PU Learning and Data Augmentation on Graphs

Jaemin Yoo

Dept. of Computer Science & Engineering

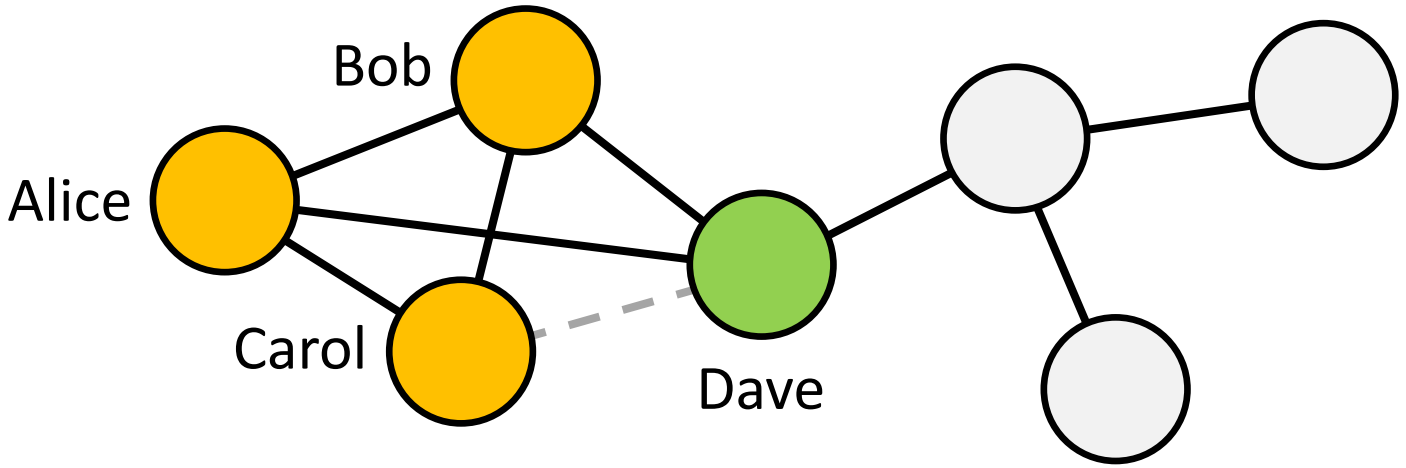
Seoul National University

Outline

- **Introduction**
- Graph-based PU learning
- Graph augmentation
- Conclusion

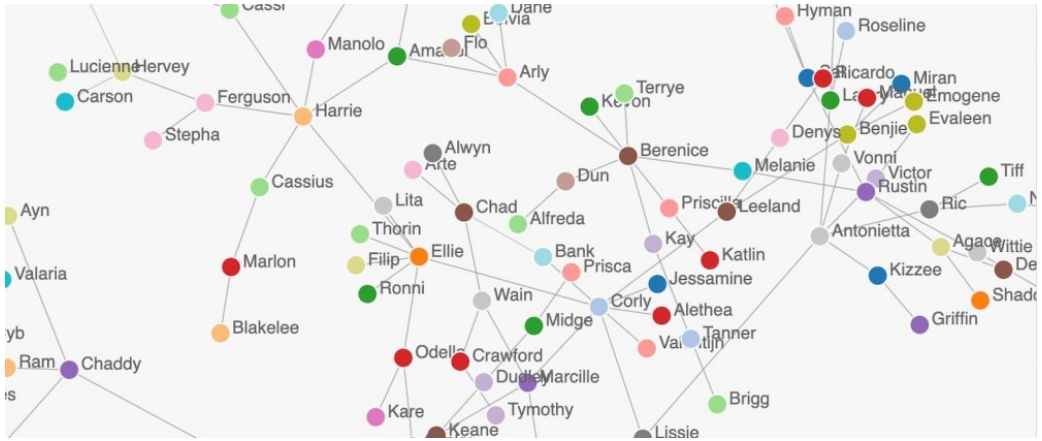
Graphs (1)

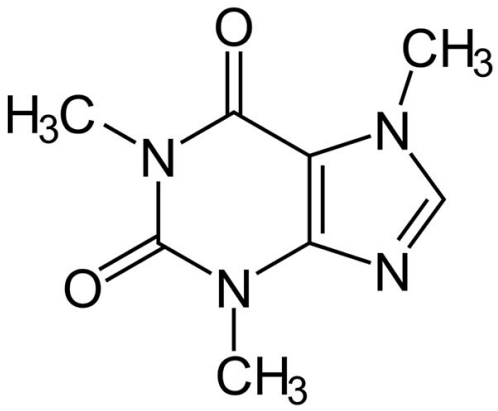
- **Graph** is a data structure for modeling relationships in data
  - Graph consists **nodes** that are connected by **edges**
- Graph provides meaningful insights and observations of data
  - Alice, Bob, and Carol are strongly connected to each other



Graphs (2)

- Graph-structured data are common in real-world applications
  - **Social network** represents the friendships of people
  - **Review graph** represents user preferences for movies
  - **Chemical compound** consists of interactions between elements



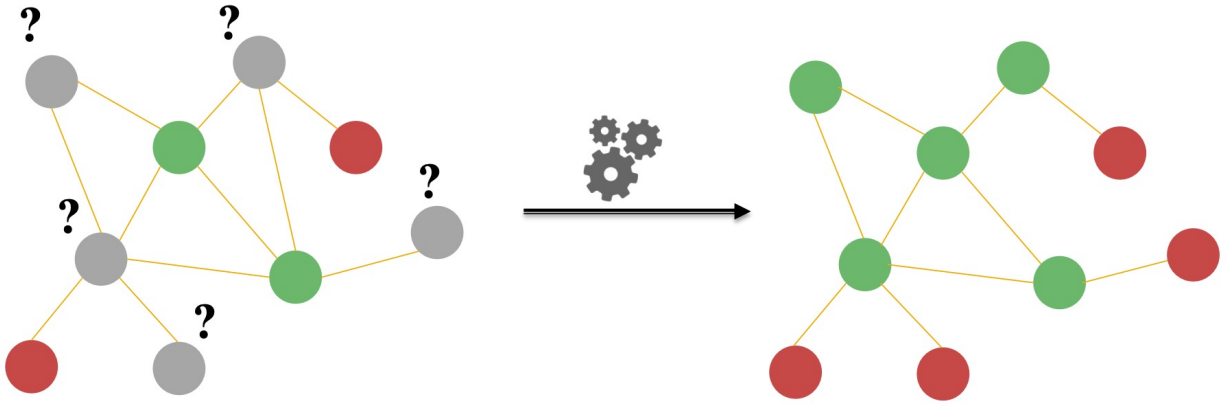


Machine Learning on Graphs

- **Graph neural networks (GNN)** have been actively studied
  - They have shown great performance in various domains
- This talk is about improving the **generalizability** of GNNs
  - Toward challenging scenarios where training data are insufficient
- Let's consider two major topics in graph machine learning
  - **Node classification** is to predict the labels of nodes in a graph
  - **Graph classification** is to predict the labels of graphs

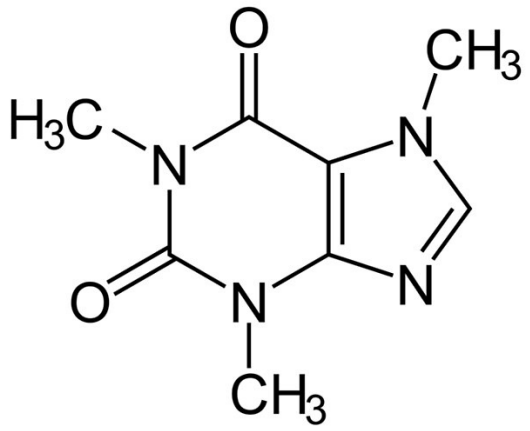
Node Classification

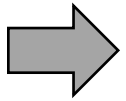
- **Given** a graph G and the labels of some nodes $\mathcal{V}_{\text{train}}$
  - Each node i has a feature vector $\mathbf{x}_i$
- **Problem** is to predict the labels of remaining nodes $\mathcal{V}_{\text{test}}$
- Real-world applications:
  - Finding malicious users in a social network
  - Categorizing movies in a streaming service



Graph Classification

- **Given** a set $\mathcal{G}$ of graphs each of which has a label y
  - Each graph consists of nodes, edges, and node attributes
- **Problem** is to predict the labels of new graphs not in $\mathcal{G}$
- Real-world applications:
  - Predicting the toxicity of a chemical compound
  - Categorizing the property of a social group



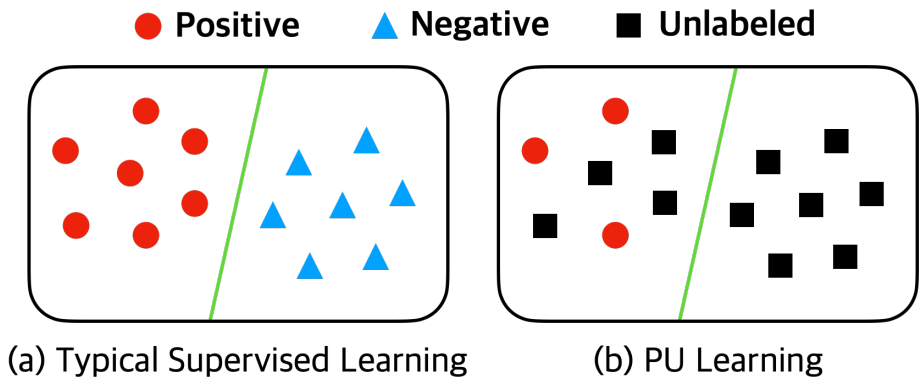


Toxic?

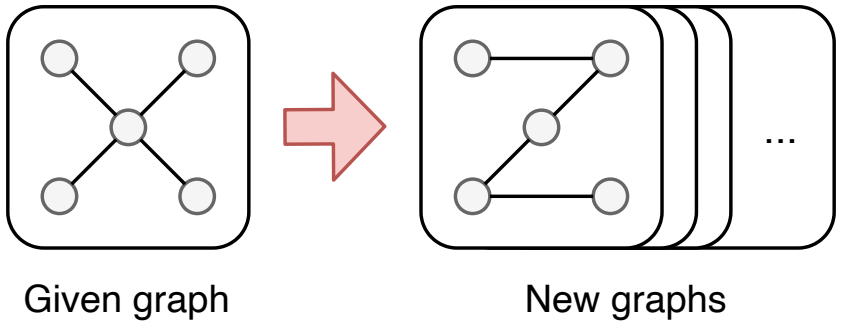
Not Toxic?

Overview

- This talk introduces two our recent works related to the problems
- **GRAB** [ICDM'21]
  - Node classification when **no negative labels** are observed



- **NodeSam & SubMix** [WWW'22]
  - **Graph augmentation** to improve the accuracy of graph classifiers
    - Increases the amount of training data

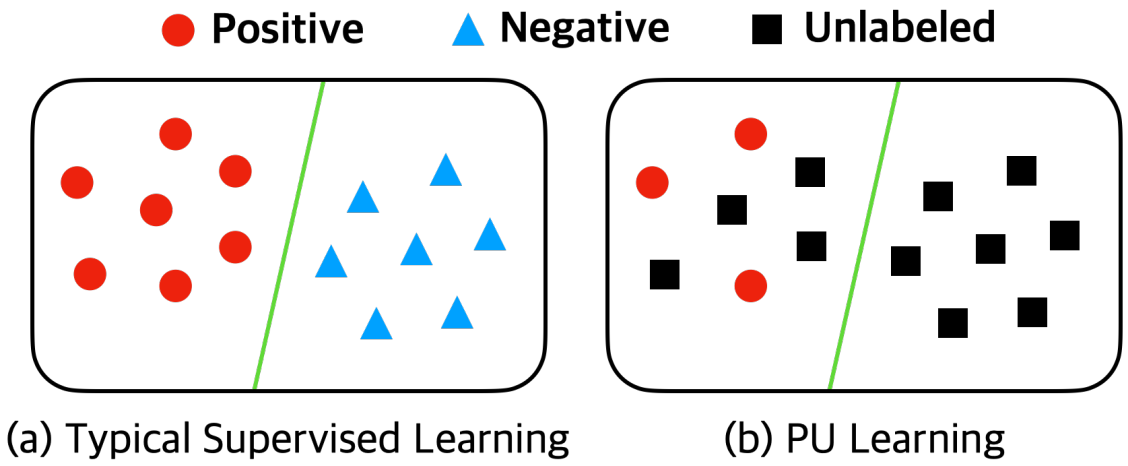


Outline

- Introduction
- **Graph-based PU learning**
  - **Research Motivation**
  - Proposed Method: GRAB
  - Experiments
  - Summary
- Graph augmentation
- Conclusion

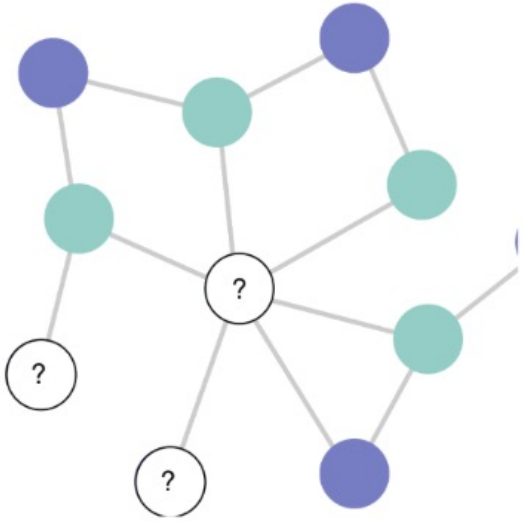
PU Learning

- **Positive-unlabeled (PU) learning** is a challenging problem
  - Consider the problem of binary classification of $\mathcal{P}$ vs. $\mathcal{N}$
  - In PU learning, negative labels are **unseen** during training
    - Decision boundary should be drawn from positive and unlabeled samples



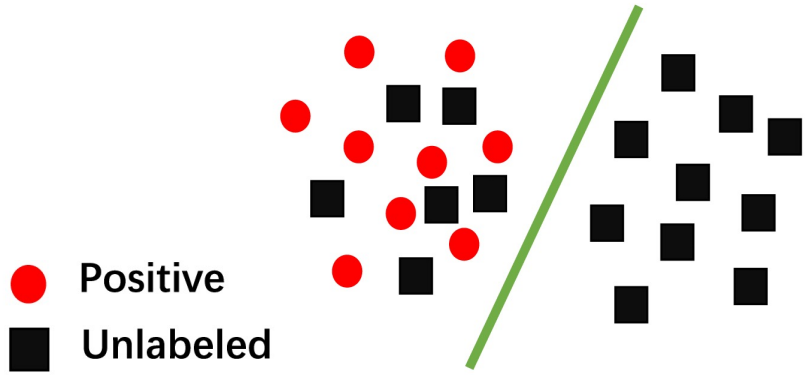
Graph-based PU Learning

- We solve PU learning on graph-structured data
  - Target examples are connected as a graph
- **Graph-based PU learning** is common
  - We are given a social network of 1,000 users
  - We have detected 100 fraud users
  - Are the remaining 900 users all normal?
  - **The answer is NO;** they are **unlabeled**, not negative



Class Prior

- Previous works assume that the **class prior** π_{p} is known
  - π_{p} is the ratio of positive examples among all unlabeled examples
- π_{p} provides **rich information** to graph-based PU learning
  - Assume that $|\mathcal{U}| = 16$ and $\pi_{\mathrm{p}} = 0.38$
    - Then, we know that exactly 6 nodes in $\mathcal{U}$ are positive



$\pi_{\mathrm{p}}|\mathcal{U}| = 0.38 \times 16 \approx 6$

We can draw a boundary after selecting the 6 nodes closest to $\mathcal{P}$

PU Learning without Class Prior

- However, π_{p} is **not available** in most real-world problems
  - E.g., what is the ratio of fraud users in a social network?
- Naïve guess of the value of π_{p} can result in a failure of training

Research Question

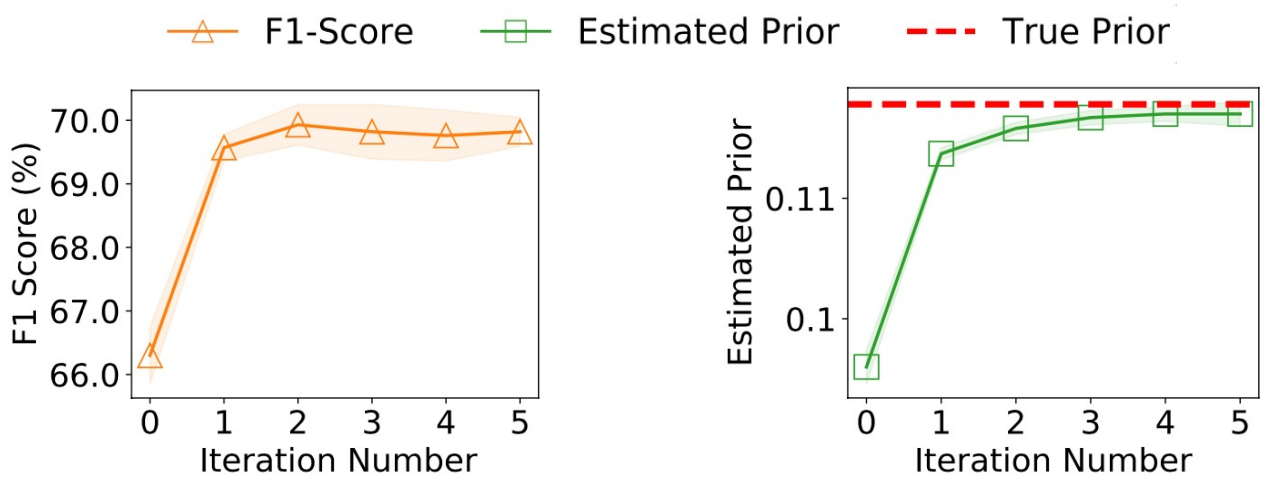
How can we solve graph-based PU learning without π_{p}?

Outline

- Introduction
- **Graph-based PU learning**
  - Research Motivation
  - **Proposed Method: GRAB**
  - Experiments
  - Summary
- Graph augmentation
- Conclusion

GRAB

- We propose **GRAB** for training a GNN classifier in PU learning
  - **Idea 1.** Treat the unlabeled nodes as latent variables
  - **Idea 2.** Model the given graph as a pairwise Markov network
  - **Idea 3.** Train the classifier through EM-like iterations



Objective Function

- We propose the following objective function to minimize:

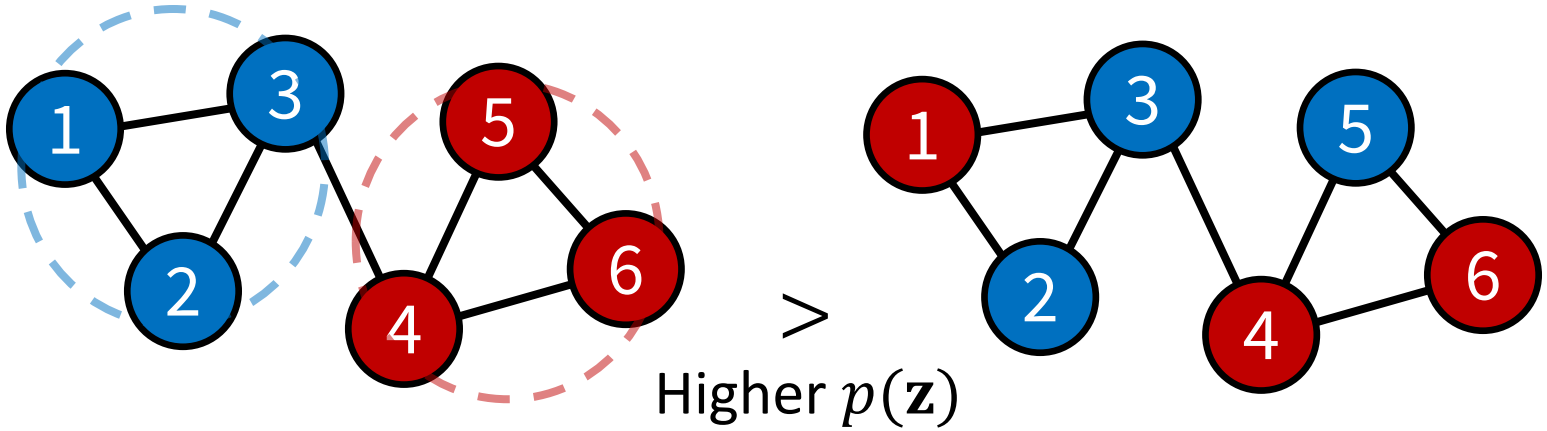
$$\mathcal{L}(\theta; \mathbf{X}, \mathbf{y}, \mathcal{P}, \mathcal{U}) = \frac{1}{|\mathcal{P}|} \sum_{i \in \mathcal{P}} (-\log \hat{y}_i(+1)) \quad \text{Positive part}$$

$$+ \mathbb{E}_{\mathbf{z} \sim p(\mathbf{z}|\mathbf{X}, \mathbf{y})} \left[\frac{1}{|\mathcal{U}|} \sum_{j \in \mathcal{U}} (-\log \hat{y}_j(z_j)) \right], \quad \text{Unlabeled part}$$

  - Positive nodes should be predicted as **positive**
  - Unlabeled nodes should be predicted following their **distribution** $p(\mathbf{z}|\mathbf{X}, \mathbf{y})$
  - The challenge is to accurately model $p(\mathbf{z}|\mathbf{X}, \mathbf{y})$ from the graph

Probabilistic Modeling of Graphs

- We model the given graph G as a **pairwise Markov network**
  - **Assumption:** Adjacent nodes are likely to have the same state
  - $p(\mathbf{z})$ becomes large if the latent assignment $\mathbf{z}$ satisfies the **homophily**



Probabilistic Inference

- The probabilistic modeling allows us to run **probabilistic inference**
  - We are given a current estimate $\hat{\pi}_{\mathrm{p}}$ of the unknown class prior π_{p}
  - We can marginalize $p(\mathbf{z}|\mathbf{X},\mathbf{y})$ as follows:

$$p(\mathbf{z}|\mathbf{X},\mathbf{y}) \approx \prod_i p_i(z_i|\mathbf{X},\mathbf{y}) = \mathrm{Marginalize}\left(G, \hat{\pi}_{\mathrm{p}}\right)$$

  - This assigns a pseudo label $p_i(z_i|\mathbf{X},\mathbf{y})$ to each unlabeled node i
    - By considering the correlations between nodes with the Markov assumption

New Objective Function

- We rewrite the objective function with the result of inference:

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

Unlabeled part

  - l is a generalized loss function, and $\bar{y}_i$ is a one-hot label
  - The joint distribution of $\mathbf{Z}$ has changed to the **marginalized belief** b_j
  - We now have a **tractable** objective function for the training of a classifier

Iterative Optimization

- GRAB runs marginalization and update steps alternately
  - **Initialization**
    - $f \leftarrow$ Initialize a graph convolutional network (GCN) classifier
    - $\hat{\pi}_{\mathrm{p}} \leftarrow$ Initialize a class prior as 0, i.e., all unlabeled nodes are negative
  - **Marginalization step**
    - $\mathbf{B} \leftarrow$ Run probabilistic inference based on the current $\hat{\pi}_{\mathrm{p}}$
  - **Update step**
    - $f \leftarrow$ Train a new classifier by using $\mathbf{B}$ as answers for unlabeled nodes
    - $\hat{\pi}_{\mathrm{p}} \leftarrow$ Update the class prior based on the prediction of the current f

Outline

- Introduction
- **Graph-based PU learning**
  - Research Motivation
  - Proposed Method: GRAB
  - **Experiments**
  - Summary
- Graph augmentation
- Conclusion

Datasets

- We run experiments on 5 datasets from different domains
  - 4 of them are public datasets for node classification
  - MMORPG is a private dataset provided by NCSOFT
    - The problem is to classify each game character into a **normal** user or a **bot**

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

Experimental Setup

- We assume 50% of all positive nodes are observed
  - The remaining 50% positive and all negative nodes are **unlabeled**
- We compare GRAB with the following baselines:
  - **Node representation learning** [KDD'16, IJCAI'18]
  - **Graph-based PU learning** [ICMEW'17, CIKM'19]
  - **General PU learning** (with GNNs) [ICML'15, NIPS'17]

Classification Accuracy

- **Q1.** How accurate is GRAB in graph-based PU learning?
- **A.** GRAB outperforms all baselines in most cases
  - The baselines fail in many cases when π_p is unknown (and naively assumed)

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**

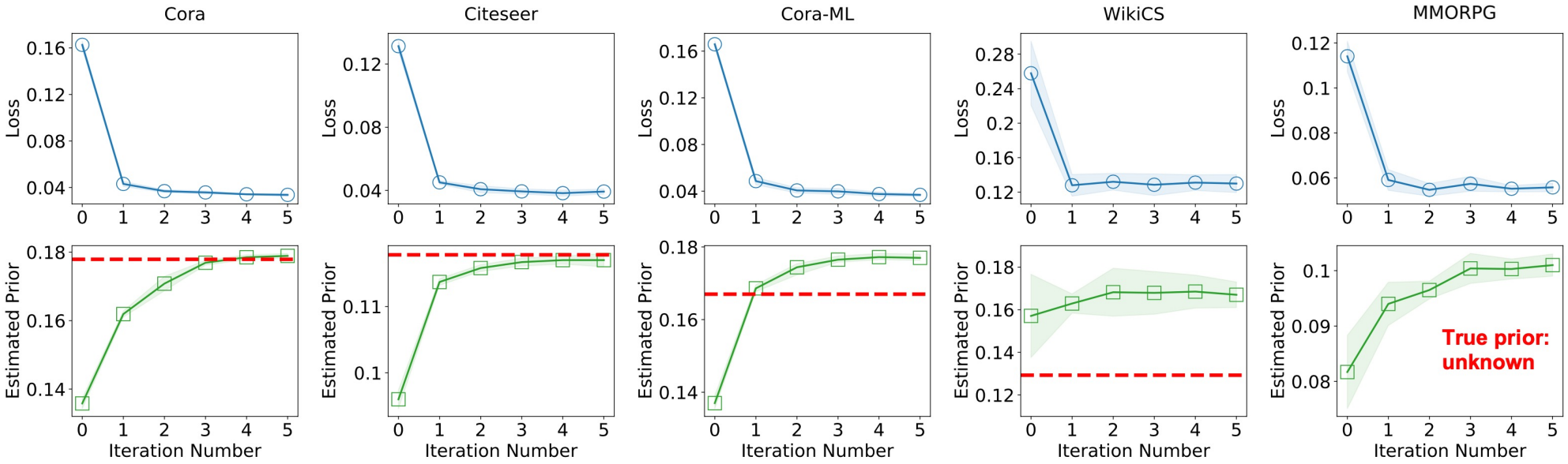
Accuracy with Class Prior

- **Q2.** Does GRAB outperform baselines even with known π_{p}?
- **A.** GRAB still shows competitive performance in all cases
  - GRAB does not utilize the known value of π_{p} unlike competitors

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

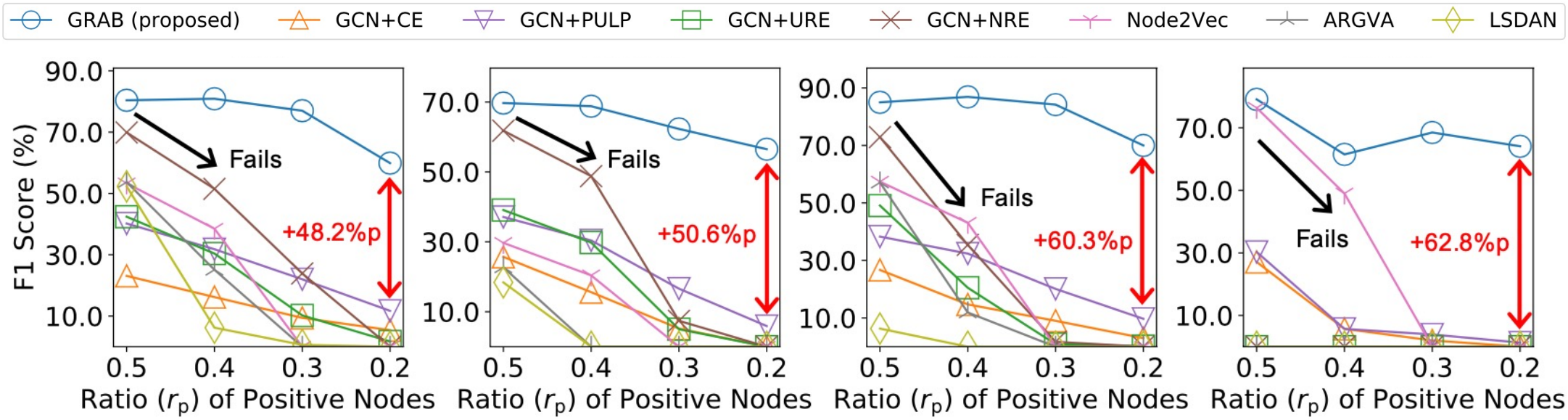
Estimation of Unknown Prior

- **Q3.** How well does GRAB estimate the unknown class prior $\hat{\pi}_{\mathrm{p}}$?
- **A.** GRAB finds unknown $\hat{\pi}_{\mathrm{p}}$ while minimizing the loss function
  - The exact value of $\hat{\pi}_{\mathrm{p}}$ is unknown in MMORPG, a true PU dataset



Accuracy with Fewer Observations

- **Q4.** How well does GRAB work with fewer observations?
- **A.** The accuracy gain greatly increases with fewer observed nodes
  - Most baselines fail to be trained due to limited observations

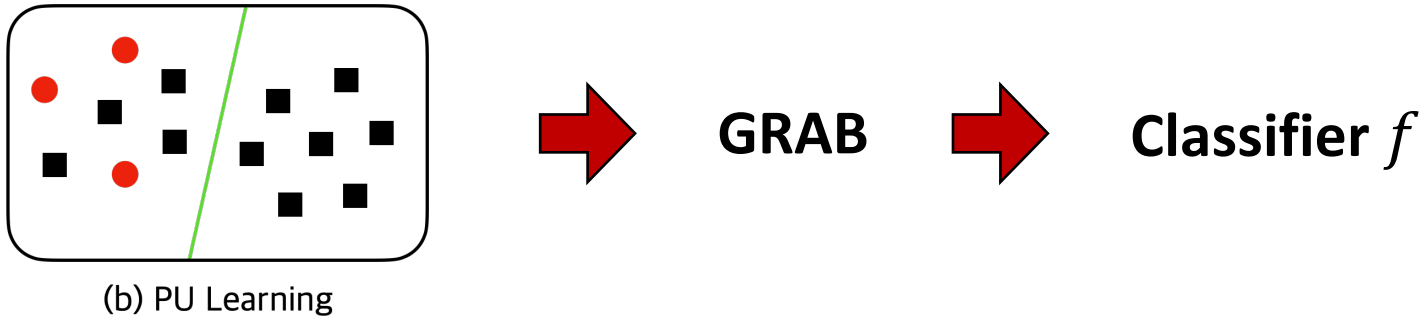


Outline

- Introduction
- **Graph-based PU learning**
  - Research Motivation
  - Proposed Method: GRAB
  - Experiments
  - **Summary**
- Graph augmentation
- Conclusion

Summary of GRAB

- We propose **GRAB** for training a GNN classifier for PU learning
  - GRAB runs **probabilistic inference** to get the distribution of $\mathcal{U}$
  - GRAB successfully estimates the **unknown class prior** π_p from the graph
- GRAB is a general framework compatible with any GNN models
- GRAB achieves SOTA performance in graph-based PU learning



Outline

- Introduction
- Graph-based PU learning
- **Graph augmentation**
  - **Research Motivation**
  - Proposed Methods: NodeSam & SubMix
  - Experiments
  - Summary
- Conclusion

Data Augmentation

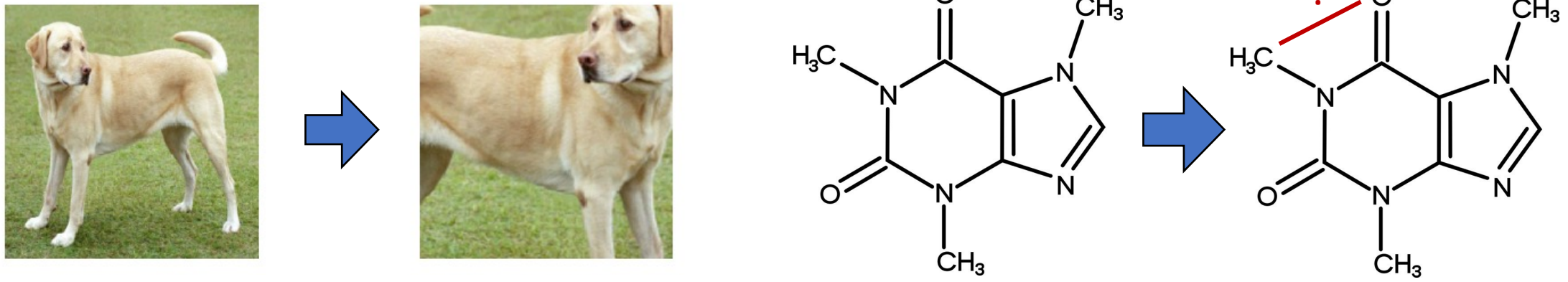
- **Data augmentation** is an essential strategy for machine learning
  - Increases the coverage of training samples in the data space
  - Improves the generalizability of models to unseen test data
- An example of data augmentation in the image domain is

Graph Augmentation

- Data augmentation can also be done in graphs
  - Recall that an image can be considered as a grid-structured graph
- Consider an undirected graph $G = (\mathcal{V}, \mathcal{E}, \mathbf{X})$
  - $\mathcal{V}$ is the set of nodes
  - $\mathcal{E}$ is the set of edges
  - $\mathbf{X}$ is the node feature matrix of size $|\mathcal{V}| \times d$
- We can change any of $\mathcal{V}$, $\mathcal{E}$, and $\mathbf{X}$ for data augmentation

Difficulties of Graph Augmentation

- Augmentation is okay only if **semantic information** is preserved
  - In images, it is possible to assure the preservation of semantic info.
- In graphs, even a single edge can change the semantic of a graph
  - We know that caffeine is rarely toxic, but what if we augment it?



Desired Properties

- We propose **five desired properties** for graph augmentation
  - **Objective 1.** To preserve semantic information as much as possible
  - **Objective 2.** To make sufficient changes of the graph structure

Method	P1	P2	P3	P4	P5
DropEdge [28]				✓	✓
GraphCrop [35]			✓	✓	✓
NodeAug [38]			✓	✓	✓
MotifSwap [55]	✓	✓			
NodeSam (proposed)	✓	✓	✓	✓	✓
SubMix (proposed)	✓	✓	✓	✓	✓

P1. Graph size should be preserved

P2. Connectivity should be preserved

P3. Node features should change

P4. # of edges should change

P5. It should be done in linear time

Outline

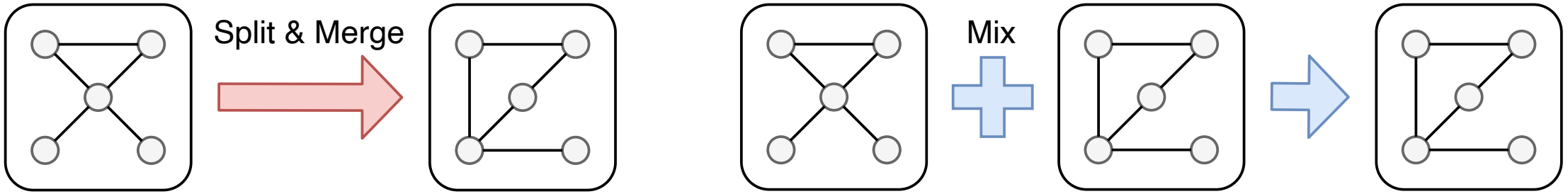
- Introduction
- Graph-based PU learning
- **Graph augmentation**
  - Research Motivation
  - **Proposed Methods: NodeSam & SubMix**
  - Experiments
  - Summary
- Conclusion

Proposed Methods

- This work proposes two novel algorithms for graph augmentation
  - Both are theoretically guaranteed to satisfy all desired properties

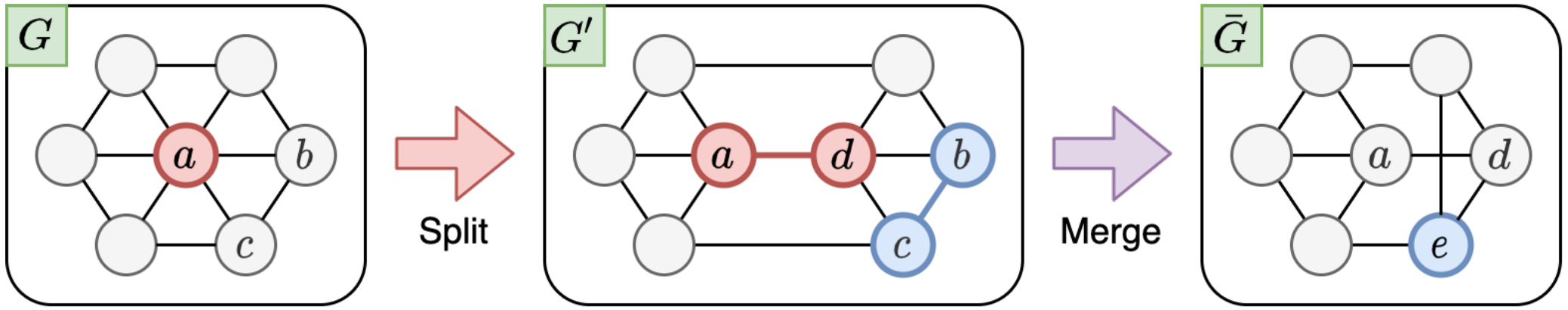
- **NodeSam** (Node Split & Merge)
  - Consists of Split and Merge
  - Makes balanced & stable changes

- **SubMix** (Subgraph Mix)
  - Combines multiple graphs
  - Makes a higher degree of changes



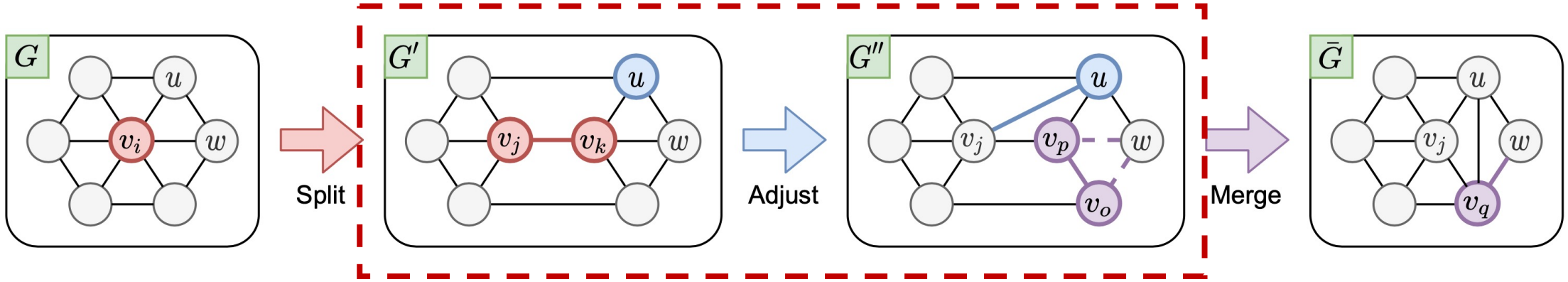
NodeSam (1)

- The goal of NodeSam is to make **balanced & stable changes**
- The main idea is to conduct two opposite operations at once
  - To **split** a random node and to **merge** a random pair of nodes
  - Each operation does not change the core structure of the graph



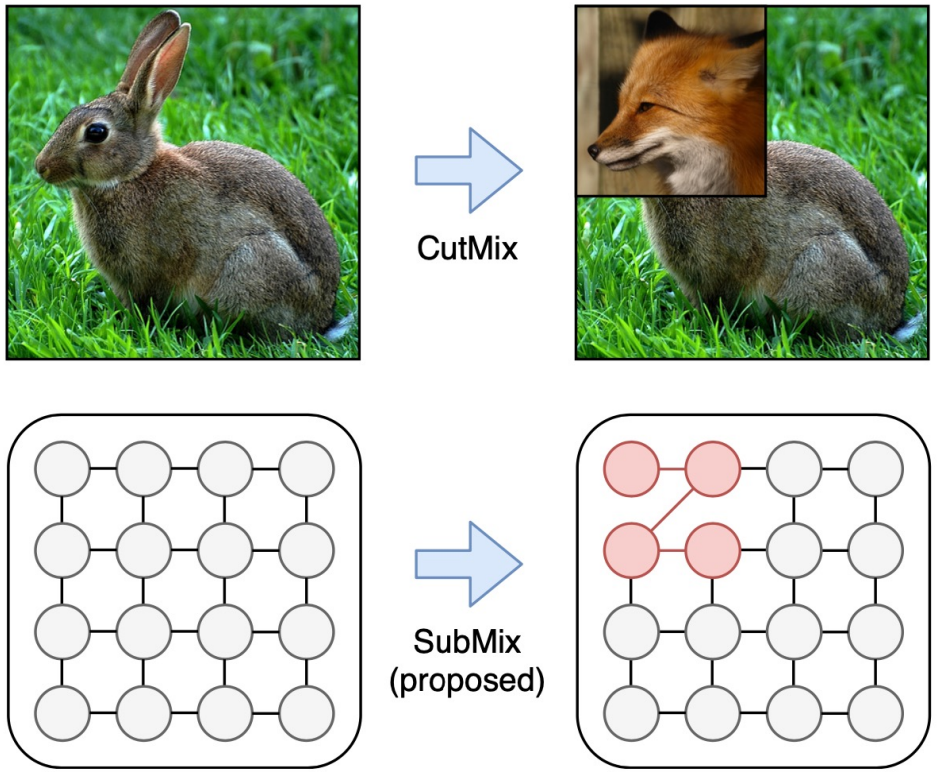
NodeSam (2)

- The basic version of NodeSam can **decrease** the number of edges
  - Because Merge can remove more than one edge in triangles
- We include an **adjustment** operation for unbiased changes
  - **Step 1.** Compute the expected number of edges removed by Merge
  - **Step 2.** Insert the same number of edges around the target node



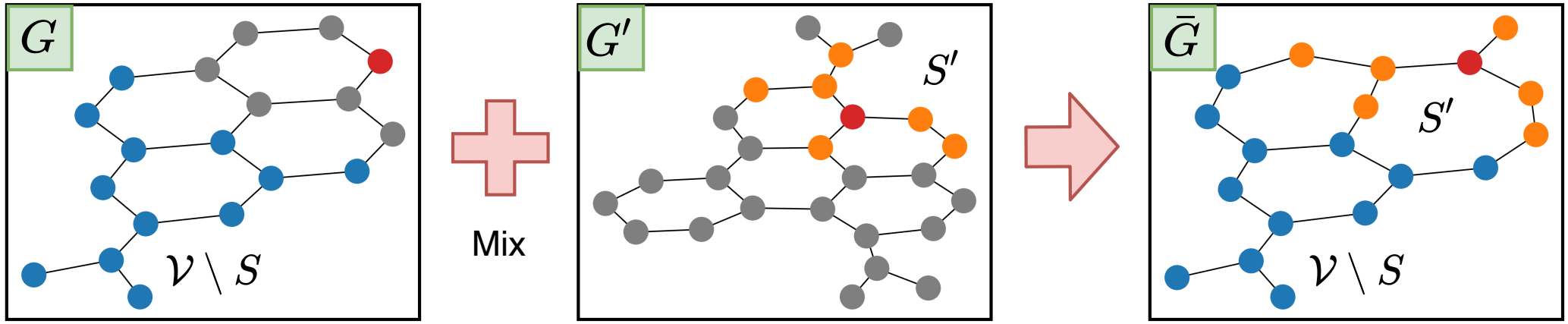
SubMix (1)

- The main idea of **SubMix** is to replace a random subgraph in a graph
  - For a higher degree of augmentation
- SubMix is a generalization of **CutMix** into the graph domain
  - CutMix is a popular augmentation algorithm in the image domain
  - The red subgraph is from another graph in the dataset
    - It corresponds to the head of the fox



SubMix (2)

- SubMix selects subgraphs S and S' from graphs G and G', resp.
  - This selection process is done by a **random walk** diffusion operator
- SubMix then inserts S' into G by replacing S with it
- This changes the structure of G a lot, but the change is **unbiased**



Outline

- Introduction
- Graph-based PU learning
- **Graph augmentation**
  - Research Motivation
  - Proposed Methods: NodeSam & SubMix
  - **Experiments**
  - Summary
- Conclusion

Datasets

- We use 9 benchmark datasets for graph classification
  - Each dataset consists of 344 to 144,033 graphs
  - The first 7 are molecular graphs, while the last 2 are large social networks

Dataset	Graphs	Nodes	Edges	Features	Labels
D&D[1]	1,178	334,925	843,046	89	2
ENZYMES[1]	600	19,580	37,282	3	6
MUTAG[1]	188	3,371	3,721	7	2
NCI1[1]	4,110	122,747	132,753	37	2
NCI109[1]	4,127	122,494	132,604	38	2
PROTEINS[1]	1,113	43,471	81,044	3	2
PTC-MR[1]	344	4,915	5,054	18	2
COLLAB[1]	5,000	372,474	12,286,079	369	3
Twitter[1]	144,033	580,768	717,558	1,323	2

Experimental Setup

- We use GIN [ICLR'19] as a graph classifier in all experiments
- We compare our two approaches with the following baselines:
  - **DropEdge** removes an edge uniformly at random
  - **DropNode** removes a node and its neighboring edges
  - **GraphCrop** selects and returns a random subgraph
  - **MotifSwap** changes the direction of an open triangle
  - …

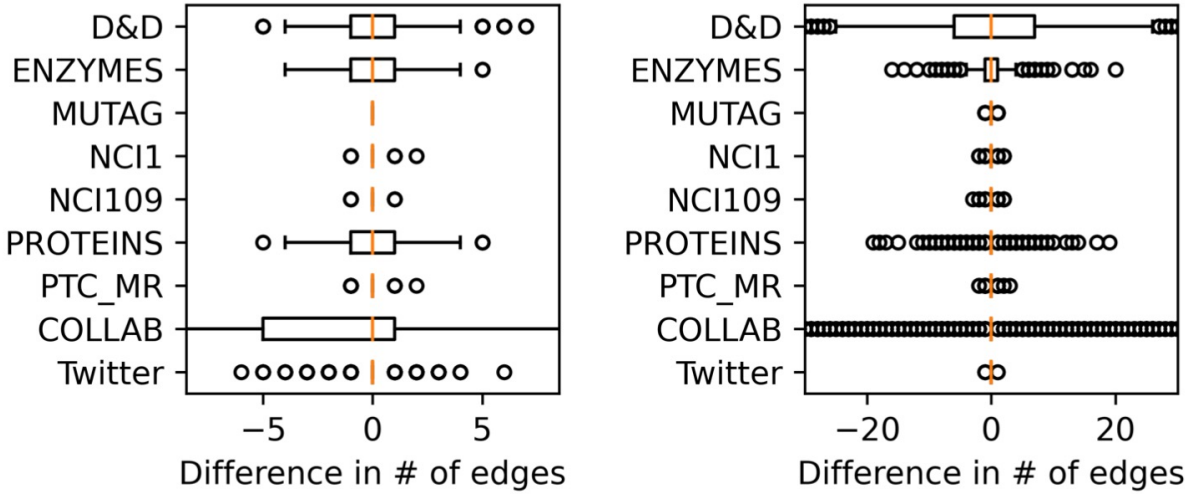
Classification Accuracy

- **Q1.** Do NodeSam and SubMix outperform the baselines?
- **A.** NodeSam and SubMix achieve the highest average accuracy
  - NodeSam makes the highest acc., while SubMix achieves the best rank

Method	D&D	ENZY.	MUTAG	NCI1	N109	PROT.	PTC-MR	COLLAB	Twitter	Average	Rank
Baseline	76.40 (4)	50.33 (10)	89.94 (4)	82.68 (9)	81.80 (9)	75.38 (9)	63.94 (7)	82.66 (7)	66.05 (7)	74.35 (8)	7.33 ± 2.18
GraphCrop	77.08 (2)	51.00 (9)	77.11 (10)	80.46 (10)	79.77 (10)	75.20 (10)	61.87 (10)	83.50 (2)	66.15 (3)	72.46 (10)	7.33 ± 3.77
DropEdge	76.14 (6)	53.67 (6)	81.93 (9)	82.82 (7)	82.60 (7)	75.74 (4)	63.68 (8)	82.50 (9)	66.05 (8)	73.90 (9)	7.11 ± 1.62
NodeAug	76.14 (8)	54.67 (5)	86.14 (7)	83.16 (4)	82.36 (8)	75.56 (6)	66.24 (2)	81.32 (10)	65.98 (9)	74.62 (7)	6.56 ± 2.55
AddEdge	76.14 (7)	55.17 (3)	85.67 (8)	83.99 (2)	83.06 (5)	75.38 (8)	64.27 (5)	82.80 (5)	66.10 (6)	74.73 (6)	5.44 ± 2.07
ChangeAttr	75.72 (9)	53.33 (8)	90.44 (3)	83.02 (5)	83.57 (2)	75.47 (7)	62.47 (9)	82.76 (6)	66.36 (2)	74.79 (5)	5.67 ± 2.83
DropNode	75.55 (10)	55.17 (3)	87.28 (6)	82.85 (6)	83.04 (6)	75.65 (5)	**66.59 (1)**	82.54 (8)	66.11 (5)	74.97 (4)	5.56 ± 2.60
MotifSwap	76.23 (5)	53.50 (7)	90.47 (2)	82.82 (7)	83.28 (4)	75.92 (3)	65.79 (3)	82.84 (3)	65.93 (10)	75.20 (3)	4.89 ± 2.62
SubMix	**78.10 (1)**	57.50 (2)	89.94 (4)	**84.33 (1)**	**84.37 (1)**	**76.19 (1)**	63.97 (6)	**83.74 (1)**	**66.44 (1)**	76.06 (2)	**2.00 ± 1.80**
NodeSam	76.57 (3)	**60.00 (1)**	**90.96 (1)**	83.33 (3)	83.52 (3)	76.10 (2)	65.48 (4)	82.82 (4)	66.13 (4)	**76.10 (1)**	2.78 ± 1.20

Unbiased Change of Edges

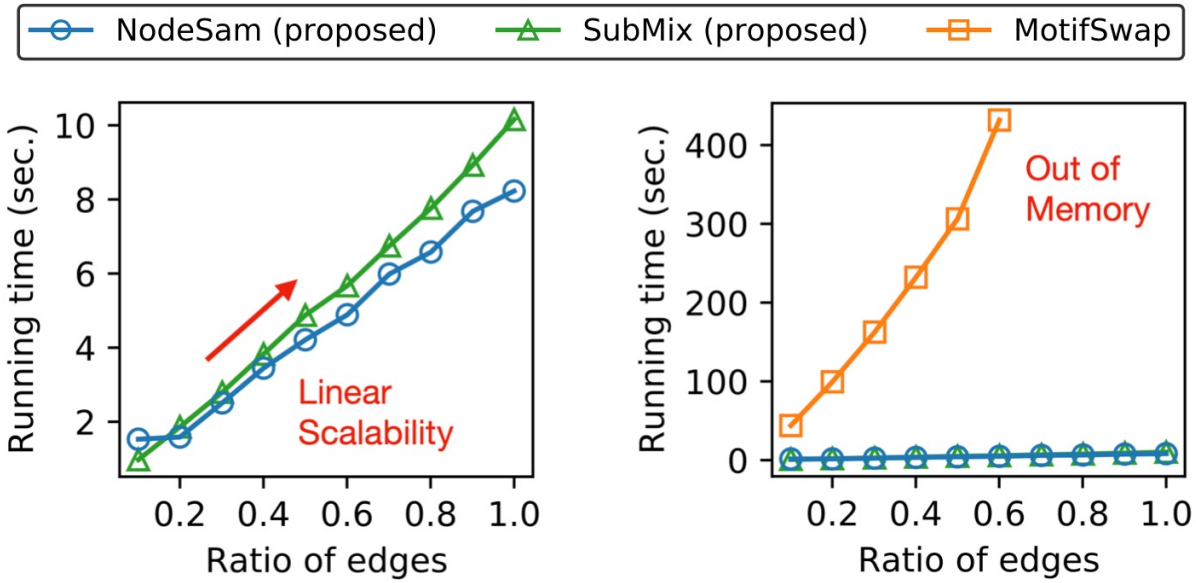
- **Q2.** How much do NodeSam and SubMix change the graph size?
- **A.** They make sufficient, unbiased changes of the number of edges
  - The variance of the change in # of edges is much larger with SubMix



(a) NodeSam (b) SubMix

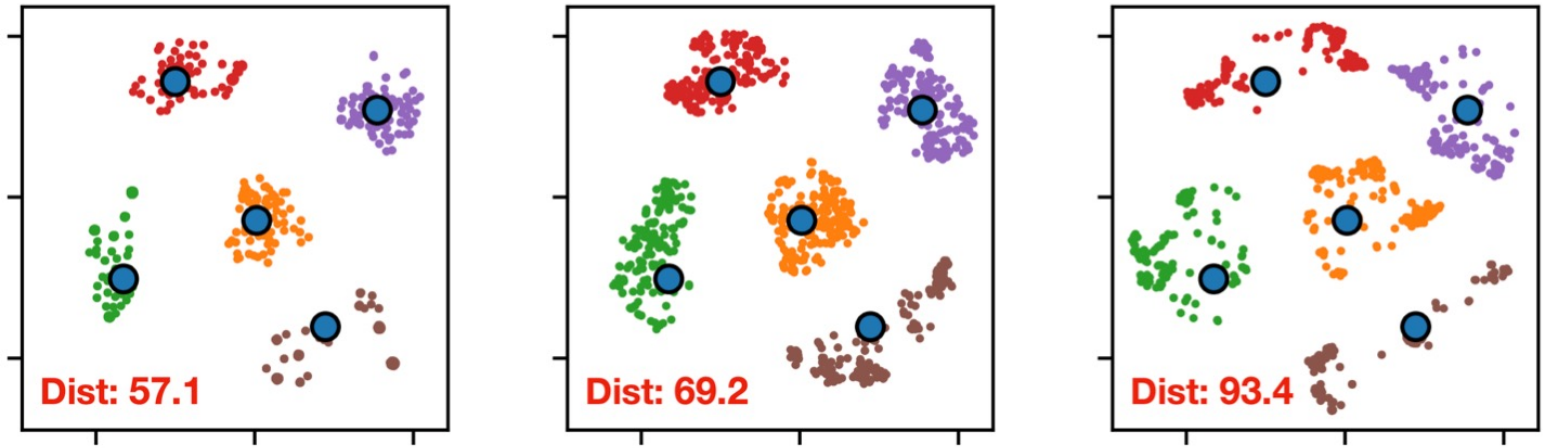
Scalability to Large Graphs

- **Q3.** How scalable are SubMix and NodeSam to large graphs?
- **A.** They show linear scalability with the number of edges
  - MotifSwap, the best competitor, takes exponential time



Visualization in 2D Space

- **Q4.** Do NodeSam and SubMix conduct sufficient augmentation?
- **A.** We visualize the space of augmentation of three algorithms
  - MotifSwap makes the smallest changes, while SubMix makes the largest



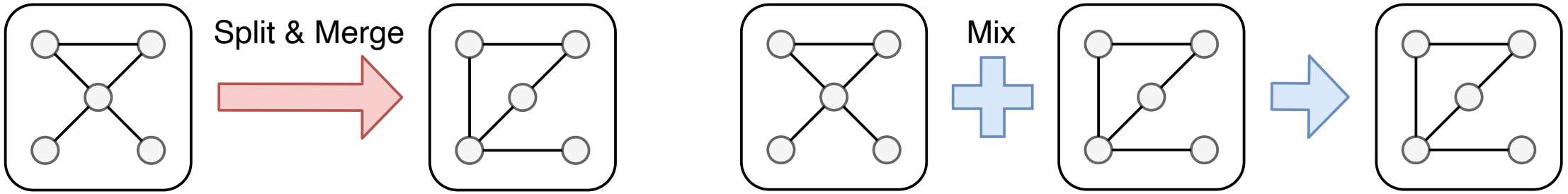
(a) MotifSwap (b) NodeSam (c) SubMix

Outline

- Introduction
- Graph-based PU learning
- **Graph augmentation**
  - Research Motivation
  - Proposed Methods: NodeSam & SubMix
  - Experiments
  - **Summary**
- Conclusion

Summary of NodeSam & SubMix

- First comprehensive study of **model-agnostic** graph augmentation
  - There are **model-specific** studies that assume specific classifiers
  - Model-agnostic augmentation is generalizable to various classifiers
- We propose **NodeSam & SubMix** satisfying the desired properties
  - To make sufficient changes while minimizing the risk of semantic change



Outline

- Introduction
- Graph-based PU learning
- Graph augmentation
- **Conclusion**

Conclusion

- This talk introduces our recent works for graph-related problems
  - **GRAB [ICDM 2021]**
    - Study the problem of graph-based PU learning
    - Model a graph as a probabilistic model and run inference for unlabeled nodes
  - **NodeSam & SubMix [WWW 2022]**
    - Study the problem of graph augmentation
    - Propose five desired properties and satisfy them with two novel algorithms
- Both works study fundamental but underexplored problems
  - They improve the generalizability of graph neural networks (GNN)

Thank you!

Jaemin Yoo

Email: jaeminyoo@snu.ac.kr

Homepage: https://jaeminyoo.github.io