Neighbor-Anchoring Adversarial Graph Neural Networks

Zemin Liu, Yuan Fang, Yong Liu, Vincent W. Zheng

SMU Singapore Management University · Nanyang Technological University Singapore · WeBank 微众银行

Motivation

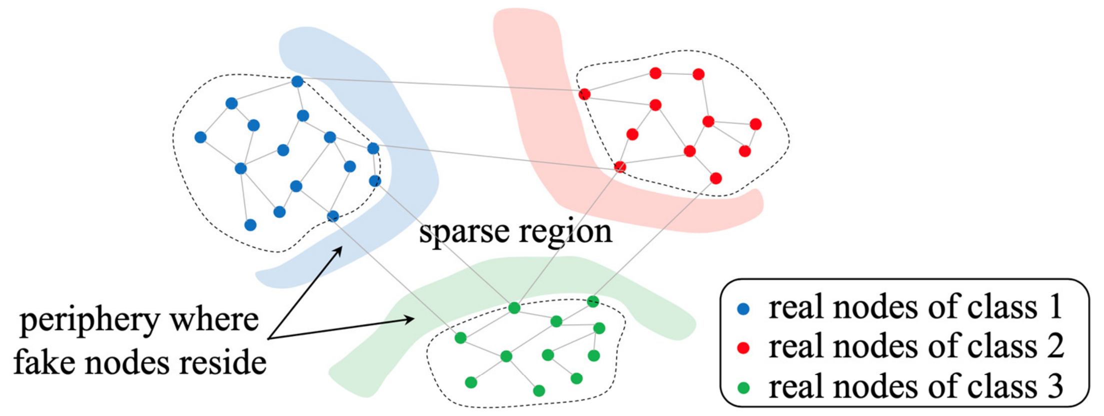

Fig. 1: Motivation of GANs on graph. Ideally, our generator would produce samples not only in the sparse regions with a low-density of links, but also in the periphery of the dense regions. The fake samples are complementary to enhance the robustness of node representations, enabling the discriminator to learn better decision boundaries.

Related work

- Graph neural networks
  - Neighborhood aggregation
- Generative adversarial networks
  - Generator vs. discriminator
- Prior studies seldom explore GANs and GNNs jointly in an end-to-end manner.

Challenges

- What is the definition of a sample on a graph?
- How do we produce good samples?

The proposed model: NAGNN

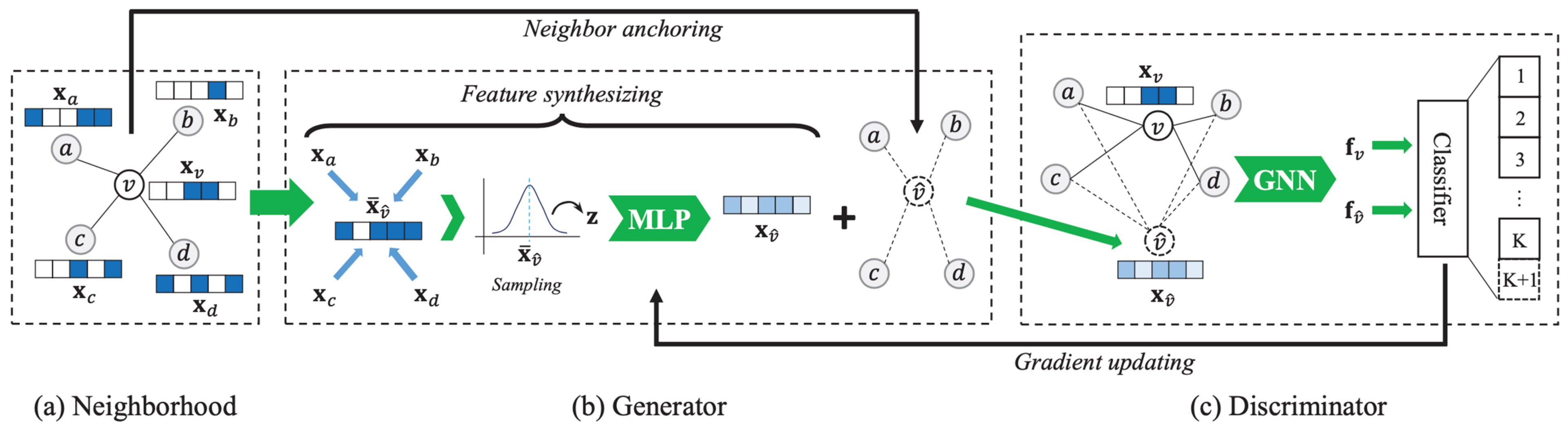

Fig. 2: Overall framework of NAGNN. (a) An existing node v and its neighboring nodes in the graph. (b) The generator, which produces a fake sample $\hat{v}$ anchored on the neighbors of the real node v. (c) The discriminator, which utilizes a GNN to classify real nodes into the first K classes, and fake samples into class $K+1$.

Algorithm 1 Model training for NAGNN

Input: graph $\mathcal{G}$, labeled set $\mathcal{L}$, number of epochs n_D for the discriminator and n_G for the generator in each iteration, number of fake samples m_D for the discriminator and m_G for the generator.
Output: θ_D, θ_G.

```
initialize parameters θ^D, θ^G;
while not converged do
    for i = 1 to n_D do                                   ▷ train discriminator
        V̂ ← generate m_D fake nodes for each labeled node
        update θ_D with L and V̂ according to Equation (4)
    end for
    for i = 1 to n_G do                                   ▷ train generator
        generate m_G fake samples w.r.t. each labeled node
        evaluate the fake samples using the discriminator
        update θ_G according to Equation (7)
    end for
end while
return θ_D, θ_G.
```

Discriminator

Graph convolution

$$\mathbf{f}_v^{(l)} = \mathrm{ReLU}\left(\frac{1}{|\tilde{\mathcal{N}}_v|}\sum_{v'\in\tilde{\mathcal{N}}_v}\mathbf{W}^{(l)}\mathbf{f}_{v'}^{(l-1)}\right)$$

Neighborhood aggregation

Loss function

$$D(y|v;\theta_D) = \frac{\exp(\mathbf{W}_y\mathbf{f}_v)}{\sum_{y'=1}^{K+1}\exp(\mathbf{W}_{y'}\mathbf{f}_v)}$$

Classification under the $K+1$ class setting

$$-\frac{1}{|\mathcal{L}|}\sum_{(v,y)\in\mathcal{L}}\log D(y|v;\theta_D) - \alpha\cdot\frac{1}{|\hat{\mathcal{V}}|}\sum_{\hat{v}\in\hat{\mathcal{V}}}\log D(K+1|\hat{v};\theta_D) + \lambda_D\|\theta_D\|_2^2. \quad (4)$$

Real nodes → classes $\{1,2,\ldots,K\}$

Fake nodes → class $K+1$

Generator

Neighbor anchoring for $\hat{v}$

- feature vector: synthesized by a neural network
- Neighborhood: anchored on v's neighborhood

$$\mathcal{N}_{\hat{v}} = \mathcal{N}_v$$

Feature synthesizing

$$Z \triangleq \mathrm{Gaussian}(\bar{\mathbf{x}}_{\hat{v}}, \sigma^2\mathbf{I})$$

Feature synthesizing with a multivariate Gaussian distribution

Mean feature vector

$$\bar{\mathbf{x}}_{\hat{v}} = \frac{1}{|\mathcal{N}_{\hat{v}}|}\sum_{v'\in\mathcal{N}_{\hat{v}}}\mathbf{x}_{v'}$$

Loss function

$$-\frac{1}{|\mathcal{L}|}\sum_{(v,y)\in\mathcal{L},\mathbf{z}\sim Z}\log D(y|G(v,\mathbf{z};\theta_G);\theta_D) + \lambda_G\|\theta_G\|_2^2. \quad (7)$$

Fool the D by classifying $\hat{v}$ into the same class of v

Experiments

Datasets

TABLE 1: Summary of datasets.

Datasets	# Nodes	# Edges	# Classes	# Features
Cora	2,708	5,429	7	1,433
Citeseer	3,327	4,732	6	3,703
Pubmed	19,717	44,338	3	500
DBLP	1,866	7,153	4	1,084

Baselines

Network embedding models
- DeepWalk

Unsupervised GAN-based models
- GraphGAN
- ARGA and ARVGA

Semi-supervised GAN-based models
- ARGA(S), ARVGA(S), Graph$\mathcal{S}$GAN

End-to-end graph neural networks
- GCN and GAT

Main results and model analysis

TABLE 2: Node classification performance (in percent) with standard deviation using 20 labeled nodes per class, averaged over 10 runs. The best results are **bolded**. In the column of "Input data", **A** denotes the adjacency matrix, **X** denotes the feature matrix, and $\mathcal{L}$ denotes the labeled nodes.

Methods	Input data	Cora Accuracy	Cora Micro-F	Cora Macro-F	Citeseer Accuracy	Citeseer Micro-F	Citeseer Macro-F	Pubmed Accuracy	Pubmed Micro-F	Pubmed Macro-F	DBLP Accuracy	DBLP Micro-F	DBLP Macro-F
DeepWalk	**A**	73.8±0.3	74.9±0.1	74.0±0.1	61.6±0.2	60.5±1.0	59.8±0.5	67.4±0.3	65.2±0.1	66.1±0.1	50.4±1.0	51.8±0.8	49.1±1.1
GraphGAN	**A**	58.8±0.2	57.9±0.1	57.2±0.1	60.4±1.4	58.5±0.1	58.6±0.1	73.2±0.1	75.3±0.1	73.2±0.1	52.4±2.5	51.1±3.5	52.1±4.3
ARGA	**A**, **X**	58.2±0.5	48.8±0.8	39.7±0.7	48.7±1.3	47.4±2.1	44.5±2.3	53.8±1.3	46.5±2.7	41.4±3.5	56.4±1.3	55.1±1.1	55.2±1.2
ARVGA	**A**, **X**	46.4±1.9	35.6±2.3	26.8±1.4	64.4±0.3	64.0±0.2	56.0±0.3	40.7±0.2	20.1±0.4	19.3±0.3	25.0±0.2	17.1±1.2	23.7±0.5
ARGA(S)	**A**, **X**, $\mathcal{L}$	72.1±0.7	68.7±0.6	56.4±0.9	61.8±1.2	59.9±1.8	57.2±1.4	62.6±1.6	55.3±2.3	50.8±2.4	59.3±1.8	57.8±1.5	58.0±1.4
ARVGA(S)	**A**, **X**, $\mathcal{L}$	63.7±1.5	62.5±1.7	50.6±1.4	68.9±0.5	68.1±0.8	61.4±0.6	50.7±1.4	46.0±0.8	39.3±0.9	41.7±1.1	43.9±1.3	42.3±1.4
Graph$\mathcal{S}$GAN	**A**, **X**, $\mathcal{L}$	79.2±0.6	79.3±0.5	78.0±0.6	67.4±0.7	65.8±0.4	61.8±0.5	68.2±0.4	68.7±0.5	67.5±0.5	58.6±0.9	57.4±0.8	56.8±0.9
GCN	**A**, **X**, $\mathcal{L}$	81.5±0.7	80.8±0.5	80.4±0.6	70.4±0.5	68.3±0.7	66.9±0.4	78.9±0.3	78.8±0.4	78.0±0.3	61.7±1.5	62.2±1.2	60.9±0.7
NAGCN	**A**, **X**, $\mathcal{L}$	**83.2**±0.6	**81.7**±0.4	**81.9**±0.5	**72.8**±0.4	**70.7**±0.5	**69.0**±0.4	**79.0**±0.3	**79.4**±0.2	**78.4**±0.3	**66.4**±0.7	**65.4**±0.9	**64.6**±0.9
GAT	**A**, **X**, $\mathcal{L}$	82.9±0.6	82.0±0.6	81.8±0.6	72.4±0.7	70.4±0.8	68.2±0.7	77.2±0.5	77.7±0.7	76.6±0.5	68.6±3.1	64.1±4.3	57.2±7.2
NAGAT	**A**, **X**, $\mathcal{L}$	**83.5**±0.4	**82.6**±0.3	**82.5**±0.2	**72.9**±0.4	**70.9**±0.5	**68.3**±0.8	**77.7**±0.4	**77.8**±0.1	**77.0**±0.3	**71.8**±1.7	**69.1**±1.4	**68.6**±1.3

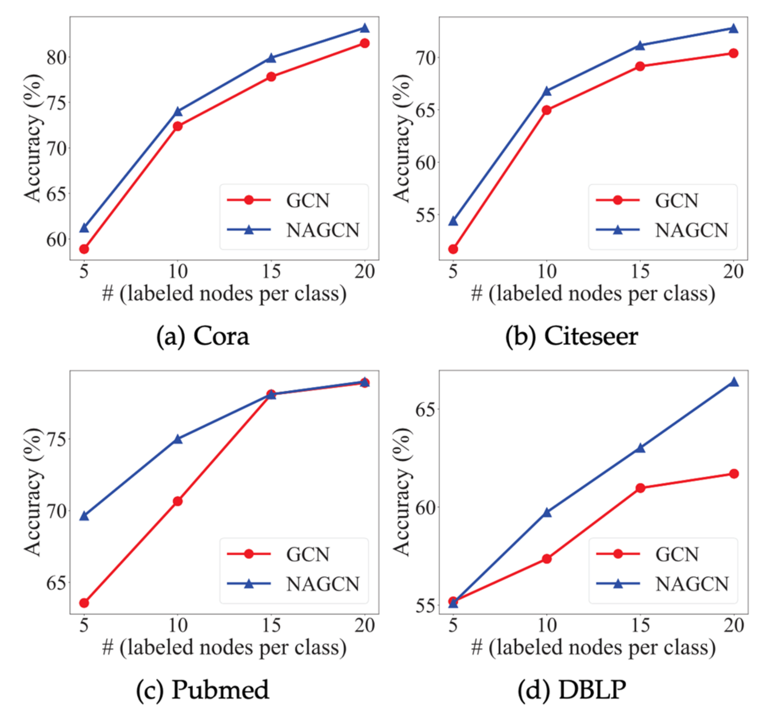

Fig. 3: Performance with fewer labeled nodes.

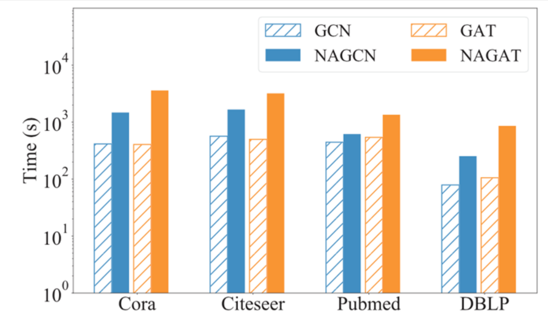

Fig. 4: Training time comparison.

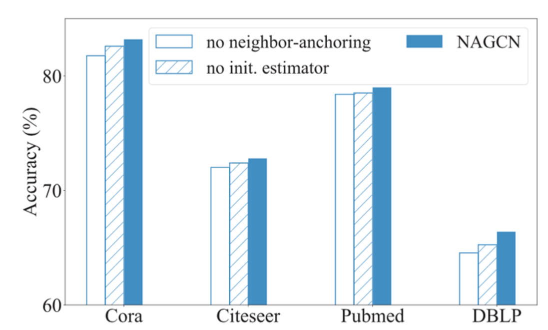

Fig. 5: Impact of our neighbor-anchoring strategy.

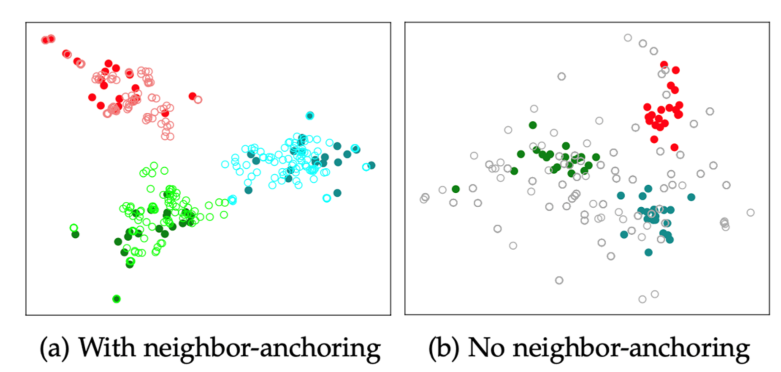

Fig. 6: Visualization of generated samples and real nodes. The solid dots denote real nodes, and different colors indicate different classes; the hollow circles denote fake samples. In (a), the fake samples are drawn in the same color as their reference nodes; in (b), the fake samples are not anchored on any real nodes and are all drawn in grey.

Conclusions

- **Problem**
  - Adversarial learning with graph neural networks
- **Challenges**
  - What is the definition of a sample on a graph?
  - How do we produce good samples?
- **Proposed model: NAGNN**
  - Generator
    - Neighbor-anchoring strategy: produce fake samples
  - Discriminator
    - Perform recursive neighborhood aggregation on the fake samples

Acknowledgments

Dr. Yuan Fang acknowledges the Lee Kong Chian Fellowship awarded by Singapore Management University for the support of this poster presentation.