Probabilistic Approaches to Solving Node-Level Tasks in Real Graphs

Jaemin Yoo

Postdoctoral Research Fellow

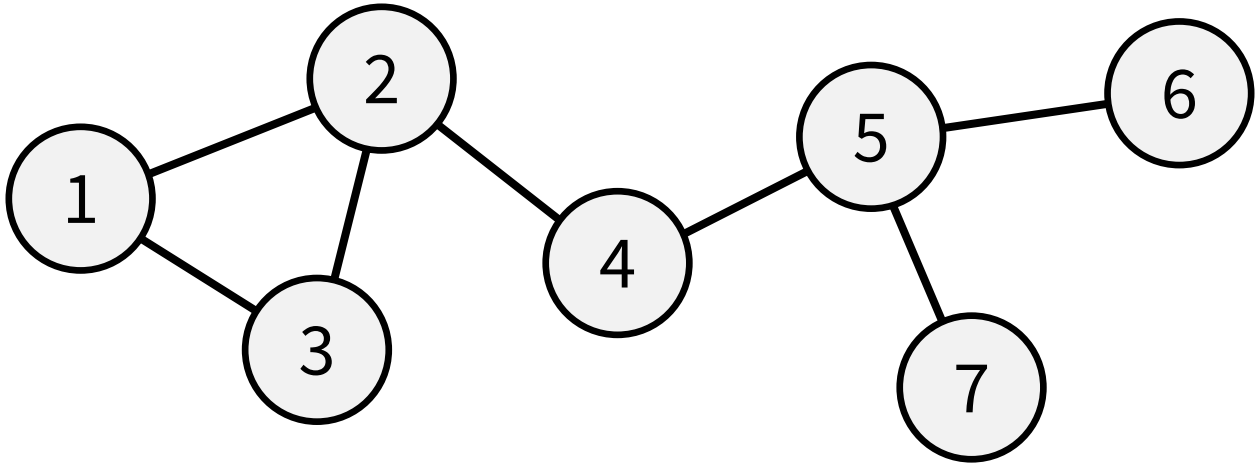
Carnegie Mellon University

Outline

- **<u>Introduction</u>**
- Part 1: Probabilistic modeling of graphs
- Part 2: Proposed approach for feature estimation
- Conclusion

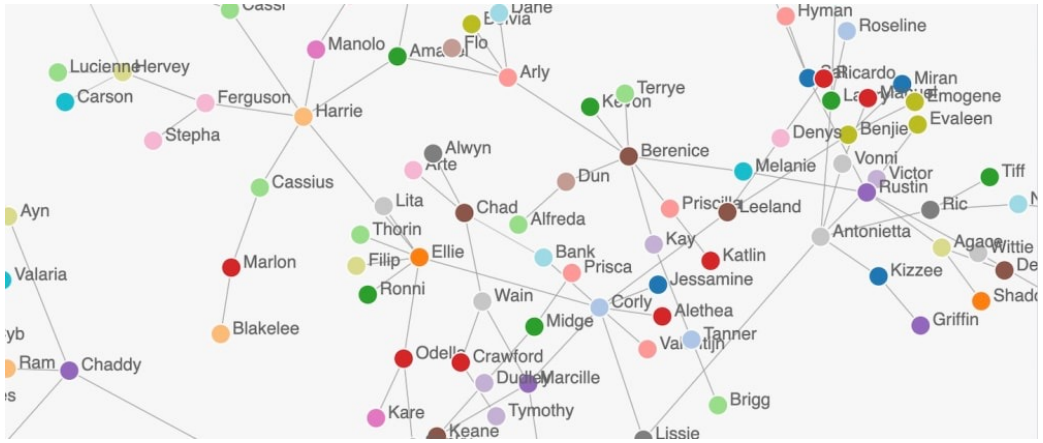
Graphs (1/2)

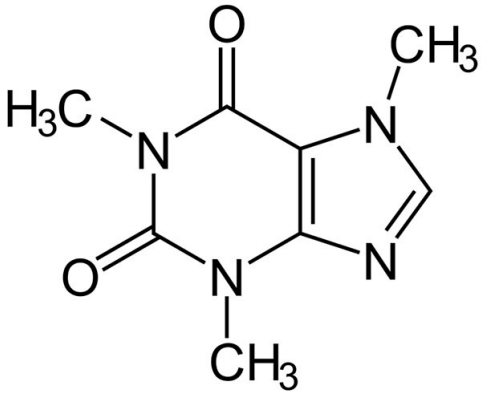
- **Graphs** are a data structure for modeling relationships
  - Graph consists **nodes** that are connected by **edges**
- Graphs provide meaningful insights and observations
  - E.g., nodes 1, 2, and 3 are closely related to each other



Graphs (2/2)

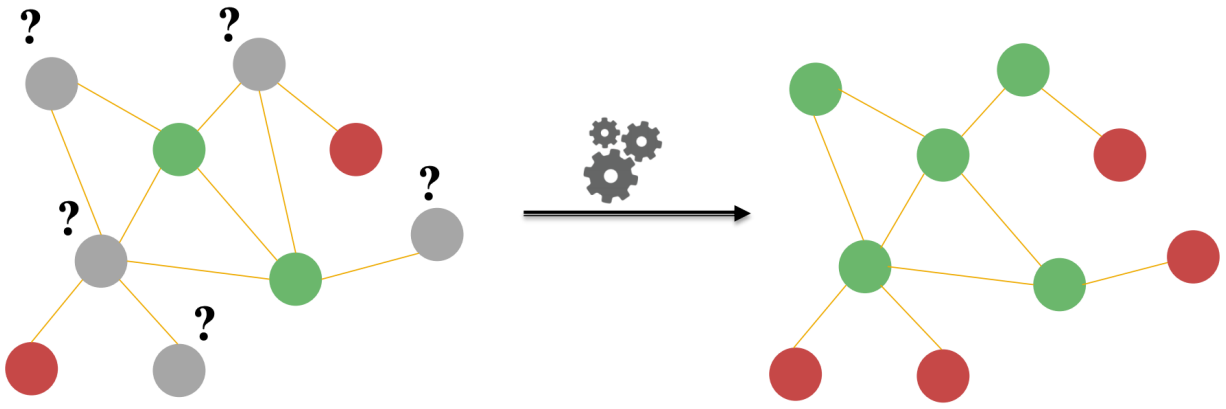
- Graph-structured data are very common in real-world applications
  - **Social network** represents the friendships between users
  - **Review system** models user experience for items as graphs
  - **Chemical compound** consists of interactions between elements





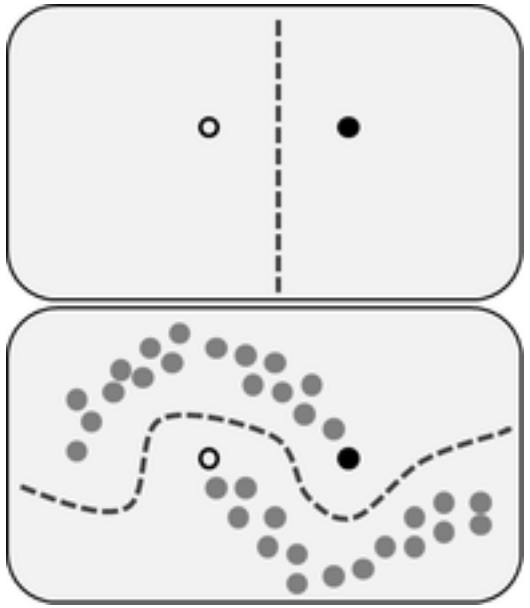
Node Classification (1/2)

- **Node classification** is a fundamental task on graphs
  - **Given** a graph G and the labels of some nodes $\mathcal{V}_{\text{train}}$
  - **Problem** is to predict the labels of remaining nodes $\mathcal{V}_{\text{test}}$
- Real-world applications:
  - Finding malicious users in a social network
  - Classifying new items in an e-commerce graph
  - Finding patients that may have a certain disease



Node Classification (2/2)

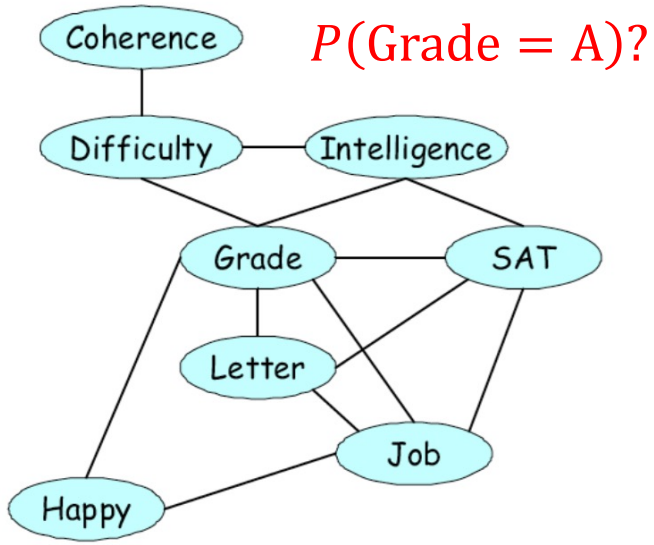
- **Node classification** is a generalization of many node-level tasks
  - **Semi-supervised learning**
    - Node features and given, but only a few nodes have labels
  - **Personalized recommendation**
    - Personalized node labels are assigned given a query user
  - **Node clustering**
    - Structure-based labels are assigned to a few anchor nodes
  - **Anomaly detection**
    - Only the “normal” node labels are observed during training



https://en.wikipedia.org/wiki/Semi-supervised_learning

Probabilistic Approaches

- **Main idea**: To understand a graph G as a probabilistic model M
  - Each node i is a (discrete) random variable $Z_i \in \{c_1, \dots, c_L\}$
  - Each edge (i, j) represents the relationship between Z_i and Z_j
- **Advantages**
  - Resulting algorithms are easily justified and interpretable
  - Effectively propagate observed information with few parameters

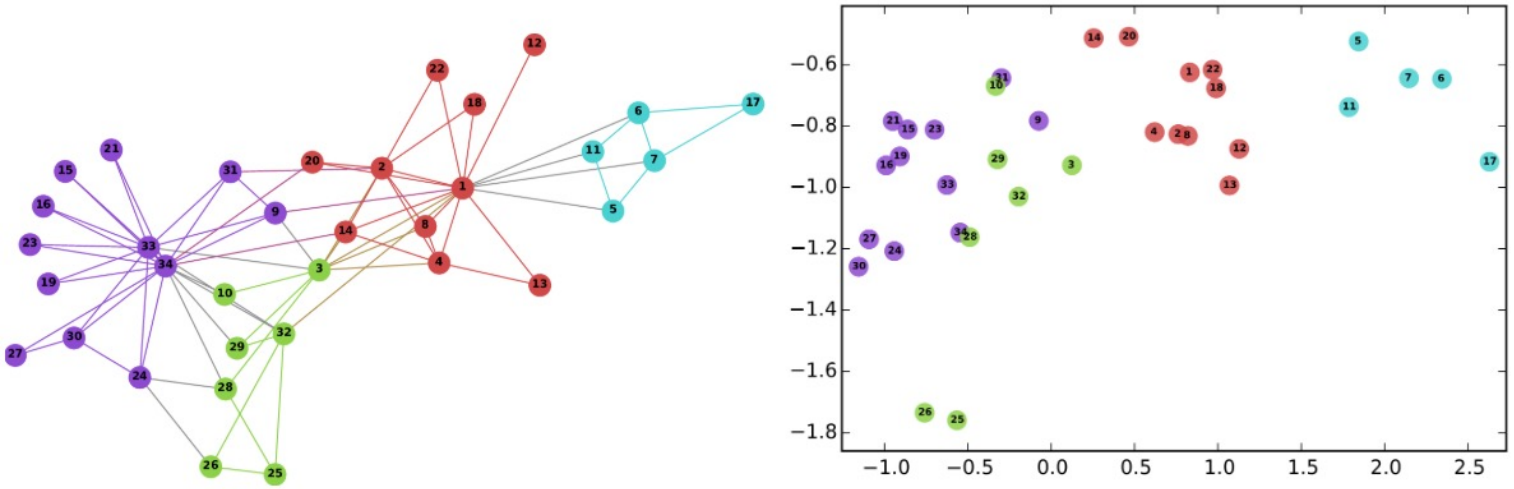


$P(\text{Grade} = \text{A})?$

https://stats.stackexchange.com/questions/244796/is-factorization-of-markov-random-field-unique

Graph Neural Networks (GNN)

- **Main idea:** To learn embeddings of nodes in an end-to-end way
  - Generates a low-dimensional representation $\mathbf{h}_i$ for each node i
  - Done by multiple layers of **graph convolutions** having nonlinearity

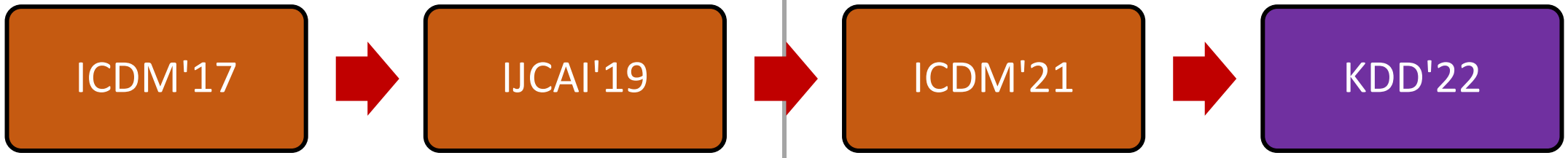


Perozzi et al., “DeepWalk: Online Learning of Social Representations”, KDD 2014

Overview of This Talk

- **Goal:** Combine ***probabilistic modeling*** and ***deep learning*** on graphs
  - To design interpretable, robust, and generalizable approaches

1. Differentiable probabilistic inference
2. Modeling intractable likelihoods



- This talk consists of two parts:
  - **Part 1:** Introduce the first three works on probabilistic modeling
  - **Part 2:** Introduce the last work [KDD'22] on node feature estimation

Outline

- Introduction
- **Part 1: Probabilistic modeling of graphs**
  - **<u>Preliminaries</u>**
  - Our previous works
- Part 2: Proposed approach for feature estimation
- Conclusion

Markov Random Fields

- **Q:** How to determine the probabilistic properties of G?
- We model G as a **pairwise Markov random field (MRF)**
  - Define the **joint probability** of all nodes $\mathbf{Z}$ as follows:

$$p(\mathbf{z}) = \frac{1}{C} \prod_{i \in \mathcal{V}} \phi_i(z_i) \prod_{(i,j) \in \mathcal{E}} \psi_{ij}(z_i, z_j)$$

  - where ϕ_i and ψ_{ij} are called **node** and **edge potential** functions

$\Rightarrow$ Higher local potentials make a higher global probability $p(\mathbf{z})$

Node Potentials

- **Node potential** ϕ_i represents the prior information of node i
- Assume a binary random variable $Z_i \in \{D, R\}$
  - $\phi_i(D) > 0.5$ means that node i is more likely to be a democrat
  - E.g., we have some evidence about the (unknown) state of node i

$$p(\mathbf{z}) = \frac{1}{C} \prod_{i \in \mathcal{V}} \phi_i(z_i) \prod_{(i,j) \in \mathcal{E}} \psi_{ij}(z_i, z_j)$$

Edge Potentials

$$p(\mathbf{z}) = \frac{1}{C}\prod_{i\in\mathcal{V}}\phi_i(z_i)\prod_{(i,j)\in\mathcal{E}}\psi_{ij}(z_i, z_j)$$

- **Edge potential** ψ_{ij} represents the correlation of nodes i and j
- Assume a binary random variable $Z_i \in \{D, R\}$
  - **Compatibility matrix** $\mathbf{M} \in \mathbb{R}^{l\times l}$ models all pairs of l discrete states:

$$\mathbf{M} = \begin{bmatrix} 0.4 & 0.1 \\ 0.1 & 0.4 \end{bmatrix} \text{ with homophily}$$

- $\psi_{ij}(D, D) = 0.4$
- $\psi_{ij}(D, R) = 0.1$

  - **Homophily** means that adjacent nodes are **positively** correlated
  - **Heterophily** is also possible; adjacent nodes are **negatively** correlated
    - In this case, non-diagonal entries are larger

Loopy Belief Propagation (1/2)

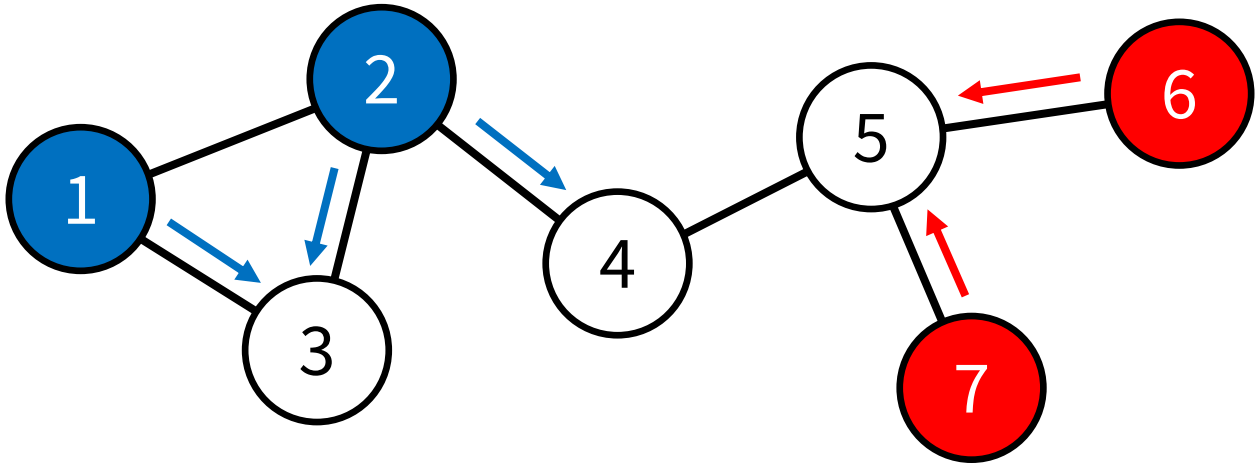
- The computation of the joint probability $p(\mathbf{Z})$ is **intractable**
  - Because of the $O(2^N)$ complexity of the **partition function** C
- **Loopy belief propagation (BP)**
  - Inference algorithm to approximate **marginal probabilities**
  - BP computes the belief $\mathbf{b}_i$ of each node $i \in G$ such that

$$b_i(z_i) \approx P_i(Z_i = z_i) = \sum_{\mathbf{z} \setminus z_i} P(Z_i = z_i \text{ and } \mathbf{Z} \setminus Z_i = \mathbf{z} \setminus z_i)$$

  - The resulting belief $\mathbf{b}_i$ can be directly used for its classification

Loopy Belief Propagation (2/2)

- BP is also known as the **message passing algorithm**
  - Because it propagates **messages** via iterations
  - Exchange the priors of nodes based on edge potentials
- Example of message passing in a graph with two states:

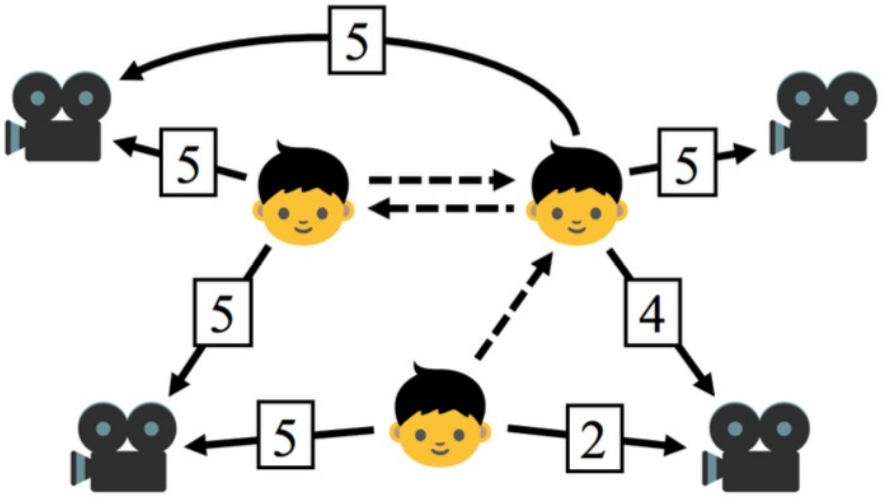


Outline

- Introduction
- **Part 1: Probabilistic modeling of graphs**
  - Preliminaries
  - **<u>Our previous works</u>**
- Part 2: Proposed approach for feature estimation
- Conclusion

SBP [ICDM'17] – Problem

- **Problem:** Node classification on **edge-attributed graphs**
- Many real-world graphs have attributes for describing their edges
  - **Rating scores** in a review network
  - **Types of interactions**, e.g., messages or posts, in a social network
- The specifics of relationships are determined by such attributes
  - Each edge can follow **homophily** or **heterophily** based on its feature



SBP [ICDM'17] – Main Ideas

- **Q:** How can we learn accurate ψ_{ij} in edge-attributed graphs?
- We propose **SBP** to learn dynamic edge potentials in BP
  - **Idea 1:** Model BP as a differentiable operator with fixed iterations
  - **Idea 2:** Model the edge potential $\psi_{ij}(\cdot)$ as a function of $\mathbf{x}_{ij}$ and θ:

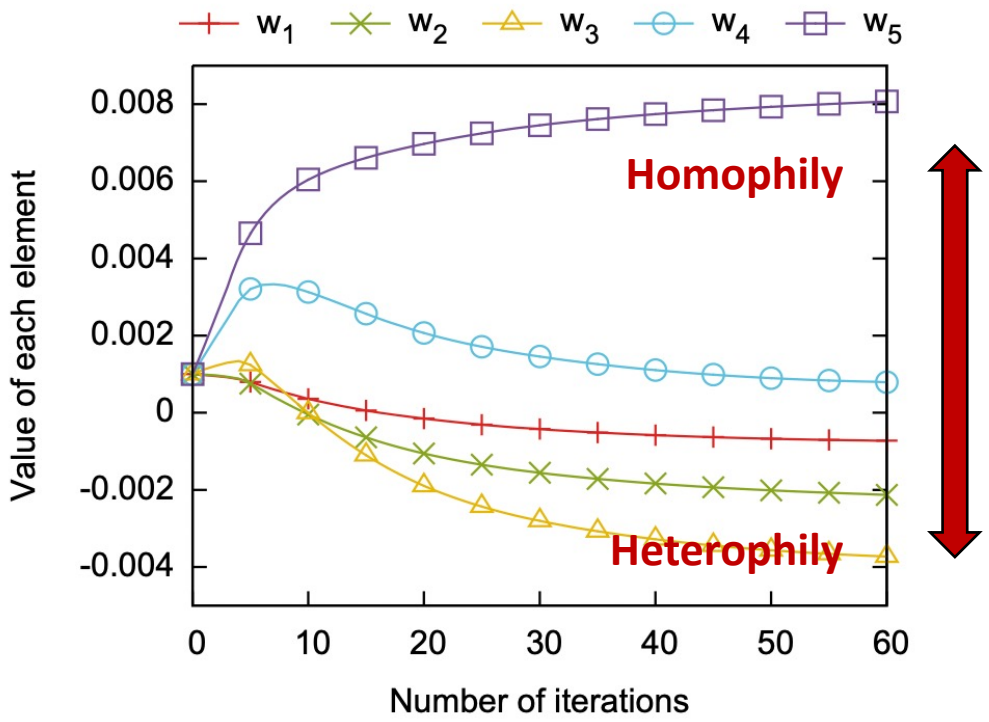
$$\psi_{ij}(\mathbf{x}_{ij};\theta) = \frac{1}{1+\exp(-\mathbf{x}_{ij}^{\top}\theta)}$$

- x_{ij}: Edge feature
- θ: parameters

  - **Idea 3:** Learn the parameters θ to minimize an objective function

SBP [ICDM'17] – Experiments

- **Experimental setting**
  - Recommendation task on review networks of ratings 1 to 5
- **Observations**
  - The strength of homophily increases with the rating score from 1 to 5
  - Low ratings even follow heterophily
    - **Homophily:** ratings 4 and 5
    - **Heterophily:** ratios 1, 2, and 3



BPN [IJCAI'19] – Problem

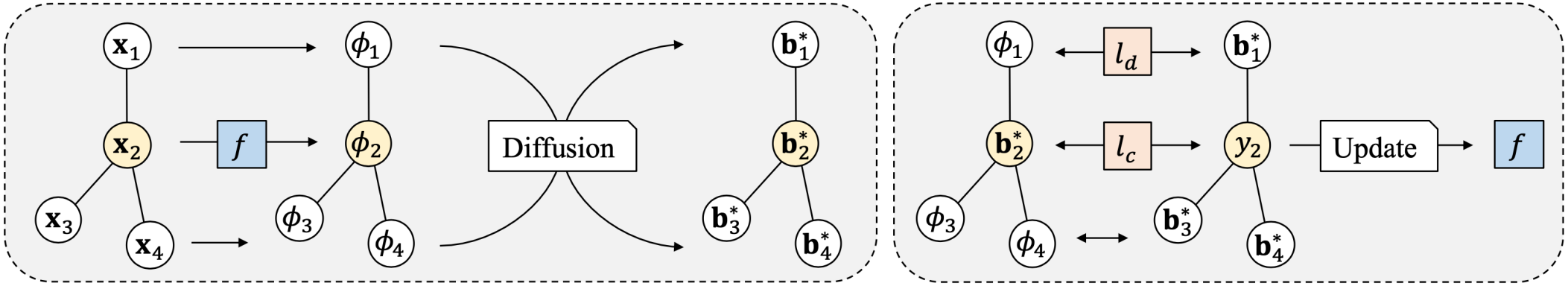
- **Problem: Cold-start inductive learning** for node classification
  - In SBP, we assume a single graph used for both training and test
  - In this work, unseen test nodes are given **without any neighborhood**
- Real-world scenarios:
  - When **new users** are added to an online social network
  - When **new movies** are uploaded to a streaming service

Existing (or training) graph

New user i
(+ feature $\mathbf{x}_i$)

BPN [IJCAI'19] – Main Ideas

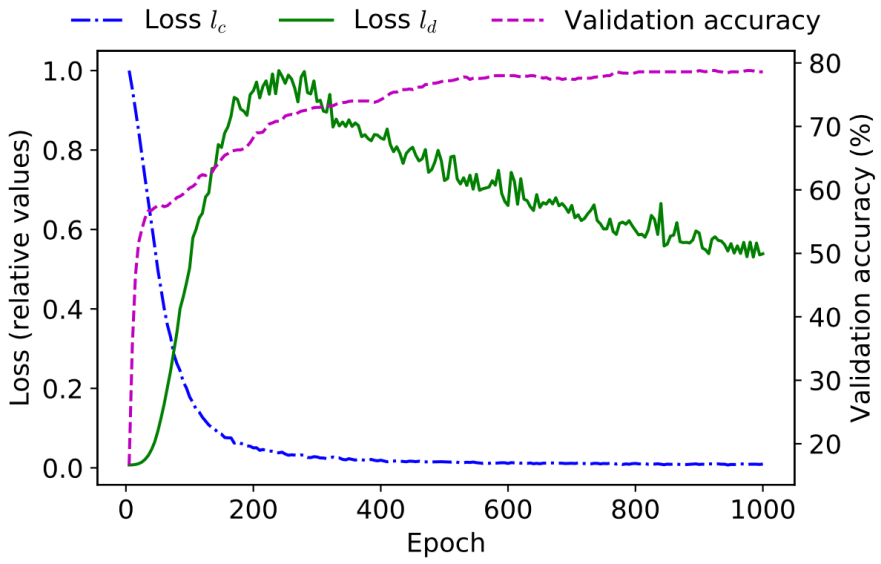
- **Q:** How can we learn a feature-based classifier f with few labels?
- We propose **BPN**, a framework for training f on inductive learning
  - **Idea 1:** Introduce a multilayer perceptron as a classifier f
  - **Idea 2:** Use the prediction of f as the node potential $\phi_i = f(\mathbf{x}_i, \theta)$
  - **Idea 3:** Run BP to give pseudo answers to the training of f



(a) Forward propagation of BPN. (b) Backward propagation of BPN.

BPN [IJCAI'19] – Experiments

- BPN effectively minimizes two different loss functions
  - The **classification loss** l_c is minimized stably through iterations
  - The **induction loss** l_d increases at first and then is minimized slowly
    - Because the updates of f change the result of diffusion, i.e., beliefs

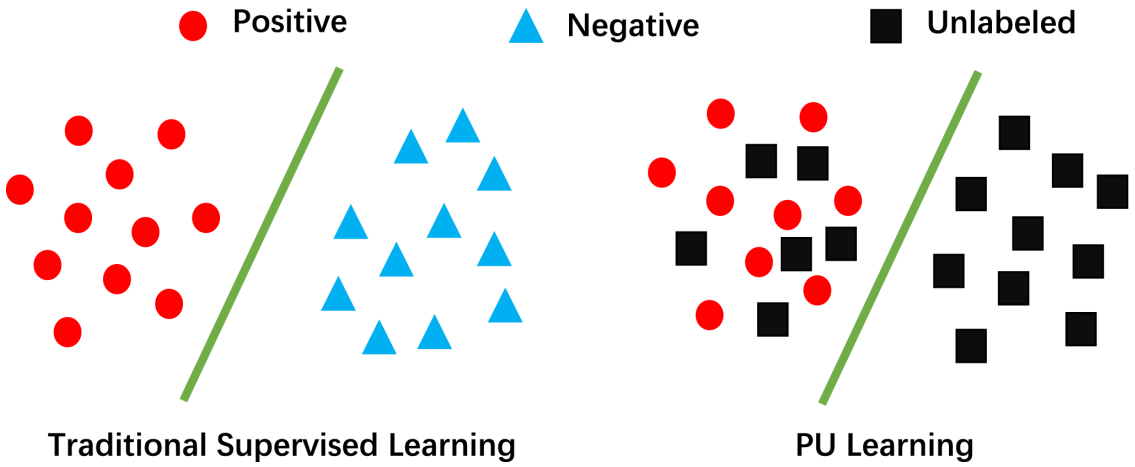


BPN outperforms various GNN models

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)	$\mathbf{78.3 \pm 0.3}$	$\mathbf{72.2 \pm 0.5}$	$\mathbf{70.1 \pm 0.9}$	$\mathbf{81.5 \pm 1.3}$

GRAB [ICDM'21] – Problem

- **Problem:** Graph-based positive-unlabeled (PU) learning
  - Goal is to train an accurate binary node classifier for $\mathcal{P}$ vs. $\mathcal{N}$
  - However, negative labels are **unseen** during training
    - Decision boundary should be drawn from positive and unlabeled samples



GRAB [ICDM'21] – Main Ideas

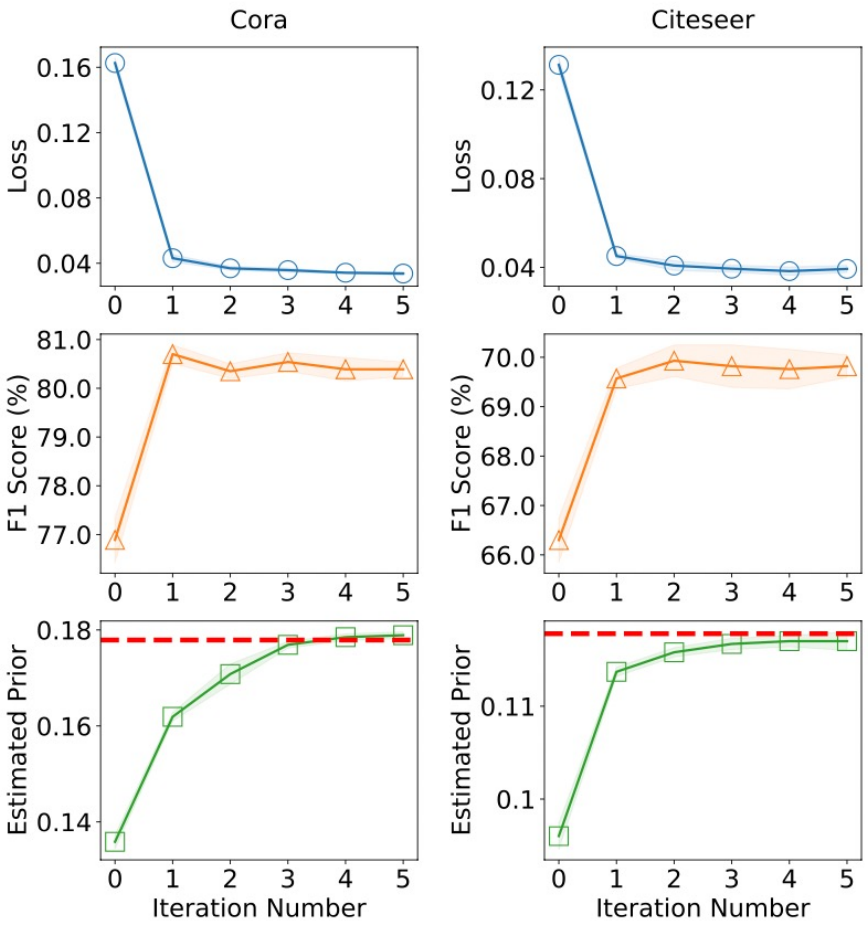
- **Q:** How can we design an objective function from PU nodes?
- We propose **GRAB** for training a node classifier on PU learning
  - **Idea 1:** Introduce latent variables $\mathbf{Z}$ for modeling unknown states

$$l(\theta) = \underbrace{\sum_{i \in \mathcal{P}} (-\log \hat{y}_i(+1))}_{\text{For positive nodes } \mathcal{P}} + \underbrace{\mathbb{E}_{\mathbf{z} \sim p(\mathbf{z}|\mathbf{X},\mathbf{y})} \left[\sum_{j \in \mathcal{U}} \left(-\log \hat{y}_j(z_j)\right) \right]}_{\text{For unlabeled nodes } \mathcal{U}}$$

  - **Idea 2:** Model the distribution $p(\mathbf{Z}|\mathbf{X},\mathbf{y})$ of $\mathbf{Z}$ as an MRF
  - **Idea 3:** Replace the expectation with marginals computed from BP

GRAB [ICDM'21] – Experiments

- **Experimental setting**
  - We use a GCN classifier f and use the **EM algorithm** for its training
  - **Why EM?** Since the updates of f also change the distribution of $\mathbf{Z}$
- **Observations**
  - Training proceeds well via iterations
  - The ratio of predicted labels, which is called a *prior*, is estimated well



Outline

- Introduction
- Part 1: Probabilistic modeling of graphs
- **Part 2: Proposed approach for feature estimation**
  - **<u>Motivation</u>**
  - Proposed approach
  - Experiments
- Conclusion

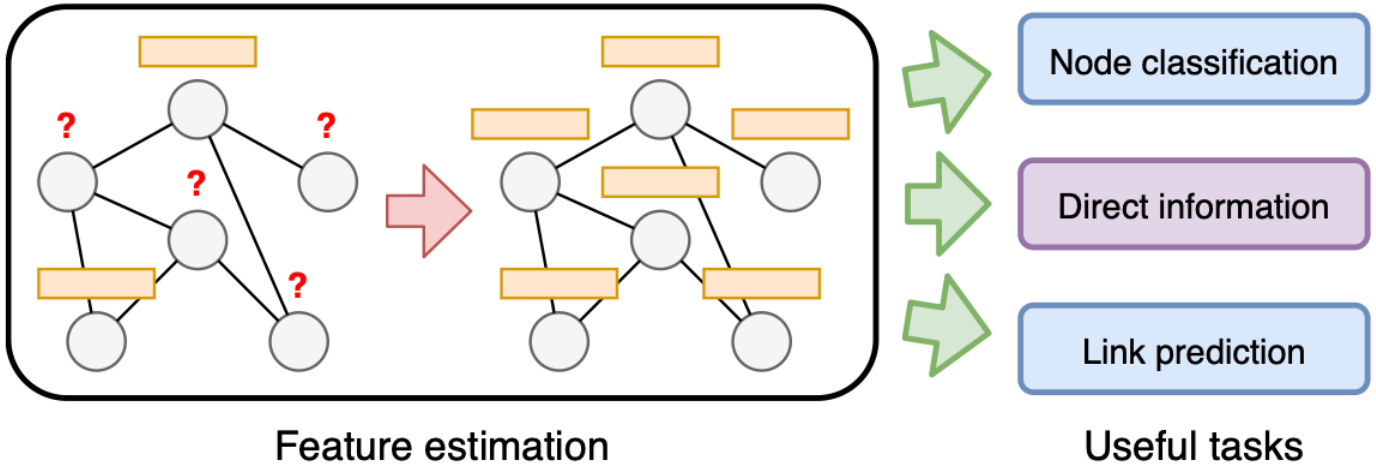
Node Features in Graphs

- Real-world graphs contain **node features (or attributes)**
  - Information of users in a social network
  - Abstracts of papers in a citation network
- Node features are essential for many graph-based tasks

https://www.shortstack.com/blog/best-social-networks-to-reach-specific-demographics

Missing Observations

- However, missing features are common in (large) real graphs
  - E.g., users in a social network with private profiles
- In such cases, it is difficult even to use existing node features
- **Question:** *How can we accurately estimate missing features?*



Problem Definition

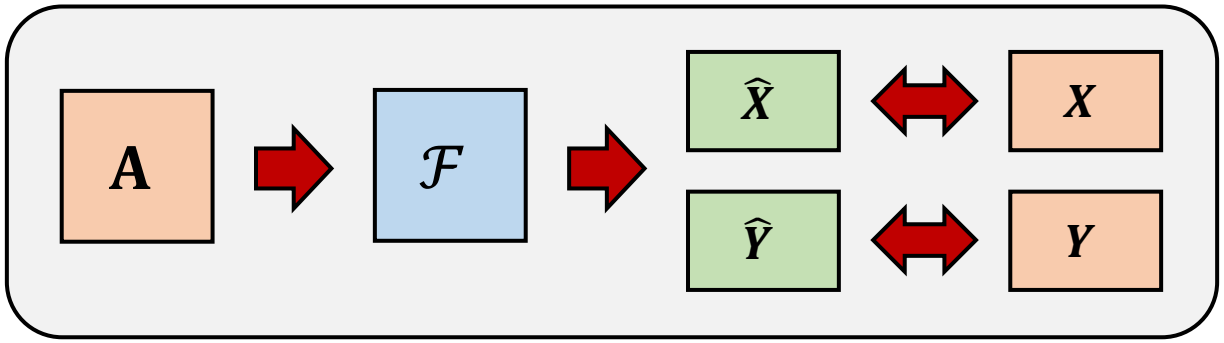
- **Problem:** Node feature estimation
- **Given**
  - An undirected graph $G = (\mathcal{V}, \mathcal{E})$
  - Node feature $\mathbf{x}_i$ for some nodes in $\mathcal{V}_x \subset \mathcal{V}$
  - (Optional) node labels y_i for nodes in $\mathcal{V}_y \subseteq \mathcal{V}$
    - Discrete labels are often easier to acquire
- **Predict**
  - Unknown feature $\mathbf{x}_j$ for nodes in $\mathcal{V} \setminus \mathcal{V}_x$

Dual Estimation

- We formulate the problem as finding Θ that maximizes

$$p_{\Theta}(\mathbf{X}, \mathbf{y}|\mathbf{A}) \text{ with } \hat{\mathbf{X}}, \hat{\mathbf{y}} = \mathcal{F}(\mathbf{A}; \Theta)$$

  - $\mathcal{F}$ is our target estimator, and Θ is the set of its parameters
  - That is, we use $\mathbf{X}$ and $\mathbf{y}$ as the estimation targets, not as inputs



Variational Inference (1/3)

- **Question:** How can we effectively maximize $p_\Theta(\mathbf{X}, \mathbf{y}|\mathbf{A})$?
- **Main approach:** We run **variational inference** with latent $\mathbf{Z}$

$$\begin{aligned}\log p_\Theta(\mathbf{X}, \mathbf{y} \mid \mathbf{A}) &\geq \mathcal{L}(\Theta) \\ &= \mathbb{E}_{\mathbf{Z}\sim q_\phi(\mathbf{Z}|\mathbf{X},\mathbf{y},\mathbf{A})}[\log p_{\theta,\rho}(\mathbf{X}, \mathbf{y} \mid \mathbf{Z}, \mathbf{A})] \\ &\quad - D_{\mathrm{KL}}(q_\phi(\mathbf{Z} \mid \mathbf{X}, \mathbf{y}, \mathbf{A}) \mid\mid p(\mathbf{Z} \mid \mathbf{A})),\end{aligned}$$

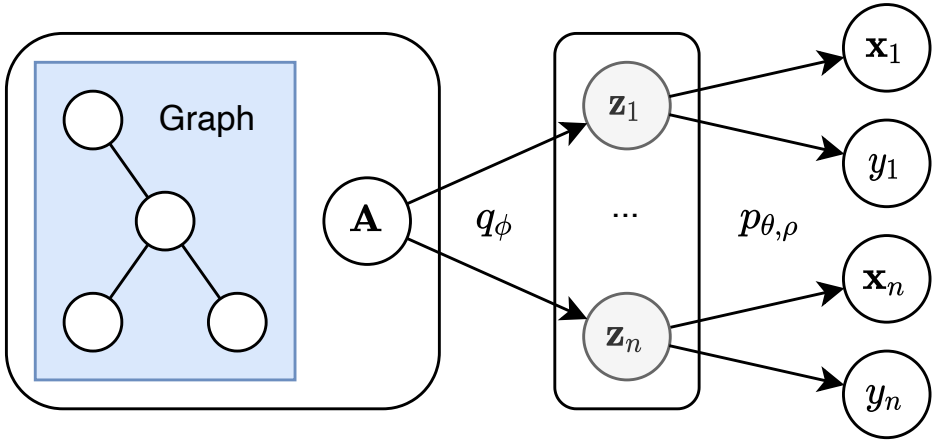
- where $\mathcal{L}(\Theta)$ is the **evidence lower bound (ELBO)** of the likelihood

Variational Inference (2/3)

- The **first term** of ELBO is the reconstruction error of $\mathbf{X}$ and $\mathbf{y}$:

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

- $\mathbf{Z}$ introduces the **conditional independence** between variables
  - Based on the assumption that $\mathbf{Z}$ has sufficient information of $\mathbf{A}$
  - This allows us to use separate decoding processes for $\mathbf{x}_i$ and y_i



Variational Inference (3/3)

- The **second term** of ELBO regularizes the distribution $q_\phi(\mathbf{Z})$ of $\mathbf{Z}$:

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

- This forces the distribution $q_\phi(\mathbf{Z}|\mathbf{X}, \mathbf{y}, \mathbf{A})$ to be closer to $p(\mathbf{Z}|\mathbf{A})$
  - The prior $p(\mathbf{Z}|\mathbf{A})$ contains no learnable parameters
  - The amount of regularization is determined by how to choose $p(\mathbf{Z}|\mathbf{A})$

Research Motivations

- Previous works ignore the **correlations** between $\mathbf{z}_i$ and $\mathbf{z}_j$ $\forall i \neq j$
  - By modeling $q_\phi(\mathbf{Z}) = \mathcal{N}(\boldsymbol{\mu}, \mathrm{diag}(\boldsymbol{\sigma}))$ and $p(\mathbf{Z}) = \mathcal{N}(0, \mathbf{I}_n)$
- However, the **correlations** are essential in our case
  - Since the graph itself means the correlations between different nodes
  - We cannot assume the conditional independence if we ignore them

Q1. *How can we consider the correlations in variational inference?*
Q2. *How can we design an efficient and stable way of inference?*

Outline

- Introduction
- Part 1: Probabilistic modeling of graphs
- **Part 2: Proposed approach for feature estimation**
  - Motivation
  - **<u>Proposed approach</u>**
  - Experiments
- Conclusion

Main Idea

- **Main idea:** To model the prior $p(\mathbf{Z}|\mathbf{A})$ as a **Gaussian MRF**
- **Differences from discrete MRF:**
  - Since $\mathbf{Z}$ is continuous, the choice of ψ_{ij} is not straightforward
  - We model ψ_{ij} based on a graph Laplacian $\mathbf{K} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$
  - This allows us to rewrite the KL divergence as

$$D_{\mathrm{KL}}(q_\phi(\mathbf{Z} \mid \mathbf{X}, \mathbf{y}, \mathbf{A}) \mid\mid p(\mathbf{Z} \mid \mathbf{A})) = 0.5(\mathrm{tr}(\mathbf{U}^\top \mathbf{K}\mathbf{U}) + d(\mathrm{tr}(\mathbf{K}\Sigma) - \log|\Sigma|)) + C,$$

  - where $q_\phi(\mathbf{Z})$ is assumed to be a multivariate Gaussian $\mathcal{N}(\mathbf{U}, \Sigma)$

Idea 1: Low-Rank Approximation

- **Observation:** The computation of $\log|\Sigma|$ is intractable
- **Idea 1:** We apply **low-rank approximation** to Σ as follows:

$$\Sigma = \beta \mathbf{I} + \mathbf{V}\mathbf{V}^\top,$$

  - β is a hyperparameter for the self-correlations
  - $\mathbf{V} \in \mathbb{R}^{n \times r}$ is a new embedding matrix introduced for Σ
- This reduces the complexity from $O(n^3)$ to $O(r^2 n + r^3)$

Idea 2: Unified Embedding

- **Observation:** The matrices $\mathbf{U}$ and $\mathbf{V}$ play similar roles in D_{KL}
  - The **two terms** in D_{KL} are identical if we assume $\mathbf{U} = \mathbf{V}$,

$$D_{\mathrm{KL}}(q_\phi(\mathbf{Z} \mid \mathbf{X}, \mathbf{y}, \mathbf{A}) \mid\mid p(\mathbf{Z} \mid \mathbf{A}))$$
$$= 0.5(\mathrm{tr}(\mathbf{U}^\top \mathbf{K}\mathbf{U}) + d(\mathrm{tr}(\mathbf{K}\Sigma) - \log|\Sigma|)) + C,$$

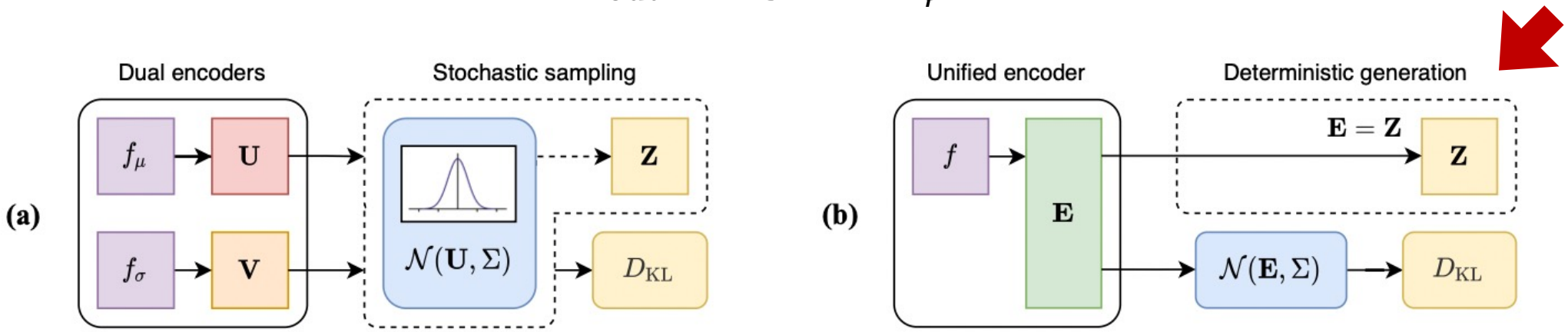
  - That is,

$$\mathrm{tr}(\mathbf{U}^\top \mathbf{K}\mathbf{U}) = \mathrm{tr}(\mathbf{K}\boldsymbol{\Sigma}) = \sum_{(i,j)\in\mathcal{E}} \left\| \frac{\mathbf{u}_i}{\sqrt{d_i}} - \frac{\boldsymbol{u}_j}{\sqrt{d_j}} \right\|$$

- **Idea 2:** We learn a single matrix $\mathbf{E}$ and use it as both $\mathbf{U}$ and $\mathbf{V}$

Idea 3: Deterministic Inference

- **Observation:** Stochastic sampling of n variables is unstable
  - Previous works sample z_i independently for each i
  - With correlations in $\mathbf{Z}$, the space of sampling becomes $O(2^n)$
- **Idea 3:** We generate deterministic $\mathbf{Z}$ from $\mathbf{E}$ without sampling
  - This is equivalent to taking $Z_{\text{out}} = \text{argmax}_Z\, q_\phi(Z)$



Objective Function

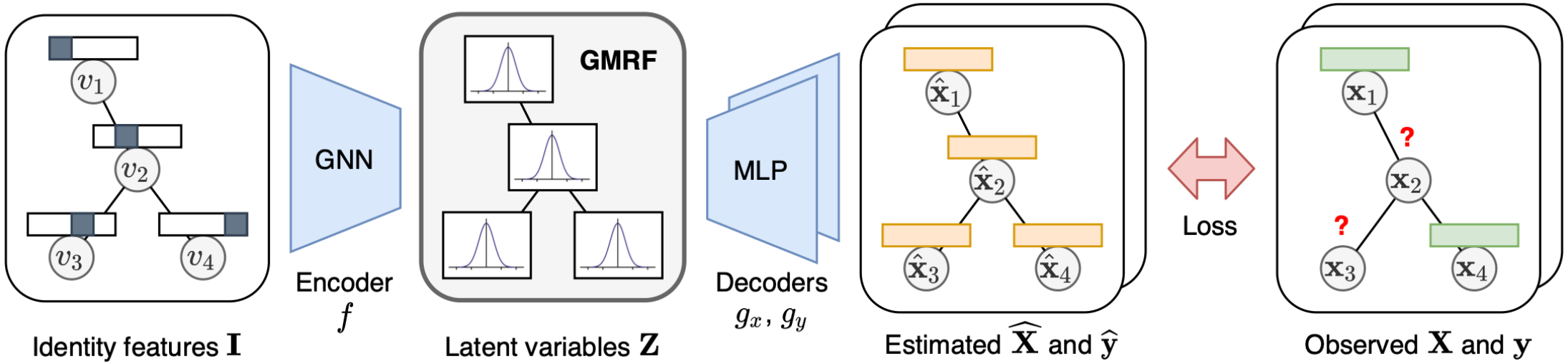
- Our three ideas result in the **proposed objective function** $l(\Theta)$:

$$\underbrace{\sum_{i \in \mathcal{V}_x} l_x(\hat{\mathbf{x}}_i, \mathbf{x}_i)}_{\text{Error for } \mathbf{X}} + \underbrace{\sum_{i \in \mathcal{V}_y} l_y(\hat{y}_i, y_i)}_{\text{Error for } \mathbf{y}} + \underbrace{\lambda(\mathrm{tr}(\mathbf{Z}^\top \mathbf{K} \mathbf{Z}) - \alpha \log|\mathbf{I} + \beta^{-1}\mathbf{Z}^\top \mathbf{Z}|)}_{\text{Proposed regularizer } l_{\mathrm{GMRF}}}$$

  - The **first term** makes adjacent nodes i and j have similar $\mathbf{z}_i$ and $\mathbf{z}_j$
  - The **second term** forces $\mathbf{Z}$ to be an orthogonal matrix
    - That is, different elements $\mathbf{z}_{:l}$ and $\mathbf{z}_{:k}$ are learned to minimize $\mathbf{z}_{:l}^\top \mathbf{z}_{:k}$

Proposed Architecture

- We propose **Structured Variational Graph Autoencoder (SVGA)**
  - GNN-based autoencoder model for dual estimation of features and labels
    - Only the features are used if we have no observed labels
  - **GNN encoder** generates latent variables, which are fed into **MLP decoders**



Outline

- Introduction
- Part 1: Probabilistic modeling of graphs
- **Part 2: Proposed approach for feature estimation**
  - Motivation
  - Proposed approach
  - **<u>Experiments</u>**
- Conclusion

Experimental Setup

- We compare SVGA with various competitors on real datasets
- We use eight public datasets of undirected graphs
  - The nodes in these graphs have either **binary** or **continuous features**

Dataset	Type	Nodes	Edges	Feat.	Classes
Cora[1]	Binary	2,708	5,429	1,433	7
Citeseer[1]	Binary	3,327	4,732	3,703	6
Photo[2]	Binary	7,650	119,081	745	8
Computers[2]	Binary	13,752	245,861	767	10
Steam[3]	Binary	9,944	266,981	352	1
Pubmed[1]	Continuous	19,717	44,324	500	3
Coauthor[2]	Continuous	18,333	81,894	6,805	15
Arxiv[4]	Continuous	169,343	1,157,799	128	40

Q1. Feature Estimation

- **Q1.** Does SVGA show higher accuracy in feature estimation?
- **A1.** SVGA performs best in binary and continuous node features
  - We use two evaluation metrics for each setting, respectively

Binary features

Metric	Model	Cora @10	Cora @20	Cora @50	Citeseer @10	Citeseer @20	Citeseer @50
Recall	NeighAgg	.0906	.1413	.1961	.0511	.0908	.1501
	VAE	.0887	.1228	.2116	.0382	.0668	.1296
	GNN*	.1350	.1812	.2972	.0620	.1097	.2058
	GraphRNA	.1395	.2043	.3142	.0777	.1272	.2271
	ARWMF	.1291	.1813	.2960	.0552	.1015	.1952
	SAT	.1653	.2345	.3612	.0811	.1349	.2431
	SVGA	**.1718**	**.2486**	**.3814**	**.0943**	**.1539**	**.2782**

Continuous features

Model	Pubmed RMSE	Pubmed CORR	Coauthor RMSE	Coauthor CORR	Arxiv RMSE	Arxiv CORR
NeighAgg	0.0186	-0.2133	0.0952	-0.2279	0.1291	-0.4943
VAE	0.0170	-0.0236	0.0863	-0.0237	0.1091	-0.4773
GNN*	0.0168	-0.0010	0.0850	0.0179	0.1091	0.0283
GraphRNA	0.0172	-0.0352	0.0897	-0.1052	0.1131	-0.0419
ARWMF	0.0165	0.0434	0.0827	0.0710	o.o.m.	o.o.m.
SAT	0.0165	0.0378	0.0820	0.0958	0.1055	0.0868
SVGA	**0.0158**	**0.1169**	**0.0798**	**0.1488**	**0.1005**	**0.1666**

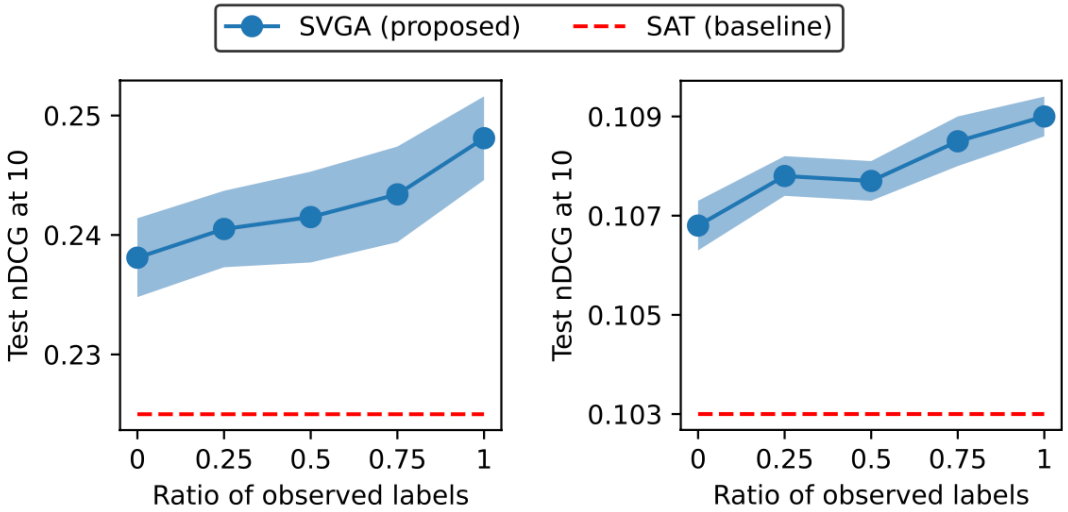
Q2. Node Classification

- **Q2.** Are the generated features meaningful for node classification?
- **A2.** SVGA works best with both MLP and GCN classifiers
  - We train a classifier on nodes whose features are generated by SVGA
  - The generated features support more accurate classification of labels

Model	Cora		Citeseer		Computers		Photo		Pubmed		Coauthor		Arxiv	
	MLP	GCN	MLP	GCN	MLP	GCN	MLP	GCN	MLP	GCN	MLP	GCN	MLP	GCN
NeighAgg	.6248	.8365	.5150	.6494	.8715	.6564	.5549	.8846	.7562	.5413	.9010	.8031	.3979	.6493
VAE	.2826	.3747	.4008	.3011	.4023	.4007	.2551	.2598	.2317	.2663	.3781	.2335	.1633	.1965
GNN*	.4852	.3747	.4013	.5779	.4034	.4203	.3933	.2598	.2317	.4278	.3789	.2335	.2607	.4721
GraphRNA	.7581	.6968	.6035	.8198	.8650	.8172	.6320	.8407	.7710	.6394	.9207	.8851	.1609	.1859
ARWMF	.7769	.5608	.6180	.8205	.7400	.8089	.2267	.4675	.2320	.2764	.6146	.8347	o.o.m.	o.o.m.
SAT	.7937	.8201	.4618	**.8579**	.8766	.7439	.6475	.8976	.7672	.6767	.9260	.8402	.3144	.5677
SVGA (proposed)	**.8493**	**.8806**	**.6227**	.8533	**.8854**	**.8808**	**.6757**	**.9209**	**.8293**	**.6879**	**.9264**	**.9037**	**.4394**	**.6644**

Q3. Observation of Labels

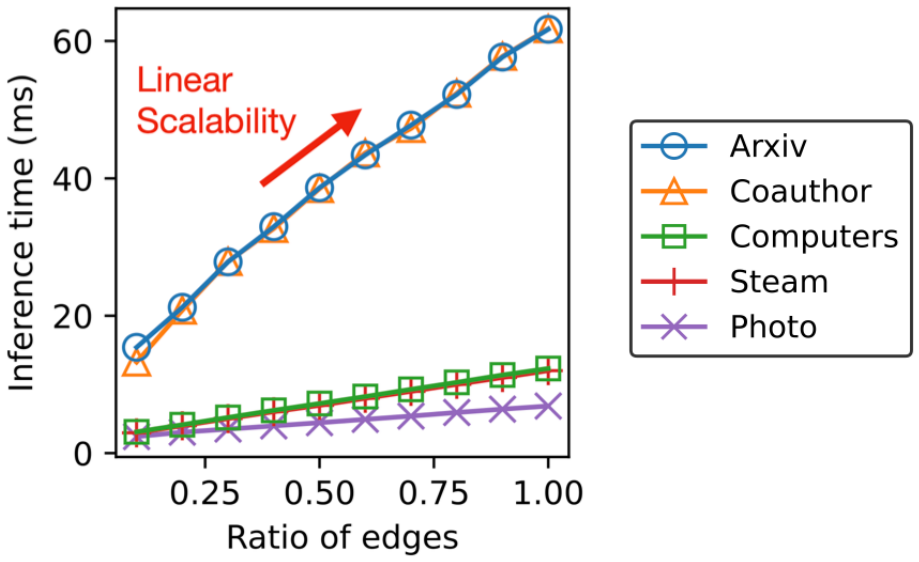
- **Q3.** Does observed labels help estimate accurate features?
- **A3.** Observed labels improve the accuracy of feature estimation
  - Implies that the dual estimation is effective for learning better $\mathbf{Z}$



(a) Cora **(b) Computers**

Q4. Scalability

- **Q4.** How does the computational time increase with graph size?
- **A4.** The computational time increases linearly with # of edges
  - The running time is instant even for large graphs with $> 1M$ edges



Outline

- Introduction
- Part 1: Probabilistic modeling of graphs
- Part 2: Proposed approach for feature estimation
- **<u>Conclusion</u>**

Conclusions (1/2)

- We propose **SVGA** for accurate node feature estimation
  - A GNN-based graph autoencoder for dual estimation
- The main ideas are summarized as follows:
  - **Idea 1:** GMRF prior of latent variables
  - **Idea 2:** Low-rank approximation of the covariance matrix
  - **Idea 3:** Unified and deterministic inference
- SVGA outperforms six competitors in eight real-world graphs
  - In estimation of binary and continuous features

Conclusions (2/2)

- I've studied probabilistic approaches for node-level tasks
  - **SBP** and **BPN** for differentiable probabilistic inference
  - **GRAB** and **SVGA** for the maximization of intractable likelihoods
- Why probabilistic approaches?
  1. To solve challenging problems with insufficient observations
  2. To improve and generalize GNNs in a new perspective
  3. To design graph algorithms with interpretable decision processes

Research Experience

- I love solving new, underexplored, and challenging problems
  - **Structural modification of graphs** [WSDM'20, WWW'22]
  - **Interpretable ML** [NeurIPS'19, ICDM'19, PAKDD'21, SDM'22]
  - **Multivariate time series analysis** [SDM'21, KDD'21]

Structural Modification

- Graph classification
- Data augmentation
- Subgraph sampling

Interpretable ML

- Zero-shot generation
- Ensemble distillation
- Probabilistic decisions

Multivariate Time Series

- Learning correlations
- Stock price prediction
- Transformer encoder

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***, J. Kim*, H. Yoon*, G. Kim, C. Jang, and U Kang, "Accurate Graph-Based PU Learning without Class Prior", **ICDM 2021** (*equal contribution)
4. **J. Yoo**, H. Jeon, J. Jung, and U Kang, "Accurate Node Feature Estimation with Structured Variational Graph Autoencoder", **KDD 2022**

Thank You!

Jaemin Yoo

Homepage: https://jaeminyoo.github.io

(for the papers, codes, and datasets)