Accurate Node Feature Estimation with Structured Variational Graph Autoencoder

Jaemin Yoo\textsuperscript{1}, Hyunsik Jeon\textsuperscript{2}, Jinhong Jung\textsuperscript{3}, and U Kang\textsuperscript{2}

\textsuperscript{1} Carnegie Mellon University
\textsuperscript{2} Seoul National University
\textsuperscript{3} Jeonbuk National University

KDD 2022
Outline

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

![Diagram showing feature estimation process and useful tasks like node classification, direct information, and link prediction.](diagram.png)
Problem Definition

- **Given**
  - An undirected graph $G = (\mathcal{V}, \mathcal{E})$
  - Node feature $x_i$ for some nodes in $\mathcal{V}_x \subseteq \mathcal{V}$
    - $x_i$ 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 $X$
    - They provide additional information to $\mathcal{V} \setminus \mathcal{V}_x$

- **Predict**
  - Unknown feature $x_j$ for nodes in $\mathcal{V} \setminus \mathcal{V}_x$
Outline

• Introduction

• Proposed Approach
  • Motivation
  • Main Ideas
  • Model Architecture

• Experiments

• Conclusion
Dual Estimation

• We formulate the problem as maximizing

\[ p_\Theta(X, y|A) \quad \text{with} \quad \hat{X}, \hat{y} = F(A; \Theta) \]

• \(F\) is our estimator, and \(\Theta\) is the parameters
• That is, we use \(X\) and \(y\) as the estimation targets, not as inputs
  • \(F\) aims to predict \(X\) and \(y\) from \(A\)
Variational Inference

• **Q:** How can we maximize $p_\Theta(X, y|A)$?

• Run *variational inference* with latent var. $Z$

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

• $\mathcal{L}(\Theta)$ is the **evidence lower bound (ELBO)** term

• **Term 1** is the conditional likelihood of $X$ and $y$

• **Term 2** is a regularizer on $q_\phi(Z)$ based on $p(Z|A)$
Reconstruction Errors

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

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

• \( Z \) introduces **conditional independence**
  - Allows to separate the decoding of \( x_i \) and \( y_i \)
KL Divergence Regularizer

- **Term 2** of ELBO regularizes the dist. of $Z$:

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

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

• Previous works ignore the correlations of $\mathbf{Z}$
  • By $q_\phi(\mathbf{Z}) = \mathcal{N}(\mathbf{u}, \text{diag}(\mathbf{\sigma}))$ 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 $Z$
    • 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 \( p(Z|A) \) as **Gaussian MRF**
  - To utilize the structure \( A \) in probabilistic modeling
- GMRF computes the **joint probability** as

\[
p(z) = \frac{1}{C} \prod_{i \in V} \psi_i(z_i) \prod_{(i,j) \in E} \psi_{ij}(z_i, z_j),
\]

- where \( \psi_i \) and \( \psi_{ij} \) are **node** and **edge potentials**
  \( \Rightarrow \) Higher potentials make a higher probability \( p(z) \)
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 \( h \) to zero for the zero-mean of \( p(Z|A) \)
• We set \( K \) to the normalized graph Laplacian:

\[ K = I - D^{-1/2} A D^{-1/2} \]
Idea 1: GMRF Prior (3/3)

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

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

$$= 0.5(\text{tr}(U^T K U) + d(\text{tr}(K \Sigma) - \log |\Sigma|)) + C,$$

• which includes $K$ as a structural regularizer
• When we parameterize $q_\phi(Z) = \mathcal{N}(U, \Sigma)$
  • $U \in \mathbb{R}^{n \times d}$ and $\Sigma \in \mathbb{R}^{n \times n}$ are generated from $f$
Idea 2: Low-Rank $\Sigma$ (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 $\Sigma$ as

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

- $\beta > 0$ is a hyperparameter for the diagonal terms
- $\mathbf{V} \in \mathbb{R}^{n \times r}$ is a new embedding matrix for $\Sigma$
Idea 2: Low-Rank $\Sigma$ (2/2)

• We rewrite the log determinant as

$$\log |\Sigma| = \log |I_r + \beta^{-1}V^TV| + \log |\beta I_n|,$$

• where $I_r \in \mathbb{R}^{r \times r}$ is the $r \times r$ identity matrix
• Its complexity is $O(r^2n + r^3)$, 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:** $U$ and $V$ play similar roles in $D_{KL}$
  - $U$ and $V$ are used to model $q_\phi(Z) = \mathcal{N}(U, \Sigma)$
    - $U \in \mathbb{R}^{n \times d}$ is for the mean
    - $V \in \mathbb{R}^{n \times r}$ is for the covariance $\Sigma = \beta I_n + VV^T$

- **Idea 3-1:** To unify $U$ and $V$ as $E = U = V$
  - In this way, we make one embedding matrix $E$
Idea 3: Stable Inference (2/3)

• **Obs. 3-2:** Stochastic sampling is unstable
  • Previous works sample $Z$ 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 $Z$
    • We need to sample $Z$ 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 $Z$ from $E$
  - This is equivalent to using $Z = \arg\max_Z q_\phi(Z')$

- **Advantages**
  - We greatly improve the stability of inference
  - We can still utilize the $D_{KL}$ regularizer on $q_\phi(Z)$
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** $l(\Theta)$:

$$
\sum_{i \in \mathcal{V}_x} l_x(\hat{x}_i, x_i) + \sum_{i \in \mathcal{V}_y} l_y(\hat{y}_i, y_i) + \lambda (\text{tr}(Z^T K Z) - \alpha \log |I + \beta^{-1}Z^T Z|)
$$

- Error for $X$
- Error for $y$
- Proposed regularizer $l_{\text{GMRF}}$
Outline

• Introduction

• Proposed Approach
  • Motivation
  • Main Ideas
  • Model Architecture

• 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 $\hat{y}$
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 $X$ and $y$
  - $l_{\text{GMRF}}$ is our proposed regularizer for $Z$

$$l(\Theta) = \sum_{i \in V_x} l_x(\hat{x}_i, x_i) + \sum_{i \in V_y} l_y(\hat{y}_i, y_i) + \lambda l_{\text{GMRF}}(Z, A),$$
Outline

• Introduction
• Proposed Approach
  • Motivation
  • Main Ideas
  • Model Architecture
• Experiments
• Conclusion
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       |
| Citeeseer\(^1\) | 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 MLP | Cora GCN | Citeseer MLP | Citeseer GCN | Computers MLP | Computers GCN | Photo MLP | Photo GCN | Pubmed MLP | Pubmed 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 $Z$
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