_m)^{2/L} + \frac{1}{N_0^{1-2\alpha}} + \frac{1}{N_0^{2\alpha}} \ln \frac{LC(2B_m)^{1/L}}{\gamma^{1/L}\delta}\right). \tag{C.10}$$

Proof. There are two main steps in the proof. In the first step, for a given constant $\beta > 0$, we first define the "prior" P and the "posterior" Q on $\mathcal{H}$ in a way complying the conditions in Lemma 5.4.8 and Lemma C.1.8. Then for all classifiers with parameters satisfying $|\tilde{\beta} - \beta| \leq \frac{\tilde{\beta}}{L}$, where $\tilde{\beta} = \left(\prod_{l=1}^L \|\widetilde{W}_l\|_2\right)^{1/L}$, we can derive a generalization bound by applying Theorem 5.4.3 and Lemma 5.4.8. In the second step, we investigate the number of β we need to cover all possible relevant classifier parameters and apply a union bound to get the final bound. The second step is essentially the same as Neyshabur et al. [109] while the first step differs by the need of incorporating Lemma 5.4.8.

We first show the first step. Given a choice of β independent of the training data, let

$$\sigma = \min\left(\frac{(\gamma/8\epsilon_m)^{1/L}}{\sqrt{2b(\lambda N_0^{-\alpha} + \ln 2bL)}}, \frac{\gamma}{84LB_m\beta^{L-1}\sqrt{b\ln(4bL)}}\right).$$

Assume the "prior" P on $\mathcal{H}$ is defined by sampling the vectorized MLP parameters from $\mathcal{N}(0, \sigma^2 I)$; and the "posterior" Q on $\mathcal{H}$ is defined by first sampling a set of random perturbations $\{U_l\}_{l=1}^L$ with $\mathrm{vec}(\{U_l\}_{l=1}^L) \sim \mathcal{N}(0, \sigma^2 I)$ and then adding them to $\{\widetilde{W}_l\}_{l=1}^L$, the parameters of $\tilde{h}$. Then for any $\tilde{h}$ with $\{\widetilde{W}_l\}_{l=1}^L$ satisfying $|\tilde{\beta} - \beta| \leq \frac{\tilde{\beta}}{L}$, by Lemma C.1.8, we have

$$\Pr_{h\sim Q}\left(\max_{i\in V_0\cup V_m} |h_i(X,G) - \tilde{h}_i(X,G)|_\infty < \frac{\gamma}{8}\right) > \frac{1}{2}.$$

Therefore, by applying Theorem 5.4.3, we know the bound (5.4) holds for $\tilde{h}$, i.e., with probability at least $1-\delta$,

$$\begin{aligned}
&\mathcal{L}_m^0(\tilde{h}) - \widehat{\mathcal{L}}_0^\gamma(\tilde{h}) \\
\leq &\frac{1}{\lambda}\left(2(D_{\mathrm{KL}}(Q\|P)+1) + \ln\frac{1}{\delta} + \frac{\lambda^2}{4N_0} + D_{m,0}^{\gamma/2}(P;\lambda)\right) \\
\leq &\frac{1}{\lambda}\left(2(D_{\mathrm{KL}}(Q\|P)+1) + \ln\frac{1}{\delta} + \frac{\lambda^2}{4N_0} + \ln 3 + \lambda cK\epsilon_m\right) \qquad \text{(C.11)} \\
\leq &\frac{2}{N_0^{2\alpha}} D_{\mathrm{KL}}(Q\|P) + \frac{1}{N_0^{2\alpha}}\left(\ln\frac{3}{\delta} + 2\right) + \frac{1}{4N_0^{1-2\alpha}} + cK\epsilon_m, \qquad \text{(C.12)}
\end{aligned}$$

where in (C.11) we have applied Lemma 5.4.8 to bound $D_{m,0}^{\gamma/2}(P;\lambda)$ under Assumptions 5.4.4, 5.4.6, and 5.4.7; and in (C.12) we have set $\lambda = N_0^{2\alpha}$.

Moreover, since both P and Q are normal distributions, we know that

$$D_{\mathrm{KL}}(Q\|P) \leq \frac{\sum_{l=1}^L \|\widetilde{W}_l\|_F^2}{2\sigma^2}.$$

By Assumption 5.4.9, both B_m and C are constant with respect to N_0. Later we

will show that we only need $\beta \leq C$. Therefore, for large enough N_0, we can have

$$\frac{(\gamma/8\epsilon_m)^{1/L}}{\sqrt{2b(N_0^\alpha + \ln 2bL)}} < \frac{\gamma}{84LB_m\beta^{L-1}\sqrt{b\ln(4bL)}},$$

which implies,

$$\sigma = \frac{(\gamma/8\epsilon_m)^{1/L}}{\sqrt{2b(N_0^\alpha + \ln 2bL)}},$$

and hence,

$$D_{\mathrm{KL}}(Q\|P) \leq \frac{b(N_0^\alpha + \ln 2bL)\sum_{l=1}^L \|\widetilde{W}_l\|_F^2}{(\gamma/8)^{2/L}}(\epsilon_m)^{2/L}. \tag{C.13}$$

Therefore, with probability at least $1-\delta$,

$$\begin{aligned}
&\mathcal{L}_m^0(\tilde{h}) - \widehat{\mathcal{L}}_0^\gamma(\tilde{h}) \\
\leq& cK\epsilon_m + \frac{2}{N_0^{2\alpha}}D_{\mathrm{KL}}(Q\|P) + \frac{1}{N_0^{2\alpha}}\left(\ln\frac{3}{\delta} + 2\right) + \frac{1}{4N_0^{1-2\alpha}} \\
\leq& O\left(cK\epsilon_m + \frac{b\sum_{l=1}^L \|\widetilde{W}_l\|_F^2}{(\gamma/8)^{2/L}N_0^\alpha}(\epsilon_m)^{2/L} + \frac{1}{N_0^{1-2\alpha}} + \frac{1}{N_0^{2\alpha}}\ln\frac{1}{\delta}\right).
\end{aligned} \tag{C.14}$$

Then we show the second step, i.e., finding out the number of β we need to cover all possible relevant classifier parameters. Similarly as Neyshabur et al. [109], we will show that we only need to consider $(\frac{\gamma}{2B_m})^{1/L} \leq \tilde{\beta} \leq C$ (recall that $\|\widetilde{W}_l\|_F \leq C, l = 1, \ldots, L$). For any $\tilde{\beta}$ outside this range, the bound (C.10) automatically holds. If $\tilde{\beta} < (\frac{\gamma}{2B_m})^{1/L}$, then for any node $i \in V_0$, $\|\tilde{h}_i(X,G)\|_\infty < \frac{\gamma}{2}$, which implies $\widehat{\mathcal{L}}_0^\gamma(\tilde{h}) = 1$ as the difference between any two output logits for any training node is smaller than γ. Also noticing that $\mathcal{L}_m^0(\tilde{h}) \leq 1$ by definition, so the bound (C.10) trivially holds. And for $\tilde{\beta}$ in this range, $|\beta - \tilde{\beta}| \leq \frac{1}{L}(\frac{\gamma}{2B_m})^{1/L}$ is a sufficient condition for β to satisfy $|\tilde{\beta} - \beta| \leq \frac{\tilde{\beta}}{L}$, and we need at most $\frac{LC(2B_m)^{1/L}}{\gamma^{1/L}}$ of β to cover all $\tilde{\beta}$ in the above range. Taking a union bound on all such β, which is equivalent to replace δ with $\frac{\delta}{\frac{LC(2B_m)^{1/L}}{\gamma^{1/L}}}$

in (C.14), it gives us the final result: with probability at least $1-\delta$,

$$\mathcal{L}_m^0(\tilde{h}) - \widehat{\mathcal{L}}_0^\gamma(\tilde{h}) \le O\left(cK\epsilon_m + \frac{b\sum_{l=1}^L \|\widetilde{W}_l\|_F^2}{(\gamma/8)^{2/L} N_0^\alpha} (\epsilon_m)^{2/L} + \frac{1}{N_0^{1-2\alpha}} + \frac{1}{N_0^{2\alpha}} \ln \frac{LC(2B_m)^{1/L}}{\gamma^{1/L}\delta} \right).$$

□

C.1.5 Discussion on Assumption 5.4.7

To better understand Assumption 5.4.7, we show a simplified scenario where this assumption holds.

We discuss in the context where the classification problem has binary labels and the MLP of the classifier h only consists of a linear layer with parameters $W \in \mathbb{R}^{D'\times 2}$. In this case, the distribution P on $\mathcal{H}$ in Assumption 5.4.7 is defined by sampling the vectorized parameters $\mathrm{vec}(W) \sim \mathcal{N}(0, \sigma^2 I_{2D'})$. Under Assumption 5.4.6, each training sample in V_0 has a near set in V_m with the same size s_m. For simplicity, we consider the case where $s_m = 1$. Let $Z^{(0)}, Z^{(m)} \in \mathbb{R}^{N_0\times D'}$ be the aggregated node features of V_0 and V_m respectively. Without loss of generality, assume for each $i = 1, \ldots, N_0$, the closest point in $Z^{(0)}$ for $Z_i^{(m)}$ is $Z_i^{(0)}$. To simplify the notations, we define $Z := Z^{(0)}$ and $\varepsilon := Z^{(m)} - Z^{(0)}$. We always treat M_i for any matrix M as the transpose of the i-th row of M and define $M_{(i)}$ as the i-th column vector of M.

Following the proof of Lemma C.1.6, and in particular, Eq. (C.7), it is easy to show that, for any $h \in \mathcal{H}$ with parameters W,

$$\begin{aligned}
&\mathcal{L}_m^{\gamma/4}(h) - \mathcal{L}_0^{\gamma/2}(h) \\
\le& \frac{1}{N_0}\sum_{i=1}^{N_0}\sum_{k=1}^{2} \eta_k(Z_i^{(m)}) \left(\mathcal{L}^{\gamma/4}((Z^{(m)}\cdot W)_i, k) - \mathcal{L}^{\gamma/2}((Z^{(0)}\cdot W)_i, k) \right) + c\epsilon_m \\
=& 2c\epsilon_m + \frac{1}{N_0}\sum_{i=1}^{N_0}\sum_{k=1}^{2} \eta_k(Z_i^{(m)}) \left(\mathbb{1} W_{(k)}^T (Z_i + \varepsilon_i) < \frac{\gamma}{4} + W_{(3-k)}^T (Z_i + \varepsilon_i) - \mathbb{1} W_{(k)}^T Z_i < \frac{\gamma}{2} + W_{(3-k)}^T Z_i \right).
\end{aligned} \tag{C.15}$$

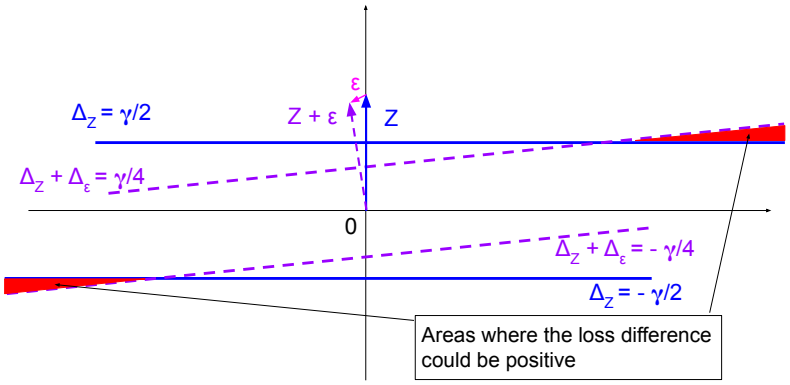


Figure C.1: An illustrative example of areas in the space of Δ_W where the loss difference term for an index i could be positive. For visual simplicity in the figure, we have used Z and ε to represent Z_i and ε_i.

For Assumption 5.4.7 to hold, a sufficient condition is to have the second term in Eq. (C.15) smaller than N^α for any h. Below we will investigate when this sufficient condition holds.

To further simplify the notations, we define $\Delta_W := W_{(1)} - W_{(2)}$, $\Delta_Z := Z\Delta_W$, $\Delta_\varepsilon := \varepsilon\Delta_W$, and $\eta_k^i := \eta_k(Z_i^{(m)})$. Then

$$\begin{aligned}
&\mathcal{L}_m^{\gamma/4}(h) - \mathcal{L}_0^{\gamma/2}(h) \\
\leq & 2c\epsilon_m + \frac{1}{N_0}\sum_{i=1}^{N_0}\sum_{k=1}^{2}\eta_k^i\left(\mathbb{1}(-1)^{k+1}(\Delta_Z+\Delta_\varepsilon)_i < \frac{\gamma}{4} - \mathbb{1}(-1)^{k+1}\Delta_{Zi} < \frac{\gamma}{2}\right). \qquad \text{(C.16)}
\end{aligned}$$

Note that since $\mathrm{vec}(W) \sim \mathcal{N}(0, \sigma^2 I_{2D'})$, we have $\Delta_W \sim \mathcal{N}(0, 2\sigma^2 I_{D'})$. And the second term in (C.16) depends on W only through Δ_W.

Table C.1: Possible values of the loss difference term for each index $i = 1, \ldots, N_0$.

	$\Delta_{Zi} > \frac{\gamma}{2}$	$\Delta_{Zi} < -\frac{\gamma}{2}$	$-\frac{\gamma}{2} \leq \Delta_{Zi} \leq \frac{\gamma}{2}$
$(\Delta_Z + \Delta_\varepsilon)_i > \frac{\gamma}{4}$	0	$\eta_2^i - \eta_1^i$	$-\eta_1^i$
$(\Delta_Z + \Delta_\varepsilon)_i < -\frac{\gamma}{4}$	$\eta_1^i - \eta_2^i$	0	$-\eta_2^i$
$-\frac{\gamma}{4} \leq (\Delta_Z + \Delta_\varepsilon)_i \leq \frac{\gamma}{4}$	η_1^i	η_2^i	0

For each $i = 1, \ldots, N_0$, the term $\sum_{k=1}^{2}\eta_k^i\left(\mathbb{1}(-1)^{k+1}(\Delta_Z+\Delta_\varepsilon)_i < \frac{\gamma}{4} - \mathbb{1}(-1)^{k+1}\Delta_{Zi} < \frac{\gamma}{2}\right)$ in (C.16) has only a few possible values, which can be summarized in the following 9

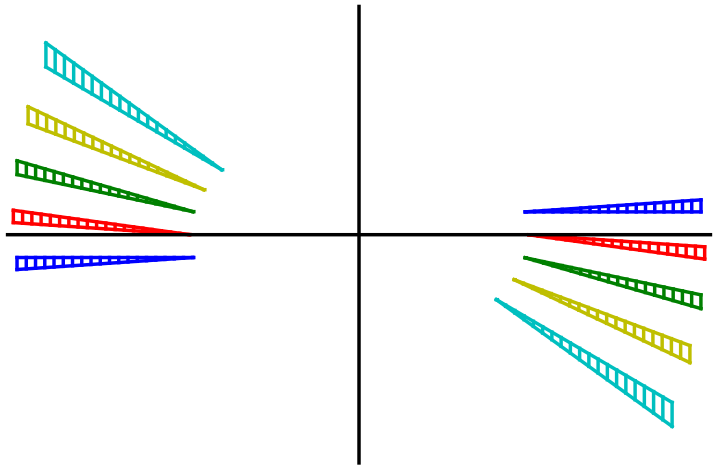
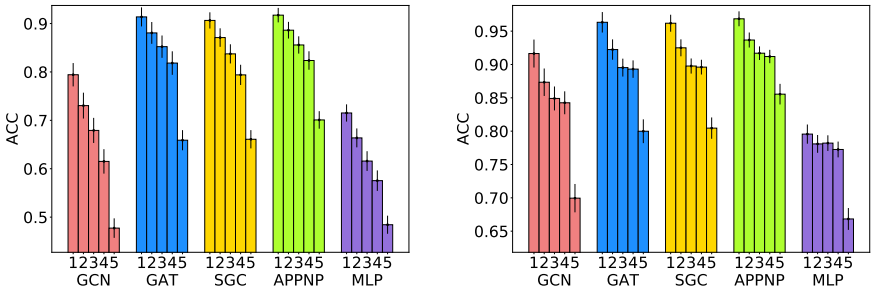


Figure C.2: An illustrative example of data points with no intersected positive areas. Only the positive areas as shown in Figure C.1 are visualized. Each color corresponds to a unique index i. As shown in the figure, there are no intersections among the areas of different data points when the Z_i's are nicely scattered and ε_i's are small.

cases in Table C.1. As can be seen, this loss difference term could be positive only when (1) $\Delta_{Zi} > \frac{\gamma}{2}$ and $(\Delta_Z + \Delta_\varepsilon)_i \leq \frac{\gamma}{4}$ or (2) $\Delta_{Zi} < -\frac{\gamma}{2}$ and $(\Delta_Z + \Delta_\varepsilon)_i \geq -\frac{\gamma}{4}$. This implies that, for fixed Z_i and ε_i, there are two linear subspaces in the space of Δ_W where the loss difference for index i could be positive. In Figure C.1, we provide an illustrative example of such linear subspaces in the case $\Delta_W \in \mathbb{R}^2$, such that we can visualize it. Qualitatively, when $\|\varepsilon_i\|_2$ is much smaller than $\|Z_i\|_2$ (which is often the case by their constructions), the areas that the loss difference term being positive will be very small.

For a classifier h to have $\mathcal{L}_m^{\gamma/4}(h) - \mathcal{L}_0^{\gamma/2}(h) > cK\epsilon_m + N_0^{-\alpha}$, a necessary condition is that its corresponding Δ_W lies in the intersection of positive areas of at least $N_0^{1-\alpha}$ samples. Conversely, if ε_i's are small and Z_i's are nicely scattered such that the N_0 samples can be divided into N_0^α groups where the positive areas of any two points from different groups do not intersect, then we know $\mathcal{L}_m^{\gamma/4}(h) - \mathcal{L}_0^{\gamma/2}(h) \leq cK\epsilon_m + N_0^{-\alpha}$ for any h. And hence this is a sufficient condition for Assumption 5.4.7 to hold. Figure C.2 provides an illustrative example of data points with no intersected positive



(a) Amazon Computers. (b) Amazon Photo.

Figure C.3: Test accuracy disparity across subgroups by aggregated-feature distance. Extra experiments on Amazon-Computers and Amazon-Photo datasets. The experiment and plot settings are the same as Figure V.1.

areas on a 2-dimensional surface. When $D' > 2$, it might be difficult to completely avoid intersections of the positive areas. However, what Assumption 5.4.7 requires is that the areas where a large number of data points intersect are small.

C.2 More Details of Experiment Setup

In this section, we describe more details of our experiment setup that are omitted in the main paper due to the space limit.

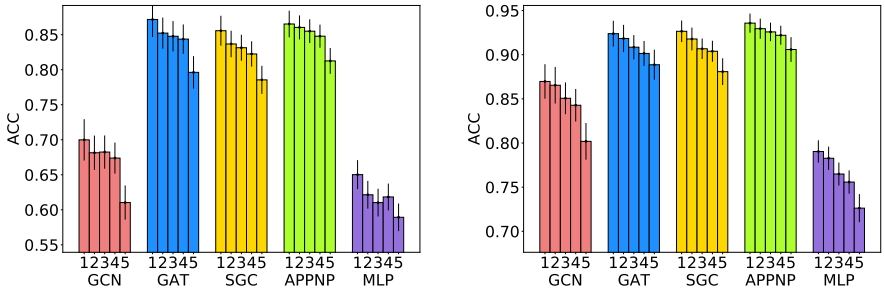
C.2.1 Detailed Training Setup

We use the default setting in Deep Graph Library [145][2] for model hyper-parameters. We use the Adam optimizer with an initial learning rate of 0.01 and weight decay of 5e-4 to train all models for 400 epochs by minimizing the cross-entropy loss, with early stopping on the validation set.

C.2.2 Detailed Setup of the Noisy Feature Experiment

In this experiment (corresponding to Figure V.4), we make the node features less homophilous by adding random noises to each node independently. Specifically, we

[2]Apache License 2.0.



(a) Amazon Computers. (b) Amazon Photo.

Figure C.4: Test accuracy disparity across subgroups by geodesic distance. Extra experiments on Amazon-Computers and Amazon-Photo datasets. The experiment and plot settings are the same as Figure V.2.

use noisy features $\tilde{X} = X + \alpha \frac{\|X\|_F}{\|U\|_F} U$, where $X \in \mathbb{R}^{N \times D}$ is the original feature matrix, and $U \in \mathbb{R}^{N \times D}$ is a random matrix with each element independently and uniformly sampled from $[0, 1]$. And we set $\alpha = 5$. In this way, the magnitude of the noise is slightly larger than the original feature to significantly reduce the homophily property. All other experiment settings are the same as those corresponding to Figure V.1.

C.2.3 Detailed Setup of the Biased Training Node Selection Experiment

In this experiment (corresponding to Section 5.5.2), we investigate the impact of biased training node selection. As briefly described in Section 5.5.2, we choose a "dominant class" and construct a manipulated training set. For each class, we still sample 20 training nodes but in a biased way. Specifically, given one choice of the four node centrality metrics (degree, closeness, betweenness, and PageRank), the training set is sampled as follows.

1. For the dominant class, uniformly sample 15 nodes from the 10% of the nodes with highest node centrality, and uniformly sample 5 nodes from the remaining.

2. For each of the other classes, uniformly sample 15 nodes from the 10% of the nodes with lowest node centrality, and uniformly sample 5 nodes from the remaining.

In this way, the training nodes of the dominant class are biased towards high-centrality nodes while the training nodes of the other classes are biased towards low-centrality nodes.

After the biased training set is constructed, we randomly sample 500 validation nodes and 1000 test nodes from the remaining nodes and perform the model training following the standard setup as the previous experiments.

C.3 Extra Experiment Results

C.3.1 Accuracy Disparity on Amazon Datasets

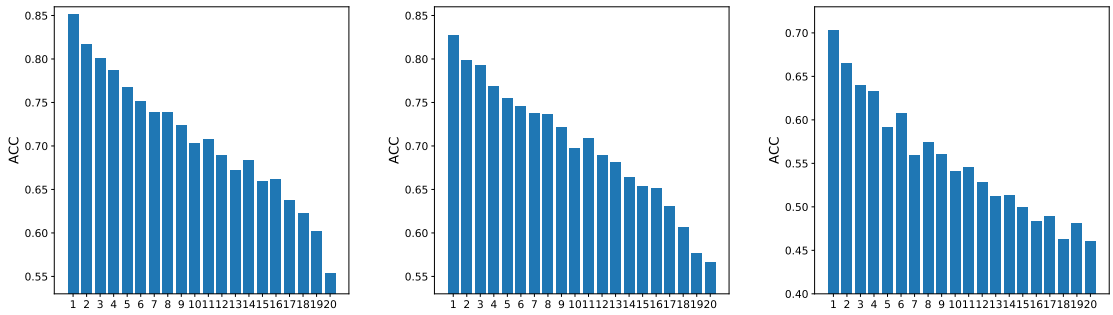
In addition to the commonly used citation network benchmarks, Cora, Citeseer, and Pubmed [126, 155], we also provide results of the test accuracy disparity experiments of subgroups by aggregated-feature distance and geodesic distance on Amazon-Computers and Amazon-Photo datasets [130], which have a similar scale but a different network type compared to the three citation networks.

For Amazon-Computers and Amazon-Photo, we follow exactly the same experiment procedure as for Cora, Citeseer, and Pubmed. The results of subgroups by aggregated-feature distance are shown in Figure C.3 and the results of subgroups by geodesic distance are shown in Figure C.4. The results are respectively similar as those in Figure V.1 and Figure V.2.

C.3.2 Accuracy Disparity on Open Graph Benchmarks

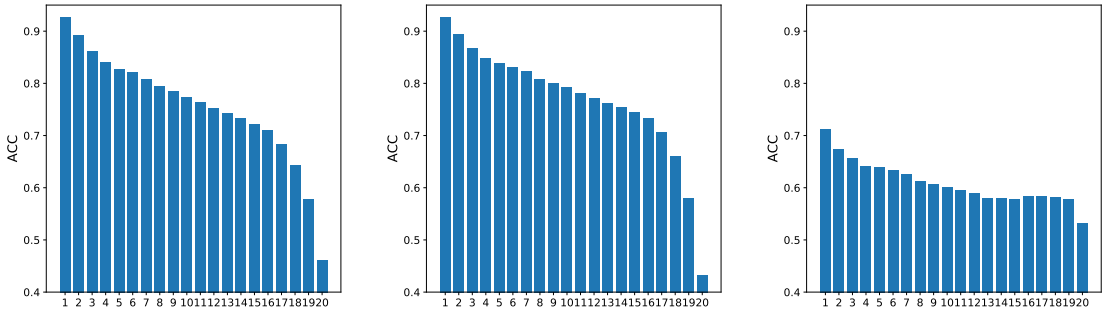
We further provide experiment results on two large-scale datasets from Open Graph Benchmark [63], OGBN-Arxiv and OGBN-Products.

For OGBN-Arxiv and OGBN-Products, we first follow the standard training procedure suggested by Open Graph Benchmark [63] to train a GCN, a GraphSAGE, and an MLP. And we split the test groups into 20 groups in terms of the aggregated



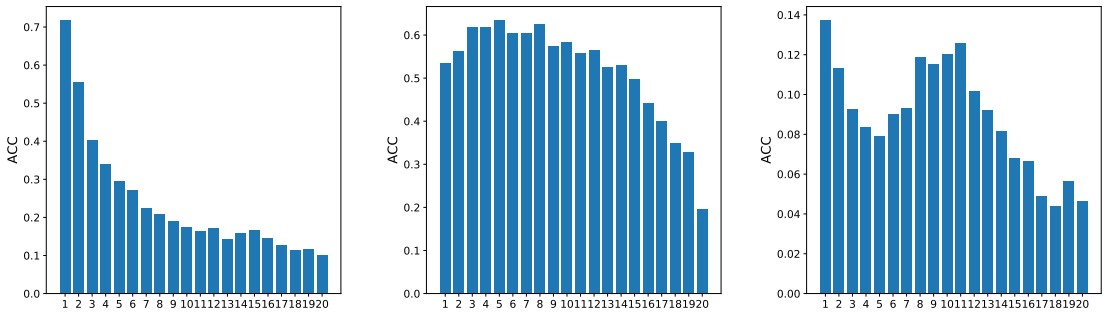
(a) GCN. (b) GraphSAGE. (c) MLP.

Figure C.5: Results on OGBN-Arxiv. Test accuracy disparity across subgroups by aggregated-feature distance. Each figure corresponds to a model. Bars labeled 1 to 20 represent subgroups with increasing distance to training set.



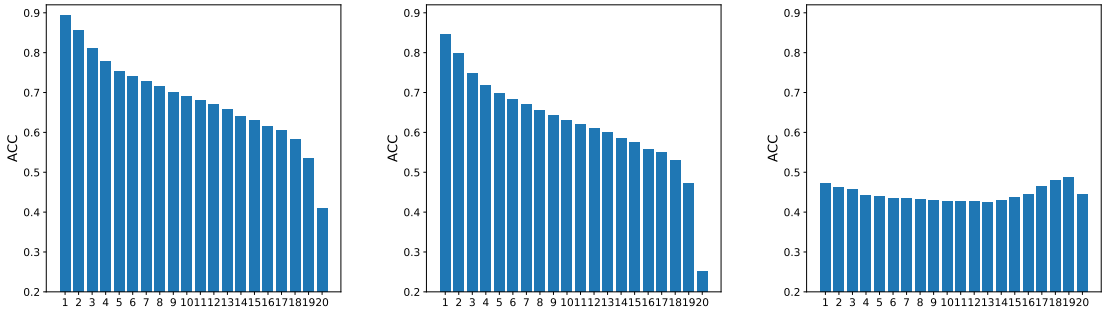
(a) GCN. (b) GraphSAGE. (c) MLP.

Figure C.6: Results on OGBN-Products. Test accuracy disparity across subgroups by aggregated-feature distance. The experiment and plot settings are the same as Figure C.5.



(a) GCN. (b) GraphSAGE. (c) MLP.

Figure C.7: Results on OGBN-Arxiv. Test accuracy disparity across subgroups by aggregated-feature distance, experimented with noisy features. The experiment and plot settings are the same as Figure C.5, except the node features are perturbed by independent noises to reduce homophily.



(a) GCN. (b) GraphSAGE. (c) MLP.

Figure C.8: Results on OGBN-Products. Test accuracy disparity across subgroups by aggregated-feature distance, experimented with noisy features. The experiment and plot settings are the same as Figure C.6, except for the node features are perturbed by independent noises to reduce homophily.

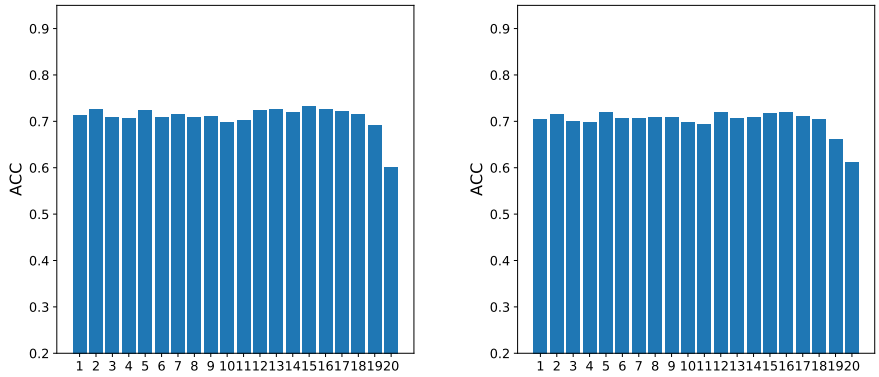
feature distance. As there are more test nodes available, we can afford the split of more groups better resolution. The results on OGBN-Arxiv and OGBN-Products are respectively shown in Figure C.5 and Figure C.6, where we observe a similar decreasing pattern of test accuracy as in Figure V.1 (on the citation networks). Since there is also a decreasing pattern for MLP, following the experiments shown in Figure V.4, we further inject independent noises to node features to reduce the homophily of the OGBN-Arxiv and OGBN-Products dataset and repeat the experiments in Figure C.5 and Figure C.6. The results are respectively shown in Figure C.7 and Figure C.8, where, similar as Figure V.4, the decreasing pattern largely remains for GNNs but disappears for the MLP.

We also experiment on subgroups split in terms of geodesic distance and node centrality metrics. The results of these experiments are slightly different on the large-scale datasets compared to those on the smaller benchmark datasets.

For geodesic distance (Figure C.9 for OGBN-Arxiv and Figure C.10 for OGBN-Products), there is not a descending trend of test accuracy until the last few groups. This is because the size of training set is large such that most test nodes are 1-hop neighbors of some training nodes. Therefore most groups are random split of such 1-hop neighbors and there will not be a descending accuracy among these subgroups. This problem is especially obvious for OGBN-Arxiv, where 60% of the nodes are in the training set. So we only see an accuracy drop on the last two subgroups. The size of training set is relatively smaller on OGBN-Products but still more than 60% of the test nodes are 1-hop neighbors of some training nodes. It is worth-noting that the plots in Figure C.10 have stair patterns, showing a clear descending trend with respect to the geodesic distance. The fluctuation of early subgroups is larger on OGBN-Arxiv because there are fewer test nodes in OGBN-Arxiv than in OGBN-Products. Overall, there is still a clear descending trend with respect to increasing geodesic distance. But the nodes are less distinguishable in terms of geodesic distance

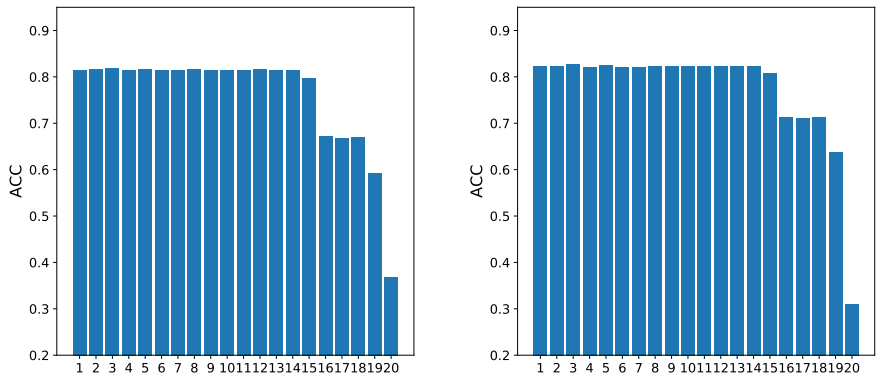
than aggregated-feature distance, especially when the size of training set is large (more discussions in Appendix C.4.1).

For node centrality metrics, we report experiments on degree and PageRank, and omit the betweenness and closeness metrics due to their high computation cost on large-scale graphs. The results on OGBN-Arxiv are shown in Figure C.11. It is intriguing that there is a descending trend with respect to the degree and PageRank metrics on this particular experiment setting, though the descending trend is not as sharp as the one in Figure C.5 (experiments on aggregated-feature distance). It is possible that, when there is a very large training set (60% in this case), the node centrality metrics become related to the aggregated-feature distance. However, node centrality metrics again fail to capture the descending trend on OGBN-Products, as shown in Figure C.12. In future work, we plan to further explore the relationship between the theoretically derived aggregated-feature distance and various more intuitive graph metrics on different graph data.



(a) GCN. (b) GraphSAGE.

Figure C.9: Results on OGBN-Arxiv. Test accuracy disparity across subgroups by geodesic distance. The experiment and plot settings are the same as Figure C.5, except for the aggregated-feature distance is replaced by the geodesic distance.



(a) GCN. (b) GraphSAGE.

Figure C.10: Results on OGBN-Products. Test accuracy disparity across subgroups by geodesic distance. The experiment and plot settings are the same as Figure C.9.

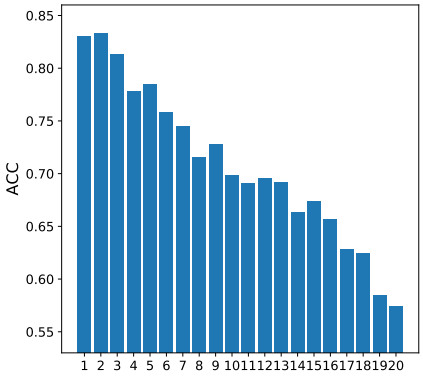
C.3.3 More Results of Biased Training Node Selection

In Figure V.5 of Section 5.5.2, we have shown that the learned GNN models will be biased towards the labels of training nodes of higher centrality (while the learned MLP models do not show a similar trend). Due to the space limit, in the main paper, we are only able to report the experiment results on Cora with a particular class selected as the "dominant" class. Here we report the full experiment results on three datasets, with each class selected as the "dominant" class. The results on Cora, Citeseer, and Pubmed are respectively shown in Figures C.14, C.15, and C.16. As can be seen from the figures, the observed phenomenon is consistent over almost all settings.

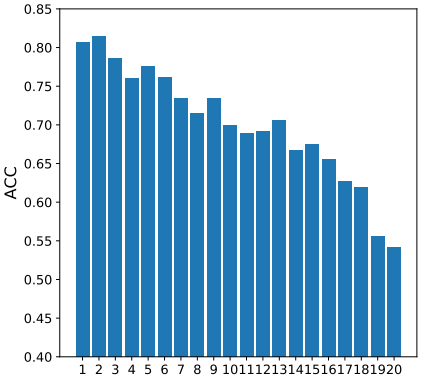
C.4 Discussions

C.4.1 Relationship Between Aggregated-Feature Distance and Geodesic Distance

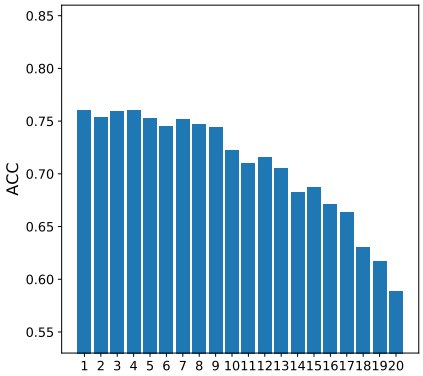
We discuss two scenarios where the aggregated-feature distance and the geodesic distance are likely to be related.



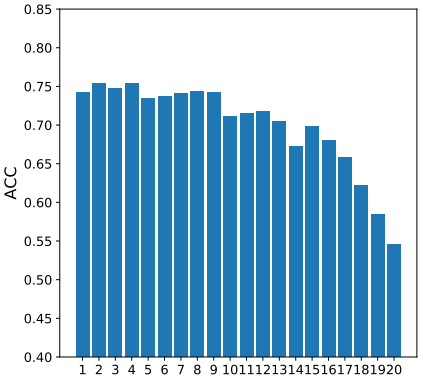
(a) GCN. Subgroup by degree.



(b) GraphSAGE. Subgroup by de-

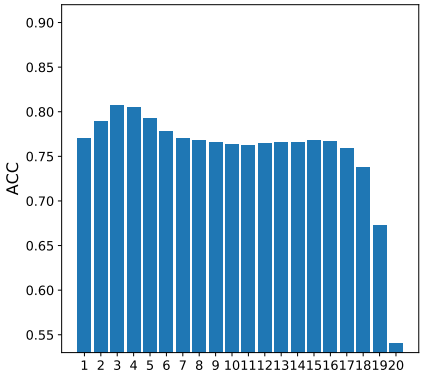


(c) GCN. Subgroup by PageRank.

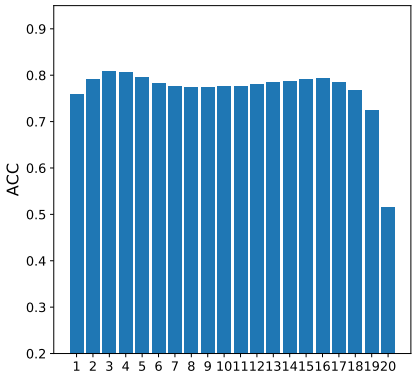


(d) GraphSAGE. Subgroup by PageRank.

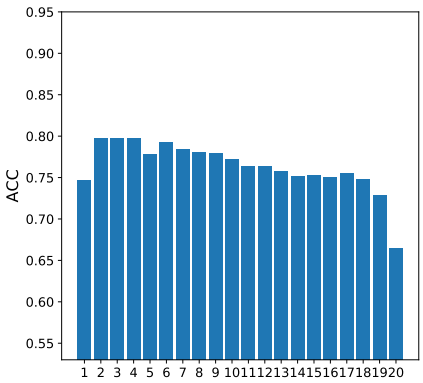
Figure C.11: Results on OGBN-Arxiv. Test accuracy disparity across subgroups by node centrality metrics. The experiment and plot settings are the same as Figure C.5.



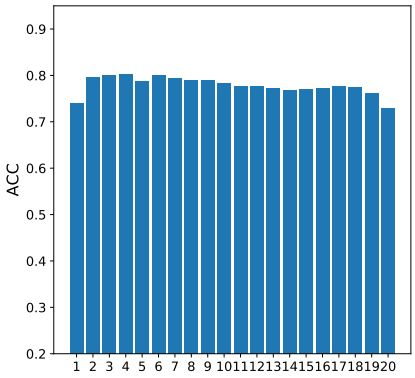
(a) GCN. Subgroup by degree.



(b) GraphSAGE. Subgroup by de-



(c) GCN. Subgroup by PageRank.



(d) GraphSAGE. Subgroup by PageRank.

Figure C.12: Results on OGBN-Products. Test accuracy disparity across subgroups by node centrality metrics. The experiment and plot settings are the same as Figure C.6.

Smoothing effect of feature aggregation in GNNs. Many existing GNN models are known to have a smoothing effect on the aggregated node features [86]. As a result, nodes with a shorter geodesic distance are likely to have more similar aggregated features.

Homophily. Many real-world graph-structured data exhibit a homophily property [104], i.e., connected nodes tend to share similar attributes. In this case, again, nodes with a shorter geodesic distance on the graph tend to have more similar aggregated features.

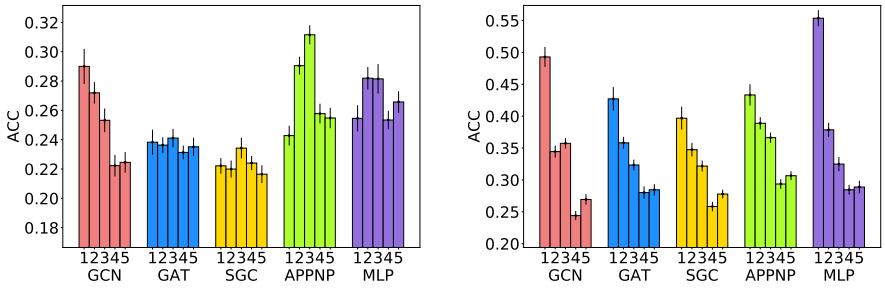
However, the geodesic distance is usually coarser-grained than the aggregated-feature distance due to its discrete nature. When the graph is a "small world" [147] and the number of training nodes is large, the geodesic distance from most test nodes to the set of training nodes will concentrate on 1 or 2 hops, making the test nodes indistinguishable with respect to this metric.

It is an interesting future direction to explore interpretable graph metrics that may better relate to the aggregated-feature distance.

C.4.2 Implications for GNN Generalization under Non-Homophily

A number of recent studies suggest that classical GNNs (e.g., GCN [77]) can only work well when the labels of connected nodes are similar [110, 61, 165], which is now commonly referred as homophily [104]. However, homophily is not a necessary condition to have small generalization errors in our analysis. Instead, good generalization can be achieved when the aggregated features of test nodes are close to those of some training nodes. Interestingly, a concurrent work [101] of this paper observes a similar phenomenon with empirical evidence.

The new results by Ma et al. [101] and our work suggest that the space of non-homophilous data can be further dissected into more fine-grained categories, which may motivate designs of new GNN models tailored for each category.



(a) Squirrel. (b) Chameleon.

Figure C.13: Test accuracy disparity across subgroups by aggregated-feature distance. Extra experiments on *non-homophilious* datasets, Squirrel and Chameleon. The experiment and plot settings are the same as Figure V.1.

Experiments on "Non-Homophilious" Datasets. We further have experiments on two datasets, Squirrel and Chameleon [122], that are usually considered as non-homophilious [90]. Interestingly, while the analysis in this work is not guaranteed to be applicable on non-homophilious datasets, we find that the decreasing-accuracy pattern appears on Chameleon while it does not appear on Squirrel (see Figure C.13). With further investigation, we find that while Chameleon is similarly "non-homophilious" as Squirrel in terms of the homophily measure proposed by Zhu et al. [165], Chameleon is much more "homophilious" than Squirrel in terms of the homophily measure proposed by Lim et al. [90].

C.4.3 Limitations of the Analysis

To our best knowledge, this work is one of the first attempts[3] to theoretically analyze the generalization ability of GNNs under non-IID node-level semi-supervised learning tasks. While we believe this work presents non-trivial contributions towards the theoretical understanding of generalization and fairness of GNNs with supportive empirical evidence, there are a few limitations of the current analysis which we hope to improve in future work.

[3]The only other work we are aware of is by Baranwal et al. [6], where strong assumptions (CSBM) on the data generating mechanisms are made.

The first limitation is that the derived generalization bounds do not yet match the practical performances of GNNs. This limitation is partly inherited from the mismatch between the theories and the practices of deep learning in general, as we utilize the results by Neyshabur et al. [109] to illustrate the characteristics of the neural-network part of GNNs. In future work, we hope to adapt stronger PAC-Bayesian bounds for neural networks under IID setup [164, 43] to the non-IID setup for GNNs.

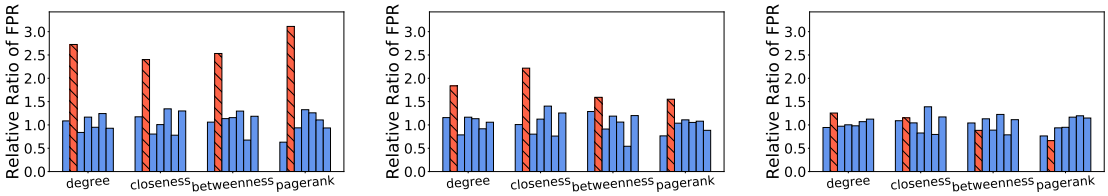
Another limitation is that we have assumed a particular form of GNNs similar as SGC [148] or APPNP [79]. This form of GNNs simplifies the analysis but does not include some common GNNs such as GCN [77] and GAT [141]. We notice that the key characteristics of GNNs we need for the analysis are that the change of outputs of GNNs under certain perturbations needs to be bounded. A recent work [89] has shown that some more general forms of GNNs (including GCN) indeed have bounded output changes under perturbations. So the analysis in this work can be potentially adapted to more general forms of GNNs by utilizing such perturbation bounds. Empirically, we have demonstrated that the accuracy disparity phenomenon predicted by our theoretical analysis indeed appears in experiments on GCN, GAT, and GraphSAGE.

Finally, our analysis requires some assumptions on the relationship between the training set and the target test subgroup. While, not surprisingly, we have to make some assumptions about this relationship to expect good generalization to the target subgroup, it is an interesting future direction to explore more relaxed assumptions than the ones used in this work.

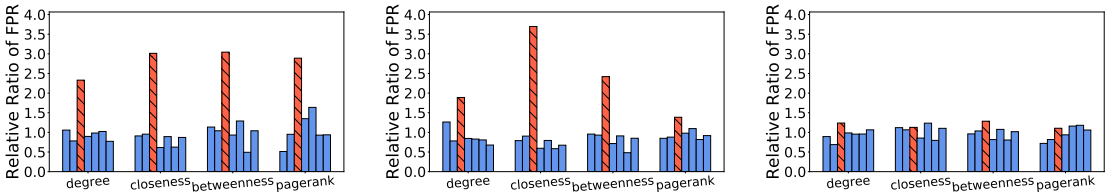
C.4.4 Societal Impacts

As GNNs have been deployed in human-related real-world applications such as recommender systems [157], understanding the fairness issues of GNNs may have direct societal impacts. On the positive side, understanding the systematic biases

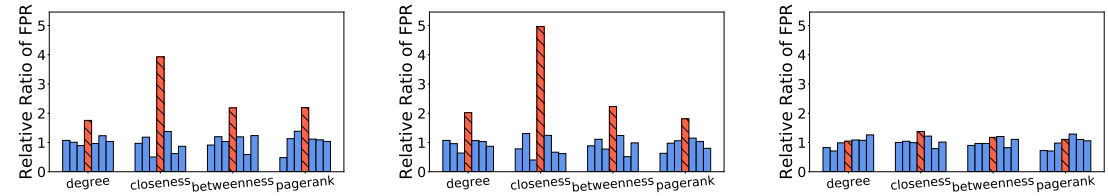
embedded in the GNN models and the graph-structured data helps researchers and practitioners come up with solutions that mitigate the potential harms resulted by such biases. On the negative side, however, such understanding may also be used for malicious purposes: e.g., performing adversarial attacks on GNNs that utilizes systematic biases. Nevertheless, we believe the theoretical understandings resulting from this work contribute to a small step towards making the GNN models more transparent to the research community, which may motivate the design of better and fairer models.



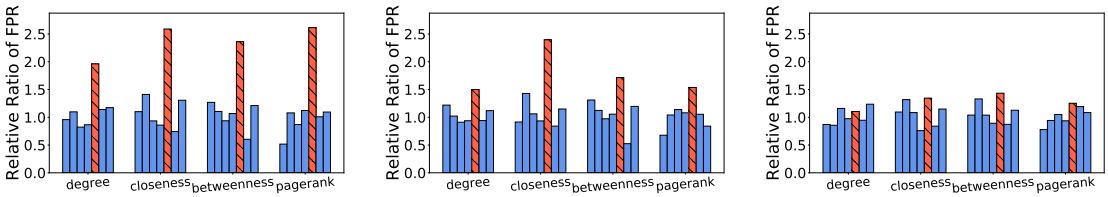
(a) Model: GCN. Class: 1. (b) Model: GAT. Class: 1. (c) Model: MLP. Class: 1.



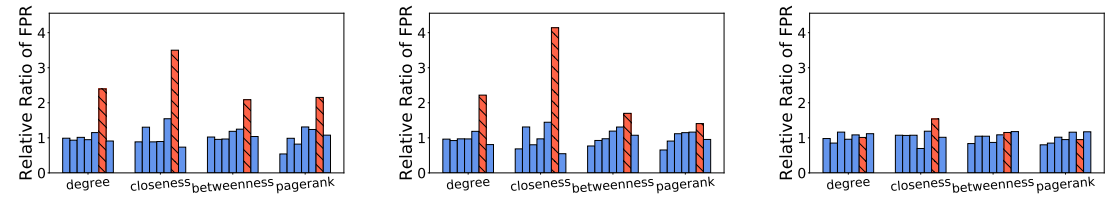
(d) Model: GCN. Class: 2. (e) Model: GAT. Class: 2. (f) Model: MLP. Class: 2.



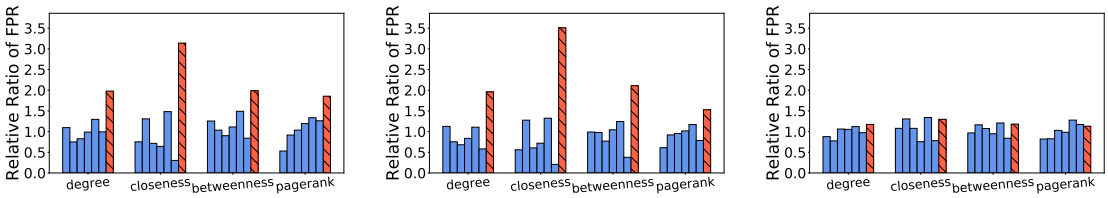
(g) Model: GCN. Class: 3. (h) Model: GAT. Class: 3. (i) Model: MLP. Class: 3.



(j) Model: GCN. Class: 4. (k) Model: GAT. Class: 4. (l) Model: MLP. Class: 4.

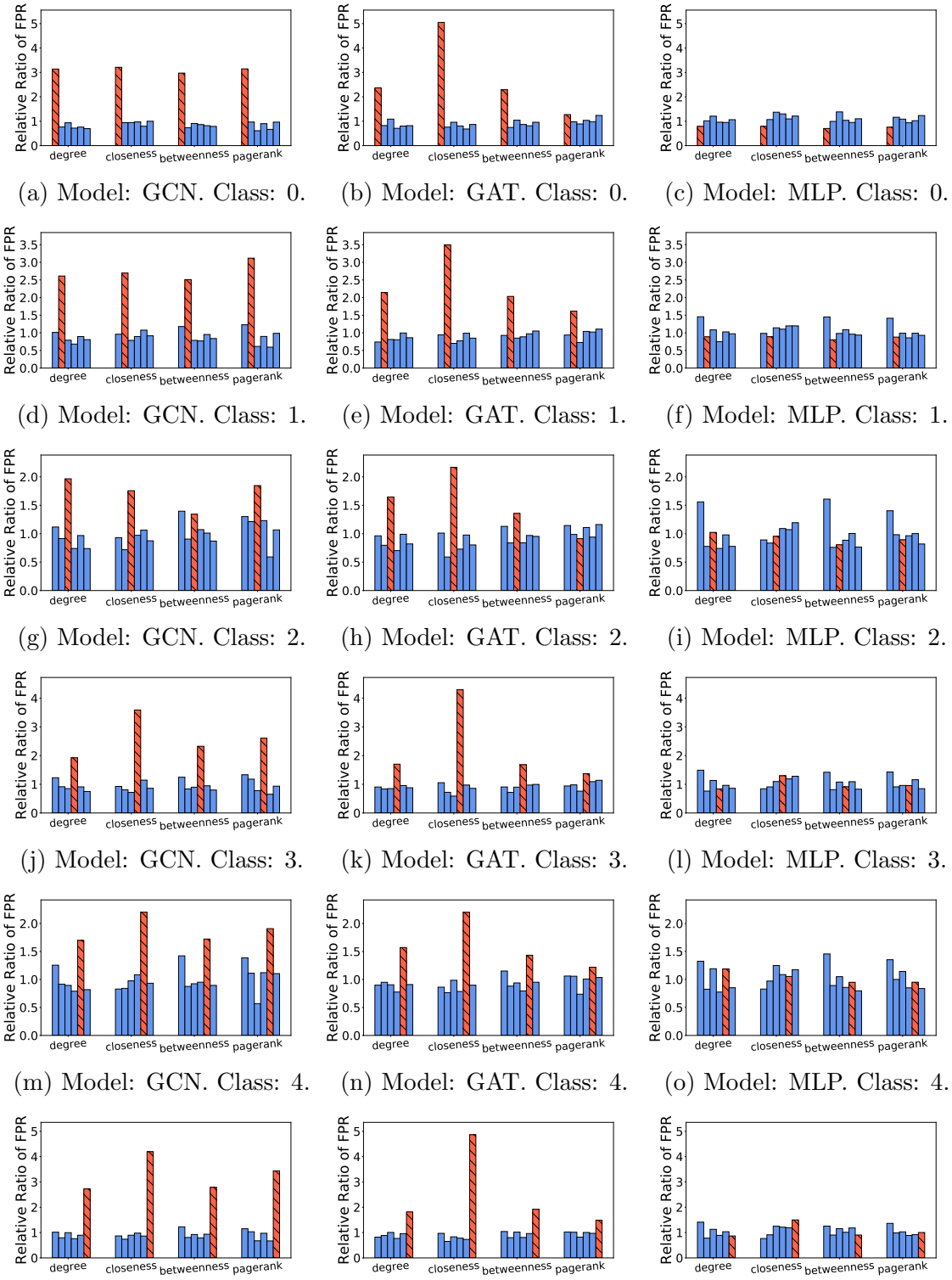


(m) Model: GCN. Class: 5. (n) Model: GAT. Class: 5. (o) Model: MLP. Class: 5.



(p) Model: GCN. Class: 6. (q) Model: GAT. Class: 6. (r) Model: MLP. Class: 6.

Figure C.14: Relative ratio of FPR in the biased training node selection experiment. Remaining results on Cora besides Figure V.5. Each row corresponds to a different dominant class of choice. See Figure V.5 for the plot settings.



(a) Model: GCN. Class: 0. (b) Model: GAT. Class: 0. (c) Model: MLP. Class: 0.

(d) Model: GCN. Class: 1. (e) Model: GAT. Class: 1. (f) Model: MLP. Class: 1.

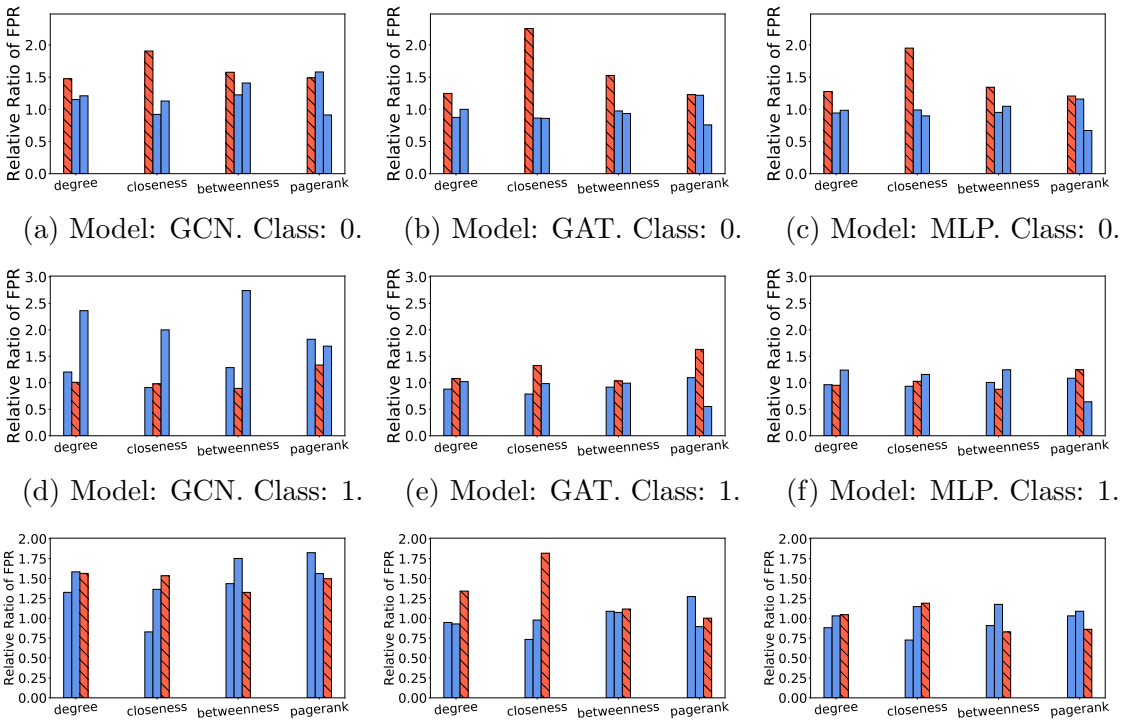
(g) Model: GCN. Class: 2. (h) Model: GAT. Class: 2. (i) Model: MLP. Class: 2.

(j) Model: GCN. Class: 3. (k) Model: GAT. Class: 3. (l) Model: MLP. Class: 3.

(m) Model: GCN. Class: 4. (n) Model: GAT. Class: 4. (o) Model: MLP. Class: 4.

(p) Model: GCN. Class: 5. (q) Model: GAT. Class: 5. (r) Model: MLP. Class: 5.

Figure C.15: Relative ratio of FPR in the biased training node selection experiment. Full results on Citeseer. Each row corresponds to a different dominant class of choice. See Figure V.5 for the plot settings.



(a) Model: GCN. Class: 0. (b) Model: GAT. Class: 0. (c) Model: MLP. Class: 0.

(d) Model: GCN. Class: 1. (e) Model: GAT. Class: 1. (f) Model: MLP. Class: 1.

(g) Model: GCN. Class: 2. (h) Model: GAT. Class: 2. (i) Model: MLP. Class: 2.

Figure C.16: Relative ratio of FPR in the biased training node selection experiment. Full results on Pubmed. Each row corresponds to a different dominant class of choice. See Figure V.5 for the plot settings.

APPENDIX D

Appendix of Chapter VI

D.1 Proofs

D.1.1 Proof of Proposition D.1.1

Before we start the proof of Proposition 4, we first introduce a helpful lemma below.

Lemma D.1.1. *Under Assumption 6.4.2, for any $i \in S_{te}$, if $S_{tr} \cap T(i) \neq \emptyset$, letting $\tau(i) := \arg\min_{l \in S_{tr} \cap T(i)} \|g(v_i) - g(v_l)\|_2$ and $\epsilon_i := \|g(v_i) - g(v_{\tau(i)})\|_2$, then we have*

$$||f(v_i) - f(v_{\tau(i)})||_\infty \leq \delta_h \epsilon_i. \tag{D.1}$$

Proof of Lemma D.1.1.

$$\begin{aligned}
&||f(v_i) - f(v_{\tau(i)})||_\infty \\
&= ||h(g((v_i)) - h(g(v_{\tau(i)}))||_\infty \\
&\leq \delta_h ||g((v_i) - g(v_{\tau(i)})||_2 \qquad (\textit{Assumption } 6.4.2) \\
&= \delta_h \epsilon_i.
\end{aligned}$$

□

We now start the proof of Proposition 4.

Proof of Proposition 4. We first consider the expected difference between the loss of v_i and $v_{\tau(i)}$:

$$
\begin{aligned}
&\mathbb{E}_{y_i}[\mathcal{L}_0(f(v_i), y_i)] - \mathbb{E}_{y_{\tau(i)}}[\mathcal{L}_{\gamma_i}(f(v_{\tau(i)}), y_{\tau(i)})] \\
=&\sum_{c=1}^{C} P[y_i = c \mid v_i]\mathcal{L}_0(f(v_i), c) \\
&- \sum_{c=1}^{C} P[y_{\tau(i)} = c \mid v_{\tau(i)}]\mathcal{L}_{\gamma_i}(f(v_{\tau(i)}), c) \\
=&\sum_{c=1}^{C} \eta_c(v_i)\mathcal{L}_0(f(v_i), c) - \sum_{c=1}^{C} \eta_c(v_{\tau(i)})\mathcal{L}_{\gamma_i}(f(v_{\tau(i)}), c) \\
=&\sum_{c=1}^{C} \eta_c(v_i)\Big[\mathcal{L}_0(f(v_i), c) - \mathcal{L}_{\gamma_i}(f(v_{\tau(i)}), c)\Big] \\
&+ \sum_{c=1}^{C} \Big[\eta_c(v_i) - \eta_c(v_{\tau(i)})\Big]\mathcal{L}_{\gamma_i}(f(v_{\tau(i)}), c).
\end{aligned}
$$

Given the smoothness assumption on the class-specific regression function, the second term can be bounded as:

$$
\begin{aligned}
&\sum_{c=1}^{C} \Big[\eta_c(v_i) - \eta_c(v_{\tau(i)})\Big]\mathcal{L}_{\gamma_i}(f(v_{\tau(i)}), c) \\
\leq&\sum_{c=1}^{C} \Big[\eta_c(v_i) - \eta_c(v_{\tau(i)})\Big] && (\mathcal{L}_{\gamma_i}(f(v_{\tau(i)}), c) \leq 1) \\
\leq&\sum_{c=1}^{C} \Big|\eta_c(v_i) - \eta_c(v_{\tau(i)})\Big| \\
\leq&\sum_{c=1}^{C} \delta_\eta \Big|\Big|g(v_i) - g(v_{\tau(i)})\Big|\Big|_2 && (\textit{Assumption } 6.4.1) \\
=&\sum_{c=1}^{C} \delta_\eta \epsilon_i \\
=&C\delta_\eta \epsilon_i.
\end{aligned}
$$

We then prove that for $\gamma_i = 2\delta_h\epsilon_i$, the first term is non-positive. We prove it by discussing two cases:

i). $\mathcal{L}_{\gamma_i}(f(v_{\tau(i)}), c) = 1$: In this case $\mathcal{L}_0(f(v_i), c) - \mathcal{L}_{\gamma_i}(f(v_{\tau(i)}), c) \leq 0$ always holds and the first term is non-positive.

ii). $\mathcal{L}_{\gamma_i}(f(v_{\tau(i)}), c) = 0$: This situation only happens if:

$$f(v_{\tau(i)})[c] > \gamma_i + \max_{b \neq c} f(v_{\tau(i)})[b].$$

By Lemma D.1.1, we know that

$$f(v_i)[c] \geq f(v_{\tau(i)})[c] - \delta_h\epsilon_i, \quad \text{and}$$

$$f(v_i)[b] \leq f(v_{\tau(i)})[b] + \delta_h\epsilon_i, \forall b = 1, \ldots, C.$$

Therefore,

$$\begin{aligned}
& f(v_i)[c] - \max_{b \neq c} f(v_i)[b] \\
\geq & \Big(f(v_{\tau(i)})[c] - \delta_h\epsilon_i \Big) - \Big(\max_{b \neq c} f(v_{\tau(i)})[b] + \delta_h\epsilon_i \Big) \\
= & f(v_{\tau(i)})[c] - 2\delta_h\epsilon_i - \max_{b \neq c} f(v_{\tau(i)})[b] \\
= & f(v_{\tau(i)})[c] - \gamma_i - \max_{b \neq c} f(v_{\tau(i)})[b] \\
> & 0,
\end{aligned}$$

which implies $\mathcal{L}_0(f(v_i), c) = 0$. Hence $\mathcal{L}_0(f(v_i), c) - \mathcal{L}_{\gamma_i}(f(v_{\tau(i)}), c) = 0$ and the first term is non-positive.

Therefore, for $\gamma_i = 2\delta_h\epsilon_i$,

$$\mathbb{E}_{y_i}[\mathcal{L}_0(f(v_i), y_i)] \leq C\delta_\eta\epsilon_i + \mathbb{E}_{y_{\tau(i)}}[\mathcal{L}_{2\delta_h\epsilon_i}(f(v_{\tau(i)}), y_{\tau(i)})].$$

□

D.2 Experiment Reproducibility

D.2.1 Datasets Descriptions

We experiment on citation networks Citeseer, Cora, and Pubmed [126], three standard node classification benchmarks. We also experiment on Corafull [11] and Ogbn-Arxiv [63], for performance on denser network with more classes, and on co-authorship networks [130] for diversity. The summary statistics of the datasets are provided in Table D.1.

Table D.1: Summary statistics of datasets.

Dataset	# Nodes	# Edges	# Features	# Classes	# Partitions
Cora	2,708	5,278	1,433	7	7
Citeseer	3,327	4,552	3,703	6	14
Pubmed	19,717	44,324	500	3	8
Coauthor-CS	18,333	81,894	6,805	15	6
Coauthor-Physics	34,493	247,962	2,000	5	5
Corafull	19,793	126,842	8,710	70	7
Ogbn-arxiv	169,343	1,166,243	128	40	9

D.2.2 GNN Architectures and Training Setup

We perform experiments over different popular GNN models, including a 2-layer GCN [77] with 16 hidden neurons, a 2-layer GraphSAGE [57] with 16 hidden neurons and a 2-layer GAT [141] with 8 attention heads with 8 hidden neurons each. For Ogbn-Arxiv dataset, the number of hidden neurons is increased to 128. To train each model, we use an Adam optimizer with initial learning rate of 1×10^{-2} and weight decay of 5×10^{-4}. As in the active learning setup, there are not enough labeled samples to be used as a validation set, we train the GNN model with fixed 300 epochs in all the experiments.

D.2.3 Implementation Details

We adopt PyTorch Geometric [46] for dataset preprocessing and model construction in our experiments. For **CoreSet** [127], since the MIP optimized version is not scalable to large datasets, we use the time-efficient greedy approximation version by choosing node closest to the cluster centers. To adapt **AGE** [19] for one-shot setting, we stick to the default parameters in the original work ($\gamma = 0.3$ for Citeseer, $\gamma = 0.7$ for Cora and $\gamma = 0.9$ for Pubmed. For benchmarks not mentioned, we set $\gamma = 0.8$. For methods that depend on K-Medoids, we apply an efficient approximation by [112] by initializing with K-Means++ [5] and selecting nodes closest to centers.

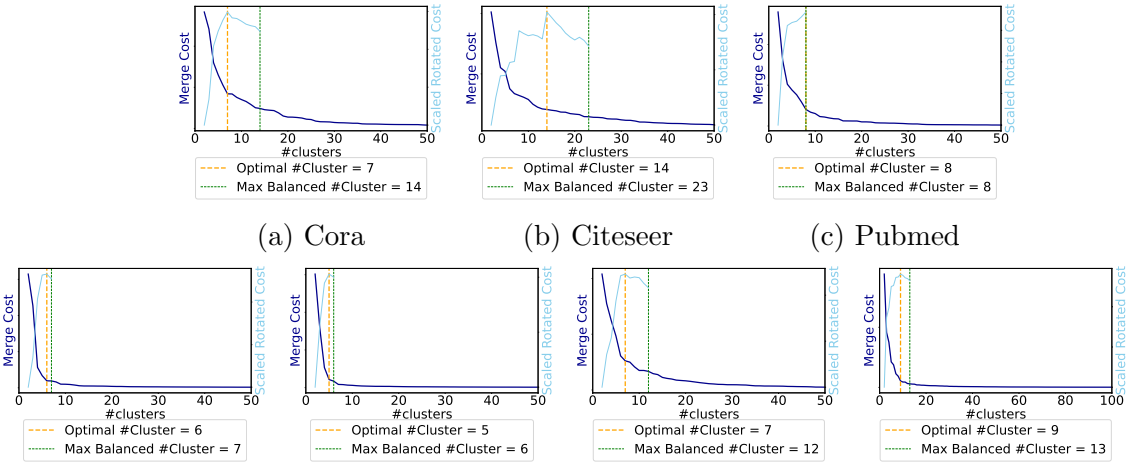
D.3 Graph Partitioning Details

When merging small outlying communities, we applied Ward's method for agglomerative hierarchical clustering [146], which defines the cost of merging two clusters as the increase of sum of squares. We define a balance score as the ratio of the average cluster sizes over the size of the smallest cluster. When the score is no larger than 3, the process reaches its balance with $K_{balance}$ communities remaining. To determine the best number of $K < K_{balance}$ communities in the graph, we find the elbow of the costs using the `kneebow` [1] toolkit. The value of K for each dataset is illustrated in Figure D.1.

D.4 Addendum to Experiments

The complete experimental results are summarized in Figures D.2, D.3, D.4, D.5 and Tables D.2, D.3, D.4, D.5. For Ogbn-Arxiv, we reported both the Macro and Micro (Accuracy) F1 score. As can be seen in Tables D.5, no statistically significant difference can be observed from the Micro-F1 evaluation among the active learning

[1] `https://pypi.org/project/kneebow/`



(a) Cora (b) Citeseer (c) Pubmed

(d) Coauthor-CS (e) Coauthor-Physics (f) Corafull (g) Ogbn-Arxiv

Figure D.1: A summary of the number of partitions, automatically determined by the elbow method.

methods when the budget is larger than 320, but our methods significantly outperformed baselines on Macro-F1.

D.5 Case Study

To illustrate our method, we created a synthetic dataset using the Contextual Stochastic Block Model (CSBM) [39], with 300 nodes, 100 features and random noise in labels and edges. The graph has $C = 3$ classes, each with 2 graph communities. We experiment with $6C$ budget, and the result is visualized in Figure D.6. As can be seen from the visualization, our method makes sure the selected nodes comes from different communities, avoiding under-representing some regions of graph.

D.6 More Analysis

In this section, we provide more detailed descriptions and results supplementing Section 6.5.3 in the main paper.

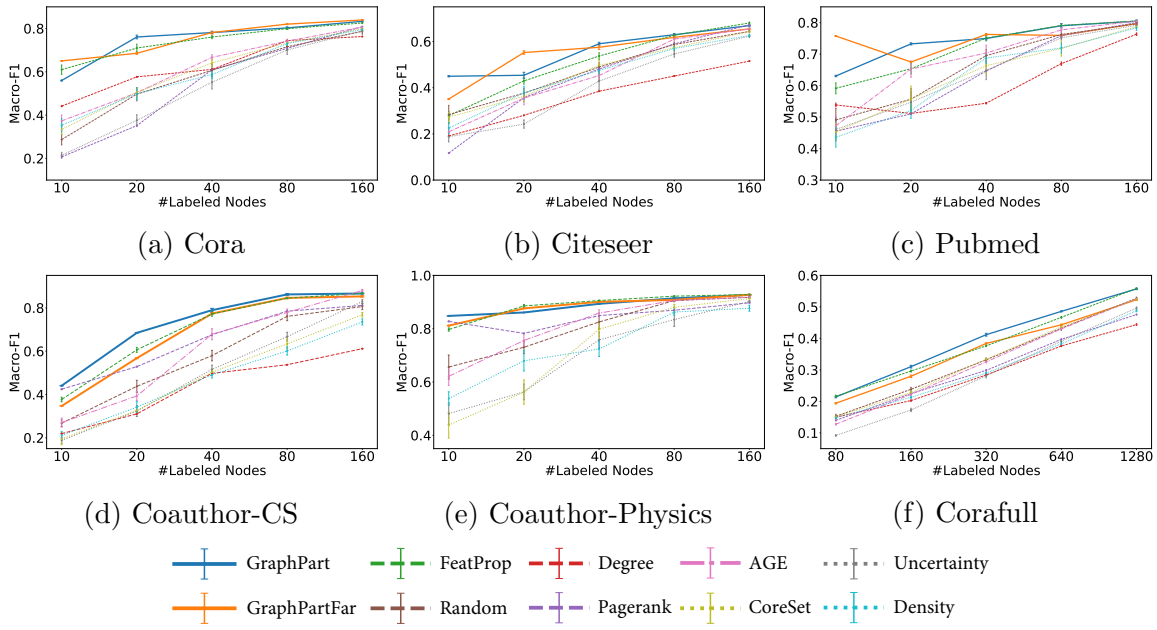


Figure D.2: Results of each baseline on benchmark datasets on GCN averaged from 10 different runs.

Table D.2: Summary of the performance of GCN on each baseline on each benchmark datasets. All the markers follow those of Table 6.1.

Baselines	Cora			Citeseer			Pubmed		
	20	40	80	10	20	40	10	20	40
Random	49.7 ± 9.7	60.7 ± 8.7	71.6 ± 5.0	28.4 ± 12.6	37.6 ± 6.7	48.9 ± 5.8	49.1 ± 11.4	55.7 ± 10.6	69.5 ± 6.2
Uncertainty	37.6 ± 8.2	55.2 ± 10.0	70.1 ± 6.4	18.8 ± 7.1	24.2 ± 5.7	42.8 ± 12.5	46.0 ± 11.5	54.7 ± 10.4	64.8 ± 9.2
Density	49.8 ± 8.6	58.7 ± 5.5	73.1 ± 4.1	22.6 ± 7.5	37.8 ± 9.0	47.5 ± 6.4	43.6 ± 9.8	52.1 ± 8.0	68.7 ± 5.3
CoreSet	50.6 ± 6.2	64.1 ± 5.3	73.4 ± 3.3	27.2 ± 6.6	36.3 ± 7.8	49.6 ± 7.5	45.3 ± 8.5	55.7 ± 13.0	66.3 ± 8.6
Degree	57.7 ± 0.6	61.1 ± 1.3	74.4 ± 0.5	19.1 ± 0.1	28.0 ± 0.3	38.5 ± 0.3	53.9 ± 1.9	51.2 ± 0.9	54.4 ± 0.7
Pagerank	35.1 ± 0.9	60.2 ± 1.2	70.8 ± 0.6	11.7 ± 0.3	33.0 ± 0.5	45.6 ± 0.9	45.5 ± 0.6	51.0 ± 0.6	66.4 ± 0.1
AGE	50.3 ± 5.0	66.7 ± 4.1	74.2 ± 2.7	20.9 ± 6.3	35.5 ± 9.2	45.2 ± 7.7	47.2 ± 9.9	65.2 ± 8.3	70.3 ± 8.0
FeatProp	71.0 ± 5.7	76.1 ± 2.5	79.9 ± 0.9	27.9 ± 5.5	42.9 ± 4.5	53.7 ± 4.5	59.1 ± 5.5	65.4 ± 5.2	75.1 ± 2.8
GraphPart	**76.1*** ± 2.7	78.1 ± 1.5	80.3 ± 1.6	**45.0*** ± 0.7	45.4 ± 4.1	**59.0*** ± 2.0	63.0 ± 0.7	**73.2*** ± 1.0	74.9 ± 1.3
GraphPartFar	68.6 ± 1.5	**78.1** ± 2.1	**82.0*** ± 0.7	35.1* ± 0.6	**55.2*** ± 2.4	57.5* ± 2.9	**75.7*** ± 0.3	67.5 ± 0.5	**76.2** ± 0.9

Baselines	Co-CS			Co-Physics			Corafull		
	20	40	80	10	20	40	160	320	640
Random	43.8 ± 8.8	58 ± 7.7	76.1 ± 6.3	65.6 ± 14.3	73.1 ± 7.5	82.5 ± 7.6	23.8 ± 2.0	33.2 ± 1.8	43.2 ± 2.1
Uncertainty	32.4 ± 8.0	51.6 ± 6.2	66.7 ± 6.4	48.1 ± 12.6	56.3 ± 9.3	75.8 ± 9.2	17.3 ± 1.7	28.1 ± 2.1	39.1 ± 1.2
Density	34.2 ± 8.0	49.1 ± 4.6	60.1 ± 6.0	53.8 ± 8.0	67.9 ± 12.0	72.4 ± 8.8	21.3 ± 1.1	29.0 ± 0.9	38.3 ± 1.1
CoreSet	32.6 ± 4.3	50.3 ± 4.8	63.4 ± 7.2	43.8 ± 15.3	56.2 ± 14.4	79.8 ± 8.3	22.8 ± 1.4	33.4 ± 1.1	43.7 ± 0.5
Degree	31.1 ± 0.5	49.8 ± 0.5	53.8 ± 0.1	13.4 ± 0.0	13.4 ± 0.0	13.4 ± 0.0	20.3 ± 0.7	28.4 ± 0.6	37.6 ± 0.6
Pagerank	52.9 ± 0.5	67.6 ± 1.5	78.5 ± 0.5	82.8 ± 0.4	78.2 ± 0.2	84.9 ± 0.2	22.5 ± 0.4	29.9 ± 0.7	39.7 ± 0.5
AGE	39.3 ± 6.6	67.8 ± 8.0	77.9 ± 5.0	62.1 ± 10.6	75.6 ± 7.9	85.9 ± 3.8	22.2 ± 1.4	32.6 ± 1.1	43.0 ± 0.9
FeatProp	60.6 ± 3.6	77.2 ± 3.2	84.5 ± 0.6	79.7 ± 1.9	**88.6** ± 1.7	**90.6*** ± 0.4	29.6 ± 1.0	37.6 ± 0.8	46.7 ± 0.8
GraphPart	**68.5*** ± 0.2	**78.9** ± 2.7	**86.2*** ± 1.2	**84.8*** ± 0.2	86.1 ± 0.1	89.3 ± 0.2	**31.0*** ± 1.3	**41.2*** ± 1.4	**48.6** ± 0.5
GraphPartFar	56.8 ± 0.7	77.4 ± 1.1	84.6 ± 1.3	81.2 ± 0.3	87.6 ± 0.4	90.1 ± 0.3	28.0 ± 1.2	38.4 ± 0.6	44.3 ± 0.7

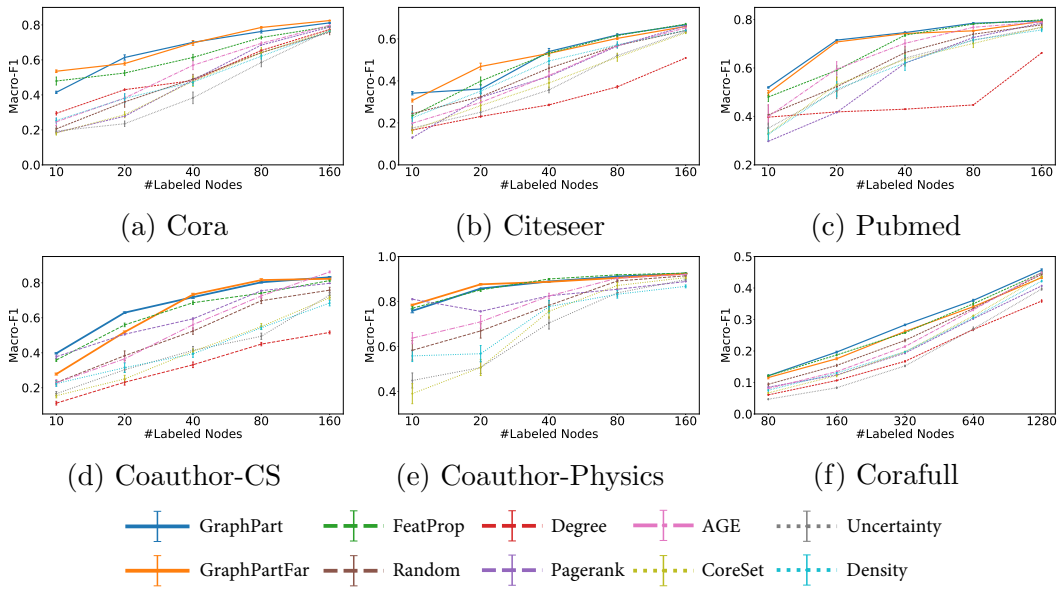


Figure D.3: Results of each baseline on benchmark datasets on GraphSAGE averaged from 10 different runs.

Table D.3: Summary of the performance of GraphSAGE on each baseline on each benchmark datasets. All the markers follow those of Table 6.1

Baselines	Cora			Citeseer			Pubmed		
	20	40	80	10	20	40	10	20	40
Random	36.0 ± 9.6	49.1 ± 8.5	64.2 ± 5.6	24.5 ± 11.9	32.4 ± 6.6	46.1 ± 5.6	40.6 ± 13.0	52.3 ± 12.2	66.3 ± 7.9
Uncertainty	23.5 ± 5.0	38.3 ± 10.6	58.6 ± 7.8	17.6 ± 6.3	25.1 ± 6.1	35.7 ± 4.4	35.2 ± 6.6	50.5 ± 10.1	64.1 ± 10.5
Density	38.0 ± 8.3	47.3 ± 6.2	62.1 ± 5.4	22.4 ± 9.1	35.1 ± 4.9	49.5 ± 4.7	32.7 ± 8.9	51.2 ± 10.1	62.1 ± 9.9
CoreSet	28.7 ± 5.4	47.8 ± 9.4	64.1 ± 9.2	16.6 ± 5.2	28.2 ± 7.4	39.0 ± 7.4	32.9 ± 7.8	53.1 ± 10.2	63.5 ± 7.4
Degree	43.0 ± 1.5	48.2 ± 1.4	65.4 ± 2.1	16.6 ± 0.9	23.1 ± 1.1	28.6 ± 1.2	39.7 ± 0.5	41.9 ± 0.1	43.0 ± 0.1
Pagerank	27.8 ± 1.6	47.9 ± 1.2	68.5 ± 0.9	13.0 ± 1.0	29.8 ± 1.3	38.3 ± 2.2	29.7 ± 0.3	41.7 ± 0.6	62.9 ± 0.3
AGE	38.2 ± 8.0	56.9 ± 7.8	69.5 ± 2.7	19.8 ± 5.0	29.7 ± 4.9	42.6 ± 6.0	39.9 ± 15.4	59.5 ± 9.8	70.2 ± 4.7
FeatProp	52.5 ± 4.6	61.4 ± 5.5	72.8 ± 2.6	23.4 ± 4.3	39.9 ± 6.2	53.3 ± 3.8	48.0 ± 5.9	59.1 ± 6.0	73.6 ± 1.7
GraphPart	**61.4*** ± 4.7	**70.0*** ± 2.4	76.2* ± 2.7	**34.1*** ± 2.6	36.1 ± 6.4	**54.0** ± 4.6	**52.0** ± 0.8	**71.5*** ± 0.5	**74.6** ± 1.1
GraphPartFar	57.9* ± 2.8	69.7* ± 4.2	**78.6*** ± 1.5	30.7* ± 2.3	**46.9** ± 5.0	53.1 ± 4.0	49.7 ± 3.1	70.7* ± 1.6	74.2 ± 0.4

Baselines	Co-CS			Co-Physics			Corafull		
	20	40	80	10	20	40	160	320	640
Random	38.3 ± 8.7	52.4 ± 5.8	69.7 ± 5.2	58.3 ± 13.8	66.9 ± 10.1	78.3 ± 7.1	15.4 ± 1.1	23.4 ± 1.4	33.5 ± 1.3
Uncertainty	36.5 ± 6.3	56.0 ± 9.4	72.5 ± 3.5	44.9 ± 10.6	50.6 ± 8.4	70.5 ± 8.5	15.3 ± 0.8	27.0 ± 1.9	39.7 ± 1.1
Density	31.5 ± 8.5	39.4 ± 6.2	54.3 ± 3.3	55.8 ± 7.9	56.8 ± 11.5	77.8 ± 8.9	13.0 ± 1.2	19.7 ± 0.4	30.6 ± 1.0
CoreSet	25.0 ± 6.5	40.8 ± 4.3	55.0 ± 5.3	39.0 ± 14.1	50.7 ± 11.3	75.6 ± 9.5	12.2 ± 1.2	19.7 ± 0.6	31.2 ± 0.7
Degree	23.1 ± 5.2	33.1 ± 5.1	44.9 ± 3.0	13.4 ± 0.0	13.4 ± 0.0	13.4 ± 0.0	10.6 ± 0.4	16.7 ± 0.7	26.8 ± 0.7
Pagerank	50.6 ± 1.3	59.4 ± 1.8	75.5 ± 0.7	**81.0*** ± 0.8	75.6 ± 0.7	82.6 ± 0.4	12.4 ± 0.3	19.4 ± 0.8	30.3 ± 0.4
AGE	36.5 ± 6.3	56.0 ± 9.4	72.5 ± 3.5	63.7 ± 7.8	71.0 ± 8.8	82.4 ± 3.9	13.5 ± 0.8	21.4 ± 0.8	33.1 ± 1.0
FeatProp	55.9 ± 3.2	68.6 ± 3.2	74.1 ± 1.7	77.0 ± 2.4	85.2 ± 2.2	**90.0** ± 0.7	18.7 ± 0.8	25.8 ± 0.7	35.0 ± 1.6
GraphPart	**62.9*** ± 1.4	71.8* ± 2.2	80.3* ± 1.3	75.8 ± 2.7	85.8 ± 0.3	88.9 ± 0.3	**19.7*** ± 0.9	**28.3** ± 0.7	**36.1*** ± 1.0
GraphPartFar	51.9 ± 1.9	**73.3*** ± 2.2	**81.5*** ± 2.4	78.5 ± 1.1	**87.6*** ± 0.9	88.8 ± 0.5	17.5 ± 1.0	26.2 ± 1.4	34.1 ± 0.9

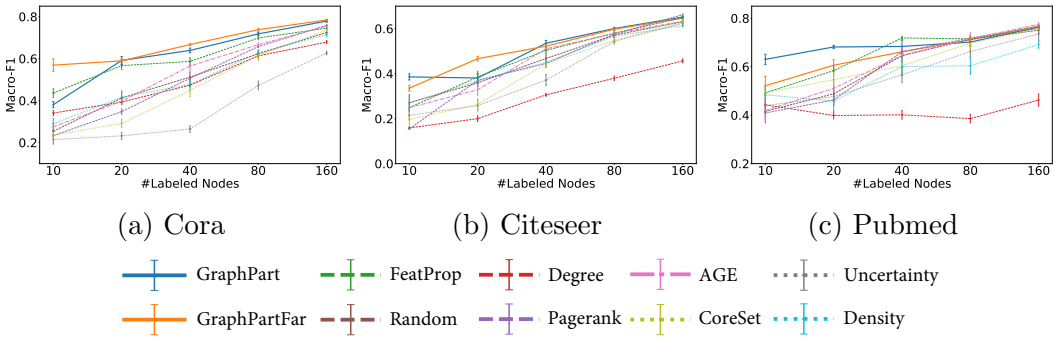


Figure D.4: Results of each baseline on benchmark datasets on GAT averaged from 10 different runs.

Table D.4: Summary of the performance of GAT on each baseline on citation networks. All the markers follow those of Table 6.1

Baselines	Cora			Citeseer			Pubmed		
	20	40	80	10	20	40	10	20	40
Random	41.2 ± 9.4	51.2 ± 8	62.3 ± 4.3	27.1 ± 11.4	35.9 ± 4.9	46.7 ± 5.3	41.6 ± 9.5	48.8 ± 11.6	64.5 ± 7.7
Uncertainty	23.2 ± 5.5	26.5 ± 4.9	47.2 ± 6.6	21.5 ± 6.7	25.8 ± 7.5	37.1 ± 8.0	43.9 ± 10.5	47.7 ± 11.2	56.7 ± 10.2
Density	41.6 ± 9.7	47.7 ± 5.6	62.8 ± 5.0	27.0 ± 9.0	36.7 ± 6.8	45.1 ± 6.8	48.4 ± 9.8	45.8 ± 13.5	60.1 ± 11.2
CoreSet	29.1 ± 6.2	44.9 ± 9.1	61.5 ± 6.9	19.6 ± 9.8	26.3 ± 7.9	44.3 ± 5.6	49.3 ± 12.9	54.6 ± 10.2	60.4 ± 13.0
Degree	39.3 ± 3.5	47.4 ± 2.3	61.3 ± 3.3	15.9 ± 1.2	20.0 ± 4.0	30.6 ± 2.0	44.3 ± 8.4	39.9 ± 4.3	40.2 ± 5.5
Pagerank	34.8 ± 4.1	50.3 ± 2.5	65.7 ± 1.2	15.4 ± 0.6	33.0 ± 2.4	42.5 ± 1.9	40.9 ± 13	46.3 ± 7.0	61.7 ± 3.7
AGE	39.8 ± 5.5	56.5 ± 6.3	66.7 ± 2.7	24.9 ± 12.5	32.9 ± 7.8	51.0 ± 5.0	42.0 ± 13.4	51.1 ± 9.7	64.8 ± 9.6
FeatProp	56.7 ± 5.8	58.7 ± 5.7	69.8 ± 2.1	24.9 ± 4.9	38.5 ± 7.9	50.4 ± 4.2	49.2 ± 8.1	58.4 ± 6.8	**71.9** ± 2.4
GraphPart	**59.1** ± 6.0	63.9* ± 3.5	71.8 ± 2.4	**38.6*** ± 4.3	38.1 ± 5.8	**53.5** ± 4.0	**63.0*** ± 6.6	**68.2** ± 2.2	68.4 ± 8.3
GraphPartFar	58.9 ± 2.9	**66.7*** ± 1.7	**73.8*** ± 2.2	33.6* ± 4.1	**46.6*** ± 3.3	52.2 ± 4.2	52.1 ± 12.4	60.6 ± 7.3	66.2 ± 2.8

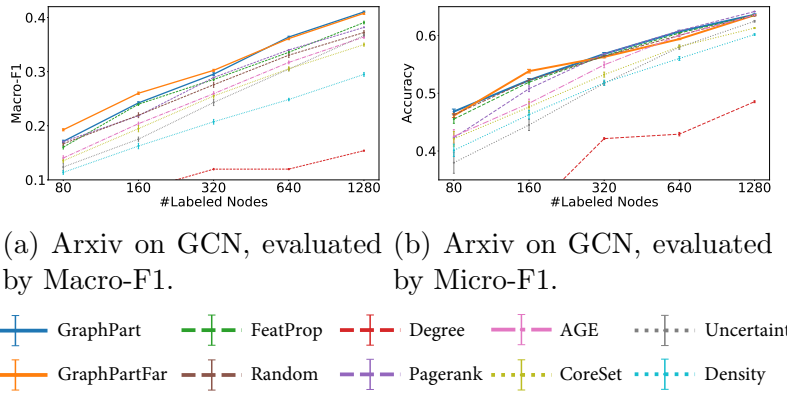
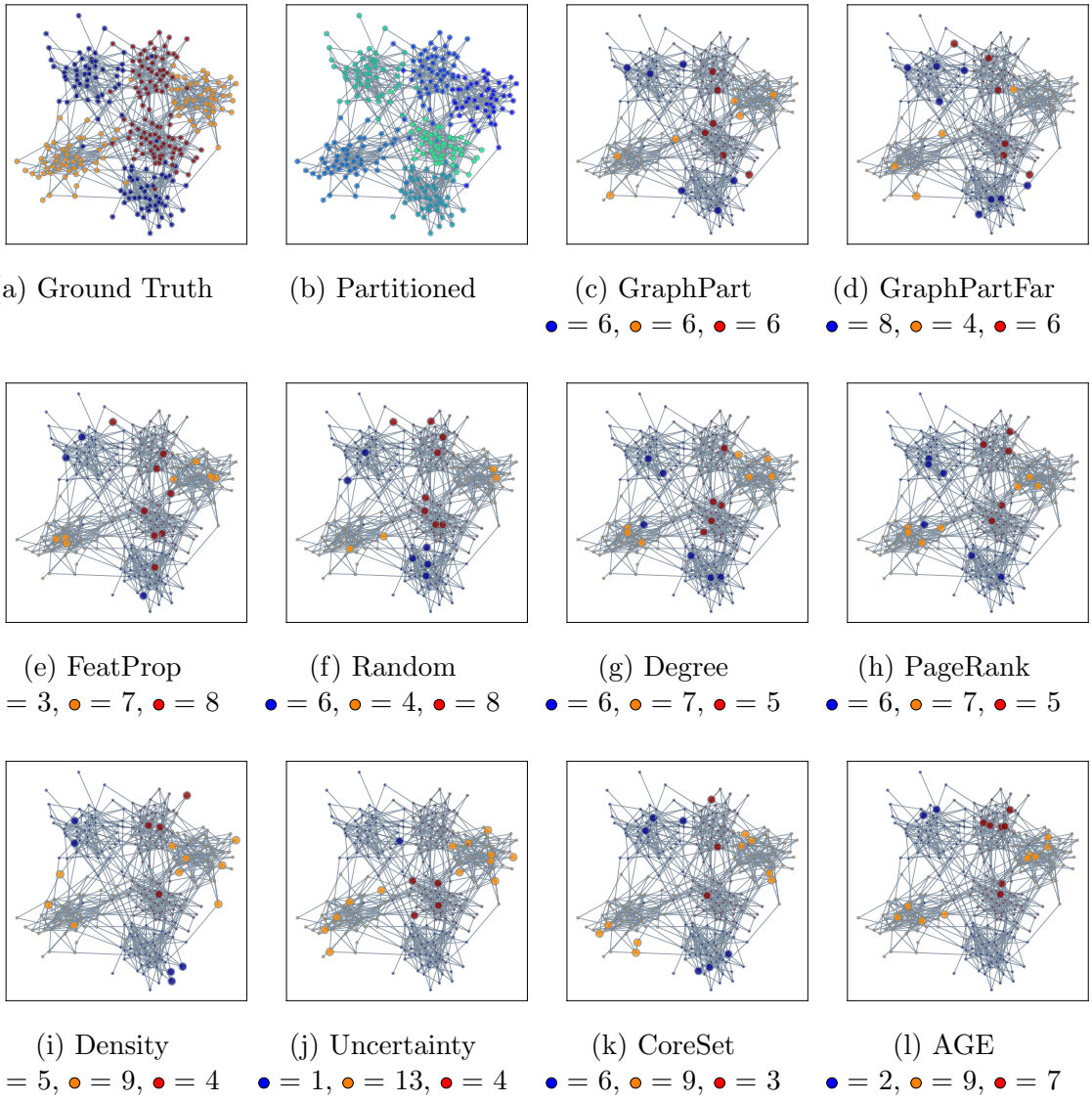


Figure D.5: Results of each baseline on Ogbn-Arxiv datasets on GCN, averaged from 10 different runs.



(a) Ground Truth (b) Partitioned (c) GraphPart ● = 6, ● = 6, ● = 6 (d) GraphPartFar ● = 8, ● = 4, ● = 6

(e) FeatProp ● = 3, ● = 7, ● = 8 (f) Random ● = 6, ● = 4, ● = 8 (g) Degree ● = 6, ● = 7, ● = 5 (h) PageRank ● = 6, ● = 7, ● = 5

(i) Density ● = 5, ● = 9, ● = 4 (j) Uncertainty ● = 1, ● = 13, ● = 4 (k) CoreSet ● = 6, ● = 9, ● = 3 (l) AGE ● = 2, ● = 9, ● = 7

Figure D.6: Results of each baseline on synthetic contextual SBM data, with 300 nodes, 100 features and 3 classes. Each color represent one class and each class has 2 graph communities. The 18 nodes selected by the active learning algorithm are **embolden**.

Table D.5: Summary of the performance of each baseline on large-scale Ogbn-Arxiv dataset. All the markers follow those of Table 6.1.

Baselines	Macro-F1				Micro-F1			
	160	320	640	1280	160	320	640	1280
Random	21.9 ± 1.4	27.6 ± 1.5	33.0 ± 1.4	37.2 ± 1.1	52.3 ± 0.8	56.4 ± 0.8	60.0 ± 0.7	63.5 ± 0.4
Uncertainty	17.5 ± 1.2	24.3 ± 1.8	30.5 ± 1.0	36.7 ± 0.9	44.5 ± 3.0	51.7 ± 1.0	58.0 ± 1.2	62.5 ± 0.4
Density	16.3 ± 1.6	20.7 ± 1.4	24.8 ± 0.7	29.5 ± 1.1	46.3 ± 2.3	51.8 ± 1.4	56.0 ± 1.1	60.2 ± 0.5
CoreSet	19.5 ± 1.8	25.5 ± 1.2	30.5 ± 1.2	35.0 ± 0.9	47.6 ± 2.9	53.2 ± 1.4	58.1 ± 0.8	61.3 ± 0.3
Degree	8.0 ± 0.2	12.0 ± 0.4	12.0 ± 0.3	15.4 ± 0.3	29.2 ± 0.9	42.1 ± 0.5	42.9 ± 0.9	48.5 ± 0.5
Pagerank	21.8 ± 0.6	28.9 ± 0.1	34.0 ± 0.2	38.2 ± 0.1	50.8 ± 1.6	56.8 ± 0.1	**60.8** ± 0.2	**64.2** ± 0.0
AGE	20.4 ± 0.9	25.9 ± 1.1	31.7 ± 0.8	36.4 ± 0.8	48.3 ± 2.3	54.9 ± 1.6	60.0 ± 0.7	63.5 ± 0.3
FeatProp	24.0 ± 0.7	28.5 ± 0.6	33.6 ± 0.4	39.1 ± 0.8	51.9 ± 1.1	56.5 ± 0.6	60.4 ± 0.4	63.5 ± 0.3
GraphPart	24.2 ± 0.7	29.5* ± 0.8	**36.4*** ± 0.5	**41.0*** ± 0.5	52.3 ± 0.9	**56.8** ± 0.8	60.7 ± 0.6	63.6 ± 0.5
GraphPartFar	**26.0*** ± 0.7	**30.2*** ± 0.6	36.1* ± 0.7	40.8* ± 0.7	**53.8*** ± 1.0	56.3 ± 0.5	59.4 ± 0.3	63.5 ± 0.4

D.6.1 Mitigating Accuracy Disparity of GNNs

Following [99], we analyze the generalization performances by evaluating accuracy disparity across 10 subgroups by aggregated-feature distance from 40 selected nodes (320 for CoraFull). The results on other five datasets not shown in the main paper are provided in Figure D.7. In most datasets, the proposed active learning methods are able to mitigate the accuracy disparity of GCN predictions compared to training with random training nodes.

D.6.2 Sensitivity Analysis on Graph Partitioning

Recall in the main paper that we are interested in the effectiveness of the partition approach when combined with distance metrics on different node representations. Specifically, (1) **Aggregation**: aggregated node features S^2X; (2) **Embedding**: the last hidden layer of GCN trained on $\mathbf{s}_0$; and (3) **Feature**: the original node features. For each choice of distance, we evaluate the active learning performance with or without the graph-partition step. We perform the experiments on Citeseer, Pubmed and Cora. On each dataset, we evaluate each active learning method with budget size of 20, 40 and 80. Each experiment is repeated with 10 random seeds.

As can be seen from Table D.6, the graph-partition step is robustly effective when combined with various types of distance metrics. This suggests that the proposed

Table D.6: Summary of the performance of different combinations of distance metric on Citeseer, Pubmed and Cora datasets with or without using graph partition. The numerical values represent the average Macro-F1 score of 10 independent trials and the error bar denotes the standard error of the mean (all in %). The **bold** marker denotes the better performance between with and without using graph partition, and asterisk (*) means this difference is statistically significant by a pairwise t-test at significance level 0.05.

Node Rep.	With Partition?	Cora			Citeseer			Pubmed		
		20	40	80	20	40	80	20	40	80
Aggregation	yes	**76.1*** ± 2.7	**78.1** ± 1.5	**80.3** ± 1.6	**45.4** ± 4.1	**59.0*** ± 2.0	**67.7** ± 1.5	**73.2*** ± 1.0	74.9 ± 1.3	**79.7** ± 2.3
	no	71.0 ± 5.7	76.1 ± 2.5	79.9 ± 0.9	42.9 ± 4.5	53.7 ± 4.5	67.4 ± 2.0	65.4 ± 5.2	**75.1** ± 2.8	79.5 ± 0.9
Embedding	yes	**61.3*** ± 4.8	**69.4*** ± 3.9	**76.7** ± 3.4	**43.9** ± 7.1	**54.4** ± 2.8	**61.1** ± 2.3	**70.1** ± 8.8	**74.9** ± 2.8	**78.1** ± 2.0
	no	54.5 ± 4.7	62.6 ± 5.7	74.1 ± 3.7	38.2 ± 9.0	50.6 ± 5.1	59.7 ± 2.3	63.4 ± 10.6	70.3 ± 6.6	77.0 ± 1.5
Feature	yes	**65.6*** ± 2.6	**71.0*** ± 2.1	77.2 ± 1.3	**15.7*** ± 0.9	**33.1** ± 3.4	**54.4** ± 1.8	56.6 ± 2.1	67.1 ± 2.1	**77.1** ± 2.0
	no	53.2 ± 5.2	64.0 ± 5.1	**77.4** ± 1.7	6.1 ± 2.4	30.9 ± 4.5	54.0 ± 4.1	**64.3*** ± 8.6	**70.0** ± 4.1	76.6 ± 0.9

graph-partition framework may be able to generalize to the active learning for other graph representation learning methods. It is also interesting to observe that methods with a graph-partition step tend to have lower standard errors.

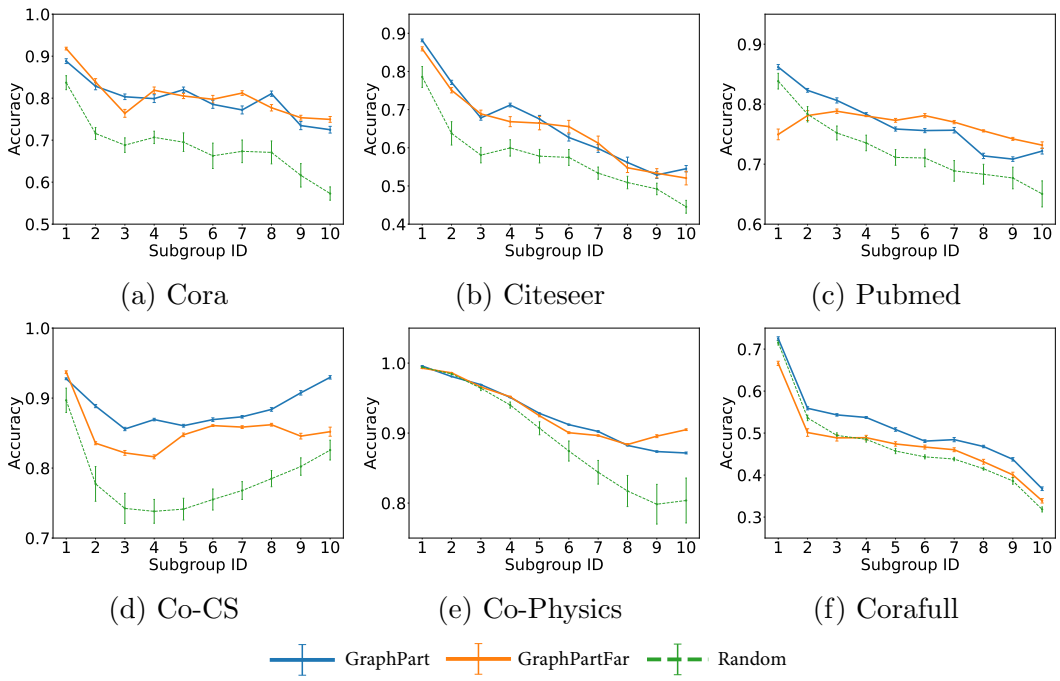


Figure D.7: Accuracy disparity across 10 subgroups. Increasing subgroup indices represent increasing distance to selected training set.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Agarwal, C., H. Lakkaraju, and M. Zitnik (2021), Towards a unified framework for fair and stable graph representation learning, *CoRR*, *abs/2102.13186*.

[2] Allen-Zhu, Z., Y. Li, A. Singh, and Y. Wang (2020), Near-optimal discrete optimization for experimental design: A regret minimization approach, *Mathematical Programming*, pp. 1–40.

[3] Alquier, P., J. Ridgway, and N. Chopin (2016), On the properties of variational approximations of gibbs posteriors, *The Journal of Machine Learning Research*, *17*(1), 8374–8414.

[4] Angwin, J., J. Larson, S. Mattu, and L. Kirchner (2016), Machine bias, in *Ethics of Data and Analytics*, pp. 254–264, Auerbach Publications.

[5] Arthur, D., and S. Vassilvitskii (2007), K-means++: The advantages of careful seeding, in *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 1027–1035, USA.

[6] Baranwal, A., K. Fountoulakis, and A. Jagannath (2021), Graph convolution for semi-supervised classification: Improved linear separability and out-of-distribution generalization, *arXiv preprint arXiv:2102.06966*.

[7] Bauer, A., C. Czado, and T. Klein (2012), Pair-copula constructions for non-gaussian dag models, *Canadian Journal of Statistics*, *40*(1), 86–109.

[8] Bégin, L., P. Germain, F. Laviolette, and J.-F. Roy (2014), Pac-bayesian theory for transductive learning, in *Artificial Intelligence and Statistics*, pp. 105–113, PMLR.

[9] Berlind, C., and R. Urner (2015), Active nearest neighbors in changing environments, in *International Conference on Machine Learning*, pp. 1870–1879, PMLR.

[10] Bojchevski, A., and S. Günnemann (2018), Adversarial attacks on node embeddings via graph poisoning, *arXiv preprint arXiv:1809.01093*.

[11] Bojchevski, A., and S. Günnemann (2018), Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking, in *International Conference on Learning Representations (UCLR)*.

[12] Bonacich, P. (1987), Power and centrality: A family of measures, *American journal of sociology*, *92*(5), 1170–1182.

[13] Borgatti, S. P. (2005), Centrality and network flow, *Social networks*, *27*(1), 55–71.

[14] Bose, A. J., and W. L. Hamilton (2019), Compositional fairness constraints for graph embeddings, *CoRR*, *abs/1905.10674*.

[15] Bowker, G. C., and S. L. Star (2000), *Sorting things out: Classification and its consequences*, MIT press.

[16] Burt, R. S. (1995), *Structural holes*, Harvard University Press, London, England.

[17] Burt, R. S. (2009), *Structural holes: The social structure of competition*, Harvard university press.

[18] Buyl, M., and T. D. Bie (2021), The kl-divergence between a graph model and its fair i-projection as a fairness regularizer, *CoRR*, *abs/2103.01846*.

[19] Cai, H., V. W. Zheng, and K. C.-C. Chang (2017), Active learning for graph embedding, *arXiv preprint arXiv:1705.05085*.

[20] Cameron, A. C., and F. A. Windmeijer (1996), R-squared measures for count data regression models with applications to health-care utilization, *Journal of Business & Economic Statistics*, *14*(2), 209–220.

[21] Carlini, N., and D. Wagner (2017), Towards evaluating the robustness of neural networks, in *2017 ieee symposium on security and privacy (sp)*, pp. 39–57, IEEE.

[22] Carton, S., et al. (2016), Identifying police officers at risk of adverse events, in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 67–76.

[23] Chami, I., S. Abu-El-Haija, B. Perozzi, C. Ré, and K. Murphy (2020), Machine learning on graphs: A model and comprehensive taxonomy, *arXiv preprint arXiv:2005.03675*.

[24] Chang, H., and R. Shokri (2021), On the privacy risks of algorithmic fairness, in *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 292–303, IEEE.

[25] Chang, H., Y. Rong, T. Xu, W. Huang, H. Zhang, P. Cui, W. Zhu, and J. Huang (2020), A restricted black-box adversarial framework towards attacking graph embedding models, in *AAAI Conference on Artificial Intelligence*.

[26] Chen, J., Y. Wu, X. Xu, Y. Chen, H. Zheng, and Q. Xuan (2018), Fast gradient attack on network embedding, *arXiv preprint arXiv:1809.02797*.

[27] Chen, X., G. Yu, J. Wang, C. Domeniconi, Z. Li, and X. Zhang (2019), Activehne: Active heterogeneous network embedding, in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pp. 2123–2129, International Joint Conferences on Artificial Intelligence Organization.

[28] Cheng, H.-F., L. Stapleton, R. Wang, P. Bullock, A. Chouldechova, Z. S. S. Wu, and H. Zhu (2021), Soliciting stakeholders' fairness notions in child maltreatment predictive systems, in *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pp. 1–17.

[29] Clauset, A., M. E. J. Newman, and C. Moore (2004), Finding community structure in very large networks, *Physical Review E*, *70*(6).

[30] Cliff, A. D., and J. K. Ord (1981), *Spatial processes: models & applications*, Taylor & Francis.

[31] Contardo, G., L. Denoyer, and T. Artières (), A meta-learning approach to one-step active learning, in *International Workshop on Automatic Selection, Configuration and Composition of Machine Learning Algorithms*.

[32] Cummings, R., V. Gupta, D. Kimpara, and J. Morgenstern (2019), On the compatibility of privacy and fairness, in *Adjunct Publication of the 27th Conference on User Modeling, Adaptation and Personalization*, pp. 309–315.

[33] Czado, C. (2019), Analyzing dependent data with vine copulas, *Lecture Notes in Statistics, Springer*.

[34] Dai, E., and S. Wang (2021), Say no to the discrimination: Learning fair graph neural networks with limited sensitive attribute information, in *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, WSDM '21, p. 680–688, Association for Computing Machinery, New York, NY, USA, doi:10.1145/3437963.3441752.

[35] Dai, H., H. Li, T. Tian, X. Huang, L. Wang, J. Zhu, and L. Song (2018), Adversarial attack on graph structured data, *arXiv preprint arXiv:1806.02371*.

[36] Dasarathy, G., R. Nowak, and X. Zhu (2015), S2: An efficient graph based active learning algorithm with application to nonparametric classification, in *Proceedings of The 28th Conference on Learning Theory*, pp. 503–522.

[37] Dastin, J. (2018), Amazon scraps secret ai recruiting tool that showed bias against women, in *Ethics of Data and Analytics*, pp. 296–299, Auerbach Publications.

[38] Derrow-Pinion, A., et al. (2021), Eta prediction with graph neural networks in google maps, *arXiv preprint arXiv:2108.11482*.

[39] Deshpande, Y., S. Sen, A. Montanari, and E. Mossel (2018), Contextual stochastic block models, in *NeurIPS*.

[40] Dobra, A., A. Lenkoski, et al. (2011), Copula Gaussian graphical models and their application to modeling functional disability data, *The Annals of Applied Statistics*, *5*(2A), 969–993.

[41] Doshi-Velez, F., and B. Kim (2017), Towards a rigorous science of interpretable machine learning, *arXiv preprint arXiv:1702.08608*.

[42] Du, S. S., K. Hou, B. Póczos, R. Salakhutdinov, R. Wang, and K. Xu (2019), Graph neural tangent kernel: Fusing graph neural networks with graph kernels, *arXiv preprint arXiv:1905.13192*.

[43] Dziugaite, G. K., K. Hsu, W. Gharbieh, G. Arpino, and D. Roy (2021), On the role of data in pac-bayes, in *International Conference on Artificial Intelligence and Statistics*, pp. 604–612, PMLR.

[44] Elidan, G. (2010), Copula bayesian networks, in *Advances in neural information processing systems*, pp. 559–567.

[45] Eykholt, K., I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, and D. Song (2018), Robust physical-world attacks on deep learning visual classification, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1625–1634.

[46] Fey, M., and J. E. Lenssen (2019), Fast graph representation learning with pytorch geometric, *ArXiv*, *abs/1903.02428*.

[47] Fortunato, S. (2010), Community detection in graphs, *Physics reports*, *486*(3-5), 75–174.

[48] Gao, L., H. Yang, C. Zhou, J. Wu, S. Pan, and Y. Hu (2018), Active discriminative network representation learning, in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pp. 2142–2148.

[49] Garg, V., S. Jegelka, and T. Jaakkola (2020), Generalization and representational limits of graph neural networks, in *International Conference on Machine Learning*, pp. 3419–3430, PMLR.

[50] Germain, P., A. Lacasse, F. Laviolette, M. March, and J.-F. Roy (2015), Risk bounds for the majority vote: From a pac-bayesian analysis to a learning algorithm, *Journal of Machine Learning Research*, *16*(26), 787–860.

[51] Girvan, M., and M. E. Newman (2002), Community structure in social and biological networks, *Proceedings of the national academy of sciences*, *99*(12), 7821–7826.

[52] Gori, M., G. Monfardini, and F. Scarselli (2005), A new model for learning in graph domains, in *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, vol. 2, pp. 729–734, IEEE.

[53] Granovetter, M. S. (1973), The strength of weak ties, *American journal of sociology*, *78*(6), 1360–1380.

[54] Grover, A., and J. Leskovec (2016), node2vec: Scalable feature learning for networks, in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 855–864.

[55] Gu, Q., and J. Han (2012), Towards active learning on graphs: An error bound minimization approach, in *2012 IEEE 12th International Conference on Data Mining*, pp. 882–887, IEEE.

[56] Guedj, B. (2019), A primer on pac-bayesian learning, *arXiv preprint arXiv:1901.05353*.

[57] Hamilton, W., Z. Ying, and J. Leskovec (2017), Inductive representation learning on large graphs, in *Advances in Neural Information Processing Systems*, pp. 1024–1034.

[58] Hoeffding, W. (1994), Probability inequalities for sums of bounded random variables, in *The Collected Works of Wassily Hoeffding*, pp. 409–426, Springer.

[59] Hoff, P. D., A. E. Raftery, and M. S. Handcock (2002), Latent space approaches to social network analysis, *Journal of the american Statistical association*, *97*(460), 1090–1098.

[60] Horel, T., and Y. Singer (2016), Maximization of approximately submodular functions, in *Advances in Neural Information Processing Systems*, pp. 3045–3053.

[61] Hou, Y., J. Zhang, J. Cheng, K. Ma, R. T. Ma, H. Chen, and M.-C. Yang (2019), Measuring and improving the use of graph information in graph neural networks, in *International Conference on Learning Representations*.

[62] Hu, S., Z. Xiong, M. Qu, X. Yuan, M.-A. Côté, Z. Liu, and J. Tang (2020), Graph policy network for transferable active learning on graphs, in *Advances in Neural Information Processing Systems*, vol. 33, pp. 10,174–10,185.

[63] Hu, W., M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, and J. Leskovec (2020), Open graph benchmark: Datasets for machine learning on graphs, *arXiv preprint arXiv:2005.00687*.

[64] Huang, D., X. Zhu, R. Li, and H. Wang (2021), Feature screening for network autoregression model, *Statistica Sinica*, *31*, 1239.

[65] Jain, A., I. Liu, A. Sarda, and P. Molino (2019), Food discovery with uber eats: Using graph learning to power recommendations.

[66] Ji, M., and J. Han (2012), A variance minimization criterion to active learning on graphs, in *Artificial Intelligence and Statistics*, pp. 556–564, PMLR.

[67] Jia, J., and A. R. Benson (2020), Residual correlation in graph neural network regression, in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 588–598.

[68] Jin, G., Q. Wang, C. Zhu, Y. Feng, J. Huang, and J. Zhou (2020), Addressing crime situation forecasting task with temporal graph convolutional neural network approach, in *2020 12th International Conference on Measuring Technology and Mechatronics Automation (ICMTMA)*, pp. 474–478, IEEE.

[69] Jin, W., R. Barzilay, and T. Jaakkola (2018), Junction tree variational autoencoder for molecular graph generation, in *International Conference on Machine Learning*, pp. 2323–2332, PMLR.

[70] Jin, W., Y. Li, H. Xu, Y. Wang, and J. Tang (2020), Adversarial attacks and defenses on graphs: A review and empirical study, *arXiv preprint arXiv:2003.00653*.

[71] Joe, H. (2014), *Dependence Modeling with Copulas*, Chapman & Hall / CRC Press, Boca Raton, FL.

[72] Jordan, M. I., et al. (2004), Graphical models, *Statistical science*, *19*(1), 140–155.

[73] Kaufman, L., and P. J. Rousseeuw (2009), *Finding groups in data: an introduction to cluster analysis*, vol. 344, John Wiley & Sons.

[74] Kazianka, H., and J. Pilz (2010), Copula-based geostatistical modeling of continuous and discrete data including covariates, *Stochastic environmental research and risk assessment*, *24*(5), 661–673.

[75] Khajehnejad, A., M. Khajehnejad, M. Babaei, K. P. Gummadi, A. Weller, and B. Mirzasoleiman (2021), Crosswalk: Fairness-enhanced node representation learning.

[76] Kingma, D. P., and J. Ba (2015), Adam: A method for stochastic optimization, in *International Conference on Learning Representations*.

[77] Kipf, T. N., and M. Welling (2017), Semi-supervised classification with graph convolutional networks, in *International Conference on Learning Representations (ICLR)*.

[78] Kleinberg, J., S. Mullainathan, and M. Raghavan (2016), Inherent trade-offs in the fair determination of risk scores, *arXiv preprint arXiv:1609.05807*.

[79] Klicpera, J., A. Bojchevski, and S. Günnemann (2019), Predict then propagate: Graph neural networks meet personalized pagerank, in *International Conference on Learning Representations (ICLR)*.

[80] Laclau, C., I. Redko, M. Choudhary, and C. Largeron (2020), All of the fairness for edge prediction with optimal transport, *CoRR*, *abs/2010.16326*.

[81] Langford, J., and J. Shawe-Taylor (2003), Pac-bayes & margins, *Advances in neural information processing systems*, pp. 439–446.

[82] Lauritzen, S. L. (1996), *Graphical models*, vol. 17, Clarendon Press.

[83] LeCun, Y., Y. Bengio, and G. Hinton (2015), Deep learning, *nature*, *521*(7553), 436–444.

[84] Lee, L.-f., X. Liu, and X. Lin (2010), Specification and estimation of social interaction models with network structures, *The Econometrics Journal*, *13*(2), 145–176.

[85] Li, C., J. Ma, X. Guo, and Q. Mei (2017), Deepcas: An end-to-end predictor of information cascades, in *Proceedings of the 26th international conference on World Wide Web*, pp. 577–586.

[86] Li, Q., Z. Han, and X.-M. Wu (2018), Deeper insights into graph convolutional networks for semi-supervised learning, in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32.

[87] Li, T., E. Levina, J. Zhu, et al. (2019), Prediction models for network-linked data, *The Annals of Applied Statistics*, *13*(1), 132–164.

[88] Li, Y., J. Yin, and L. Chen (2021), Seal: Semisupervised adversarial active learning on attributed graphs, *IEEE Transactions on Neural Networks and Learning Systems*, *32*, 3136–3147.

[89] Liao, R., R. Urtasun, and R. Zemel (2021), A pac-bayesian approach to generalization bounds for graph neural networks, in *ICLR*.

[90] Lim, D., F. Hohne, X. Li, S. L. Huang, V. Gupta, O. Bhalerao, and S. N. Lim (2021), Large scale learning on non-homophilous graphs: New benchmarks and strong simple methods, *Advances in Neural Information Processing Systems*, *34*, 20,887–20,902.

[91] Liu, H., F. Han, M. Yuan, J. Lafferty, L. Wasserman, et al. (2012), High-dimensional semiparametric Gaussian copula graphical models, *The Annals of Statistics*, *40*(4), 2293–2326.

[92] Liu, J., Y. Wang, B. Hooi, R. Yang, and X. Xiao (2020), Active learning for node classification: The additional learning ability from unlabelled nodes, *ArXiv*, *abs/2012.07065*.

[93] Lou, T., and J. Tang (2013), Mining structural hole spanners through information diffusion in social networks, in *Proceedings of the 22nd international conference on World Wide Web*, pp. 825–836.

[94] Lovász, L., et al. (1993), Random walks on graphs: A survey, *Combinatorics, Paul erdos is eighty*, *2*(1), 1–46.

[95] Ma, C., Y. Li, F. Yang, Z. Zhang, Y. Zhuang, H. Jia, and X. Xie (2019), Deep association: End-to-end graph-based learning for multiple object tracking with conv-graph neural network, in *Proceedings of the 2019 on International Conference on Multimedia Retrieval*, pp. 253–261.

[96] Ma, J., W. Tang, J. Zhu, and Q. Mei (2019), A flexible generative framework for graph-based semi-supervised learning, *arXiv preprint arXiv:1905.10769*.

[97] Ma, J., S. Ding, and Q. Mei (2020), Towards more practical adversarial attacks on graph neural networks, *Advances in neural information processing systems*, *33*, 4756–4766.

[98] Ma, J., B. Chang, X. Zhang, and Q. Mei (2021), Copulagnn: Towards integrating representational and correlational roles of graphs in graph neural networks, in *International Conference on Learning Representations*.

[99] Ma, J., J. Deng, and Q. Mei (2021), Subgroup generalization and fairness of graph neural networks, in *Advances in Neural Information Processing Systems (NeurIPS) 34*.

[100] Ma, J., Z. Ma, J. Chai, and Q. Mei (2022), Partition-based active learning for graph neural networks, *arXiv preprint arXiv:2201.09391*.

[101] Ma, Y., X. Liu, N. Shah, and J. Tang (2021), Is homophily a necessity for graph neural networks?, *arXiv preprint arXiv:2106.06134*.

[102] Malikov, E., and Y. Sun (2017), Semiparametric estimation and testing of smooth coefficient spatial autoregressive models, *Journal of Econometrics*, *199*(1), 12–34.

[103] McAllester, D. (2003), Simplified pac-bayesian margin bounds, in *Learning theory and Kernel machines*, pp. 203–215, Springer.

[104] McPherson, M., L. Smith-Lovin, and J. M. Cook (2001), Birds of a feather: Homophily in social networks, *Annual review of sociology*, *27*(1), 415–444.

[105] Mei, Q., D. Zhang, and C. Zhai (2008), A general optimization framework for smoothing language models on graph structures, in *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 611–618, ACM.

[106] Monti, F., D. Boscaini, J. Masci, E. Rodola, J. Svoboda, and M. M. Bronstein (2017), Geometric deep learning on graphs and manifolds using mixture model cnns, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5115–5124.

[107] Newman, M. (2018), *Networks*, Oxford university press.

[108] Newman, M. E. J. (2004), Fast algorithm for detecting community structure in networks, *Physical Review E*, *69*(6).

[109] Neyshabur, B., S. Bhojanapalli, and N. Srebro (2018), A pac-bayesian approach to spectrally-normalized margin bounds for neural networks.

[110] NT, H., and T. Maehara (2019), Revisiting graph neural networks: All we have is low-pass filters, *arXiv preprint arXiv:1905.09550*.

[111] Ozkan, T. (2019), Criminology in the age of data explosion: New directions, *The social science journal*, *56*(2), 208–219.

[112] Park, H.-S., and C.-H. Jun (2009), A simple and fast algorithm for k-medoids clustering, *Expert systems with applications*, *36*(2), 3336–3341.

[113] Pearl, J. (2009), *Causality*, Cambridge university press.

[114] Peng, H., J. Li, Z. Wang, R. Yang, M. Liu, M. Zhang, P. S. Yu, and L. He (2020), Lifelong property price prediction: A case study for the toronto real estate market, *arXiv preprint arXiv:2008.05880*.

[115] Perozzi, B., R. Al-Rfou, and S. Skiena (2014), Deepwalk: Online learning of social representations, in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 701–710.

[116] Pukelsheim, F. (2006), *Optimal Design of Experiments*, Society for Industrial and Applied Mathematics (SIAM).

[117] Qu, M., Y. Bengio, and J. Tang (2019), Gmnn: Graph markov neural networks, *arXiv preprint arXiv:1905.06214*.

[118] Rahman, T., B. Surma, M. Backes, and Y. Zhang (2019), Fairwalk: Towards fair graph embedding, in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pp. 3289–3295, International Joint Conferences on Artificial Intelligence Organization, doi: 10.24963/ijcai.2019/456.

[119] Regol, F., S. Pal, Y. Zhang, and M. Coates (2020), Active learning on attributed graphs via graph cognizant logistic regression and preemptive query generation, in *International Conference on Machine Learning*, pp. 8041–8050, PMLR.

[120] Ren, P., Y. Xiao, X. Chang, P.-Y. Huang, Z. Li, X. Chen, and X. Wang (2020), A survey of deep active learning, *arXiv preprint arXiv:2009.00236*.

[121] Rivasplata, O. (2012), Subgaussian random variables: An expository note, *Internet publication, PDF*.

[122] Rozemberczki, B., C. Allen, and R. Sarkar (2021), Multi-scale attributed node embedding, *Journal of Complex Networks*, *9*(2), cnab014.

[123] Sato, R. (2020), A survey on the expressive power of graph neural networks, *arXiv preprint arXiv:2003.04078*.

[124] Scarselli, F., M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini (2008), The graph neural network model, *IEEE transactions on neural networks*, *20*(1), 61–80.

[125] Scarselli, F., A. C. Tsoi, and M. Hagenbuchner (2018), The vapnik–chervonenkis dimension of graph and recursive neural networks, *Neural Networks*, *108*, 248–259.

[126] Sen, P., G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad (2008), Collective classification in network data, *AI magazine*, *29*(3), 93–93.

[127] Sener, O., and S. Savarese (2018), Active learning for convolutional neural networks: A core-set approach, in *International Conference on Learning Representations (ICLR)*.

[128] Settles, B. (2012), Active learning, *Synthesis lectures on artificial intelligence and machine learning*, *6*(1), 1–114.

[129] Settles, B., and M. Craven (2008), An analysis of active learning strategies for sequence labeling tasks.

[130] Shchur, O., M. Mumme, A. Bojchevski, and S. Günnemann (2018), Pitfalls of graph neural network evaluation, *arXiv preprint arXiv:1811.05868*.

[131] Shi, C., M. Xu, Z. Zhu, W. Zhang, M. Zhang, and J. Tang (2020), Graphaf: a flow-based autoregressive model for molecular graph generation, *arXiv preprint arXiv:2001.09382*.

[132] Singh, T., Y. Jain, and V. Kumar (2017), Predicting parole hearing result using machine learning, in *2017 International Conference on Emerging Trends in Computing and Communication Technologies (ICETCCT)*, pp. 1–3, IEEE.

[133] Sklar, A. (1959), Fonctions de répartition à n dimensions et leurs marges, *Publications de l'Institut de Statistique de l'Université de Paris*, *8*, 229–231.

[134] Spinelli, I., S. Scardapane, A. Hussain, and A. Uncini (2021), Biased edge dropout for enhancing fairness in graph representation learning.

[135] Sun, Y., S. Wang, X. Tang, T.-Y. Hsieh, and V. Honavar (2019), Node injection attacks on graphs via reinforcement learning, *arXiv preprint arXiv:1909.06543*.

[136] Szegedy, C., W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus (2014), Intriguing properties of neural networks, in *2nd International Conference on Learning Representations, ICLR 2014*.

[137] Tang, J., J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su (2008), Arnetminer: Extraction and mining of academic social networks, in *KDD'08*, pp. 990–998.

[138] Tang, J., M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei (2015), Line: Large-scale information network embedding, in *Proceedings of the 24th international conference on world wide web*, pp. 1067–1077, International World Wide Web Conferences Steering Committee.

[139] Tropp, J. A. (2015), An introduction to matrix concentration inequalities, *Foundations and Trends in Machine Learning*, *8*(1-2), 1–230.

[140] Ugander, J., L. Backstrom, C. Marlow, and J. Kleinberg (2012), Structural diversity in social contagion, *Proceedings of the National Academy of Sciences*, *109*(16), 5962–5966.

[141] Veličković, P., G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio (2018), Graph attention networks, in *International Conference on Learning Representations*.

[142] Verma, S., and Z.-L. Zhang (2019), Stability and generalization of graph convolutional neural networks, in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1539–1548.

[143] Vishwanathan, S. V. N., N. N. Schraudolph, R. Kondor, and K. M. Borgwardt (2010), Graph kernels, *The Journal of Machine Learning Research*, *11*, 1201–1242.

[144] Wainwright, M. J., and M. I. Jordan (2008), Graphical models, exponential families, and variational inference, *Foundations and Trends® in Machine Learning*, *1*(1-2), 1–305.

[145] Wang, M., et al. (2019), Deep graph library: Towards efficient and scalable deep learning on graphs, *arXiv preprint arXiv:1909.01315*.

[146] Ward, J. H. (1963), Hierarchical grouping to optimize an objective function, *Journal of the American Statistical Association*, *58*, 236–244.

[147] Watts, D. J., and S. H. Strogatz (1998), Collective dynamics of 'small-world'networks, *nature*, *393*(6684), 440–442.

[148] Wu, F., A. Souza, T. Zhang, C. Fifty, T. Yu, and K. Weinberger (2019), Simplifying graph convolutional networks, in *International conference on machine learning*, pp. 6861–6871, PMLR.

[149] Wu, H., C. Wang, Y. Tyshetskiy, A. Docherty, K. Lu, and L. Zhu (2019), Adversarial examples for graph data: Deep insights into attack and defense, in *IJCAI*.

[150] Wu, Y., Y. Xu, A. Singh, Y. Yang, and A. Dubrawski (2019), Active learning for graph neural networks via node feature propagation, *arXiv preprint arXiv:1910.07567*.

[151] Wu, Z., S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip (2020), A comprehensive survey on graph neural networks, *IEEE Transactions on Neural Networks and Learning Systems*.

[152] Xu, K., W. Hu, J. Leskovec, and S. Jegelka (2018), How powerful are graph neural networks?, *arXiv preprint arXiv:1810.00826*.

[153] Xu, K., C. Li, Y. Tian, T. Sonobe, K.-i. Kawarabayashi, and S. Jegelka (2018), Representation learning on graphs with jumping knowledge networks, *arXiv preprint arXiv:1806.03536*.

[154] Xu, K., H. Chen, S. Liu, P.-Y. Chen, T.-W. Weng, M. Hong, and X. Lin (2019), Topology attack and defense for graph neural networks: An optimization perspective, *arXiv preprint arXiv:1906.04214*.

[155] Yang, Z., W. W. Cohen, and R. Salakhutdinov (2016), Revisiting semi-supervised learning with graph embeddings, *arXiv preprint arXiv:1603.08861*.

[156] Ye, X., L. Duan, and Q. Peng (2021), Spatiotemporal prediction of theft risk with deep inception-residual networks, *Smart Cities*, *4*(1), 204–216.

[157] Ying, R., R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec (2018), Graph convolutional neural networks for web-scale recommender systems, in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 974–983.

[158] Yu, B., H. Yin, and Z. Zhu (2018), Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting, in *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pp. 3634–3640.

[159] Zeng, Z., R. Islam, K. N. Keya, J. R. Foulds, Y. Song, and S. Pan (2021), Fair representation learning for heterogeneous information networks, *CoRR*, *abs/2104.08769*.

[160] Zhang, W., Y. Shen, Y. Li, L. Chen, Z. Yang, and B. Cui (2021), *ALG: Fast and Accurate Active Learning Framework for Graph Convolutional Networks*, p. 2366–2374, Association for Computing Machinery, New York, NY, USA.

[161] Zhang, W., Z. Yang, Y. Wang, Y. Shen, Y. Li, L. Wang, and B. Cui (2021), Grain: Improving data efficiency of graph neural networks via diversified influence maximization, *Proc. VLDB Endow.*, *14*(11), 2473–2482, doi: 10.14778/3476249.3476295.

[162] Zheng, L., Z. Zuo, W. Wang, C. Dong, M. Momma, and Y. Sun (2021), Heterogeneous graph neural networks with neighbor-sim attention mechanism for substitute product recommendation.

[163] Zhou, D., O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf (2004), Learning with local and global consistency, in *Advances in neural information processing systems*, pp. 321–328.

[164] Zhou, W., V. Veitch, M. Austern, R. P. Adams, and P. Orbanz (2018), Non-vacuous generalization bounds at the imagenet scale: a pac-bayesian compression approach, in *International Conference on Learning Representations*.

[165] Zhu, J., Y. Yan, L. Zhao, M. Heimann, L. Akoglu, and D. Koutra (2020), Beyond homophily in graph neural networks: Current limitations and effective designs, *Advances in Neural Information Processing Systems*, *33*.

[166] Zhu, X., Z. Ghahramani, and J. D. Lafferty (2003), Semi-supervised learning using gaussian fields and harmonic functions, in *Proceedings of the 20th International conference on Machine learning (ICML-03)*, pp. 912–919.

[167] Zhu, X., R. Pan, G. Li, Y. Liu, and H. Wang (2017), Network vector autoregression, *The Annals of Statistics*, *45*(3), 1096 – 1123, doi:10.1214/16-AOS1476.

[168] Zügner, D., and S. Günnemann (2019), Adversarial attacks on graph neural networks via meta learning, in *International Conference on Learning Representations (ICLR)*.

[169] Zügner, D., A. Akbarnejad, and S. Günnemann (2018), Adversarial attacks on neural networks for graph data, in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2847–2856.