Dynamic Graph Convolutional Network: A Topology Optimization Perspective

Bowen Deng, Aimin Jiang

College of Internet of Things Engineering
Hohai University, China

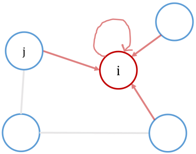
MLSP, Oct, 2021

Contents

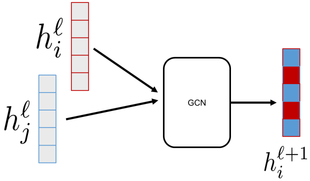
Node Classification over Graphs

The node classification over a graph $G(\mathcal{V}, \mathcal{E})$ can be done via MLP and graph convolutional network(GCN).

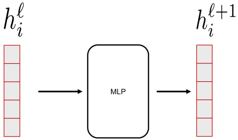
$$h_i^{(\ell+1)} = \mathrm{ReLU}\left(W \cdot \mathrm{MEAN}\left\{h_j^{(\ell)}, \forall j \in \mathcal{N}(i) \cup \{i\}\right\}\right)$$



(a) message passing



(b) GCN



(c) MLP

Motivation: Topology Optimization

In both node clustering and classification tasks, we are inclined to make

- features of intra-class nodes similar
- features of inter-class nodes different

This can be achieved in GCN by

- enhancing the connection between intra-class nodes
- weakening those between inter-class vertices

Motivation: Topology Optimization

In the context of graph neural networks, a quantitative index to reflect topology optimization is defined as follows:

$$
\begin{aligned}
\text{(inter-ratio)} IR &= \frac{w_2}{w_1 + w_2} \\
w_1 &= \sum_{i=1}^{N} \sum_{j \in \mathcal{N}_1(i)} \frac{P_{ij} \cdot |\mathcal{N}_1(i)|}{\sum_{k \in \mathcal{N}_1(i)} P_{ik}} \mathcal{I}(c_i, c_j) \\
w_2 &= \sum_{i=1}^{N} \sum_{j \in \mathcal{N}_1(i)} \frac{P_{ij} \cdot |\mathcal{N}_1(i)|}{\sum_{k \in \mathcal{N}_1(i)} P_{ik}} \left[1 - \mathcal{I}(c_i, c_j)\right]
\end{aligned}
$$

- c_i: the true label of node i
- $\mathcal{I}(c_i, c_j) = 1$ if $c_i = c_j$ else 0
- $\mathcal{N}_1(i)$: the set of 1-hop neighbors/predecessors of node i
- P: the propagation matrix controlling message passing

Scaling the edge weights of Cora citation network with different factors works in our expectation.

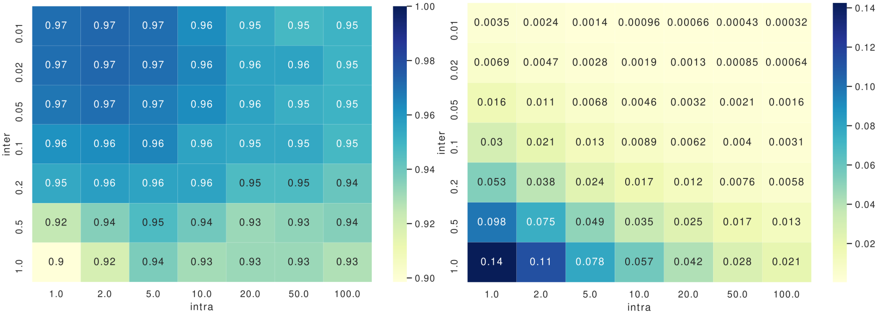


Figure: Accuracy (left) and IR (right) when applying different intensities of **soft** topology optimization

Contents

Vanilla GCN

Given the whole graph, a L-layer vanilla GCN [KW17] is formulated as:

$$
\begin{aligned}
\mathbf{H}^{(L+1)} &= \mathbf{P}^{(L)}\sigma\left(\cdots\sigma\left(\mathbf{P}^{(0)}\mathbf{H}^{(0)}\mathbf{W}^{(0)}\right)\cdots\right)\mathbf{W}^{(L)} && (1)\\
\mathbf{H}^{(\ell+1)} &= \sigma\left(\tilde{\mathbf{P}}\mathbf{H}^{(\ell)}\mathbf{W}^{(\ell)}\right) && (2)\\
\tilde{\mathbf{P}} &= \mathbf{P}^{(L)} = \cdots = \mathbf{P}^{(0)} = \tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}
\end{aligned}
$$

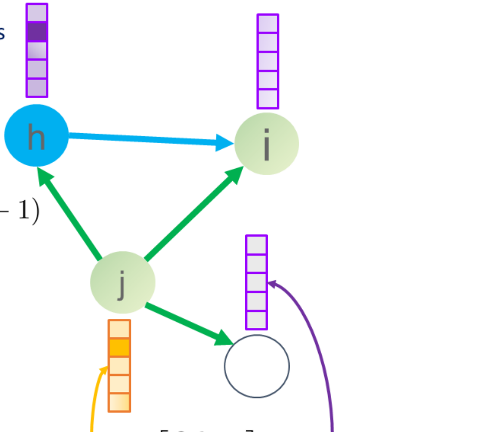
- $\tilde{\mathbf{A}}$: the adjacency matrix with added self-loops
- $\tilde{\mathbf{D}}$: the degree matrix of $\tilde{\mathbf{A}}$
- $\sigma(\cdot)$: ReLU activation

Gradients of Propagation Matrix: Case1

The last layer of GCN: $\mathbf{H}^{(L+1)} = \mathbf{P}^{(L)}\mathbf{H}^{(L)}\mathbf{W}^{(L)} = \mathbf{P}^{(L)}\mathbf{X}^{(L)}$.

The CE loss: $\mathbf{Y} = \text{softmax}(\mathbf{H}^{(L+1)}); \mathcal{L} = \text{CrossEntropy}(\mathbf{Y}, \mathbf{T})$.

Node ***h*** is misclassified as the class of node ***j*** with a very high probability



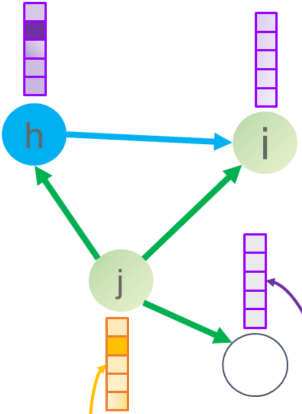
$$\left[\frac{\partial \mathcal{L}_{gcn}}{\partial \mathbf{P}^{(L)}}\right]_{ij} = (\mathbf{y}_i - \mathbf{t}_i) \cdot \mathbf{x}_j^\top$$

$$\begin{aligned}
\left[\frac{\partial \mathcal{L}_{gcn}}{\partial \mathbf{P}^{(L)}}\right]_{hj} &= x_{j,c_j} y_{h,c_j} + x_{j,c_h}\left(y_{h,c_h} - 1\right) \\
&+ \sum_{k \neq c_h, c_j} x_{j,k} y_{h,k} \\
&= \left(x_{j,c_j} - x_{j,c_h}\right) y_{h,c_j} \\
&+ \sum_{k \neq c_h, c_j} y_{h,k}\left(x_{j,k} - x_{j,c_h}\right) \\
&\approx x_{j,c_j} - x_{j,c_h} > 0
\end{aligned}$$

Gradients of Propagation Matrix: Case2

The last layer of GCN: $\mathbf{H}^{(L+1)} = \mathbf{P}^{(L)}\mathbf{H}^{(L)}\mathbf{W}^{(L)} = \mathbf{P}^{(L)}\mathbf{X}^{(L)}$.

The CE loss: $\mathbf{Y} = \text{softmax}(\mathbf{H}^{(L+1)}); \mathcal{L} = \text{CrossEntropy}(\mathbf{Y}, \mathbf{T})$.



To be correctly recognized or not, its ***not*** a question!

$$\left[\frac{\partial \mathcal{L}_{gcn}}{\partial \mathbf{P}^{(L)}}\right]_{ij} = x_{j,c_i}\left(y_{i,c_i} - 1\right) + \sum_{k \neq c_i} x_{j,k} y_{i,k}$$

$$= \sum_{k \neq c_i} y_{i,k}\left(x_{j,k} - x_{j,c_i}\right) < 0$$

$$\left[\frac{\partial \mathcal{L}_{gcn}}{\partial \mathbf{P}^{(L)}}\right]_{ij} = (\mathbf{y}_i - \mathbf{t}_i) \cdot \mathbf{x}_j^\top$$

Gradients of Propagation Matrix: Summary

- Phenomenon: The last propagation matrix has gradients that coincide with the direction of soft topology optimization.
- Hypothesis: We can also train propagation matrices of some previous layers towards soft topology optimization, regardless of ReLU activation.
- Evidences: A group of experiments in Subsection 3.3 of the paper support this hypothesis.

Proposed: DyGCN

$$\mathbf{H}^{(\ell+1)} = \sigma\left(\mathbf{P}^{(\ell)}\mathbf{H}^{(\ell)}\mathbf{W}^{(\ell)}\right) \tag{3}$$

$$\mathbf{P}^{(\ell)} = \text{ SparseSoftMax } \left(\text{ LeakyReLU } \left(\tilde{\mathbf{A}}^{(\ell)}\right)\right) \tag{4}$$

where

- $\tilde{\mathbf{A}}^{(\ell)}$: the augmented trainable adjacency matrix
- $\mathbf{P}^{(\ell)}$: the propagation matrix
- $\mathbf{H}^{(\ell)}$: the input feature matrix of ℓ-the layer
- $\mathbf{H}^{(\ell+1)}$: the output feature matrix of ℓ-the layer

Proposed: GCNII(Dy)

Though DyGCN employs different propagation matrices, the aggregation scheme is the same as vanilla GCN:

$$\mathbf{H}^{(\ell+1)} = \sigma\left(\mathbf{P}^{(\ell)}\mathbf{H}^{(\ell)}\mathbf{W}^{(\ell)}\right).$$

Theorem of Linear Expressive Ability

If an L-layer DyGCN model is linear and the Markov chain corresponding to $\mathbf{P}^{(\ell)}$ is ergodic, the final output $\mathbf{H}^{(L)}$ converges to a rank-1 matrix as $L \to \infty$.

We improve DyGCN with those deep architectures, *e.g.*, JKNet [XLT+18] and GCNII [CWH+20] to get decent deep variants GCNII(Dy) and JKNet(Dy).

Dynamic Models & Soft Topology Optimization

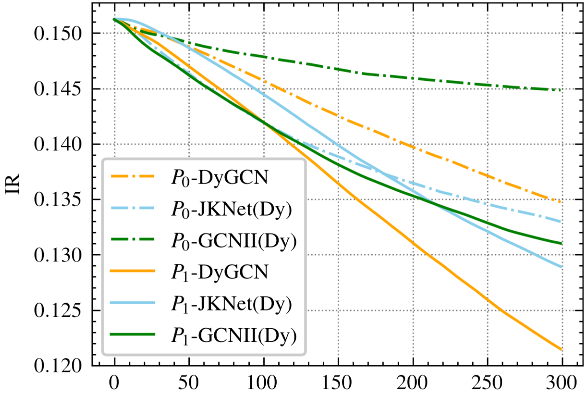
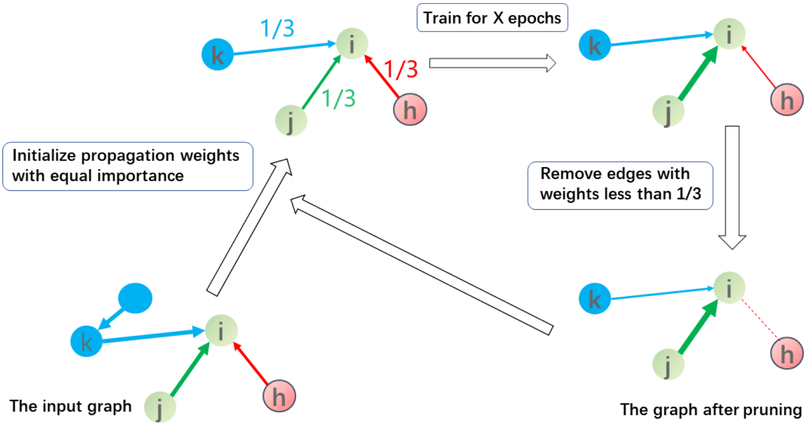


Figure: IR-epoch Curves of 4-layer(2-block) DyGCN, GCNII(Dy), and JKNet(Dy), drawn from subsection3.3 of the paper

Hard Topology Optimization

- IR decreases
  - $\Rightarrow$ Many inter-class edges are weakened
  - $\Rightarrow$ Many weakened edges are inter-class
- Adaptive dropping(AdaDrop) removes the weakened edges.



Contents

Experiment Setting

- Evaluate models in node classification tasks on three citation networks [SNB+08].
- Train, Validation, and Test sets are split according to 6:2:2.
- Report the averaging test accuracy over 10 random splits.
- Test accuracies are reported when the model gets the best performance on the validation set.

Dataset	Classes	Nodes	Edges	Features
Cora	7	2708	5429	1433
Citeseer	6	3327	4723	3703
Pubmed	3	19717	44338	500

Table: The dataset information

Results on Cora

Method	$L = 2$	$L = 4$	$L = 8$	$L = 16$	$L = 32$
GCN	87.95	87.55	50.85	29.98	29.98
JKNet	-	86.03	85.59	86.53	86.05
GCN-LPA	87.20	86.62	60.46	31.88	31.22
GCNII	-	88.15	**88.82**	88.78	88.89
DyGCN	89.06	87.93	74.91	28.52	30.06
DyGCN(Lin)	88.80	87.88	86.66	81.66	33.47
DyGCN(AD)	**89.28**	88.19	85.87	33.75	30.00
GCNII(Dy)	-	88.36	88.36	**88.89**	**89.26**
GCNII(AD,Dy)	-	**89.00**	88.71	88.71	89.21
JKNet(Dy)	-	88.41	87.93	88.06	85.92
JKNet(AD,Dy)	-	88.76	88.49	88.14	88.08

Table: Test accuracies (in percent) on Cora

Summary

- Conclusions
  - DyGCN is more robust to over-smoothness than GCN and GCN-LPA, though it is a theoretically shallow GCN model.
  - Differentiable propagation matrices also succeed in GCNII and JKNet architectures.
  - AdaDrop is effective as it can enhance dynamic models in most cases.
- Potential future directions
  - Apply dynamic models to graph-level tasks.
  - Use dynamic models and AdaDrop in graph adversarial learning [ZAG18].

Thank you for listening!

References I

[CWH+20] Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding Ding, and Yaliang Li, *Simple and deep graph convolutional networks*, Proceedings of the 37th International Conference on Machine Learning (2020).

[KW17] Thomas N. Kipf and Max Welling, *Semi-supervised classification with graph convolutional networks*, 5th International Conference on Learning Representations (2017).

[SNB+08] Prithviraj Sen, Galileo Mark Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad, *Collective classification in network data*, AI Magazine (2008).

References II

[XLT+18] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken Ichi Kawarabayashi, and Stefanie Jegelka, *Representation learning on graphs with jumping knowledge networks*, 35th International Conference on Machine Learning (2018).

[ZAG18] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann, *Adversarial attacks on neural networks for graph data*, Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2018).