Cross-Space Active Learning on Graph Convolutional Networks

Yufei Tao [1] **Hao Wu** [1] **Shiyuan Deng** [1]

Abstract

This paper formalizes *cross-space* active learning on a graph convolutional network (GCN). The objective is to attain the most accurate hypothesis available in any of the instance spaces generated by the GCN. Subject to the objective, the challenge is to minimize the *label cost*, measured in the number of vertices whose labels are requested. Our study covers both *budget algorithms* which terminate after a designated number of label requests, and *verifiable algorithms* which terminate only after having found an accurate hypothesis. A new separation in label complexity between the two algorithm types is established. The separation is unique to GCNs.

1. Introduction

In traditional active learning, the samples are drawn from $\mathcal{X} \times \mathcal{Y}$ where $\mathcal{X}$ is some instance space and $\mathcal{Y} := \{-1, 1\}$ is the label space. The goal is to find a hypothesis with a small classification error by requesting the labels for only a subset of the samples. This paper studies a new setup — we call *cross-space* active learning — where each sample is represented in multiple instance spaces but still carries a single label. The objective is to identify the best instance space promising the lowest classification error.

Cross-space active learning is best formulated on a *finite* set V of objects. Let $n := |V|$. Each object $e \in V$ has a label $Y_e \in \mathcal{Y}$ that is hidden initially. $\mathcal{X}_1, \mathcal{X}_2, ..., \mathcal{X}_T$ are $T \geq 2$ instance spaces such that e has a representation $X_e^{(t)} \in \mathcal{X}_t$ for each $t \in [T]$ (the notation $[x]$ represents the set $\{1, 2, ..., x\}$). $\mathcal{X}_t$ is associated with a class $\mathbb{H}_t$ of hypotheses each being a function $h : \mathcal{X}_t \to \mathcal{Y}$. The error of an $h \in \mathbb{H}_t$ is $\text{er}(t, h) := \frac{1}{n}|\{e \in V \mid h(X_e^{(t)}) \neq Y_e\}|$. An algorithm $\mathcal{A}$ can ask an oracle to reveal the labels for a subset $Q \subseteq V$. Once revealed, an object's label is visible to all the instance spaces. $\mathcal{A}$ needs to output a pair (t, h) and use h to classify the objects of $V \setminus Q$ in $\mathcal{X}_t$. The *cost* of $\mathcal{A}$ is $|Q|$. We are interested in algorithms with cost *far less* than n, in which case the number of misclassified objects is essentially $n \cdot \text{er}(t, h)$.

Nowadays, the above setup arises frequently due to the popularity of convolutional networks. *Convolution*, in general, refers to an iterative process that synthesizes new features based on some notion of adjacency. Applying the process to instance space $\mathcal{X}_t$ produces instance space $\mathcal{X}_{t+1}$ $(t \geq 1)$. The iteration can go on $T - 1$ times to generate $\mathcal{X}_2, ..., \mathcal{X}_T$ for some $T \geq 2$. Hopefully, one of these spaces would be particularly friendly to the underlying learning task, in which case we could achieve a better accuracy with fewer samples compared to working in the original space $\mathcal{X}_1$.

We will devote our attention to *graph convolutional networks* (GCN), which have attracted considerable interest in recent years (a brief survey will appear in Section 4). The input is an undirected graph $G := (V, E)$ and an $n \times d$ real-valued matrix $\mathbf{X}$, where $n := |V|$ and $d \geq 1$ is an integer. Each row of $\mathbf{X}$ represents the raw features of a distinct vertex in V; define $\mathbf{X}^{(1)} := \mathbf{X}$. Convolution computes an $n \times d_{t+1}$ matrix $\mathbf{X}^{(t+1)}$ from $\mathbf{X}^{(t)}$, for each $t \in [T-1]$. Effectively, this casts V into instance spaces $\mathcal{X}_1, ..., \mathcal{X}_T$ where $\mathcal{X}_t \subset \mathbb{R}^{d_t}$ is associated with a hypothesis class $\mathbb{H}_t$ for $t \in [T]$. Every vertex has a label that is hidden initially. An algorithm may request the labels for a small $Q \subseteq V$, after which it needs to predict the labels for $V \setminus Q$ using a hypothesis $h_t \in \mathbb{H}_t$, for some $t \in [T]$.

Define $\nu^* := \min_{t \in [T], h \in \mathbb{H}_t} \text{er}(t, h)$, namely, the minimum error across all the instance spaces. Define t^* as the $t \in [T]$ such that $\nu^* = \min_{h \in \mathbb{H}_t} \text{er}(t, h)$, that is, $\mathcal{X}_{t^*}$ is the best instance space. We aim to understand the label complexity of discovering a pair (t, h) with $\text{er}(t, h) \leq \nu^* + \epsilon$ for an arbitrary $\epsilon \in (0, 0.5)$.

Our study covers two algorithm classes.

- *The budget class*: An algorithm is given a budget L on the maximum number of label requests and must terminate after exhausting the budget.
- *The verifiable class*: An algorithm is given a value of ϵ and terminates only when it can ensure $\text{er}(t, h) \leq$

[1]Department of Computer Science and Engineering, Chinese University of Hong Kong, Hong Kong, China. Correspondence to: Yufei Tao <taoyf@cse.cuhk.edu.hk>.

Proceedings of the 39th International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022. Copyright 2022 by the author(s).

$\nu^* + \epsilon$ with high confidence.

The notions of budget and verifiable algorithms are well-known in traditional active learning (where there is only one instance space); see (Balcan et al., 2010; Hanneke, 2012) and the references therein. Intuitively, budget algorithms are less powerful because they need not be aware of the precision (namely, ϵ) that they can guarantee. Although such algorithms can *find* a good hypothesis given a large enough budget L, they may not be able to *verify* the precision of the hypothesis. Verifiable algorithms, in contrast, must monitor the precision of the best hypothesis found so far. A verifiable algorithm can usually be turned into a budget algorithm with nearly the same label complexity, but the opposite may not be true. Understanding when the two algorithm classes can be separated — that is, finding a scenario where budget algorithms achieve a provably smaller label complexity — has been an intriguing topic in active learning (Balcan et al., 2010; Hanneke, 2012).

This paper presents the first study of budget and verifiable algorithms on GCNs under the cross-space learning framework. Our most important contribution is a new separation unique to GCNs, which reveals a somewhat surprising fundamental difference of the two algorithm classes. Specifically, when T is $O(\text{polylog}\, n)$ — a condition that typically holds in reality — a budget algorithm can ensure the same asymptotic label complexity (up to a polylog factor) as if we *were* performing active learning directly in $\mathcal{X}_{t^*}$, even though t^* is unknown! The same guarantee, however, can be proved to be impossible for verifiable algorithms.

Our budget and verifiable algorithms are sensitive to the disagreement coefficients[1] of the T instance spaces. Their label complexities never exceed, but can be considerably lower than, $\tilde{O}(\frac{\nu^*+\epsilon}{\epsilon^2} \cdot \max_{t=1}^T \lambda_t)$ where λ_t is the VC dimension of $(\mathcal{X}_t, \mathbb{H}_t)$ and $\tilde{O}(.)$ hides a polylog factor. These results complement the previous work (to be reviewed in Section 4) that focused on developing the underlying convolution process. For example, Kipf and Welling 2017 recommended $t^* = 2$ for their now-famous convolution design, namely, the best instance space is obtained by performing convolution only once. They came to this conclusion after trying $t = 2, 3, 4, ...$, only to realize that $t = 2$ yields the best learning outcome. However, inspecting all those t values requires a non-trivial label cost, the minimization of which is exactly the cross-space learning problem we aim to solve.

Furthermore, our results motivate a theoretical model (to be introduced in Section 7) for studying cross-space learning on GCNs in a *stochastic* manner. At a high level, an oracle starts by choosing an appropriate graph $G := (V, E)$ and an integer $t^* \geq 1$. It then generates an $n \times d$ matrix $\mathbf{X}^{(t^*)}$ and the vertex labels according to a certain distribution. After that, the oracle computes $\mathbf{X}^{(1)}$ from $\mathbf{X}^{(t^*)}$ through a "reverse convolution" process on G, effectively concealing $\mathbf{X}^{(t^*)}$. Finally, it releases $\mathbf{X}^{(1)}$, G, and T to an algorithm for cross-space learning. The model captures a scenario where GCNs promise more accurate classification than learning from $\mathbf{X}^{(1)}$ directly, thereby offering a plausible explanation about the usefulness of GCNs.

[1]Disagree coefficient is a key concept in the theory of active learning, to be reviewed in Section 3.

2. Cross-Space Active Learning on a GCN

This section formalizes the main problem studied in this paper. The GCN variant behind our formulation is *simple graph convolution* (SGC) (Wu et al., 2019). This choice is based on several considerations. First, SGC has been shown to be just as effective as many (more) sophisticated GCN architectures; see, for example, the empirical evaluations in (Wu et al., 2019; Chen et al., 2020; He et al., 2020; Wang et al., 2021; Zhu & Koniusz, 2021). Second, SGC has a clean convolution process especially suitable for theoretical studies. Third, for proving strong negative results, we *should* assume a convolution process that is as primitive as possible because, intuitively, more sophisticated convolution can only increase the problem's hardness. Fourth, extending our algorithms to general GCN architectures is easy, as discussed in Appendix C.

All matrices will be written in bold fonts. Given a matrix $\mathbf{M}$, we use $\mathbf{M}_i$ to denote the i-th row of $\mathbf{M}$ and $\mathbf{M}[i, j]$ to denote the element at the i-th row and j-th column of $\mathbf{M}$.

Problem. Let $G := (V, E)$ be an undirected graph where every vertex has a self-loop, and no parallel edges exist between two vertices. We consider, w.l.o.g., $V = [n]$. Denote by $\mathbf{A}$ the adjacency matrix of G, and by $\mathbf{D} := \text{diag}(\deg(1), ..., \deg(n))$ the degree matrix, where $\deg(i)$ is the degree of vertex $i \in [n]$. Define

$$\mathbf{S} \quad := \quad \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2} \tag{1}$$

where $\mathbf{D}^{-1/2} := \text{diag}(\frac{1}{\sqrt{\deg(1)}}, ..., \frac{1}{\sqrt{\deg(n)}})$.

Let $\mathbf{X}$ be an $n \times d$ matrix where $\mathbf{X}_i$ contains the raw features of vertex $i \in [n]$. With $\mathbf{X}^{(1)} := \mathbf{X}$, the convolution process of SGC generates

$$\mathbf{X}^{(t+1)} \quad := \quad \mathbf{S}\mathbf{X}^{(t)} = \mathbf{S}^t\mathbf{X} \tag{2}$$

for each $t \geq 1$. $\mathbf{X}^{(t+1)}$ is an $n \times d$ matrix with $\mathbf{X}_i^{(t+1)}$ representing the synthesized features of vertex i. The convolution is repeated $T - 1$ times for some $T \geq 2$, which yields $\mathbf{X}^{(2)}, ..., \mathbf{X}^{(T)}$.

The oracle secretly chooses an $n \times 1$ matrix $\mathbf{Y}$, where $\mathbf{Y}_i \in \mathcal{Y} := \{-1, 1\}$ is the *label* of vertex $i \in [n]$. The goal of learning is to classify the label of each vertex with the fewest mistakes.

To perform label classification, we resort to a class $\mathbb{H}$ of hypotheses each being a function $h : \mathbb{R}^d \to \mathcal{Y}$. Conveniently, we regard each $\mathbf{X}_i^{(t)}$ as a point in $\mathbb{R}^d$ such that $h(\mathbf{X}_i^{(t)})$ gives the classification result for vertex i based on its features in $\mathbf{X}^{(t)}$. For any $t \in [T]$ and $h \in \mathbb{H}$, define

$$\mathrm{er}(t, h) := \frac{|\{i \in [n] \mid h(\mathbf{X}_i^{(t)}) \neq \mathbf{Y}_i\}|}{n}. \tag{3}$$

The minimum achievable error is

$$\nu^* := \min_{t \in [T], h \in \mathbb{H}} \mathrm{er}(t, h). \tag{4}$$

Let $\lambda := \mathrm{vc}(\mathbb{R}^d, \mathbb{H})$ where vc represents "VC dimension". We will treat $\mathbf{X}^{(t)}$ as a set of n points in $\mathbb{R}^d$. Clearly, $\mathrm{vc}(\mathbf{X}^{(t)}, \mathbb{H}) \leq \lambda$.

A learning algorithm $\mathcal{A}$ has complete details of $G, \mathbf{X}, T$, and $\mathbb{H}$, but knows nothing about $\mathbf{Y}$. It can designate any vertex $i \in [n]$ and request the oracle to reveal the label $\mathbf{Y}_i$. The *cost* of $\mathcal{A}$ is the total number of requests. In the end, $\mathcal{A}$ must choose a pair $(t, h) \in [T] \times \mathbb{H}$ and, for each vertex i whose label remains hidden, predicts the label as $h(\mathbf{X}_i^{(t)})$. We consider only the situation where the cost is far less than n such that the number of mistakes by $\mathcal{A}$ is essentially $n \cdot \mathrm{er}(t, h)$. The objective of *cross-space active learning* (CSAL) on a GCN is to minimize $\mathrm{er}(t, h)$.

Remark 2.1. Under the terminology in Section 1, the instance space $\mathcal{X}_t$, for each $t \in [T]$, is the set of d-dimensional points in $\mathbf{X}^{(t)}$, and $\mathbb{H}_t = \mathbb{H}$. Using the same dimensionality d and hypothesis class in all spaces allows us to present the core ideas with minimum technical complication. Removing these restrictions is easy, as shown in Appendix C.

Two Algorithm Classes. Besides G, $\mathbf{X}$, T, and $\mathbb{H}$, a *budget CSAL algorithm* $\mathcal{A}$ takes two extra parameters: $\delta \in (0, 1)$ and $L \geq 1$; $\mathcal{A}$ is allowed to request at most L labels.

Definition 2.2. A budget CSAL algorithm $\mathcal{A}$ achieves a *label complexity* $\Lambda(\epsilon, \delta, G, \mathbf{X}, \mathbf{Y}, T, \mathbb{H})$ if, for any $G, \mathbf{X}, \mathbf{Y}, \mathbb{H}, \epsilon \in (0, 0.5), \delta \in (0, 1), T \geq 2$, and $L \geq \Lambda(\epsilon, \delta, G, \mathbf{X}, \mathbf{Y}, T, \mathbb{H})$, the following statement holds with probability at least $1 - \delta$: $\mathcal{A}(L, \delta, G, \mathbf{X}, T, \mathbb{H})$ outputs (t, h) with $\mathrm{er}(t, h) \leq \nu^* + \epsilon$.

Besides G, $\mathbf{X}$, T, and $\mathbb{H}$, a *verifiable CSAL algorithm* $\mathcal{A}$ takes two extra parameters: $\delta \in (0, 1)$ and $\epsilon \in (0, 0.5)$; $\mathcal{A}$ is allowed to request an arbitrary number of labels.

Definition 2.3. A verifiable CSAL algorithm $\mathcal{A}$ achieves a *label complexity* $\Lambda(\epsilon, \delta, G, \mathbf{X}, \mathbf{Y}, T, \mathbb{H})$ if, for any $G, \mathbf{X}, \mathbf{Y}, \mathbb{H}, \epsilon \in (0, 0.5), \delta \in (0, 1)$, and $T \geq 2$, the following statement holds with probability at least $1 - \delta$: $\mathcal{A}(\epsilon, \delta, G, \mathbf{X}, T, \mathbb{H})$ outputs (t, h) with $\mathrm{er}(t, h) \leq \nu^* + \epsilon$ by making at most $\Lambda(\epsilon, \delta, G, \mathbf{X}, \mathbf{Y}, T, \mathbb{H})$ label requests.

A key difference between Definitions 2.2 and 2.3 is what happens when L is far greater than $\Lambda(\epsilon, \delta, G, \mathbf{X}, \mathbf{Y}, T, \mathbb{H})$. A verifiable algorithm must stop with a cost far less than L, while a budget algorithm is not required to do so.

3. Review of Active Learning

This section discusses several concepts and results from active learning that are relevant to our discussion.

Basic Definitions. Let $\mathcal{X}$ be an instance space and $\mathcal{Y} := \{-1, 1\}$ be a label space. Denote by $\mathcal{D}$ a distribution (i.e., probability measure) over $\mathcal{X} \times \mathcal{Y}$. $\mathbb{H}$ is a class of hypotheses, each being a function $h : \mathcal{X} \to \mathcal{Y}$. Let $\lambda := \mathrm{vc}(\mathcal{X}, \mathbb{H})$. For each $h \in \mathbb{H}$, define $\mathrm{er}_{\mathcal{D}}(h) := \Pr_{(x,y) \sim \mathcal{D}}[h(x) \neq y]$ and $\nu := \inf_{h \in \mathbb{H}} \mathrm{er}_{\mathcal{D}}(h)$. We consider that $\mathbb{H}$ has at least one *optimal* hypothesis h^* with $\mathrm{er}_{\mathcal{D}}(h^*) = \nu$.

An active learning algorithm $\mathcal{A}$ draws a sequence of independent samples $(x_i, y_i) \sim \mathcal{D}$, where $i := 1, 2, ..., m$ for some finite m. For each $i \in [m]$, the instance x_i is disclosed to $\mathcal{A}$, but the label y_i is hidden; $\mathcal{A}$ can ask an oracle to reveal y_i if necessary. In the end, $\mathcal{A}$ must return a hypothesis $h \in \mathbb{H}$. Its *cost* is the total number of label requests made.

Disagreement Coefficients. Given any $S \subseteq \mathcal{X}$, define $\Pr_{\mathcal{D}}[S] := \Pr_{(x,y) \sim \mathcal{D}}[x \in S]$. For any $\mathcal{H} \subseteq \mathbb{H}$, the *disagreement region* of $\mathcal{H}$ is

$$\mathrm{DIS}(\mathcal{H}) \quad := \quad \{x \in \mathcal{X} \mid \exists h, g \in \mathcal{H} \text{ s.t. } h(x) \neq g(x)\}.$$

Given any $h \in \mathbb{H}$ and $r > 0$, define

$$B_{\mathcal{D}}(h, r) \quad := \quad \{g \in \mathbb{H} \mid \Pr_{\mathcal{D}}[\mathrm{DIS}(\{g, h\})] \leq r\}.$$

For any $h \in \mathbb{H}$, its *disagreement coefficient* under $\mathcal{D}$ is a function $\theta^h_{\mathbb{H},\mathcal{D}} : (0, 1] \to [1, \infty)$ defined as

$$\theta^h_{\mathbb{H},\mathcal{D}}(r) := \max\left\{1, \sup_{r' > r} \frac{\Pr_{\mathcal{D}}[\mathrm{DIS}(B_{\mathcal{D}}(h, r'))]}{r'}\right\}. \tag{5}$$

The *disagreement coefficient* of $\mathbb{H}$ under $\mathcal{D}$ is the function

$$\theta_{\mathbb{H},\mathcal{D}}(r) \quad := \quad \theta^{h^*}_{\mathbb{H},\mathcal{D}}(r) \tag{6}$$

for an arbitrary optimal h^*. All the results stated in this paper hold regardless of the choice of h^*.

Proposition 3.1 (Chapter 7 of (Hanneke, 2014))**.** *Disagreement coefficients have the following properties:*

1. $\theta_{\mathbb{H},\mathcal{D}}(r) \leq \min\{1/r, |\mathbb{H}|\}$.
2. $\theta_{\mathbb{H},\mathcal{D}}(r)$ *is non-increasing in* r.
3. $\theta_{\mathbb{H},\mathcal{D}}(r) \leq c \cdot \theta_{\mathbb{H},\mathcal{D}}(cr)$ *for any constant* $c > 1$.
4. $\theta_{\mathbb{H} \cup \mathbb{G},\mathcal{D}}(r) \leq \theta_{\mathbb{H},\mathcal{D}}(r) + \theta_{\mathbb{G},\mathcal{D}}(r) + 2$ *for any hypothesis classes* $\mathbb{H}$ *and* $\mathbb{G}$.

Label Complexities. A *verifiable active learning algorithm* $\mathcal{A}$ takes three parameters: $\epsilon \in (0, 0.5)$, $\delta \in (0, 1)$, and $\mathbb{H}$. It achieves a *label complexity* $\Lambda(\epsilon, \delta, \mathcal{D}, \mathbb{H})$ if, for any $\mathcal{D}$, $\epsilon \in (0, 0.5)$, and $\delta \in (0, 1)$, the following holds with probability at least $1 - \delta$: $\mathcal{A}(\epsilon, \delta, \mathbb{H})$ returns a hypothesis $h \in \mathbb{H}$ with $\mathrm{er}_{\mathcal{D}}(h) \le \nu + \epsilon$ by making at most $\Lambda(\epsilon, \delta, \mathcal{D}, \mathbb{H})$ label requests.

Lemma 3.2. *There is a verifiable active learning algorithm achieving a verifiable label complexity* $\Lambda(\epsilon, \delta, \mathcal{D}, \mathbb{H})$ *at most*

$$\theta_{\mathbb{H},\mathcal{D}}(\nu + \epsilon) \cdot (1 + \nu/\epsilon)^2 \cdot \lambda \cdot \mathrm{polylog}(1/(\delta\epsilon)). \tag{7}$$

Proof. See Chapters 5.2 and 8.7 of (Hanneke, 2014). □

The algorithm in the lemma will be referred to as $\mathrm{ActLearn}_{\mathrm{V}}(\epsilon, \delta, \mathbb{H})$.

A *budget active learning algorithm* $\mathcal{A}$ takes three parameters: $L \ge 1$, $\delta \in (0, 1)$, and $\mathbb{H}$; the algorithm is allowed to request at most L labels. $\mathcal{A}$ achieves a *label complexity* $\Lambda(\epsilon, \delta, \mathcal{D}, \mathbb{H})$ if, for any $\mathcal{D}$, $\epsilon \in (0, 0.5), \delta \in (0, 1)$, and $L \ge \Lambda(\epsilon, \delta, \mathcal{D}, \mathbb{H})$, the following holds with probability at least $1 - \delta$: $\mathcal{A}(L, \delta, \mathbb{H})$ returns a hypothesis $h \in \mathbb{H}$ with $\mathrm{er}_{\mathcal{D}}(h) \le \nu + \epsilon$.

Lemma 3.3. *There is a budget active learning algorithm achieving a budget label complexity* $\Lambda(\epsilon, \delta, \mathcal{D}, \mathbb{H})$ *at most the value given in* (7).

Proof. See Chapter 5.2 of (Hanneke, 2014). □

The algorithm in the lemma will be referred to as $\mathrm{ActLearn}_{\mathrm{B}}(L, \delta, \mathbb{H})$.

Remark 3.4. Our *verifiable* definition corresponds to the notion of *self-verifying* in (Hanneke, 2014). Balcan et al. (2010) defined *verifiable* in an alternative manner that is equivalent to self-verifying (and hence also to our version), up to a $\mathrm{polylog}(1/(\delta\epsilon))$ factor in label complexity, as noted in (Balcan et al., 2010; Hanneke, 2014).

4. Related Work

Active Learning. In *passive* learning, an algorithm's cost is measured by the number of instance-label pairs drawn. In practical applications, labeling an instance can be considerably more expensive than drawing an instance. The phenomenon motivated active learning where, as discussed in Section 3, the primary goal is label minimization. In their seminal work, Balcan, Beygelzimer, and Langford (2009) developed the A^2 algorithm which paved the foundation of modern active learning algorithms (see (Dasgupta, 2005; Hanneke, 2007; Dasgupta et al., 2007; Castro & Nowak, 2008; Balcan et al., 2007; Koltchinskii, 2010; Balcan et al., 2010; Hanneke, 2012) and the references therein). Many of those algorithms achieve near-optimal label complexities with respect to known lower bounds (Dasgupta, 2005; Kaariainen, 2006; Castro & Nowak, 2008; Beygelzimer et al., 2009; Hanneke, 2011). Disagreement coefficient, which was introduced by Hanneke (2007), has played a major role in characterizing the label complexity of active learning. The reader may refer to (Hanneke, 2014) for a comprehensive survey on this area.

Balcan et al. (2010) and Hanneke (2012) showed that budget algorithms achieve an asymptotically lower label complexity than verifiable algorithms when ν is treated as a constant, and ϵ is treated as an asymptotic parameter approaching 0 (see Section 3 for the definitions of ν and ϵ).

Graph Convolutional Networks. Recent years have witnessed extensive research on GCNs. We refer the reader to the surveys of (Liu & Zhou, 2020; Wu et al., 2021) for a systematic introduction to this exciting topic. The following discussion will focus on several categories in the literature most relevant to this paper. For each category, we will provide citations to some representative work as entry points for further readings.

The "main-stream" research on GCNs aims to design a convolution process to maximize the usefulness of vertices' features in the generated instance spaces. Current solutions can be further divided into *spectral-based* (Bruna et al., 2014; Defferrard et al., 2016; Kipf & Welling, 2017; Wu et al., 2019; Zhu & Koniusz, 2021) and *spatial-based* (Atwood & Towsley, 2016; Gilmer et al.; Hamilton et al., 2017; Li et al., 2018b; Micheli, 2009). Our algorithms, which are developed under a generic cross-space setting (Appendix C), can be applied to find the best instance space regardless of whether the GCN is spectral- or spatial-based.

Most existing GCNs suffer from the *over-smoothing* issue, namely, for each vertex, its features in the generated instance spaces converge quickly as T (i.e., the number of instance spaces) increases. Considerable effort has been dedicated to understanding and remedying this issue (Li et al., 2018a; Klicpera et al., 2019; Oono & Suzuki, 2020; Zhao & Akoglu, 2020; Xu et al., 2018; Rong et al., 2020; Liu et al., 2020). So far, the largest value of T even in the "deepest" GCN can safely be regarded to be $O(\mathrm{polylog}\, n)$ (recall that n is the number of vertices). This is why $T = O(\mathrm{polylog}\, n)$ presents an important scenario in this field.

The statistic model to be developed in Section 7 belongs to the *explainablility* category, where the objective is to discover qualitative justifications on the learning outcome of a GCN; see (Luo et al., 2020; Lin et al., 2021; Schlichtkrull et al., 2021; Ying et al., 2019; Yuan et al., 2021) and the references therein. Finally, our work is also remotely relevant to the category studying the expressive power of GCNs (Morris et al., 2019; Maron et al., 2019; Srinivasan & Ribeiro, 2020; Xu et al., 2019).

Multi-View Learning. Cross-space learning proposed in this paper should not be confused with "multi-view learning". The latter also concerns multiple instance spaces, but the goal there is to produce a composite hypothesis that utilizes hypotheses of *different* spaces to collaboratively accomplish a learning task. In contrast, cross-space learning's aim is to discover the *instance space* most effective for learning, which in essence is to discover the best feature representations for the vertices (i.e., embedding the graph appropriately), as is a primary motivation of GCNs. The interested reader may refer to (Balcan & Blum, 2010; Lan & Huan, 2015; Muslea et al., 2006; Wang & Zhou, 2008; 2010) for representative work on (active) multi-view learning.

5. Algorithms

This section will describe the proposed CSAL algorithms.

Notations. For $t \in [T]$, set $\nu_t := \min_{h \in \mathbb{H}} \mathrm{er}(t,h)$ — the function $\mathrm{er}(t,h)$ was defined in (3) — and let h_t^* be a hypothesis in $\mathbb{H}$ satisfying $\mathrm{er}(t, h_t^*) = \nu_t$. In other words, ν_t is the minimum error attainable in instance space $\mathcal{X}_t$, while h_t^* is a hypothesis giving that error. Define t^* as an integer in $[T]$ such that $\nu_{t^*} = \nu^*$ where ν^* is the cross-space minimum error, as defined in (4).

For each $t \in [T]$, we impose a distribution $\mathcal{U}_t$ over $\mathbf{X}^{(t)} \times \mathcal{Y}$: a sample $(x,y) \sim \mathcal{U}_t$ satisfies $\Pr[(x,y) = (\mathbf{X}_i^{(t)}, \mathbf{Y}_i)] = 1/n$ for each $i \in [n]$. Define

$$\theta_t(r) := \theta_{\mathbb{H},\mathcal{U}_t}(r) \tag{8}$$

where $\theta_{\mathbb{H},\mathcal{U}_t}(r)$ is the disagreement coefficient of $\mathbb{H}$ under $\mathcal{U}_t$ (see (6) for the definition of disagreement coefficient). Impose also a distribution $\mathcal{U}$ over $V \times \mathcal{Y}$: a sample $(x,y) \sim \mathcal{U}$ satisfies $\Pr[(x,y) = (i, \mathbf{Y}_i)] = 1/n$ for each $i \in [n]$.

5.1. A Verifiable Algorithm

To motivate our approach, let us first understand the deficiency of a "standard" method for tackling CSAL. Suppose that, for each $t \in [T]$, we can (i) find a hypothesis $h_t \in \mathbb{H}$ with $\mathrm{er}(t,h_t) \le \nu_t + \epsilon/2$, and (ii) obtain an estimate $\hat{\mathrm{er}}(t,h_t)$ of $\mathrm{er}(t,h_t)$ within absolute error $\epsilon/2$. The (t,h_t) having the lowest $\hat{\mathrm{er}}(t,h_t)$ must enjoy the guarantee $\mathrm{er}(t,h_t) \le \nu^* + \epsilon$ and, therefore, can be returned as a legal output. By textbook results from the PAC theory, we can obtain the aforementioned (t,h_t) and $\hat{\mathrm{er}}(t,h_t)$ for each t with high confidence by probing $\tilde{O}(\frac{\lambda}{\epsilon^2} \cdot (\nu_t + \epsilon))$ labels. The overall cost is thus $\tilde{O}(\frac{\lambda}{\epsilon^2} \sum_{t=1}^T (\nu_t + \epsilon))$.

We will show how to achieve a label complexity that never exceeds, but can be much smaller than, $\tilde{O}(\frac{\lambda}{\epsilon^2}(\nu^* + \epsilon))$. Note that $\frac{\lambda}{\epsilon^2}(\nu^* + \epsilon)$ can be far less than $\frac{\lambda}{\epsilon^2}\sum_{t=1}^T(\nu_t + \epsilon)$ even if T is as small as 2 (e.g., consider $\nu^* = \nu_1 \ll \nu_2$). The vast difference between the two complexities suggests that — for designing verifiable algorithms — the instance spaces should not be explored independently, because doing so will have the cost dominated by the *hardest* instance space, which can be much more expensive to learn than $\mathcal{X}_{t^*}$.

Instead of learning the instance spaces individually, we combine them into one. For this purpose, it is crucial to think of every $(t,h) \in [T] \times \mathbb{H}$ as a function from V to $\mathcal{Y}$. For each (t,h), we define an induced function

$$g_{t,h}(i) := h(\mathbf{X}_i^{(t)})$$

and accordingly

$$\mathrm{er}(g_{t,h}) \quad := \quad \mathrm{er}(t,h) = \frac{|\{i \in [n] \mid g_{t,h}(i) \neq \mathbf{Y}_i\}|}{n}.$$

Naturally, any $\mathcal{H} \subseteq [T] \times \mathbb{H}$ induces a hypothesis class

$$\mathbb{G}_{\mathcal{H}} \quad := \quad \{g_{t,h} \mid (t,h) \in \mathcal{H}\}. \tag{9}$$

Given ϵ, δ, and $\mathbb{H}$, our verifiable algorithm is:

run $\mathrm{ActLearn_V}(\epsilon, \delta, \mathbb{G}_{[T]\times\mathbb{H}})$ by setting $\mathcal{D} = \mathcal{U}$.

Recall (from Section 3) that $\mathrm{ActLearn_V}$ samples from an agnostic distribution $\mathcal{D}$. We *manually* fix the distribution to $\mathcal{U}$. Every time $\mathrm{ActLearn_V}$ draws a sample $(x,y) \sim \mathcal{D}$, we supply a pair $(x,y) \sim \mathcal{U}$ with y hidden. If $\mathrm{ActLearn_V}$ requests y, the request is relayed to the oracle. Given the output $g_{t,h} \in \mathbb{G}_{[T]\times\mathbb{H}}$ from $\mathrm{ActLearn_V}$, we return (t,h).

For analysis, define

$$\theta_{\mathrm{all}}(r) := \theta_{\mathbb{G}_{[T]\times\mathbb{H}},\mathcal{U}}(r)$$

where $\theta_{\mathbb{G}_{[T]\times\mathbb{H}},\mathcal{U}}(r)$ is the disagreement coefficient of $\mathbb{G}_{[T]\times\mathbb{H}}$ under $\mathcal{U}$.

Lemma 5.1. *For any $r > 0$, $\theta_{\mathrm{all}}(r) < 3\sum_{t=1}^T \theta_t(r)$.*

Proof. Define $\mathbb{G}_t = \mathbb{G}_{\{t\}\times\mathbb{H}}$. By (9), $\mathbb{G}_{[T]\times\mathbb{H}} = \bigcup_{t\in[T]} \mathbb{G}_t$. For any $t \in [T]$, each $(t,h) \in t \times \mathbb{H}$ essentially maps V to $\mathcal{Y}$ in the same way as $g_{t,h}$. The reader can verify $\theta_{\mathbb{G}_t,\mathcal{U}}(r) = \theta_t(r)$ from the disagreement coefficient definition in Section 3. The lemma now follows from Property 4 in Proposition 3.1 and the fact that $\theta_t(r) \ge 1$. □

Theorem 5.2. *Our algorithm has a label complexity*

$$\lambda \cdot \theta_{\mathrm{all}}(\nu^* + \epsilon) \cdot (1 + \nu^*/\epsilon)^2 \cdot \mathrm{polylog}(nT/(\delta\epsilon)). \tag{10}$$

Proof. Let $g^* := \arg\min_{g \in \mathbb{G}_{[T]\times\mathbb{H}}} \mathrm{er}(g)$ be an optimal hypothesis in $\mathbb{G}_{[T]\times\mathbb{H}}$; note that $\mathrm{er}(g^*) = \nu^*$. Set $\lambda' =$

$\mathrm{vc}(V, \mathbb{G}_{[T]\times\mathbb{H}})$. By Lemma 3.2, with probability at least $1-\delta$, the output $g_{t,h}$ of $\mathrm{ActLearn_V}(\epsilon, \delta, \mathbb{G}_{[T]\times\mathbb{H}})$ satisfies $\mathrm{er}(g_{t,h}) \le \mathrm{er}(g^*) + \epsilon = \nu^* + \epsilon$ and $\mathrm{ActLearn_V}(\epsilon, \delta, \mathbb{G}_{[T]\times\mathbb{H}})$ has a cost at most $\lambda' \cdot \theta_{\mathrm{all}}(\nu^*+\epsilon)\cdot(1+\nu^*/\epsilon)^2 \cdot \mathrm{polylog}(1/(\delta\epsilon))$. Next, we will show $\lambda' < \log_2 T + \lambda \log_2(3n)$, which will complete the proof.

We will bound the number of ways to classify V using the hypotheses in $\mathbb{G}_{[T]\times\mathbb{H}} = \bigcup_{t\in[T]} \mathbb{G}_t$, where $\mathbb{G}_t = \mathbb{G}_{\{t\}\times\mathbb{H}}$. For each $t \in [T]$, $\mathrm{vc}(V, \mathbb{G}_t) = \mathrm{vc}(\mathbf{X}^{(t)}, \mathbb{H}) \le \mathrm{vc}(\mathbb{R}^d, \mathbb{H}) = \lambda$. By Sauer's lemma (Sauer, 1972), V can be classified in less than $(3n)^\lambda$ ways using the hypotheses in $\mathbb{G}_t$. Hence, $\mathbb{G}_{[T]\times\mathbb{H}}$ can classify V in less than $T(3n)^\lambda$ ways, implying $\lambda' < \log_2 T + \lambda\log_2(3n)$ (applying the definition of VC dimension). □

Corollary 5.3. *The label complexity in Theorem 5.2 is at most* $\frac{\lambda}{\epsilon^2}\cdot(\epsilon+\nu^*)\cdot\mathrm{polylog}(nT/(\delta\epsilon))$.

Proof. The corollary follows from $\theta_{\mathrm{all}}(\nu^*+\epsilon) \le \frac{1}{\nu^*+\epsilon}$ (Property 1, Proposition 3.1). □

Remark 5.4. Our solution can be easily converted to a budget algorithm. It suffices to change $\mathrm{ActLearn_V}(\epsilon, \delta, \mathbb{G}_{[T]\times\mathbb{H}})$ to $\mathrm{ActLearn_B}(L, \delta, \mathbb{G}_{[T]\times\mathbb{H}})$ where L is the label budget. Theorem 5.2 still holds (replacing Lemma 3.2 with Lemma 3.3 in the proof).

5.2. A Budget Algorithm

We have seen earlier that, as far as verifiable algorithms are concerned, it is a bad idea to learn each instance space independently. However, the situation changes dramatically for budget algorithms. Our budget CSAL algorithm — named Budget-CSAL — simply learns a hypothesis from every instance space independently and then returns the best of all, as shown below.

```
Budget-CSAL($L, \delta, \mathbb{H}$)
1. **for** $t \leftarrow 1$ **to** $T$ **do**
2.     $h_t \leftarrow$ output of $\mathrm{ActLearn_B}(L/(2T), \delta/2, \mathbb{H})$
       with $\mathcal{D} = \mathcal{U}_t$
3. $\mathcal{H} = \{(t, h_t) \mid t \in [T]\}$
4. $g_{t,h} \leftarrow$ output of $\mathrm{ActLearn_B}(L/2, \delta/2, \mathbb{G}_\mathcal{H})$
   with $\mathcal{D} = \mathcal{U}$          /* see (9) for $\mathbb{G}_\mathcal{H}$ */
5. **return** $(t, h)$
```

At Line 2, we run $\mathrm{ActLearn_B}$ (the algorithm in Lemma 3.3) by manually fixing the agnostic distribution $\mathcal{D}$ to $\mathcal{U}_t$ (in the way described in Section 5.1), whereas $\mathcal{D}$ is manually fixed to $\mathcal{U}$ at Line 4.

Theorem 5.5. Budget-CSAL *has a label complexity*

$$\lambda \cdot T \cdot \theta_{t^*}(\nu^*+\epsilon)\cdot(1+\nu^*/\epsilon)^2\cdot\mathrm{polylog}(T/(\delta\epsilon)). \quad (11)$$

Before giving the proof, let us first point out several important facts.

- For $T = \mathrm{polylog}(1/(\delta\epsilon))$ — a scenario with practical importance — the label complexity becomes $\tilde{O}(\lambda \cdot \theta_{t^*}(\nu^*+\epsilon)\cdot(1+\nu^*/\epsilon)^2)$. This is the "ideal" complexity achievable if we *had* known t^* in advance (in that case, ignore all instance spaces except $\mathcal{X}_{t^*}$ and apply Lemma 3.3 directly in $\mathcal{X}_{t^*}$). Theorem 5.2, on the other hand, does not offer such a guarantee: even for $T = 2$, the term $\theta_{\mathrm{all}}(\nu^*+\epsilon)$ in (10) can be considerably larger than $\theta_{t^*}(\nu^*+\epsilon)$, as we will see in Remark 6.7.
- For arbitrarily large T, the relationship between $\theta_{\mathrm{all}}(\nu^*+\epsilon)$ and $T\cdot\theta_{t^*}(\nu^*+\epsilon)$ is not fixed. It is possible for $\theta_{\mathrm{all}}(\nu^*+\epsilon)$ to be far greater than $T\cdot\theta_{t^*}(\nu^*+\epsilon)$; an example will appear in Remark 6.7. Conversely, there are also scenarios[2] where $\theta_{\mathrm{all}}(\nu^*+\epsilon)$ is far less than $T\cdot\theta_{t^*}(\nu^*+\epsilon)$.
- Theorems 5.2 and 5.5 (see also Remark 5.4) together imply that there is a budget algorithm whose label complexity is upper bounded by the smaller between (10) and (11).

Proof of Theorem 5.5. We require

$$\frac{L}{2T} \ge \theta_{t^*}\Big(\nu^* + \frac{\epsilon}{2}\Big)\cdot\Big(1+\frac{\nu^*}{\epsilon/2}\Big)^2\lambda\cdot\mathrm{polylog}\Big(\frac{1}{\delta\epsilon/4}\Big). \quad (12)$$

By Lemma 3.3, the hypothesis h_{t^*} obtained at Line 2 of Budget-CSAL for $t = t^*$ satisfies $\mathrm{er}(t^*, h_{t^*}) \le \nu^* + \epsilon/2$ with probability at least $1-\delta/2$. Assume that the event occurs. Let $g^* = \arg\min_{g\in\mathbb{G}_\mathcal{H}} \mathrm{er}(g)$. Because $h_{t^*} \in \mathcal{H}$, we know $\mathrm{er}(g^*) = \mathrm{er}(t^*, h_{t^*}) \le \nu^* + \epsilon/2$.

We also require

$$\frac{L}{2} \ \ge\ \theta_{\mathbb{G}_\mathcal{H},\mathcal{U}}\Big(\mathrm{er}(g^*) + \frac{\epsilon}{2}\Big)\cdot\Big(1+\frac{\mathrm{er}(g^*)}{\epsilon/2}\Big)^2\cdot \lambda'\cdot\mathrm{polylog}\Big(\frac{1}{\delta\epsilon/4}\Big) \quad (13)$$

where $\lambda' := \mathrm{vc}(V, \mathbb{G}_\mathcal{H})$. By Lemma 3.2, with probability at least $1-\delta/2$, the hypothesis $g_{t,h}$ returned at Line 4 satisfies $\mathrm{er}(g_{t,h}) \le \mathrm{er}(g^*) + \epsilon/2 \le \nu^* + \epsilon$. Therefore, the output (t, h) of Budget-CSAL satisfies $\mathrm{er}(t, h) \le \nu^* + \epsilon$ with probability at least $1-\delta$.

Properties 2 and 3 in Proposition 3.1 tell us $\theta_{t^*}(\nu^* + \frac{\epsilon}{2}) \le \theta_{t^*}(\frac{\nu^*+\epsilon}{2}) \le 2\theta_{t^*}(\nu^*+\epsilon)$. From Property 1 of the same proposition, we know $\theta_{\mathbb{G}_\mathcal{H},\mathcal{U}}(r) \le |\mathbb{G}_\mathcal{H}| = T$ for any $r > 0$. Furthermore, $\lambda' \le \log_2|\mathbb{G}_\mathcal{H}| = \log_2 T$. Equipped with these bounds, it is straightforward to verify that both (12) and (13) are fulfilled when L is at least the value in (11). □

[2]For example, this happens when G is the special graph where each vertex has only one edge to itself (a self-loop). In this case, $\mathbf{X}^{(t)}$ remains the same for all $t \ge 1$ and $\theta_{\mathrm{all}}(\nu^*+\epsilon) = \theta_{t^*}(\nu^*+\epsilon)$.

6. Hardness of Verification in CSAL

We are now ready to discuss the paper's most important result: a separation between verifiable and budget CSAL algorithms. Section 6.1 will describe the separation and present a complete proof. Section 6.2 will then adapt the proof to analyze the tightness of Theorem 5.2.

6.1. Separation between Budget and Verification

The separation can now be formally stated as:

Theorem 6.1. *No verifiable algorithm can achieve a label complexity in the form of* (11)*, even if* $T = 2$*.*

The rest of the subsection serves as a proof of the theorem. The hardness stems from the intuitive fact that a verifiable algorithm must learn sufficiently well in $T = 2$ instance spaces simultaneously. To prevent the algorithm from ignoring either space, we must force it to return a hypothesis from a space that varies depending on the GCN input. Furthermore, the GCN input must have certain properties — some of which are related to disagreement coefficients — for us to control the algorithm's cost according to (11). The crux of the proof is to design a suitable family of GCN inputs, which is unique to cross-space learning on GCNs and manifests how our argument differs from the known hardness proofs in active learning.

A Problem of Distinguishing Distributions. Fix arbitrary values ϵ and n such that $\epsilon \in (0, 1/12]$, and $\frac{1}{6\epsilon}$ and ϵn are both integers. Set

$$\ell \quad := \quad 6\epsilon n$$

and define interval

$$I_j := ((j-1)\ell, j\ell], \text{ for } j \in [1/(6\epsilon)].$$

Note that $(0, n]$ is partitioned by $I_1, I_2, ..., I_{\frac{1}{6\epsilon}}$.

Next, we introduce $1 + \frac{1}{6\epsilon}$ matrices $\mathbf{Y}$ of dimensions $n \times 1$ with $\mathbf{Y}_i \in \{-1, 1\}$ for all $i \in [n]$. For each $j \in [\frac{1}{6\epsilon}]$, matrix $\mathbf{Y}^j$ is defined as follows.

- $\mathbf{Y}_i^j := -1$ for $i \notin I_j$;
- $\mathbf{Y}_i^j := -1$ for the $\ell/3$ smallest values of $i \in I_j$ and $\mathbf{Y}_i^j := 1$ for the other $2\ell/3$ values of $i \in I_j$.

In addition, define $\mathbf{Y}^{\text{neg}}$ as the matrix with $\mathbf{Y}_i^{\text{neg}} := -1$ for all $i \in [n]$.

Consider two distributions over $\mathbb{Y} := \{\mathbf{Y}^{\text{neg}}, \mathbf{Y}^1, ..., \mathbf{Y}^{\frac{1}{6\epsilon}}\}$.

- $\mathcal{D}_1$: uniform over $\{\mathbf{Y}^1, ..., \mathbf{Y}^{\frac{1}{6\epsilon}}\}$ (i.e., probability 0 for $\mathbf{Y}^{\text{neg}}$);
- $\mathcal{D}_2$: probability 1 for $\mathbf{Y}^{\text{neg}}$.

Define distribution $\mathcal{D}_3 := 0.5\mathcal{D}_1 + 0.5\mathcal{D}_2$. In the *distinguishing problem*, given a $\mathbf{Y} \sim \mathcal{D}_3$, a (deterministic/randomized) algorithm $\mathcal{B}$ needs to decide whether $\mathbf{Y}$ comes from $\mathcal{D}_1$ or $\mathcal{D}_2$. In the beginning, $\mathbf{Y}$ is invisible. $\mathcal{B}$ can request an oracle to reveal $\mathbf{Y}_i$ for any $i \in [n]$ — we will call the request a *probe* — and must output either $\mathcal{D}_1$ or $\mathcal{D}_2$ in the end. The *cost* of $\mathcal{B}$ is the number of probes.

Lemma 6.2. *For the distinguishing problem, if an algorithm* $\mathcal{B}$ *ensures* $\Pr_{\mathbf{Y}\sim\mathcal{D}_3}[\mathcal{B}$ *errs on* $\mathbf{Y}] \le 1/8$*,* $\mathcal{B}$ *must make more than* $1/(12\epsilon)$ *probes on at least one* $\mathbf{Y} \in \mathbb{Y}$ *with a non-zero probability.*

Proof. Let $\mathcal{B}_{\text{det}}$ be a deterministic algorithm that has cost at most $1/(12\epsilon)$ on any $\mathbf{Y} \in \mathbb{Y}$. We will show that $\Pr_{\mathbf{Y}\sim\mathcal{D}_3}[\mathcal{B}_{\text{det}}$ errs on $\mathbf{Y}] \ge 1/4$. Let Z be the output of $\mathcal{B}_{\text{det}}$ when all the probes reveal label -1. If $Z = \mathcal{D}_1$, $\mathcal{B}_{\text{det}}$ errs on $\mathbf{Y}^{\text{neg}}$ and, hence, $\Pr_{\mathbf{Y}\sim\mathcal{D}_3}[\mathcal{B}_{\text{det}}$ errs on $\mathbf{Y}] \ge 1/2$. On the other hand, if $Z = \mathcal{D}_2$, then $\mathcal{B}_{\text{det}}$ errs on at least half of the inputs from $\{\mathbf{Y}^1, ..., \mathbf{Y}^{\frac{1}{6\epsilon}}\}$,[3] meaning $\Pr_{\mathbf{Y}\sim\mathcal{D}_3}[\mathcal{B}_{\text{det}}$ errs on $\mathbf{Y}] \ge 1/4$.

Consider now a randomized algorithm $\mathcal{B}$ that always has cost at most $1/(12\epsilon)$ on every $\mathbf{Y} \in \mathbb{Y}$. $\mathcal{B}$ can be regarded as a distribution over a family of deterministic algorithms $\mathcal{B}_{\text{det}}$ each incurring cost no more than $1/(12\epsilon)$ on any $\mathbf{Y}$. As $\Pr_{\mathbf{Y}\sim\mathcal{D}_3}[\mathcal{B}_{\text{det}}$ errs on $\mathbf{Y}] \ge 1/4$ for every $\mathcal{B}_{\text{det}}$, $\Pr_{\mathbf{Y}\sim\mathcal{D}_3}[\mathcal{B}$ errs on $\mathbf{Y}]$ must be at least $1/4$.

Therefore, if $\Pr_{\mathbf{Y}\sim\mathcal{D}_3}[\mathcal{B}$ errs on $\mathbf{Y}] \le 1/8$, there must be at least one $\mathbf{Y} \in \mathbb{Y}$, on which $\mathcal{B}$ makes at least $1/(12\epsilon)$ probes with a non-zero probability. □

From Verification to Distinguishing. We now construct a family of GCN inputs such that if a verifiable algorithm $\mathcal{A}$ can achieve the label complexity (11) on those inputs, we can obtain an algorithm $\mathcal{B}$ for the distinguishing problem to beat Lemma 6.2.

Let ϵ and n be chosen as in the distinguishing problem. To build a GCN input, we design G, $\mathbf{X}$, $\mathbf{Y}$, T, and $\mathbb{H}$ as follows.

- G is a complete graph on n vertices with self-loops.
- $\mathbf{X}$ is an $n \times 1$ matrix where $\mathbf{X}_i := i$ for all vertices $i \in [n]$. We draw $\mathbf{Y} \sim \mathcal{D}_3$ and fix $T := 2$.

[3] Among all the intervals $I_1, ..., I_{1/(6\epsilon)}$, $\mathcal{B}_{\text{det}}$ can probe vertices of at most $1/(12\epsilon)$ intervals. Let I_j, for some $j \in [1/(6\epsilon)]$, be an interval that $\mathcal{B}_{\text{det}}$ never probes. When $\mathbf{Y} = \mathbf{Y}^j$, all the probes must return -1 and, hence, $\mathcal{B}_{\text{det}}$ errs by returning $\mathcal{D}_2$.

- Given an interval $I := (a, b]$, define $h^I(x)$ as the hypothesis that returns 1 if $x \in I$, or -1 otherwise. Let

$$\mathbb{H} := \{h^{I_j} \mid j \in [1/(6\epsilon)]\}. \tag{14}$$

We will use $\mathbf{Y}$ to identify a GCN input because all the other components of an input are fixed. Let $\lambda := (\mathbb{R}, \mathbb{H}) = 2$.

Proposition 6.3. *For any $i \in [n]$, $\mathbf{X}_i^{(2)} = (n+1)/2$.*

Proof. See Appendix A. □

Define

$$\rho := \text{the only } j \in [1/(6\epsilon)] \text{ s.t. } (n+1)/2 \in I_j. \tag{15}$$

Proposition 6.3 implies that there are, effectively, only two different hypotheses on $\mathbf{X}^{(2)}$:

- h^{I_ρ}, which classifies all vertices as 1;
- every other hypothesis in $\mathbb{H}$, which classifies all vertices as -1.

Proposition 6.4. *If $\mathbf{Y}$ is from $\mathcal{D}_1$, $t^* = 1$ and $\nu^* = 2\epsilon$. Otherwise, $t^* = 2$ and $\nu^* = 0$. In both cases, any $h \in \mathbb{H}$ with $\mathrm{er}(t^*, h) \le \nu^* + \epsilon$ is an optimal hypothesis on $\mathbf{X}^{(t^*)}$, i.e., $\mathrm{er}(t^*, h) = \nu^*$.*

Proof. Consider $\mathbf{Y} = \mathbf{Y}^k$ for some $k \in [1/(6\epsilon)]$. On $\mathbf{X}^{(1)}$, h^{I_k} correctly classifies all vertices, except for the $\ell/3$ smallest vertices in I_k; hence, $\mathrm{er}(1, h^{I_k}) = \frac{\ell/3}{n} = 2\epsilon$. For any $j \ne k$, h^{I_j} mis-classifies the ℓ vertices in I_j and also the $2\ell/3$ largest vertices in I_k; thus, $\mathrm{er}(1, h^{I_j}) = \frac{5\ell/3}{n} = 10\epsilon$. On $\mathbf{X}^{(2)}$, h^{I_ρ} mis-classifies all vertices, except the $2\ell/3$ largest vertices in I_j; hence, $\mathrm{er}(2, h^{I_\rho}) = (n - 2\ell/3)/n = 1 - 4\epsilon$. For any $j \ne \rho$, h^{I_j} correctly classifies all vertices, except for the $2\ell/3$ largest vertices in I_k; thus, $\mathrm{er}(2, h^{I_j}) = \frac{2\ell/3}{n} = 4\epsilon$. Therefore, $\nu^* = 2\epsilon$, $t^* = 1$, $(1, h^{I_k})$ is the only legal output, and $\mathrm{er}(1, h^{I_k}) = \nu^*$.

For $\mathbf{Y} = \mathbf{Y}^{\text{neg}}$, in the same fashion, one can verify that $\mathrm{er}(1, h) = 6\epsilon$ for every $h \in \mathbb{H}$, $\mathrm{er}(2, h^{I_j}) = 0$ for every $j \ne \rho$, and $\mathrm{er}(2, h^{I_\rho}) = 1$. Therefore, $\nu^* = 0$, $t^* = 2$, the set $\{(2, h^{I_j}) \mid j \ne \rho\}$ contains all the legal outputs, and $\mathrm{er}(2, h^{I_j}) = \nu^*$ for all $j \ne \rho$. □

Henceforth, we say that an algorithm *fails* on a GCN input $\mathbf{Y} \in \mathbb{Y}$ if it does not return a pair (t, h) with $\mathrm{er}(t, h) = \nu^*$.

Proposition 6.5. *If $\mathbf{Y}$ comes from $\mathcal{D}_1$, $\theta_{t^*}(\nu^* + \epsilon) = 1/(12\epsilon)$. Otherwise, $\theta_{t^*}(\nu^* + \epsilon) = 1$.*

Proof. See Appendix B. □

Fix $\delta = 1/16$. Using Propositions 6.4 and 6.5, we can simplify the label complexity (11) of $\mathcal{A}$ into

$$\theta_{t^*}(\nu^* + \epsilon) \cdot \alpha$$

for some $\alpha = \mathrm{polylog}(\frac{1}{\epsilon})$. Choose ϵ small enough to make

$$\alpha \quad < \quad 1/(12\epsilon).$$

Given any GCN input $\mathbf{Y} \in \mathbb{Y}$, with probability $\ge 15/16$, $\mathcal{A}$ returns an optimal hypothesis on $\mathbf{X}^{(t^*)}$ (due to Proposition 6.4) with cost at most $\theta_{t^*}(\nu^* + \epsilon) \cdot \alpha$.

Lemma 6.6. *From $\mathcal{A}$, we can obtain an algorithm $\mathcal{A}'$ with the following properties.*

- *Given any GCN input $\mathbf{Y} \in \mathbb{Y}$, $\mathcal{A}'$ requests at most $\alpha \cdot \theta_{t^*}(\nu^* + \epsilon)$ labels with probability $\ge 15/16$;*
- *Given any GCN input $\mathbf{Y} \in \mathbb{Y}$, $\mathcal{A}'$ outputs $(2, h^{I_j})$ for some $j \ne \rho$ when all the label requests return -1;*
- *$\Pr_{\mathbf{Y} \sim \mathcal{D}_3}[\mathcal{A}'$ fails on $\mathbf{Y}] \le 1/16$.*

Proof. $\mathcal{A}$ can be regarded as a distribution over a family of deterministic algorithms $\mathcal{A}_{\text{det}}$. We modify each $\mathcal{A}_{\text{det}}$ as follows. Let Z be the output of $\mathcal{A}_{\text{det}}$ in the scenario where all the label requests return -1. If $Z = (2, h_j)$ for some $j \ne \rho$, leave $\mathcal{A}_{\text{det}}$ unchanged. Otherwise, modify the output to $(2, h_j)$ for an arbitrary $j \ne \rho$. Let $\mathcal{A}'_{\text{det}}$ be the altered algorithm. Since $\mathcal{A}_{\text{det}}$ fails on $\mathbf{Y}^{\text{neg}}$ but $\mathcal{A}'_{\text{det}}$ does not, $\Pr_{\mathbf{Y} \sim \mathcal{D}_3}[\mathcal{A}_{\text{det}}$ fails on $\mathbf{Y}] \ge 1/2$ and $\Pr_{\mathbf{Y} \sim \mathcal{D}_3}[\mathcal{A}'_{\text{det}}$ fails on $\mathbf{Y}] \le 1/2$.

The above changes turn $\mathcal{A}$ into an alternative randomized algorithm $\mathcal{A}'$. Our modification does not affect how many labels are requested. Hence, $\mathcal{A}'$ inherits from $\mathcal{A}$ the property of ensuring cost at most $\theta_{t^*}(\nu^* + \epsilon) \cdot \alpha$ with probability at least $15/16$ on any $\mathbf{Y}$. Furthermore, $\Pr_{\mathbf{Y} \sim \mathcal{D}_3}[\mathcal{A}'$ fails on $\mathbf{Y}] \le \Pr_{\mathbf{Y} \sim \mathcal{D}_3}[\mathcal{A}$ fails on $\mathbf{Y}] \le 1/16$. □

$\mathcal{A}'$ need not be verifiable because it guarantees only a small *average* failure probability over $\mathcal{D}_3$, rather than a small probability on every $\mathbf{Y}$. Nevertheless, it allows us to design an algorithm $\mathcal{B}$ for the distinguishing problem:

```
algorithm $\mathcal{B}$
1. run $\mathcal{A}'$ up to $\alpha$ label requests
2. if $\mathcal{A}'$ has not terminated or has found a label 1 then
3.     return "$\mathbf{Y}$ from $\mathcal{D}_1$"
4. else return "$\mathbf{Y}$ from $\mathcal{D}_2$"
```

$\mathcal{B}$ errs only in two events

1. $\mathbf{Y} \ne \mathbf{Y}^{\text{neg}}$ yet $\mathcal{A}'$ terminates within α label requests all of which returned -1;

2. $\mathbf{Y} = \mathbf{Y}^{\text{neg}}$ yet $\mathcal{A}'$ requests more than α labels.

In event 1, by Lemma 6.6, $\mathcal{A}'$ returns $(2, h^{I_j})$ for some $j \neq \rho$ and thus fails on $\mathbf{Y}$. Hence, event 1 occurs with probability at most $\Pr_{\mathbf{Y}\sim\mathcal{D}_3}[\mathcal{A}' \text{ fails on } \mathbf{Y}] \leq 1/16$. On the other hand:

$$\begin{aligned}\Pr[\text{event 2}] &= \Pr_{\mathcal{D}_3}[\mathbf{Y}^{\text{neg}}] \cdot \Pr[\mathcal{A}' \text{ has cost} > \alpha \text{ on } \mathbf{Y}^{\text{neg}}] \\ &\leq (1/2)\cdot(1/16) = 1/32\end{aligned}$$

where the inequality used Lemma 6.6 and $\alpha \cdot \theta_{t^*}(\nu^* + \epsilon) = \alpha$ when $\mathbf{Y} = \mathbf{Y}^{\text{neg}}$ (see Proposition 6.5).

Therefore, $\mathcal{B}$ satisfies $\Pr_{\mathbf{Y}\sim\mathcal{D}_3}[\mathcal{B} \text{ errs on } \mathbf{Y}] \leq 1/16 + 1/32 < 1/8$ but makes at most $\alpha < 1/(12\epsilon)$ probes with probability 1 on every $\mathbf{Y} \in \mathbb{Y}$. This contradicts Lemma 6.2 and completes the proof of Theorem 6.1.

Remark 6.7. When $\mathbf{Y} = \mathbf{Y}^{\text{neg}}$, one can verify that $\theta_{\text{all}}(\nu^* + \epsilon) = \Omega(1/\epsilon)$ and $\theta_{t^*}(\nu^* + \epsilon) = 1$. In this case, the budget label complexity in (11) is lower than the verifiable one in (10) by a factor of nearly $1/\epsilon$.

6.2. Tightness of Theorem 5.2

The complexity in (10) may seem "obviously" tight at first glance. After all, when $T = 1$, cross-space active learning degenerates into traditional active learning such that $\theta_{\text{all}} = \theta_1$, in which case Theorem 5.2 degenerates into Lemma 3.2 up to a polylog factor. The argument, although correct, overlooks the influence of T. For example, it does not rule out the possibility of improving (10) by a factor of T^c for some arbitrarily small constant $c > 0$. Such a factor vanishes when $T = 1$ and, thus, does not improve Lemma 3.2.

By modifying our construction in Section 6.1, we can rule out the aforementioned T^c improvement. In fact, this is true even if $\theta_{\text{all}}(\nu^* + \epsilon)$ is replaced with $\sum_{t=1}^{T} \theta_t(\nu^* + \epsilon)$ in (10) (the former is asymptotically bounded by the latter, as shown in Lemma 5.1). A crucial observation is that Proposition 6.3 implies $\mathbf{X}_i^t = (1+n)/2$ for any $i \in [n]$ and $t \geq 2$. By setting $T = 1/\epsilon$, we thus create $T - 1$ identical instance spaces $\mathcal{X}_2, \mathcal{X}_3, ..., \mathcal{X}_T$. Proposition 6.5 implies $\sum_{t=1}^{T} \theta_t(\nu^* + \epsilon) = \Theta(1/\epsilon)$. Hence, under a T^c improvement, the resulting algorithm must request $(1/\epsilon)^{1-c'}$ labels on every GCN input $\mathbf{Y} \in \mathbb{Y}$ for some constant $c' > 0$. Such an algorithm can then be leveraged to tackle the distinguishing problem with a guarantee contradicting Lemma 6.2.

7. Stochastic Learning on a GCN

The problem formulated in Section 2 is non-stochastic because the input matrices $\mathbf{X}$ and $\mathbf{Y}$ are arbitrarily given. This section will describe an alternative formulation where $\mathbf{X}$ and $\mathbf{Y}$ are generated in a stochastic manner.

The oracle first chooses a graph $G := (V, E)$ whose matrix $\mathbf{S}$ as defined in (1) is invertible[4]. For every vertex $i \in V := [n]$, the oracle draws a sample (x_i, y_i) from a certain distribution $\mathcal{D}_i$ over $\mathbb{R}^d \times \mathcal{Y}$, where $\mathcal{Y} := \{-1, 1\}$ and $d \geq 1$ is some integer. The oracle then decides on some integer $k \geq 2$, generates an $n \times d$ matrix $\mathbf{X}^{(k)}$ by setting $\mathbf{X}_i^{(k)} = x_i$ for each $i \in [n]$, and creates an $n \times 1$ matrix $\mathbf{Y}$ with $\mathbf{Y}_i = y_i$ for each i. Finally, the oracle computes $\mathbf{X} = (\mathbf{S}^{-1})^{k-1}\mathbf{X}^{(k)}$ and releases $\mathbf{X}$, G, an integer $T \geq k$ to a learning algorithm, which then tackles the CSAL problem defined in Section 2 by choosing an appropriate hypothesis class $\mathbb{H}$.

In the above, every vertex $i \in [n]$ has a dedicated distribution $\mathcal{D}_i$. This does not mean that there needs to be n distinct distributions because many vertices can share the same distribution. This, naturally, is related to *graph clustering* (i.e., spectral clustering), which divides V into disjoint clusters $V_1, V_2, ..., V_s$ for some $s \geq 1$. Vertices in V_j (for $j \in [s]$) share the same distribution, which we denote as $\mathcal{D}_{V_j}$.

The stochastic model allows us to study algorithms that may not work well on *every* $(\mathbf{X}, \mathbf{Y})$, but achieve good *overall* performance on the inputs generated according to $\mathcal{D}_{V_1}, ..., \mathcal{D}_{V_s}$. Next, we illustrate the idea on the special case $s = 1$ where all vertices obey the same distribution $\mathcal{D}$. If $\nu := \inf_{h\in\mathbb{H}} \text{er}_{\mathcal{D}}(h)$, a reasonable goal is to find a pair $(t, h) \in [T] \times \mathbb{H}$ with $\text{er}(t, h) \leq \nu + \epsilon$. It is known (see Chapter 7 of (Hanneke, 2014)) that many choices of $\mathcal{D}$ and $\mathbb{H}$ lead to *constant-bounded* disagreement coefficients $\theta_{\mathbb{H},\mathcal{D}}$. This translates to $\theta_k(\nu^* + \epsilon) = O(1)$ — where $\theta_k(r)$ follows the definition in (8) — with high probability as long as n is not too small (Tao & Wang, 2019). Therefore, our algorithm in Theorem 5.5 achieves a label complexity $\lambda \cdot T \cdot (1 + \nu^*/\epsilon)^2 \cdot \text{polylog}(T/(\delta\epsilon))$ with high probability, over an input generated from $\mathcal{D}$.

The model echoes the rationale behind GCNs that the topology structure in G should provide valuable hints for discovering features more amenable to learning than those in $\mathbf{X}$. Indeed, because of the matrix inversion performed by the oracle, $\mathbf{X}$ alone (i.e., without G) would be useless for learning. The algorithm essentially aims to recover $\mathbf{X}^{(k)}$, namely, the "genuine" features that the oracle concealed using G. When viewed through this perspective, the success of GCNs receives a plausible explanation.

Acknowledgements

This research was supported by GRF projects 14203421 and 14207820 from HKRGC. We thank the anonymous reviewers for their insightful comments.

[4]As each vertex has a positive degree, this is equivalent to requiring the adjacency matrix $\mathbf{A}$ to be invertible. Many, but not all, graphs fulfill the requirement.

References

Atwood, J. and Towsley, D. Diffusion-convolutional neural networks. In *Proceedings of Neural Information Processing Systems (NIPS)*, pp. 1993–2001, 2016.

Balcan, M. and Blum, A. A discriminative model for semi-supervised learning. *Journal of the ACM (JACM)*, 57(3): 19:1–19:46, 2010.

Balcan, M., Broder, A. Z., and Zhang, T. Margin based active learning. In *Proceedings of Conference on Learning Theory (COLT)*, volume 4539, pp. 35–50, 2007.

Balcan, M., Beygelzimer, A., and Langford, J. Agnostic active learning. *Journal of Computer and System Sciences (JCSS)*, 75(1):78–89, 2009.

Balcan, M., Hanneke, S., and Vaughan, J. W. The true sample complexity of active learning. *Mach. Learn.*, 80 (2-3):111–139, 2010.

Beygelzimer, A., Dasgupta, S., and Langford, J. Importance weighted active learning. In *Proceedings of International Conference on Machine Learning (ICML)*, volume 382, pp. 49–56, 2009.

Bruna, J., Zaremba, W., Szlam, A., and LeCun, Y. Spectral networks and locally connected networks on graphs. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2014.

Castro, R. M. and Nowak, R. D. Minimax bounds for active learning. *IEEE Trans. Inf. Theory*, 54(5):2339–2353, 2008.

Chen, L., Wu, L., Hong, R., Zhang, K., and Wang, M. Revisiting graph based collaborative filtering: A linear residual graph convolutional network approach. In *Association for the Advancement of Artificial Intelligence (AAAI)*, pp. 27–34, 2020.

Dasgupta, S. Coarse sample complexity bounds for active learning. In *Proceedings of Neural Information Processing Systems (NIPS)*, pp. 235–242, 2005.

Dasgupta, S., Hsu, D. J., and Monteleoni, C. A general agnostic active learning algorithm. In *Proceedings of Neural Information Processing Systems (NIPS)*, pp. 353–360, 2007.

Defferrard, M., Bresson, X., and Vandergheynst, P. Convolutional neural networks on graphs with fast localized spectral filtering. In *Proceedings of Neural Information Processing Systems (NIPS)*, pp. 3837–3845, 2016.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *Proceedings of International Conference on Machine Learning (ICML)*, volume 70, pp. 1263–1272.

Hamilton, W. L., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. In *Proceedings of Neural Information Processing Systems (NIPS)*, pp. 1024–1034, 2017.

Hanneke, S. A bound on the label complexity of agnostic active learning. In *Proceedings of International Conference on Machine Learning (ICML)*, pp. 353–360, 2007.

Hanneke, S. Rates of convergence in active learning. *The Annals of Statistics*, 39(1):333–361, 2011.

Hanneke, S. Activized learning: Transforming passive to active with improved label complexity. *Journal of Machine Learning Research (JMLR)*, 13:1469–1587, 2012.

Hanneke, S. Theory of disagreement-based active learning. *Foundations and Trends in Machine Learning*, 7(2-3): 131–309, 2014.

He, X., Deng, K., Wang, X., Li, Y., Zhang, Y., and Wang, M. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Special Interest Group on Information Retrieval (SIGIR)*, pp. 639–648, 2020.

Kaariainen, M. Active learning in the non-realizable case. In *Proceedings of International Conference on Algorithmic Learning Theory (ALT)*, pp. 63–77, 2006.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2017.

Klicpera, J., Bojchevski, A., and Gunnemann, S. Predict then propagate: Graph neural networks meet personalized pagerank. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2019.

Koltchinskii, V. Rademacher complexities and bounding the excess risk in active learning. *Journal of Machine Learning Research (JMLR)*, 11:2457–2485, 2010.

Lan, C. and Huan, J. Reducing the unlabeled sample complexity of semi-supervised multi-view learning. In Cao, L., Zhang, C., Joachims, T., Webb, G. I., Margineantu, D. D., and Williams, G. (eds.), *Proceedings of ACM Knowledge Discovery and Data Mining (SIGKDD)*, pp. 627–634, 2015.

Li, Q., Han, Z., and Wu, X. Deeper insights into graph convolutional networks for semi-supervised learning. In *Association for the Advancement of Artificial Intelligence (AAAI)*, pp. 3538–3545, 2018a.

Li, Y., Yu, R., Shahabi, C., and Liu, Y. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2018b.

Lin, W., Lan, H., and Li, B. Generative causal explanations for graph neural networks. In *Proceedings of International Conference on Machine Learning (ICML)*, volume 139, pp. 6666–6679, 2021.

Liu, M., Gao, H., and Ji, S. Towards deeper graph neural networks. In *Proceedings of ACM Knowledge Discovery and Data Mining (SIGKDD)*, 2020.

Liu, Z. and Zhou, J. *Introduction to Graph Neural Networks*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2020.

Luo, D., Cheng, W., Xu, D., Yu, W., Zong, B., Chen, H., and Zhang, X. Parameterized explainer for graph neural network. In *Proceedings of Neural Information Processing Systems (NIPS)*, 2020.

Maron, H., Ben-Hamu, H., Serviansky, H., and Lipman, Y. Provably powerful graph networks. In *Proceedings of Neural Information Processing Systems (NIPS)*, pp. 2153–2164, 2019.

Micheli, A. Neural network for graphs: A contextual constructive approach. *IEEE Trans. Neural Networks*, 20(3): 498–511, 2009.

Morris, C., Ritzert, M., Fey, M., Hamilton, W. L., Lenssen, J. E., Rattan, G., and Grohe, M. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Association for the Advancement of Artificial Intelligence (AAAI)*, pp. 4602–4609, 2019.

Muslea, I., Minton, S., and Knoblock, C. A. Active learning with multiple views. *J. Artif. Intell. Res.*, 27:203–233, 2006.

Oono, K. and Suzuki, T. Graph neural networks exponentially lose expressive power for node classification. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2020.

Rong, Y., Huang, W., Xu, T., and Huang, J. Dropedge: Towards deep graph convolutional networks on node classification. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2020.

Sauer, N. On the density of families of sets. *J. Comb. Theory, Ser. A*, 13(1):145–147, 1972.

Schlichtkrull, M. S., Cao, N. D., and Titov, I. Interpreting graph neural networks for NLP with differentiable edge masking. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2021.

Srinivasan, B. and Ribeiro, B. On the equivalence between positional node embeddings and structural graph representations. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2020.

Tao, Y. and Wang, Y. Distribution-sensitive bounds on relative approximations of geometric ranges. In *Proceedings of Symposium on Computational Geometry (SoCG)*, volume 129, pp. 57:1–57:14, 2019.

Wang, W. and Zhou, Z. On multi-view active learning and the combination with semi-supervised learning. In *Proceedings of International Conference on Machine Learning (ICML)*, volume 307, pp. 1152–1159, 2008.

Wang, W. and Zhou, Z. Multi-view active learning in the non-realizable case. In *Proceedings of Neural Information Processing Systems (NIPS)*, pp. 2388–2396, 2010.

Wang, Y., Wang, Y., Yang, J., and Lin, Z. Dissecting the diffusion process in linear graph convolutional networks. In *Proceedings of Neural Information Processing Systems (NIPS)*, 2021.

Wu, F., Jr., A. H. S., Zhang, T., Fifty, C., Yu, T., and Weinberger, K. Q. Simplifying graph convolutional networks. In *Proceedings of International Conference on Machine Learning (ICML)*, volume 97, pp. 6861–6871, 2019.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Yu, P. S. A comprehensive survey on graph neural networks. *IEEE Trans. Neural Networks Learn. Syst.*, 32(1):4–24, 2021.

Xu, K., Li, C., Tian, Y., Sonobe, T., Kawarabayashi, K., and Jegelka, S. Representation learning on graphs with jumping knowledge networks. In *Proceedings of International Conference on Machine Learning (ICML)*, pp. 5449–5458, 2018.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? In *Proceedings of International Conference on Learning Representations (ICLR)*, 2019.

Ying, Z., Bourgeois, D., You, J., Zitnik, M., and Leskovec, J. Gnnexplainer: Generating explanations for graph neural networks. In *Proceedings of Neural Information Processing Systems (NIPS)*, pp. 9240–9251, 2019.

Yuan, H., Yu, H., Wang, J., Li, K., and Ji, S. On explainability of graph neural networks via subgraph explorations. In *Proceedings of International Conference on Machine Learning (ICML)*, volume 139, pp. 12241–12252, 2021.

Zhao, L. and Akoglu, L. Pairnorm: Tackling oversmoothing in gnns. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2020.

Zhu, H. and Koniusz, P. Simple spectral graph convolution. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2021.

Appendix

A. Proof of Proposition 6.3

We use $\mathbf{1}$ to denote an $n \times n$ matrix where every element equals 1. Since G is a complete graph with self-loops, we have $\mathbf{A} = \mathbf{1}$. As the degree of each vertex is n, $\mathbf{D}^{-1/2} = \text{diag}(\frac{1}{\sqrt{n}}, ..., \frac{1}{\sqrt{n}}) = \frac{1}{\sqrt{n}}\mathbf{I}$ where $\mathbf{I}$ is the $n \times n$ identity matrix. Therefore, we have

$$\mathbf{S} = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2} = \frac{1}{\sqrt{n}}\mathbf{I}\mathbf{A}\frac{1}{\sqrt{n}}\mathbf{I} = \frac{1}{n}\mathbf{1}$$

Hence,

$$\mathbf{X}^{(2)} = \mathbf{S}\mathbf{X}^{(1)} = \frac{1}{n}\mathbf{1}\mathbf{X}^{(1)}.$$

We can now compute each element $\mathbf{X}_i^{(2)}$, $i \in [n]$, as

$$\mathbf{X}_i^{(2)} = \frac{1}{n}\sum_{j=1}^{n}\mathbf{1}[i,j]\cdot\mathbf{X}_j^{(1)} = \frac{1}{n}\sum_{j=1}^{n}\mathbf{X}_j^{(1)} = \frac{1}{n}\sum_{j=1}^{n} j = \frac{1}{n}\frac{(1+n)\cdot n}{2} = \frac{1+n}{2}.$$

B. Proof of Proposition 6.5

As in Section 5, for $t = 1$ and 2, impose a distribution $\mathcal{U}_t$ over $\mathbf{X}^{(t)} \times \mathcal{Y}$ where a sample $(x, y) \sim \mathcal{U}_t$ satisfies $\Pr[(x, y) = (\mathbf{X}_i^{(t)}, \mathbf{Y}_i)] = 1/n$ for each $i \in [n]$.

First, consider the case where $\mathbf{Y}$ comes from $\mathcal{D}_1$, and assume, w.l.o.g. (due to symmetry), $\mathbf{Y} = \mathbf{Y}^1$. As explained in the proof of Proposition 6.4, $t^* = 1$, $\nu^* = 2\epsilon$, and h^{I_1} is optimal on $\boldsymbol{X}^{(1)}$. For any $j \neq 1$, $h^{I_1}(x) \neq h^{I_j}(x)$ if and only if $x \in I_1 \cup I_j$. Therefore,

$$\Pr_{(x,y)\sim\mathcal{U}_1}[h^{I_1}(x) \neq h^{I_j}(x)] = \Pr_{(x,y)\sim\mathcal{U}_1}[x \in I_1 \cup I_j] = 2\ell/n = 12\epsilon$$

meaning that

$$B_{\mathcal{U}_1}(h^{I_1}, r) = \begin{cases} \{h^{I_1}\}, & \text{if } r < 12\epsilon \\ \mathbb{H}, & \text{otherwise} \end{cases}$$

where $\mathbb{H}$ is given in (14). When $B_{\mathcal{U}_1}(h^{I_1}, r) = \{h^{I_1}\}$, $\text{DIS}(B_{\mathcal{U}_1}(h^{I_1}, r')) = \emptyset$; when $B_{\mathcal{U}_1}(h^{I_1}, r) = \mathbb{H}$, $\text{DIS}(B_{\mathcal{U}_1}(h^{I_1}, r)) = [n]$. We thus obtain

$$\Pr_{\mathcal{U}_1}[\text{DIS}(B_{\mathcal{U}_1}(h^{I_1}, r))] = \begin{cases} 0, & \text{if } r < 12\epsilon \\ 1, & \text{otherwise.} \end{cases}$$

By the definition of disagreement coefficient — specifically, (5) and (6) — the reader can now easily verify that $\theta_{\mathbb{H},\mathcal{U}_1}(\nu^* + \epsilon) = \theta_{\mathbb{H},\mathcal{U}_1}(3\epsilon) := \theta_{\mathbb{H},\mathcal{U}_1}^{h^{I_1}}(3\epsilon) = 1/(12\epsilon)$. This is also the value of $\theta_1(\nu^* + \epsilon)$, as defined in (8).

Next, we consider the case where $\mathbf{Y}$ comes from $\mathcal{D}_2$, that is, $\mathbf{Y} = \mathbf{Y}^{\text{neg}}$. As explained in the proof of Proposition 6.4, $t^* = 2$, $\nu^* = 0$, and any h^{I_j} with $j \neq \rho$ is optimal on $\boldsymbol{X}^{(2)}$. Fix k to an arbitrary value in $[1/(6\epsilon)]$ different from ρ. Clearly,

$$\Pr_{(x,y)\in\mathcal{D}_2}[h^{I_k}(x) \neq h^{I_j}(x)] = \begin{cases} 0, & \text{if } j \neq \rho \\ 1, & \text{if } j = \rho. \end{cases}$$

When $r < 1$, $B_{\mathcal{U}_2}(h^{I_k}, r) = \mathbb{H}\backslash\{h^{I_\rho}\}$ and $\text{DIS}(B_{\mathcal{U}_2}(h^{I_k}, r)) = \emptyset$; when $r \geq 1$, $B_{\mathcal{U}_2}(h^{I_k}, r) = \mathbb{H}$ and $\text{DIS}(B_{\mathcal{U}_2}(h^{I_k}, r)) = \{(n+1)/2\}$. We thus obtain

$$\Pr_{\mathcal{U}_2}[\text{DIS}(B_{\mathcal{U}_2}(h^{I_k}, r))] = \begin{cases} 0, & \text{if } r < 1 \\ 1, & \text{otherwise.} \end{cases}$$

It is now straightforward to verify $\theta_2(\nu^* + \epsilon) = \theta_2(\epsilon) := \theta_{\mathbb{H},\mathcal{U}_2}^{h^{I_k}}(\epsilon) = 1$.

C. General GCNs

We now generalize the problem definition of Section 2. Let $G := (V, E)$, n, and the $n \times d$ matrix $\mathbf{X}$ all be defined as in Section 2.

Define $\mathbf{X}^{(1)} := \mathbf{X}$ and $d_1 := d$. We consider a deterministic convolution process Conv on G, which

- takes an $n \times d_t$ matrix $\mathbf{X}^{(t)}$ (for some $t \geq 1$) as the parameter, and
- outputs an $n \times d_{t+1}$ matrix $\mathbf{X}^{(t+1)}$.

Apply Conv for $T - 1$ times (for some $T \geq 2$) to obtain $\mathbf{X}^{(2)}, ..., \mathbf{X}^{(T)}$. Each $t \in [T]$ is associated with a class $\mathbb{H}_t$ of hypotheses, each being a function from $\mathbb{R}^{d_t}$ to $\mathcal{Y}$. Let $\lambda_t = \mathrm{vc}(\mathbb{R}^{d_t}, \mathbb{H}_t)$. For any $t \in [T]$ and $h \in \mathbb{H}_t$, define $\mathrm{er}(t, h)$ as in (3). The cross-space minimum error ν^* now equals $\min_{t \in [T], h \in \mathbb{H}_t} \mathrm{er}(t, h)$.

The oracle chooses an $n \times 1$ matrix $\mathbf{Y}$ as in Section 2. The meanings of budget/verfiable algorithm and label complexity remain the same except that $\mathbb{H}$ should be replaced with $\{\mathbb{H}_1, ..., \mathbb{H}_T\}$.

Re-define $\mathbb{G}_t := \{t\} \times \mathbb{H}_t$ and $\mathbb{G}_{\mathrm{all}} := \bigcup_{t \in [T]} \mathbb{G}_t$. For any $\mathcal{H} \subseteq \mathbb{G}_{\mathrm{all}}$, define $\mathbb{G}_{\mathcal{H}}$ as in (9). Regarding disagreement coefficients, re-define $\theta_t(r) := \theta_{\mathbb{H}_t, \mathcal{U}_t}(r)$ for each $t \in [T]$ and $\theta_{\mathrm{all}}(r) := \theta_{\mathbb{G}_{\mathrm{all}}, \mathcal{U}}(r)$. Our algorithms and analysis can be adapted to the above setup. Theorem 5.2 still holds after replacing λ with $\max_{t=1}^{T} \lambda_t$ in (10). Similarly, Theorem 5.5 holds after replacing λ with λ_{t^*} in (11).