Accurate Node Feature Estimation with Structured Variational Graph Autoencoder

Jaemin Yoo[1], **Hyunsik Jeon**[2], **Jinhong Jung**[3], and **U Kang**[2]

[1] Carnegie Mellon University

[2] Seoul National University

[3] Jeonbuk National University

KDD 2022

Outline

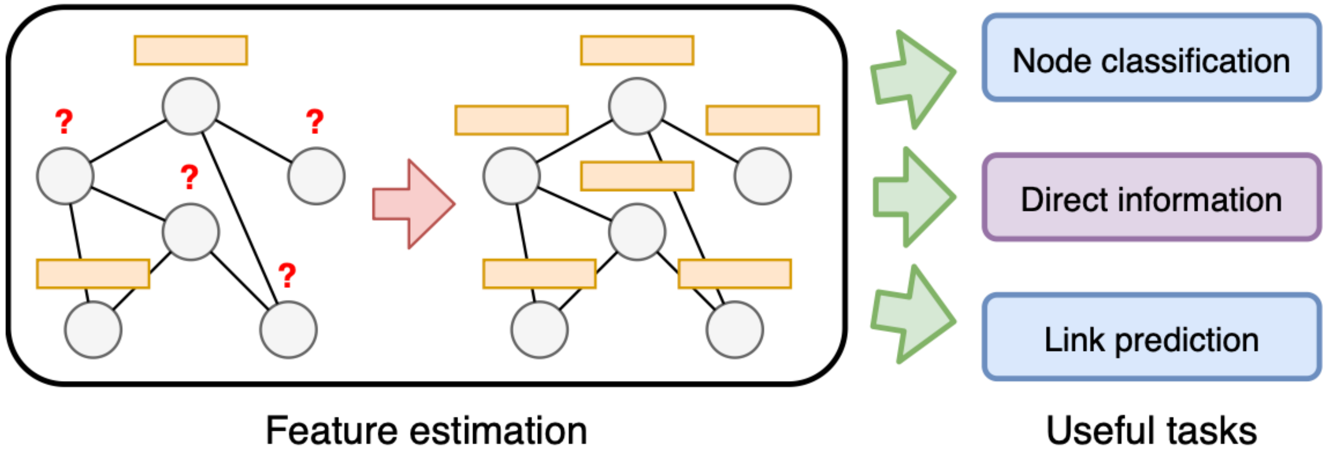
Node Features in Graphs

- Real-world graphs contain **node features**
  - Activity logs of users in a social network
  - Abstracts of papers in a citation network
- Many tasks on graphs require such features
  - Node classification, link prediction, etc.

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

Feature Estimation

- **Missing features** are common in real graphs
  - E.g., user nodes with private profiles
- **Feature estimation** is essential to utilize node features in large real graphs



Problem Definition

- **Given**
  - An undirected graph $G = (\mathcal{V}, \mathcal{E})$
  - Node feature $\mathbf{x}_i$ for some nodes in $\mathcal{V}_x \subset \mathcal{V}$
    - $\mathbf{x}_i$ can be either discrete or continuous vectors
  - (Optional) node labels y_i for nodes in $\mathcal{V}_y \subseteq \mathcal{V}$
    - Discrete labels are often easier to acquire than $\mathbf{X}$
    - They provide additional information to $\mathcal{V} \setminus \mathcal{V}_x$
- **Predict**
  - Unknown feature $\mathbf{x}_j$ for nodes in $\mathcal{V} \setminus \mathcal{V}_x$

Outline

- Introduction
- **Proposed Approach**
  - **<u>Motivation</u>**
  - Main Ideas
  - Model Architecture
- Experiments
- Conclusion

Dual Estimation

- We formulate the problem as maximizing

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

  - $\mathcal{F}$ is our estimator, and Θ is the parameters
- That is, we use $\mathbf{X}$ and $\mathbf{y}$ as the estimation targets, not as inputs
  - $\mathcal{F}$ aims to predict $\mathbf{X}$ and $\mathbf{y}$ from $\mathbf{A}$

Variational Inference

- **Q:** How can we maximize $p_\Theta(\mathbf{X}, \mathbf{y}|\mathbf{A})$?
- Run **variational inference** with latent var. $\mathbf{Z}$

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

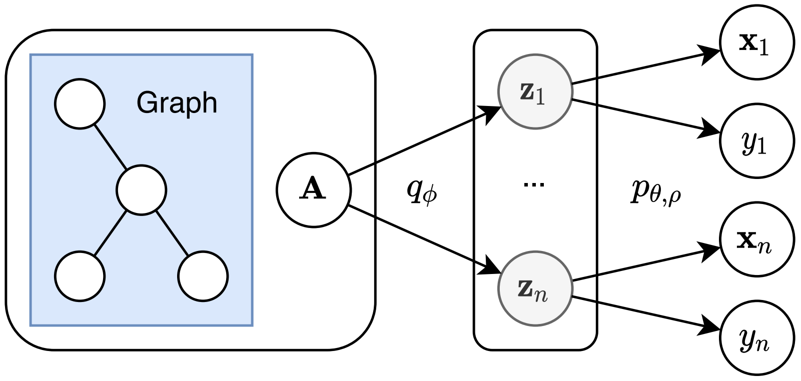
  - $\mathcal{L}(\Theta)$ is the **evidence lower bound (ELBO)** term
  - **Term 1** is the conditional likelihood of $\mathbf{X}$ and $\mathbf{y}$
  - **Term 2** is a regularizer on $q_\phi(\mathbf{Z})$ based on $p(\mathbf{Z}|\mathbf{A})$

Reconstruction Errors

- **Term 1** of ELBO is the reconstruction error:

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

- $\mathbf{Z}$ introduces **conditional independence**
  - Allows to separate the decoding of $\mathbf{x}_i$ and y_i



KL Divergence Regularizer

- Term 2 of ELBO regularizes the dist. of $\mathbf{Z}$:

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

- D_{KL} forces $q_\phi(\mathbf{Z})$ to be closer to $p(\mathbf{Z}|\mathbf{A})$
  - The effect of regularization is determined by how we choose the prior $p(\mathbf{Z}|\mathbf{A})$
  - Note that $p(\mathbf{Z}|\mathbf{A})$ is assumed with no parameters

Research Motivation

- Previous works ignore the correlations of $\mathbf{Z}$
  - By $q_\phi(\mathbf{Z}) = \mathcal{N}(\boldsymbol{\mu}, \mathrm{diag}(\boldsymbol{\sigma}))$ and $p(\mathbf{Z}) = \mathcal{N}(0, \mathbf{I}_n)$
- The **correlations** are essential in our case
  - Since the graph itself represents the correlations between target and observed nodes

Q1. *How can we consider the correlations of* $\mathbf{Z}$*?*

Q2. *How can we run efficient and stable inference?*

Outline

- Introduction
- **Proposed Approach**
  - Motivation
  - **Main Ideas**
  - Model Architecture
- Experiments
- Conclusion

Main Ideas

- Our main ideas for **structured inference**:
- **Idea 1:** GMRF-based prior of $\mathbf{Z}$
  - To utilize the graph in probabilistic modeling
- **Idea 2:** Low-rank approximation
  - To make tractable computation of the D_{KL} term
- **Idea 3:** Unified deterministic inference
  - To improve the stability and efficiency of inference

Idea 1: GMRF Prior (1/3)

- **Idea 1:** We model $p(\mathbf{Z}|\mathbf{A})$ as **Gaussian MRF**
  - To utilize the structure $\mathbf{A}$ in probabilistic modeling
- GMRF computes the **joint probability** as

$$p(\mathbf{z}) = \frac{1}{C} \prod_{i \in \mathcal{V}} \psi_i(z_i) \prod_{(i,j) \in \mathcal{E}} \psi_{ij}(z_i, z_j), \tag{3}$$

  - where ψ_i and ψ_{ij} are **node** and **edge potentials**

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

Idea 1: GMRF Prior (2/3)

- The **potential functions** are defined as

$$\psi_i(z_i) = \exp(-0.5K_{ii}z_i^2 + h_i z_i)$$
$$\psi_{ij}(z_i, z_j) = \exp(-K_{ij}z_i z_j),$$

  - We set $\mathbf{h}$ to zero for the zero-mean of $p(\mathbf{Z}|\mathbf{A})$
  - We set $\mathbf{K}$ to the **normalized graph Laplacian**:

$$\mathbf{K} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$$

Idea 1: GMRF Prior (3/3)

- The GMRF prior allows us to write D_{KL} as

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

  - which includes $\mathbf{K}$ as a structural regularizer
- When we parameterize $q_\phi(\mathbf{Z}) = \mathcal{N}(\mathbf{U}, \Sigma)$
  - $\mathbf{U} \in \mathbb{R}^{n \times d}$ and $\Sigma \in \mathbb{R}^{n \times n}$ are generated from f

Idea 2: Low-Rank Σ (1/2)

- **Q:** How can we efficiently compute $\log|\Sigma|$?
    - Naïve computation is $O(n^3)$ due to $\Sigma \in \mathbb{R}^{n\times n}$
- **Idea 2:** We apply **low-rank approximation**
    - We assume the low-rank structure of Σ as

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

    - $\beta > 0$ is a hyperparameter for the diagonal terms
    - $\mathbf{V} \in \mathbb{R}^{n\times r}$ is a new embedding matrix for Σ

Idea 2: Low-Rank Σ (2/2)

- We rewrite the log determinant as

$$\log|\Sigma| = \log|\mathbf{I}_r + \beta^{-1}\mathbf{V}^\top\mathbf{V}| + \log|\beta\mathbf{I}_n|,$$

  - where $\mathbf{I}_r \in \mathbb{R}^{r\times r}$ is the $r \times r$ identity matrix
  - Its complexity is $O(r^2 n + r^3)$, where $r \ll n$

Idea 3: Stable Inference (1/3)

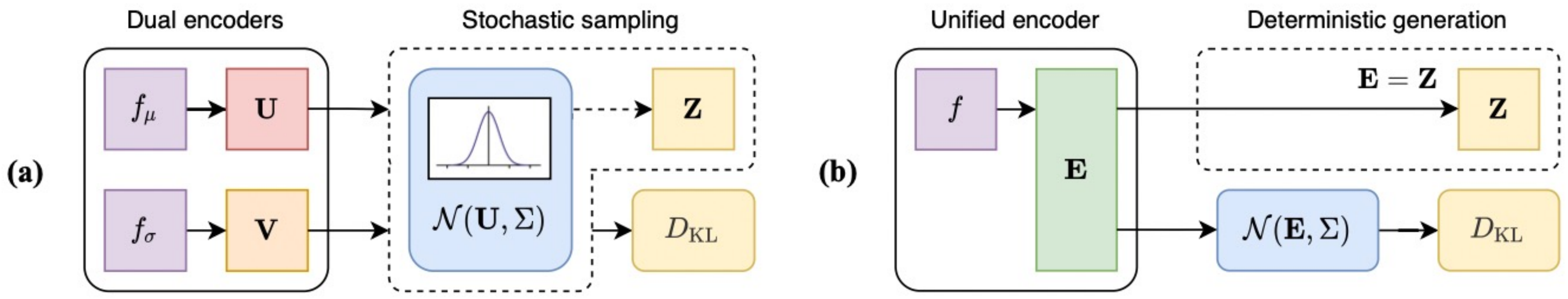
- **Idea 3:** We improve the stability of inference
  - By **1) unified** and **2) deterministic** modeling
- **Obs. 3-1:** $\mathbf{U}$ and $\mathbf{V}$ play similar roles in D_{KL}
  - $\mathbf{U}$ and $\mathbf{V}$ are used to model $q_\phi(\mathbf{Z}) = \mathcal{N}(\mathbf{U}, \Sigma)$
    - $\mathbf{U} \in \mathbb{R}^{n \times d}$ is for the mean
    - $\mathbf{V} \in \mathbb{R}^{n \times r}$ is for the covariance $\Sigma = \beta \mathbf{I}_n + \mathbf{V}\mathbf{V}^\top$
- **Idea 3-1:** To unify $\mathbf{U}$ and $\mathbf{V}$ as $\mathbf{E} = \mathbf{U} = \mathbf{V}$
  - In this way, we make one embedding matrix $\mathbf{E}$

Idea 3: Stable Inference (2/3)

- **Obs. 3-2:** Stochastic sampling is unstable
  - Previous works sample $\mathbf{Z}$ in a stochastic way
    - They sample $z_i \sim q_i(Z; \phi)$ independently for each i
  - Not effective if we consider the **correlations** of $\mathbf{Z}$
    - We need to sample $\mathbf{Z}$ simultaneously for all nodes
    - The space of sampling is **exponential** with # of nodes

Idea 3: Stable Inference (3/3)

- **Idea 3-2:** We generate deterministic $\mathbf{Z}$ from $\mathbf{E}$
  - This is equivalent to using $\mathbf{Z} = \mathrm{argmax}_{\mathbf{Z}'}\, q_\phi(\mathbf{Z}')$
- **Advantages**
  - We greatly improve the stability of inference
  - We can still utilize the D_{KL} regularizer on $q_\phi(\mathbf{Z})$



Summary of Main Ideas

- We propose **Idea 1** to model correlations
  - By modeling GMRF prior of latent variables
- We propose **Idea 2** and **3** to improve efficiency
  - Low-rank approx. and deterministic inference
- They result in our **objective function $l(\Theta)$**:

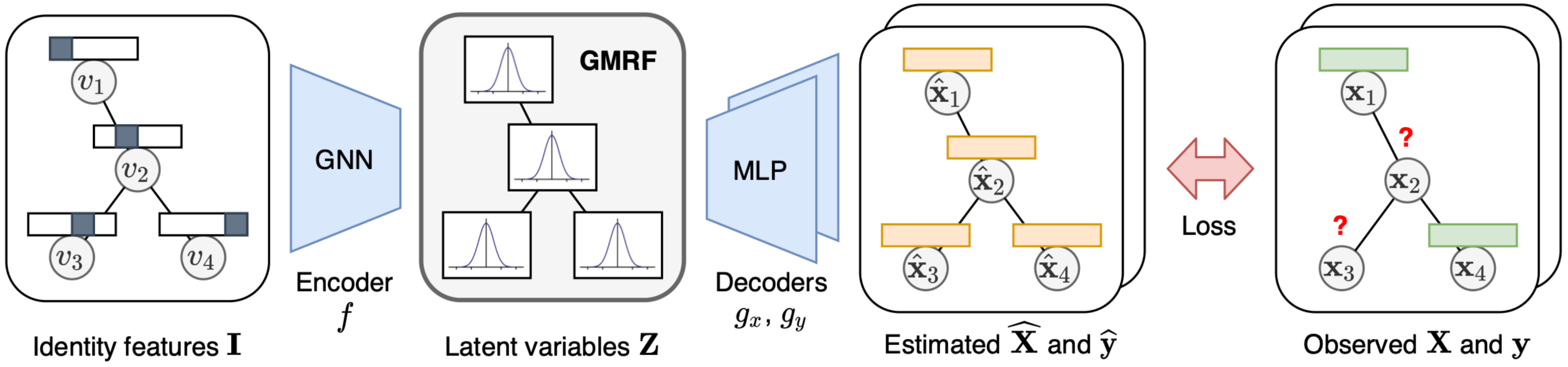
$$\underbrace{\sum_{i \in \mathcal{V}_x} l_x(\hat{\mathbf{x}}_i, \mathbf{x}_i)}_{\text{Error for } \mathbf{X}} + \underbrace{\sum_{i \in \mathcal{V}_y} l_y(\hat{y}_i, y_i)}_{\text{Error for } \mathbf{y}} + \underbrace{\lambda(\mathrm{tr}(\mathbf{Z}^\top \mathbf{K} \mathbf{Z}) - \alpha \log|\mathbf{I}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

Outline

- Introduction
- **Proposed Approach**
    - Motivation
    - Main Ideas
    - **<u>Model Architecture</u>**
- Experiments
- Conclusion

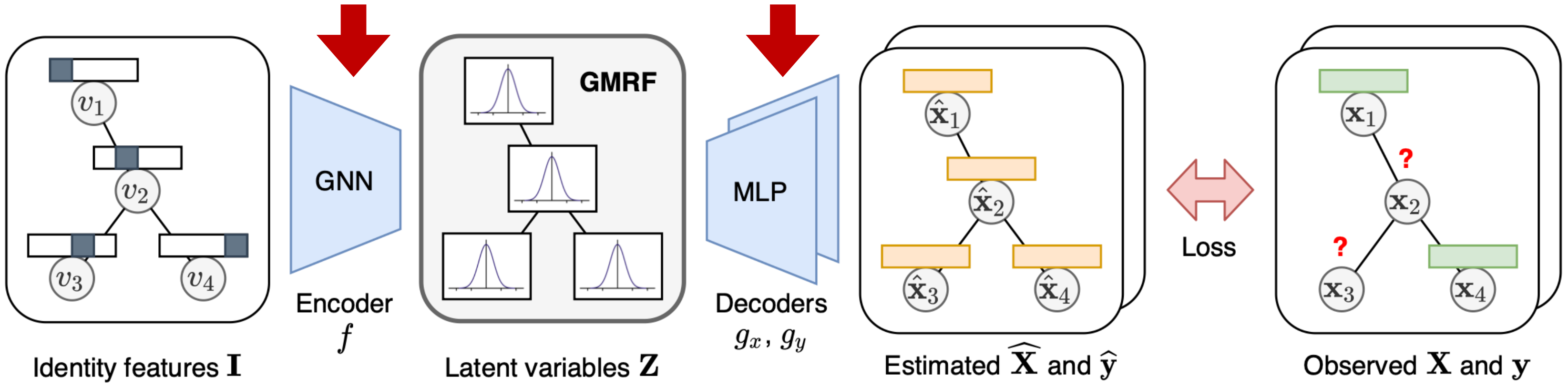
Proposed Architecture

- We propose **SVGA** for feature estimation
  - Structured Variational Graph Autoencoder
- GNN-based autoencoder for dual estimation
  - **GNN encoder** generates latent variables $\mathbf{Z}$
  - **MLP decoders** make estimations $\hat{\mathbf{X}}$ and $\hat{\mathbf{y}}$



Encoder and Decoders

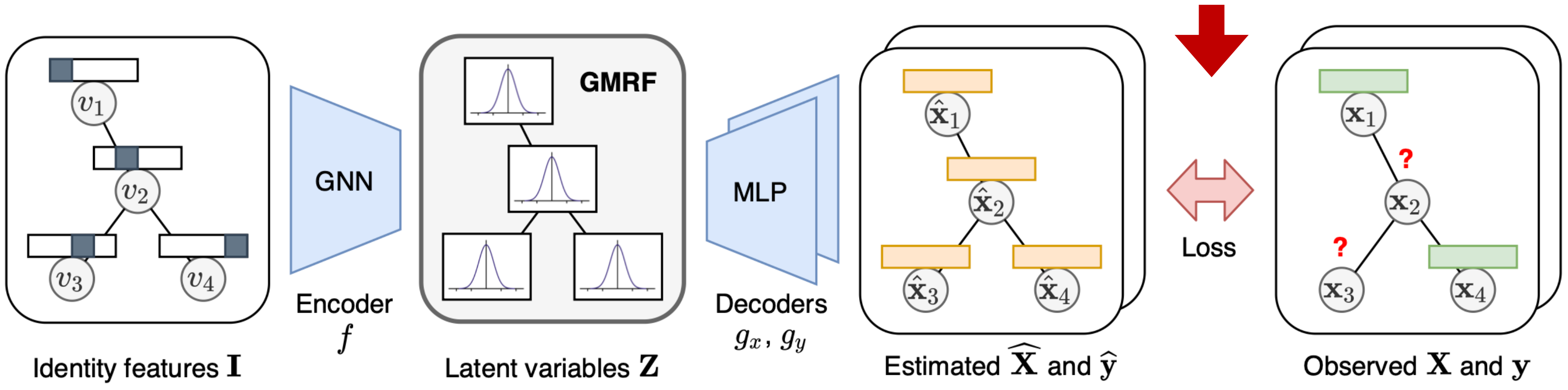
- **Graph convolutional network** as f
  - Make an identity matrix $\mathbf{I} \in \mathbb{R}^{n \times n}$ as an input
    - Allows f to learn independent embeddings for nodes
- **Multilayer perceptrons** as g_x and g_y
  - Estimate features and (optionally) labels, resp.



Objective Function

- We minimize our objective function $l(\Theta)$
  - l_x and l_y are **reconstruction errors** for $\mathbf{X}$ and $\mathbf{y}$
  - l_{GMRF} is our **proposed regularizer** for $\mathbf{Z}$

$$l(\Theta) = \sum_{i \in \mathcal{V}_x} l_x(\hat{\mathbf{x}}_i, \mathbf{x}_i) + \sum_{i \in \mathcal{V}_y} l_y(\hat{\mathbf{y}}_i, \mathbf{y}_i) + \lambda l_{\text{GMRF}}(\mathbf{Z}, \mathbf{A}),$$

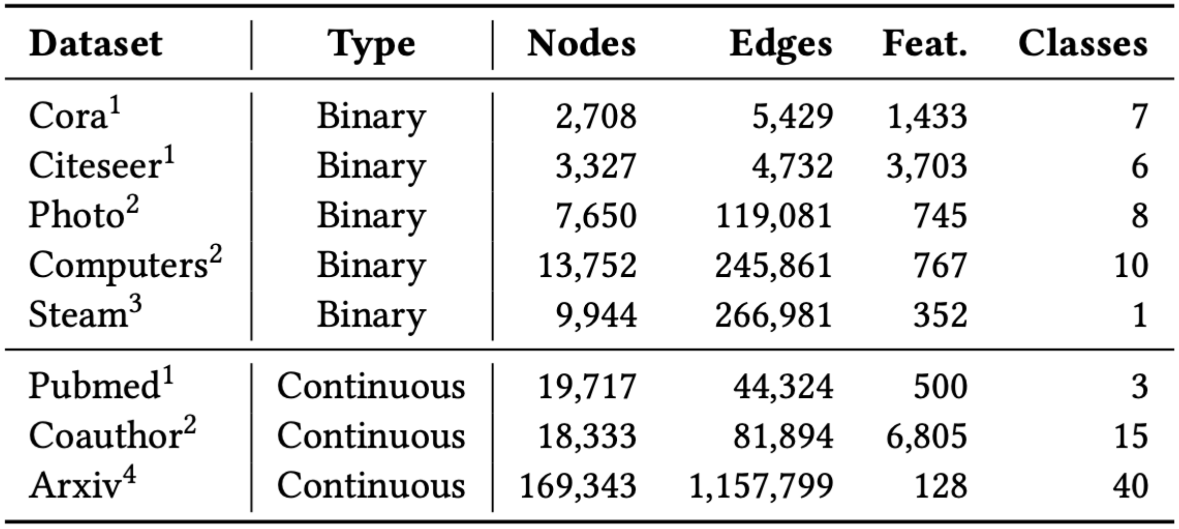


Outline

- Introduction
- Proposed Approach
  - Motivation
  - Main Ideas
  - Model Architecture
- **Experiments**
- Conclusion

Experimental Setup

- We compare SVGA with various models:
  - VAE, GCN, GAT, GraphRNA, ARWMF, SAT, etc.
- We use eight public graphs datasets

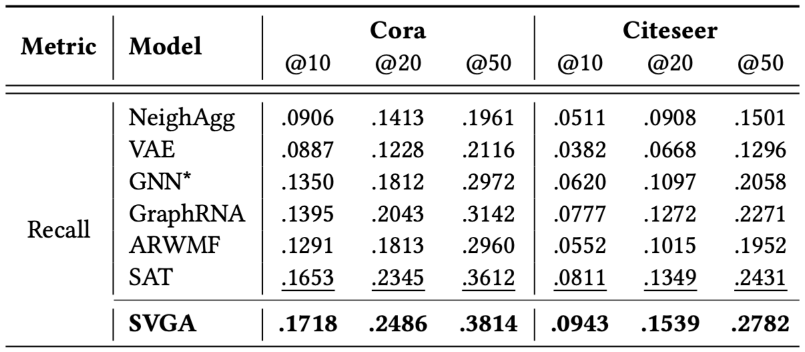


Dataset	Type	Nodes	Edges	Feat.	Classes
Cora[1]	Binary	2,708	5,429	1,433	7
Citeseer[1]	Binary	3,327	4,732	3,703	6
Photo[2]	Binary	7,650	119,081	745	8
Computers[2]	Binary	13,752	245,861	767	10
Steam[3]	Binary	9,944	266,981	352	1
Pubmed[1]	Continuous	19,717	44,324	500	3
Coauthor[2]	Continuous	18,333	81,894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

Experimental Results (1/4)

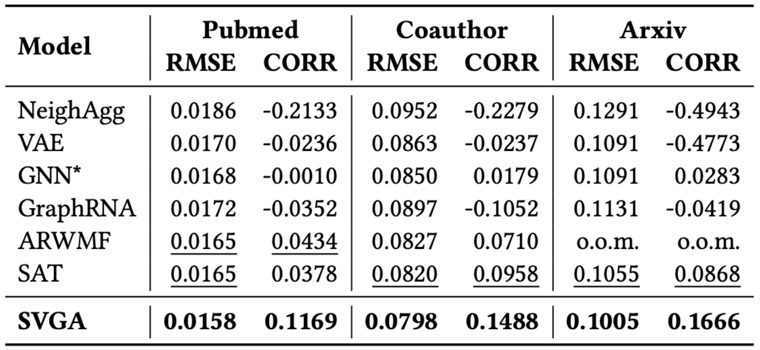
- **Feature estimation**
  - **Q1.** How accurate is SVGA in feature estimation?
  - **A1.** SVGA performs best in two types of features
    - Binary and continuous features
    - We use two evaluation metrics for each type

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**

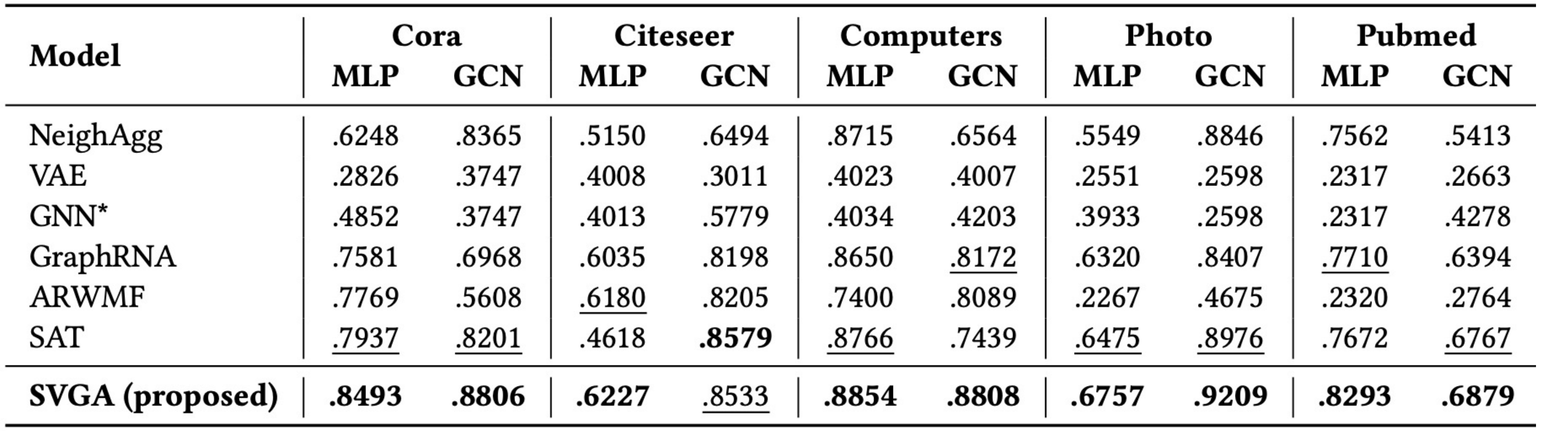
Continuous features



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**

Experimental Results (2/4)

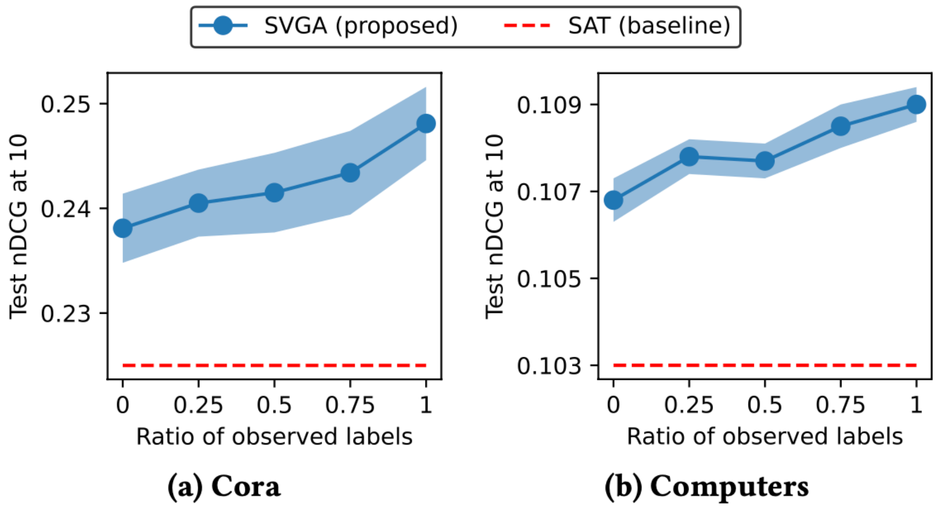
- **Node classification**
  - **Q2.** Does SVGA help node classification?
  - **A2.** SVGA works best with 2 different classifiers
    - We train a classifier based on generated features
    - SVGA outperforms baselines with both MLP and GCN



Model	**Cora**		**Citeseer**		**Computers**		**Photo**		**Pubmed**	
	MLP	**GCN**	**MLP**	**GCN**	**MLP**	**GCN**	**MLP**	**GCN**	**MLP**	**GCN**
NeighAgg	.6248	.8365	.5150	.6494	.8715	.6564	.5549	.8846	.7562	.5413
VAE	.2826	.3747	.4008	.3011	.4023	.4007	.2551	.2598	.2317	.2663
GNN*	.4852	.3747	.4013	.5779	.4034	.4203	.3933	.2598	.2317	.4278
GraphRNA	.7581	.6968	.6035	.8198	.8650	.8172	.6320	.8407	.7710	.6394
ARWMF	.7769	.5608	.6180	.8205	.7400	.8089	.2267	.4675	.2320	.2764
SAT	.7937	.8201	.4618	**.8579**	.8766	.7439	.6475	.8976	.7672	.6767
SVGA (proposed)	**.8493**	**.8806**	**.6227**	.8533	**.8854**	**.8808**	**.6757**	**.9209**	**.8293**	**.6879**

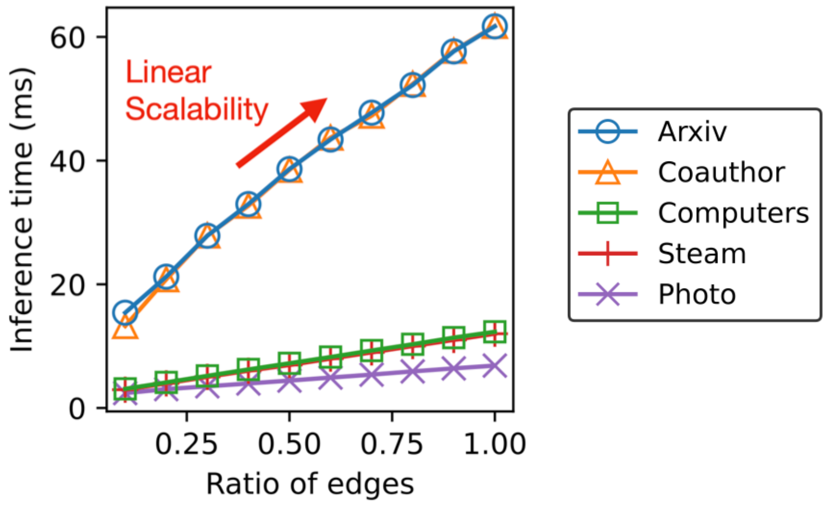
Experimental Results (3/4)

- **Observation of labels**
  - **Q3.** Do observed labels help feature estimation?
  - **A3.** They improve the accuracy of estimation
    - The dual estimation is effective for learning better $\mathbf{z}$



Experimental Results (4/4)

- **Scalability**
  - **Q4.** How does running time scale with graph size?
  - **A4.** It increases linearly with # of edges
    - The running time is instant even for large graphs



Outline

- Introduction
- Proposed Approach
  - Motivation
  - Main Ideas
  - Model Architecture
- Experiments
- **Conclusion**

Conclusion

- We propose **SVGA** for feature estimation
- The main ideas are summarized as follows:
  - **Idea 1:** GMRF prior of latent variables
  - **Idea 2:** Low-rank approximation of the covariance
  - **Idea 3:** Unified and deterministic inference
- We achieve SOTA accuracy in 8 real graphs
  - In estimation of binary and continuous features

Thank You!

Jaemin Yoo

Homepage: https://jaeminyoo.github.io

GitHub: https://github.com/snudatalab/SVGA