# Combining Graph Convolutional Neural Networks and Label Propagation

TOIS2021

**Hongwei Wang and Jure Leskovec**

Reporter: Minjie Cheng

March 8, 2023

# Overview

- Probelm Formulation
- GCNs and LPA
- GCN-LPA(an end-to-end model that combines GCN and LPA)
- Experiments
- Conclusion

# Probelm Formulation

Consider a graph $G = (V, \boldsymbol{A}, \boldsymbol{X}, \boldsymbol{Y})$

- Node set $V = \{v_1, ..., v_n\}$
- Adjacency matrix $\boldsymbol{A} \in \{0, 1\}^{n \times n}$
- Node feature matrix $\boldsymbol{X} \in R^{n \times d}$
- Label set $L = \{1, ..., c\}$
- The first m nodes have labels
- Node label matrix $\boldsymbol{Y} \in R^{n \times c}$, $\{\boldsymbol{y}_1, ..., \boldsymbol{y}_m\}$ are one-hot label indicator vectors.
- Learn a mapping $\mathcal{M} : V \to L$ and predict labels of unlabeled $\{\boldsymbol{y}_{m+1}, ..., \boldsymbol{y}_n\}$.

# GCNs and LPA

- GCN propagates and transforms **node features** across the graph.
- LPA propagates **labels** along the edges.

# GCNs

The feature propagation scheme of GCN in layer $k$ is as follows:

$$\boldsymbol{X}^{(k+1)} = \sigma(\tilde{\boldsymbol{A}}\boldsymbol{X}^{(k)}\boldsymbol{W}^{(k)}) \tag{1}$$

- $\tilde{\boldsymbol{A}}$ is the normalized adjacency matrix.
- $\boldsymbol{X}^{(k)}$ is the $k$th layer node representations.
- $\boldsymbol{W}^{(K)}$ is trainable matrix in the layer $k$.
- $\sigma(.)$ is an activation function.

# LPA

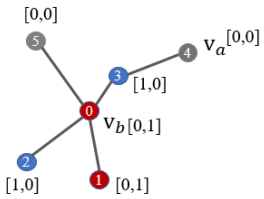
The feature propagation scheme of GCN in layer $k$ is as follows:

$$\boldsymbol{Y}^{(k+1)} = \sigma(\tilde{\boldsymbol{A}}\boldsymbol{Y}^{(k)}) \tag{2}$$

$$\boldsymbol{y}^{(k+1)} = \boldsymbol{y}_i^{(0)}, \forall i \leq m \tag{3}$$

- Labels of all labels of nodes are reset to their initial values.

# LPA-An Example




**Y**

| | | |
|---|---|---|
| 0 | 1 | 0 |
| 1 | 1 | 0 |
| 2 | 0 | 1 |
| 3 | 0 | 1 |
| 4 | 0 | 0 |
| 5 | 0 | 0 |

**A**

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 2 | 1 | 0 | 0 | 0 | 0 | 0 |
| 3 | 1 | 0 | 0 | 0 | 1 | 0 |
| 4 | 0 | 0 | 0 | 1 | 0 | 0 |
| 5 | 1 | 0 | 0 | 0 | 0 | 0 |

**D**

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 0 | 4 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 1 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 2 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 1 | 0 |
| 5 | 0 | 0 | 0 | 0 | 0 | 1 |

$\widetilde{\mathbf{A}} = \boldsymbol{D}^{-1}\boldsymbol{A}$

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 0 | 0 | 1/4 | 1/4 | 1/4 | 0 | 1/4 |
| 1 | 1/4 | 0 | 0 | 0 | 0 | 0 |
| 2 | 1/4 | 0 | 0 | 0 | 0 | 0 |
| 3 | 1/4 | 0 | 0 | 0 | 1/4 | 0 |
| 4 | 0 | 0 | 0 | 1/4 | 0 | 0 |
| 5 | 1/4 | 0 | 0 | 0 | 0 | 0 |

$\mathbf{Y^{(1)}} = \widetilde{\mathbf{A}}\,\mathbf{Y^{(0)}}$

| | | |
|---|---|---|
| 0 | 1/4 | 1/2 |
| 1 | 1/4 | 0 |
| 2 | 1/4 | 0 |
| 3 | 1/4 | 1/4 |
| 4 | 0 | 1/4 |
| 5 | 1/4 | 0 |

→

$\mathbf{Y^{(1)}}$ $\quad \boldsymbol{y}^{(k+1)} = \boldsymbol{y}_i^{(0)}, \forall i \leq m$

| | | |
|---|---|---|
| 0 | 1 | 0 |
| 1 | 1 | 0 |
| 2 | 0 | 1 |
| 3 | 0 | 1 |
| 4 | 0 | 1/4 |
| 5 | 1/4 | 0 |

# GCNs and LPA

The relationship between the two equations(GCNs and LPA)

- GCNs:

$$\boldsymbol{X}^{(k+1)} = \sigma(\tilde{\boldsymbol{A}}\boldsymbol{X}^{(k)}\boldsymbol{W}^{(k)}) \tag{4}$$

- LPA:

$$\boldsymbol{Y}^{(k+1)} = \sigma(\tilde{\boldsymbol{A}}\boldsymbol{Y}^{(k)}) \tag{5}$$

$$\boldsymbol{y}^{(k+1)} = \boldsymbol{y}_i^{(0)}, \forall i \leq m \tag{6}$$

# GCNs and LPA

The relationship between the two equations(GCNs and LPA)

- GCNs:

$$\boldsymbol{X}^{(k+1)} = \sigma(\tilde{\boldsymbol{A}}\boldsymbol{X}^{(k)}\boldsymbol{W}^{(k)}) \tag{7}$$

- LPA:

$$\boldsymbol{Y}^{(k+1)} = \sigma(\tilde{\boldsymbol{A}}\boldsymbol{Y}^{(k)}) \tag{8}$$

$$\boldsymbol{y}^{(k+1)} = \boldsymbol{y}_i^{(0)}, \forall i \leq m \tag{9}$$

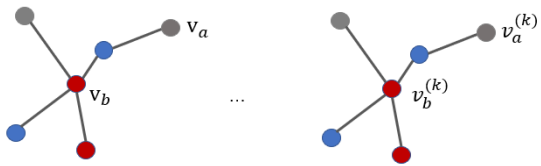
# Two definitions

- **Feature influence**:
  - The feature influence of node $v_b$ on node $v_a$

$$I_f(v_a, v_b; k) = \|\mathbb{E}_{\boldsymbol{W}(.)} \frac{\partial \boldsymbol{x}_a^{(k)}}{\partial \boldsymbol{x}_b}\|_1 \tag{10}$$

  - The normalized feature influence is defined as

$$\tilde{I}_f(v_a, v_b; k) = \frac{I_f(v_a, v_b; k)}{\sum_{v_i \in V} I_f(v_a, v_i; k)} \tag{11}$$



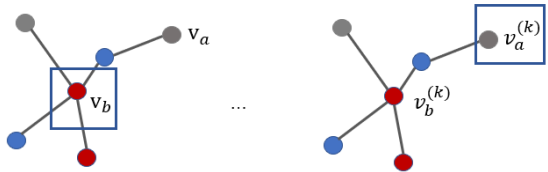
# Two definitions

- **Feature influence**:
  - The feature influence of node $v_b$ on node $v_a$

$$I_f(v_a, v_b; k) = \|\mathbb{E}_{\boldsymbol{W}^{(\cdot)}} \frac{\partial \boldsymbol{x}_a^{(k)}}{\partial \boldsymbol{x}_b}\|_1 \tag{12}$$

  - The normalized feature influence is defined as

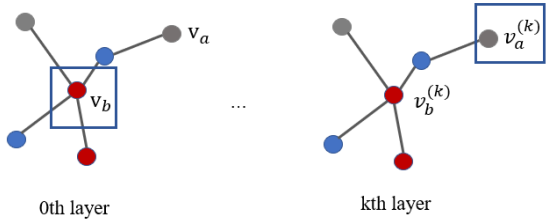
$$\tilde{I}_f(v_a, v_b; k) = \frac{I_f(v_a, v_b; k)}{\sum_{v_i \in V} \tilde{I}_f(v_a, v_i; k)} \tag{13}$$



# Two definitions

- **Label influence**:
  - The label influence of labeled node $v_b$ on unlabeled node $v_a$

$$I_l(v_a, v_b; k) = \frac{\partial \boldsymbol{y}_a^{(k)}}{\partial \boldsymbol{y}_b} \tag{14}$$



# GCNs-LPA

**Theorem1**

$$\mathbb{E}_{\boldsymbol{W}^{(.)}}\left[I_l(v_a, v_b; k)\right] = \sum_{j=1}^{k} \beta^j I_f(v_a, v_b; j)$$

- $\beta$ is the fraction of unlabeled nodes.
- The labels influence of $v_b$ after $k$ iterations of LPA equals to the cumulative normalized feature influence of $v_b$ on $v_a$ after k layers of GCN.

# GCNs-LPA

- We defined **intra-class** feature influence:

$$\sum_{i \in L} \sum_{v_a, v_b : y_a = i, y_b = i} I_f(\tilde{v_a}, v_b) \tag{15}$$

- We can optimize the **intra-class** label influence instead:

$$\sum_{i \in L} \sum_{v_a, v_b : y_a = i, y_b = i} I_l(v_a, v_b) \tag{16}$$

# GCNs-LPA

$$\sum_{i \in L} \sum_{v_a, v_b : y_a = i, y_b = i} I_l(v_a, v_b) = \sum_{v_a} \sum_{v_b : y_b = y_a} I_l(v_a, v_b) \tag{17}$$

**Theorem2**

$$\sum_{v_b : y_b = y_a} I_l(v_a, v_b; k) \propto Pr(\hat{y_a}^{lpc} = y_a)$$

- $\hat{y_a}^{LPA}$ is the predicted label of $v_a$ using a k-iteration LPA.

# GCNs-LPA

According to **Theorem2**, We can therefore first learn the optimal edge weights $\boldsymbol{A}^*$ by minimizing the loss of predicted labels by LPA.

$$\boldsymbol{A}^* = \arg\min_{\boldsymbol{A}} L_{lpa}(\boldsymbol{A}) = \arg\min_{\boldsymbol{A}} \frac{1}{m} \sum_{v_a : a \leq m} J(\hat{\boldsymbol{y}}_a^{lpa}, \boldsymbol{y}_a) \tag{18}$$

- $J$ is the cross-entry loss.
- The optimal $\boldsymbol{A}^*$ maximizes the probability that each node is correctly labeled by LPA, also maximizes the **intra-class label** influence(according to Theorem2) and **intra-class feature** influence(according to Theorem1).

# GCNs-LPA

- We can plug $\boldsymbol{A}^*$ into GCN to predict labels:

$$\boldsymbol{X}^{(k+1)} = \sigma(\boldsymbol{A}^* \boldsymbol{X}^{(k)} \boldsymbol{W}^{(k)}), k = 0, 1, ..., K-1 \tag{19}$$

- The optimal transformation matrices can be learned by minimizing the loss of predicted labels by GCN:

$$\boldsymbol{W}^* = \arg\min_{\boldsymbol{W}} L_{gcn}(\boldsymbol{W}, \boldsymbol{A}^*) = \arg\min_{\boldsymbol{W}} \frac{1}{m} \sum_{v_a : a \leq m} J(\hat{\boldsymbol{y}_a}^{gcn}, \boldsymbol{y}_a) \tag{20}$$

- Combine the above LPA and GCN and train the whole model in an end-to-end fashion:

$$\boldsymbol{W}^*, \boldsymbol{A}^* = \arg\min_{\boldsymbol{W}, \boldsymbol{A}} L_{gcn}(\boldsymbol{W}, \boldsymbol{A}) + \lambda L_{lpa}(\boldsymbol{A}) \tag{21}$$

# GCNs-LPA

$\boldsymbol{A}^* = \boldsymbol{M} \circ \boldsymbol{A}$

- Each element $M_{i,j}$ can be a free variable during training.
  - It can only work in **transductive** setting while the trained model cannot be used for new graphs.
  - For large graphs, the number of **model paramaters** increases linearly with the number of edges.
- Each element $M_{ij}$ can be set as a function of features of two endpoints e.g., $M_{ij} = k(\boldsymbol{x}_i^T \boldsymbol{H} \boldsymbol{x}_j)$
  - The learned $\boldsymbol{H}$ can be applied to new graphs in inductive setting.
  - The model size is also independent with the graph size.

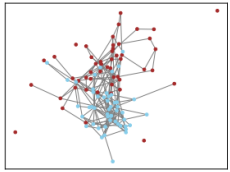
# GCNs-LPA

$\boldsymbol{A}^* = \boldsymbol{M} \circ \boldsymbol{A}$

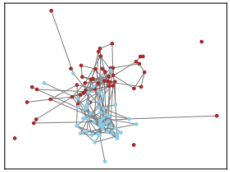
- Each element $M_{i,j}$ can be a free variable during training.Semi-Supervised Node Classification
  - It can only work in **transductive** setting while the trained model cannot be used for new graphs.
  - For large graphs, the number of **model paramaters** increases linearly with the number of edges.
- Each element $M_{ij}$ can be set as a function of features of two endpoints e.g., $M_{ij} = k(\boldsymbol{x}_i^T \boldsymbol{H} \boldsymbol{x}_j)$. Knowledge-Graph-Aware Recommendation
  - The learned $\boldsymbol{H}$ can be applied to new graphs in inductive setting.
  - The model size is also independent with the graph size.

# Experiments

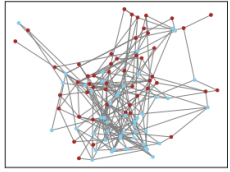
## GCN-LPA is more robust to inter-class edges than GCNs



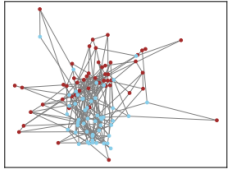
(a) GCN on the original network



(b) GCN-LPA on the original network



(c) GCN on the noisy network



(d) GCN-LPA on the noisy network

Fig. 2. Node embeddings of a subgraph of Cora trained on a node classification task (red vs. blue). Node coordinates in Figures 2(a)–2(d) are the embedding coordinates. Notice that GCN does not produce linearly separable embeddings (Figure 2(a) vs. Figure 2(b)), while GCN-LPA performs much better even in the presence of noisy edges (Figure 2(c) vs. Figure 2(d)). Additional visualizations are included in Appendix D.

# Semi- Supervised Node Classification

Three variants of GCN-TPA

- GCN-LPA(T) learns the optimal adjacency matrix $\boldsymbol{A}$ by minimizing $L_{lpa}$ first, then freezes $\boldsymbol{A}$ and optimizes $\boldsymbol{W}$ by minimizing $L_{gcn}$.
- GCN-LPA(S) optimizes $\boldsymbol{A}$ and $\boldsymbol{W}$, but the gradient of $\boldsymbol{A}$ only propagates back form $L_{lpa}$.
- GCN+LPA simply adds predictions of GCN and LPA together.

# Semi- Supervised Node Classification

Table 3. Mean and the 95% Confidence Intervals of Test Set Accuracy for All Methods and Datasets in Node Classification Task

| Method | Cora | Citeseer | Pubmed | Coauthor-CS | Coauthor-Phy |
|---|---|---|---|---|---|
| Logistic Regression | 57.0 ± 1.6 | 61.2 ± 1.5 | 64.0 ± 2.3 | 86.1 ± 0.8 | 86.7 ± 1.3 |
| LPA | 74.4 ± 2.0 | 67.5 ± 1.3 | 70.6 ± 3.4 | 73.8 ± 1.4 | 86.2 ± 1.5 |
| GCN | 81.4 ± 0.8 | 71.8 ± 1.3 | 77.6 ± 2.0 | 90.9 ± 0.5 | 92.3 ± 0.9 |
| GAT | 80.8 ± 1.2 | 71.3 ± 1.4 | 76.8 ± 1.6 | 90.4 ± 0.5 | 92.2 ± 0.7 |
| JK-Net | 81.1 ± 1.2 | 70.3 ± 1.0 | 77.4 ± 0.5 | 90.3 ± 0.3 | 90.8 ± 0.6 |
| GIN | 74.2 ± 0.9 | 60.5 ± 1.1 | 73.3 ± 1.0 | 84.2 ± 1.1 | 87.1 ± 0.9 |
| GDC | **83.2** ± 0.8 | 72.0 ± 0.9 | 77.8 ± 0.7 | 91.1 ± 0.6 | 92.1 ± 0.4 |
| GCN-LPA(T) | 82.6 ± 0.7 | 72.4 ± 0.7 | 78.2 ± 1.0 | 91.5 ± 0.5 | 93.1 ± 0.7 |
| GCN-LPA(S) | 82.9 ± 0.8 | 72.4 ± 0.9 | 78.4 ± 0.9 | **91.8** ± 0.4 | 93.4 ± 0.8 |
| GCN+LPA | 78.3 ± 0.5 | 69.9 ± 1.1 | 74.0 ± 0.6 | 84.4 ± 0.8 | 89.9 ± 0.7 |
| GCN-LPA | 83.1 ± 0.7 | **72.6** ± 0.8 | **78.6** ± 1.3 | **91.8** ± 0.4 | **93.6** ± 1.0 |

The best result is highlighted in bold, while the result falling within the confidence interval of the highest one is highlighted with underline.

# Efficacy of LPA regularization

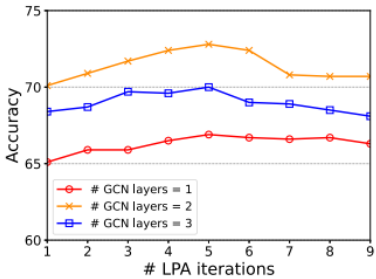



Fig. 3. Sensitivity to the number of LPA iterations on Citeseer dataset.

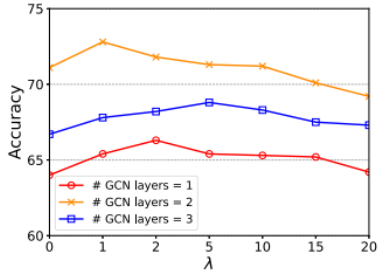



Fig. 4. Sensitivity to $\lambda$ (weight of LPA loss) on Citeseer dataset.

- Figure 4 shows that training without the LPA loss term($\lambda$=0) is more difficult than the case where $\lambda = 1 - 5$

# Influence of labeled node rate

Table 4. Accuracy of LPA, GCN, and GCN-LPA on Citeseer Dataset with Different Labeled Node Rate

| Labeled node rate | 2% | 5% | 10% | 20% | 40% | 60% |
|---|---|---|---|---|---|---|
| LPA | 65.0 ± 1.1 | 67.7 ± 1.5 | 68.2 ± 1.0 | 70.5 ± 1.4 | 71.6 ± 1.3 | 73.9 ± 0.8 |
| GCN | 69.6 ± 0.9 | 72.1 ± 1.2 | 72.4 ± 1.0 | 74.4 ± 0.6 | 75.7 ± 0.5 | 79.1 ± 0.8 |
| GCN-LPA | **70.2** ± 0.7 | **72.7** ± 1.0 | **73.3** ± 0.8 | **75.3** ± 1.1 | **77.3** ± 0.8 | **80.9** ± 1.0 |

- GCN-LPA outperformers GCN and LPA consistently, the improvement achieved by GCN-LPA increases when labeled node rate is larger.

# Visualization of learned edge weights

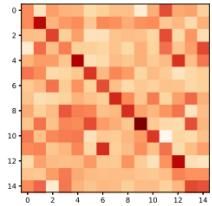



Fig. 5. Visualization of learned edge weights on Coauthor-CS dataset. The numbers along axes denote different node labels.

- It is clear that values along the diagonal(**intra-class** edges weights) are significantly larger than off-diagonal values(**inter-class** edge weights).
- GCN-LPA is able to identify the importance of edges and distinguish inter-class and intra-class edges.

# Knowledge-Graph-Aware Recommendation

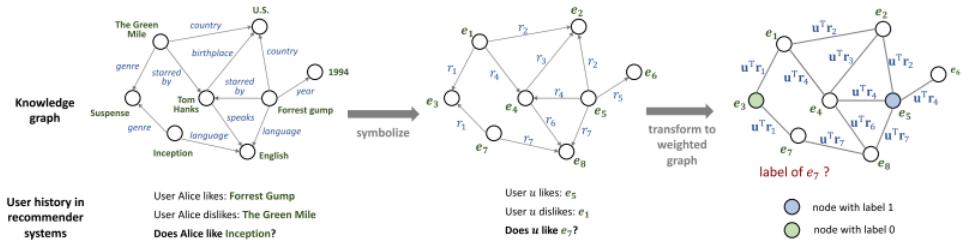



Fig. 7. Left: An instance of movie knowledge graph and the rating history of user Alice in recommender systems. Middle: The symbolized knowledge graph and user history. Right: Transforming the knowledge graph into a weighted graph, in which the weight of edge with relation $r$ is $\mathbf{u}^\top \mathbf{r}$, where $\mathbf{u}$ and $\mathbf{r}$ are the trainable embeddings for user $u$ and relation $r$, respectively. Nodes that user $u$ likes are assigned with label 1 and nodes that user $u$ dislikes are assigned with label 0. In this way, the KG-aware recommendation problem is transformed to a binary node classification problem.

- We apply GCN-LPA to KG-aware recommendation problem.

# Knowledge-Graph-Aware Recommendation

Table 8. The Results of *Recall@K* for Book-Crossing Dataset in Recommendation Task

| Model | R@2 | R@10 | R@50 | R@100 |
|---|---|---|---|---|
| SVD | 2.5 ± 0.3 | 4.6 ± 0.5 | 7.8 ± 0.4 | 10.9 ± 0.4 |
| LibFM | 3.1 ± 0.5 | 6.1 ± 0.5 | 9.2 ± 0.5 | 12.5 ± 0.6 |
| LibFM + TransE | 3.7 ± 1.3 | 6.5 ± 1.2 | 9.7 ± 1.2 | 13.0 ± 1.3 |
| PER | 2.1 ± 0.8 | 4.0 ± 1.0 | 6.4 ± 1.1 | 7.0 ± 1.0 |
| CKE | 2.6 ± 0.4 | 5.2 ± 0.4 | 7.8 ± 0.5 | 11.2 ± 0.6 |
| RippleNet | 3.5 ± 0.6 | 7.4 ± 0.6 | 10.7 ± 0.7 | 12.8 ± 0.7 |
| KGIN | 3.3 ± 0.4 | 7.0 ± 0.5 | 10.2 ± 0.7 | 12.6 ± 0.5 |
| GCN-LPA | **4.5** ± 0.4 | **8.3** ± 0.5 | **11.6** ± 0.4 | **14.9** ± 0.4 |

- GCN-LPA outperforms baselines by a significant margin.

# Conclusion

- We study the theoretical relationship between GCNs and LPA.
- We propose a unified model, which learns transformation matrices and edge weights simultaneously on GCN with the assistance of **LPA regularizer**.

**The connection to my work (OTGNN)**

$$\min_{\boldsymbol{q},\boldsymbol{\theta}} \|\boldsymbol{W_e q}\|_1 + \lambda \|\boldsymbol{Y}_L - (\mathrm{GNN_L}(\boldsymbol{X}, \boldsymbol{q}, \boldsymbol{\theta}) + \boldsymbol{J}_L \boldsymbol{q})\|_2^2 \tag{22}$$

- We design the **update strategy** for edge weights based on the labeled nodes from the OT on graph perspective.