Accurate Graph-Based PU Learning without Class Prior

Jaemin Yoo[1,*], Junghun Kim[1,*], Hoyoung Yoon[1,*], Geonsoo Kim[2], Changwon Jang[2], U Kang[1]
[1]Seoul National University, South Korea
{jaeminyoo, bandalg97, crazy8597, ukang}@snu.ac.kr
[2]NCSOFT, South Korea
{geostatman, jangcw}@ncsoft.com

***Abstract*—How can we classify graph-structured data only with positive labels? Graph-based positive-unlabeled (PU) learning is to train a binary classifier given only the positive labels when the relationship between examples is given as a graph. The problem is of great importance for various tasks such as detecting malicious accounts in a social network, which are difficult to be modeled by supervised learning when the true negative labels are absent. Previous works for graph-based PU learning assume that the prior distribution of positive nodes is known in advance, which is not true in many real-world cases. In this work, we propose GRAB (Graph-based Risk minimization with iterAtive Belief propagation), a novel end-to-end approach for graph-based PU learning that requires no class prior. GRAB models a given graph as a Markov network and runs the marginalization and update steps iteratively. The marginalization step estimates the marginals of latent variables, while the update step trains a classifier network utilizing the computed priors in the objective function. Extensive experiments on five datasets show that GRAB achieves state-of-the-art accuracy, even compared with previous methods that are given the true prior.**

***Index Terms*—PU learning, graph neural networks, loopy belief propagation, Markov networks**

I. Introduction

How can we classify graph-structured data only with positive labels? Positive-unlabeled (PU) learning [1] is to train a binary classifier given only the positive labels when acquiring true negative labels is very difficult or even not possible. Such scenarios include detecting malicious users in a social network service, which can behave like normal users to fool the labeler. In this case, the problem is to find malicious accounts without having concrete labels for normal accounts. PU learning has been successfully applied to text [2] or image [3] data.

Classifying nodes in graph-structured data is a fundamental problem of data mining [4]–[6]. The problem has drawn much attention with the advancement of graph neural networks [7]–[9], which learn effective low-dimensional representations of nodes by considering feature vectors and the graph structure at the same time. PU learning on graph data can advance existing approaches for node classification by removing the assumption on the observation of negative labels, enlarging the coverage of applications toward challenging scenarios.

The main limitation of previous works [2], [10]–[13] for graph-based PU learning is that the true class prior is assumed to be known as $\pi_\mathrm{p} = P(Y = +1)$, where Y denotes the target variable. This is not a realistic assumption in most cases, since the exact value of π_p cannot be found if no negative labels are given. Another limitation is that previous approaches [2], [12] rely on heuristics for selecting *relatively positive* nodes from the unlabeled ones to assign pseudo labels, increasing the risk of misclassification based on the result of selection.

In this work, we propose GRAB (Graph-based Risk minimization with iterAtive Belief propagation), an iterative algorithm for PU learning on graph-structured data. GRAB models the given graph as a pairwise Markov network that represents the probabilistic relationships between nodes. This enables us to propagate a few positive observations to the entire graph by a graphical inference with an approximate prior $\hat{\pi}_\mathrm{p}$. The estimated marginals work as approximate answers to train a node classifier f, which is then used to improve the estimation $\hat{\pi}_\mathrm{p}$. Thus, each iteration of GRAB consists of a) the estimation of approximate marginals of nodes given $\hat{\pi}_\mathrm{p}$ and b) the training of a classifier f given the computed marginals, which is then used to improve the quality of $\hat{\pi}_\mathrm{p}$ for the next iteration.

As a result, GRAB effectively addresses the limitations of previous approaches. First, GRAB does not require the class prior π_p, since it estimates an approximate prior $\hat{\pi}_\mathrm{p}$ during the iterative process. Second, GRAB does not rely on heuristics for selecting relatively positive nodes, because the estimated marginals of unlabeled nodes become the answers of training the classifier f. Each unlabeled node is treated softly as both positive and negative based on its estimated marginal, and it provides rich soft labels for the effective training of f.

Our contributions are summarized as follows:

- **Method.** We propose GRAB, an accurate approach for PU learning on graph data. GRAB is prior-agnostic, i.e., it does not require to know the true prior π_p, which is not observable in real-world scenarios. We show that GRAB successfully estimates unknown π_p via iterations.
- **Theory.** We provide theoretical analysis on the properties of GRAB, including the relation to the EM algorithm, and the time and space complexities.
- **Experiments.** We perform extensive experiments on real-world datasets. GRAB shows the best performance in all datasets, even when all baselines fail to make meaningful prediction due to the limited observations as in Fig. 1.

[*]These authors contributed equally to this work.

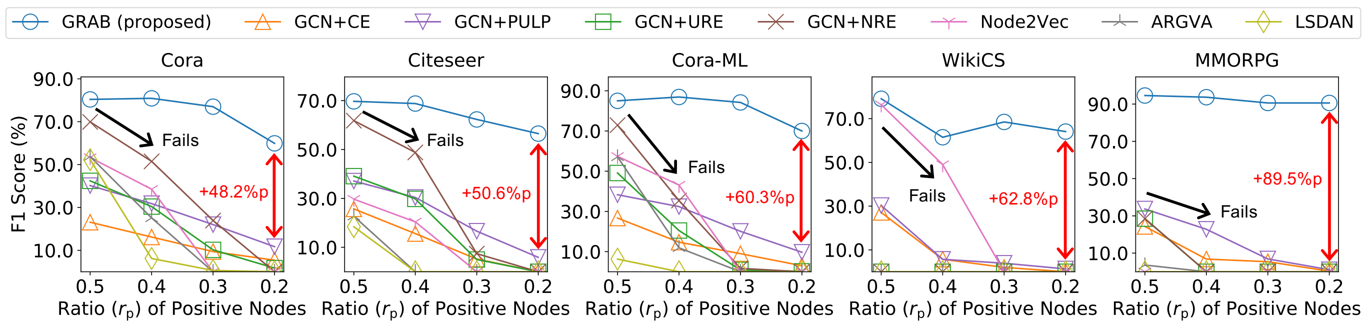

Fig. 1: The F1 scores of GRAB and the baseline approaches for graph-based PU learning. We change the ratio r_p of *observed* positive nodes among all positive ones; the problem is more difficult with smaller r_p. GRAB shows high F1 scores with all values of r_p while all competitors fail at performing meaningful classification. This indicates the robustness of GRAB even in imbalanced PU learning scenarios, where the number of observations is much smaller than the number of all nodes.

TABLE I: Symbols.

Symbol	Description
G	Undirected graph consisting of $\mathcal{V}$ and $\mathcal{E}$
$\mathcal{V}, \mathcal{E}$	Sets of all nodes and edges, respectively
$\mathcal{P}$	Set of labeled positive nodes
$\mathcal{U}$	Set of unlabeled nodes that we aim to classify
$\mathbf{X}$	Feature matrix for all nodes in G
$\mathbf{y}$	Label vector for the labeled nodes in $\mathcal{P}$
$\mathbf{B}$	Belief matrix of nodes
π_p	(Unknown) class prior
$\hat{\pi}_\mathrm{p}$	Estimation of unknown π_p
$f(\cdot)$	Classifier network with parameters θ
$\hat{y}_i(u)$	Prediction of f for node i being labeled as u
$\tilde{\mathcal{L}}(\cdot)$	Objective function that GRAB aims to minimize

The rest of this paper is organized as follows. In Section II, we introduce related works on graph-based PU learning. In Section III, we propose GRAB with its theoretical properties. We introduce our experimental setup in Section IV and show experimental results on real-world datasets in Section V. We conclude at Section VI.

II. Related Works

We present the definition of graph-based PU learning and introduce related works. Table I summarizes the symbols used in this paper.

A. Graph-Based PU Learning

PU learning is to train a binary classifier given only positive and unlabeled examples [14]–[16]. A promising approach to solve the problem is to utilize the relations between examples, assigning pseudo labels to the unlabeled ones for the training of a classifier. Thus, the problem can be solved more easily if the relations between examples are given explicitly as a graph. We define graph-based PU learning as follows.

Problem 1 (Graph-based PU learning)**.** *We are given an undirected graph $G = (\mathcal{V}, \mathcal{E})$ and a feature matrix $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times d}$, where $\mathcal{V}$ and $\mathcal{E}$ are the sets of nodes and edges, respectively, and d is the number of features. We know only a fraction of all positive nodes as a set $\mathcal{P}$. The other nodes $\mathcal{V} \backslash \mathcal{P}$ are unlabeled and represented as a set $\mathcal{U}$. Given G, $\mathbf{X}$, and $\mathcal{P}$, the goal of graph-based PU learning is to train a binary classifier f that accurately predicts the unknown labels of $\mathcal{U}$.*

PE-PUC [11] and PU-LP [12] find relatively positive examples from $\mathcal{U}$ based on the graphical structure, treating the remaining examples as (relatively) negative. This changes the problem into positive-negative (PN) learning where a typical binary classifier can be used directly. The limitation is that the overall performance highly depends on the way to select such examples, which is based on heuristics.

PULD [10] and puNet [2] find the labels of unlabeled examples by solving optimization problems with latent variables. Such approaches naturally incorporate the graph information and feature vectors in a single optimization problem without separate steps such as in [11], [12]. However, they assume the linear relationships between variables and between examples in the graph, which is not true in many cases.

LSDAN [13] is a graph neural network (GNN)-based model that considers both long- and short-distance relations between nodes to improve the performance of PU learning. The limitation is that heavy computation is required for generating $\mathbf{A}^n$ to consider long-distance relations, where $\mathbf{A}$ is the adjacency matrix of a graph, which becomes intractable in large graphs or when the target distance n is large.

The existing approaches require the exact prior π_p, which is not available in real-world scenarios of PU learning. We address the limitation by designing an iterative approach for estimating the unknown prior based on the graphical structure and feature vectors of nodes. At the same time, GRAB does not select relatively positive nodes by heuristics, as it assigns continuous marginals to the unlabeled nodes by the graphical inference, providing richer evidence for the training.

B. Unbiased Risk Estimators

Unbiased risk estimators [17], [18] are objective functions specifically designed for PU learning. The idea is utilizing the following equality to address the absence of negative data:

$$(1 - \pi_\mathrm{p})\hat{R}_\mathrm{n}^-(f) = \hat{R}_\mathrm{u}^-(f) - \pi_\mathrm{p}\hat{R}_\mathrm{p}^-(f), \tag{1}$$

where π_{p} is the class prior $P(Y=+1)$ of unlabeled nodes being labeled as positive, f is a classifier, and $\hat{R}_{\mathrm{n}}^{-}(f)$, $\hat{R}_{\mathrm{u}}^{-}(f)$, and $\hat{R}_{\mathrm{p}}^{-}(f)$ are the risk terms for the negative, unlabeled, and positive nodes, respectively, of classifying them as negative by f. The hat sign represents that a risk is calculated empirically from the given data, instead of random variables. Equation (1) represents $\hat{R}_{\mathrm{n}}^{-}(f)$, which is unknown in PU learning, by the known risks $\hat{R}_{\mathrm{u}}^{-}(f)$ and $\hat{R}_{\mathrm{p}}^{-}(f)$.

Kiryo et al. [3] proposed the non-negative risk estimator to prevent the unbiased risk estimator from going negative:

$$\hat{R}_{\mathrm{pu}}(f)=\pi_{\mathrm{p}}\hat{R}_{\mathrm{p}}^{+}(f)+\max(0,\hat{R}_{\mathrm{u}}^{-}(f)-\pi_{\mathrm{p}}\hat{R}_{\mathrm{p}}^{-}(f)), \quad (2)$$

where $\hat{R}_{\mathrm{p}}^{+}(g)$ is the risk of classifying the positive nodes as positive. The training is done by minimizing Equation (2).

The main limitation of these risk estimators is that the prior π_{p} should be known in advance of the training, which is not possible in real-world PU learning scenarios. Moreover, they do not work well in imbalanced settings where the number of observed examples is very small, since the unlabeled examples dominate the training. We address the two limitations by our GRAB, which does not require π_{p} and performs well in highly imbalanced settings by using the unlabeled examples as both positive and negative ones with approximate marginals.

C. Loopy Belief Propagation

Loopy belief propagation (LBP) is an inference algorithm for graphical models, which has been used widely for the node classification task [19]–[22]. A common approach is to treat a graph as a pairwise Markov network with respect to the target classes and run LBP to estimate the class distribution of each node as an approximate marginal. In other words, each node is treated as a random variable that is correlated with its direct neighbors following the Markov property. The strength of LBP is its stable performance even without feature vectors and the linear scalability with the number of edges. We utilize LBP in GRAB to estimate the marginal probabilities of nodes.

D. Graph Neural Networks

Graph neural networks (GNN) refer to deep neural networks specialized for graph-structured data [23], [24]. Graph convolution networks (GCN) [7] have shown a great performance for semi-supervised node classification. Graph attention networks (GAT) [8] improve GCNs by replacing the simple convolution operation with a learnable attention mechanism that takes the node representations as inputs. Deep graph informax (DGI) [9] generalizes GNNs into challenging scenarios where node labels are not observed. We use a GCN as the main classifier f for PU learning in both GRAB and the baseline approaches considering its robust and consistent performance.

III. Proposed Method

We propose GRAB (Graph-based Risk minimization with iterAtive Belief propagation), an accurate approach for PU learning on graph data without knowing the class prior. The overall process of GRAB is shown in Algorithm 1. We first introduce the objective function in Section III-A and describe

Algorithm 1: GRAB (Graph-based Risk minimization with iterAtive Belief propagation).

```
Input  : Graph $G=(\mathcal{V},\mathcal{E})$, set $\mathcal{P}$ of positive nodes,
         set $\mathcal{U}$ of unlabeled nodes, feature matrix $\mathbf{X}$,
         and classifier $f$ with initial parameters $\theta^{\mathrm{new}}$
Output: Best parameters $\theta$ and estimated prior $\hat{\pi}_{\mathrm{p}}$
1  $l^{\mathrm{new}} \leftarrow \infty$ ;                          // Initial loss value
2  $\hat{\pi}_{\mathrm{p}}^{\mathrm{new}} \leftarrow 0$ ;          // Initial estimation for $\pi_{\mathrm{p}}$
3  repeat
4      $l, \theta, \hat{\pi}_{\mathrm{p}} \leftarrow l^{\mathrm{new}}, \theta^{\mathrm{new}}, \hat{\pi}_{\mathrm{p}}^{\mathrm{new}}$ ;
5      $\mathbf{B} \leftarrow \mathrm{LBP}(\hat{\pi}_{\mathrm{p}}, G, \mathcal{P}, \mathcal{U})$ ;            // Algorithm 2
6      $\theta^{\mathrm{new}} \leftarrow \arg\min_{\theta} \tilde{\mathcal{L}}(\theta; \mathbf{X}, \mathbf{y}, \mathbf{B}, \mathcal{P}, \mathcal{U})$ ;  // Eq. (10)
7      $l^{\mathrm{new}} \leftarrow \tilde{\mathcal{L}}(\theta^{\mathrm{new}}; \mathbf{X}, \mathbf{y}, \mathbf{B}, \mathcal{P}, \mathcal{U})$ ;
8      $\hat{y}_i \leftarrow f(\mathbf{X}, i; \theta^{\mathrm{new}})$ for all $i \in \mathcal{V}$;
9      $\hat{\pi}_{\mathrm{p}}^{\mathrm{new}} \leftarrow |\mathcal{U}|^{-1} \sum_{i\in\mathcal{U}} \mathbb{I}[\hat{y}_i(+1) > 0.5]$ ;
10 until $l^{\mathrm{new}} > l$;
```

how to minimize it through the iterative process in Sections III-B to III-D. We then analyze the theoretical properties of GRAB in Section III-E.

A. Objective Function

In PU learning, we are given two sets of examples, $\mathcal{P}$ and $\mathcal{U}$, which contain positive and unlabeled ones, respectively. Existing risk estimators [3], [17] replace the loss function for negative data $\mathcal{N}$, which is not given, with the loss functions for positive and unlabeled data assuming that the exact prior $\pi_{\mathrm{p}}=P(Y=+1)$ is known. However, this assumption is not feasible in real-world scenarios, considering the restriction that negative data are *not* given; the exact calculation of π_{p} is not possible without additional knowledge.

Instead, we propose a latent variable $z_i \in \{+1,-1\}$ for each unlabeled node $i \in \mathcal{U}$ indicating whether node i is positive or negative. Let f be a node classifier with learnable parameters θ. We propose the following objective function for the training of f, which measures the negative log likelihood (NLL) for all observed nodes and all possible states of unobserved nodes by the expectation over $p(\mathbf{z}\mid\mathbf{X},\mathbf{y})$:

$$\mathcal{L}(\theta;\mathbf{X},\mathbf{y},\mathcal{P},\mathcal{U})=\frac{1}{|\mathcal{P}|}\sum_{i\in\mathcal{P}}(-\log\hat{y}_i(+1)) + \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z}\mid\mathbf{X},\mathbf{y})}\Big[\frac{1}{|\mathcal{U}|}\sum_{j\in\mathcal{U}}(-\log\hat{y}_j(z_j))\Big], \quad (3)$$

where $\mathbf{X}$ is a feature matrix for all nodes, $\mathbf{y}$ is the label vector for nodes $\mathcal{P}$, and $\hat{y}_i(u)=f(\mathbf{X},i;\theta)$ is the prediction of f for node i being labeled as $u\in\{+1,-1\}$. Thus, $\hat{y}_i(+1)+\hat{y}_i(-1)=1$ for every node i.

Equation (3) considers each unlabeled node $j\in\mathcal{U}$ as both positive and negative by a soft label $p(\mathbf{z}\mid\mathbf{X},\mathbf{y})$. For instance, if $P(Z_j=+1\mid\mathbf{X},\mathbf{y})=0.7$, f is trained to produce $(0.7,0.3)$ as the prediction $\hat{y}_j$ for node j, where the first and second elements correspond to the positive and the negative labels, respectively. We use $p(\mathbf{z}\mid\mathbf{X},\mathbf{y})$ instead of the unknown prior

π_p in existing risk estimators of Equations (1) and (2). The important factor for the performance of f is the appropriate modeling of $p(\mathbf{z} \mid \mathbf{X}, \mathbf{y})$. Our idea is to model the given graph G as a Markov network, assuming that all labeled and unlabeled nodes are correlated with each other following the structure of G, to fully leverage all given information.

B. Modeling Pairwise Markov Networks

The given graph G is an essential evidence for estimating the relations between examples. We consider G as a pairwise Markov network [25], based on the assumption that adjacent nodes are correlated with each other with the Markov property. This enables us to make a consistent probabilistic assumption over the entire graph and to propagate limited observations to the unlabeled nodes by probabilistic inference.

Then, the joint probability of $\mathbf{z}$ for all unlabeled nodes in G is computed as the product of all *node* and *edge potentials*, which determine the stochastic property of the graph. Given a node potential ϕ_i for each node i and an edge potential ψ_{ij} for each edge (i, j), the joint conditional probability $p(\mathbf{z} \mid \mathbf{X}, \mathbf{y})$ given $\mathbf{X}$ and $\mathbf{y}$ is represented as follows:

$$p(\mathbf{z} \mid \mathbf{X}, \mathbf{y}) = \frac{1}{T} \prod_{i \in \mathcal{U}} \phi_i(z_i \mid \mathbf{X}) \prod_{(i,j) \in \mathcal{E}} \psi_{ij}(v_i, v_j \mid \mathbf{X}), \quad (4)$$

where T is a normalizing constant that makes the distribution sum to one, and v_i is a temporary variable representing either y_i or z_i based on whether $i \in \mathcal{P}$ or $i \in \mathcal{U}$. We omit $\mathbf{X}$ in ϕ_i and ψ_{ij} in the rest of this section for brevity.

The node potential ϕ_i represents the prior of node i for being labeled as positive or negative without considering the other nodes in the graph. Since the true prior π_p is unknown, we introduce and use an approximate prior $\hat{\pi}_\mathrm{p}$, which we aim to improve through the iterative process of GRAB. The value of $\hat{\pi}_\mathrm{p}$ determines the node potential of each unlabeled node i as $\phi_i(+1) = \hat{\pi}_\mathrm{p}$ and $\phi_i(-1) = 1 - \hat{\pi}_\mathrm{p}$.

We set the initial value of $\hat{\pi}_\mathrm{p}$ to zero, as no prior information is given at first, treating all unlabeled nodes as negative. Then, $\hat{\pi}_\mathrm{p}$ becomes accurate as the iterations of GRAB proceed, based on the information of feature vectors and graphical structure. It is noteworthy that the initial value of $\hat{\pi}_\mathrm{p}$ should be less than 0.5 for the stability of the algorithm, since a classifier can be trained to predict all unlabeled nodes as positive if $\hat{\pi}_\mathrm{p} \geq 0.5$, which indicates that all nodes are likely to be positive.

The edge potential ψ_{ij} models an unnormalized joint probability of adjacent nodes i and j by a simple function that is shared for all edges in the graph:

$$\psi_{ij}(v_i, v_j) = \begin{cases} \epsilon & \text{if } v_i = v_j \\ 1 - \epsilon & \text{if } v_i \neq v_j. \end{cases} \quad (5)$$

The parameter ϵ determines the degree of homophily [19]. We assume that adjacent nodes are likely to have the same label by setting $\epsilon > 0.5$, which is true in most real networks where the edges represent the interactions between entities. We set $\epsilon = 0.9$ in all of our experiments.

Algorithm 2: Loopy belief propagation (LBP) with $\hat{\pi}_\mathrm{p}$.

```
input     : Approximate prior $\hat{\pi}_\mathrm{p}$, graph $G = (\mathcal{V}, \mathcal{E})$,
            set $\mathcal{P}$ of positive nodes, and set $\mathcal{U}$ of
            unlabeled nodes
parameter: Edge potential function $\psi$
output    : Belief $\mathbf{b}_i$ for each $i \in \mathcal{V}$
1  $\phi_i(+1), \phi_i(-1) \leftarrow 1, 0$ for each $i \in \mathcal{P}$ ;
2  $\phi_i(+1), \phi_i(-1) \leftarrow \hat{\pi}_\mathrm{p}, 1 - \hat{\pi}_\mathrm{p}$ for each $i \in \mathcal{U}$ ;
3  $m_{ij}(+1), m_{ij}(-1) \leftarrow 0.5, 0.5$ for each $(i, j) \in \mathcal{E}$ ;
4  while convergence criterion is not met do
5      for each $(i, j) \in \mathcal{E}$ and $v \in \{+1, -1\}$ do
6          $m^{\mathrm{new}}_{ij}(v) \leftarrow$
               $\sum_u \phi_i(u)\psi(u, v) \prod_{k \in \mathcal{N}_i \setminus \{j\}} m_{ki}(u)$ ;
7      end
8      $m_{ij} \leftarrow m^{\mathrm{new}}_{ij}$ for each $(i, j) \in \mathcal{E}$ ;
9  end
10 for each $j \in \mathcal{V}$ and $u \in \{+1, -1\}$ do
11     $T \leftarrow \sum_{v \in \{+1,-1\}} \phi_j(v) \prod_{i \in \mathcal{N}_j} m_{ij}(v)$ ;
12     $b_j(u) \leftarrow \frac{1}{T} \phi_j(u) \prod_{i \in \mathcal{N}_j} m_{ij}(u)$ ;
13 end
```

C. Estimation by Approximate Marginals

The computation of objective function $\mathcal{L}$ is intractable, due to the expectation over $\mathbf{z}$ that involves $O(2^{|\mathcal{U}|})$ computations. Thus, we decompose the joint distribution into the product of marginals, instead of computing the expectation directly:

$$p(\mathbf{z} \mid \mathbf{X}, \mathbf{y}) \approx \prod_i p_i(z_i \mid \mathbf{X}, \mathbf{y}), \quad (6)$$

where p_i is the marginal distribution for node i.

We run loopy belief propagation (LBP) to get the marginals of nodes approximately. The computation of LBP takes linear time with the number of edges, and is thus an efficient way to marginalize the joint distribution in large graphs. LBP updates *messages* for all edges via iterations. A message $m_{ij}(v)$ from node i to j represents the probability of node j having state $v \in \{+1, -1\}$ estimated by node i. Every message $m_{ij}(v)$ is initialized as 0.5 and updated through iterations as follows:

$$m_{ij}(v) \leftarrow \sum_{u \in \{+1,-1\}} \phi_i(u)\psi_{ij}(u, v) \prod_{k \in \mathcal{N}_i \setminus \{j\}} m_{ki}(u), \quad (7)$$

where $\mathcal{N}_i$ is the set of neighbors of node i. LBP is run until the messages converge, taking typically a few iterations.

The *belief* $b_j(u)$ of each node j for state u, which approximates the marginal probability $p_j(u \mid \mathbf{X}, \mathbf{y})$, is computed as follows after the convergence of messages:

$$b_j(u) = \frac{\phi_j(u) \prod_{i \in \mathcal{N}_j} m_{ij}(u)}{\sum_{v \in \{+1,-1\}} \phi_j(v) \prod_{i \in \mathcal{N}_j} m_{ij}(v)}. \quad (8)$$

Algorithm 2 describes the process of LBP, which takes an approximate prior $\hat{\pi}_\mathrm{p}$ to determine the potentials of unlabeled nodes. The algorithm stops if the difference between messages in consecutive iterations is smaller than a threshold θ, which is set to 10^{-4} in all of our experiments.

Given the approximate marginals, which are represented as a belief matrix $\mathbf{B} \in \mathbb{R}^{|\mathcal{V}|\times 2}$, we rewrite the objective function $\mathcal{L}$ of Equation (3) without the expectation term:

$$\mathcal{L}(\theta; \mathbf{X}, \mathbf{y}, \mathbf{B}, \mathcal{P}, \mathcal{U}) \approx \frac{1}{|\mathcal{P}|}\sum_{i\in\mathcal{P}}(-\log \hat{y}_i(+1)) + \frac{1}{|\mathcal{U}|}\sum_{j\in\mathcal{U}}\sum_{z_j\in\{+1,-1\}}(-b_j(z_j)\log \hat{y}_j(z_j)). \quad (9)$$

This involves the approximate marginal b_j of each unlabeled node j as the answer for training f, considering the relations between all positive and unlabeled nodes in the graph by the modeling of a pairwise Markov network.

We then propose our final objective function that generalizes the negative log likelihood as a loss function l:

$$\tilde{\mathcal{L}}(\theta; \mathbf{X}, \mathbf{y}, \mathbf{B}, \mathcal{P}, \mathcal{U}) = \frac{1}{|\mathcal{P}|}\sum_{i\in\mathcal{P}} l(\bar{y}_i, \hat{y}_i) + \frac{1}{|\mathcal{U}|}\sum_{j\in\mathcal{U}} l(b_j, \hat{y}_j), \quad (10)$$

where $\bar{y}_i$ is the one-hot representation of y_i, which is either $(1, 0)$ or $(0, 1)$, and b_j and $\hat{y}_j$ are probability vectors of length two representing the beliefs and predictions, respectively, for node j. Equation (10) becomes the same as Equation (9) if we adopt the cross entropy function as l, which is defined as $l(\bar{y}_i, \hat{y}_i) = -\bar{y}_i(+1)\log \hat{y}_i(+1) - \bar{y}_i(-1)\log \hat{y}_i(-1)$.

D. Iterative Risk Minimization

GRAB iteratively improves the quality of approximate prior $\hat{\pi}_\mathrm{p}$, which is the evidence for computing the approximate marginals of unlabeled nodes. GRAB takes two alternate steps at each iteration as shown in Algorithm 1: *marginalization* and *update* steps. The estimate $\hat{\pi}_\mathrm{p}$ is initialized to zero as we have no prior information, as discussed in Section III-B.

In the marginalization step, we run LBP to get approximate marginals of latent variables $\mathbf{z}$ given the current estimate $\hat{\pi}_\mathrm{p}$. The result is given as a belief matrix $\mathbf{B}$. In the update step, we find the best parameters θ^{new} of f that minimizes the objective function of Equation (10) given $\mathbf{B}$:

$$\theta^{\mathrm{new}} = \arg\min_{\theta} \tilde{\mathcal{L}}(\theta; \mathbf{B}). \quad (11)$$

We write only the belief matrix $\mathbf{B}$ as an input variable, as it is the only variable that changes through iterations.

Based on the new parameters θ^{new}, we update the approximate prior $\hat{\pi}_\mathrm{p}$ using the predictions of f:

$$\hat{\pi}_\mathrm{p}^{\mathrm{new}} = \frac{1}{|\mathcal{U}|}\sum_{i\in\mathcal{U}} \mathbb{I}[\hat{y}_i(+1; \theta^{\mathrm{new}}) > 0.5], \quad (12)$$

where $\mathbb{I}$ is an indicator function that returns 1 if the condition holds, and 0 otherwise. As a result, we leverage the knowledge of f with θ^{new} to improve the estimation $\hat{\pi}_{\mathrm{new}}$, which is used at the next iteration to get better estimates of marginals.

The iterations stop if the current $\tilde{\mathcal{L}}(\theta^{\mathrm{new}})$ is larger than that of the previous iteration. GRAB returns the trained classifier f with the estimated prior $\hat{\pi}_\mathrm{p}$ as a result.

E. Theoretical Analysis

We analyze GRAB theoretically in terms of its relation to the EM algorithm, and the time and space complexities.

1) Relation to the EM Algorithm: The EM algorithm [26] is an iterative approach to maximize the likelihood $p(\mathbf{y} \mid \mathbf{X}; \theta)$ of target variables $\mathbf{y}$ given input variables $\mathbf{X}$ and parameters θ by introducing latent variables $\mathbf{z}$. Each iteration of the EM algorithm consists of expectation and maximization steps.

The expectation (E) step computes the expectation of the log likelihood with respect to the conditional distribution of $\mathbf{z}$ given the current parameters $\theta^{(t)}$ at the t-th iteration:

$$Q(\theta \mid \theta^{(t)}) = \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z}|\mathbf{X},\mathbf{y},\theta^{(t)})}[\log p(\mathbf{y}, \mathbf{z} \mid \mathbf{X}, \theta)]. \quad (13)$$

The maximization (M) step finds new parameters that maximize the computed expectation:

$$\theta^{(t+1)} = \arg\max_{\theta} Q(\theta \mid \theta^{(t)}). \quad (14)$$

The convergence is checked for either the log likelihood or the parameters θ. The algorithm goes back to the E step if the convergence criterion is not satisfied, and returns the current parameters otherwise. The EM algorithm always improves the likelihood for observed data through the iterations [27].

The main objective of GRAB is to find the best parameters θ^* of the classifier f through the iterative optimization by the approximate marginals $\mathbf{B}$ and prior $\hat{\pi}_\mathrm{p}$. The iterative process of GRAB is similar to the EM algorithm as it introduces latent variables to model the unlabeled nodes. With this perspective, GRAB approximates the two distributions $p(\mathbf{z} \mid \mathbf{X}, \mathbf{y}, \theta^{(t)})$ and $p(\mathbf{y}, \mathbf{z} \mid \mathbf{X}, \theta)$ in Equation (13) with different marginalization to address the difficulty of PU learning.

First, the conditional distribution $p(\mathbf{z} \mid \mathbf{X}, \mathbf{y}, \theta^{(t)})$ given the current parameters $\theta^{(t)}$ is approximated by the multiplication of beliefs computed by the LBP algorithm with $\hat{\pi}_\mathrm{p}$:

$$p(\mathbf{z} \mid \mathbf{X}, \mathbf{y}, \theta^{(t)}) \approx \prod_{i\in\mathcal{U}} b_i(z_i). \quad (15)$$

Second, the distribution $p(\mathbf{y}, \mathbf{z} \mid \mathbf{X}, \theta)$ with new parameters θ is approximated by the classifier f, which is also considered as a marginalization function that gives the label distribution of each node based on all given information.

$$p(\mathbf{y}, \mathbf{z} \mid \mathbf{X}, \theta) \approx \prod_{i\in\mathcal{P}} \hat{y}_i(+1) \prod_{j\in\mathcal{U}} \hat{y}_j(z_j), \quad (16)$$

where $\hat{y}_i(u)$ is the prediction $f(\mathbf{X}, i; \theta)$ of f for node i being labeled as $u \in \{+1, -1\}$.

The marginalization assumption of Equations (15) and (16) leads to Lemma 1 and Theorem 1 that summarize the relation between GRAB and the EM algorithm.

Lemma 1. *Given the assumptions of Equations* (15) *and* (16)*, the objective function of GRAB in Equation* (10) *reduces to the expectation function Q of Equation* (13) *if $|\mathcal{P}| = |\mathcal{U}|$ and the loss function l is the cross entropy function: $l(\bar{y}_i, \hat{y}_i) = -\bar{y}_i(+1)\log \hat{y}_i(+1) - \bar{y}_i(-1)\log \hat{y}_i(-1)$.*

Proof. We derive $-\tilde{\mathcal{L}}(\theta)$ from Equation (13), because the goal of training is to minimize the objective function:

$$\begin{aligned}
&\mathbb{E}_{\mathbf{z}\sim p(\mathbf{z}\mid\mathbf{X},\mathbf{y},\theta^{(t)})}[\log p(\mathbf{y},\mathbf{z}\mid\mathbf{X},\theta)] \\
&= \sum_{\mathbf{z}} p(\mathbf{z}\mid\mathbf{X},\mathbf{y},\theta^{(t)})\log p(\mathbf{y},\mathbf{z}\mid\mathbf{X},\theta) \\
&\approx \sum_{\mathbf{z}} p(\mathbf{z}\mid\mathbf{X},\mathbf{y},\theta^{(t)})(\sum_{i\in\mathcal{P}}\log\hat{y}_i(+1)+\sum_{j\in\mathcal{U}}\log\hat{y}_j(z_j)) \\
&\approx \sum_{i\in\mathcal{P}}\log\hat{y}_i(+1)+\sum_{j\in\mathcal{U}}\sum_{z_j\in\pm 1} b_j(z_j)\log\hat{y}_j(z_j) \\
&= -\sum_{i\in\mathcal{P}} l(\bar{y}_i,\hat{y}_i)-\sum_{j\in\mathcal{U}} l(b_j,\hat{y}_j).
\end{aligned}$$

We prove the lemma by the assumptions of $|\mathcal{P}|=|\mathcal{U}|$ and the loss function, resulting in Equation (10). □

Theorem 1. *The iterative process of GRAB is a special case of the EM algorithm with the same assumptions as in Lemma 1 and assuming that the training of f finds the global optimum of new parameters as $\theta^{\text{new}}=\arg\min_\theta\tilde{\mathcal{L}}(\theta;\mathbf{B})$.*

Proof. The objective function of Equation (10) is the result of running the E step due to Lemma 1. Finding new parameters θ^{new} in Equation (11) corresponds to the M step of Equation (14). GRAB takes the two steps alternately at every iteration, which is the same as the EM algorithm, and thus the iterative process of GRAB is a special case of the EM algorithm. □

2) Time and Space Complexities: GRAB is scalable to large graphs, due to the linear scalability with the number of edges and nodes in the given graph G. For simplicity, we assume a GCN having r layers as the classifier f, where the number d of features of each node is the same in all layers.

Lemma 2. *The time complexity of GRAB (Algorithm 1) is*

$$O(n_{\text{o}}(n_{\text{b}}|\mathcal{E}|+n_f rd(|\mathcal{E}|+|\mathcal{V}|d))),$$

where n_{o} is the number of outer iterations in Algorithm 1, n_{b} and n_f are the number of LBP iterations and the number of epochs for training f, respectively, for each outer iteration.

Proof. The time complexity of LBP is $O(n_{\text{b}}|\mathcal{E}|)$, because $2|\mathcal{E}|$ messages are updated at each iteration, and each update is $O(1)$. The complexity of training f is $O(n_f rd(|\mathcal{E}|+|\mathcal{V}|d))$, which runs f for n_f times until the parameters converge. The two steps are run for the n_{o} iterations of GRAB. □

Lemma 3. *The space complexity of GRAB (Algorithm 1) is $O(rd^2+rd|\mathcal{V}|+|\mathcal{E}|))$.*

Proof. GRAB consists of marginalization and update steps. In the marginalization step, LBP uses $O(|\mathcal{V}|+|\mathcal{E}|)$ space to save the messages and beliefs. In the update step, the training of f requires storing the node features and weight matrices at all layers, using $O(rd^2+rd|\mathcal{V}|)$ space [28]. □

IV. Experimental Settings

We introduce our experimental settings including datasets, competitors, and the split of training and test data. All of our experiments are done at a workstation with GTX 1080 Ti.

TABLE II: Summary of datasets.

Name	Nodes	Edges	Features	Pos.	Neg.
Cora[1]	2,708	5,278	1,433	818	1,890
Citeseer[1]	3,327	4,552	3,703	701	2,626
Cora-ML[2]	2,995	8,158	2,879	857	2,138
WikiCS[3]	11,701	215,603	300	2,679	9,022
MMORPG[4]	6,312	68,012	136	298	401

[1] https://github.com/kimiyoung/planetoid
[2] https://github.com/abojchevski/graph2gauss
[3] https://github.com/pmernyei/wiki-cs-dataset
[4] Private to a company.

A. Datasets

We use five datasets in our experiments, which are summarized in Table II. The first four datasets have been used widely for node classification tasks. Cora, Citeseer, and Cora-ML are citation networks [29], [30] whose nodes represent scientific publications classified by the research areas and have bag-of-words feature vectors about their textual contents. WikiCS [31] consists of computer science articles in Wikipedia connected by hyperlinks. The classes represent the different branches of fields, and the features represent the contents of articles as the average of pre-trained GloVe word embeddings [32].

Since each dataset has multiple target labels, we treat the label with the largest number of nodes as positive and the rest as negative for binary classification. The resulting numbers of positive and negative nodes are reported in Table II with other statistics of datasets such as the number of features.

In addition to the public datasets, we perform experiments on a private dataset crawled from a massively multiplayer online role-playing game (MMORPG). In this dataset, fraudulent users use automatic scripts to quickly collect the game money for the purpose of exchanging it to real money, causing normal users to have negative experience. The nodes represent users, and the edges represent the interactions between game characters such as participating in the same party. Our goal is to detect fraudulent users in the graph by a classifier.

The MMORPG dataset is different from the public datasets in that the exact labeling of nodes is very difficult, especially for the negative label. This is because all fraudulent users act normally until they get caught, and thus a labeler cannot have evidence to conclude that a user is normal. On the other hand, the positive label can be assigned confidently to the fraudulent users whose usage of automatic scripts is detected. Thus, this dataset is a real-world example of graph-based PU learning, where the true class prior π_{p} is unknown. For the evaluation, we give negative labels to users who are highly likely to be normal based on the domain knowledge of the game.

B. Baselines

We compare GRAB with previous models for graph-based PU learning, including those for unsupervised representation learning. We adopt a graph convolutional network (GCN) as a base classifier for GRAB and most baselines.

TABLE III: The performance of GRAB and baselines, evaluated by the F1 score and accuracy over the unlabeled nodes. We perform two kinds of experiments: a) the prior π_{p} is unknown for all methods, and b) π_{p} is known only for the competitors. The F1 scores of zero represent that a model predicts all nodes as negative, and thus the recall is zero. GRAB outperforms all baselines in most cases, without using the knowledge of the prior π_{p}. OOM denotes the out of memory error.

Unknown Prior

Method	Cora		Citeseer		Cora-ML		WikiCS		MMORPG	
	F1 (%)	ACC (%)	F1 (%)	ACC (%)	F1 (%)	ACC (%)	F1 (%)	ACC (%)	F1 (%)	ACC (%)
GCN+CE	23.1±1.5	84.4±0.2	25.7±2.3	89.6±0.2	26.7±2.7	85.7±0.3	27.1±12.	88.9±0.9	24.3±9.2	76.7±1.5
GCN+PULP	40.2±1.7	86.1±0.2	37.1±3.0	90.3±0.4	38.3±1.3	86.4±0.2	30.3±7.7	87.9±1.1	33.7±12.	78.6±2.1
GCN+URE	42.4±1.5	86.8±0.2	39.1±2.0	90.6±0.2	49.1±3.6	88.6±0.5	0.0±0.0	87.1±0.0	28.5±10.	77.5±1.6
GCN+NRE	70.1±1.6	91.4±0.3	61.8±1.9	92.8±0.2	72.9±2.0	92.5±0.4	0.0±0.0	87.1±0.0	28.5±10.	77.5±1.6
Node2Vec	53.3±2.1	87.0±0.6	29.7±2.2	88.9±0.3	57.4±1.7	88.8±0.4	76.4±0.7	**94.8±0.1**	0.0±0.0	72.9±0.0
ARGVA	53.7±16.	88.1±2.4	22.7±30.	89.8±2.2	57.1±21.	89.7±2.5	0.0±0.0	87.1±0.0	3.6±8.0	73.5±1.2
LSDAN	52.3±3.9	87.5±0.6	18.4±25.	89.8±2.2	6.3±17.	83.9±1.6	OOM	OOM	OOM	OOM
GRAB (ours)	**80.4±0.2**	**93.0±0.1**	**69.7±0.4**	**92.9±0.1**	**85.0±0.1**	**94.9±0.0**	**79.1±1.4**	93.9±0.5	**94.6±0.5**	**97.2±0.3**

Known Prior (except for GRAB)

Method	Cora		Citeseer		Cora-ML		WikiCS		MMORPG	
	F1 (%)	ACC (%)	F1 (%)	ACC (%)	F1 (%)	ACC (%)	F1 (%)	ACC (%)	F1 (%)	ACC (%)
GCN+CE	23.0±1.9	84.3±0.2	25.8±2.2	89.6±0.2	26.8±3.0	85.8±0.4	29.8±8.4	89.4±0.7	24.2±9.2	76.7±1.5
GCN+PULP	40.2±1.7	86.1±0.2	37.1±3.1	90.3±0.4	38.3±1.3	86.4±0.2	28.9±5.9	87.5±0.7	33.8±12.	78.6±2.1
GCN+URE	50.9±0.8	88.0±0.1	42.6±1.7	90.9±0.2	54.6±1.8	89.4±0.3	26.9±34.	89.5±3.2	89.0±2.5	94.7±1.1
GCN+NRE	76.7±0.9	92.7±0.2	66.2±1.1	**93.2±0.2**	80.0±0.6	94.1±0.2	26.9±34.	89.5±3.2	**95.3±1.9**	**97.6±1.0**
Node2Vec	58.1±1.5	87.1±0.4	32.7±2.2	88.4±0.5	62.3±1.9	89.1±0.6	**81.0±0.3**	**95.5±0.1**	81.4±2.1	91.2±1.0
ARGVA	62.3±9.4	89.2±1.9	17.9±29.	89.5±2.2	50.3±28.	88.7±3.1	0.0±0.0	87.1±0.0	89.3±4.4	94.5±2.2
LSDAN	63.5±4.1	89.4±1.0	47.0±19.	91.2±1.3	63.4±3.7	90.1±0.7	OOM	OOM	OOM	OOM
GRAB (ours)	**80.4±0.2**	**93.0±0.1**	**69.7±0.4**	92.9±0.1	**85.0±0.1**	**94.9±0.0**	79.4±1.0	93.9±0.5	94.6±0.5	97.2±0.3

- **GCN+CE** uses the cross entropy (CE) loss for training a GCN, treating all unlabeled nodes as negative.
- **GCN+PULP** uses PU-LP [12] to find relatively positive nodes from the unlabeled ones. The remaining nodes are assumed to be negative in the loss function.
- **GCN+URE** uses the unbiased risk estimator (URE) [17], [18], which is an objective function specifically designed for PU learning to estimate the unknown negative risk.
- **GCN+NRE** uses the non-negative risk estimator (NRE) [3] instead of URE, forcing the objective function to be positive and increasing the stability of training.
- **Node2Vec** [33] is a random walk-based model for learning low-dimensional representations of nodes.
- **ARGVA** [34] utilizes a variational autoencoder to learn the embeddings of nodes with adversarial regularization, based on the graph structure and node features.
- **LSDAN** [13] is the state-of-the-art model that improves GCN by the long-short distance attention that effectively combines the information of multi-hop neighbors.

For unsupervised models such as Node2Vec and ARGVA, which require explicit classifiers on top of the generated node embeddings, we use a linear classifier trained by NRE. This is because NRE shows better performance than the typical loss functions in many PU learning problems.

For the experiments in Sections V-A to V-D, we adopt the default hyperparameter settings in GRAB and all baselines, as we have no validation data for hyperparameter tuning due to the non-existence of negative labels. For GCN models, we set the number of layers to 2 and the size of hidden layers to 16. We set $\epsilon = 0.9$, which determines the degree of homophily of our Markov Network modeling in Equation (5). We train each model using the Adam optimizer [35] for 1,000 epochs and stop the training if the training loss starts to increase. The number of epochs is set to 2,000 only in the WikiCS dataset, where the loss fluctuates a lot during the training. We use the sigmoid function [3] as our loss function l.

Although the above setting correctly reflects the real-world PU learning, we additionally report the performance of GRAB and all baseline methods when negative labels are given for the validation purpose. The result is given in Section V-E.

C. Evaluation

For each dataset, we use 50% of all positive nodes as the set $\mathcal{P}$ of observed positive nodes, and treat the rest as unobserved positive nodes $\mathcal{P}_{\mathrm{t}}$. All negative nodes are treated as unobserved and denoted by $\mathcal{N}_{\mathrm{t}}$, as we assume PU learning. Then, we aim to predict the label of each node in $\mathcal{U} = \mathcal{P}_{\mathrm{t}} \cup \mathcal{N}_{\mathrm{t}}$ as test data by training a classifier for the labeled nodes $\mathcal{P}$ and unlabeled nodes $\mathcal{U}$; all nodes are accessible during training, but the labels of only $\mathcal{P}$ are observable as training data. If a dataset contains nodes whose true labels are unknown even for the test, which is the case of MMORPG, we include such nodes in $\mathcal{U}$ but exclude them from calculating evaluation metrics.

We use the F1 score as the main evaluation metric, but report also the classification accuracy for a thorough evaluation. We run all experiments ten times with different random seeds and report the average and standard deviation.

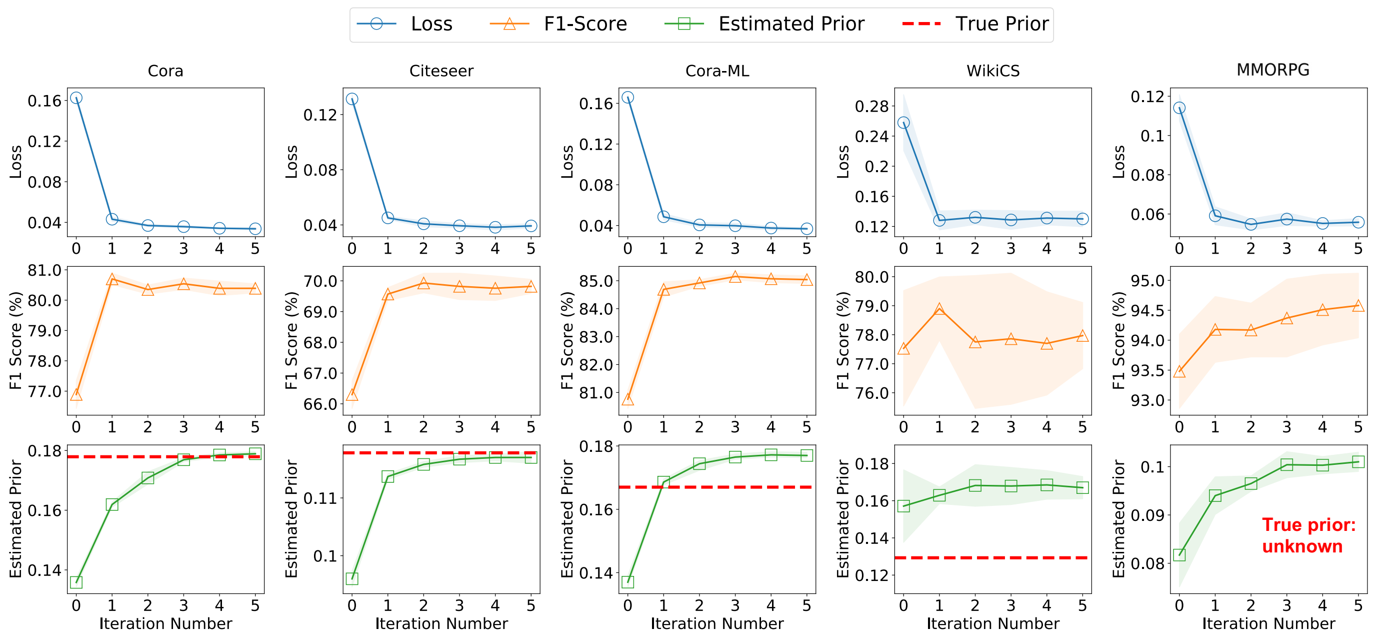

Fig. 2: The performance of GRAB changing through the iterative optimization. The three rows of figures represent the value of the objective function $\hat{\mathcal{L}}(\theta)$, the F1 score, and the estimated prior $\hat{\pi}_\mathrm{p}$, respectively. The losses and F1 scores converge quickly as the iterations proceed, while the approximate priors become close to the true priors.

A notable difference between GRAB and previous models is that we do not require the true prior π_p, which is defined as $\pi_\mathrm{p} = \mathcal{P}_\mathrm{t}/(\mathcal{P}_\mathrm{t}+\mathcal{N}_\mathrm{t})$ in our experiments. Thus, we conduct two kinds of experiments for the baselines a) with and b) without the prior π_p. When the prior is not given, we use the number of labeled positive nodes over the number of all nodes as an approximate prior for baselines, i.e., $\pi_\mathrm{p} \approx |\mathcal{P}|/|\mathcal{V}|$, which is the only observable statistic in our experiments and most PU learning problems where $\mathcal{P}_\mathrm{t}$ and $\mathcal{N}_\mathrm{t}$ are not accessible.

V. Experimental Results

We perform various experiments on the real world datasets to answer the following questions:

Q1. Classification accuracy (Section V-A). How accurate is GRAB compared to the baseline methods? How do the results change with different ratios of positive nodes?

Q2. Effect of iterative learning (Section V-B). Does GRAB predict the unknown value of π_p accurately? How do the accuracy and loss change during the iterations?

Q3. Scalability (Section V-C). Does GRAB show linear time complexity with the number of edges in the graph?

Q4. Parameter sensitivity (Section V-D). How does the accuracy of GRAB change with different hyperparameters for the classifier and potential functions?

Q5. Accuracy with validation data (Section V-E). Assuming that negative labels exist for the validation purpose, does GRAB still outperform the baselines?

A. *Classification Accuracy (Q1)*

We compare GRAB with the baselines for graph-based PU learning in Table III, with and without the true class prior π_p; GRAB does not use π_p in both cases. GRAB shows the state-of-the-art performance in most cases, even when π_p is given only to the competitors.

Most of the baselines completely fail in MMORPG when no prior information is given, since few labeled examples are given in this dataset; the ratio of observed positive nodes over all nodes is only 4.7%, which is much smaller than those of the other datasets. GRAB performs well even in this case, due to the robust estimation of approximate prior $\hat{\pi}_\mathrm{p}$.

We summarize the performance of GRAB and the baselines with different ratios of observed positive nodes to unlabeled ones in Fig. 1. We gradually decrease the ratio r_p of positive nodes from 0.5, which is the same as in Table III, to 0.2. The problem becomes more difficult with smaller r_p.

Fig. 1 shows that GRAB presents the best performance among other models for different values of r_p. All baseline models fail completely when $r_\mathrm{p} = 0.2$, because of the shortage of observed labels, but GRAB shows a marginal decrease of the F1 score even in this case. This is because GRAB assigns soft approximate labels, which evolve via iterations, to the unlabeled nodes for the training of a GCN classifier.

B. *Effect of Iterative Learning (Q2)*

GRAB is an iterative algorithm that gradually minimizes the objective function $\hat{\mathcal{L}}$ as shown in Algorithm 1. We study how the performance of GRAB changes through the iterations in Fig. 2 with respect to the value of $\hat{\mathcal{L}}(\theta)$, the F1 score for unlabeled nodes, and the estimated prior $\hat{\pi}_\mathrm{p}$. The performance of the classifier f improves through the iterations in terms of both the F1 score and the objective function. GRAB converges in few iterations, demonstrating its stability in various types of

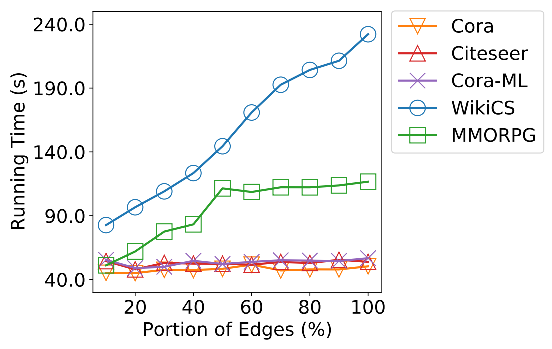

Fig. 3: The running time of GRAB on induced subgraphs for each dataset. In WikiCS and MMORPG, which are larger than other datasets, the time increases linearly with the number of edges. In the other datasets, the time does not make a notable difference, due to the parallel computation of GPUs.

datasets, even though the best iteration number is different for each graph; for instance, GRAB stops at iteration 1 at WikiCS, while GRAB stops at iteration 2 for MMORPG.

The estimated prior $\hat{\pi}_{\mathrm{p}}$ starts from zero, because we have no information about the unlabeled nodes before the iterations begin. The value of $\hat{\pi}_{\mathrm{p}}$ at iteration 0 in Fig. 2 represents the estimation from the initial GCN trained by the objective function $\tilde{\mathcal{L}}$ that is generated with the assumption of $\hat{\pi}_{\mathrm{p}} = 0$. Then, $\hat{\pi}_{\mathrm{p}}$ becomes close to the true prior π_{p} through iterations, which is represented as the red dashed lines. The estimation is almost accurate in Cora and Citeseer even though GRAB has no prior knowledge about the class distribution. This implies that the graph structure and feature vectors make meaningful clusters of nodes that are related to their classes, and GRAB extracts such information effectively through the iterations.

C. Scalability (Q3)

We validate the linear time complexity of GRAB stated in Lemma 2 by measuring the computational time on subgraphs with different sizes in Fig. 3. We select a random subset of nodes for each dataset, whose induced subgraph contains $p|\mathcal{E}|$ edges, where p varies in $\{0.1, 0.2, \cdots, 0.9\}$. Thus, we create nine subgraphs for each dataset.

The figure shows that the computational time of GRAB is sublinear with the number of edges in a graph. In WikiCS and MMORPG, which are large datasets that contain 215,603 and 68,012 edges, respectively, the time increases linearly with the number of edges. In the other datasets having at most 8,158 edges, the computational time remains similar even with the different number of edges. This is because these datasets are small enough to be processed in parallel by GPUs.

D. Parameter Sensitivity (Q4)

We investigate the accuracy of GRAB with different values of hyperparameters in Fig. 4. We tune the dimension h of GCN hidden layers in $[16, 128]$, the number r of GCN layers in $[2, 5]$, and the degree ϵ of homophily for the Markov network modeling in $[0.6, 0.9]$. The figure shows that the performance

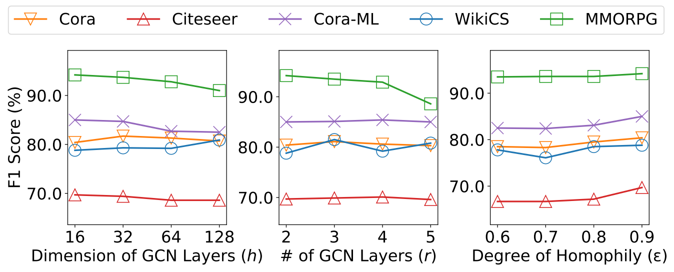

Fig. 4: The accuracy of GRAB with different hyperparameters: the dimension of GCN layers, the number of GCN layers, and the degree of homophily for the Markov network modeling. GRAB produces a superior performance in most hyperparameter settings, showing its robustness and stability.

of GRAB is robust to such hyperparameters, producing stable and consistent results in all cases.

Still, the results suggest that fewer parameters lead to better performance in general. For instance, the accuracy decreases in most datasets when $h = 128$ or $r \geq 4$. This is because we deal with the PU learning problem where a classifier with many parameters is expected to overfit to the positive observations. It is also observed that larger ϵ leads to better performance in all cases, because large ϵ effectively propagates the observed information of $\mathcal{P}$ to the entire graph, increasing the degree of correlation between adjacent nodes.

E. Accuracy with Validation Data (Q5)

In real world PU learning, there is no validation step for tuning the hyperparameters since we have no negative nodes during training; using only positive ground truth labels in validation leads to learning a trivial classifier that always outputs ‘positive’. That is the reason we performed all experiments in Sections V-A to V-D with fixed hyperparameters as discussed in Section IV-B. Although such setting exactly reflects the real world PU learning, we perform an additional study assuming that a few negative labels, in addition to the observed positive labels, are given for the validation purpose.

Specifically, we search the dimension of GCN layers in {16, 32, 64, 128} and the number of GCN layers in {2, 3, 4} for models that use GCN classifiers including GRAB. For Node2Vec and ARGVA, which are designed for unsupervised representation learning, we tune the size of embeddings in {16, 32, 64, 128}. We also generalize the linear classifier used for Node2Vec and ARGVA to multilayer perceptrons with the number of layers in {2, 3, 4}. We use 10% of all positive nodes for validation, 50% for training, and 40% for test. We use 20% of all negative nodes for validation and 80% for test. The true prior π_{p} is unknown for all approaches.

Table IV compares the performance of all models. Every method performs better than in Table III, due to the existence of validation data. Still, GRAB consistently produces the best performance among all methods.

TABLE IV: The performance of GRAB and baselines with tuned hyperparameters, evaluated by the F1 score and accuracy over the unlabeled nodes. The hyperparameters are tuned assuming that there exist negative labels, in addition to positive labels, for the validation purpose. GRAB outperforms all baseline models even in this case. OOM denotes the out of memory error.

Performance with Hyperparameter Tuning										
Method	Cora		Citeseer		Cora-ML		WikiCS		MMORPG	
	F1 (%)	ACC (%)	F1 (%)	ACC (%)	F1 (%)	ACC (%)	F1 (%)	ACC (%)	F1 (%)	ACC (%)
GCN+CE	69.6±1.4	91.2±0.3	56.1±3.6	92.4±0.4	76.0±0.9	93.3±0.2	67.5±4.8	93.2±0.7	43.3±8.6	80.5±1.8
GCN+PULP	76.6±1.4	92.7±0.3	67.8±1.6	**93.5±0.2**	81.4±0.3	94.3±0.1	56.5±5.8	90.5±0.8	72.5±6.2	88.4±2.0
GCN+URE	65.3±1.5	90.4±0.3	54.0±1.6	92.1±0.2	58.1±1.6	90.0±0.3	32.4±17.	67.5±25.	48.9±19.	67.3±21.
GCN+NRE	75.4±0.5	92.3±0.1	64.6±0.3	93.1±0.1	75.1±0.3	93.0±0.1	24.0±3.0	46.8±29.	53.8±18.	64.7±23.
Node2Vec	52.8±3.3	86.8±0.5	26.2±2.2	88.9±0.3	59.1±1.7	89.0±0.3	80.3±0.7	**95.4±0.2**	33.7±37.	77.8±11.
ARGVA	61.9±2.5	88.4±1.3	59.2±7.2	91.9±1.3	60.1±3.5	88.9±0.6	25.5±12.	72.3±17.	40.7±7.2	79.0±1.3
LSDAN	62.1±1.0	89.3±0.2	25.4±0.0	45.5±0.0	70.5±1.9	91.8±0.4	OOM	OOM	OOM	OOM
GRAB (ours)	**81.5±0.8**	**93.2±0.4**	**73.4±1.7**	92.8±0.8	**84.5±0.2**	**94.5±0.1**	**83.1±0.8**	**95.4±0.1**	**98.0±0.5**	**98.9±0.2**

VI. Conclusion

In this work, we have proposed GRAB, an accurate method for PU learning on graph-structured data without class prior. The objective function of GRAB does not require the true prior by modeling unlabeled nodes in a graph as latent variables and using the expectation of their joint distribution as pseudo labels for training a classifier. For the accurate modeling of the joint distribution, GRAB infers the relationships between variables from the graph by assuming it as a pairwise Markov network. Experiments on five real-world datasets show that GRAB achieves the state-of-the-art performance in all cases even when all baselines fail to learn any knowledge from the limited observations. Future works include extending GRAB to edge-attributed graphs that provide rich information about the relations between nodes.

Acknowledgment

This research is supported from NCSOFT Co. U Kang is the corresponding author.

References

[1] H. Yu, J. Han, and K. C. Chang, "PEBL: positive example based learning for web page classification using SVM," in *KDD*, 2002.

[2] M. Li, S. Pan, Y. Zhang, and X. Cai, "Classifying networked text data with positive and unlabeled examples," *Pattern Recognit. Lett.*, 2016.

[3] R. Kiryo, G. Niu, M. C. du Plessis, and M. Sugiyama, "Positive-unlabeled learning with non-negative risk estimator," in *NIPS*, 2017.

[4] S. Nandanwar and M. N. Murty, "Structural neighborhood based classification of nodes in a network," in *KDD*, 2016.

[5] Y. Ma, S. Wang, C. C. Aggarwal, and J. Tang, "Graph convolutional networks with eigenpooling," in *KDD*, 2019.

[6] Y. Wang, W. Wang, Y. Liang, Y. Cai, J. Liu, and B. Hooi, "Nodeaug: Semi-supervised node classification with data augmentation," in *KDD*, 2020.

[7] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR*, 2017.

[8] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *ICLR*, 2018.

[9] P. Velickovic, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm, "Deep graph infomax," in *ICLR*. OpenReview.net, 2019.

[10] C. Zhang, D. Ren, T. Liu, J. Yang, and C. Gong, "Positive and unlabeled learning with label disambiguation," in *IJCAI*, 2019.

[11] S. Yu and C. Li, "PE-PUC: A graph based pu-learning approach for text classification," in *MLDM*, 2007.

[12] S. Ma and R. Zhang, "PU-LP: A novel approach for positive and unlabeled learning by label propagation," in *ICMEW*, 2017.

[13] M. Wu, S. Pan, L. Du, I. W. Tsang, X. Zhu, and B. Du, "Long-short distance aggregation networks for positive unlabeled graph learning," in *CIKM*, 2019.

[14] J. Bekker and J. Davis, "Learning from positive and unlabeled data: a survey," *Mach. Learn.*, 2020.

[15] K. Jaskie and A. Spanias, "Positive and unlabeled learning algorithms and applications: A survey," in *IISA*, 2019.

[16] X. Li and B. Liu, "Learning from positive and unlabeled examples with different data distributions," in *ECML*, 2005.

[17] M. C. du Plessis, G. Niu, and M. Sugiyama, "Analysis of learning from positive and unlabeled data," in *NIPS*, 2014.

[18] M. Du Plessis, G. Niu, and M. Sugiyama, "Convex formulation for learning from positive and unlabeled data," in *ICML*, 2015.

[19] J. Yoo, S. Jo, and U. Kang, "Supervised belief propagation: Scalable supervised inference on attributed networks," in *ICDM*, 2017.

[20] J. Yoo, U. Kang, M. Scanagatta, G. Corani, and M. Zaffalon, "Sampling subgraphs with guaranteed treewidth for accurate and efficient graphical inference," in *WSDM*, 2020.

[21] M. McGlohon, S. Bay, M. G. Anderle, D. M. Steier, and C. Faloutsos, "SNARE: a link analytic system for graph labeling and risk detection," in *KDD*, 2009.

[22] J. Yoo, H. Jeon, and U. Kang, "Belief propagation network for hard inductive semi-supervised learning," in *IJCAI*, 2019.

[23] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE*, 2021.

[24] Z. Zhang, P. Cui, and W. Zhu, "Deep learning on graphs: A survey," *CoRR*, 2018.

[25] D. Koller and N. Friedman, *Probabilistic Graphical Models - Principles and Techniques*. MIT Press, 2009.

[26] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal Statistical Society: Series B (Methodological)*, 1977.

[27] K. P. Murphy, *Machine learning - a probabilistic perspective*, ser. Adaptive computation and machine learning series. MIT Press, 2012.

[28] W. Chiang, X. Liu, S. Si, Y. Li, S. Bengio, and C. Hsieh, "Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks," in *KDD*, 2019.

[29] Z. Yang, W. W. Cohen, and R. Salakhutdinov, "Revisiting semi-supervised learning with graph embeddings," in *ICLR*, 2016.

[30] A. Bojchevski and S. Günnemann, "Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking," in *ICLR*, 2018.

[31] P. Mernyei and C. Cangea, "Wiki-cs: A wikipedia-based benchmark for graph neural networks," *CoRR*, 2020.

[32] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *EMNLP*, 2014.

[33] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *KDD*, 2016.

[34] S. Pan, R. Hu, G. Long, J. Jiang, L. Yao, and C. Zhang, "Adversarially regularized graph autoencoder for graph embedding," in *IJCAI*, 2018.

[35] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR*, 2015.