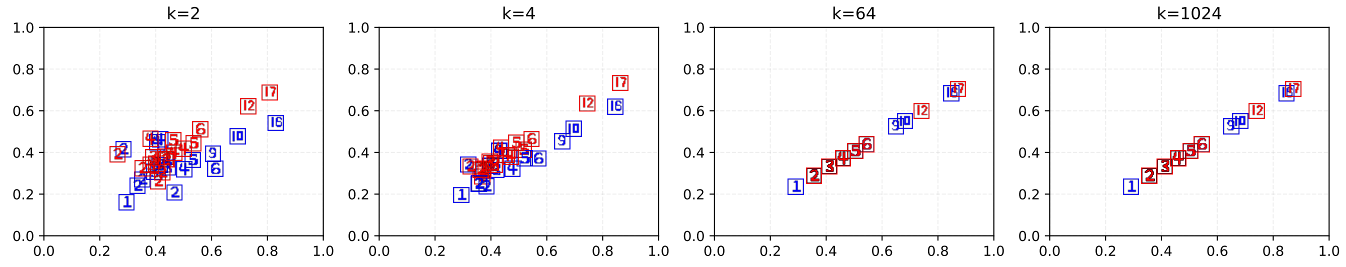
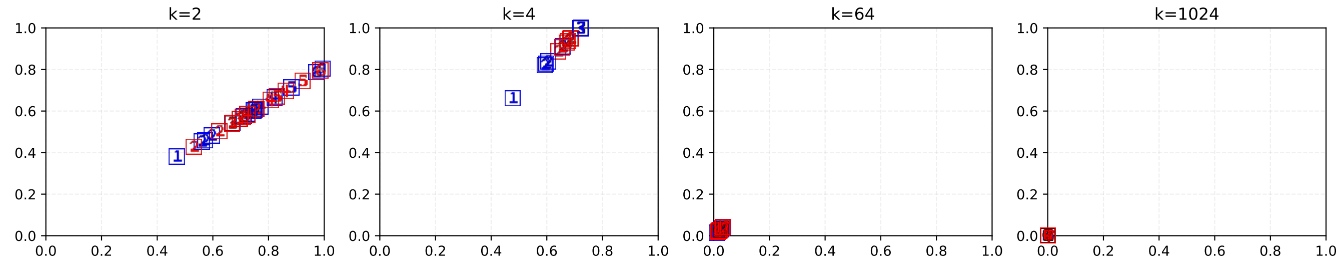
Supplemental Materials: From Cluster Assumption to Graph Convolution: Graph-based Semi-Supervised Learning Revisited



(a) SGC



(b) GCN

Fig. 5. Embedding visualization on Zachary's karate club network. Colors denote class labels, and numbers (in squares) denote node degrees.

TABLE XI
PARTS OF THE DETAILED NODE EMBEDDING RESULTS IN ZACHARY'S KARATE CLUB NETWORK.

No. (Deg.)	SGC (k=64/1024)	GCN (k=64)	GCN (k=1024)
11 (1)	$[.2904,\ .2344]$	$[.0111,\ .0137]$	$[1.09e-20, 4.39e-21]$
...	...	...	...
4 (3)	$[.4107,\ .3314]$	$[.0157,\ .0193]$	$[1.53e-20, 6.21e-21]$
10 (3)	$[.4107,\ .3314]$	$[.0157,\ .0193]$	$[1.53e-20, 6.21e-21]$
...	...	...	...
5 (4)	$[.4591,\ .3705]$	$[.0175,\ .0216]$	$[1.72e-20, 6.94e-21]$
6 (4)	$[.4591,\ .3705]$	$[.0175,\ .0216]$	$[1.72e-20, 6.94e-21]$
7 (4)	$[.4591,\ .3705]$	$[.0175,\ .0216]$	$[1.72e-20, 6.94e-21]$
...	...	...	...
0 (16)	$[.8466,\ .6832]$	$[.0323,\ .0399]$	$[3.16e-20, 1.28e-20]$
33 (17)	$[.8712,\ .7031]$	$[.0333,\ .0410]$	$[3.26e-20, 1.32e-20]$

A NUMERICAL VERIFICATION OF THEOREM 1

To verify our theoretical analysis, we conduct a numerical verification experiment on Zachary's karate club network. This graph has two classes (groups) with 34 nodes connected by 154 (undirected and unweighted) edges. As this graph has no node attributes, we randomly generate two dimensional features for each node. Then, we test SGC and GCN by varying the number of layers in these methods. Specifically, for GCN, we set the dimensions of all hidden layers to two, use ReLU activation, and adopt uniform weight initialization. At last, in these two methods, we always use the outputs of the last layer as the final node embedding results.

Figure 5 shows the visualization of the obtained node embeddings. For a better verification, we also list parts of the corresponding numerical results in Table XI. Here, we combine the results of SGC with $k = 64$ and $k = 1024$ to one column, since their results are exactly the same. We can clearly find that in both SGC and GCN, the nodes (in the same connected component) with the same degree tend to converge to the same embedding results. We also note that the convergence in GCN is less obvious than that in SGC, which may be due to the existence of many nonlinearities and network weights in GCN. In addition, we can see that the ratio of the embeddings of nodes v_i and v_j is always $\frac{\sqrt{D_{ii}+1}}{\sqrt{D_{jj}+1}}$. For example, in SGC and GCN, the ratios of node 11's embeddings and node 4's embeddings are both $\frac{\sqrt{3+1}}{\sqrt{1+1}} = \sqrt{2}$. All these findings are consistent with our Theorem 1, successfully verifying our analysis.

Algorithm 1 OGC

```
Require: Graph information ($A$ and $X$), the max iteration number $m$, and the label information of a small node set $\mathcal{V}_L$
Ensure: The label predictions
1: Initialize $U^{(0)} = X$ and $k = 0$
2: repeat
3:     $k = k + 1$
4:     Update $W$ manually or by some automatic-differentiation toolboxes (Sect. IV-A)
5:     Update $U^{(k)}$ via (lazy) supervised graph convolution (Eq. 12)
6:     Get the label predictions $\hat{Y}^{(k)}$ from: $Z^{(k)} = U^{(k)}W$
7: until $\hat{Y}^{(k)}$ converges or $k \geq m$
8: return $\hat{Y}^{(k)}$
```

Algorithm 2 GGC

```
Require: Graph information ($A$ and $X$), the max iteration number $m$, and moving out probability $\beta$
Ensure: The learned node embedding result set $\{U^{(k)} | k = 0 : m\}$
1: Initialize $U^{(0)} = X$
2: for $k = 1$ to $m$ do
3:     Get $U^{(k)}_{smo}$ via (lazy) graph convolution
4:     Get $U^{(k)}_{sharp}$ via (lazy) IGC (Eq. 16)
5:     Get $U^{(k)} = {(U^{(k)}_{smo} + U^{(k)}_{sharp})}/{2}$
6:     Decline $\beta$ with a decay factor
7: end for
8: return $\{U^{(0)}, U^{(1)}, ..., U^{(m)}\}$
```

Algorithm 3 GGCM

```
Require: Graph information ($A$ and $X$), the max iteration number $m$, and moving out probability $\beta$
Ensure: The learned multi-scale node embedding result set $\{{U_M}^{(k)} | k = 0 : m\}$
1: Initialize $U^{(0)} = X$
2: for $k = 1$ to $m$ do
3:     Get $U^{(k)}_{smo}$ via (lazy) graph convolution
4:     Get $U^{(k)}_{sharp}$ via (lazy) IGC (Eq. 16)
5:     Set $U^{(k)} = U^{(k)}_{smo}$
6:     ${U_M}^{(k)} = \alpha X + (1-\alpha)\frac{1}{k}\sum_{t=1}^{k}[{(U^{(t)}_{smo} + U^{(t)}_{sharp})}/{2}]$
7:     Decline $\beta$ with a decay factor
8: end for
9: return $\{{U_M}^{(0)}, {U_M}^{(1)}, ..., {U_M}^{(m)}\}$
```

B Derivation Details

A. Derivative of Cross-Entropy Loss

To simplify writing and differentiation, we let $Z = UW$ and $P = \mathrm{softmax}(UW)$. The cross-entropy loss of a single sample j is: $\mathcal{L}_j = -\sum_{k=1}^{c} Y_{jk} log(P_{jk})$.

First of all, we introduce the derivative of softmax operator[9]:

$$\frac{\partial P_{jk}}{\partial Z_{ji}} = \frac{\partial \frac{e^{Z_{jk}}}{\sum_{i^*=1}^{c} e^{Z_{ji^*}}}}{\partial Z_{ji}} = P_{jk}(\delta_{ki} - P_{ji}) \tag{25}$$

where δ_{ki} is the Kronecker delta:

$$\delta_{ki} = \begin{cases} 1, & \text{if } k = i; \\ 0, & \text{if } k \neq i. \end{cases}$$

Then, we can calculate the derivative of $\mathcal{L}_j$ w.r.t. Z_{ji} as follows:

$$\frac{\partial \mathcal{L}_j}{\partial Z_{ji}} = -\sum_{k=1}^{c} Y_{jk} \frac{\partial log(P_{jk})}{\partial P_{jk}} \times \frac{\partial P_{jk}}{\partial Z_{ji}} = P_{ji} - Y_{ji} \tag{26}$$

After that, the derivative of $\mathcal{L}_j$ w.r.t. U_{ji} can be obtained as follows:

$$\frac{\partial \mathcal{L}_j}{\partial U_{ji}} = \sum_{i^*=1}^{c} \frac{\partial \mathcal{L}_j}{\partial Z_{ji^*}} \times \frac{\partial Z_{ji^*}}{\partial U_{ji}} = \sum_{i^*=1}^{c} (P_{ji^*} - Y_{ji^*}) W_{ii^*} \tag{27}$$

[9]https://deepnotes.io/softmax-crossentropy

We use $\mathcal{L}_{all}$ to denote the overall loss, i.e., $\mathcal{L}_{all} = \sum_{j=1}^{n} \mathcal{L}_j$. Combining the derivatives of all the samples together, we can finally get:

$$\frac{\partial \mathcal{L}_{all}}{\partial U} = S(P - Y)W^T \tag{28}$$

where S is a diagonal matrix with $S_{jj} = 1$ if the simple j is labeled, and $S_{jj} = 0$ otherwise.

B. Derivative of Squared Loss

The squared loss of a single sample j is: $\mathcal{L}_j = \frac{1}{2} \sum_{k=1}^{c} (Y_{jk} - Z_{jk})^2$. Then, we can get the derivative of $\mathcal{L}_j$ w.r.t. U_{ji} as follows:

$$\frac{\partial \mathcal{L}_j}{\partial U_{ji}} = \sum_{i^*=1}^{c} \frac{\partial \mathcal{L}_j}{\partial Z_{ji^*}} \times \frac{\partial Z_{ji^*}}{\partial U_{ji}} = -\sum_{i^*=1}^{c} (Y_{ji^*} - Z_{ji^*}) W_{ii^*} \tag{29}$$

Considering all the samples together, we can finally get:

$$\frac{\partial \mathcal{L}_{all}}{\partial U} = S(Z - Y)W^T \tag{30}$$

where S is a diagonal matrix with $S_{jj} = 1$ if the simple j is labeled, and $S_{jj} = 0$ otherwise.