정점 및 그래프 분류를 위한 확률 기반 접근법

Probabilistic Approaches for Node and Graph Classification

Jaemin Yoo

Ph.D. Candidate

Computer Science & Engineering

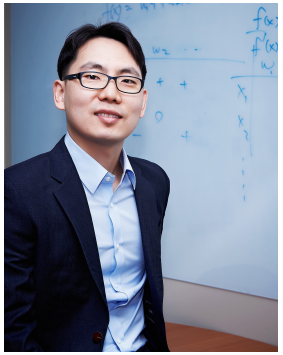
Seoul National University

Thesis Committee

- Prof. Hyoung-Joo Kim (김형주 교수님)
- Prof. U Kang (강유 교수님)
- Prof. Sun Kim (김선 교수님)
- Prof. Kunsoo Park (박근수 교수님)
- Prof. Hwanjo Yu (유환조 교수님)





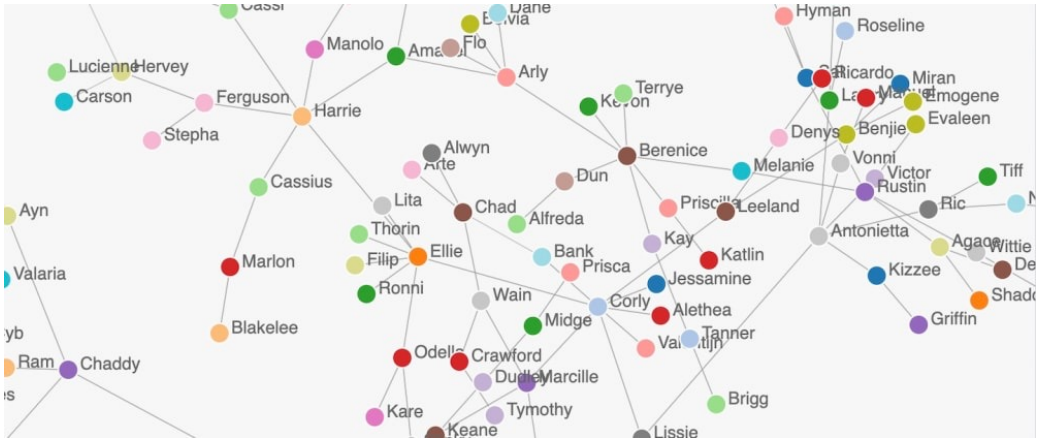


Outline

- **Introduction**
- Background
- Structural exploitation
- Structural modification
- Conclusion

Graphs

- Many real-world data are modeled as **graphs**
  - Social networks (users ↔ users)
  - Streaming services (movies ↔ users)
  - Chemical compounds (elements ↔ elements)



Graph Attributes

- Graphs often include **attributes**
  - Provide rich information about graphs
- Attributes in a streaming service
  - **Node features**: movie descriptions
  - **Node labels**: movie genre
  - **Edge features**: ratings of reviews
  - **Graph labels**: country of the service

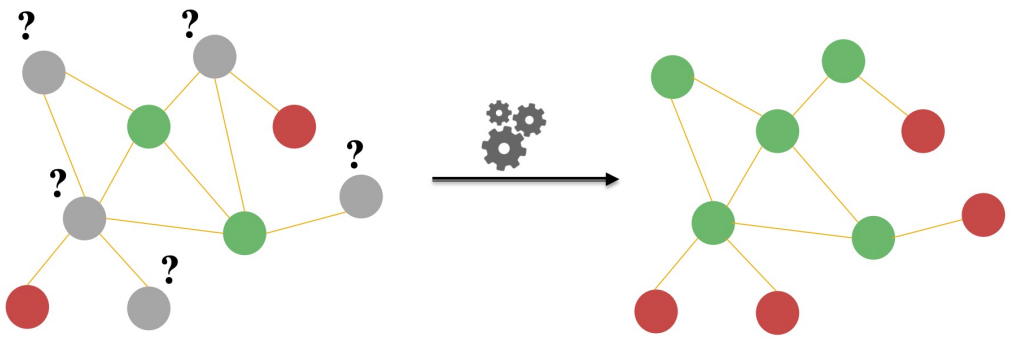
Research Goal

To understand and analyze the relationships between various graph attributes

- Done by solving two fundamental problems
  - **Node classification** and **graph classification**
- In various settings with different constraints
  - Inductive learning, feature estimation, subgraph sampling, graph augmentation, …

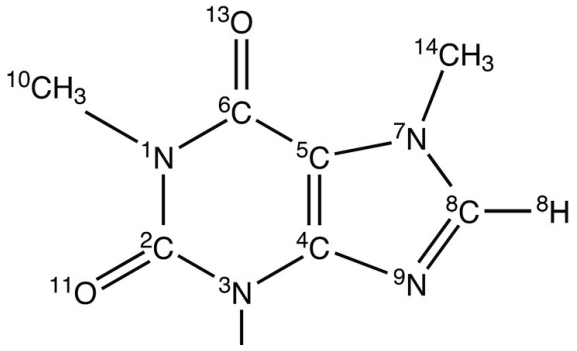
Node Classification

- To predict the labels of nodes in a graph
- Real-world applications
  - Finding malicious users in a social network
  - Classifying new items in an e-commerce graph

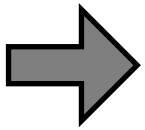


Graph Classification

- To predict the label of a given graph
- Real-world applications
  - Predicting the toxicity of a chemical compound
  - Classifying of a social group of people



Chemical compound graph



Toxic

Not Toxic

Research Overview

- We present an overview of my research
  - X, Y, and G: features, labels, and structures, resp.

	Structural Exploitation		Structural Modification
	$P(Y\|X,G)$	$P(X\|G)$	$P(G'\|G)$
Purely Probabilistic	Transductive Learning **[ICDM'17]**	-	Subgraph Sampling **[WSDM'20]**
Fusion with Deep Learning	Inductive Learning **[IJCAI'19]**	Feature Estimation **[submitted]**	Graph Augmentation **[submitted]**

Outline

- Introduction
- **Background**
- Structural exploitation
- Structural modification
- Conclusion

Markov Networks (1)

- **Pairwise Markov networks**
  - Probabilistic model for undirected graphs
  - Each node represents a random variable
  - Models their relations by **potential functions**
    - **Node potentials** and **edge potentials**

Markov Networks (2)

- **Node & edge potential functions**
  - **Node potential** ϕ_i for each node i
    - Represents the prior information for node i
    - E.g., how much a user is likely to be malicious?
  - **Edge potential** ψ_{ij} for each edge (i, j)
    - Represents the relation between adjacent nodes
    - E.g., how much information can we get for node i by knowing that its friend j is malicious?

LBP (1)

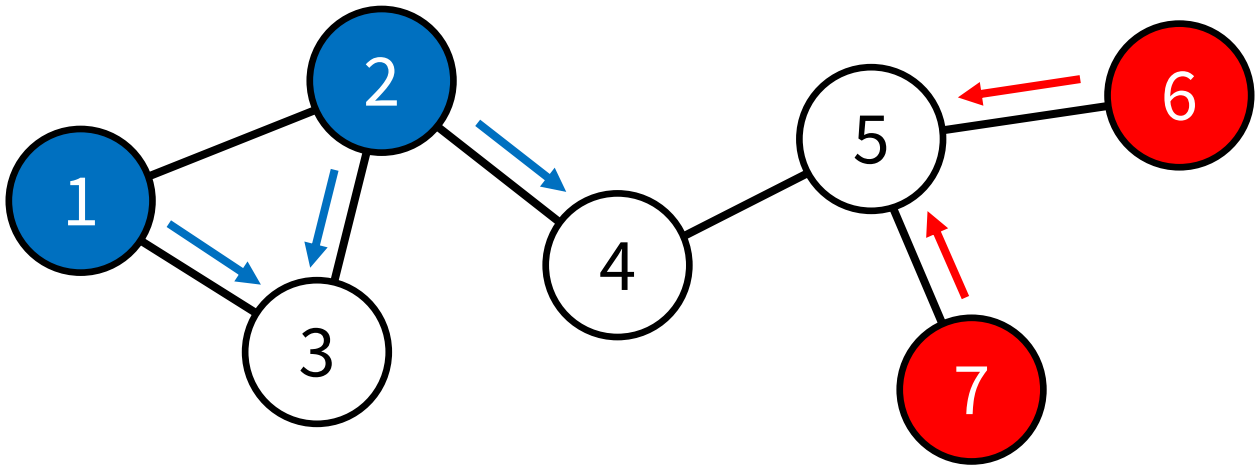
- **Loopy belief propagation (LBP)**
  - Approximate algorithm for marginalization
    - Estimates the posterior probabilities of all nodes
  - **Given** a pairwise Markov network G
  - **Computes** the belief $\mathbf{b}_i$ of node i such that

$$b_i(z_i) \approx P_i(Z_i = z_i)$$

    - P_i is the marginal distribution of node i

LBP (2)

- LBP as a **message passing algorithm**
  - It propagates **messages** through the edges
  - In other words, it exchanges the priors of nodes until the beliefs of all nodes converge

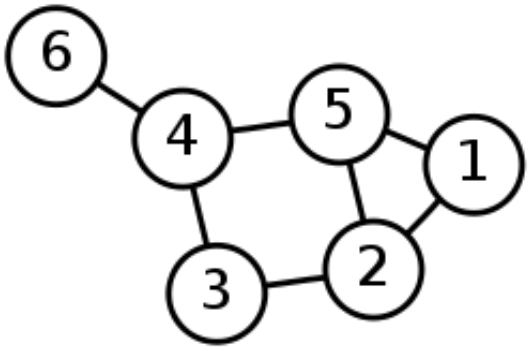


Outline

- Introduction
- Background
- **Structural exploitation**
- Structural modification
- Conclusion

Structural Exploitation

- Basic approach of many graph algorithms
  - Consider a graph as fixed constant evidence
  - Use the structure to estimate missing attributes
- We assumed many challenging scenarios



Graph structure

Estimate attributes

- Transductive learning
- Inductive learning
- Feature estimation

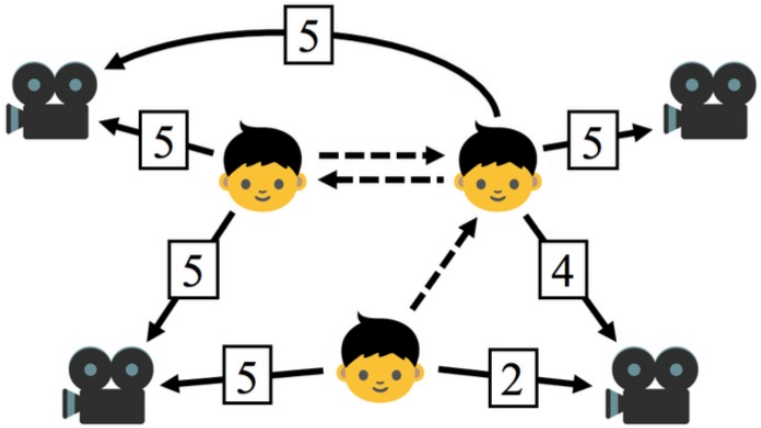
Research Overview

- **Q.** How can we accurately classify nodes in a graph having edge attributes?

	Structural Exploitation		Structural Modification
	$P(Y\|X,G)$	$P(X\|G)$	$P(G'\|G)$
Purely Probabilistic	**Transductive Learning [ICDM'17]**	-	Subgraph Sampling [WSDM'20]
Fusion with Deep Learning	Inductive Learning [IJCAI'19]	Feature Estimation [submitted]	Graph Augmentation [submitted]

Edge-Attributed Graphs

- **Edge-attributed graphs**
  - Each edge contains a sign, a rating, or a general feature vector describing its characteristic
- Examples of edge-attributed graphs
  - User-item graph with rating edges: $[1, 5]$
  - Social network with signed interactions: ± 1



Research Motivation (1)

- Modeling of LBP in previous works
  - The edge potential ψ_{ij} is defined heuristically as

$$\psi_{ij}(u_i, u_j) = \begin{cases} \epsilon & \text{if } u_i = u_j \\ 1 - \epsilon & \text{otherwise} \end{cases}$$

  - This applies to every edge (i, j) in the graph
  - Thus, the value of ϵ becomes very important
    - Large ϵ makes nodes more correlated with each other

Research Motivation (2)

- Inappropriate in edge-attributed graphs
  - Each edge has a different degree of propagation
  - Difficult to determine ψ_{ij} for all pairs of types
    - More difficult if edges have numerical feature vectors

Research Question

How can we assign a suitable edge potential ψ in edge-attributed graphs with no prior knowledge?

Proposed Method (1)

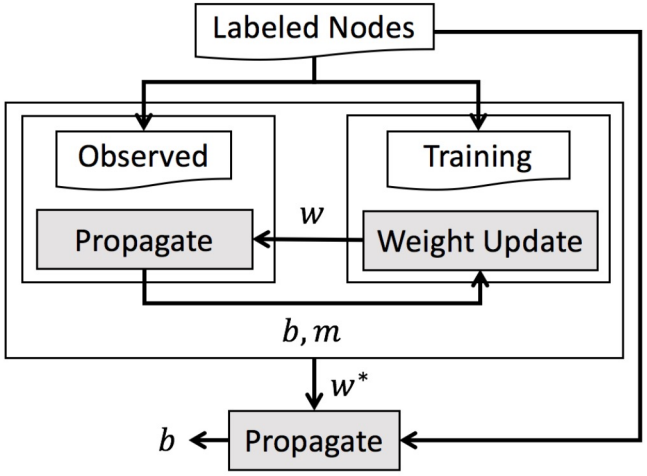
- **SBP (Supervised Belief Propagation)**
  - Assume a feature $\mathbf{x}_{ij}$ for every edge (i, j)
  - Model $\psi_{ij}(\cdot)$ as a function of $\mathbf{x}_{ij}$ such that

$$\psi_{ij}(\mathbf{x}_{ij}; \theta) = \frac{1}{1 + \exp(-\mathbf{x}_{ij}^{\top}\theta)}$$

  - θ is a weight vector learned globally in a graph

Proposed Method (2)

- SBP learns optimal values of θ as follows:
  1. Model LBP as a differentiable operation $f(\cdot)$
  2. Introduce an objective function $\mathcal{L}$ to minimize
  3. Update θ minimizing $\mathcal{L}$ by backpropagation



Algorithm 1: Supervised Belief Propagation

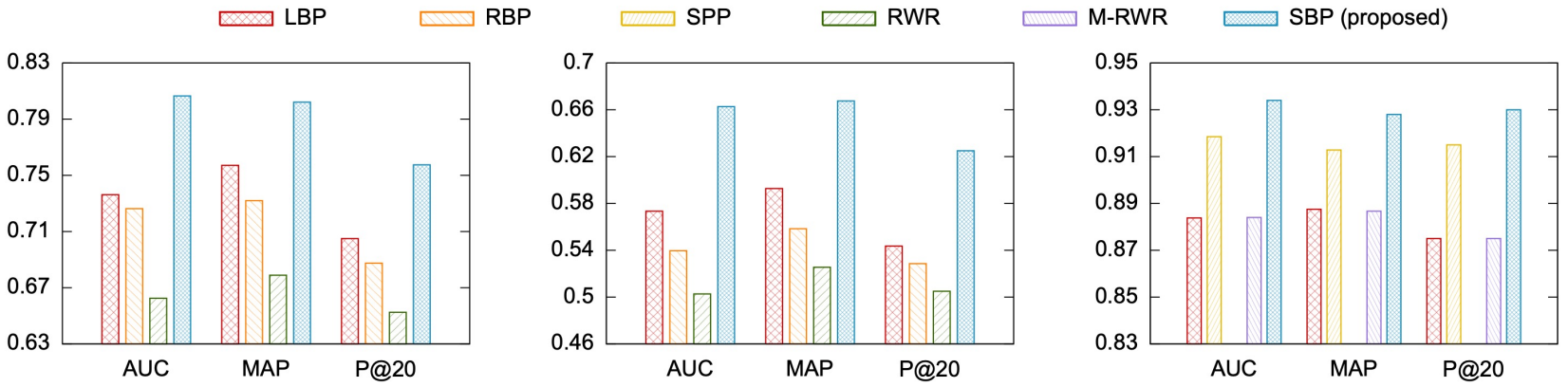
Input: attributed network G, sets P and N of positive and negative nodes, and node potential ϕ

Output: beliefs b for all nodes

$P_{\text{obs}}, P_{\text{trn}} \leftarrow$ randomly split P into two sets
$N_{\text{obs}}, N_{\text{trn}} \leftarrow$ randomly split N into two sets
$w \leftarrow$ an initial weight vector
while convergence criterion of w is not met **do**
  $b, m \leftarrow propagate(w, P_{\text{obs}}, N_{\text{obs}}, \phi)$
  $w \leftarrow weight\_update(w, b, m, P_{\text{trn}}, N_{\text{trn}})$
end while
$b, m \leftarrow propagate(w, P, N, \phi)$
return b

Experiments (1)

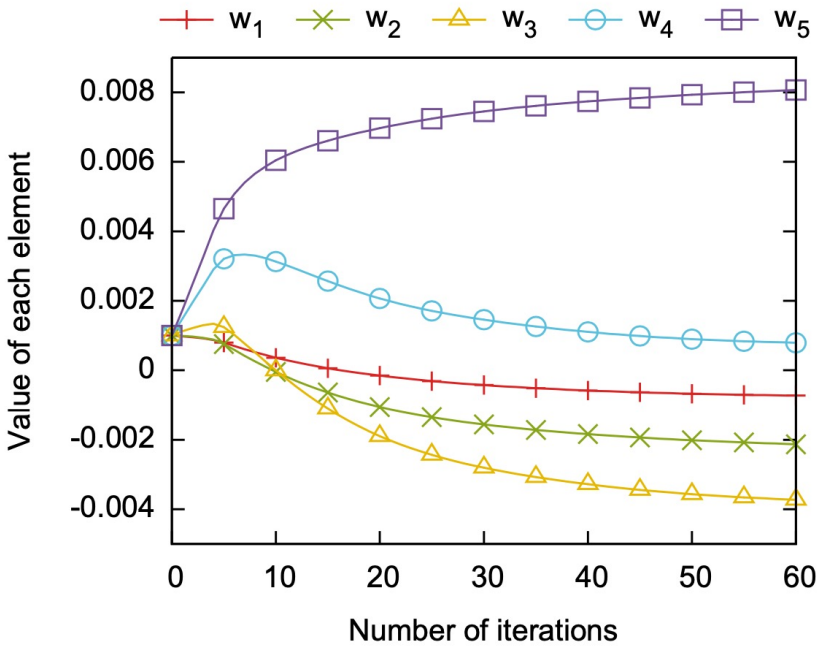
- SBP works the best in three graph datasets
  - MovieLens is a graph with ratings
  - Epinions-R is a social network with ratings
  - Epinions-S is a social network with signs



(a) MovieLens (b) Epinions-R (c) Epinions-S

Experiments (2)

- The weights are updated through iterations
  - $w_1, \ldots, w_5$ are weights for different ratings



The learned weights are ordered naturally:

$w_1, w_2, w_3 < w_4 \ll w_5$

Research Overview

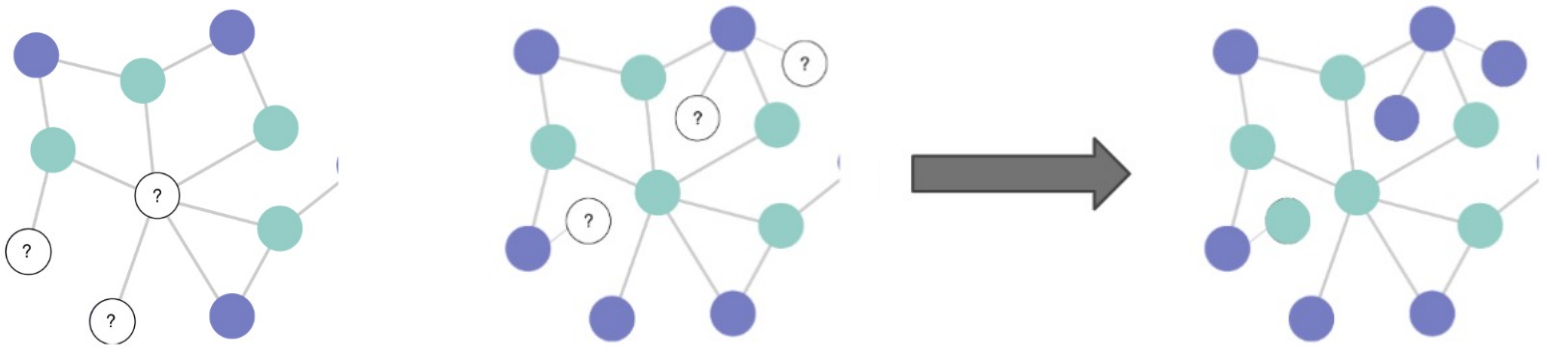
- **Q.** How can we classify nodes if unseen test nodes are given in isolation?

	Structural Exploitation		Structural Modification
	$P(Y\|X,G)$	$P(X\|G)$	$P(G'\|G)$
Purely Probabilistic	Transductive Learning [ICDM'17]	-	Subgraph Sampling [WSDM'20]
Fusion with Deep Learning	**Inductive Learning [IJCAI'19]**	Feature Estimation [submitted]	Graph Augmentation [submitted]

Inductive Learning (1)

- Test nodes are **not** given during training
  - We train a model f for a training graph G_{trn}
  - We then apply f to unseen nodes in G_{test}

Training **Test**



Node Masking: Making Graph Neural Networks Generalize and Scale Better (arXiv 2020)

Inductive Learning (2)

- **Hard inductive learning**
  - We consider a more challenging scenario
  - Test nodes are given **without neighborhood**
- Real-world scenarios
  - New users in an online social network
  - New movies uploaded to a streaming service

Research Motivation

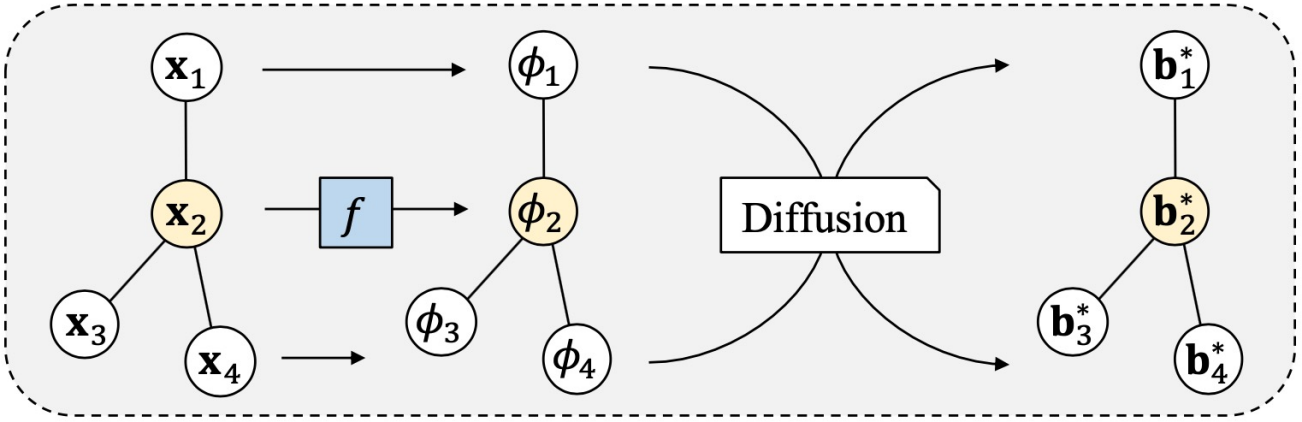
- Limitations of previous works
  - Features and a structure are used together
  - Accuracy drops severely in isolated test nodes
    - Since neighborhood information is not given

Research Question

How can we avoid the accuracy drop while using both node features and a graph structure?

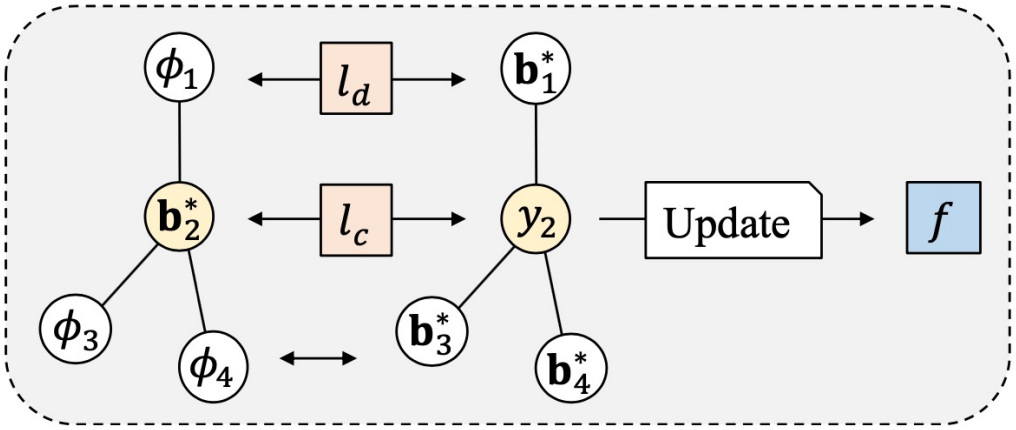
Proposed Method (1)

- **BPN (Belief Propagation Network)**
  - Separates classification and diffusion steps
    - **Classifier** f first predicts the prior of each node
      - Uses only node features ignoring the graph
    - The **diffuser** runs LBP to compute the beliefs



Proposed Method (2)

- Two types of **objective functions** are used
  - l_c uses the observed labels for the training
  - l_d makes f mimic the result of diffusion
    - This gives f pseudo answers based on the graph



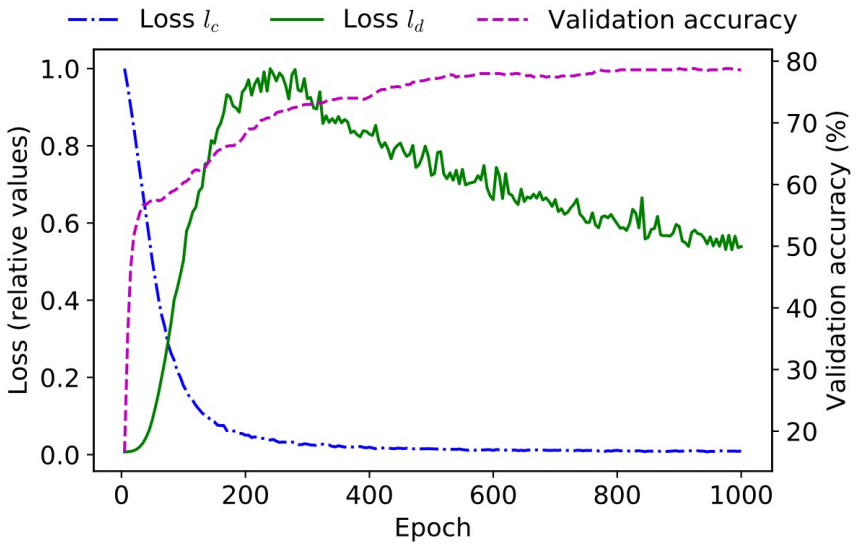
Experiments (1)

- BPN performs the best in four datasets
  - All baselines are designed for inductive learning
  - They fail when test nodes have no neighborhood

Method	Pubmed	Cora	Citeseer	Amazon
Planetoid	74.6 ± 0.5	66.2 ± 0.9	66.8 ± 1.0	70.1 ± 1.9
GCN-I	74.1 ± 0.2	67.8 ± 0.6	63.6 ± 0.5	76.5 ± 0.3
SEANO	75.7 ± 0.4	64.5 ± 1.2	66.3 ± 0.8	78.6 ± 0.6
GAT	76.5 ± 0.4	70.1 ± 1.0	66.7 ± 1.0	77.5 ± 0.4
BPN (ours)	$\mathbf{78.3 \pm 0.3}$	$\mathbf{72.2 \pm 0.5}$	$\mathbf{70.1 \pm 0.9}$	$\mathbf{81.5 \pm 1.3}$

Experiments (2)

- BPN effectively minimizes the two losses
  - l_c is minimized through iterations stably
  - l_d **increases at first** and then is minimized
    - Because f changes the result of diffusion via training



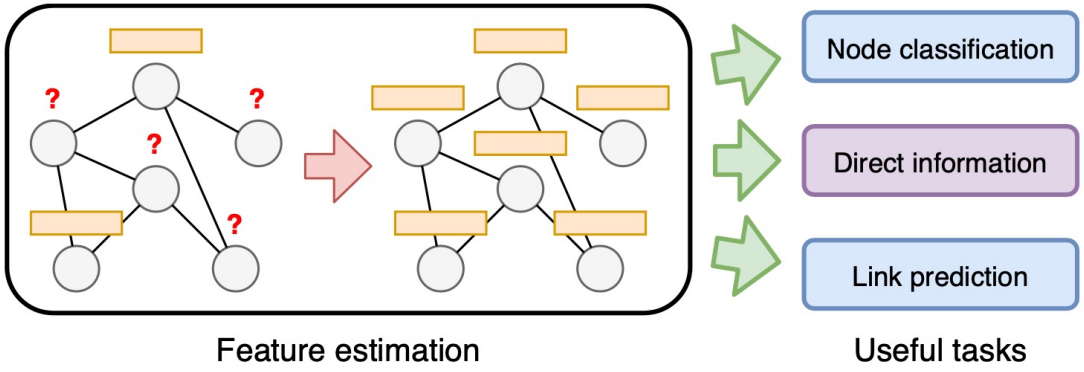
Research Overview

- **Q.** How can we accurately estimate missing node features in a graph?

	Structural Exploitation		Structural Modification
	$P(Y\|X,G)$	$P(X\|G)$	$P(G'\|G)$
Purely Probabilistic	Transductive Learning [ICDM'17]	-	Subgraph Sampling [WSDM'20]
Fusion with Deep Learning	Inductive Learning [IJCAI'19]	**Feature Estimation [submitted]**	Graph Augmentation [submitted]

Feature Estimation

- Graphs commonly have missing features
  - E.g., items without descriptions in e-commerce
- **Node feature estimation**
  - Allow us to get missing information of nodes
  - Improve the performance of graph-related tasks



Research Motivation

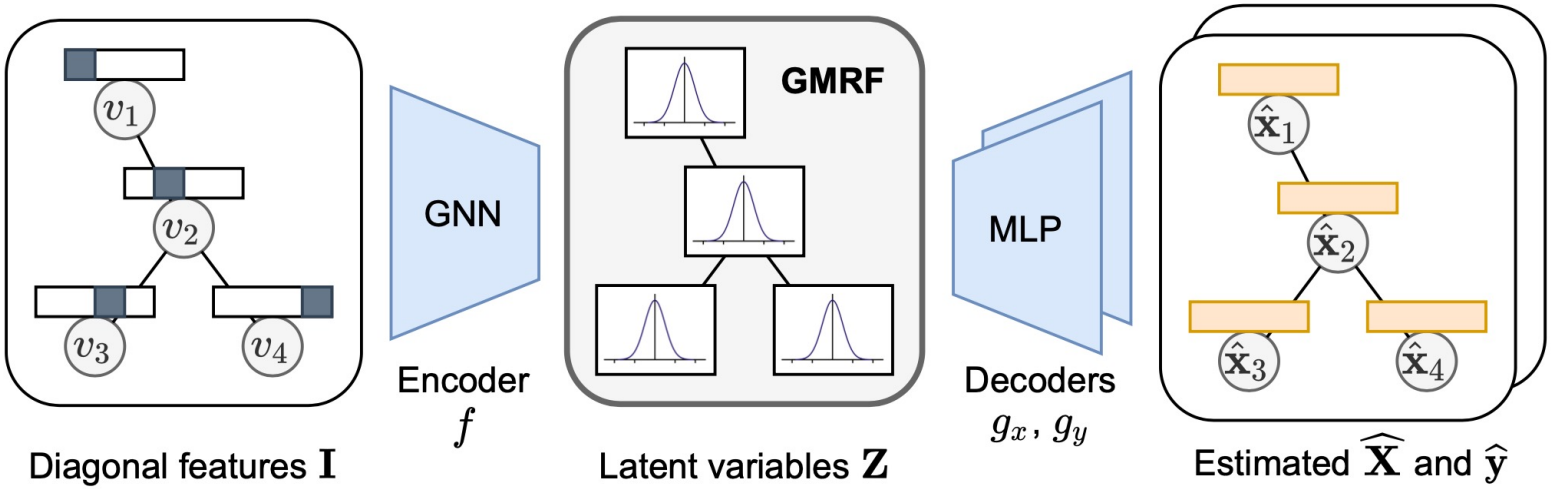
- Feature estimation is a **challenging** problem
  - Target variables are high-dimensional vectors
  - No information is given for the target nodes
    - In classification, every test node has a feature vector
    - In feature estimation, we have **only a graph structure**

Research Question

How can we accurately estimate missing features?

Proposed Method (1)

- **MGA (Markov Graph Autoencoder)**
  - Models all nodes in a graph as latent variables
  - Runs **variational inference** assuming their prior as a **Gaussian Markov random field (GMRF)**



Proposed Method (2)

- **Variational inference**
  - Technique to maximize the intractable likelihood
  - Popular for *variational autoencoders (VAE)*

Target likelihood Evidence lower bound (ELBO)

$$\log p_\Theta(\mathbf{X}, \mathbf{y} \mid \mathbf{A}) \geq \mathcal{L}(\Theta)$$

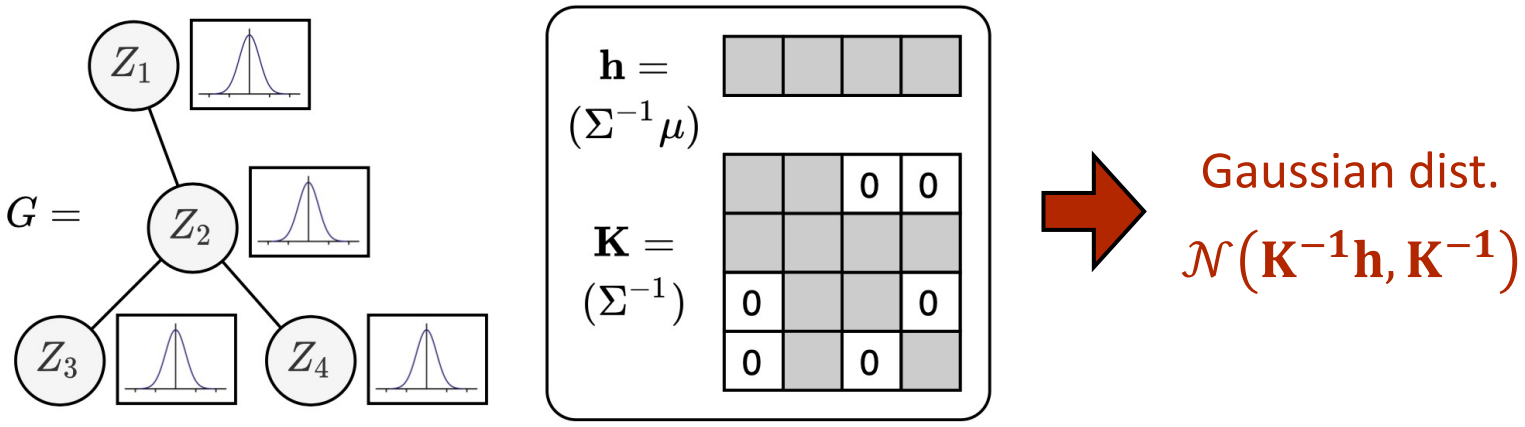
$$= \mathbb{E}_{\mathbf{Z} \sim q_\phi(\mathbf{Z} \mid \mathbf{X}, \mathbf{y}, \mathbf{A})} [\log p_{\theta,\rho}(\mathbf{X}, \mathbf{y} \mid \mathbf{Z}, \mathbf{A})]$$

$$- D_{\mathrm{KL}}(q_\phi(\mathbf{Z} \mid \mathbf{X}, \mathbf{y}, \mathbf{A}) \,||\, p(\mathbf{Z} \mid \mathbf{A})),$$

Reconstruction error Regularization by KL divergence

Proposed Method (3)

- **Regularization with the GMRF prior**
  - Previous works ignore the correlations in $\mathbf{Z}$
  - **GMRF** allows us to model the correlations
    - The graph structure provides a covariance matrix



Experiments (1)

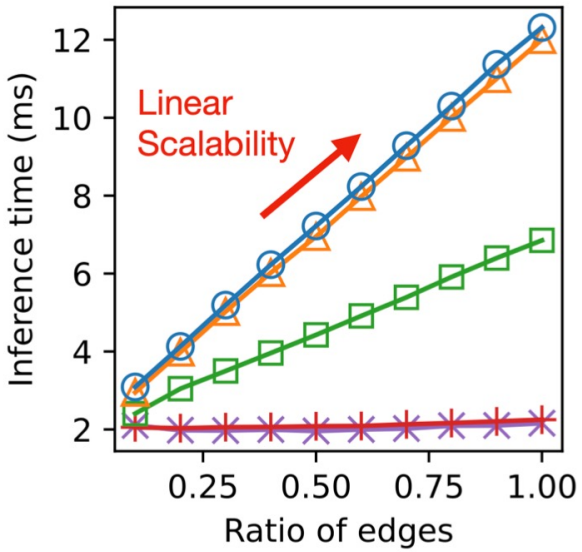
- MGA works the best in feature estimation
  - Each value is a recall score for nonzero features
  - SAT [TPAMI'20] is the state-of-the-art approach

Model	Cora			Citeseer			Computers		
	@10	@20	@50	@10	@20	@50	@10	@20	@50
NeighAggre	.0906	.1413	.1961	.0511	.0908	.1501	.0321	.0593	.1306
VAE	.0887	.1228	.2116	.0382	.0668	.1296	.0255	.0502	.1196
GNN*	.1350	.1812	.2972	.0620	.1097	.2058	.0273	.0533	.1278
GraphRNA	.1395	.2043	.3142	.0777	.1272	.2271	.0386	.0690	.1465
ARWMF	.1291	.1813	.2960	.0552	.1015	.1952	.0280	.0544	.1289
SAT-SAGE	.1356	.1981	.3165	.0704	.1163	.2174	.0419	.0738	.1562
SAT-GAT	.1653	.2345	.3612	.0811	.1349	.2431	.0421	.0746	.1577
MGA (proposed)	**.1718**	**.2486**	**.3814**	**.0943**	**.1539**	**.2782**	**.0437**	**.0769**	**.1602**

Experiments (2)

- MGA is effective for node classification
  - (Left) MGA improves accuracy of classification
  - (Right) MGA runs in linear time with # of edges

Model	Classifier	Cora	Cite.	Comp.
NeighAggre	MLP	.6248	.5549	.8365
VAE	MLP	.2826	.2551	.3747
GNN*	MLP	.4852	.3933	.3747
GraphRNA	MLP	.7581	.6320	.6968
ARWMF	MLP	.7769	.2267	.5608
SAT-SAGE	MLP	.7032	.5936	.8396
SAT-GAT	MLP	.7937	.6475	.8201
MGA (proposed)	MLP	**.8493**	**.6757**	**.8806**



Discussion

- We solved various problems for estimating node attributes
- **SBP** is our basic approach for classification
- **BPN** and **MGA** improve it in different ways
  - They utilize deep neural networks as modules
  - **BPN** focuses on cold-start inductive learning
  - **MGA** can estimate high-dimensional variables

Feature Estimation (1)

- MGA can generate various types of features
  - Let x_i be the i-th element of a feature $\mathbf{x}$
  - **Categorical** features: $\sum_i x_i = 1$
    - Node classes, categorical information, etc.
  - **Bernoulli** features: $x_i \in \{0,1\}\ \forall i$
    - Bag-of-words vectors, membership, etc.
  - **Gaussian** features: $x_i \in \mathbb{R}\ \forall i$
    - TF-IDF scores, RGB image pixels, etc.

Feature Estimation (2)

- Difference from generative learning
  - Generative learning: $p(X)$
    - The goal is to generate **plausible** data samples
    - There are no explicit answers for the training
  - Feature estimation: $p(X|G)$
    - The problem is **supervised** with exist **answers**
    - Each node i has a unique index in the graph G
      - Thus, we aim to predict the specific unseen feature $\mathbf{x}_i$

Feature Estimation (3)

- Two perspectives for feature estimation
  - **Intermediate problem** for node classification
    - The generated features help improve accuracy
    - That is, $G \rightarrow X$ and then $(G, X) \rightarrow Y$
  - **Generalization** of node classification
    - Node labels can also be predicted by MGA
    - That is, node features are generalization of labels

Either way, feature estimation is closely related to node classification

Graph Neural Networks (1)

- **Graph neural networks (GNN)**
  - Powerful tool to solve many graph-related tasks
- **Weaknesses** over probabilistic approaches
  - Require many trainable parameters
  - Much less powerful without node features
  - Not easily generalizable to test scenarios

Graph Neural Networks (2)

- SBP shows similar performance to GNNs
  - SBP effectively utilizes edge attributes with only five parameters (up to 127K times fewer)
  - Each parameter is for each rating score in [1,5]

Method	AUC	Parameters
SGC [38]	78.91 ± 6.11	19,848
GCN [70]	69.67 ± 9.86	317,634
GraphSAGE [39]	81.60 ± 5.53	635,234
GAT [38]	80.12 ± 6.76	635,398
SBP (proposed)	80.65	5

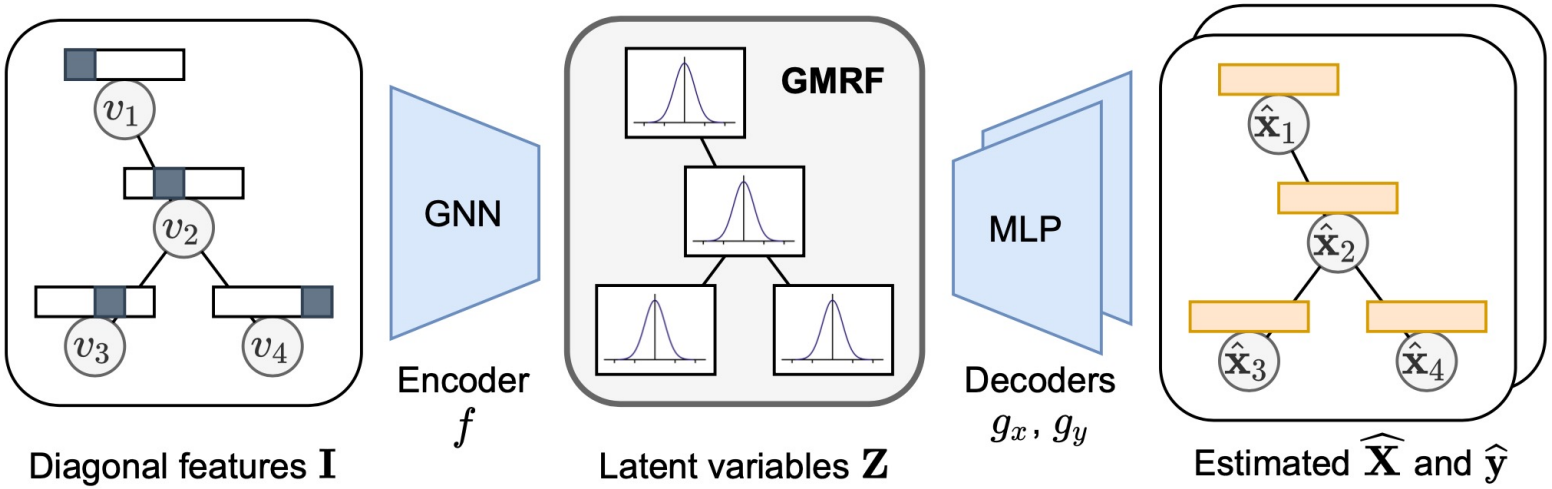
Graph Neural Networks (3)

- BPN outperforms GNNs in inductive learning
  - GNNs mix prediction and diffusion steps
    - It is generally powerful but makes accuracy drop
  - BPN takes advantage of probabilistic diffusion

Method	Pubmed	Cora	Citeseer	Amazon
Planetoid	74.6 ± 0.5	66.2 ± 0.9	66.8 ± 1.0	70.1 ± 1.9
GCN-I	74.1 ± 0.2	67.8 ± 0.6	63.6 ± 0.5	76.5 ± 0.3
SEANO	75.7 ± 0.4	64.5 ± 1.2	66.3 ± 0.8	78.6 ± 0.6
GAT	76.5 ± 0.4	70.1 ± 1.0	66.7 ± 1.0	77.5 ± 0.4
BPN (ours)	$\mathbf{78.3 \pm 0.3}$	$\mathbf{72.2 \pm 0.5}$	$\mathbf{70.1 \pm 0.9}$	$\mathbf{81.5 \pm 1.3}$

Graph Neural Networks (4)

- MGA uses a GNN as an encoder function
  - GNNs are suitable for making latent variables
    - Due to their large representation power
  - Our GMRF regularizer makes it avoid overfitting

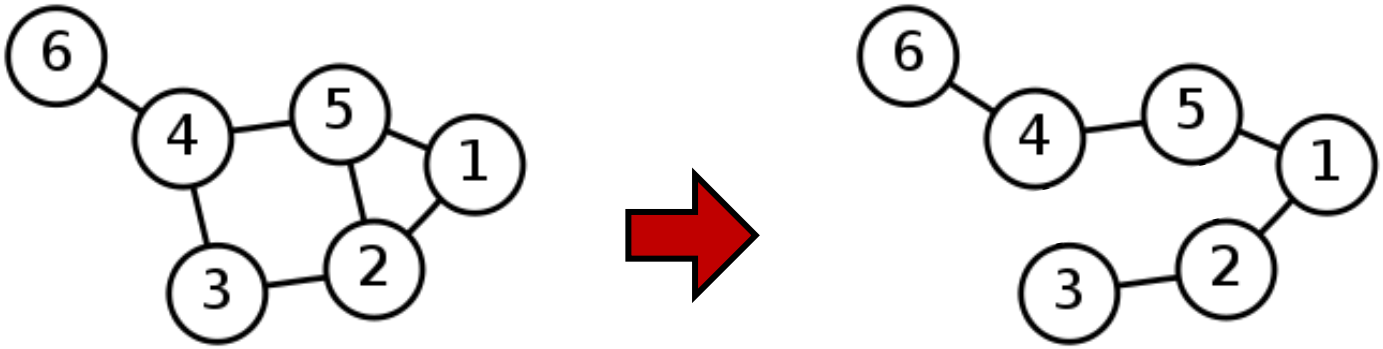


Outline

- Introduction
- Background
- Structural exploitation
- **Structural modification**
- Conclusion

Structural Modification

- Modify a graph to make better performance
- Have a larger potential with a higher risk
  - Can remove noises and unnecessary parts
  - Essential components can be modified or removed



Given structure

New structure

Research Overview

- **Q.** How can we improve accuracy and speed of inference for node classification?

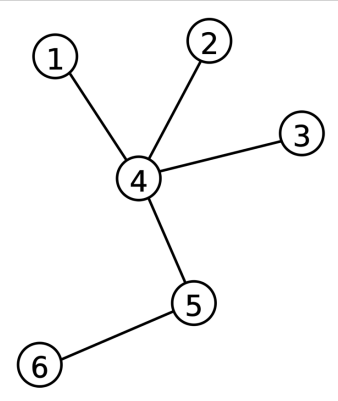
	Structural Exploitation		Structural Modification
	$P(Y\|X,G)$	$P(X\|G)$	$P(G'\|G)$
Purely Probabilistic	Transductive Learning [ICDM'17]	-	**Subgraph Sampling [WSDM'20]**
Fusion with Deep Learning	Inductive Learning [IJCAI'19]	Feature Estimation [submitted]	Graph Augmentation [submitted]

Graphical Inference

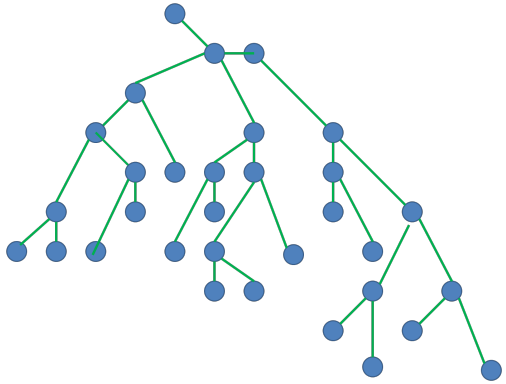
- To compute nodes' marginal distributions
  - To get $p_i(x_i) = \sum p(x_1, \dots, x_N)$ for each node i
- Popular algorithms
  - **Loopy belief propagation (LBP)**
    - **Pros:** Fast and scalable
    - **Cons:** Unguaranteed convergence
  - **Junction tree algorithm (JT)**
    - **Pros:** Exact and guaranteed termination
    - **Cons:** Takes exponential time with **treewidth** k

Treewidth

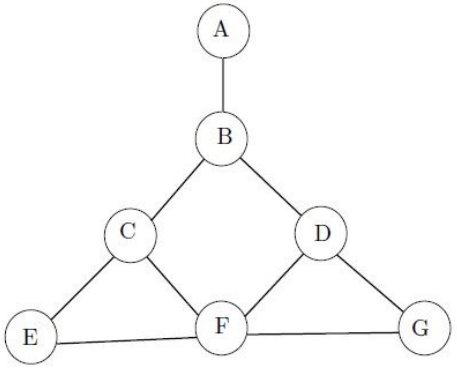
- How much a graph resembles a tree
  - **Low treewidth:** simple tree-like structures
  - **High treewidth:** complex loops and cliques



$k = 1$



$k = 1$



$k = 2$

Research Motivation

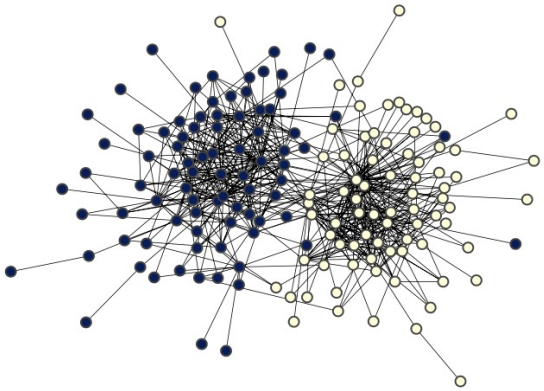
- Real-world graphs have large treewidth k
- **Research motivation**
  - Assume that not every edge is essential
  - We can decrease k by sampling a subgraph
  - Then, we can run JT on sampled subgraphs

Research Question

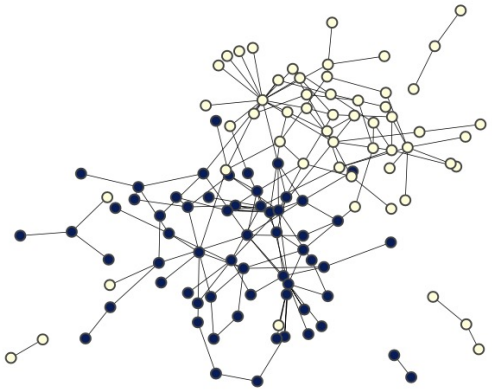
How can we sample a subgraph with small k?

Proposed Method (1)

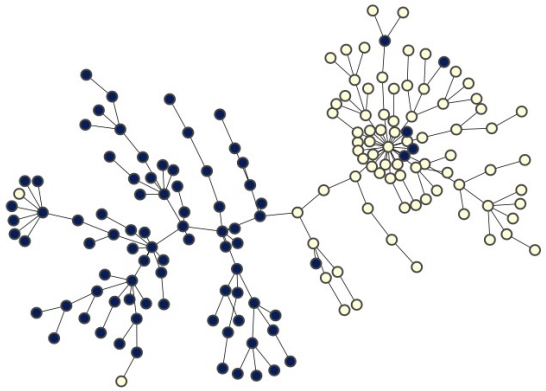
- **BTW (Bounded Treewidth Sampling)**
  - Samples a subset of original edges
  - Given k, it generates a subgraph U such that
    - The treewidth of U is smaller than or equal to k



(a) Original graph.
(TW ≈ 24)



(b) Sampled subgraph by RE.
(TW ≈ 8)



(d) Sampled subgraph by BTW.
(TW = 1)

Proposed Method (2)

- BTW runs iterations to select nodes
  - Initializes a k-tree K and a subgraph U
  - Adds a node to K and U until $|\mathcal{V}_U| = |\mathcal{V}|$
    - The treewidth of U is always bounded by k due to K

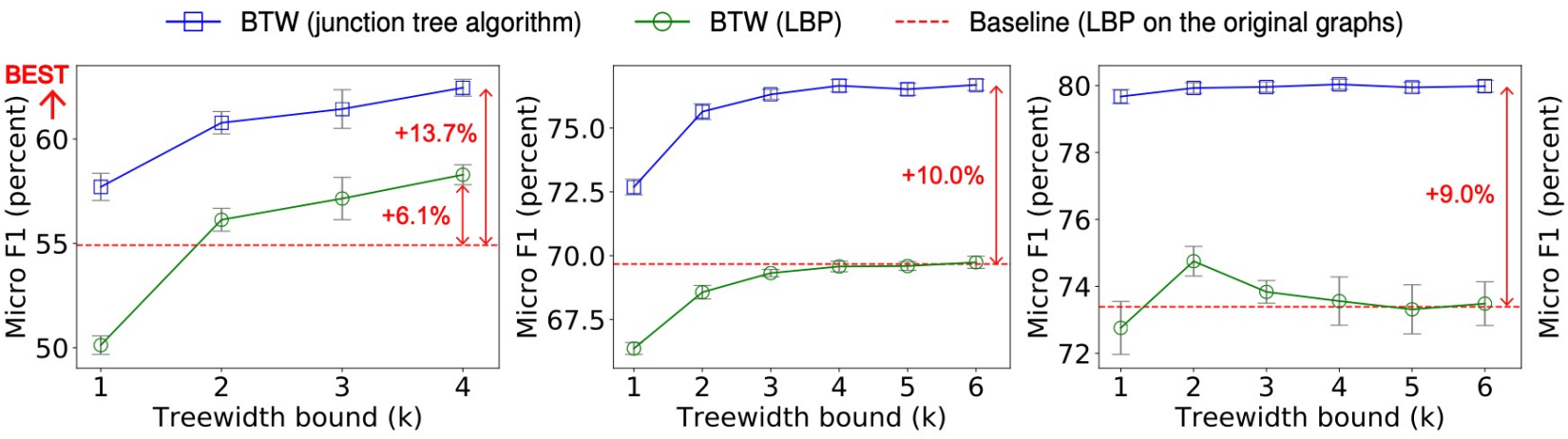
Algorithm 1 Bounded treewidth (BTW) sampling

Require: a graph $G = (\mathcal{V}, \mathcal{E})$ and a bound k
Ensure: a sampled subgraph U and a k-tree K

$K, U, H \leftarrow$ initialize(G, k)
while $|\mathcal{V}_U| < |\mathcal{V}|$ **do**
  $u, C \leftarrow$ next_node(H)
  $U \leftarrow (\mathcal{V}_U \cup \{u\}, \mathcal{E}_U \cup \{(u, v) | v \in \mathcal{N}_G(u) \cap C\})$
  $K \leftarrow (\mathcal{V}_K \cup \{u\}, \mathcal{E}_K \cup \{(u, v) | v \in C\})$
  $H \leftarrow$ update_heap(H, u, C)
end while
return U, K

Experiments (1)

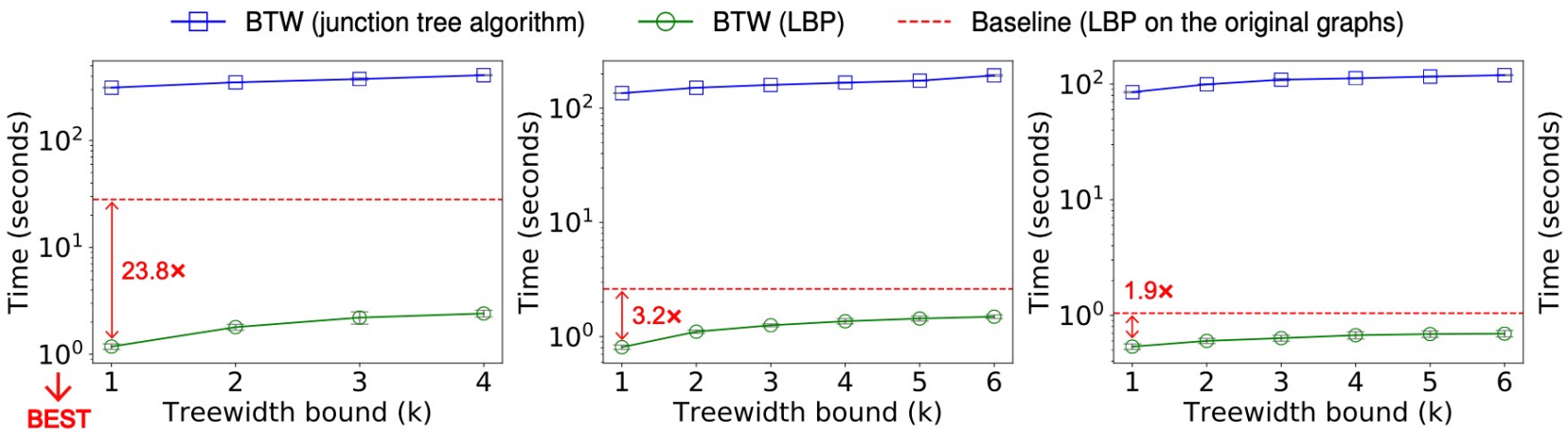
- Micro F1 score on three graph datasets
  - JT consistently shows higher accuracy
  - LBP shows similar (or even higher) accuracy



(a) Wikipedia. (b) CoRA. (c) PubMed.

Experiments (2)

- Inference time on three graph datasets
  - LBP is consistently faster than the baseline
  - JT is slower due to the exact inference



(e) Wikipedia. (f) CoRA. (g) PubMed.

Research Overview

- **Q.** How can we perform data augmentation on graphs for accurate graph classification?

	Structural Exploitation		Structural Modification
	$P(Y\|X,G)$	$P(X\|G)$	$P(G'\|G)$
Purely Probabilistic	Transductive Learning [ICDM'17]	-	Subgraph Sampling [WSDM'20]
Fusion with Deep Learning	Inductive Learning [IJCAI'19]	Feature Estimation [submitted]	**Graph Augmentation [submitted]**

Graph Classification

- This work studies **graph classification**
  - Given a set of training graphs
  - Predict the label y_i of each test graph G_i
- Difficulties of graph classification
  - **Fewer training examples** are available
    - # of graphs $\ll$ # of nodes
  - Hard to **summarize** the graph structures
    - Graphs have different # of nodes and edges

Graph Augmentation

- **Data augmentation** is a popular technique
  - Increases the amount of training data
  - Improves the generalizability of classifiers
- Graphs can also be augmented like images
  - Each graph corresponds to a single image

https://miro.medium.com/max/1400/0*LR1ZQucYW96prDte

Research Motivation

- We propose **five properties** for augmentation
  - Include scalability, unbiasedness, …
  - All previous works fail to satisfy them

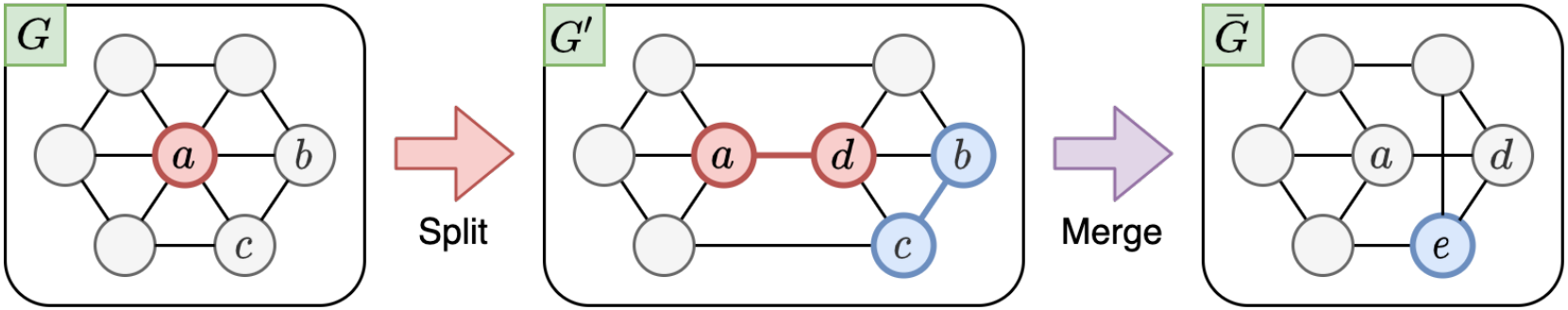
Method	P1	P2	P3	P4	P5
DropEdge [26]				✓	✓
GraphCrop [34]			✓	✓	✓
NodeAug [36]			✓	✓	✓
MotifSwap [51]	✓	✓			

Research Question

How can we design augmentation algorithms that satisfy all desired properties?

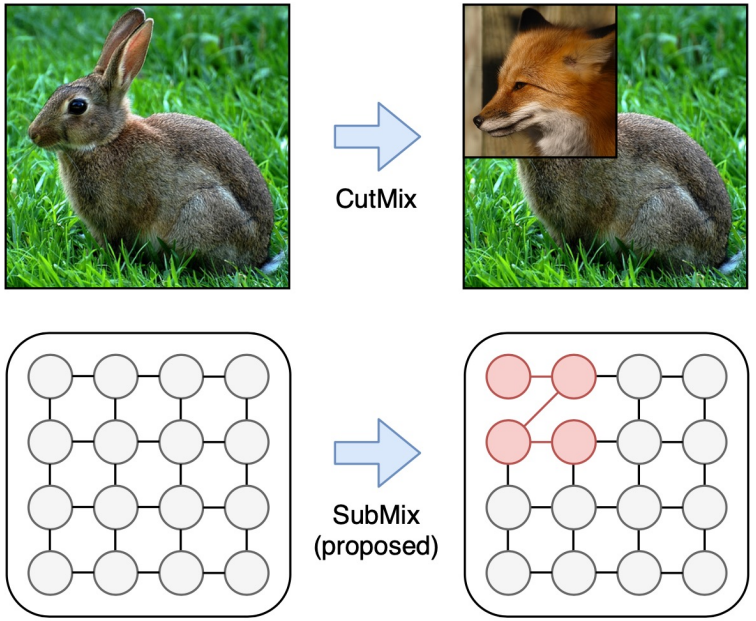
Proposed Methods (1)

- **NodeSam (Node Split and Merge)**
  - Performs two opposite operations at once
    - **Splits** a random node into adjacent nodes
    - **Merges** a random pair of connected nodes
  - Makes a balanced and stable change
    - Minimizes the amount of structural change



Proposed Methods (2)

- **SubMix (Subgraph Mix)**
  - Combines two different graphs by a mix approach
    - Motivated by CutMix in the image domain
    - Understands images as grid-structured graphs
  - Advantages
    - Makes a large degree of augmentation
    - Combines graphs even with different labels



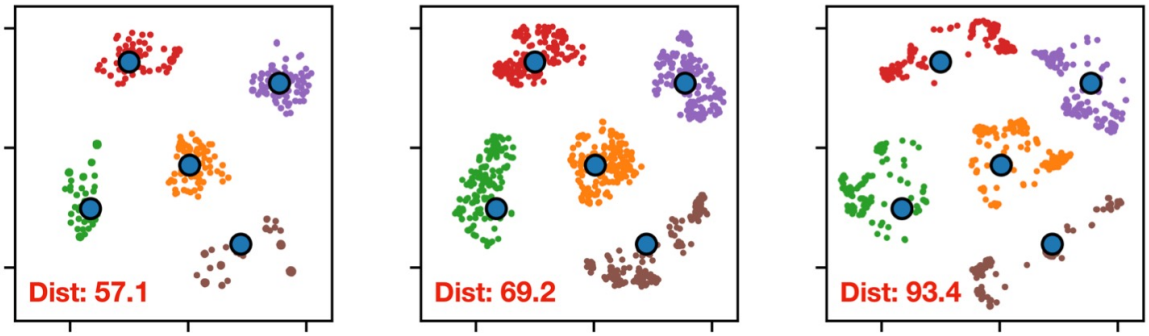
Experiments (1)

- Our methods make the **highest accuracy**
  - We measure the accuracy of graph classification
  - Previous methods often **decrease** the accuracy

Method	D&D	ENZYMES	MUTAG	NCI1	N109		Average	Rank
Baseline (no aug.)	76.40 (4)	50.33 (10)	89.94 (4)	82.68 (9)	81.80 (9)		74.35 (8)	7.43 ± 2.51
GraphCrop [34]	77.08 (2)	51.00 (9)	77.11 (10)	80.46 (10)	79.77 (10)		71.78 (10)	8.71 ± 2.98
DropEdge [26]	76.14 (6)	53.67 (6)	81.93 (9)	82.82 (7)	82.60 (7)		73.80 (9)	6.71 ± 1.60
AddEdge	76.14 (7)	55.17 (3)	85.67 (8)	83.99 (2)	83.06 (5)		74.81 (7)	5.43 ± 2.37
ChangeAttr	75.72 (9)	53.33 (8)	90.44 (3)	83.02 (5)	83.57 (2)	...	74.86 (6)	6.14 ± 2.85
NodeAug [36]	76.14 (8)	54.67 (5)	86.14 (7)	83.16 (4)	82.36 (8)		74.90 (5)	5.71 ± 2.21
DropNode	75.55 (10)	55.17 (3)	87.28 (6)	82.85 (6)	83.04 (6)		75.16 (4)	5.29 ± 2.81
MotifSwap [51]	76.23 (5)	53.50 (7)	90.47 (2)	82.82 (7)	83.28 (4)		75.43 (3)	4.43 ± 1.99
SubMix (proposed)	**78.10 (1)**	57.50 (2)	89.94 (4)	**84.33 (1)**	**84.37 (1)**		76.34 (2)	**2.29 ± 1.98**
NodeSam (proposed)	76.57 (3)	**60.00 (1)**	**90.96 (1)**	83.33 (3)	83.52 (3)		**76.57 (1)**	2.43 ± 1.13

Experiments (2)

- Our approaches make **higher diversity**
  - We visualize augmented graphs with t-SNE
  - **Avg. distance of augmentation:** (a) < (b) < (c)



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

Discussion

- We proposed two approaches that modify the structure of a given graph
  - **BTW** for subgraph sampling
  - **NS** (NodeSam and SubMix) for augmentation
- Structural modification is effective but difficult
  - There are no explicit answers
  - Performance largely depends on datasets
  - Risk of reducing the original performance

Summary of Approaches

- Our **BTW** and **NS** have different objectives
  - **BTW** for node classification
    - **Goal:** To make an efficient tree-like structure
    - **How:** By removing most of the original edges
    - **Assumption:** connectivity ≫ communities
  - **NS** for graph classification
    - **Goal:** To make a new graph with similar properties
    - **How:** By making balanced and unbiased changes
    - **Assumption:** both communities & connectivity

BTW for Augmentation

- BTW can be used for graph augmentation
  - Each sampled graph becomes a new graph
- The assumptions of BTW should be satisfied
  - BTW changes many properties of graphs
  - BTW removes most of existing communities
  - BTW focuses on the efficiency of structures

Applicable Scenarios

- When do we use BTW?
  - (+) When a graph is too large to handle
  - (–) When a graph contains edge attributes
  - (–) When community information is important
- When do we use NS?
  - (+) When a graph structure is delicate
  - (+) When there is no domain knowledge
  - (–) When the size of each graph is too small

Considering Attributes

- Both BTW and NS rely only on structures
  - To develop general and robust algorithms
  - That is, node or edge attributes are not used
- They can be improved in this perspective
  - Edge attributes
    - Represent the importance & role of each edge
  - Node attributes
    - Imply the similarities between adjacent nodes

Outline

- Introduction
- Background
- Structural exploitation
- Structural modification
- **Conclusion**

Conclusion

- **Research goals**
  - To connect features, labels, and a structure
  - To make solutions for challenging graph tasks
- **Contributions**
  - **SBP** and **BPN** for direct node classification
  - **MGA** for missing feature estimation
  - **BTW** for sampling tree-like subgraphs
  - **NodeSam** and **SubMix** for graph augmentation

Discussion (1)

- I'm interested in **probabilistic approaches**
- **Strengths**
  - Straightforward decision rules & interpretability
  - Easily generalizable to unseen test data
- **Limitations**
  - Performance relies on the choice of distributions
  - Limited representation power in large datasets
  - Domain knowledge may often be required

Discussion (2)

- Fusion with deep learning solves the problem
  - We utilized neural networks as sub-modules
    - **BPN** uses an MLP to estimate node priors
    - **MGA** uses a GNN to generate latent variables
  - This resulted in strong but robust approaches
- How I proceeded my research
  - Structural exploitation → structural modification
  - Purely probabilistic → deep learning-based ways

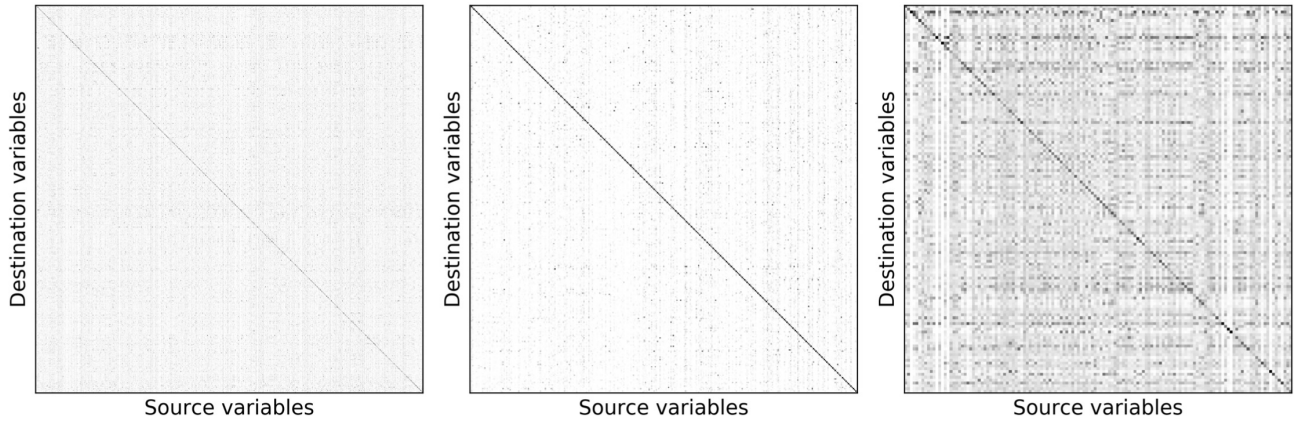
Future Work (1)

- One future work is to **learn** a graph structure
  - That is, learn a structure G only from features X

	Structural Exploitation		Structural Modification	**Structural Learning**
	$P(Y\|X,G)$	$P(X\|G)$	$P(G'\|G)$	$P(G\|X)$
Purely Probabilistic	Transductive Learning [ICDM'17]	-	Subgraph Sampling [WSDM'20]	**Future Work**
Fusion with Deep Learning	Inductive Learning [IJCAI'19]	Feature Estimation [submitted]	Graph Augmentation [submitted]	

Future Work (2)

- I studied the problem of learning correlations between time series variables
  - **AttnAR [SDM'21]** for general data domains
  - **DTML [KDD'21]** for stock price movements



(a) Traffic (b) Electricity (c) Solar-Energy

Future Work (3)

- However, both AttnAR and DTML make **dense correlation matrices** of size $|V|^2$
  - $|V|$ refers to the number of target variables
- **Goal:** to make sparse graphs of size $|E| \ll |V|^2$
  - A graph is more informative with less edges
  - Dense matrix of size $|V|^2$ provides no meaningful information for the independence

References

- [1] J. Yoo, S. Jo, and U Kang, “Supervised Belief Propagation: Scalable Supervised Inference on Attributed Networks”, **ICDM 2017**
- [2] J. Yoo, H. Jeon, and U Kang, “Belief Propagation Network for Hard Inductive Semi-Supervised Learning”, **IJCAI 2019**
- [3] J. Yoo, U Kang, M. Scanagatta, G. Corani, and M. Zaffalon, “Sampling Subgraphs with Guaranteed Treewidth for Accurate and Efficient Graphical Inference”, **WSDM 2020**
- [4] J. Yoo and U Kang, “Attention-Based Autoregression for Accurate and Efficient Multivariate Time Series Forecasting”, **SDM 2021**
- [5] J. Yoo, Y. Soun, Y. Park, and U Kang, “Accurate Multivariate Stock Movement Prediction via Data-Axis Transformer with Multi-Level Contexts”, **KDD 2021**

Thank You

Jaemin Yoo (jaeminyoo@snu.ac.kr)