공학석사 학위논문

Positive-Unlabeled node classification under sparse labels with Mixup based GNNs

믹스업 기반의 그래프 신경망을 사용한 희소 라벨에서의 양성-비라벨 노드 분류

2023년 2월

서울대학교 대학원

컴퓨터공학부

홍 한 결

Positive-Unlabeled node classification under sparse labels with Mixup based GNNs

지도 교수 권 태 경

이 논문을 공학석사 학위논문으로 제출함
2022년 11월

서울대학교 대학원
컴퓨터공학부
홍 한 결

홍한결의 공학석사 학위논문을 인준함
2022년 11월

위 원 장 이 광 근 (인)

부위원장 권 태 경 (인)

위 원 허 충 길 (인)

Abstract

Positive-Unlabeled node classification under sparse labels with Mixup based GNNs

Hangyeol Hong

Department of Computer Science and Engineering

The Graduate School

Seoul National University

Recently, semi-supervised learning has gained substantial interest due to the sparsity of real-world datasets. This is general to graph-structured data, where few labeled nodes are available during training. In this paper, we integrate Positive-Unlabeled (PU) learning [1, 2, 3] with Graph Neural Networks (GNNs) to address binary node classification utilizing plentiful unlabeled nodes. Specifically, PU learning aims to excavate potential positive and negative interactions between nodes by using only positive labeled nodes and unlabeled nodes. Here, we propose a novel framework named Positive-Unlabeled node classification with Mixup-based GNNs (PUM-GNN). It addresses limited labeled cases and gives supervision to the PU learning using Mixup regularization [4, 5]. Mixup is a promising study in image data augmentation but has not been studied much in GNNs because of the irregularity of the graph. We use Mixup in the

embedding space to not only augment data but also transform the marginal pseudo-negative instances into partially positive augmented instances, and improve the imprecise supervision within unlabeled instances. We conduct experiments using various positive label ratios and found that PUM-GNN not only reduces over-fitting but also outperforms state-of-the-art methods under sparse labels.

Keyword: Semi-supervised learning, Graph Neural Networks, Regularization, Data augmentation, Node classification
Student Number: 2021-26464

Table of Contents

List of Figures

List of Tables

Chapter 1. Introduction

Graph Neural Networks (GNNs) have shown great advantages in graph-structured data. By mapping non-regular data into nodes and edges, GNNs tackle various challenges using graph topology. For example, molecular prediction, anomaly detection, recommender system, and link prediction is the common problem that GNNs can solve. However, due to the labeling cost, real-world data has low data sparsity. For example, in the recommender system, users cannot rate all vast items, so we're only given a few rated items from users and predict the relevance between the majority of unlabeled items and users. According to this property of data, most existing GNNs approaches are based on semi-supervised learning that utilizes several unlabeled nodes, including GCN [6], GAT [7], and so on. Learning from a limited amount of labeled data is a longstanding challenge in modern machine learning [8]. To tackle these issues, Positive-Unlabeled (PU) learning [9], a kind of semi-supervised learning, has arisen. It deals with binary classification where only P and U data are available.

In this paper, we study graph-based PU learning, which is a common situation in the real world. Suppose there are 100 fraud users out of 1,000 users. Then is it true that the remaining 900 users are normal users? The answer is *NO*. 900 users are unlabeled, which can be potentially positive or negative, and cannot

be affirmed as negative. Therefore, our goal is *1) to propose a GNN framework under PU labels to achieve good performance where there are many unlabeled nodes.* Instead of taking the unlabeled samples as negative, PU learning learns a binary classifier in the absence of explicitly labeled negative samples. Therefore, it can be helpful in many real-world where only one type of label is often abundant.

There are two categories of PU learning, which differ in how unlabeled (U) data is handled. As a first category, reliable negative mining approaches choose data close to negative (N) from U and then perform supervised learning using P and N [10]. On the other hand, the second category regards U as weighted N [11, 31]. There are some limitations of existing PU methods. Since negative data is selected heuristically in the first category, the performance heavily relies on the heuristics. And the second one needs to determine the weight, which is computationally expensive. To avoid tuning the weights, cost-sensitive PU learning [2, 3, 30] comes out as a subcategory of the second category. Cost-sensitive PU learning utilizes risk estimators which can reduce bias for PU classification. Specifically, [2] proposes an unbiased risk estimator, and Kiryo [3] proposes a non-negative risk estimator which can reduce over-fitting.

However, since these methods classify U as N, they may suffer from a negative-prediction preference. As the decision boundary is biased toward positive (P), the number of incorrectly classified P increases. To deal with this issue, we consider a data

augmentation-based regularizer, Mixup [4]. Mixup linearly interpolates randomly selected samples and their label pair and adds it to the training set. Flexible decision boundaries can be obtained by expanding the training distribution, and it can also generalize the model by reducing over-fitting. In this paper, we focus on marginal pseudo-negative instances [12], whose ground truth is positive but predicted by negative. We supervise those samples to be positive by mixing between them rather than random sampling. Therefore, our other goal is *2) to give supervision to those misclassified P while augmenting data.*

Putting these goals together, we propose Positive-Unlabeled node classification with Mixup-based GNNs, namely **PUM-GNN**. PUM-GNN transforms the marginal pseudo-negative instances into augmented instances that are partially positive, so that learned boundary moves to fully supervised ones.

The main contributions of this paper are summarized below:

- We propose a novel GNN framework called PUM-GNN and formulate the node classification under PU labels, which can address a label sparsity issue.
- We apply Mixup, which is commonly used in image classification, to the graph domain, and propose a new Mixup strategy that can both augment data and refine supervision.
- We experiment with various label ratios, and the result shows that PUM-GNN can enhance the performance of PU classification and can beat PN classification.

Chapter 2. Related Work

2.1 Graph-based PU learning

There have been several studies that apply PU learning to GNNs. Wu et al [13] study positive-unlabeled node classification task for the first time. They propose LSDAN (long-short distance aggregated networks), which addresses the limitations of GAT and captures long-distance relationships with a target node using a higher-order adjacency matrix. It also employs two PU risk estimators and the utilizes outputs of the model to achieve optimized PU graph learning.

Zhou et al propose PURE [14], a novel approach for positive-unlabeled recommender systems under GAN [15] framework. The generator of PURE continuously generates a fake item(user) that might be relevant to the user(item) to fool the discriminator. PURE trains an unbiased positive-unlabeled discriminator which assigns low scores to items/users that have not been rated.

Yoo et al address more practical graph-based PU learning and propose GRAB [16]. It considers PU classification without class prior π_P, which is not given in the real-world. GRAB models a graph as a Markov network and iteratively estimates π_P using the graph topology and the node feature vector.

2.2 Mixup for graph data

Mixup [4] was first proposed in the field of computer vision and has been actively studied until now. Despite the advantages of data augmentation, Mixup has not been applied much to graph data because it has irregular topology and connectivity. Nevertheless, there have been novel attempts at this topic.

Wang et al [17] propose a novel Mixup method for graph data. To deal with the irregularity and non-linearity of graph data, they propose the two-branch Mixup graph convolution to mix nodes' features and topology.

SubMix [18] generalizes CutMix [19] to the graph domain, which creates a new graph while preserving the core structure of the graph. It considers the adjacency matrix as an image and pastes a random patch of the matrix which has been cut to another matrix.

In the recommendation system problem, a fundamental challenge is to find a negative signal from implicit feedback. To address this challenge, MixGCF [20] aims to create hard negative items through embedding mixup, rather than sampling raw negatives from data.

Chapter 3. Problem Definition

In this paper, we study a node classification task using a set of positive and unlabeled samples. Let $\mathrm{x} \in \mathbb{R}^{\mathrm{d}}, \mathrm{d} \in \mathbb{N}$ be the input variable and $\mathrm{y} \in \{+1, 0\}$ be the label ($+1$: positive, 0: negative), and $p(x, y)$ be the true underlying joint distribution of (x, y). PU dataset $\mathcal{X}_{PU}$ consists of two independent datasets uniformly sampled from $p(x, y)$: an incomplete set of $\mathcal{X}_P$ of n_p data-points with positive label and a set of $\mathcal{X}_U$ of n_u unlabeled samples:

$$\mathcal{X}_P = \{x_i^P\}_{i=1}^{n_p} \sim p(x \mid y = 1); \quad \mathcal{X}_U = \{x_i^U\}_{i=1}^{n_u} \sim p(x); \quad \mathcal{X}_{PU} = \mathcal{X}_P \cup \mathcal{X}_U.$$

Let $M = n_p + n_u$ be the size of $\mathcal{X}_{PU}$. In the real world, each user would rate a small number of nodes on average. Therefore, we set the class prior probability $\pi_P = \mathrm{p}(y = +1)$ to fall within the range of $[0.2, 0.4]$ which means the true distribution of the positive samples is sparse.

Assumption 1 (Known class prior).

Throughout the paper, we assume π_P is either known or can be efficiently estimated from $\mathcal{X}_{PU}$ via mixture proportion estimation algorithm [21, 22, 23, 24]. This assumption is dominant in most cost-sensitive PU learning algorithms [1, 2, 3, 14].

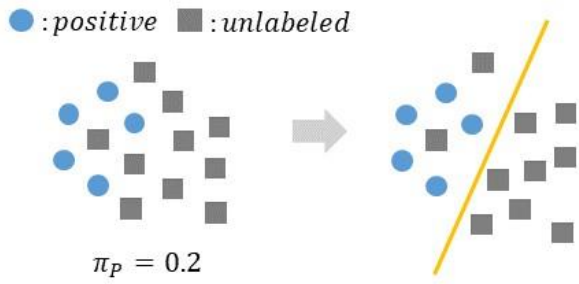

Figure 3.1: Positive–Unlabeled classification with $\boldsymbol{\pi_P}$.

We can estimate the number of positive samples in a set of unlabeled samples using π_P. For example, as shown in Figure 3.1, we can see that $\pi_P * |\mathcal{X}_U| = 0.2 * 10 = 2$ samples in $\mathcal{X}_U$ are P. Based on this information, PU learning task is thus to train a classifier f using only PU observations and knowledge of π_P.

Definition (Node classification problem).

Given: set of nodes $\mathcal{N} = \{n_1, n_2, \dots, n_M\}$, set of node features $\mathcal{F} = \{x_1, x_2, \dots, x_M\}$, label set of nodes $\mathcal{L} = \{+1, -1, \dots, +1\}$ ($+1$: positive, -1: unlabeled), class prior probability π_P, adjacency matrix $\boldsymbol{A}$.

Output: the estimated label of the unlabeled nodes

Chapter 4. Methodology

In this section, we introduce the proposed PUM−GNN for graph−based PU learning. After running GNNs, we apply Mixup [4] in the embedding space to regularize the model and produce partially positive nodes to add more supervision. Then we optimize our PU loss function. Figure 4.1 shows the overall process of PUM−GNN.

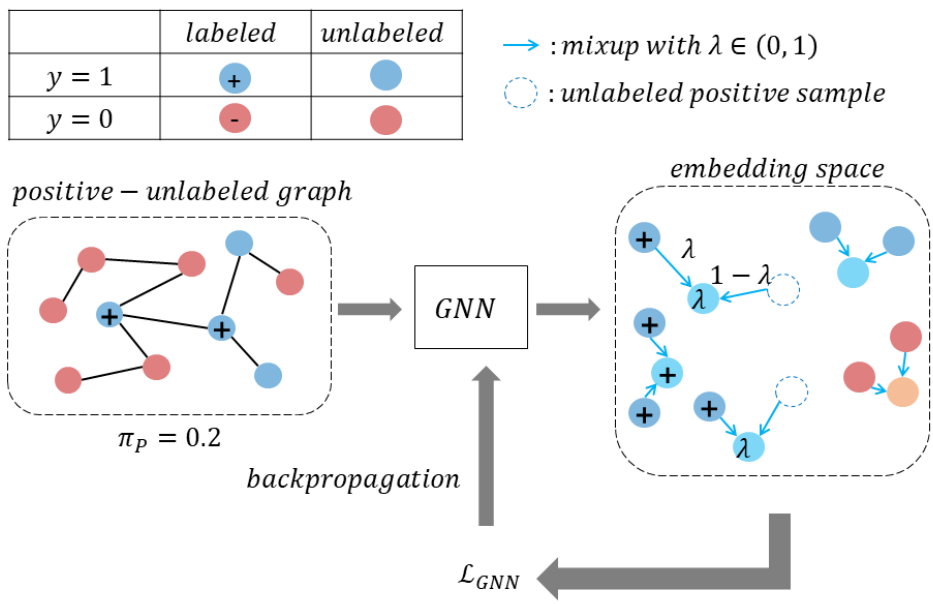

Figure 4.1: Overview of PUM−GNN.

We first introduce the common architecture of GNNs in section 4.1 and describe our Mixup strategy in section 4.2. Finally, in section 4.3, we introduce PUM−GNN optimization compared with the original PU optimization strategy.

4.1 GNN

We define a graph as $G = (V, E)$ where V denotes the set of nodes, and E is the set of edges. Each node i belonging to V has an initial feature vector x_i and its neighborhood is defined as $N(i) = \{j \in V \mid (i,j) \in E\}$. GNNs obtain the node representations $h_i^{(l)}$ at layer l through the message passing mechanism described as below:

$$m_i^{(l)} = AGGREGATE\left(\left\{h_j^{(l-1)} \mid j \in N(i)\right\}\right), \quad (1)$$

where $m_i^{(l)}$ is the aggregated message vector at layer l and $AGGREGATE$ is a function that aggregates neighbors' information. At the input layer ($l = 0$), nodes' representations are initialized to x_i. After aggregating the representations of the neighborhood nodes, node i's representation at $l-$th layer is updated as follows.

$$h_i^{(l)} = UPDATE\left(h_i^{(l-1)}, m_i^{(l)}, W^{(l)}\right). \quad (2)$$

In equation (2), $W^{(l)}$ denotes the weight matrix at layer l. Finally, we can use $h_i^{(l)}$ as the embedding of node i.

4.2 Mixup strategy

Mixup [4, 5] produces the synthetic sample $(\tilde{x}, \tilde{y})$ by linearly interpolating samples (x_i, y_i) and (x_j, y_j), where x denotes the input feature and y denotes the label.

$$\tilde{x} = \lambda x_i + (1-\lambda) x_j, \quad (3)$$

$$\tilde{y} = \lambda y_i + (1-\lambda) y_j. \quad (4)$$

In equation (3) and (4), λ is drawn from Beta distribution [25] ($\lambda \sim \beta(\alpha, \alpha), \alpha \in (0, \infty)$). Mixup usually picks two random images from the same mini-batch and linearly interpolates them.

In our work, we focus on not only the data augmentation effect of Mixup but also the supervision correction through the mixing of certain samples. Unlike previous studies that randomly select samples, we carefully select mixup partners to improve the imprecise supervision within $\mathcal{X}_U$. Figure 4.2 shows the Mixup strategy of our method.

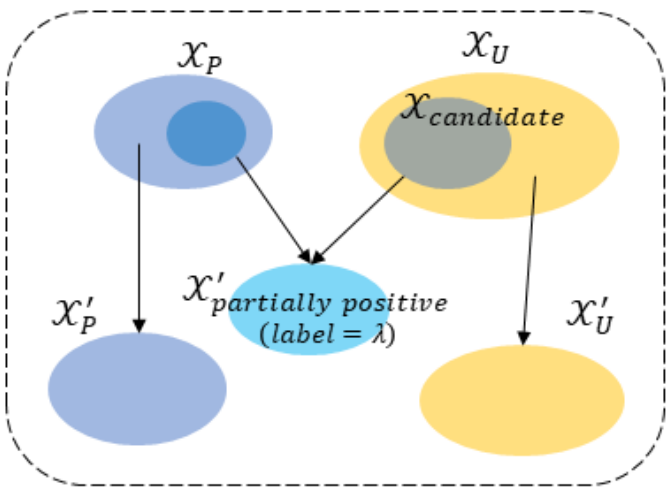

Figure 4.2: Mixup strategy of PUM-GNN.

4.2.1 Mixup between $\mathcal{X}_P$ and $\mathcal{X}_U$.

When mixing between $\mathcal{X}_P$ and $\mathcal{X}_U$, we first estimate unlabeled instances whose ground truth is positive but incorrectly classified as negative. We define these samples as potential positive samples, called candidate mixup pool $\mathcal{X}_{candidate}$ ($\subset \mathcal{X}_U$). We build $\mathcal{X}_{candidate}$

using the idea of graph convolution in homophilic graph. In the homophilic graph, nodes of the same label tend to be connected, so their embeddings are trained to be similar to each other. Therefore, we measure the similarity of trained embedding between $\mathcal{X}_P$ and $\mathcal{X}_U$. Top$-k$ unlabeled samples with high similarity to positive nodes are selected to build $\mathcal{X}_{candidate}$. To measure relativeness between samples, we use cosine similarity which is often used as a distance metric function.

$$\mathcal{X}_{candidate} = \left\{ x \mid x \in \mathcal{X}_U,\ top-k\left(cosine\ similarity(x_p, x)\right)\right\} \quad (5)$$

We also randomly select positive embeddings from $\mathcal{X}_P$ paired with $\mathcal{X}_{candidate}$. To produce partially positive embedding, we define λ to guarantee that the mixed sample $\tilde{x}$ is closer to positive sample x_i than the unlabeled sample x_j. Consequently, $(\tilde{x}, \tilde{y})$ is assigned to partially positive set ($\mathcal{X}'_{partially\ positive}$).

4.2.2 Mixup between $\mathcal{X}_P$ ($\mathcal{X}_U$) *and* $\mathcal{X}_P$ ($\mathcal{X}_U$).

After mixing between $\mathcal{X}_P$ and $\mathcal{X}_U$, the rest in $\mathcal{X}_P$ and $\mathcal{X}_U$ are mixed in each set to obtain a new embedding set $\mathcal{X}'_P$ and $\mathcal{X}'_U$ with an extended training distribution.

To sum up, the overall mixup partner selection is formulated as follows:

$$(x_i, y_i) = \begin{cases} Uniform(\mathcal{X}_P), & if\ (x_j, y_j) \in \mathcal{X}_P \cup \mathcal{X}_{candidate} \\ Uniform(\mathcal{X}_U), & if\ (x_j, y_j) \in \mathcal{X}_U \setminus \mathcal{X}_{candidate} \end{cases} \quad (6)$$

4.3 Optimization

4.3.1 Original PU optimization

Let x be an input node variable and $p(x)$ be the joint distribution of nodes. Then, $p(x)$ can be rewritten as follows using the law of the total probability:

$$p(x) = \pi_P p_p(x) + (1 - \pi_P) p_n(x), \quad (7)$$

where $\pi_P = p(y = +1)$ is a positive class prior probability. Equation (7) means probability π_P of unlabeled samples is positive and $(1 - \pi_P)$ is negative. $p_p(x) = p(x \mid y = +1)$ is a marginal distribution from which positive samples are drawn and $p_n(x) = p(x \mid y = 0)$ is a marginal distribution from which negative samples are drawn.

To train the model, let l be the loss function, then $l(y', y)$ measures the loss for predicted output y' and ground truth y. Given a set of representations $\mathcal{O}^L = \{o_1^L, o_2^L, \dots, o_M^L\}, o_i^L \in \mathbb{R}^2$ where L is final layer and o_i is feature representation of node i after running GNNs. Let mapping function $f : \mathcal{O} \to \mathcal{Y}$, to classify o_i into $\mathcal{Y} = \{+1, 0\}$. $f(o_i)$ maps the final representation of node i in the range $(0, 1)$. Then, the binary risk minimizer for the classifier can be learned as:

$$R(f) = \mathbb{E}[l(f(o), y)] = \pi_P R_p^+(f) + (1 - \pi_P) R_n^-(f), \quad (8)$$

where $R_p^+(f) = \mathbb{E}_{p_p(x)}[l(f(o), +1)]$ is the risk of samples with a positive label $(y = +1)$ and $R_n^-(f) = \mathbb{E}_{p_n(x)}[l(f(o), 0)]$ is the risk of samples with a negative label $(y = 0)$. In practice, we can

approximate $R_p^+(f)$ using the observed positive samples, but $R_n^-(f)$ is unknown because negative data is unavailable during a training phase.

PU learning treats the unobserved samples directly as unlabeled samples, not negative samples. $R_n^-(f)$ can be estimated via equation (8) because we assume that both unlabeled and total data follow the same distribution. Following [3], we express $p_u(x)$ as $\pi_P p_p(x) + (1-\pi_P)p_n(x)$. Then, $R_n^-(f)$ can be expressed as follows:

$$(1-\pi_P)R_n^-(f) = R_u^-(f) - \pi_P R_p^-(f), \tag{9}$$

where $R_u^-(f)$ is the risk of unlabeled samples with a negative label and $R_p^-(f)$ is the risk of positive samples with a negative label. Thus, the final risk minimization is as follows:

$$R(f) = \pi_P R_p^+(f) - \pi_P R_p^-(f) + R_u^-(f). \tag{10}$$

4.3.2 PUM-GNN optimization

Equation (10) computes loss for positive samples (1st term), and negative ones (2nd, 3rd term). In our method, we produce a new embedding set in section 4.2. Using this, the loss function for PUM-GNN is as follows:

$$R(f) = \pi_P R_{p'}^+(f) + R_{pp}^{\lambda}(f) - \pi_P R_{p'}^-(f) + R_{u'}^-(f), \tag{11}$$

where $R_{p'}(f)$ calculates loss from $\mathcal{X}_P'$ and $R_{u'}(f)$ from $\mathcal{X}_U'$. $R_{pp}^{\lambda}(f)$ calculates the risk of $\mathcal{X}'_{partially\ positive}$ with soft label λ.

In this paper, we use a sigmoid activation function $f(x) = \frac{1}{1+\exp(-x)}$ to map the input x. Binary cross entropy is used for loss

function $l(y', y)$ of each sample, which is defined as:

$$l(y_i', y_i) = -[y_i \cdot \log(y_i') + (1 - y_i) \cdot \log(1 - y_i')]. \quad (12)$$

Chapter 5. Experiment

5.1 Dataset

We employ three datasets in our experiments. The basic information for each dataset is reported in Table 5.1. Cora, Citeseer, and PubMed [26] are citation networks, which is commonly used for node classification. On all datasets, each node represents scientific publications classified according to the research category it belongs to. Each publication in these datasets is described by word vectors which contain their textual contents. The nodes in datasets are classified into multiple classes. Therefore, following [16], we treat one class with the largest number of nodes as positive and the rest as negative for binary classification. Under this setting, we calculate positive class prior π_p. The resulting number of positive and negative nodes, and class prior π_p are summarized in Table 5.1.

Table 5.1: Dataset

Name	Cora	Citeseer	PubMed
# of nodes	2,708	3,327	19,717
# of edges	10,556	9,104	88,648
# of features	1,433	3,703	500
# of classes	7	6	3
# of pos	818	701	7,875
# of neg	1,890	2,626	11,842
π_P	0.3	0.213	0.397

5.2 Baselines

We perform binary node classification under not only PU labels, but also PN (supervised) labels. For each case, the baseline below was used.

5.2.1 positive-unlabeled

Since PUM-GNN is the first paper to apply PU learning and Mixup to GNNs, we compare with/without our Mixup strategy. For the variants of PUM-GNN, we use three base models - MLP, GCN, and GAT.

- **PU-GNN** is a basic model for graph-based PU learning.

Unbiased risk estimator [2] is utilized for GNN.

● PU-GNN-M simply mixes embedding from each of $\mathcal{X}_P$ and $\mathcal{X}_U$.

● PUM-GNN utilizes a partially positive set and also randomly mixed embedding from each of $\mathcal{X}_P$ and $\mathcal{X}_U$ (proposed method).

5.2.2 positive-negative

To verify the effectiveness of PUM-GNN compared to supervised methods, we utilize SOTA GNNs and regularization methods. They are trained on fully supervised datasets. For regularization methods, we use GCN [6] as a base model. The details of baseline methods are presented below.

- MLP is a fully-connected feed-forward neural network without using an edge connection.
- GCN [6] integrates the structure and feature information of nodes using the adjacency matrix.
- GAT [7] uses the attention mechanism to learn the different strengths between the ego node and each neighboring node.
- GIN [27] adopts an injective mapping function to aggregate neighbors' information to learn more powerful representations.
- DropEdge [28] simply drops edges by random to improve generalization in deep GCNs.
- P-reg [29] adopts graph Laplacian regularization to provide extra supervision signal and simply simulate a deep GCN.

- **Mixup-conv** [17] first mixes two node features and performs graph convolution under each topology. Two aggregated messages are mixed to obtain the node's final representations.

5.3 Implementation details

We implement PUM-GNN by using PyTorch and torch_geometric with Adam optimizer. When calculating the positive-unlabeled risk of Equation (11), we employ BCELoss. The α value of the beta distribution is fixed at 4.0, to obtain a value that is skewed to the center value. Also, we fix the size of candidate mixup pool $\mathcal{X}_{candidate}$ to $|\mathcal{X}_P| * 10\%$, and choose positive label ratio r from $\{10\%, 30\%, 50\%, 70\%, 90\%\}$. For each dataset, we randomly select $r\%$ from $\mathcal{X}_P$ as a positive training set, and the rest positive nodes and negative nodes as an unlabeled training set. If r is high (low), it means more (fewer) positive samples are used during training. The training session is repeated for 300 epochs, and every training, model is updated with a newly obtained training set via Mixup.

5.4 Experimental results

In this section, we perform various experiments to answer the following research questions.

- RQ1: How accurate is PUM-GNN compared to baseline methods under both PN and PU label cases? How do the results change with different ratios of positive training nodes?
- RQ2: Does PUM-GNN also perform well when π_P is high (more positive nodes and fewer unlabeled nodes)?
- RQ3: Does PUM-GNN alleviate the negative-prediction problem of PU learning?
- RQ4: Does PUM-GNN distinguish unlabeled nodes well and enable stable learning during training?
- RQ5: How does the accuracy of PUM-GNN change with different hyperparameters for the classifier?

5.4.1 Node classification performance (RQ1)

We vary r to adjust the positive label ratio and the results of average node classification performance (F1 score) are reported in Table 5.2. The best result is bolded, and the second-best is underlined.

Table 5.2: Node classification performance under PU labels, showing average F1 score.

Cora ($\pi_P = 0.3$)				
Base model		$r = 90\%$	$r = 50\%$	$r = 10\%$
MLP	PU−MLP	0.7179	0.5921	0.0914
	PU−MLP−M	0.7183	0.5242	0.0914
	PUM−MLP	0.7532	0.6181	0.1116
GCN	PU−GCN	0.8529	0.6616	0.0819
	PU−GCN−M	0.8698	0.6560	0.0706
	PUM−GCN	<u>0.8785</u>	<u>0.8506</u>	0.1140
GAT	PU−GAT	0.8724	0.7655	0.2258
	PU−GAT−M	0.8614	0.7918	<u>0.2590</u>
	PUM−GAT	**0.8902**	**0.8652**	**0.2714**

Citeseer ($\pi_P = 0.213$)				
Base model		$r = 90\%$	$r = 50\%$	$r = 10\%$
MLP	PU−MLP	0.6996	0.4503	0.0438
	PU−MLP−M	0.6975	0.4333	0.0580
	PUM−MLP	0.7075	0.5168	0.0979
GCN	PU−GCN	0.7715	0.5361	0.0584
	PU−GCN−M	<u>0.7826</u>	0.5758	0.1389
	PUM−GCN	**0.7865**	**0.7619**	0.1686
GAT	PU−GAT	0.7722	0.6301	0.2013
	PU−GAT−M	0.7638	0.6359	<u>0.2484</u>
	PUM−GAT	0.7761	<u>0.7311</u>	**0.2757**

PubMed ($\pi_P = 0.397$)				
Base model		$r = 90\%$	$r = 50\%$	$r = 10\%$
MLP	PU−MLP	0.8362	0.7782	0.5211
	PU−MLP−M	0.8449	0.7968	0.6137
	PUM−MLP	0.8463	0.8227	0.6591
GCN	PU−GCN	0.8550	0.7697	0.4097
	PU−GCN−M	0.8556	0.8128	0.5201
	PUM−GCN	0.8663	0.8327	0.6397
GAT	PU−GAT	<u>0.8683</u>	0.8255	0.7362
	PU−GAT−M	0.8664	<u>0.8342</u>	<u>0.7555</u>
	PUM−GAT	**0.8728**	**0.8357**	**0.7879**

Generally, PUM−GNN shows the best performance on all datasets, and PU−GNN−M shows the second best. Also, we can see that Mixup−based methods are better than the basic PU model, which shows the regularization effect of Mixup. Mixup extends the training distribution by linearly interpolating features, which lead to that of the associated targets. It has been demonstrated to get better representation and higher generalization ability [4].

In addition, it shows that the smaller r and more unlabeled nodes, the greater the performance improvement of the PUM−GNN. When $r = 90\%$, the performance improvement is $2 \sim 4\%$, $r = 10\%$ results in a much larger performance improvement of $25 \sim 30\%$ than the basic PU model, PU−GNN. This is because adding new supervision is more effective in situations where labels are limited.

From the model perspective, MLP does not use edge connections

and fails to capture embedding similarities, thus resulting in minimal performance improvements over other models. In general, GAT [7] has better performance than GCN [6], and the same results came out when PU learning was applied. We can conclude that producing a new training set and adding partially positive samples based on embedding similarity is helpful in PU learning.

Also, we experiment with the original binary labels to compare PUM−GNN with SOTA GNN and regularization methods. In this section, we want to find whether PUM−GNN can beat other methods which are trained on fully supervised datasets. As we can see in Table 5.3, in Cora and Citeseer network, PUM−GCN and PUM−GAT show similar results with others when r is more than 80%. Especially, when $r = 90\%$, PUM−GAT and PUM−GCN can beat other promising comparison methods. The second−best method on both datasets was graph regularizer P−reg [29], which shows the importance of regularization during training. However, when the graph size is large, Mixup−conv [17], which applies Mixup both in feature space and embedding space, shows the best result. If the graph is large, mixing the final representation is not enough, and additional work is needed in the convolution process.

Therefore, when there are limited labeled nodes during training, we can transform the problem into a PU learning task which can result in a similar performance to supervised methods using only at least 80% of the positive samples.

Table 5.3: Node classification performance under PN labels, showing average F1 score.

	Cora ($\pi_P = 0.3$)		
MLP	0.7477		
GCN	0.8742		
GAT	0.8580		
GIN	0.8564		
DropEdge	0.8643		
P−reg	<u>0.8821</u>		
Mixup−conv	0.8746		
	$r = 70\%$	$r = 80\%$	$r = 90\%$
PUM−MLP	0.7018	0.7193	0.7532
PUM−GCN	0.8531	0.8554	0.8785
PUM−GAT	0.8683	0.8784	**0.8902**

	Citeseer ($\pi_P = 0.213$)		
MLP	0.6923		
GCN	0.7619		
GAT	0.7752		
GIN	0.7521		
DropEdge	0.7613		
P−reg	<u>0.7854</u>		
Mixup−conv	0.7721		
	$r = 70\%$	$r = 80\%$	$r = 90\%$
PUM−MLP	0.6552	0.7097	0.7075
PUM−GCN	0.7633	0.7716	**0.7865**
PUM−GAT	0.7519	0.7566	0.7761

	PubMed ($\pi_P = 0.397$)		
MLP	0.8725		
GCN	0.8654		
GAT	<u>0.8787</u>		
GIN	0.8540		
DropEdge	0.8641		
P−reg	0.8584		
Mixup−conv	**0.8810**		
	$r = 70\%$	$r = 80\%$	$r = 90\%$
PUM−MLP	0.8276	0.8237	0.8463
PUM−GCN	0.8468	0.8441	0.8663
PUM−GAT	0.8609	0.8581	0.8728

5.4.2 Effect of π_P (RQ2)

In section 5.4.1, we investigate the performance of PUM-GNN under various r. Since the main challenge of machine learning is to achieve superior performance with a limited number of data, we focus on the results of PUM-GNN where π_P is small $(0.2 \sim 0.4)$ to address the label sparsity issue.

In this section, we want to answer the question: *how PUM-GNN performs under reversed* π_P*?* For the experiment, we flip the label when binarizing the dataset, so that π_P is set to a slightly larger value $(0.6 \sim 0.7)$. The results of average node classification performance (F1 score) under reversed π_P are described in Table 5.4. Similar to section 5.4.1, PUM-GNN performs the best and second best over three datasets except in PubMed, $r = 90\%$. This shows the strength of PUM-GNN that produces consistent results regardless of π_P.

Table 5.4: Node classification performance under reversed π_P, showing average F1 score.

Cora ($\pi_P = 0.7$)				
Base model		$r = 90\%$	$r = 50\%$	$r = 10\%$
MLP	PU-MLP	0.8634	0.6902	0.2646
	PUM-MLP	0.8898	0.7759	0.4202
GCN	PU-GCN	0.9326	0.7697	0.2431
	PUM-GCN	0.9415	0.8522	0.4582
GAT	PU-GAT	0.9290	0.7883	0.4033
	PUM-GAT	**0.9518**	**0.8988**	**0.5478**
Citeseer ($\pi_P = 0.787$)				
Base model		$r = 90\%$	$r = 50\%$	$r = 10\%$
MLP	PU-MLP	0.9075	0.7486	0.2044
	PUM-MLP	0.9281	0.7914	0.4094
GCN	PU-GCN	0.9255	0.7516	0.2252
	PUM-GCN	0.9435	0.8386	0.3547
GAT	PU-GAT	0.9245	0.7243	0.2780
	PUM-GAT	**0.9437**	**0.8472**	**0.4448**
PubMed ($\pi_P = 0.603$)				
Base model		$r = 90\%$	$r = 50\%$	$r = 10\%$
MLP	PU-MLP	0.8888	0.8150	0.6068
	PUM-MLP	0.8890	0.8906	0.8230
GCN	PU-GCN	0.8982	0.8203	0.5209
	PUM-GCN	0.9065	**0.9072**	0.8192
GAT	PU-GAT	**0.9117**	0.9020	0.7038
	PUM-GAT	0.9002	0.9027	**0.8843**

5.4.3 Analysis of negative-prediction preference (RQ3)

PU learning learns a binary classifier without any labeled negative data. Especially, cost-sensitive PU approaches [1, 2, 3] classify unlabeled samples to weighted N. Therefore, they tend to predict unlabeled samples as N. This problem is crucial in the real world, where it is important to correctly distinguish unlabeled samples containing potential P and N. In our method, we prevent this problem with our new Mixup strategy. It transforms misclassified P samples into augmented samples that are partially positive, so that learned boundary moves to the fully supervised ones.

To find whether PUM-GNN can mitigate the negative-prediction preference of PU learning, we analyze the false negative and true positive of test data. In Figure 5.1, PUM-GCN shows fewer false negatives (orange) and more true positives (green) than PU-GCN in all datasets. In Cora and Citeseer, when $r = 30\% \sim 70\%$, improvement was large. In PubMed, which has a large graph size, small r gives more improvement. These results imply that our method benefits the supervision correction within marginal pseudo-negative instances. Thus, we can conclude that PUM-GNN can alleviate the negative-prediction preference of PU learning.

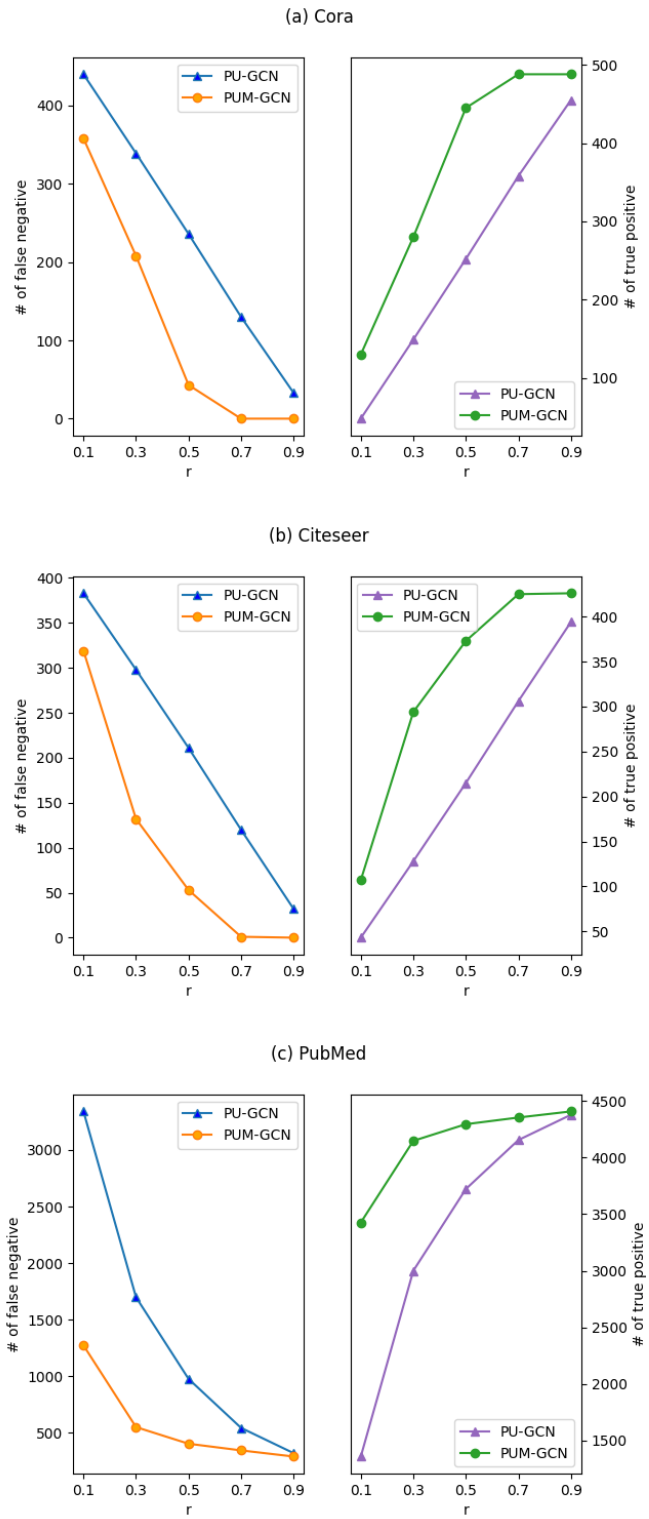

Figure 5.1: Analysis of false negatives and true positives.

5.4.4 Analysis of training session (RQ4)

In this section, we study the effects of PUM−GCN during training.

We show the test loss of PU−GCN and PUM−GCN during training in Figure 5.2 on the Cora (up) and Citeseer (down) networks. As we can see, for both methods, the loss decreases in the beginning. However, PUM−GCN significantly reduces the loss as training progresses and helps GCN [6] to converge to a lower loss. This shows that our method can regularize GCN to enable stable training and reduce over−fitting.

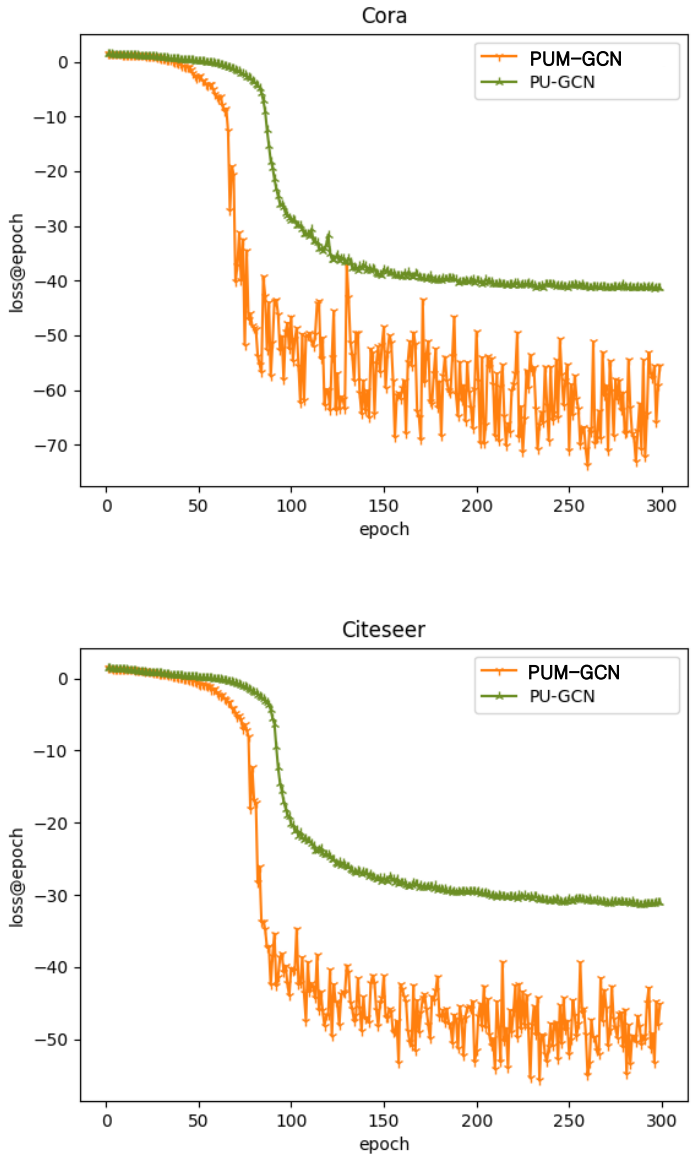

Figure 5.2: The training curves of PU−GCN and PUM−GCN.

(up: Cora, down: Citeseer)

It is important for PU learning to successfully perform binary classification without any negative data. In cost-sensitive PU learning, $\mathcal{X}_P$ is well classified but to distinguish $\mathcal{X}_U$ well is the challenge. To check the discriminative ability of PUM-GNN, we present t-SNE visualization of trained node representations obtained by PU-GCN and PUM-GCN in Figure 5.3.

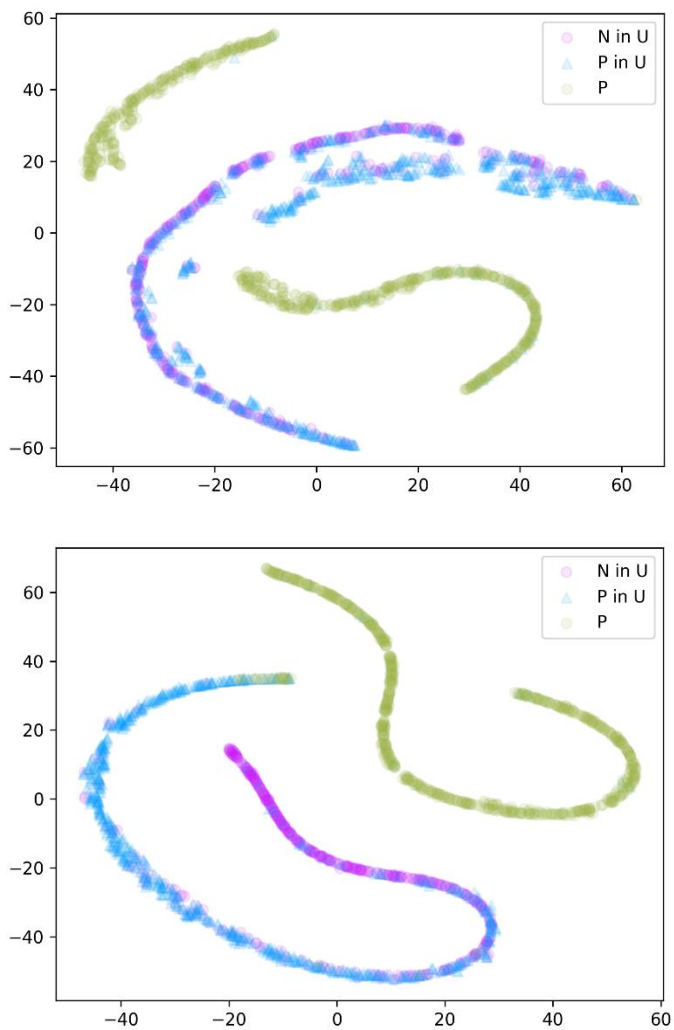

Figure 5.3: t-SNE visualization of trained embedding.
(up: PU-GCN, down: PUM-GCN)

Node representations of positive class (green), are discriminative compared to unlabeled ones in both methods. However, there is a significant difference in distinguishing unlabeled nodes. Without applying our Mixup strategy (up), node representations of positive (blue) and negative (purple) that belong to $\mathcal{X}_U$ are hardly distinguishable. In contrast, those representations obtained by PUM-GCN (down) are easily distinguished. These highly discriminative representations eventually lead to better label predictions than less discriminative ones.

5.4.5 Parameter Sensitivity (RQ5)

We investigate the performance of PUM-GNN with different values of hyper-parameters. We first tune the number of GCN [6] layers. Figure 5.4 shows that when the number of layers is **1**, the F1 score is the lowest. In general, when the number of layers is **2 ~ 4**, the ego node aggregates deeper information and the performance is excellent. Even if the number of layers increases, there is no significant performance difference up to **0.05**, and shows stable results.

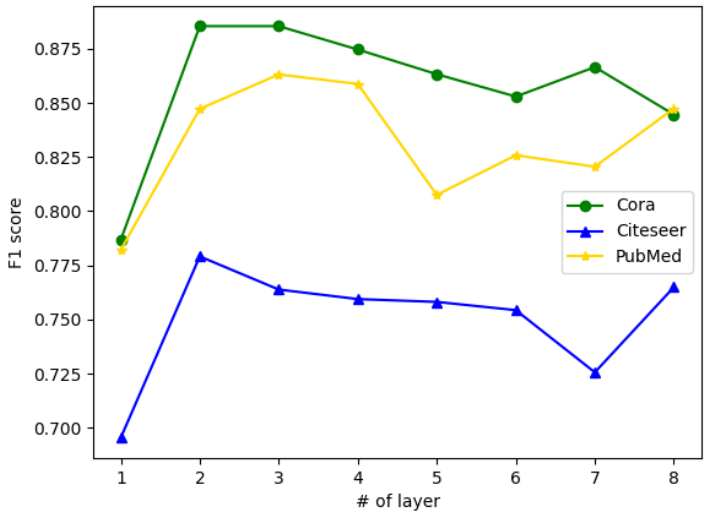

Figure 5.4: Parameter analysis with respect to # of layers.

We also evaluate how sensitive PUM-GNN is to the selection of Mixup weight λ. Therefore, we differ hyper-parameter value: α, which controls the Beta distribution and plot the experimental results of PUM-GCN with $r = 0.5$ in Table 5.5. The smaller the α, the more skewed the distribution is formed at both ends, and when the α is large, the distribution is concentrated in the center.

As we can see, when α is more than 1, the performance is good. We think the reason is that if α is 1 or higher, λ is chosen close to 0.5, so two data are evenly mixed. However, if α is less than 1, distribution is biased to both ends and particular data has a big impact on the mixed result. Empirically, we choose $\alpha = 4.0$ as the default setting.

Table 5.5: Node classification performance under different α values.

α	Cora	Citeseer	PubMed
0.2	0.5922	0.7820	0.8411
0.5	0.6547	0.8208	0.8467
1.0	0.6607	0.8553	0.8515
2.0	**0.6926**	0.8599	0.8521
4.0	0.6872	**0.8690**	**0.8534**
5.0	0.6757	0.8617	0.8532

Chapter 6. Conclusion

In this paper, we propose PUM-GNN, a novel graph-based PU learning method utilizing Mixup [4, 5] strategy. After running GNNs, PUM-GNN finds candidate unlabeled nodes which are close to positive but easily misclassified to negative. It then mixes candidate nodes with positive ones to produce partially positive nodes with label λ. Also, it mixes the rest P and U nodes individually to make a new training set. In this way, PUM-GNN can not only regularizes the model but also gives more supervision to positive-unlabeled classification. We found PUM-GNN can increase performance in positive-unlabeled classification, and even in positive-negative classification. We experiment with various positive label ratios to show the advantage of PUM-GNN which is strong in label sparsity.

Bibliography

[1] Elkan, C., & Noto, K. (2008, August). Learning classifiers from only positive and unlabeled data. In Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 213-220).

[2] Du Plessis, M., Niu, G., & Sugiyama, M. (2015, June). Convex formulation for learning from positive and unlabeled data. In International conference on machine learning (pp. 1386-1394). PMLR.

[3] Kiryo, R., Niu, G., Du Plessis, M. C., & Sugiyama, M. (2017). Positive-unlabeled learning with non-negative risk estimator. Advances in neural information processing systems, 30.

[4] Zhang, H., Cisse, M., Dauphin, Y. N., & Lopez-Paz, D. (2017). mixup: Beyond empirical risk minimization. arXiv preprint arXiv:1710.09412.

[5] Verma, V., Lamb, A., Beckham, C., Najafi, A., Mitliagkas, I., Lopez-Paz, D., & Bengio, Y. (2019, May). Manifold mixup: Better representations by interpolating hidden states. In International Conference on Machine Learning (pp. 6438-6447). PMLR.

[6] Kipf, T. N., & Welling, M. (2016). Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907.

[7] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., & Bengio, Y. (2017). Graph attention networks. arXiv preprint arXiv:1710.10903.

[8] Acharya, A., Sanghavi, S., Jing, L., Bhushanam, B., Choudhary, D., Rabbat, M., & Dhillon, I. (2022). Positive Unlabeled Contrastive Learning. arXiv preprint arXiv:2206.01206.

[9] Bekker, J., & Davis, J. (2020). Learning from positive and unlabeled data: A survey. Machine Learning, 109(4), 719-760.

[10] Li, X., & Liu, B. (2003, August). Learning to classify texts using positive and unlabeled data. In IJCAI (Vol. 3, No. 2003, pp. 587-592).

[11] Schölkopf, B., Platt, J. C., Shawe-Taylor, J., Smola, A. J., & Williamson, R. C. (2001). Estimating the support of a high-dimensional distribution. Neural computation, 13(7), 1443-1471.

[12] Li, C., Li, X., Feng, L., & Ouyang, J. (2021, September). Who Is Your Right Mixup Partner in Positive and Unlabeled Learning. In International Conference on Learning Representations.

[13] Wu, M., Pan, S., Du, L., Tsang, I., Zhu, X., & Du, B. (2019, November). Long-short distance aggregation networks for positive unlabeled graph learning. In Proceedings of the 28th ACM International Conference on Information and Knowledge Management (pp. 2157-2160).

[14] Zhou, Y., Xu, J., Wu, J., Taghavi, Z., Korpeoglu, E., Achan, K., & He, J. (2021, August). Pure: Positive-unlabeled recommendation with generative adversarial network. In Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining (pp. 2409-2419).

[15] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... & Bengio, Y. (2020). Generative adversarial

networks. Communications of the ACM, 63(11), 139-144.
[16] Yoo, J., Kim, J., Yoon, H., Kim, G., Jang, C., & Kang, U. (2021, December). Accurate Graph-Based PU Learning without Class Prior. In 2021 IEEE International Conference on Data Mining (ICDM) (pp. 827-836). IEEE.
[17] Wang, Y., Wang, W., Liang, Y., Cai, Y., & Hooi, B. (2021, April). Mixup for node and graph classification. In Proceedings of the Web Conference 2021 (pp. 3663-3674).
[18] Yoo, J., Shim, S., & Kang, U. (2022, April). Model-Agnostic Augmentation for Accurate Graph Classification. In Proceedings of the ACM Web Conference 2022 (pp. 1281-1291).
[19] Yun, S., Han, D., Oh, S. J., Chun, S., Choe, J., & Yoo, Y. (2019). Cutmix: Regularization strategy to train strong classifiers with localizable features. In Proceedings of the IEEE/CVF international conference on computer vision (pp. 6023-6032).
[20] Huang, T., Dong, Y., Ding, M., Yang, Z., Feng, W., Wang, X., & Tang, J. (2021, August). Mixgcf: An improved training method for graph neural network-based recommender systems. In Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining (pp. 665-674).
[21] Christoffel, M., Niu, G., & Sugiyama, M. (2016, February). Class-prior estimation for learning from positive and unlabeled data. In Asian Conference on Machine Learning (pp. 221-236). PMLR.
[22] Ramaswamy, H., Scott, C., & Tewari, A. (2016, June). Mixture proportion estimation via kernel embeddings of distributions. In

International conference on machine learning (pp. 2052-2060). PMLR.

[23] Ivanov, D. (2020, December). Dedpul: Difference-of-estimated-densities-based positive-unlabeled learning. In 2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA) (pp. 782-790). IEEE.

[24] Garg, S., Wu, Y., Smola, A. J., Balakrishnan, S., & Lipton, Z. (2021). Mixture proportion estimation and pu learning: A modern approach. Advances in Neural Information Processing Systems, 34, 8532-8544.

[25] Gupta, A. K., & Nadarajah, S. (2004). Handbook of beta distribution and its applications. CRC press.

[26] Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., & Eliassi-Rad, T. (2008). Collective classification in network data. AI magazine, 29(3), 93-93.

[27] Xu, K., Hu, W., Leskovec, J., & Jegelka, S. (2018). How powerful are graph neural networks?. arXiv preprint arXiv:1810.00826.

[28] Rong, Y., Huang, W., Xu, T., & Huang, J. (2019). Dropedge: Towards deep graph convolutional networks on node classification. arXiv preprint arXiv:1907.10903.

[29] Yang, H., Ma, K., & Cheng, J. (2021, May). Rethinking graph regularization for graph neural networks. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 35, No. 5, pp. 4573-4581).

[30] Du Plessis, M. C., Niu, G., & Sugiyama, M. (2014). Analysis of

learning from positive and unlabeled data. Advances in neural information processing systems, 27.
[31] Schölkopf, B., Platt, J. C., Shawe-Taylor, J., Smola, A. J., & Williamson, R. C. (2001). Estimating the support of a high-dimensional distribution. Neural computation, 13(7), 1443-1471.

초록

믹스업 기반의 그래프 신경망을 사용한 희소 라벨에서의 양성-비라벨 노드 분류

홍한결
컴퓨터공학부
서울대학교 대학원

최근, 준지도학습은 실제 데이터 세트의 희소성으로 인해 상당한 관심을 얻고 있다. 훈련 중에 라벨이 지정된 노드가 부족한 그래프 데이터에서 데이터 희소성은 일반적인 문제이다. 본 논문에서는 양성-비라벨 (PU) 학습 방법을 그래프 신경망에 적용하여, 라벨이 지정되지 않은 많은 양의 노드를 학습에 활용하는 이진 노드 분류를 다룬다. 특히, 양성-비라벨 학습 방법은 양성 노드와 라벨이 없는 노드만을 사용해서 노드 간의 잠재적인 긍정적 및 부정적 상호 작용을 발굴하는 것을 목표로 한다. 본 논문은 믹스업 (Mixup) 기반의 그래프 신경망 (PUM-GNN) 이라는 새로운 프레임워크를 제안한다. 이 방법은 라벨링된 데이터가 적은 사례를 다루고, 믹스업 정규화를 사용하여 양성-비라벨 학습을 지도한다. 믹스업은 이미지 데이터 분류에서 유망한 연구이지만, 그래프 신경망 분야에서는 그래프의 불규칙성으로

인해 많이 연구되지 않았다. 우리는 임베딩 공간에서 믹스업을 사용하여 데이터를 증강시킬 뿐만 아니라 주변 의사 음성 인스턴스를 부분 긍정 라벨을 갖는 새로운 인스턴스로 변환하고, 라벨이 지정되지 않은 인스턴스 내의 부정확한 지도를 개선한다. 우리는 긍정 라벨 비율을 다양하게 조절하며 실험을 수행했고, 제안 모델이 과적합을 줄일 뿐만 아니라 희소 라벨에서 최첨단 방법을 능가한다는 것을 확인했다.

주요어: 준지도학습, 그래프 신경망, 정규화, 데이터 증강, 노드 분류

학번: 2021-26464