Leveraging Label Non-Uniformity for Node Classification in Graph Neural Networks

Feng Ji [1] **See Hian Lee** [1] **Hanyang Meng** [2] **Kai Zhao** [1] **Jielong Yang** [2] **Wee Peng Tay** [1]

Abstract

In node classification using graph neural networks (GNNs), a typical model generates logits for different class labels at each node. A softmax layer often outputs a label prediction based on the largest logit. We demonstrate that it is possible to infer hidden graph structural information from the dataset using these logits. We introduce the key notion of label non-uniformity, which is derived from the Wasserstein distance between the softmax distribution of the logits and the uniform distribution. We demonstrate that nodes with small label non-uniformity are harder to classify correctly. We theoretically analyze how the label non-uniformity varies across the graph, which provides insights into boosting the model performance: increasing training samples with high non-uniformity or dropping edges to reduce the maximal cut size of the node set of small non-uniformity. These mechanisms can be easily added to a base GNN model. Experimental results demonstrate that our approach improves the performance of many benchmark base models.

1. Introduction

Graph neural networks (GNNs) are neural networks that learn from graph-structured data (Defferrard et al., 2016). A problem of interest is the node classification problem. In a graph, one is given the labels of a subset of nodes and must predict the labels of remaining nodes using features associated with each node. Through many years of development, numerous GNN models have been proposed to tackle the node classification problem. Though many more recent approaches (e.g., Kang et al. (2021); Rusch et al. (2022); Song et al. (2022); Yang et al. (2021); Zhao et al. (2023); Zhao & Akoglu (2020); Zhu et al. (2020)) have sophisticated mechanisms, they are influenced by earlier models such as GCN (Defferrard et al., 2016) and GAT (Veličković et al., 2018). Quite a few important features of these primitive models are inherited by their up-to-date counterparts.

We briefly recall a few features of a model such as GCN that are most relevant. The basic idea of a GNN model is to generate an embedding of nodes in an appropriate geodesic metric space such as Euclidean space. Nodes are subsequently grouped into different classes using a union of hyperplanes. Such a strategy is realized by applying a message-passing algorithm. In each iteration of the algorithm, each node updates its feature vector by using a weighted average of the feature vectors from its neighbors. The generated embedding is then input to fully connected layers that output logits, which after normalization via softmax, are interpreted as probability weights of the label classes. The predicted label of each node is the class with the largest probability. Geometric information (e.g., hyperbolicity Gulcehre et al. (2019); Zhang et al. (2021); Zhu et al. (2020)) on the graph plays a fundamental role in the process as we need to utilize the local neighborhood of each node.

On the other hand, there is hidden graph structural information crucial to the success of the classification task. For example, we may be interested to know what are the boundary nodes (i.e., those nodes that have neighbors with different classes) between different label classes. In this work, we aim to retrieve such graph structural information using predicted logits from a model such as GCN. Therefore, in contrast to the procedure described in the previous paragraph, information retrieval is in the reverse direction.

The main tool we use is the notion of non-uniformity of the probability weights of the label classes derived from logits, inspired by the idea of distributional graph signals (Ji et al., 2023a;b). It measures the extent to which the probability distribution is not uniform. The insight is that for a node near class boundaries, during training, we have mixed contributions from different label classes as some of its neighbors belong to different classes. As message-passing has a smoothing effect (Oono & Suzuki, 2020), the resulting predicted logits should reflect the phenomenon

[1]School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore [2]School of Internet of Things Engineering, Jiangnan University, Wuxi, Jiangsu, China. Correspondence to: J. Yang <jyang022@e.ntu.edu.sg>, W. P. Tay <wptay@ntu.edu.sg>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

that several weights for different classes can have similar values. Therefore, non-uniformity may reveal hidden graph structural information associated with the embedding of different node classes in the ambient graph. In this paper, we theoretically study how the above-mentioned notion of non-uniformity provides us with graph structural knowledge (e.g., whether a node is close in graph distance to a class boundary). Based on the findings, we propose a simple multi-step model with independent modulo components, whose effectiveness is demonstrated with numerical experiments. Proofs of theoretical results are provided in Appendix B. Our main contributions are summarized as follows:

- We introduce the notion of label non-uniformity of probability weights associated with label classes. We demonstrate experimentally that nodes with small non-uniformity are harder to classify correctly.
- We analyze locations (with respect to class boundaries) of nodes with small label non-uniformity and provide insights to increase their non-uniformity.
- We propose two algorithms to boost the performance of a given base model: increasing training samples with high predicted label non-uniformity or dropping edges to reduce the maximal cut size of a node set of small non-uniformity. Experiments indicate that by adding our algorithm modules to a base model, its performance can be improved.

2. Label non-uniformity and node selection

Suppose $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is an undirected graph with $\mathcal{V}$ the vertex set and $\mathcal{E}$ the edge set. Let $d_{\mathcal{G}}$ be the metric with $d_{\mathcal{G}}(v, v')$ being the length of the shortest path between $v, v' \in \mathcal{V}$.

We consider the node classification problem. Recall that there is a finite set $\mathbb{S}$ of class labels. Each $v \in \mathcal{V}$ has a label $s \in \mathbb{S}$. For each $s \in \mathbb{S}$, let $\mathcal{V}_s$ be the set of nodes with class label s. We want to train a model based on a training set $\mathfrak{R}$ to predict the (unknown) labels of a test set $\mathfrak{T}$. To motivate the concepts we want to bring about, consider a variant of node classification as follows, which is interesting in its own right. We point out that the problem is a hypothetical thought experiment, but not the main subject of the paper. The purpose of the study is to investigate empirical evidence of the relations among easily classifiable nodes, the predicted logits, and graph topology.

Problem 1. *We use $\mathfrak{R}$ to train a GCN. For given $\alpha \leq 1$, one is required to choose a subset $\mathfrak{T}' \subset \mathfrak{T}$ of size $\alpha|\mathfrak{T}|$ so that the test accuracy of the trained model on $\mathfrak{T}'$ is maximized.*

As a special case, $\alpha = 1$ is the original node classification problem. The modified problem essentially asks one to find a subset of $\mathfrak{T}$ with a prescribed size, on which one can make an accurate prediction. Intuitively, suppose v has label s. Then it is more likely to predict correctly for v if it is close in distance to training nodes in $\mathcal{V}_s$. This prompts:

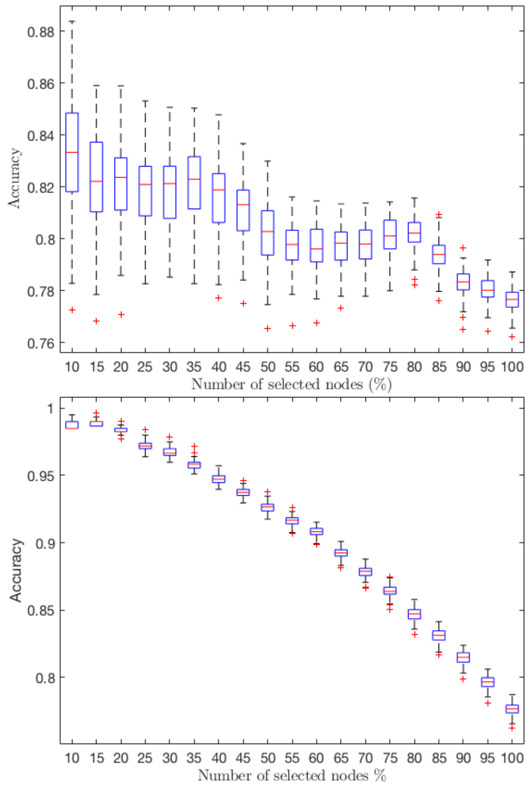

Figure 1. Accuracy for M1, M2 respectively on the Cora dataset.

Method 1 (M1: the geometric method)**.** *For each node $v \in \mathfrak{T}$, we assgin with it a pair of numbers $(f_1(v), f_2(v))$, where $f_1(v) = \min_{v' \in \mathfrak{R}} d_{\mathcal{G}}(v, v')$. For f_2, we first find the set $\arg\min_{v' \in \mathfrak{R}} d_{\mathcal{G}}(v, v')$. Then let $g(v)$ be the percentage of the largest label class in $\arg\min_{v' \in \mathfrak{R}} d_{\mathcal{G}}(v, v')$ and $f_2(v) = 1 - g(v)$. We rank $\mathfrak{T}$ based on the lexographic order of $\big(f_1(v), f_2(v)\big)$ and $\mathfrak{T}'$ is chosen following the ordering. The second number $f_2(v)$ is the tie-breaker for nodes with the same f_1 value.*

We run multiple experiments on the Cora dataset and the results are shown in Fig. 1 (top). We see that the performance pattern as α increases indeed supports the geometric intuition. However, we may push the intuition further to enhance the performance. We first introduce the notion of the boundary of a vertex set. Let $\mathcal{V}' \subset \mathcal{V}$ be a subset of vertices. Its *boundary* $\partial\mathcal{V}'$ is:

$$\partial\mathcal{V}' = \{v \in \mathcal{V}' \mid \exists\, v' \notin \mathcal{V}' \text{ s.t. } (v, v') \in \mathcal{E}\}.$$

Another insight on graph structure we may leverage is that when we want to determine the class label s of $v \in \mathcal{V}_s$, it is likely that we are less certain if v is closer to the boundary

$\partial\mathcal{V}_s$. This is because there are nodes with different class labels nearby (in $d_{\mathcal{G}}$).

An immediate challenge to exploit the above insight is that before knowing the true labels of all the nodes, $\partial\mathcal{V}_s$ is usually obscure to us. Hence, we need to find a way to estimate: *for any given v with label s, how far away it is from $\partial\mathcal{V}_s$ without knowing the label s.* For this, we make use of the concept of non-uniformity associated with label distribution.

Recall that from the logits of the base model trained on $\mathfrak{R}$, we may apply softmax to obtain a vector of numbers $\boldsymbol{\mu}_v$ with $\boldsymbol{\mu}_v(s) \in [0,1], s \in \mathbb{S}$ for each $v \in \mathcal{V}$. Moreover, $\mathbb{S}$ is a finite set and $\sum_{s\in\mathbb{S}} \boldsymbol{\mu}_v(s) = 1$, therefore $\boldsymbol{\mu}_v(s)$ can be interpreted as the probability weight of node v having label s. Following Ji et al. (2023b), the *label non-uniformity* at v is defined by

$$w(v) = \sum_{s\in\mathbb{S}} \left| \boldsymbol{\mu}_v(s) - \frac{1}{|\mathbb{S}|} \right|. \tag{1}$$

The notion is derived from the 2-Wasserstein distance (Ji et al., 2023b; Villani, 2009) between $\boldsymbol{\mu}_v$ and the uniform distribution. We justify in Appendix B. The function $w(\cdot)$ is the key player of our approach suggested by our title.

We propose to use $w(v)$ to measure whether v is close in distance to some class boundary. Nodes closer to class boundaries are expected to be harder to classify. This prompts the following approach to Problem 1.

Method 2 (M2: distribution non-uniformity)**.** *In training, we obtain a probability distribution $\boldsymbol{\mu}_v$ on $\mathbb{S}$ for each $v \in \mathcal{V}$ (in the last layer). We rank $\mathfrak{T}$ according to non-uniformity $w(v)$ and $\mathfrak{T}'$ is chosen following the ordering.*

The results for the approach M2 are also shown in Fig. 1 (bottom). It has a much better performance, which is also more consistent. In the following, we summarize observations from the experiments that eventually lead to our proposed GNN model.

(a) Using the non-uniformity $w(\cdot)$, we can reasonably identify nodes whose labels are correctly predicted.

(b) Comparing M1 and M2, we notice the correctly labeled nodes are not necessarily close to training nodes in $\mathfrak{R}$.

In the next section, we theoretically justify our graph structural intuition regarding the function of label non-uniformity $w(\cdot)$. Moreover, we propose a new GNN model based on the theoretical findings.

3. Label non-uniformity and graph structural information

As we speculate in the previous section, we want to analyze the non-uniformity $w(v)$ in view of the structural information of $\mathcal{G}$ in this section. Intuitively, if node v has large non-uniformity $w(v)$, its distribution weights $\boldsymbol{\mu}_v(s), s \in \mathbb{S}$ are either close to 0 or 1. As $w(\cdot)$ is computed from $\boldsymbol{\mu}_v$, to study $w(\cdot)$, we may instead analyze $\boldsymbol{\mu}_v$ in this section.

3.1. Flow of probability weights

For an overview, we first understand the flow of probability weights from a subset of nodes to another by analyzing the solution of an optimization problem. The study allows us to acquire geometric information such as the location of class boundaries using the weights. Furthermore, we analyze how to create a bottleneck near class boundaries to widen the difference in probability weights for nodes near class boundaries with different labels.

For the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, let $L_{\mathcal{G}}$ be the Laplacian of $\mathcal{G}$. To simplify the analysis, we study one class label at a time. Fix a class label $s \in \mathbb{S}$ and consider the *graph signal of probability weights* $(\boldsymbol{\mu}_v(s))_{v\in\mathcal{V}}$. We expect a model to make an accurate fitting on the training set $\mathfrak{R}$, therefore it is reasonable to assume that $\boldsymbol{\mu}_v(s) \approx 1, v \in \mathfrak{R} \cap \mathcal{V}_s$ and $\boldsymbol{\mu}_{v'}(s) \approx 0, v' \in \mathfrak{R} \cap \mathcal{V}_{s'}, s' \neq s$.

With this in mind, we assume that there are disjoint nonempty subsets $\mathcal{O}_0$ and $\mathcal{O}_1$ of $\mathcal{V}$ and an approximation $\mathbf{f}$ of $(\boldsymbol{\mu}_v(s))_{v\in\mathcal{V}}$ such that the graph signal $\mathbf{f}$ is observed at $\mathcal{O} = \mathcal{O}_0 \cup \mathcal{O}_1$.[1] Moreover, $\mathbf{f}_v = 0, v \in \mathcal{O}_0$ and $\mathbf{f}_{v'} = 1, v' \in \mathcal{O}_1$. The primary example is $\mathcal{O}_1 \subset \mathfrak{R} \cap \mathcal{V}_s$ and $\mathcal{O}_0 \subset \mathfrak{R} \backslash \mathcal{V}_s$. A *smooth interpolation* $\widetilde{\mathbf{f}}$ of $\mathbf{f}$ is

$$\widetilde{\mathbf{f}} = \mathop{\arg\min}_{\mathbf{f}':\mathbf{f}'_v=\mathbf{f}_v, v\in\mathcal{O}} \mathbf{f}'^{\intercal} L_{\mathcal{G}} \mathbf{f}'.$$

This is the well-studied Laplacian quadratic form, and its use in interpolating graph signals is justified in Ando & Zhang (2007); Narang et al. (2013); Shuman et al. (2013); Zhou et al. (2004); Zhu et al. (2003). We shall justify (in Appendix B) that if $\mathbf{f}$ is a good approximation of $(\boldsymbol{\mu}_v(s))_{v\in\mathcal{V}}$, then they have similar smooth interpolations. We use $\widetilde{\mathbf{f}}$ as a proxy of the true $\mathbf{f}$ and hence $(\boldsymbol{\mu}_v(s))_{v\in\mathcal{V}}$ on the entire graph, based on the assumption that $\mathbf{f}$ and $(\boldsymbol{\mu}_v(s))_{v\in\mathcal{V}}$ are smooth. Therefore, we have reduced the study of $(\boldsymbol{\mu}_v(s))_{v\in\mathcal{V}}$ to that of $\widetilde{\mathbf{f}}$, and the fundamental result is the following *averaging property*.

Lemma 1. *For every $v \notin \mathcal{O}$, let $\mathfrak{d}_v$ be its degree. We have*

$$\widetilde{\mathbf{f}}_v = \frac{1}{\mathfrak{d}_v} \sum_{(v,v')\in\mathcal{E}} \widetilde{\mathbf{f}}_{v'}. \tag{2}$$

As a consequence, we can view $\mathcal{O}_0$ and $\mathcal{O}_1$ analogous to the poles in a magnet, illustrated in Fig. 2 (a). To be more precise, for each $v \in \mathcal{V}$, define the *level-component* $\mathcal{C}_v$ of v w.r.t. $\widetilde{\mathbf{f}}$ to be the connected component of $\{v' \in \mathcal{V} \mid \widetilde{\mathbf{f}}_{v'} = \widetilde{\mathbf{f}}_v\}$ containing v. Then the following holds.

[1]To avoid cluttered notations, we omit s from the symbol $\mathbf{f}$.

Theorem 1. *For each $v \in \mathcal{V}$, there is a path $\mathcal{P} = \{v_0, \ldots, v_m\}$ such that the following holds:*

(a) $v_0 \in \mathcal{O}_0$ and $v_m \in \mathcal{O}_1$.

(b) $\widetilde{\mathbf{f}}$ is strictly increasing on $\mathcal{P}$: $\widetilde{\mathbf{f}}_{v_i} < \widetilde{\mathbf{f}}_{v_{i+1}}, 1 \leq i < m$.

(c) $\mathcal{P} \cap \mathcal{C}_v \neq \emptyset$.

The definition of a level-component is for the technical issue that there might be neighboring nodes with the same $\widetilde{\mathbf{f}}$ value. Disregarding this technicality, intuitively, Theorem 1 claims that $\widetilde{\mathbf{f}}$ values at nodes close to $\mathcal{O}_0$ should be close to 0 and gradually increase to 1 along the path $\mathcal{P}$. The following corollary of the theorem describes the signal pattern emanating from $\mathcal{O}_1$ as illustrated in Fig. 2(b).

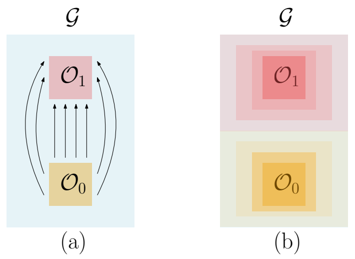

Figure 2. The Venn diagrams illustrate Theorem 1 and Corollary 1. In (a), the arrows are paths along which $\widetilde{\mathbf{f}}$ increases. In (b), if we color the graph according to $\widetilde{\mathbf{f}}$ value, then it should be layered as shown.

Corollary 1. *Suppose for any $u, v \notin \mathcal{O}_0 \cup \mathcal{O}_1$, we have $\widetilde{\mathbf{f}}_u \neq \widetilde{\mathbf{f}}_v$. For any $0 < r < 1$, define $\mathcal{V}_r = \{v \in \mathcal{V} \mid \widetilde{\mathbf{f}}_v \leq r\}$ and let $\mathcal{G}_{\mathcal{V}_r}$ be its induced subgraph. Denote the complement of $\mathcal{V}_r$ by $\mathcal{V}_r^c$. If $\mathcal{G}_{\mathcal{V}_{r_0}}$ is connected for some r_0, then so is $\mathcal{G}_{\mathcal{V}_r}$ for any $r \geq r_0$. Similarly, if $\mathcal{G}_{\mathcal{V}_{r_0}^c}$ is connected for some r_0, then so is $\mathcal{G}_{\mathcal{V}_r^c}$ for any $r \leq r_0$.*

Intuitively, the corollary describes the picture that if $\mathcal{G}_{\mathcal{V}_{r_0}}$ is connected, then $\mathcal{G}_{\mathcal{V}_r}$ must "grow" from $\mathcal{G}_{\mathcal{V}_{r_0}}$ for any $r \geq r_0$. We have seen an overall pattern of $\widetilde{\mathbf{f}}$ on $\mathcal{G}$. Next, we study more refined details of its values along a set boundary.

3.2. Weights near a boundary

To gain further insights, we describe another consequence of the averaging property in this subsection. For any subset $\mathcal{V}_0 \subset \mathcal{V}$, in addition to its boundary $\partial\mathcal{V}_0$, we define $\Gamma(\mathcal{V}_0) \subset \mathcal{E}$ to be the set of edges (v, v') with $v \in \mathcal{V}_0$ and $v' \in \mathcal{V}_1$, where $\mathcal{V}_1 = \mathcal{V} \backslash \mathcal{V}_0$. Moreover, we introduce the quantity

$$A(\widetilde{\mathbf{f}}, \mathcal{V}_0) = \sum_{\substack{v \in \partial\mathcal{V}_0, \\ (v,v') \in \Gamma(\mathcal{V}_0)}} \widetilde{\mathbf{f}}_v$$

as a weighted sum of $\widetilde{\mathbf{f}}_v, v \in \partial\mathcal{V}_0$.

Theorem 2. *Suppose $\mathcal{V}_0$ and $\mathcal{V}_1 = \mathcal{V} \backslash \mathcal{V}_0$ are disjoint subsets of $\mathcal{V}$ such that $\mathcal{O}_i$ is contained in the interior $\mathcal{V}_i \backslash \partial\mathcal{V}_i$ of $\mathcal{V}_i, i = 0, 1$. Assume that $0 < \widetilde{\mathbf{f}}_v < 1$ for $v \notin \mathcal{O}$ and let $a = \min_{v \notin \mathcal{O}} \widetilde{\mathbf{f}}_v$ and $b = \min_{v \notin \mathcal{O}}\{1 - \widetilde{\mathbf{f}}_v\}$. Then*

$$A(\widetilde{\mathbf{f}}, \mathcal{V}_0) \leq A(\widetilde{\mathbf{f}}, \mathcal{V}_1) - \max(b|\Gamma(\mathcal{O}_1)|, a|\Gamma(\mathcal{O}_0)|). \quad (3)$$

Theorem 2 claims that if the graph is partitioned into two parts $\mathcal{V}_0, \mathcal{V}_1$ (e.g., $\mathcal{V}_1$ consists of nodes with class label s, where s is as in the first paragraph of this section) containing $\mathcal{O}_0$ and $\mathcal{O}_1$ respectively, then the weighted sum of the values of $\widetilde{\mathbf{f}}$ along the boundary on the $\mathcal{O}_0$ side are smaller than those along the boundary on the $\mathcal{O}_1$ side. To find the average difference, it suffices to divide both sides of (3) by $|\Gamma(\mathcal{V}_0)|$, as both $A(\widetilde{\mathbf{f}}, \mathcal{V}_0)$ and $A(\widetilde{\mathbf{f}}, \mathcal{V}_1)$ have $|\Gamma(\mathcal{V}_0)|$ terms in their respective summation. Increasing the average $\widetilde{\mathbf{f}}$ difference is exactly what we are aiming for from Section 2: reduce the number of nodes with small non-uniformity.

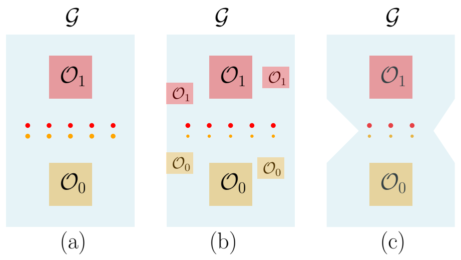

Figure 3. The Venn diagram illustrates that to differentiate nodes using $\widetilde{\mathbf{f}}$ value along the boundary, we may either increase the size of $\mathcal{O}_0$ and $\mathcal{O}_1$ as in (b) or create a bottleneck as in (c).

From Theorem 1 and Theorem 2, we have an overall picture:

(a) By Theorem 1, the values of $\widetilde{\mathbf{f}}$ gradually increase from nodes near $\mathcal{O}_0$ to those near $\mathcal{O}_1$. The boundary should be more likely to occur at nodes with $\widetilde{\mathbf{f}}$ further away from 0 and 1.

(b) By Theorem 2, to make $\widetilde{\mathbf{f}}$ closer to 0 or 1 on average even near class boundaries, we may try two options (illustrated in Fig. 3): (i) enlarge $\mathcal{O}_1$ and $\mathcal{O}_0$, or (ii) reduce the size of $\Gamma(\mathcal{V}_i), i = 0, 1$, i.e., we want to create a bottleneck. We discuss the second option further in the next subsection.

3.3. Graph bottleneck

The *Cheeger constant* or *edge expansion* (Mohar, 1989) $h(\mathcal{G})$ is usually used to measure the "bottleneck size" of a connected graph $\mathcal{G}$:

$$h(\mathcal{G}) = \min_{\mathcal{V}' \subset \mathcal{V}} \frac{|\Gamma(\mathcal{V}')|}{\min(|\mathcal{V}'|, |\mathcal{V}'^c|)}.$$

Other related concepts are the *max-cut size* $C(\mathcal{G})$ and *min-cut size* $c(\mathcal{G})$. Recall that a *cut* is $\Gamma(\mathcal{V}')$ for $\emptyset \neq \mathcal{V}' \subset \mathcal{V}$ such that $\mathcal{V}'^c \neq \emptyset$. Then $C(\mathcal{G})$ (resp. $c(\mathcal{G})$) is the largest (resp. smallest) among the size of every cut of $\mathcal{G}$.

For any $\mathcal{V}' \subset \mathcal{V}$, let $\mathcal{G}_{\mathcal{V}'}$ be its induced subgraph. We say a numerical invariant is *independent of* $\mathcal{G}_{\mathcal{V}'}$ if it does not change, even if the edge set of $\mathcal{G}_{\mathcal{V}'}$ is modified. As we shall see, the proposed method requires one to modify the edge set of $\mathcal{G}_{\mathcal{V}'}$. Therefore, this notion of independence is important for theoretical results.

Theorem 3. *For $\mathcal{V}' \subset \mathcal{V}$, suppose $\mathcal{G}_{\mathcal{V}'^c}$ has 2-connected components $\mathcal{U}_0, \mathcal{U}_1$, whose respective neighbors in $\mathcal{V}'$ are disjoint. Then there are constants $c_0, c_1 > 0$ independent of $\mathcal{G}_{\mathcal{V}'}$ such that: if $C(\mathcal{G}_{\mathcal{V}'}) < c_0$, then $h(\mathcal{G}) \leq C(\mathcal{G}_{\mathcal{V}'})/c_1$. Moreover, if $\mathcal{U} \subset \mathcal{V}$ realizes $h(\mathcal{G})$, then $\mathcal{U} \cap \mathcal{V}' \neq \emptyset$ and $\mathcal{U}^c \cap \mathcal{V}' \neq \emptyset$.*

The constants c_0 and c_1 are made explicit in the proof (cf. Appendix B). In particular, they are related to the cut size of $\mathcal{U}_0$ and $\mathcal{U}_1$, as expected. The appendix contains additional discussions and illustrations.

Theorem 3 essentially says that to create a bottleneck of the graph, we may reduce the cut size of a separating subgraph, i.e., a subgraph that separates the ambient graph into two connected components. This subgraph consists of nodes near class boundaries in our setup. This idea inspires Algorithm 2 in the next section.

4. Utilizing label non-uniformity

In Section 2, we use hypothetical experiments on Problem 1 to motivate the study of non-uniformity $w(\cdot)$ in Section 3. Our experiments on Problem 1 suggest that nodes with large non-uniformity are those that we can classify more accurately. Based on the theoretical insights derived in Section 3, we now propose a model that increases the number of nodes with large non-uniformity in the training set, which leads to a boost in the model performance in testing.

As alluded to in the last paragraph of Section 3.2, we can achieve our goal by (i) introducing more nodes in the training set with accurate labels (cf. Fig. 3(b)), or (ii) creating a bottleneck near class boundaries (cf. Fig. 3(c)). Both are associated with the non-uniformity function $w(\cdot)$ in (1). However, $w(\cdot)$ is used in different ways in these two approaches. In view of the experimental and theoretical findings in the previous sections, for (i), we prioritize nodes with large non-uniformity, while for (ii), we consider nodes with smaller non-uniformity as candidates for boundary nodes. The algorithms are presented in Algorithms 1 and 2.

To give some intuitions, in Algorithm 1, we use $w(\cdot)$ to select nodes with possibly high prediction accuracy. These are then included in the new training set $\mathfrak{R}'$ with their predicted test labels as the "ground-truth". Since the predicted test labels are mostly accurate, the larger training set is expected to lead to better model performance during testing. In Algorithm 2, we use $w(\cdot)$ to identify a set of nodes close in distance to class boundaries. Edge dropping aims to reduce the maximal cut size of this set, which may create a graph bottleneck in view of Section 3.3. On the new graph $\mathcal{G}'$, the

Algorithm 1 Supplement the training set $\mathfrak{R}$ using $w(\cdot)$

(a) Pick a base GNN model $\mathfrak{M}$ (e.g., GCN, GAT) and train the model $\mathfrak{M}$ to obtain the label class probability $\boldsymbol{\mu}_v(s)$, where $s \in \mathbb{S}$ are the label classes, for each node $v \in \mathcal{V}$. Compute $w(v) = \sum_{s \in \mathbb{S}} |\boldsymbol{\mu}_v(s) - \frac{1}{|\mathbb{S}|}|$.

(b) Order the test nodes v in $\mathfrak{T}$ in decreasing order according to $w(v)$.

(c) For a hyperparameter η_0, we form $\mathcal{V}'$ by taking η_0 fraction of nodes with the largest $w(\cdot)$ values, i.e., nodes higher in the ordering above.

(d) Nodes in $\mathcal{V}'$ are added to the training set $\mathfrak{R}$ with their *predicted test labels* in (a) to form $\mathfrak{R}'$, so that no ground-truth information is leaked.

(e) Retrain $\mathfrak{M}$ with $\mathfrak{R}'$

Algorithm 2 Edge dropping using $w(\cdot)$

(a) Same as Algorithm 1(a) and (b).

(b) For a hyperparameter η_1, we form $\mathcal{V}'$ by taking η_1 fraction of nodes with the smallest $w(\cdot)$ values, i.e., nodes lower in the ordering above.

(c) Construct $\mathcal{G}_{\mathcal{V}'}$ the induced subgraph of $\mathcal{V}'$ in $\mathcal{G}$. Let $\mathcal{T}_{\mathcal{V}'}$ be a spanning tree of $\mathcal{G}_{\mathcal{V}'}$ and $\mathcal{E}^c_{\mathcal{V}'}$ be the edges of $\mathcal{G}_{\mathcal{V}'}$ outside $\mathcal{T}_{\mathcal{V}'}$.

(d) For a second hyperparameter η_2, we randomly remove η_2 fractions of edges in $\mathcal{E}^c_{\mathcal{V}'}$ (cf. Fig. 4).

(e) Let $\mathcal{G}'$ be the resulting graph on $\mathcal{V}$ with the following edge sets: (i) those outside $\mathcal{G}_{\mathcal{V}'}$, (ii) those in $\mathcal{T}_{\mathcal{V}'}$ and (iii) remaining edges in $\mathcal{E}^c_{\mathcal{V}'}$ after dropping.

(f) Retrain $\mathfrak{M}$ on $\mathcal{G}'$.

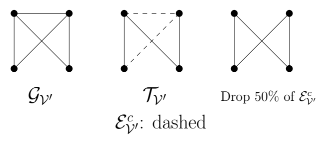
$\mathcal{G}_{\mathcal{V}'}$
$\mathcal{T}_{\mathcal{V}'}$
$\mathcal{E}^c_{\mathcal{V}'}$: dashed
Drop 50% of $\mathcal{E}^c_{\mathcal{V}'}$

Figure 4. The edge dropping step.

base model is expected to learn embeddings that are easier to distinguish among different classes.

We remark that each algorithm can be applied as a standalone module to any chosen base model, or combined together, which we do in our experiments. The combined approach means that we retrain $\mathfrak{M}$, the base model, with $\mathcal{G}'$ in step (e) of Algorithm 2 as the graph and $\mathfrak{R}'$ in step (d) of Algorithm 1 as the training set in the last step. An illustration is shown in Fig. 5.

For Algorithm 1, the final test accuracy is obtained by tallying the accuracy of all the nodes in $\mathfrak{T}$, i.e., the predicted labels of nodes in $\mathfrak{R}'\backslash\mathfrak{R}$ are from the initial step (a). In Algorithm 2(c), we propose not dropping any edge from a spanning tree so that the number of connected components remains the same after edge dropping, which also holds true for the combined approach.

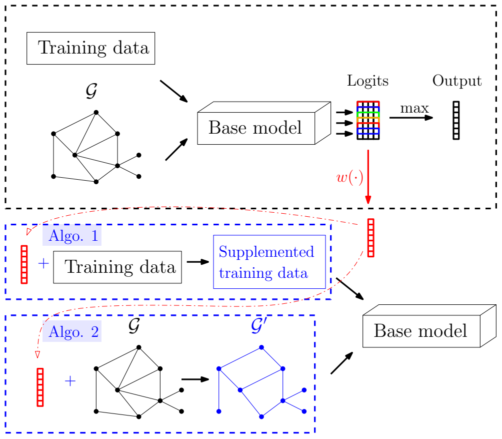

Figure 5. The figure is the scheme of the proposed model. We see that there are two separate modules (in the dashed blue boxes corresponding to the two algorithms. They can be applied either separately or jointly to the base model.

Related work

From Section 3, our approach is based on the study of geometric information, more precisely the graph structural information, associated with features and model output. This is not the sole work that emphasizes the importance of geometric knowledge regarding both graph topology and feature embedding. For example, one group of works (Chami et al., 2019; Gulcehre et al., 2019; Zhang et al., 2021) argue that each graph can be associated with a measure of hyperbolicity (Bridson & Haefliger, 1999), and therefore a graph with small hyperbolicity should be studied using hyperbolic geometry (Bachmann et al., 2019). There are also hybrid models combining both hyperbolic and Euclidean geometries (Zhu et al., 2020). Some works (Bodnar et al., 2021; Ebli et al., 2020; Lee et al., 2022) use the concept of simplicial complexes that generalizes graphs to account for higher-order relations among nodes. Our model is different in the sense that our objective is not to find a geometric space best suited for the dataset including both the graph and features. There are also works that tweak the graph topology. For example, DropEdge (Rong et al., 2020) proposes to randomly drop a fraction of edges in each iteration to alleviate the side-effects of oversmoothing Chen et al. (2020); NT & Maehara (2019); Oono & Suzuki (2020). A similar goal is pursued in Luo et al. (2021) by filtering out task-specific noisy edges. Our objective, however, is to infer hidden graph structural information associated with label class boundaries. Algorithm 1 is similar to self-training (Li et al., 2018) by using a part of the test nodes for training (cf. Appendix D). However, we select test nodes based on the newly introduced $w(\cdot)$, which is different from Li et al. (2018). Moreover, though Algorithm 1 and Algorithm 2 are different in nature, they are coherently derived from the same theoretical analysis.

5. Experiments

In this section, we evaluate the proposed model.

5.1. Node classification results

We perform experiments on the node classification problem. The datasets used are Cora, Citeseer, Pubmed, (Amazon) Photo, CS, Airport, and Disease (Chami et al., 2019; Fey & Lenssen, 2019; Namata et al., 2012; Sen et al., 2008; Shchur et al., 2018; Zhang & Chen, 2018). Our approach requires a base model. We use GCN (Defferrard et al., 2016), GAT (Veličković et al., 2018), DropEdge (Rong et al., 2020), GIL (Zhu et al., 2020), GraphCON (Rusch et al., 2022), MaskGAE (Li et al., 2022). We call our model the *graph neural network using* $w(\cdot)$ (wGNN), though the name does not explicitly refer to the base model being used. We single out discussions of heterophilic graphs in Section 5.2. More comparisons are given in Appendix D.

There are 3 hyperparameters: η_0 in Algorithm 1 and η_1, η_2 in Algorithm 2. They are tuned using a grid search (of 0.1 in stepsize) based on the accuracy of the validation set. We propose two ways to apply Algorithm 2. The more principled way is to apply the same base model for Algorithm 1 and Algorithm 2. On the other hand, for any dataset, we can also apply Algorithm 2 using a fixed model such as GCN once to generate $\mathcal{G}'$, which is stored for any other models on the same dataset. This is a compromise that is very efficient. We justify this procedure in Section 5.4.2. Other details regarding datasets and source code are in Appendix C. The results are shown in Table 1. In all cases, our model shows a performance improvement. We perform statistical tests and notice that the p-value of 37 among 42 (about 88%) comparisons is < 0.05, i.e., the improvement of wGNN is significant.

Table 1. Node classification result. Performance score averaged over ten runs. The best performance is boldfaced. MaskGAE uses a non-standard split for CS, and the results are to show improvements only and are not used for comparison with other benchmarks.

Method	CS	Photo	Cora	Citeseer	Pubmed	Airport	Disease
GCN	88.14 ± 0.42	90.66 ± 0.52	80.65 ± 0.49	71.23 ± 0.66	79.03 ± 0.38	85.08 ± 2.02	87.68 ± 3.67
wGNN	89.29 ± 0.14	92.35 ± 0.18	83.12 ± 0.31	$\mathbf{73.95} \pm 0.46$	80.48 ± 0.25	87.77 ± 1.57	89.02 ± 4.33
GAT	88.51 ± 0.73	90.36 ± 0.85	81.91 ± 0.48	70.21 ± 0.52	78.91 ± 0.42	91.23 ± 3.40	83.74 ± 2.31
wGNN	89.64 ± 0.38	91.82 ± 0.25	84.84 ± 0.50	72.85 ± 0.49	79.59 ± 0.28	$\mathbf{93.18} \pm 1.22$	86.42 ± 1.00
DropEdge	88.27 ± 0.43	90.49 ± 0.64	81.00 ± 0.55	70.72 ± 0.56	79.18 ± 0.33	86.80 ± 1.50	87.72 ± 2.60
wGNN	88.92 ± 0.27	92.09 ± 0.24	83.89 ± 0.36	73.13 ± 0.22	80.20 ± 0.21	87.17 ± 1.59	$\mathbf{92.32} \pm 0.43$
MaskGAE*	92.72 ± 0.11	91.55 ± 0.23	82.03 ± 0.76	70.10 ± 1.37	80.11 ± 0.51	70.56 ± 1.25	70.51 ± 3.57
wGNN	95.16 ± 0.68	$\mathbf{93.50} \pm 0.28$	82.85 ± 0.26	72.38 ± 0.97	$\mathbf{81.78} \pm 0.22$	85.62 ± 1.00	71.77 ± 4.26
GIL	88.69 ± 0.93	89.60 ± 1.30	79.65 ± 1.38	66.43 ± 1.56	77.18 ± 1.00	90.34 ± 1.29	89.96 ± 1.02
wGNN	91.44 ± 0.18	91.88 ± 0.32	83.16 ± 0.53	69.51 ± 0.45	80.73 ± 0.34	91.15 ± 1.05	92.01 ± 0.27
GraphCON	90.19 ± 0.70	90.04 ± 0.54	82.36 ± 0.84	70.80 ± 1.40	79.11 ± 1.78	57.32 ± 1.87	70.32 ± 4.45
wGNN	$\mathbf{93.03} \pm 0.64$	93.08 ± 0.37	$\mathbf{85.23} \pm 0.69$	71.40 ± 0.87	79.70 ± 1.07	63.82 ± 1.81	71.73 ± 3.18

5.2. Heterophilic graphs

Recall that for a dataset, the graph is heterophilic if there are many edges connecting nodes with different label classes. In this subsection, we study the performance of wGNN on datasets Texas and Chameleon (Zhu et al., 2021) with heterophilic graphs. In addition to the base models in Section 5.1, we also consider ACM-GCN (Luan et al., 2022), which is dedicated to addressing graph heterophily. We first show the results in Table 2. Similar to Section 5.1, each wGNN is paired with its base model. We see that wGNN is able to improve all the base models, including ACM-GCN.

We offer possible explanations for why wGNN also works for datasets with heterophilic graphs. In the extreme case that every edge connects a pair of nodes of different classes, then we have a k-partite graph, where k is the number of label classes. Therefore, a heterophilic graph is almost k-partite with very sparse connections within each of the k-components. In principle, our approach can also be helpful for heterophilic graphs. For example, Algorithm 2 reduces the connections among different components, and message passings in GNNs rely more on connections within each component. This is favorable for a shallow GNN model, where each node aggregates information only from close neighbors in message passing. Therefore, the predictive power of the model depends largely on the small neighborhood of each node.

The challenge for a heterophilic graph is that each node is likely to receive "noisy information" from neighbors belonging to different classes, due to a large number of edges connecting different types of nodes. For wGNN, though edge dropping cannot add connections between nodes of the same class, it can reduce connections between nodes of different classes with a high chance (due to heterophily). When this happens, during message passing, each node can potentially receive less "noisy information" from nodes of different classes. Consequently, the contribution from nodes of the same class increases. As illustrated in Fig. 6, v initially aggregates information from one node of class 1 and two nodes from class 2. After edge dropping in wGNN, v has only one node from class 2 as a neighbor. Heuristically, contributions (in fraction) from the same class 1 have increased by 50%.

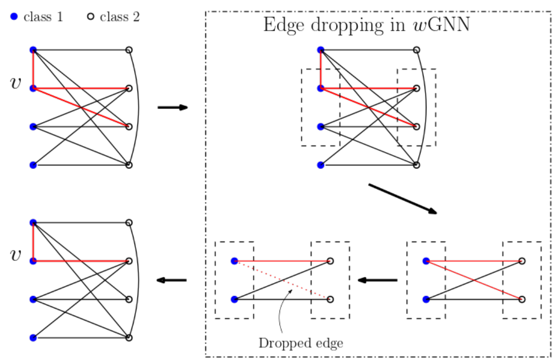

Figure 6. An illustration of the edge dropping step Algorithm 2.

In addition, if a base model handles heterophilic edges well, then wGNN (in particular Algorithm 1) allows us to have more reliable nodes for each class. In message passing, each node again receives more reliable information. This is possibly the reason why wGNN can also improve the specialized model ACM-GCN.

5.3. Ablation study

Our approach has two submodules Algorithm 1 and Algorithm 2, each of which can be applied as a stand-alone algorithm to the base model. In this subsection, we perform the ablation study to demonstrate that both algorithms contribute to the observed performance. For clarity, we use wGNN1 (resp. wGNN2) for the variant that applies Algo-

Table 2. Results for datasets with heterophilic graphs

	GCN	wGNN	GAT	wGNN	Dropedge	wGNN	GIL	wGNN	GraphCON	wGNN	ACM-GCN	wGNN
Texas	54.60 $\pm$6.47	58.11 $\pm$9.72	52.70 $\pm$8.38	55.94 $\pm$9.71	55.95 $\pm$6.25	58.11 $\pm$9.97	57.84 $\pm$5.44	61.89 $\pm$6.91	80.81 $\pm$4.12	85.95 $\pm$4.73	88.38 $\pm$3.21	**90.27** $\pm$4.05
Chameleon	63.62 $\pm$3.14	64.45 $\pm$2.40	64.25 $\pm$3.41	66.12 $\pm$2.58	63.57 $\pm$1.97	64.61 $\pm$2.39	64.41 $\pm$1.96	64.89 $\pm$1.04	50.77 $\pm$2.35	55.53 $\pm$2.26	65.92 $\pm$2.32	**80.79** $\pm$1.95

rithm 1 (resp. Algorithm 2) only. The Disease dataset is excluded from the study because the graph has a tree structure and Algorithm 2 is not involved. The results are shown in Table 3. Apart from seeing contributions from both algorithms, Algorithm 1 appears to have a stronger impact.

5.4. Further analysis

In this subsection, we present further analysis of wGNN.

5.4.1. Choice of parameters

There are 3 hyperparameters η_0, η_1, η_2 in the model. In this subsection, we study how they impact the model performance. It would be cumbersome to present the results for all possible combinations of η_0, η_1, η_2. Instead, we choose a few typical combinations to show the overall pattern. We choose $\eta_1 = \eta_2 \in [0.1 : 0.1 : 0.8]$ so that both light edge drop and heavy edge drop are considered. We let $\eta_0 \in \{0.2, 0.4, 0.6, 0.8\}$. GCN is used as the base model for the study, and heatmaps for the results (Cora and Citeseer datasets) are shown in Fig. 7. We see that the model performance is generally better for larger η_0, while the performance can drop if η_0 is large enough. Intuitively, if η_0 is too large, then Algorithm 1 may introduce more errors that may offset any benefits it brings about. On the other hand, we observe that good choices of η_1, η_2 that work for all η_0 happen at $\approx 0.6, \approx 0.5$ for Cora and Citeseer, respectively. This demonstrates the useful role played by Algorithm 2.

5.4.2. Model mismatch between Algorithm 1 and Algorithm 2

As discussed in Section 5.1, an experimental option is to use the same $\mathcal{G}'$ generated by Algorithm 2 from a single model such as GCN for any algorithm on the same dataset. Here, we analyze this by studying model mismatch between Algorithm 1 and Algorithm 2 using the following experiment.

We use GCN as the base model for Cora and Citeseer datasets. To apply Algorithm 2 to generate $\mathcal{G}'$, we compare using both GCN and GAT, while the latter accounts for the model mismatch. The hyperparameter η_0 is chosen to be 0.8 for Cora and 0.6 for Citeseer for better performance, as we have observed in Fig. 7. The parameters $\eta_1 = \eta_2$ vary from 0.1 to 0.8 as in Section 5.4.1.

The results are shown in Fig. 8. We see that for $\eta_1 = \eta_2 \leq 0.4$, when the accuracies are relatively high, the mismatch

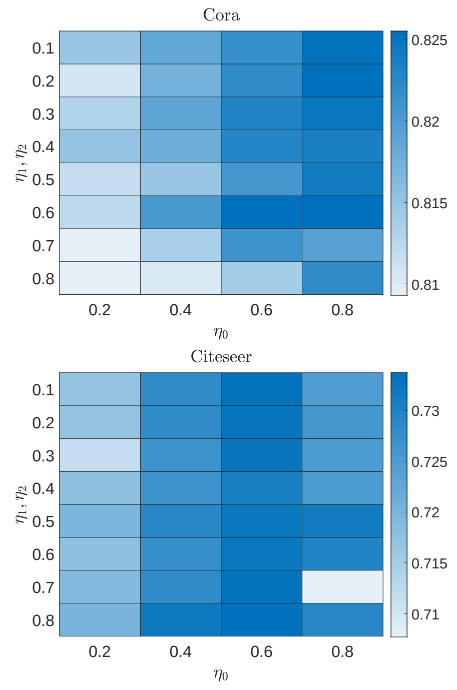

Figure 7. Performance against parameter choices.

does not have a large impact on the outcome. For large $\eta_1 = \eta_2$, there is an expected difference as heavier edge drop may cause a larger difference in graph topologies. However, for both experiments, the optimal accuracy does not suffer much from using a graph with a mismatched generating base model. Therefore, in practice, we may run Algorithm 2 and store the generated graph as an overhead, in resource-constrained applications.

5.4.3. Bad base models

Our approach relies on a base model. In practice, it is possible that a base model has poor performance, particularly at the initial investigation stage of a new dataset. In this subsection, we study the performance of our approach if such a base model is given. The candidates for base models are plain GAT with more layers. It is observed that the performance can be poor if more layers are added without fundamentally tweaking the GAT model structure, due to reasons such as oversmoothing and oversquashing (Topping et al., 2022). We consider variants of GAT with $L = 2$ to 8 layers, with deteriorating performance. We choose $\eta_0 = 0.4, 0.8$

Table 3. The ablation study: performance score averaged over ten runs. The best performance is boldfaced.

Method	CS	Photo	Cora	Citeseer	Pubmed	Airport
wGNN (GCN)	89.29 ± 0.14	$\mathbf{92.35} \pm 0.18$	83.12 ± 0.31	$\mathbf{73.95} \pm 0.46$	$\mathbf{80.48} \pm 0.25$	87.77 ± 1.57
wGNN1	89.29 ± 0.14	91.93 ± 0.25	82.88 ± 0.41	73.55 ± 0.46	80.36 ± 0.39	87.55 ± 1.89
wGNN2	88.54 ± 0.37	91.23 ± 0.30	81.10 ± 0.25	71.98 ± 0.52	79.19 ± 0.37	86.07 ± 1.52
wGNN (GAT)	$\mathbf{89.64} \pm 0.38$	91.82 ± 0.25	$\mathbf{84.84} \pm 0.50$	72.85 ± 0.49	79.59 ± 0.28	$\mathbf{93.18} \pm 1.22$
wGNN1	89.54 ± 0.33	91.50 ± 0.35	83.99 ± 0.29	72.35 ± 0.50	79.59 ± 0.28	92.11 ± 0.95
wGNN2	88.60 ± 0.66	89.97 ± 0.86	82.03 ± 0.63	70.55 ± 0.62	78.97 ± 0.32	91.80 ± 1.07

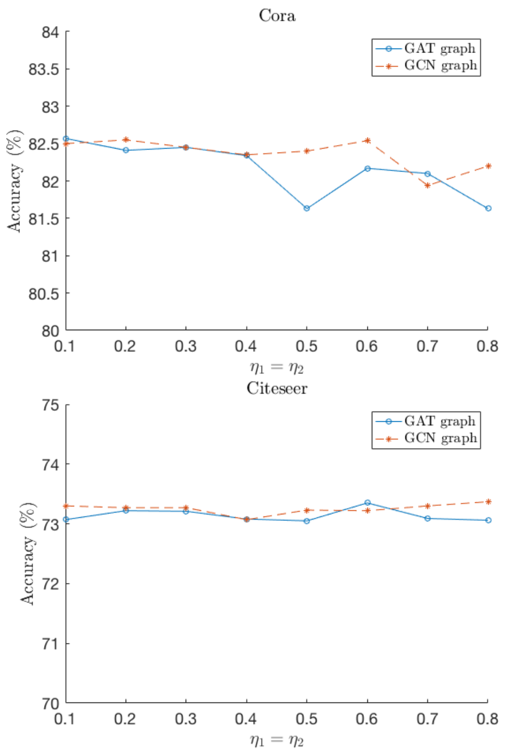

Figure 8. Performance for model mismatch between Algorithm 1 and Algorithm 2.

and run the experiments fixing $\eta_1 = \eta_2 = 0$. The results for the Cora dataset are shown in Fig. 9. Unless $L = 8$ when the base model accuracy is very low, our approach can reasonably improve the base model performance. For example, when $L = 7$, choosing $\eta_0 = 0.4$ improves the accuracy of the based model by $\approx 15\%$. Moreover, in this case, $\eta_0 = 0.4$ is much better than $\eta_0 = 0.8$, in contrast to $L = 2$ when $\eta_0 = 0.8$ has higher accuracy. This is because when the base model is highly inaccurate, then we are likely supplementing "datapoints with bad quality" in Algorithm 1 if η_0 is set to be large. This might also be the reason that when $L = 8$, applying our approach alone is insufficient to enhance the base model performance to a reasonable level. The findings also suggest $w(\cdot)$ can help with efficiently selecting useful nodes as long as the base model has reasonable performance.

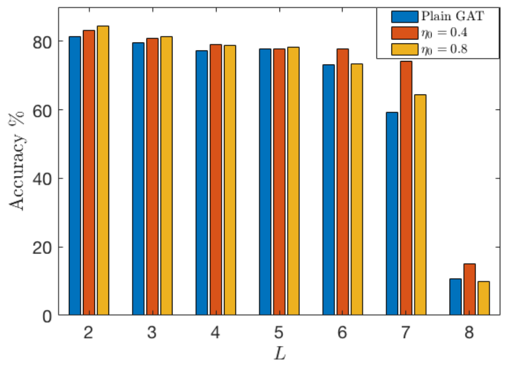

Figure 9. Performance based on GAT with $L = 2$ to 8 layers.

6. Conclusion

In this paper, we study the logits from the intermediate step of a typical GNN model, using a non-uniformity function. We gain hidden graph structural insights and propose a GNN model based on theoretical findings. Our model requires a base GNN model and we demonstrate with experiments to show that our approach can significantly improve the performance of base models in most cases.

Acknowledgements

F. Ji, S. H. Lee, K. Zhao and W. P. Tay are supported by the Singapore Ministry of Education Academic Research Fund Tier 2 grant MOE-T2EP20220-0002, and the National Research Foundation, Singapore and Infocomm Media Development Authority under its Future Communications Research and Development Programme.

H. Meng and J. Yang are supported in part by the Young Scientists Fund of the National Natural Science Foundation of China (No.62106082).

S. H. Lee is also supported by Shopee Pte. Ltd. under the Economic Development Board Industrial Postgraduate Programme (EDB IPP).

The computational work for this article was partially performed on resources of the National Supercomputing Centre, Singapore (https://www.nscc.sg).

References

Ando, R. and Zhang, T. Learning on graph with Laplacian regularization. In *Advances in Neural Information Processing Systems*, pp. 25–32, 2007.

Bachmann, G., Bécigneul, G., and Ganea, O.-E. Constant curvature graph convolutional networks. In *International Conference on Learning Representations*, 2019.

Bodnar, C., Frasca, F., Wang, Y., Otter, N., Montufar, G. F., Lió, P., and Bronstein, M. Weisfeiler and Lehman go topological: Message passing simplicial networks. In *International Conference on Machine Learning*, 2021.

Bridson, M. and Haefliger, A. *Metric Spaces of Non-positive Curvature*. Springer, 1999.

Chami, I., Ying, Z., Ré, C., and Leskovec, J. Hyperbolic graph convolutional neural networks. In *Advances in Neural Information Processing Systems*, pp. 4869–4880, 2019.

Chen, D., Lin, Y., Li, W., Li, P., Zhou, J., and Sun, X. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *AAAI Conference on Artificial Intelligence*, pp. 3438–3445, 2020.

Chen, D., Lin, Y., Zhao, G., Ren, X., Li, P., Zhou, J., and Sun, X. Topology-imbalance learning for semi-supervised node classification. In *Advances in Neural Information Processing Systems*, December 2021.

Defferrard, M., Bresson, X., and Vandergheynst, P. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, pp. 3844–3852, 2016.

Ebli, S., Defferrard, M., and Spreemann, G. Simplicial neural networks. In *Advances in Neural Information Processing Systems*, 2020.

Fey, M. and Lenssen, J. Fast graph representation learning with PyTorch Geometric. In *International Conference on Learning Representations*, 2019.

Gulcehre, C., Denil, M., Malinowski, M., Razavi, A., Pascanu, R., Hermann, K. M., Battaglia, P., Bapst, V., Raposo, D., and Santoro, A. Hyperbolic attention networks. In *International Conference on Learning Representations*, 2019.

Hwang, S. Cauchy's interlace theorem for eigenvalues of Hermitian matrices. *Amer. Math. Monthly*, 111:157–159, 2004.

Ji, F., Jian, X., and Tay, W. P. On distributional graph signals. *arXiv:2302.11104*, 2023a.

Ji, F., Lee, S., Zhao, K., Tay, W. P., and Yang, J. Distributional signals for node classification in graph neural networks. *arXiv:2304.03507*, 2023b.

Kang, Q., Song, Y., Ding, Q., and Tay, W. P. Stable neural ODE with Lyapunov-stable equilibrium points for defending against adversarial attacks. In *Advances in Neural Information Processing Systems*, virtual, December 2021.

Lee, S. H., Ji, F., and Tay, W. P. SGAT: Simplicial graph attention network. In *International Joint Conference on Artificial Intelligence*, 2022.

Li, J., Ruofan Wu, W. S., Chen, L., Tian, S., Zhu, L., Meng, C., Zheng, Z., and Wang, W. MaskGAE: Masked graph modeling meets graph autoencoders. *arXiv:2205.10053v1*, 2022.

Li, Q., Han, Z., and Wu, X. Deeper insights into graph convolutional networks for semi-supervised learning. In *AAAI Conference on Artificial Intelligence*, 2018.

Luan, S., Hua, C., Lu, Q., Zhu, J., Zhao, M., Zhang, S., Chang, X.-W., and Precup, D. Revisiting heterophily for graph neural networks. In *Advances in Neural Information Processing Systems*, New Orleans, USA, November 2022.

Luo, D., Cheng, W., Yu, W., Zong, B., Ni, J., Chen, H., and Zhang, X. Learning to drop: Robust graph neural network via topological denoising. In *ACM International Conference on Web Search and Data Mining*, pp. 779–7878888, March 2021.

Mohar, B. Isoperimetric numbers of graphs. *J. Comb. Theory. Ser. B*, 47(3), 1989.

Namata, G., London, B., Getoor, L., and Huang, B. Query-driven active surveying for collective classification. In *Mining and Learning with Graphs*, 2012.

Narang, S., Gadde, A., and Ortega, A. Signal processing techniques for interpolation in graph structured data. In *International Conference on Acoustics, Speech, and Signal Processing*, 2013.

NT, H. and Maehara, T. Revisiting graph neural networks: All we have is low-pass filters. *arXiv:1905.09550*, 2019.

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

Rong, Y., Huang, W., Xu, T., and Huang, J. DropEdge: Towards deep graph convolutional networks on node classification. In *International Conference on Learning Representations*, 2020.

Rusch, T., Chamberlain, B., Rowbottom, J., Mishra, S., and Bronstein, M. Graph-coupled oscillator networks. In *International Conference on Machine Learning*, 2022.

Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., and Eliassi-Rad, T. Collective classification in network data. *AI Mag.*, 29(3):93, 2008.

Shchur, O., Mumme, M., Bojchevski, A., and Günnemann, S. Pitfalls of graph neural network evaluation. In *Advances in Neural Information Processing Systems*, 2018.

Shuman, D. I., Narang, S. K., Frossard, P., Ortega, A., and Vandergheynst, P. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Process. Mag.*, 30(3):83–98, 2013.

Song, Y., Kang, Q., Wang, S., Zhao, K., and Tay, W. P. On the robustness of graph neural diffusion to topology perturbations. In *Advances in Neural Information Processing Systems*, New Orleans, USA, November 2022.

Topping, J., Giovanni, F. D., Chamberlain, B. P., Dong, X., and Bronstein, M. M. Understanding over-squashing and bottlenecks on graphs via curvature. In *International Conference on Learning Representations*, 2022.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks. In *International Conference on Learning Representations*, 2018.

Villani, C. *Optimal Transport, Old and New*. Springer, 2009.

Yang, H., Ma, K., and Cheng, J. Rethinking graph regularization for graph neural networks. In *AAAI Conference on Artificial Intelligence*, pp. 4573–4581, 2021.

Zhang, M. and Chen, Y. Link prediction based on graph neural networks. In *Advances in Neural Information Processing Systems*, pp. 5171–5181, 2018.

Zhang, Y., Wang, X., Shi, C., Liu, N., and Song, G. Lorentzian graph convolutional networks. In *World Wide Web Conference*, pp. 1249–1261, 2021.

Zhao, K., Kang, Q., Song, Y., She, R., Wang, S., and Tay, W. P. Graph neural convection-diffusion with heterophily. In *Proc. International Joint Conference on Artificial Intelligence*, Macao, China, Aug. 2023.

Zhao, L. and Akoglu, L. PairNorm: Tackling oversmoothing in GNNs. In *International Conference on Learning Representations*, 2020.

Zhou, D., Bousquet, O., Lal, T., Weston, J., and Schölkopf, B. Learning with local and global consistency. In *Advances in Neural Information Processing Systems*, pp. 321–328, 2004.

Zhu, J., Rossi, R., Rao, A., Mai, T., Lipka, N., Ahmed, N., and Koutra, D. Graph neural networks with heterophily. In *AAAI Conference on Artificial Intelligence*, 2021.

Zhu, S., Pan, S., Zhou, C., Wu, J., Cao, Y., and Wang, B. Graph geometry interaction learning. In *Advances in Neural Information Processing Systems*, pp. 7548–7558, 2020.

Zhu, X., Ghahramani, Z., and Lafferty, J. Semi-supervised learning using Gaussian fields and harmonic functions. In *International Conference on Machine Learning*, pp. 912–919, 2003.

A. List of notations

For easy reference, we list the most used notations in Table 4.

Table 4. List of notations

Graph, Vertex set, Edge set	$\mathcal{G}, \mathcal{V}, \mathcal{E}$
Nodes	v, v'
Training, Test sets	$\mathfrak{R}, \mathfrak{T}$
Class labels	$s, s' \in \mathbb{S}$
Distribution at v	$\boldsymbol{\mu}_v$
Label non-uniformity	$w(\cdot)$
Graph signals	$\mathbf{f}, \mathbf{f}', \tilde{\mathbf{f}}$
Graph Laplacian	$L_{\mathcal{G}}$
Subsets of nodes	$\mathcal{P}, \mathcal{O}, \mathcal{V}', \mathcal{U}$
Level component of v	$\mathcal{C}_v$
Boundary	∂
Induced subgraph	$\mathcal{G}_{\mathcal{V}'}$
Base GNN model	$\mathfrak{M}$

B. Theoretical results and discussions

In this appendix, we prove all the results of the paper and present other theoretical findings. We first define the Wasserstein distance (Villani, 2009) and justify (1).

Definition 1. *For two probability distribution μ_1, μ_2 with finite second momments on a metric space $\mathbb{M}$, the* 2-Wasserstein metric *$W(\mu_1, \mu_2)$ between μ_1, μ_2 is defined by*

$$W(\mu_1, \mu_2)^2 = \inf_{\gamma \in \Gamma(\mu_1, \mu_2)} \int d(x, y)^2 \, \mathrm{d}\gamma(x, y),$$

where $\Gamma(\mu_1, \mu_2)$ is the set of couplings *of μ_1, μ_2, i.e., the collection of probability measures on $\mathbb{M} \times \mathbb{M}$ whose marginals are μ_1 and μ_2, respectively.*

Suppose $\mathbb{S} = \{s_1, \ldots, s_m\}$ is a finite discrete set and d is the discrete metric on $\mathbb{S}$. For distributions μ, ν on $\mathbb{S}$, let $(\mu(s_i))_{1 \leq i \leq n}$ and $(\nu(s_i))_{1 \leq i \leq n}$ be their respective probability weights.

Lemma 2.

$$W(\mu, \nu)^2 = \frac{1}{2} \sum_{1 \leq i \leq m} |\mu(s_i) - \nu(s_i)|.$$

Proof. This is a known result and we present an elementary self-contained proof here. Let $\gamma = \big(\gamma(s_i, s_j)\big)_{1 \leq i,j \leq m}$ be in

$\Gamma(\mu,\nu)$. We have

$$\begin{aligned}
&\sum_{1\le i\le m}\sum_{1\le j\le m}\gamma(s_i,s_j)d(s_i,s_j)^2\\
&=\sum_{1\le i\le m}\sum_{1\le j\ne i\le m}\gamma(s_i,s_j)\\
&=\sum_{1\le i\le m}\left(\sum_{1\le j\le m}\gamma(s_i,s_j)-\gamma(s_i,s_i)\right) \qquad (4)\\
&=\sum_{1\le i\le m}(\mu(s_i)-\gamma(s_i,s_i))\\
&\ge\sum_{1\le i\le m}(\mu(s_i)-\min(\mu(s_i),\nu(s_i)).
\end{aligned}$$

As $W(\mu,\nu)^2$ is defined by taking the infimum of the left-hand side over all $\gamma\in\Gamma(\mu,\nu)$, we have $W(\mu,\nu)^2\ge\sum_{1\le i\le m}\big(\mu(s_i)-\min(\mu(s_i),\nu(s_i)\big)$. By the same argument, we also have $W(\mu,\nu)^2\ge\sum_{1\le i\le m}\big(\nu(s_i)-\min(\mu(s_i),\nu(s_i)\big)$. Summing up these two inequalities, we have

$$\begin{aligned}
2W(\mu,\nu)^2&\ge\sum_{1\le i\le m}\big(\mu(s_i)+\nu(s_i)-2\min(\mu(s_i),\nu(s_i)\big)\\
&=\sum_{1\le i\le m}|\mu(s_i)-\nu(s_i)|.
\end{aligned}$$

Therefore, to prove the lemma, it suffices to show that there is a γ such that $\gamma(s_i,s_i)=\min\big(\mu(s_i),\nu(s_i)\big)$. For this, we prove a slightly more general claim: if non-negative numbers $(x_i)_{1\le i\le m}$ and $(y_i)_{1\le i\le m}$ satisfy $\sum_{1\le i\le m}x_i=\sum_{1\le j\le m}y_i=a$, then there are non-negative $(z_{i,j})_{1\le i,j\le m}$ such that $\sum_{1\le j\le m}z_{i,j}=x_i, 1\le i\le m$, $\sum_{1\le i\le m}z_{i,j}=y_j, 1\le j\le m$, and $z_{i,i}=\min(x_i,y_i), 1\le i\le m$.

We prove this by induction on m. The case for $m=1$ is trivially true by taking $z_{1,1}=x_1=y_1$. For $m\ge 2$, without loss of generality, we assume that $x_1\ge y_1$ and $x_2\le y_2$. Then we choose $z_{1,1}=y_1$, $z_{2,2}=x_2$, $z_{1,j}=0, 1<j\le m$ and $z_{i,2}=0, 1\le i\ne 2\le m$. As a result, we form another two sequences of non-negative numbers $x_1-y_1,x_3,\ldots,x_m$ and $y_2-x_2,y_3,\ldots,y_m$ with both summing to $a-x_2-y_1$. By the induction hypothesis, we are able to find non-negative $(z'_{i,j})_{1\le i,j\le m-1}$ for the two new sequences of length $m-1$ each. It suffices to let $z_{i,j}=z'_{i-1,j-1}$ for $i>1$ or $j>2$ and $z_{i,1}=z'_{i-1,1}$ for $i>1$ (illustrated in Fig. 10). This proves the claim and hence the lemma. □

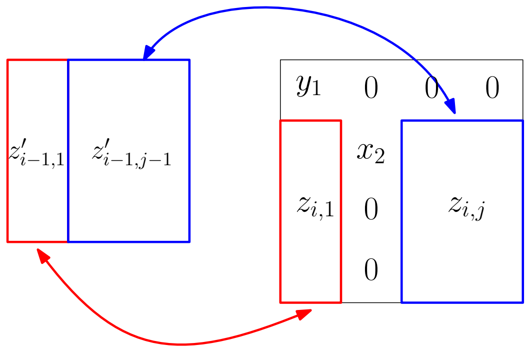

Figure 10. The relations between $z_{i,j}$ and $z'_{i,j}$.

Regarding non-uniformity defined by (1), it suffices to let μ be $\boldsymbol{\mu}_v$ and ν be the uniform distribution on $\mathbb{S}$. Therefore. $w(v)$ is nothing but $2W(\boldsymbol{\mu}_v,\nu)^2$ for the uniform distribution ν on $\mathbb{S}$.

We now move on to the results in Section 3 by first proving the averaging property Lemma 1.

Proof of Lemma 1. Taking partial derivatives of ${\mathbf{f}'}^\top L_{\mathcal{G}}\mathbf{f}'$ w.r.t. $\mathbf{f}'_v$ and setting it to be 0 yield the stated identity. $\square$

Next, we need the following result to justify using $\mathbf{f}$ and $(\boldsymbol{\mu}_v(s))_{v\in\mathcal{V}}$ have similar smooth interpolations, in view of the fact that $\mathbf{f}$ is an approximation of $(\boldsymbol{\mu}_v(s))_{v\in\mathcal{V}}$.

Lemma 3. *If $\mathcal{G}$ is connected and $\mathcal{O}$ is non-empty, then smooth interpolation $\mathbf{f}\mapsto\widetilde{\mathbf{f}}$ as a function $\mathbb{R}^n\to\mathbb{R}^n$ is well-defined and linear.*

Proof. To see this, setting the partial derivatives of ${\mathbf{f}'}^\top L_{\mathcal{G}}\mathbf{f}'$ to be 0 yields a linear relation between $\mathbf{g}$ and $\mathbf{h}$, where $\mathbf{g}$ (resp. $\mathbf{h}$) is the subvector of $\widetilde{\mathbf{f}}$ corresponds to nodes $\mathcal{V}\backslash\mathcal{O}$ (resp. $\mathcal{O}$). The relation takes the form $L_1\mathbf{g}=L_2\mathbf{h}$, where L_1 is the submatrix of $L_{\mathcal{G}}$ whose rows and columns are indexed by $\mathcal{V}\backslash\mathcal{O}$. It suffices to show that L_1 is invertible. As L_1 has a strictly smaller size, by the eigenvalue interlacing property (Hwang, 2004), we only need to show the case for $\mathcal{O}=\{v\}$ being a singleton. Let N_v be the neighbors of $v\in\mathcal{G}$. Notice that $L_1=D+L'$, where L' is the Laplacian of the induced graph $\mathcal{G}'$ on $\mathcal{V}\backslash\{v\}$ and D is the diagonal matrix with 1 at the entries indexed by N_v. If $L_1\mathbf{x}=0$, then $\mathbf{x}_u=0$ for $u\in N_v$ and so does $\mathbf{x}_{v'}$ for any v' in the connected component of $\mathcal{G}'$ containing u. As $\mathcal{G}$ is connected, this forces $\mathbf{x}$ to be the zero vector and the result is proved. $\square$

Recall that $\mathbf{f}_v$ is observed either 0 or 1 at $v\in\mathcal{O}$. An immediate consequence of the averaging property is the following.

Corollary 2. *For every $v\in\mathcal{V}$, $0\le\widetilde{\mathbf{f}}_v\le 1$.*

Proof. Suppose on the contrary that there is a v such that $\widetilde{\mathbf{f}}_v>1$. Without loss of generality, we assume that $\widetilde{\mathbf{f}}_v=\max\{\widetilde{\mathbf{f}}_{v'},v'\in\mathcal{V}\}$. By the averaging property, the signal values of all the neighbors of v are the same as $\widetilde{\mathbf{f}}_v$. As $\mathcal{G}$ is connected, the same holds for the value of $\widetilde{\mathbf{f}}$ at every node of $\mathcal{G}$, which is a contradiction. The same argument shows that $\widetilde{\mathbf{f}}_v\ge 0$ for $v\in\mathcal{V}$. $\square$

Suppose $\mathcal{V}'$ is a subset of $\mathcal{V}$. We define the *contraction* $\mathcal{G}_{/\mathcal{V}'}$ of $\mathcal{G}$ w.r.t. $\mathcal{V}'$ as follows. The vertex set of $\mathcal{G}_{/\mathcal{V}'}$ is $(\mathcal{V}\backslash\mathcal{V}')\cup\{u\}$. For $(v,v')\in\mathcal{E}$ and $v,v'\in\mathcal{V}\backslash\mathcal{V}'$, the edge (v,v') remains in $\mathcal{G}_{/\mathcal{V}'}$. For $(v,v')\in\mathcal{E}$ and $v\notin\mathcal{V}',v'\in\mathcal{V}'$, there is the edge (v,u) in $\mathcal{G}_{/\mathcal{V}'}$. Intuitively, we have replaced the entire vertex set V' be a single node u, and thus called a contraction.

Lemma 4. *$\mathcal{G}_{/\mathcal{V}'}$ is connected.*

Proof. Consider any two distinct nodes $v,v'\in\mathcal{G}_{/\mathcal{V}'}$. If $v\neq u,v'\neq u$, then they are connected by a path $\mathcal{P}$ in $\mathcal{G}$. In $\mathcal{G}_{/\mathcal{V}'}$, they are connected by the path $\mathcal{P}_{/(\mathcal{P}\cap V')}$.

Otherwise, without loss of generality, we assume $v=u$ and $v'\neq u$. Let $w(\cdot)$ be any node in $\mathcal{V}'$ and $\mathcal{P}$ be a path connecting $w(\cdot)$ and v' in $\mathcal{G}$. Then $\mathcal{P}_{/(\mathcal{P}\cap\mathcal{V}')}$ is a path connecting $v=u$ and v' in $\mathcal{G}_{/\mathcal{V}'}$. $\square$

We can now prove Theorem 1.

Proof. Let $\mathcal{G}'$ be the graph obtained from $\mathcal{G}$ be contracting all the level-components of $\mathcal{G}$, one at a time. The graph $\mathcal{G}'$ is connected by Lemma 4. Denote the vertices of $\mathcal{G}'$ by c_v where $\mathcal{C}_v$ is a level-component of $\mathcal{G}$. It is possible that $c_v=c_{v'}$ as long as $v'\in\mathcal{C}_v$. In addition, $\mathcal{G}$ is no longer simple as there can be multiple edges between the same pair of nodes. Each edge in $\mathcal{G}'$ is associated with a unique edge $(v,v')\in\mathcal{E}$. Therefore, for convenience, we remain using (v,v') for the edge in $\mathcal{G}'$.

The signal $\widetilde{\mathbf{f}}$ induces a signal $\widetilde{\mathbf{f}}'$ on $\mathcal{G}'$ by $\widetilde{\mathbf{f}}'_{c_v}=\widetilde{\mathbf{f}}_v$, this is well-defined as $\widetilde{\mathbf{f}}$ has the same value over a level-component. Moreover, if c_v and $c_{v'}$ are connected by an edge in $\mathcal{G}'$, then $\widetilde{\mathbf{f}}'_{c_v}\neq\widetilde{\mathbf{f}}_v$. We give $\mathcal{G}'$ an orientation by requiring (v,v') is oriented from v to v' if $\widetilde{\mathbf{f}}_v<\widetilde{\mathbf{f}}_{v'}$, or equivalently $\widetilde{\mathbf{f}'}_{c_v}<\widetilde{\mathbf{f}'}_{c_{v'}}$. By the averaging property, if (v,v') is an oriented edge, then there is an oriented edge (v',u') unless $v'\in\mathcal{O}_1$. Similarly, there is also an oriented edge (u,v) unless $v\in\mathcal{O}_0$. Pairs of directed edges (u,v) and (v,w) are called *matched*.

For $v\in\mathcal{V}$, we consider c_v of $\mathcal{G}'$. We can consecutively find matching edges in both directions until reaching $c_{v_0},v_0\in\mathcal{O}_0$ and $c_{v_m},v_m\in\mathcal{O}_1$ respectively. Connected the two paths at c_v, we obtain $\mathcal{P}$ passing through c_v such that consecutive edges are matched. By construction, $\mathcal{P}$ also gives a path (with the same edge labels) that satisfies (a)-(c) for v. $\square$

Theorem 1 is used to deduce Corollary 1 as follows.

Proof. We assume that $\mathcal{G}_{\mathcal{V}_{r_0}}$ is connected and $r \geq r_0$. We prove $\mathcal{G}_{\mathcal{V}_r}$ is connected by induction on $|\mathcal{V}_r| - |\mathcal{V}_{r_0}|$. If $|\mathcal{V}_r| = |\mathcal{V}_{r_0}|$, then there is nothing to show. Suppose there is an $r_0 \leq r' < r$ such that $|\mathcal{V}_{r'}| - |\mathcal{V}_{r_0}| = |\mathcal{V}_r| - |\mathcal{V}_{r_0}| - 1$. By the induction hypothesis, $\mathcal{G}_{\mathcal{V}_{r'}}$ is connected. Let $\mathcal{V}_r \backslash \mathcal{V}_{r'}$ be the singleton set containing v'.

As we assume that $\widetilde{\mathbf{f}}_u \neq \widetilde{\mathbf{f}}_v$ for $u, v \notin \mathcal{O}_0 \cup \mathcal{O}_1$, the level-component $\mathcal{C}_v = \{v\}$ for any $v \notin \mathcal{O}_0 \cup \mathcal{O}_1$. By Theorem 1, there is a path $\mathcal{P}$ with increasing $\widetilde{\mathbf{f}}$ value connecting $\mathcal{O}_0$ and $\mathcal{O}_1$ that passes through v'. The node before $\mathcal{P}$ must belong to $\mathcal{V}_{r'}$, and hence v' is connected to $\mathcal{G}_{\mathcal{V}_{r'}}$. As a result, $\mathcal{G}_{\mathcal{V}_r}$ is connected.

The proof regarding connectedness of $\mathcal{G}_{\mathcal{V}_r^c}$ is identical. $\square$

We proceed to prove Theorem 2 with the following observation.

Lemma 5. *Let $\mathcal{O}'$ be the immediate neighbors of $\mathcal{O}$ in $V\backslash\mathcal{O}$. For each $v' \in \mathcal{O}'$, let $d_{v'}$ be the number of nodes $v \in \mathcal{O}$ such that $(v, v') \in \mathcal{E}$. Then we have*

$$\sum_{v' \in \mathcal{O}'} d_{v'} \widetilde{\mathbf{f}}_{v'} = \sum_{v' \in \mathcal{O}'} \sum_{(v,v') \in \mathcal{E}, v \in \mathcal{O}} \mathbf{f}_v.$$

Proof. For each $v \in \mathcal{V}\backslash\mathcal{O}$, by Lemma 1, $\sum_{(v,v') \in \mathcal{E}} \widetilde{\mathbf{f}}_v = \sum_{(v,v') \in \mathcal{E}} \widetilde{\mathbf{f}}_{v'}$. If we sum up these identities, we have

$$\sum_{v \notin \mathcal{O}} \sum_{(v,v') \in \mathcal{E}} \widetilde{\mathbf{f}}_v = \sum_{v \notin \mathcal{O}} \sum_{(v,v') \in \mathcal{E}} \widetilde{\mathbf{f}}_{v'}. \tag{5}$$

In (5), if both v and v' are not in $\mathcal{O}$ and $(v, v') \in \mathcal{E}$, then $\widetilde{\mathbf{f}}_v$ and $\widetilde{\mathbf{f}}_{v'}$ occur in both sides of (5). Canceling these common terms, we have $\sum_{v' \in \mathcal{O}'} d_{v'} \widetilde{\mathbf{f}}_{v'} = \sum_{v' \in \mathcal{O}'} \sum_{(v,v') \in \mathcal{E}, v \in \mathcal{O}} \mathbf{f}_v$. $\square$

The proof does not require that $\mathcal{O}$ in Lemma 5 be exactly $\mathcal{O}_0 \cup \mathcal{O}_1$. The only requirement is that we interpolate $\widetilde{\mathbf{f}}$ by requiring that it must agree with observed $\mathbf{f}$ at $\mathcal{O}$, so that the averaging property can be applied.

Proof of Theorem 2. Let $\mathcal{G}_0$ (resp. $\mathcal{V}_1$) be the induced subgraph of $\mathcal{G}$ with nodes $\partial\mathcal{V}_1 \cup (V\backslash\mathcal{V}_1)$ (resp. $\partial\mathcal{V}_0 \cup (V\backslash\mathcal{V}_0)$). We apply Lemma 5 to $\partial\mathcal{V}_1 \cup \mathcal{O}_0$ in $\mathcal{G}_0$ and $\partial\mathcal{V}_0 \cup \mathcal{O}_1$ in $\mathcal{G}_1$ respectively to obtain the following identities:

$$A(\widetilde{\mathbf{f}}, \mathcal{V}_0) + \sum_{(v,v') \in \Gamma(\mathcal{O}_0), v \in \mathcal{O}_0} \widetilde{\mathbf{f}}_{v'} = A(\widetilde{\mathbf{f}}, \mathcal{V}_1), \text{ and} \tag{6}$$

$$A(\widetilde{\mathbf{f}}, \mathcal{V}_0) + \sum_{(v,v') \in \Gamma(\mathcal{O}_1), v \in \mathcal{O}_1} 1 = A(\widetilde{\mathbf{f}}, \mathcal{V}_1) + \sum_{(v,v') \in \Gamma(\mathcal{O}_1), v \in \mathcal{O}_1} \widetilde{\mathbf{f}}_{v'}. \tag{7}$$

As $0 < \widetilde{\mathbf{f}}_v < 1$ for $v \notin \mathcal{O}$, (6) yields that $A(\widetilde{\mathbf{f}}, \mathcal{V}_0) \leq A(\widetilde{\mathbf{f}}, \mathcal{V}_1) - a\Gamma(\mathcal{O}_0)$, while (7) yields that $A(\widetilde{\mathbf{f}}, \mathcal{V}_0) \leq A(\widetilde{\mathbf{f}}, \mathcal{V}_1) - b\Gamma(\mathcal{O}_1)$. By combining the two inequalities, we obtain the desired result. $\square$

It remains to discuss Theorem 3. We first prove the result.

Proof of Theorem 3. Let $\mathcal{U} \subset \mathcal{V}$ be a subset that realizes $h(\mathcal{G})$. If $\mathcal{U} \cap \mathcal{V}' = \emptyset$ or $\mathcal{U}^c \cap \mathcal{V}' = \emptyset$, we have the following possibilities.

Case 1: $\mathcal{U} = \mathcal{U}_0$ or $\mathcal{U} = \mathcal{U}_1$. For the former, we have

$$h(\mathcal{G}) = \frac{|\Gamma(\mathcal{U}_0)|}{\min(|\mathcal{U}_0|, |\mathcal{U}_1| + |\mathcal{V}'|)} \geq \frac{|\Gamma(\mathcal{U}_0)|}{|\mathcal{U}_0|}.$$

Similarly, if $\mathcal{U} = \mathcal{U}_1$, then $h(\mathcal{G}) \geq |\Gamma(\mathcal{U}_1)|/|\mathcal{U}_1|$.

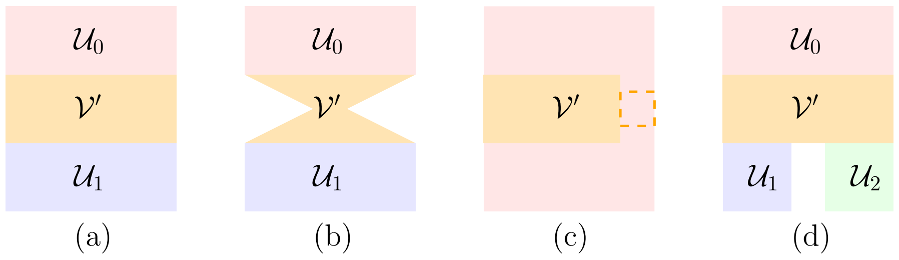

Figure 11. Illustration of Theorem 3 with Venn diagrams.

Case 2: $\mathcal{U}$ is a proper subset of $\mathcal{U}_0$. Let $\overline{\mathcal{U}}$ be the complement of $\mathcal{U}$ in $\mathcal{U}_0$. The set $\Gamma(\mathcal{U})$ consists of two parts $\mathcal{C}_1 \neq \emptyset$ and $\mathcal{C}_2 \subset \Gamma(\mathcal{U})$. The set $\mathcal{C}_1$ is a cut in the induced graph $\mathcal{G}_{\mathcal{U}_0}$. Therefore, we may estimate $h(\mathcal{G})$ as follows:

$$h(\mathcal{G}) = \frac{|\mathcal{C}_1| + |\mathcal{C}_2|}{\min(|\mathcal{U}|, |\overline{\mathcal{U}}| + |\mathcal{U}_1| + |\mathcal{V}'|)} \geq \frac{|\mathcal{C}_1|}{|\mathcal{U}|} \geq \frac{c(\mathcal{G}_{\mathcal{U}_0})}{|\mathcal{U}_0|}.$$

Case 3: $\mathcal{U}$ is a proper subset of $\mathcal{U}_0$. Similar to Case 2, we have

$$h(\mathcal{G}) \geq \frac{c(\mathcal{G}_{\mathcal{U}_1})}{|\mathcal{U}_1|}.$$

On the other hand, let $\mathcal{N}_0$ (resp. $\mathcal{N}_1$) be union $\mathcal{U}_0$ (resp. $\mathcal{U}_1$) and its the neighbors contained in $\mathcal{V}'$. By our assumptions, $\mathcal{N}_0 \cap \mathcal{N}_1 = \emptyset$. For $\mathcal{U} = \mathcal{N}_0$, notice that $\Gamma(\mathcal{U})$ is a cut of $\mathcal{G}_{\mathcal{V}'}$ and we estimate:

$$h(\mathcal{G}) \leq \frac{|\Gamma(\mathcal{U})|}{\min(|\mathcal{N}_0|, |\mathcal{N}_1|)} \leq \frac{C(\mathcal{G}_{\mathcal{V}'})}{\min(|\mathcal{N}_0|, |\mathcal{N}_1|)}.$$

Therefore, if $C(\mathcal{G}_{\mathcal{V}'})$ satifies

$$C(\mathcal{G}_{\mathcal{V}'}) < c_0 = \min(|\mathcal{N}_0|, |\mathcal{N}_1|) \cdot \min(\frac{|\Gamma(\mathcal{U}_0)|}{|\mathcal{U}_0|}, \frac{|\Gamma(\mathcal{U}_1)|}{|\mathcal{U}_1|}, \frac{c(\mathcal{G}_{\mathcal{U}_0})}{|\mathcal{U}_0|}, \frac{c(\mathcal{G}_{\mathcal{U}_1})}{|\mathcal{U}_1|})$$
$$\text{(remark: } c_0 \text{ is independent of } \mathcal{G}_{\mathcal{V}'} \text{),}$$

then Case 1 - Case 3 are impossible and we must have both $\mathcal{U} \cap \mathcal{V}' \neq \emptyset$ and $\mathcal{U}^c \cap \mathcal{V}' \neq \emptyset$. In this case, we have seen that $h(\mathcal{G}) \leq C(\mathcal{G}_{\mathcal{V}'})/c_1$, with $c_1 = \min(|\mathcal{N}_0|, |\mathcal{N}_1|)$. □

We illustrate Theorem 3 with Fig. 11. To create a bottleneck for $\mathcal{G}$ depicted in (a), we may create a bottleneck in $\mathcal{V}'$ according to the theorem as in (b). The conditions of the theorem may not always be satisfied. For example in (c), $\mathcal{V}'^c$ can be connected. In this case, if $\mathcal{V}'$ is large enough, then including a small number of additional nodes makes the conditions hold. It is also possible that $\mathcal{V}'^c$ has more than two components as in (d), we can then apply Theorem 3 repeatedly each time dealing with two components.

C. Dataset information and source code

In Table 5, we provide statistics of datasets used in the paper. For data splitting, we follow the cited references. More specifically, Cora, Citeseer, Pubmed, CS, and Photo use 20 training examples per class, with 500 validation samples and 1000 test samples. Disease and Airport use random 30/10/60, 70/15/15 splits respectively. Both Texas and Chameleon use 48/32/20 split.

In http://github.com/amblee0306/label-non-uniformity-gnn, we provide the source code and instructions to use the code.

Table 5. Dataset statistics

Dataset	Nodes	Edges	Classes	Features
Cora	2708	5429	7	1433
Citeseer	3327	4732	6	3703
Pubmed	19717	44338	3	500
Photo	7487	119043	8	745
CS	18333	81894	15	6805
Disease	1044	1043	2	1000
Airport	3188	18631	4	4
Texas	183	309	5	1703
Chameleon	2277	36101	4	2325

D. More comparisons

In this appendix, we make direct comparisons with a few more benchmarks: Renode (Chen et al., 2021), Self-train (Li et al., 2018), and PTDNet (Luo et al., 2021). As the implementations of Renode and PTDNet provided by the respective authors are both based on GCN, we also use the **GCN version** of wGNN. Therefore unlike Section 5.1, wGNN and its variants in this appendix are not based on the models they compare with.

Self-training shares some common features with Algorithm 1 by introducing test nodes to the training set, we compare it with wGNN1 (cf. Section 5.3) that does not use Algorithm 2. Similarly, PTDNet involves edge dropping (for a different purpose), and we compare it with wGNN2 without using Algorithm 1. As Algorithm 2 (almost) does not play a role if the graph is a tree or very sparse, the comparisons between wGNN2 and PTDNet do not consider Diseases and Texas datasets.

Comparison results are shown in Table 6. We see that wGNN (resp. wGNN1) has an overall better performance than Renode (resp. Self-training). On the other hand, each of wGNN2 and PTDNet has its own advantages over certain datasets. For the densest datasets Airport, Chameleon, and Photo, wGNN2 has much better performance (each with $> 10\%$ improvement). For sparser graphs, Algorithm 2 does not permit dropping too many edges as we still need to maintain a spanning tree among nodes selected by $w(\cdot)$, and hence it is less impactful.

Table 6. Comparisons with Renode, Self-train and PTDNet

	Cora	Citeseer	Pubmed	CS	Photo	Airport	Disease	Texas	Chameleon
wGNN	83.12 ± 0.31	73.95 ± 0.46	80.48 ± 0.25	89.29 ± 0.14	92.35 ± 0.18	87.77 ± 1.57	89.02 ± 4.33	58.11 ± 9.72	64.45 ± 2.40
Renode	81.28 ± 0.75	69.54 ± 0.79	79.62 ± 0.38	90.08 ± 0.56	89.25 ± 1.20	76.40 ± 3.73	83.35 ± 1.26	58.11 ± 4.72	52.98 ± 2.84
wGNN1	82.88 ± 0.41	73.55 ± 0.46	80.36 ± 0.39	89.29 ± 0.14	91.93 ± 0.25	87.55 ± 1.89	89.02 ± 4.33	55.41 ± 7.35	63.95 ± 1.94
Self-training	82.27 ± 0.33	73.24 ± 0.44	80.32 ± 0.18	88.92 ± 0.19	90.62 ± 0.43	85.69 ± 0.89	87.18 ± 1.18	54.14 ± 7.45	63.97 ± 2.33
wGNN2	81.10 ± 0.25	71.98 ± 0.52	79.19 ± 0.37	88.54 ± 0.37	91.23 ± 0.30	86.07 ± 1.52	–	–	64.28 ± 1.94
PTDNet	82.80 ± 2.60	72.70 ± 1.80	79.80 ± 2.40	90.37 ± 0.17	80.11 ± 0.44	64.28 ± 1.94	–	–	50.35 ± 1.93