Deep Iterative and Adaptive Learning for Graph Neural Networks

Presenter: Yu Chen

Department of Computer Science
Rensselaer Polytechnic Institute, Troy, NY 12180

Feb 8, 2020

Joint work with Dr. Lingfei Wu and Dr. Mohammed J. Zaki

Motivation

GNNs are powerful only if the graph-structured data is available.

- Is the initial graph-structure optimal for the downstream task?
  - error-prone data measurement or collection
    - noisy graphs, incomplete graphs, etc.
  - graph construction process might not be ideal for the downstream task
- Can we learn graph structure supplementary to the initial graph?

What if the graph-structured input data is NOT available.

- Manual graph construction
  - requires a lot of domain expertise
  - might introduce inevitable noise
- Can we do automatic graph construction?

Problem Definition

Input: a set of n objects associated with a feature matrix $\mathbf{X} \in \mathbb{R}^{d \times n}$ and an (optional and potentially noisy) initial graph topology.

Output: an adjacency matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ and node embeddings with respect to some (semi-)supervised downstream task.

Contributions

- We propose a novel iterative deep graph learning framework for jointly learning the graph structure and graph embedding. It dynamically stops when the learned graph structure approaches the optimized graph.

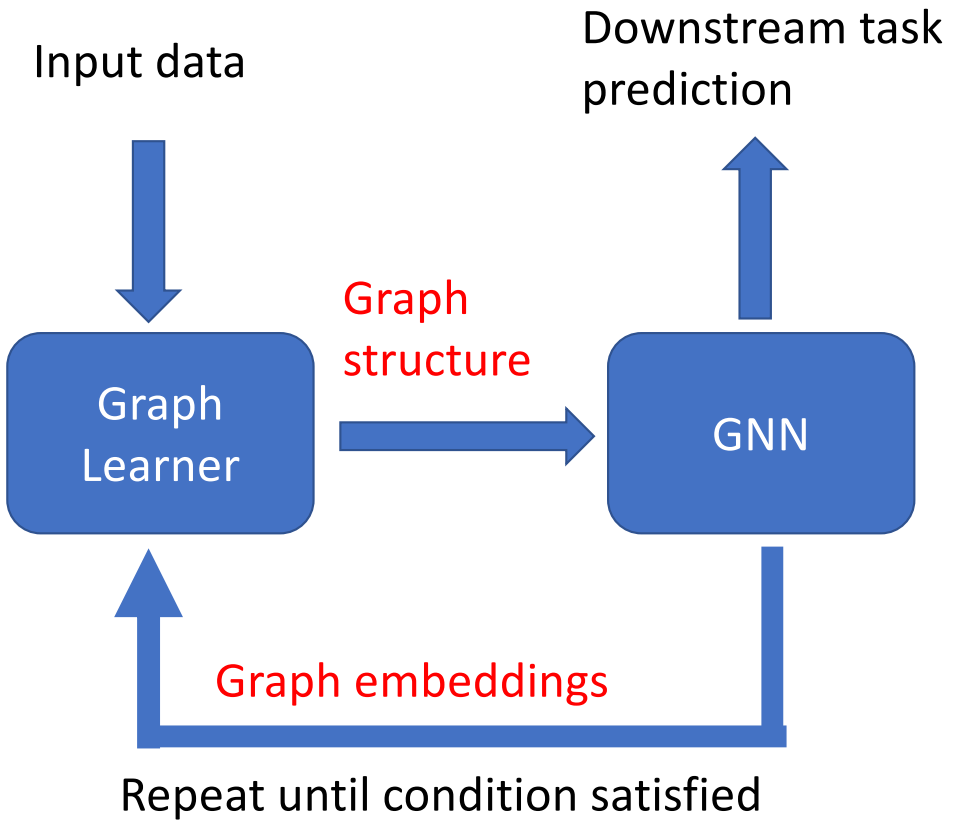
- We cast the graph structure learning problem as a graph similarity metric learning problem and leverage adaptive graph regularization for controlling smoothness, connectivity and sparsity of the learned graph.

- Besides its good performance on downstream tasks, it can be more robust to adversarial graph examples and can cope with both transductive and inductive learning.

Graph Learning and Graph Embedding: A Unified Perspective

- Jointly learn the graph structure and the GNN parameters.
- Better node embeddings -> better graph structure
- Better graph structure -> better node embeddings.
- Dynamically stop the iterative learning procedure.

A sketch of the proposed framework



Graph Learning as Similarity Metric Learning

- Graph similarity metric learning.
- Graph sparsification via ε-neighborhood.
- Obtaining a symmetric sparse non-negative adjacency matrix.

Multi-head weighted cosine similarity

$$s_{ij}^k = \cos\left(\mathbf{w}_k \odot \mathbf{v}_i, \mathbf{w}_k \odot \mathbf{v}_j\right)$$

Similarity for k-th perspective

m independent weight vectors

$$s_{ij} = \frac{1}{m}\sum_{k=1}^{m} s_{ij}^k \qquad \mathbf{A}_{ij} = \begin{cases} s_{ij} & s_{ij} > \varepsilon \\ 0 & \text{otherwise} \end{cases}$$

Taking average of m similarity scores

Symmetric sparse non-negative adjacency matrix

Incorporating the Initial Graph Structure

- Assumption: optimized graph structure is potentially a shift from the initial graph structure.
- We combine the learned graph with the initial graph.
- If such an initial graph is not available, we instead use a kNN graph.

Learning a shift from the initial graph structure

Initial graph structure

Learned graph structure

$$\tilde{\mathbf{A}} = \lambda \mathbf{L}_0 + (1-\lambda)\frac{\mathbf{A}_{ij}}{\sum_j \mathbf{A}_{ij}}$$

Normalized adjacency matrix

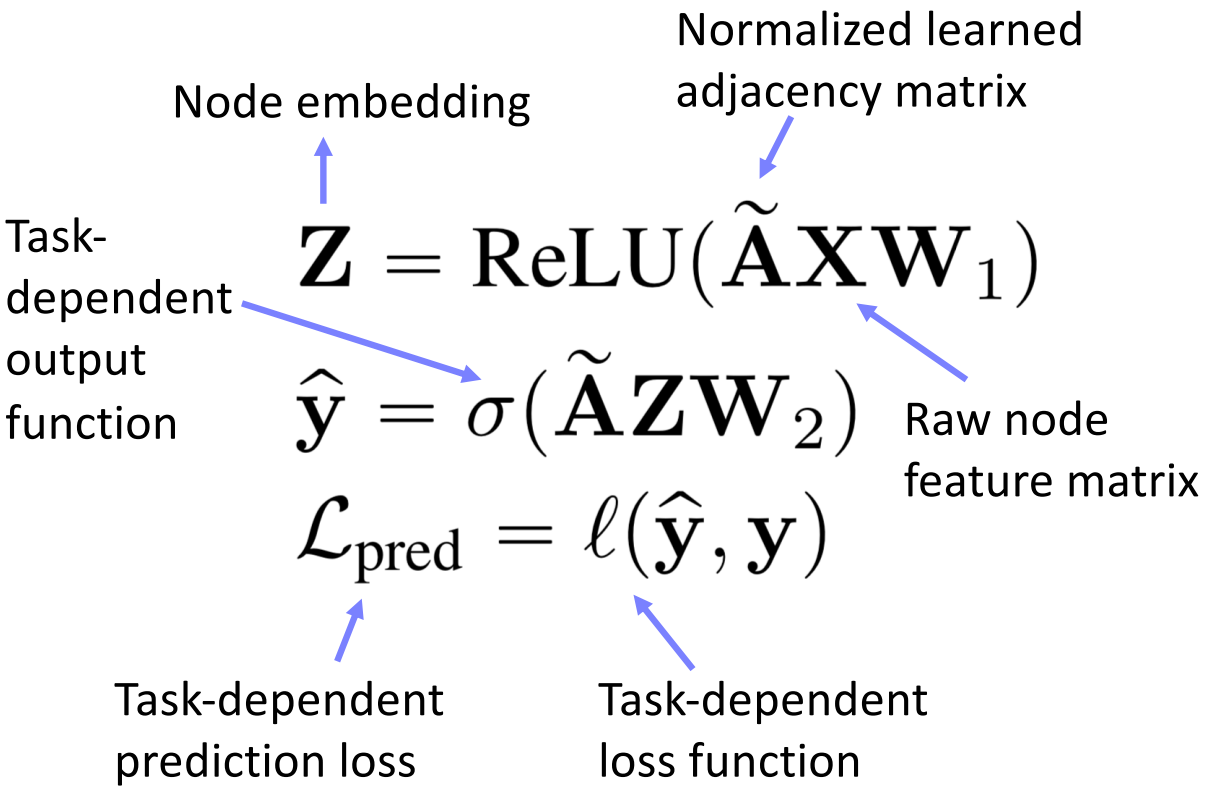
$$\mathbf{L}_0 = \mathbf{D}_0^{-1/2}\mathbf{A}_0\mathbf{D}_0^{-1/2}$$

$\mathbf{D}_0$ is the degree matrix.

Graph Node Embeddings and Prediction

- Our graph learning framework is agnostic to various GNNs.
- We adopt two-layered GCN in this work.
  - First layer: node feature -> node embedding.
  - Second layer: node embedding -> output space.

Task-dependent prediction



Graph Regularization

- Control the smoothness, connectivity and sparsity of the resulting learned graph.
- A is the adjacency matrix obtained through graph similarity metric learning.

Smoothness

$$\Omega(\mathbf{A}, \mathbf{X}) = \frac{1}{2n^2} \sum_{i,j} \mathbf{A}_{ij} ||\mathbf{x}_i - \mathbf{x}_j||^2 = \frac{1}{n^2} \text{tr}(\mathbf{X}^T \mathbf{L} \mathbf{X})$$

$$f(\mathbf{A}) = \frac{-\beta}{n} \mathbf{1}^T \log(\mathbf{A}\mathbf{1}) + \frac{\gamma}{n^2} ||\mathbf{A}||_F^2$$

Connectivity

Sparsity

$$\mathcal{L}_{\mathcal{G}} = \alpha \Omega(\mathbf{A}, \mathbf{X}) + f(\mathbf{A})$$

Joint Graph Structure and Representation Learning

- Minimize a joint loss function combining both the prediction loss and the graph regularization loss.

Minimizing a joint loss function

$$\mathcal{L} = \mathcal{L}_{\text{pred}} + \mathcal{L}_{\mathcal{G}}$$

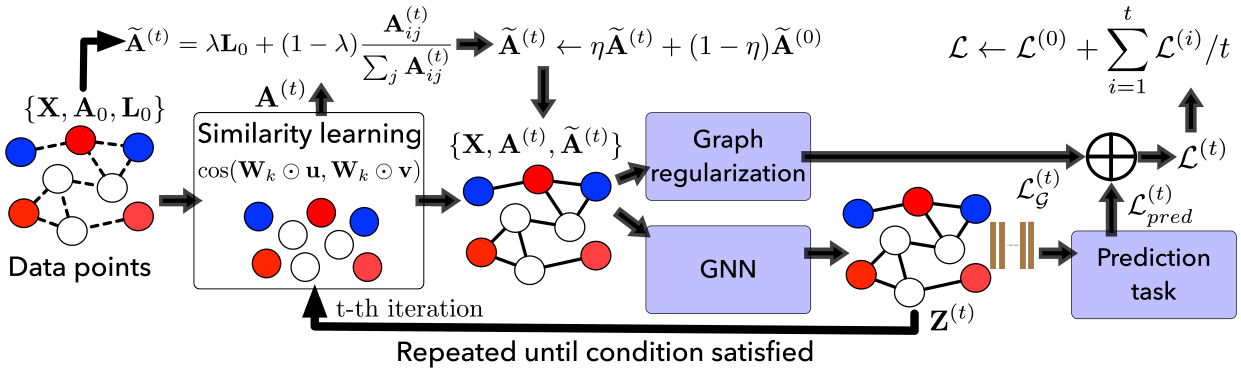
Task-dependent prediction loss

Graph regularization loss

Iterative Deep Graph Learning Framework

- A graph learning network for generating a graph topology
- A graph embedding network for generating node embeddings
- A prediction network for the downstream task
- A graph regularization network

The overall architecture of our framework



Experimental Results and Analysis: Transductive Setting

Table 1: Test accuracy ($\pm$ standard deviation) in percentage on various classification datasets in the transductive setting. The star symbol means that we run the experiments and report the results.

Methods	Cora	Citeseer	Wine	Cancer	Digits
LogReg	60.8 (0.0)	62.2 (0.0)	92.1 (1.3)	93.3 (0.5)	85.5 (1.5)
Linear SVM	58.9 (0.0)	58.3 (0.0)	93.9 (1.6)	90.6 (4.5)	87.1 (1.8)
RBF SVM	59.7 (0.0)	60.2 (0.0)	94.1 (2.9)	91.7 (3.1)	86.9 (3.2)
RF	58.7 (0.4)	60.7 (0.7)	93.7 (1.6)	92.1 (1.7)	83.1 (2.6)
FFNN	56.1 (1.6)	56.7 (1.7)	89.7 (1.9)	92.9 (1.2)	36.3 (10.3)
LP	37.8 (0.2)	23.2 (6.7)	89.8 (3.7)	76.6 (0.5)	91.9 (3.1)
ManiReg	62.3 (0.9)	67.7 (1.6)	90.5 (0.1)	81.8 (0.1)	83.9 (0.1)
SemiEmb	63.1 (0.1)	68.1 (0.1)	91.9 (0.1)	89.7 (0.1)	90.9 (0.1)
LDS	84.1 (0.4)	75.0 (0.4)	97.3 (0.4)	94.4 (1.9)	92.5 (0.7)
GCN	81.0 (0.2)	70.9 (0.3)	—	—	—
GAT	82.5 (0.4)	70.9 (0.4)	—	—	—
kNN-GCN	—	—	95.9 (0.9)	94.7 (1.2)	89.5 (1.3)
LDS*	83.9 (0.6)	**74.8 (0.3)**	96.9 (1.4)	93.4 (2.4)	90.8 (2.5)
IDGL	**84.5 (0.3)**	74.1 (0.2)	**97.8 (0.6)**	**95.1 (1.0)**	**93.1 (0.5)**

Transductive setting!!!

- IDGL outperforms LDS in 4 out of 5 benchmarks.
- IDGL can greatly help the downstream task even when the graph topology is given (Cora & Citeseer).
- Compared to kNN-GCN, IDGL consistently achieves much better results on all datasets.

Experimental Results and Analysis: Inductive Setting

Table 2: Test scores ($\pm$ standard deviation) in percentage on classification (accuracy) and regression (R^2) datasets in the inductive setting.

Inductive setting!!!

Methods	20News	MRD
BiLSTM	80.0 (0.4)	53.1 (1.4)
kNN-GCN	81.3 (0.6)	60.1 (1.5)
IDGL	**83.6 (0.4)**	**63.7 (1.8)**

❖ IDGL performs well in inductive learning as well.

Experimental Results and Analysis: Ablation Study

Table 3: Ablation study on various classification datasets.

Methods	Cora	Citeseer	Wine	Cancer	Digits	20News
IDGL	**84.5 (0.3)**	**74.1 (0.2)**	**97.8 (0.6)**	**95.1 (1.0)**	**93.1 (0.5)**	**83.6 (0.4)**
w/o graph reg.	84.3 (0.4)	71.5 (0.9)	97.3 (0.8)	94.9 (1.0)	91.5 (0.9)	83.4 (0.5)
w/o IL	83.5 (0.6)	71.0 (0.8)	97.2 (0.8)	94.7 (0.9)	92.4 (0.4)	83.0 (0.4)

- Both the iterative learning component and the graph regularization component are important!

Experimental Results and Analysis: Robustness Testing

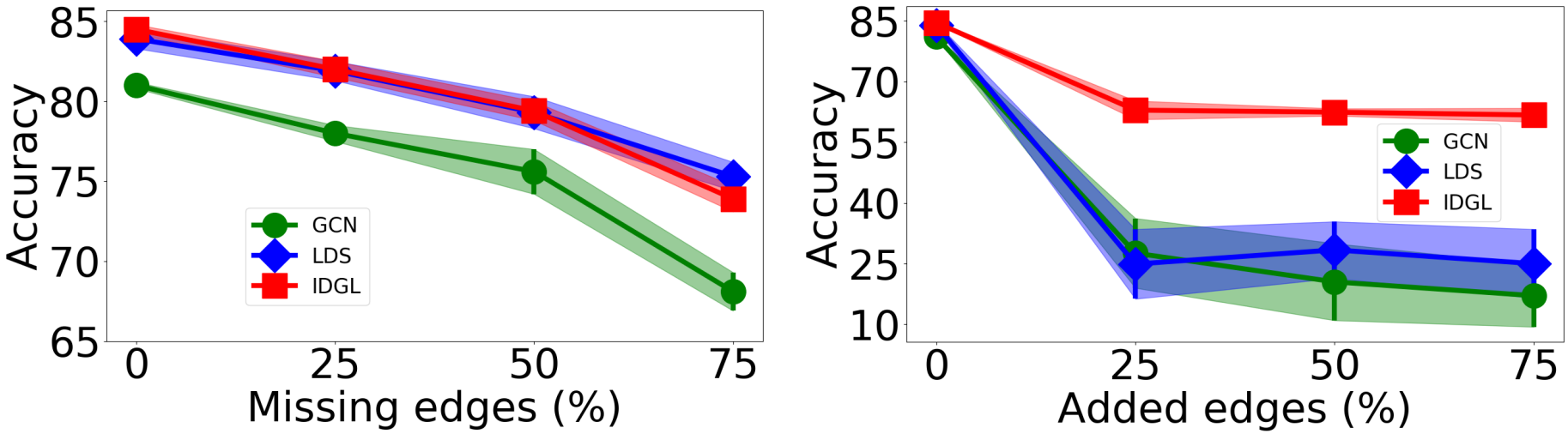


Figure 3. Test accuracy (± standard deviation) in percentage for edge deletion and addition scenarios on Cora test set.

Experimental Results and Analysis: Convergence Testing

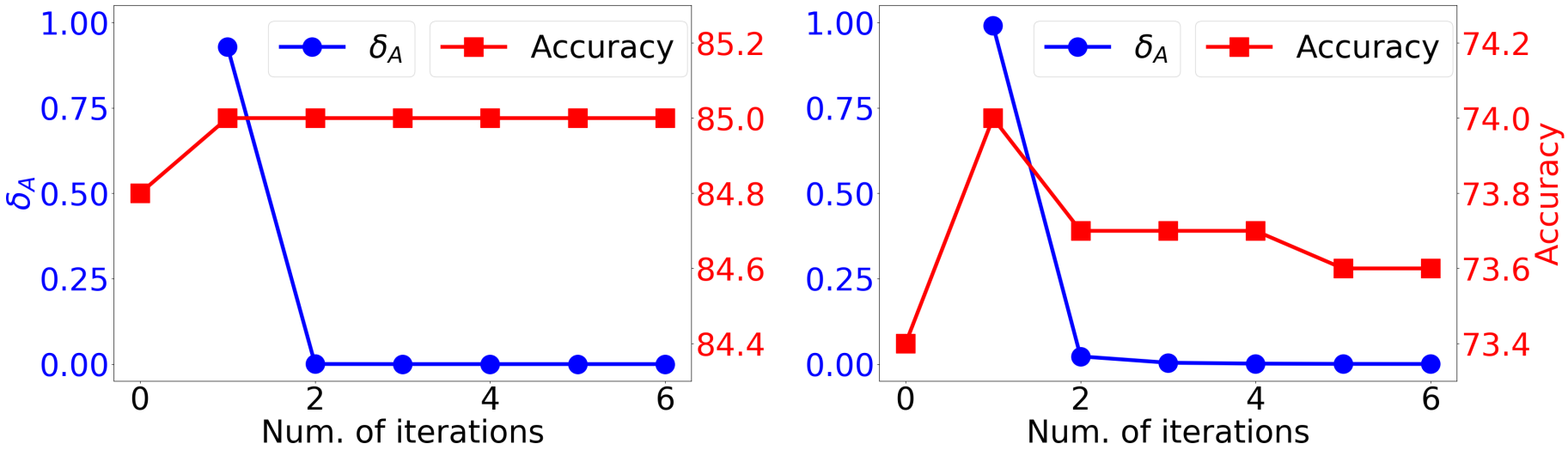


Figure 4. Evolution of the learned adjacency matrix and test accuracy (in %) through iterations in the iterative learning procedure.

Experimental Results and Analysis: Stopping Criterion

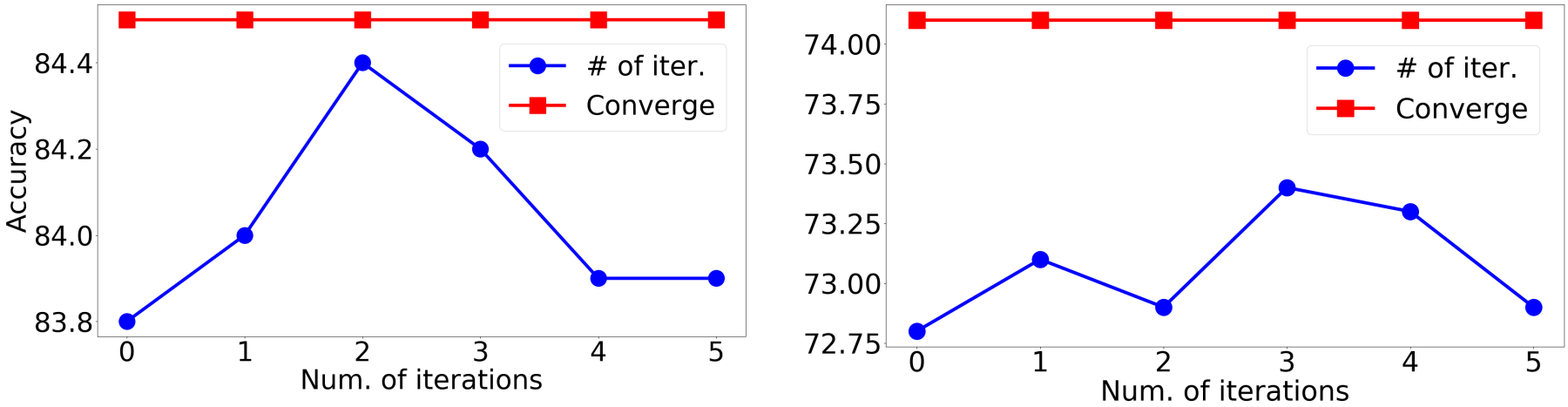


Figure 5. Performance comparison (i.e., test accuracy in %) of two different stopping strategies: i) using a fixed number of iterations (blue line), and ii) using a stopping criterion to dynamically determine the convergence (red line).

Experimental Results and Analysis: Graph Visualization

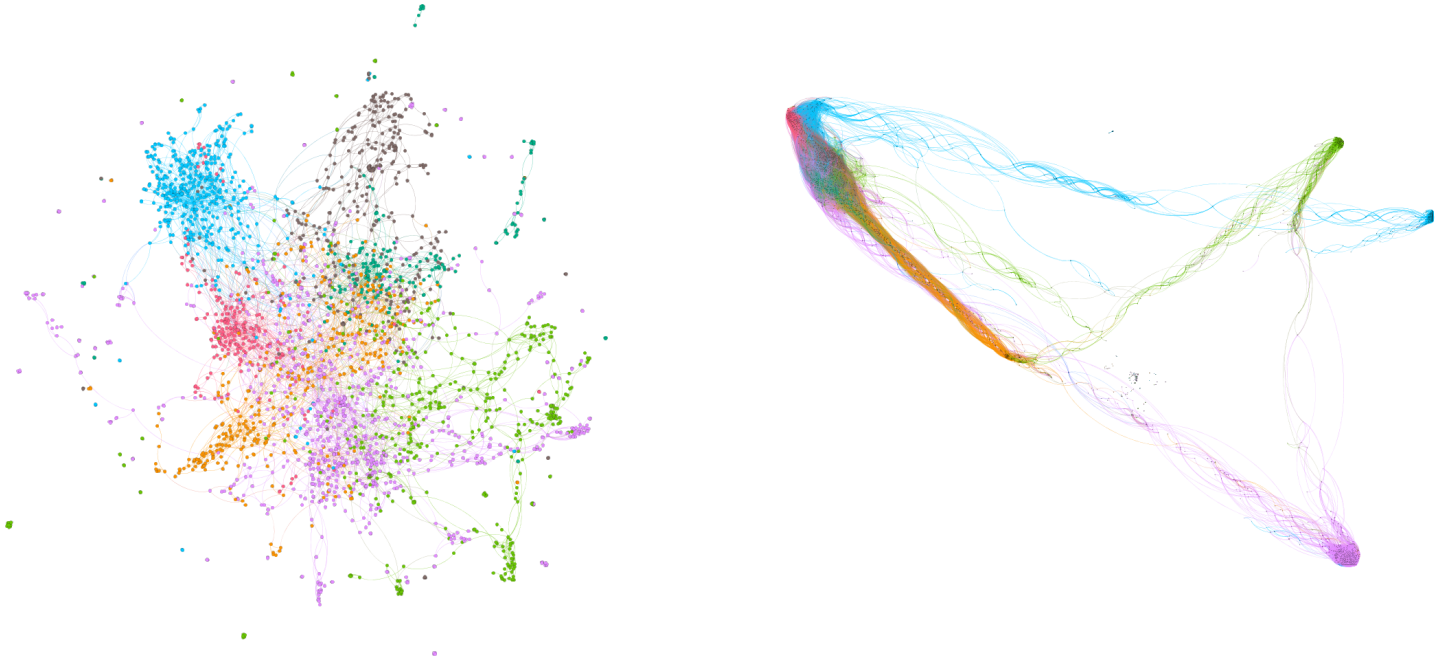


Figure 6. Visualization of the initial graph structure and the learned graph structure ($\mathbf{A}^{(t)}$) on Cora. Colors indicate different node labels.

Conclusion & Future Work

- We proposed a novel end-to-end graph learning framework for jointly learning the graph structure and graph embedding that are optimized towards the prediction task at hand.

- Our extensive experiments demonstrate the effectiveness of the proposed model.

- Future work includes improving the scalability of the proposed method for large graphs.

References

[1] Dong, Xiaowen, et al. "Learning Laplacian matrix in smooth graph signal representations." *IEEE Transactions on Signal Processing* 64.23 (2016): 6160-6173.

[2] Kalofolias, Vassilis. "How to learn a graph from smooth signals." Artificial Intelligence and Statistics. 2016.

[3] Kalofolias, Vassilis, and Nathanaël Perraudin. "Large scale graph learning from smooth signals." arXiv preprint arXiv:1710.05654 (2017).

[4] Egilmez, Hilmi E., Eduardo Pavez, and Antonio Ortega. "Graph learning from data under Laplacian and structural constraints." *IEEE Journal of Selected Topics in Signal Processing* 11.6 (2017): 825-841.

[5] Sukhbaatar, Sainbayar, and Rob Fergus. "Learning multiagent communication with backpropagation." Advances in Neural Information Processing Systems. 2016.

[6] Van Steenkiste, Sjoerd, et al. "Relational neural expectation maximization: Unsupervised discovery of objects and their

[7] Kipf, Thomas, et al. "Neural relational inference for interacting systems." arXiv preprint arXiv:1802.04687 (2018).

[8] interactions." arXiv preprint arXiv:1802.10353 (2018).

[9] Chen, Yu, Lingfei Wu, and Mohammed J. Zaki. "Graphflow: Exploiting conversation flow with graph neural networks for conversational machine comprehension." *arXiv preprint arXiv:1908.00059* (2019).

[10] Chen, Yu, Lingfei Wu, and Mohammed J. Zaki. "Reinforcement Learning Based Graph-to-Sequence Model for Natural Question Generation." *arXiv preprint arXiv:1908.04942*(2019).

[11] Li, Ruoyu, et al. "Adaptive graph convolutional neural networks." *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.

[12] Liu, Pengfei, et al. "Contextualized non-local neural networks for sequence learning." *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 2019.

[13] Belkin, Mikhail, and Partha Niyogi. "Laplacian eigenmaps and spectral techniques for embedding and clustering." Advances in neural information processing systems. 2002.

[14] Franceschi, Luca, et al. "Learning discrete structures for graph neural networks." arXiv preprint arXiv:1903.11960 (2019).

[15] Veličković, Petar, et al. "Graph attention networks." arXiv preprint arXiv:1710.10903 (2017).

[16] Jiang, Bo, et al. "Semi-Supervised Learning With Graph Learning-Convolutional Networks." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2019.

Thank you!
Q&A

Backup

Experimental Results and Analysis: Setup

Data statistics

Benchmarks	Train/Dev/Test	Task	Setting
Cora	140/500/1,000	node classification	transductive
Citeseer	120/500/1,000	node classification	transductive
Wine	10/20/158	node classification	transductive
Cancer	10/20/539	node classification	transductive
Digits	50/100/1,647	node classification	transductive
20News	7,919/3,395/7,532	graph classification	inductive
MRD	3,003/1,001/1,002	graph regression	inductive

We ran our experiments 5 times with different random seeds.

Full Algorithm

- Repeatedly refining the adjacency matrix with the updated node embeddings.
- Repeatedly refining the node embeddings with the updated adjacency matrix.
- After all iterations, the overall loss will be back-propagated through all previous iterations to update the model parameters.

The IDGL algorithm

Algorithm 1: IDGL: Iterative Deep Graph Learning Framework

Input: $\mathbf{X}, \mathbf{y}[, \mathbf{A}_0]$
Parameters : $m, \varepsilon, \alpha, \beta, \gamma, \lambda, \delta, T, \eta[, k]$
Output: $\Theta, \tilde{\mathbf{A}}^{(t)}, \hat{\mathbf{y}}$

$[\mathbf{A}_0 \leftarrow \text{kNN}(\mathbf{X}, k)]$ // Init. $\mathbf{A}_0$ to kNN graph if $\mathbf{A}_0$ is unavailable
$\mathbf{A}^{(0)}, \tilde{\mathbf{A}}^{(0)} \leftarrow \{\mathbf{X}, \mathbf{A}_0\}$ using Eqs. (2), (3) and (10) // Learn the adj. matrix
$\mathbf{Z}^{(0)} \leftarrow \{\tilde{\mathbf{A}}^{(0)}, \mathbf{X}\}$ using Eq. (7) // Compute node embeddings
$\mathcal{L}_{\text{pred}}^{(0)} \leftarrow \{\tilde{\mathbf{A}}^{(0)}, \mathbf{Z}^{(0)}, \mathbf{y}\}$ using Eqs. (8) and (9) // Compute prediction loss
$\mathcal{L}_{\mathcal{G}}^{(0)} \leftarrow \{\mathbf{A}^{(0)}, \mathbf{X}\}$ using Eqs. (4)–(6) // Compute graph regularization loss
$\mathcal{L}^{(0)} \leftarrow \mathcal{L}_{\text{pred}}^{(0)} + \mathcal{L}_{\mathcal{G}}^{(0)}$ // Compute joint loss
$t \leftarrow 0$
while $(t == 0 \quad or \quad ||\mathbf{A}^{(t)} - \mathbf{A}^{(t-1)}||_F^2 > \delta||\mathbf{A}^{(0)}||_F^2) \quad and \quad t < T$ **do**
  $t \leftarrow t + 1$
  $\mathbf{A}^{(t)}, \tilde{\mathbf{A}}^{(t)} \leftarrow \{\mathbf{Z}^{(t-1)}, \mathbf{A}_0\}$ using Eqs. (2), (3) and (10) // Refine the adj. matrix
  $\bar{\mathbf{A}}^{(t)} \leftarrow \{\tilde{\mathbf{A}}^{(t)}, \tilde{\mathbf{A}}^{(0)}\}$ using Eq. (11)
  $\mathbf{Z}^{(t)} \leftarrow \{\bar{\mathbf{A}}^{(t)}, \mathbf{X}\}$ using Eq. (7) // Refine node embeddings
  $\hat{\mathbf{y}} \leftarrow \{\bar{\mathbf{A}}^{(t)}, \mathbf{Z}^{(t)}\}$ using Eq. (8) // Compute task output
  $\mathcal{L}_{\text{pred}}^{(t)} \leftarrow \{\hat{\mathbf{y}}, \mathbf{y}\}$ using Eq. (9)
  $\mathcal{L}_{\mathcal{G}}^{(t)} \leftarrow \{\mathbf{A}^{(t)}, \mathbf{X}\}$ using Eqs. (4)–(6)
  $\mathcal{L}^{(t)} \leftarrow \mathcal{L}_{\text{pred}}^{(t)} + \mathcal{L}_{\mathcal{G}}^{(t)}$
end
$\mathcal{L} \leftarrow \mathcal{L}^{(0)} + \sum_{i=1}^{t} \mathcal{L}^{(i)}/t$
if *Training* **then**
  Back-propagate $\mathcal{L}$ to update model weights Θ
end

Iterative Method for Graph Learning (cont'd)

- The iterative method dynamically stops when the learned graph structure approaches close enough to the optimal graph.

Dynamic stopping criterion

$$||\mathbf{A}^{(t)} - \mathbf{A}^{(t-1)}||_F^2 \leq \delta||\mathbf{A}^{(0)}||_F^2$$

Iteration stops when the graph structure stops improving