Cross-Space Active Learning on Graph Convolutional Networks

Yufei Tao 1 Hao Wu 1 Shiyan 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 $X \times Y$ where $X$ is some instance space and $Y := \{-1, 1\}$ 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 $Y_e \in Y$ that is hidden initially. $X_1, X_2, ..., X_T$ are $T \geq 2$ instance spaces such that $e$ has a representation $X_e^{(t)} \in X_t$ for each $t \in [T]$ (the notation $|x|$ represents the set $\{1, 2, ..., x\}$). $X_t$ is associated with a class $\mathbb{H}_t$ of hypotheses each being a function $h : X_t \rightarrow Y$. The error of an $h \in \mathbb{H}_t$ is $\epsilon_r(t, h) := \frac{1}{n} |\{ e \in V | h(X_e^{(t)}) \neq Y_e \}|$. An algorithm $A$ 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. $A$ needs to output a pair $(t, h)$ and use $h$ to classify the objects of $V \setminus Q$ in $X_t$. The cost of $A$ is $|Q|$. We are interested in algorithms with cost far less than $n$, in which case the number of misclassified objects is essentially $n \cdot \epsilon_r(t, h)$.

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 $X_t$ produces instance space $X_{t+1}$ ($t \geq 1$). The iteration can go on $T - 1$ times to generate $X_2, ..., X_T$ 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 $X_1$.

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 $X$, where $n := |V|$ and $d \geq 1$ is an integer. Each row of $X$ represents the raw features of a distinct vertex in $V$; define $X^{(1)} := X$. Convolution computes an $n \times d_{t+1}$ matrix $X^{(t+1)}$ from $X^{(t)}$, for each $t \in [T - 1]$. Effectively, this casts $V$ into instance spaces $X_1, ..., X_T$ where $X_t \subset \mathbb{R}^d$ 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 $\nu^* := \min_{t \in [T], h \in \mathbb{H}_t} \epsilon_r(t, h)$, namely, the minimum error across all the instance spaces. Define $t^*$ as the $t$ such that $\nu^* = \min_{t \in [T]} \epsilon_r(t, h)$, that is, $X_{t^*}$ is the best instance space. We aim to understand the label complexity of discovering a pair $(t, h)$ with $\epsilon_r(t, h) \leq \nu^* + \epsilon$ 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 $\epsilon$ and terminates only when it can ensure $\epsilon_r(t, h) \leq \nu^* + \epsilon$. 

---

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 that 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 learning, to be reviewed in Section 3.

Section 2 starts by choosing an appropriate graph stochastic on GCNs in a manner. At a high level, an oracle only once. They came to this conclusion after trying the best instance space is obtained by performing convolution is repeated. For example, Kipf and Welling 2017 recommended for their now-famous convolution design, namely, as exactly the cross-space learning problem we aim to solve. However, inspecting all those \( t^* = 2 \) values, 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, \( \tilde{O}(\frac{d^2}{\epsilon^2} \cdot \max_{t=1}^T \lambda_t) \), where \( \lambda_t \) is the VC dimension of \( (X_t, \mathbb{H}_t) \) and \( \tilde{O}(\cdot) \) 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 \( t = 2, 3, 4, \ldots \), 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 \( X^{(t^*)} \) and the vertex labels according to a certain distribution. After that, the oracle computes \( X^{(1)} \) from \( X^{(t^*)} \) through a “reverse convolution” process on \( G \), effectively concealing \( X^{(t^*)} \). Finally, it releases \( X^{(1)}, G, \) and \( T \) to an algorithm for cross-space learning. The model captures a scenario where GCNs promise more accurate classification than learning from \( X^{(1)} \) directly, thereby offering a plausible explanation about the usefulness of GCNs.

### 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 \( M \), we use \( M_t \) to denote the \( i \)-th row of \( M \) and \( M[i, j] \) to denote the element at the \( i \)-th row and \( j \)-th column of \( M \).

**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 \( A \) the adjacency matrix of \( G \), and by \( D := \text{diag}(\deg(1), \ldots, \deg(n)) \) the degree matrix, where \( \deg(i) \) is the degree of vertex \( i \in [n] \). Define

\[
S := D^{-1/2}AD^{-1/2}
\]

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

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

\[
X^{(t+1)} := SX^{(t)} = S'X
\]

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

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

\(^1\)Disagree coefficient is a key concept in the theory of active learning, to be reviewed in Section 3.
To perform label classification, we resort to a class $\mathbb{H}$ of hypotheses each being a function $h : \mathbb{R}^d \to Y$. Conveniently, we regard each $X^{(i)}_t$ as a point in $\mathbb{R}^d$ such that $h(X^{(i)}_t)$ gives the classification result for vertex $i$ based on its features in $X^{(i)}$. For any $t \in [T]$ and $h \in \mathbb{H}$, define
\[
er(t, h) := \frac{|\{i \in [n] \mid h(X^{(i)}_t) \neq Y_i\}|}{n}.
\] (3)

The minimum achievable error is
\[
\nu^* := \min_{t \in [T], h \in \mathbb{H}} er(t, h).
\] (4)

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

A learning algorithm $\mathcal{A}$ has complete details of $G, X, T,$ and $\mathbb{H}$, but knows nothing about $Y$. It can designate any vertex $i \in [n]$ and request the oracle to reveal the label $Y_i$. 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 $h(X^{(i)}_t)$. We consider only the situation where the cost is far less than $n$ such that the number of mistakes by $\mathcal{A}$ is essentially $n \cdot er(t, h)$. The objective of cross-space active learning (CSAL) on a GCN is to minimize $er(t, h)$.

**Remark 2.1.** Under the terminology in Section 1, the instance space $X_t$, for each $t \in [T]$, is the set of $d$-dimensional points in $X^{(i)}$, 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 $G, X, T,$ 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 $\Lambda(\epsilon, \delta, G, X, Y, T, \mathbb{H})$ if, for any $G, X, Y, \mathbb{H}, \epsilon \in (0, 0.5), \delta \in (0, 1), T \geq 2,$ and $L \geq \Lambda(\epsilon, \delta, G, X, Y, T, \mathbb{H})$, the following statement holds with probability at least $1 - \delta$: $\mathcal{A}(L, \delta, G, X, Y, T, \mathbb{H})$ outputs $(t, h)$ with $er(t, h) \leq \nu^* + \epsilon$.

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

A key difference between Definitions 2.2 and 2.3 is what happens when $L$ is far greater than $\Lambda(\epsilon, \delta, G, X, Y, T, \mathbb{H})$. 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 $D$ 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 $\lambda := \text{vc}(\mathcal{X}, \mathbb{H})$. For each $h \in \mathbb{H}$, define $\text{er}_D(h) := \Pr_{(x, y) \sim D}[h(x) \neq y]$ and $\nu := \inf_{h \in \mathbb{H}} \text{er}_D(h)$. We consider that $\mathbb{H}$ has at least one optimal hypothesis $h^*$ with $\text{er}_D(h^*) = \nu$.

An active learning algorithm $\mathcal{A}$ draws a sequence of independent samples $(x_i, y_i) \sim D$, 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 $\Pr_D[S] := \Pr_{(x, y) \sim D}[x \in S]$. For any $\mathcal{H} \subseteq \mathbb{H}$, the disagreement region of $\mathcal{H}$ is
\[
\text{DIS}(\mathcal{H}) := \{x \in \mathcal{X} \mid \forall h \in \mathcal{H} \text{ s.t. } h(x) \neq g(x)\}.
\]

Given any $h \in \mathbb{H}$ and $r > 0$, define
\[
B_D(h, r) := \{g \in \mathbb{H} \mid \Pr_D[\text{DIS}(\{g, h\})] \leq r\}.
\]

For any $h \in \mathbb{H}$, its disagreement coefficient under $D$ is a function $\theta_{D, \mathcal{H}}^r : (0, 1) \to [1, \infty)$ defined as
\[
\theta_{D, \mathcal{H}}^r(r) := \max \left\{1, \sup_{r' > r} \frac{\Pr_D[\text{DIS}(B_D(h, r'))]}{r'}\right\}.
\] (5)

The disagreement coefficient of $\mathcal{H}$ under $D$ is the function
\[
\theta_{D, \mathcal{H}}(r) := \theta_{D, \mathcal{H}}^r(r)
\] (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. $\theta_{\mathbb{H}, D}(r) \leq \min\{1/r, |\mathbb{H}|\}$.
2. $\theta_{\mathbb{H}, D}(r)$ is non-increasing in $r$.
3. $\theta_{\mathbb{H}, D}(r) \leq c \cdot \theta_{\mathbb{H}, D}(cr)$ for any constant $c > 1$.
4. $\theta_{0, G, D}(r) \leq \theta_{\mathbb{H}, D}(r) + \theta_{G, D}(r) + 2$ for any hypothesis classes $\mathbb{H}$ and $G$.
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 \( \mathcal{D}, \epsilon \in (0, 0.5), \delta \in (0, 1), \) the following holds with probability at least \( 1 - \delta \): \( \mathcal{A}(\epsilon, \delta, \mathbb{H}) \) returns a hypothesis \( h \in \mathbb{H} \) with \( \varepsilon_{\mathcal{D}}(h) \leq \nu + \epsilon \) 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

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

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

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

A **budget active learning algorithm** \( \mathcal{A} \) takes three parameters: \( L \geq 1, \delta \in (0, 1), \) 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 \( L \geq \Lambda(\epsilon, \delta, \mathcal{D}, \mathbb{H}) \), the following holds with probability at least \( 1 - \delta \): \( \mathcal{A}(L, \delta, \mathbb{H}) \) returns a hypothesis \( h \in \mathbb{H} \) with \( \varepsilon_{\mathcal{D}}(h) \leq \nu + \epsilon \).

**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 \( \text{ActLearn}_{\delta}(L, \delta, \mathbb{H}) \).

**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 \( \text{polylog}(1/(\delta \epsilon)) \) factor in label complexity, as noted in (Balcan et al., 2010; Hanneke, 2012).

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 \( A^\alpha \) 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 these 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), plays 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 \( \nu \) is treated as a constant, and \( \epsilon \) is treated as an asymptotic parameter approaching 0 (see Section 3 for the definitions of \( \nu \) and \( \epsilon \)).

**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 remediying 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 \( O(\text{polylog} n) \) (recall that \( n \) is the number of vertices). This is why \( T = O(\text{polylog} n) \) presents an important scenario in this field.

The statistic model to be developed in Section 7 belongs to the explainability 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 (Mor-
ris 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 \( \nu_t := \min_{h \in \mathbb{H}} \text{er}(t, h) \) — the function \( \text{er}(t, h) \) was defined in (3) — and let \( h_t^* \) be a hypothesis in \( \mathbb{H} \) satisfying \( \text{er}(t, h_t^*) = \nu_t \). In other words, \( \nu_t \) is the minimum error attainable in instance space \( X_t \), while \( h_t^* \) is a hypothesis giving that error. Define \( t^* \) as an integer in \( [T] \) such that \( \nu_{t^*} = \nu^* \) where \( \nu^* \) is the cross-space minimum error, as defined in (4).

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

\[
\theta_t(r) := \theta_{\mathbb{H}, \mathcal{U}_t}(r)
\]

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 \( \Pr[(x, y) = (i, Y_i)] = 1/n \) 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 \( \text{er}(t, h_t) \leq \nu_t + \epsilon/2 \), and (ii) obtain an estimate \( \hat{\text{er}}(t, h_t) \) of \( \text{er}(t, h_t) \) within absolute error \( \epsilon/2 \). The \((t, h_t)\) having the lowest \( \hat{\text{er}}(t, h_t) \) must enjoy the guarantee \( \text{er}(t, h_t) \leq \nu^* + \epsilon \), 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 \( \hat{\text{er}}(t, h_t) \) for each \( t \) with high confidence by probing \( \mathcal{O}(\frac{1}{\epsilon} (\nu^* + \epsilon)) \). The overall cost is thus \( \mathcal{O}(\frac{1}{\epsilon} \Sigma_{t=1}^T (\nu_t + \epsilon)) \).

We will show how to achieve a label complexity that never exceeds, but can be much smaller than, \( \mathcal{O}(\frac{1}{\epsilon} (\nu^* + \epsilon)) \). Note that \( \frac{1}{\epsilon} (\nu^* + \epsilon) \) can be far less than \( \frac{1}{\epsilon} \Sigma_{t=1}^T (\nu_t + \epsilon) \) 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 \( X_t \).

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

\[
g_{t,h}(i) := h(X^{(t)}_i)
\]

and accordingly

\[
\text{er}(g_{t,h}) := \text{er}(t, h) = \frac{|\{i \in [n] \mid g_{t,h}(i) \neq Y_i\}|}{n}.
\]

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

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

Given \( \epsilon, \delta \), and \( \mathbb{H} \), our verifiable algorithm is:

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

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

For analysis, define

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

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

Lemma 5.1. For any \( r > 0 \), \( \theta_{\mathcal{G}_{[T] \times \mathbb{H}}}(r) < 3 \sum_{t=1}^T \theta(t)(r) \).

Proof. Define \( \mathcal{G}_t = \mathcal{G}_{(t) \times \mathbb{H}} \). By (9), \( \mathcal{G}_{[T] \times \mathbb{H}} = \bigcup_{t \in [T]} \mathcal{G}_t \). 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 \( \theta_{\mathcal{G}_t, \mathcal{U}}(r) = \theta_t(r) \) from the disagreement coefficient definition in Section 3. The lemma now follows from Property 4 in Proposition 3.1 and the fact that \( \theta_t(r) \geq 1 \).

Theorem 5.2. Our algorithm has a label complexity

\[
\lambda \cdot \theta_{\mathcal{G}_{[T] \times \mathbb{H}}}(\nu^* + \epsilon) \cdot (1 + \nu^*/\epsilon)^2 \cdot \text{polylog}(nT/(\epsilon \delta)). \tag{10}
\]

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

The cross-space active learning on graph convolutional networks.
We will bound the number of ways to classify the ways described in Section 5.1), whereas with Lemma 3.3 in the proof).

\[ \text{Theorem 5.5. Budget} \]

Remark 5.4. Our solution can be easily converted to a budget algorithm. It suffices to change ActLearn(V, G|T|\times H), to ActLearnB(L, Δ, G|T|\times 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, Δ, H)

1. for \( t \leftarrow 1 \) to Do
2. \( h_t \leftarrow \text{output of ActLearnB}(L/(2T), Δ/2, H) \)
   with \( D = U_t \)
3. \( \mathcal{H} = \{ (t, h_t) \mid t \in [T] \} \)
4. \( g_{t,h} \leftarrow \text{output of ActLearnB}(L/2, Δ/2, \mathcal{H}) \)
   with \( D = U_t \) \( t^* \) see (9) for \( \mathcal{H} \)
5. return \( (t, h) \)

At Line 2, we run ActLearnB (the algorithm in Lemma 3.3) by manually fixing the agnostic distribution \( D \) to \( U_t \) (in the way described in Section 5.1), whereas \( D \) is manually fixed to \( U_t \) at Line 4.

\[ \text{Theorem 5.5. Budget-CSAL has a label complexity} \]

\[ \lambda \cdot T \cdot \theta_{t^*}(\nu^* + \epsilon) \cdot (1 + \nu^*/\epsilon)^2 \cdot \text{polylog}(T/(\delta \epsilon)) \]  \hspace{1cm} (11)

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

- For \( T = \text{polylog}(1/(\delta \epsilon)) \) — a scenario with practical importance — the label complexity becomes \( O(\lambda \cdot \theta_{t^*}(\nu^* + \epsilon) \cdot (1 + \nu^*/\epsilon)^2) \). This is the “ideal” complexity achievable if we had known \( t^* \) in advance (in that case, ignore all instance spaces except \( X_t \) and apply Lemma 3.3 directly in \( X_t \)). Theorem 5.2, on the other hand, does not offer such a guarantee: even for \( T = 2 \), the term \( \theta_{\text{all}}(\nu^* + \epsilon) \) 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 \( \theta_{\text{all}}(\nu^* + \epsilon) \) and \( T \cdot \theta_{t^*}(\nu^* + \epsilon) \) is not fixed. It is possible for \( \theta_{\text{all}}(\nu^* + \epsilon) \) 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 \( \theta_{\text{all}}(\nu^* + \epsilon) \) 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).

\[ \text{Proof of Theorem 5.5. We require} \]

\[ \frac{L}{2T} \geq \theta_{t^*}(\nu^* + \epsilon) \cdot \left(1 + \frac{\nu^*}{\epsilon/2} \right)^2 \lambda \cdot \text{polylog} \left( \frac{1}{\delta \epsilon/4} \right) \]  \hspace{1cm} (12)

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

We also require

\[ \frac{L}{2} \geq \lambda' \cdot \text{polylog} \left( \frac{1}{\delta \epsilon/4} \right) \]  \hspace{1cm} (13)

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

\[ \text{Properties 2 and 3 in Proposition 3.1 tell us} \]

\[ \theta_{t^*}(\nu^* + \frac{\epsilon}{2}) \leq \theta_{t^*}(\nu^* + \frac{\epsilon}{2}) \leq 2\theta_{t^*}(\nu^* + \epsilon). \]  \hspace{1cm} (12)

From Property 1 of the same proposition, we know \( \theta_{\mathcal{G}_H}(r) \leq |\mathcal{G}_H| = T \) for any \( r > 0 \). Furthermore, \( \lambda' \leq \log_2 |\mathcal{G}_H| = \log_2 T \). 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). \( \Box \)

\(^2\)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, \( X^{(t)} \) remains the same for all \( t \geq 1 \) and \( \theta_{\text{all}}(\nu^* + \epsilon) = \theta_{t^*}(\nu^* + \epsilon) \).
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 algorithms, 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 \( \epsilon \) and \( n \) such that \( \epsilon \in (0, 1/12) \), and \( \frac{1}{6\epsilon} \) and \( \epsilon n \) are both integers. Set

\[
\ell := 6\epsilon n
\]

and define interval

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

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

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 \( \mathcal{Y} := \{\mathbf{Y}^{\text{neg}}, \mathbf{Y}^1, \ldots, \mathbf{Y}^{\frac{1}{6\epsilon}}\} \).

- \( \mathcal{D}_1 \): uniform over \( \{\mathbf{Y}^1, \ldots, \mathbf{Y}^{\frac{1}{6\epsilon}}\} \) (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 \( \Pr_{\mathbf{Y} \sim \mathcal{D}_3}[\mathcal{B} \text{ errs on } \mathbf{Y}] \leq 1/8 \), \( \mathcal{B} \) must make more than \( 1/(12\epsilon) \) probes on at least one \( \mathbf{Y} \in \mathcal{Y} \) 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 \( \mathbf{Y} \in \mathcal{Y} \). We will show that \( \Pr_{\mathbf{Y} \sim \mathcal{D}_3}[\mathcal{B}_{\text{det}} \text{ errs on } \mathbf{Y}] \geq 1/4 \). Let \( \mathcal{Z} \) be the output of \( \mathcal{B}_{\text{det}} \) when all the probes reveal label \( -1 \). If \( \mathcal{Z} = \mathcal{D}_1 \), \( \mathcal{B}_{\text{det}} \) can probe vertices \( 1/n \) and manifest how our argument differs from the known hardness proofs in active learning. For the other hand, if \( \mathcal{Z} = \mathcal{D}_2 \), then \( \mathcal{B}_{\text{det}} \) can probe at least half of the inputs from \( \{\mathbf{Y}^1, \ldots, \mathbf{Y}^{\frac{1}{6\epsilon}}\} \), meaning \( \Pr_{\mathbf{Y} \sim \mathcal{D}_3}[\mathcal{B}_{\text{det}} \text{ errs on } \mathbf{Y}] \geq 1/4 \).

Consider now a randomized algorithm \( \mathcal{B} \) that always has cost at most \( 1/(12\epsilon) \) on every \( \mathbf{Y} \in \mathcal{Y} \). \( \mathcal{B} \) 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 \( \Pr_{\mathbf{Y} \sim \mathcal{D}_3}[\mathcal{B}_{\text{det}} \text{ errs on } \mathbf{Y}] \geq 1/4 \) for every \( \mathcal{B}_{\text{det}} \), \( \Pr_{\mathbf{Y} \sim \mathcal{D}_3}[\mathcal{B} \text{ errs on } \mathbf{Y}] \) must be at least 1/4.

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

**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 \( \epsilon \) and \( n \) be chosen as in the distinguishing problem. To build a GCN input, we design \( G, X, Y, T, \) and \( \mathcal{H} \) as follows.

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

---

3Among all the intervals \( I_1, \ldots, I_{n/(6\epsilon)} \), \( \mathcal{B}_{\text{det}} \) 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 \( Y \Rightarrow Y^j \), 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
\[
\mathbb{H} := \{h^I \mid j \in [1/(6\epsilon)]\}.
\]
We will use $\mathbb{Y}$ 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 $i \in [n]$, $X_i^{(2)} = (n + 1)/2$.

**Proof.** See Appendix A.

Define
\[
\rho := \text{the only } j \in [1/(6\epsilon)] \text{ s.t. } (n + 1)/2 \in I_j.
\]
Proposition 6.3 implies that there are, effectively, only two different hypotheses on $X^{(2)}$:

• $h^I$, which classifies all vertices as 1;

• every other hypothesis in $\mathbb{H}$, which classifies all vertices as −1.

**Proposition 6.4.** If $Y$ is from $D_1$, $t^* = 1$ and $\nu^* = 2\epsilon$. Otherwise, $t^* = 2$ and $\nu^* = 0$. In both cases, any $h \in \mathbb{H}$ with $\text{er}(t^*, h) \leq \nu^* + \epsilon$ is an optimal hypothesis on $X^{(t^*)}$, i.e., $\text{er}(t^*, h) = \nu^*$.

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

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

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

**Proposition 6.5.** If $Y$ comes from $D_1$, $\text{er}^*(\nu^* + \epsilon) = 1/(12\epsilon)$. Otherwise, $\text{er}^*(\nu^* + \epsilon) = 1$.

**Proof.** See Appendix B.

Fix $\delta = 1/16$. Using Propositions 6.4 and 6.5, we can simplify the label complexity (11) of $A$ into
\[
\theta_{\nu^*}(\nu^* + \epsilon) \cdot \alpha
\]
for some $\alpha = \text{polylog}(\frac{1}{\epsilon})$. Choose $\epsilon$ small enough to make $\alpha < 1/(12\epsilon)$.

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

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

• Given any GCN input $Y \in \mathbb{Y}$, $A'$ requests at most $\alpha \cdot \theta_{\nu^*}(\nu^* + \epsilon)$ labels with probability $\geq 15/16$;

• Given any GCN input $Y \in \mathbb{Y}$, $A'$ outputs $(2, h^I)$ for some $j \neq j$ when all the label requests return −1;

• $\text{Pr}_{Y \sim D_3}[A' \text{ fails on } Y] \leq 1/16$.

**Proof.** $A$ can be regarded as a distribution over a family of deterministic algorithms $A_{\text{det}}$. We modify each $A_{\text{det}}$ as follows. Let $Z$ be the output of $A_{\text{det}}$ in the scenario where all the label requests return −1. If $Z = (2, h_j)$ for some $j \neq j$, leave $A_{\text{det}}$ unchanged. Otherwise, modify the output to $(2, h_j)$ for an arbitrary $j \neq j$. Let $A'_{\text{det}}$ be the altered algorithm. Since $A_{\text{det}}$ fails on $Y^{\neg}$ but $A'_{\text{det}}$ does not, $\text{Pr}_{Y \sim D_3}[A_{\text{det}} \text{ fails on } Y] \geq 1/2$ and $\text{Pr}_{Y \sim D_3}[A'_{\text{det}} \text{ fails on } Y] \leq 1/2$.

The above changes turn $A$ into an alternative randomized algorithm $A'$. Our modification does not affect how many labels are requested. Hence, $A'$ inherits from $A$ the property of ensuring cost at most $\theta_{\nu^*}(\nu^* + \epsilon) \cdot \alpha$ with probability at least $15/16$ on any $Y$. Furthermore, $\text{Pr}_{Y \sim D_3}[A' \text{ fails on } Y] \leq \text{Pr}_{Y \sim D_3}[A \text{ fails on } Y] \leq 1/16$.

$A'$ need not be verifiable because it guarantees only a small average failure probability over $D_3$, rather than a small probability on every $Y$. Nevertheless, it allows us to design an algorithm $B$ for the distinguishing problem:

**algorithm $B$**

1. run $A'$ up to $\alpha$ label requests
2. if $A'$ has not terminated or has found a label 1 then
3. return “$Y$ from $D_1$”
4. else return “$Y$ from $D_2$”

$B$ errs only in two events

1. $Y \neq Y^{\neg}$ yet $A'$ terminates within $\alpha$ label requests
2. all of which returned −1;
2. \( Y = Y^{\text{neg}} \) yet \( A' \) requests more than \( \alpha \) labels.

In event 1, by Lemma 6.6, \( A' \) returns \((2, h^2)\) for some \( j \neq \rho \) and thus fails on \( Y \). Hence, event 1 occurs with probability at most \( \Pr_{Y \sim D_3}[A' \text{ fails on } Y] \leq 1/16 \). On the other hand:

\[
\Pr[\text{event 2}] = \Pr_{D_3}[Y^{\text{neg}}] \cdot \Pr[A' \text{ has cost } > \alpha \text{ on } Y^{\text{neg}}] \\
\leq (1/2) \cdot (1/16) = 1/32
\]

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

Therefore, \( B \) satisfies \( \Pr_{Y \sim D_3}[8 \text{ errs on } Y] \leq 1/16 + 1/32 < 1/8 \) and makes at most \( \alpha < 1/(12\epsilon) \) probes with probability 1 on every \( Y \in \mathcal{Y} \). This contradicts Lemma 6.2 and completes the proof of Theorem 6.1.

Remark 6.7. When \( Y = Y^{\text{neg}} \), one can verify that \( \theta_{\text{all}}(\nu^* + \epsilon) = \Omega(1/\epsilon) \) and \( \theta_\ast(\nu^* + \epsilon) = 1 \). 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 \( T^r \) 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 \( T^r \) improvement. In fact, this is true even if \( \theta_{\text{all}}(\nu^* + \epsilon) \) is replaced with \( \sum_{i=1}^T \theta_i(\nu^* + \epsilon) \) 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 \( X_i^k = (1 + n)/2 \) for any \( i \in [n] \) and \( t \geq 2 \). By setting \( T = 1/\epsilon \), we thus create \( T - 1 \) identical instance spaces \( \mathcal{X}_2, \mathcal{X}_3, \ldots, \mathcal{X}_T \). Proposition 6.5 implies \( \sum_{i=1}^T \theta_i(\nu^* + \epsilon) = \Theta(1/\epsilon) \). Hence, under a \( T^r \) improvement, the resulting algorithm must request \( (1/\epsilon)^{1-c'} \) labels on every GCN input \( Y \in \mathcal{Y} \) 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 \( X \) and \( Y \) are arbitrarily given. This section will describe an alternative formulation where \( X \) and \( Y \) are generated in a stochastic manner.

The oracle first chooses a graph \( G := (V, E) \) whose matrix \( S \) 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 \( D_i \) over \( \mathbb{R}^d \times \mathbb{Y} \), where \( \mathbb{Y} := \{-1, 1\} \) and \( d \geq 1 \) is some integer. The oracle then decides on some integer \( k \geq 2 \), generates an \( n \times d \) matrix \( X^{(k)} \) by setting \( X_i^{(k)} = x_i \) for each \( i \in [n] \), and creates an \( n \times 1 \) matrix \( Y \) with \( Y_i = y_i \) for each \( i \). Finally, the oracle computes \( X = (S^{-1})^{k-1}X^{(k)} \) and releases \( X, G \), 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 \( \mathcal{H} \).

In the above, every vertex \( i \in [n] \) has a dedicated distribution \( D_i \). 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 \( V_1, V_2, \ldots, V_s \) for some \( s \geq 1 \). Vertices in \( V_j \) (for \( j \in [s] \)) share the same distribution, which we denote as \( D_{V_j} \).

The stochastic model allows us to study algorithms that may not work well on every \((X, Y)\), but achieve good overall performance on the inputs generated according to \( D_{V_1}, \ldots, D_{V_s} \). Next, we illustrate the idea on the special case \( s = 1 \) where all vertices obey the same distribution \( D \). If \( \nu := \inf_{h \in \mathcal{H}} \epsilon_{D}(h) \), a reasonable goal is to find a pair \((t, h) \in [T] \times \mathbb{H}\) with \( \epsilon_{V}(t, h) \leq \nu + \epsilon \). It is known (see Chapter 7 of (Hanneke, 2014)) that many choices of \( D \) and \( \mathbb{H} \) lead to constant-bounded disagreement coefficients \( \theta_{\mathbb{H}, D} \). 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 \( D \).

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 \( X \). Indeed, because of the matrix inversion performed by the oracle, \( X \) alone (i.e., without \( G \)) would be useless for learning. The algorithm essentially aims to recover \( X^{(k)} \), 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 \( A \) to be invertible. Many, but not all, graphs fulfill the requirement.
References


Appendix

A. Proof of Proposition 6.3

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

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

Hence,

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

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

$$X_i^{(2)} = \frac{1}{n} \sum_{j=1}^{n} 1[i,j] \cdot X_j^{(1)} = \frac{1}{n} \sum_{j=1}^{n} X_j^{(1)} = \frac{1}{n} \sum_{j=1}^{n} \frac{(1 + n) \cdot n}{2} = \frac{1 + n}{2}.$$  

B. Proof of Proposition 6.5

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

First, consider the case where $Y$ comes from $\mathcal{D}_1$, and assume, w.l.o.g. (due to symmetry), $Y = Y^1$. As explained in the proof of Proposition 6.4, $t^* = 1$, $\nu^* = 2\epsilon$, and $h^{I_1}$ is optimal on $X^{(1)}$. For any $j \neq 1$, $h^{I_1}(x) \neq h^{I_2}(x)$ if and only if $x \in I_1 \cup I_2$. Therefore,

$$\Pr_{(x,y) \sim \mathcal{U}_1} [h^{I_1}(x) \neq h^{I_2}(x)] = \Pr_{(x,y) \sim \mathcal{U}_1} [x \in I_1 \cup I_2] = 2\epsilon/n = 12\epsilon$$

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

$$\Pr_{U_1}[\text{DIS}(B_{\mathcal{U}_1}(h^{I_1}, r))] = \begin{cases} 0, & \text{if } r < 12\epsilon \\ 1, & \text{otherwise} \end{cases}$$

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

Next, we consider the case where $Y$ comes from $\mathcal{D}_2$, that is, $Y = Y_{\text{neg}}$. As explained in the proof of Proposition 6.4, $t^* = 2$, $\nu^* = 0$, and any $h^{I_2}$ with $j \neq \rho$ is optimal on $X^{(2)}$. Fix $k$ to an arbitrary value in $[1/(6\epsilon)]$ different from $\rho$. Clearly,

$$\Pr_{(x,y) \in \mathcal{D}_2} [h^{I_2}(x) \neq h^{I_2}(x)] = \begin{cases} 0, & \text{if } j \neq \rho \\ 1, & \text{if } j = \rho \end{cases}$$

When $r < 1$, $B_{\mathcal{D}_2}(h^{I_2}, r) = \mathbb{H} \backslash \{h^{I_2}\}$ and $\text{DIS}(B_{\mathcal{D}_2}(h^{I_2}, r)) = \emptyset$; when $r \geq 1$, $B_{\mathcal{D}_2}(h^{I_2}, r) = \mathbb{H}$ and $\text{DIS}(B_{\mathcal{D}_2}(h^{I_2}, r)) = \{(n + 1)/2\}$. We thus obtain

$$\Pr_{\mathcal{D}_2}[\text{DIS}(B_{\mathcal{D}_2}(h^{I_2}, r))] = \begin{cases} 0, & \text{if } r < 1 \\ 1, & \text{otherwise} \end{cases}$$

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

---

Cross-Space Active Learning on Graph Convolutional Networks
C. General GCNs

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

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

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

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

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

Re-define $G_t := \{t\} \times \mathbb{H}_t$ and $G_{\text{all}} := \bigcup_{t \in [T]} G_t$. For any $\mathcal{H} \subseteq G_{\text{all}}$, define $G_{\mathcal{H}}$ 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 $\theta_{\text{all}} (r) := \theta_{G_{\text{all}}, \mathcal{U}_t} (r)$. Our algorithms and analysis can be adapted to the above setup. Theorem 5.2 still holds after replacing $\lambda$ with $\max_{t=1}^T \lambda_t$ in (10). Similarly, Theorem 5.5 holds after replacing $\lambda$ with $\lambda_t$ in (11).