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

A Theoretical Proofs

A. Proof of Theorem 1

Proof. In SGC, as conducting graph convolution only affects the Laplacian smoothing loss (i.e., $\bar{\mathcal{Q}}_{smo}$) in Eq. 5, in the following, we only consider this loss term. In a connected component, setting $U_i^{(k)} = \sqrt{D_{ii}+1}\overrightarrow{\tau} = \sqrt{\tilde{D}_{ii}}\overrightarrow{\tau}$ and $U_j^{(k)} = \sqrt{D_{jj}+1}\overrightarrow{\tau} = \sqrt{\tilde{D}_{jj}}\overrightarrow{\tau}$ will lead the involved Laplacian smoothing loss part in this component to reach 0, where $\overrightarrow{\tau}$ is an arbitrary d-dimensional vector. Repeating this setting in every connected graph component will lead the loss $\bar{\mathcal{Q}}_{smo}$ in Eq. 5 to reach its lower bound (i.e., 0), which proves the theorem in SGC.

In GCN, the analysis is analogous to that in SGC. Specifically, we can similarly prove this theorem by setting $H_i^{(k)} = \sqrt{D_{ii}+1}\overrightarrow{\tau^{(k)}} = \sqrt{\tilde{D}_{ii}}\overrightarrow{\tau^{(k)}}$ and $H_j^{(k)} = \sqrt{D_{jj}+1}\overrightarrow{\tau^{(k)}} = \sqrt{\tilde{D}_{jj}}\overrightarrow{\tau^{(k)}}$ to reach the lower bound (i.e., 0) of $\hat{\mathcal{Q}}_{smo}^{(k)}$ defined in Eq. 10, where $\overrightarrow{\tau^{(k)}}$ is an arbitrary $d^{(k)}$-dimensional vector at the k-th iteration. □

B. Proof of Theorem 2

Proof. We can minimize the loss $\mathcal{Q}_{igc}$ in Theorem 2 by the standard gradient descent:

$$\begin{aligned} U^{(k+1)} &= U^{(k)} - \eta_{igc}\frac{\partial \mathcal{Q}_{igc}}{\partial U^{(k)}} \\ &= U^{(k)} - 2\eta_{igc}{}^{\neg}\tilde{D}^{-\frac{1}{2}}({}^{\neg}\tilde{D} - {}^{\neg}\tilde{A}){}^{\neg}\tilde{D}^{-\frac{1}{2}}U^{(k)} \end{aligned} \tag{18}$$

where η_{igc} is the learning rate in this part. If we set $\eta_{igc} = \frac{1}{2}$, we can get:

$$\begin{aligned} U^{(k+1)} &= U^{(k)} - {}^{\neg}\tilde{D}^{-\frac{1}{2}}({}^{\neg}\tilde{D} - {}^{\neg}\tilde{A}){}^{\neg}\tilde{D}^{-\frac{1}{2}}U^{(k)} \\ &= ({}^{\neg}\tilde{D}^{-\frac{1}{2}}{}^{\neg}\tilde{A}{}^{\neg}\tilde{D}^{-\frac{1}{2}})U^{(k)} \end{aligned} \tag{19}$$

The above equation is exactly the convolution equation (i.e., Eq. 16) used in IGC. Therefore, the theorem is proved. □

C. Proof of Theorem 3

Proof. First of all, we consider a "negative" regular graph whose degree is ξ. We can write its diagonal degree matrix as ${}^{\neg}D = \xi I_n$, and write its symmetrically normalized Laplacian matrix as ${}^{\neg}L_{sym} = I_n - {}^{\neg}A/\xi$ (and ${}^{\neg}A = \xi I_n - \xi {}^{\neg}L_{sym}$). According to the definition of IGC (in Eq. 16), we can get:

$$\begin{aligned} {}^{\neg}\tilde{D}^{-\frac{1}{2}}{}^{\neg}\tilde{A}{}^{\neg}\tilde{D}^{-\frac{1}{2}} &= {}^{\neg}\tilde{D}^{-\frac{1}{2}}(2I_n - {}^{\neg}A){}^{\neg}\tilde{D}^{-\frac{1}{2}} \\ &= \frac{2I_n - \xi I_n + \xi {}^{\neg}L_{sym}}{\xi + 2} \quad \text{(By replacing variables)} \\ &= Q\left(\frac{(2-\xi)I_n + \xi\,\mathrm{diag}(\boldsymbol{\lambda})}{\xi+2}\right)Q^T \end{aligned} \tag{20}$$

Therefore, w.r.t. this regular graph, we can get the following filter function:

$$\Phi(\boldsymbol{\lambda}) = \frac{2 - \xi + \xi\boldsymbol{\lambda}}{\xi + 2} \tag{21}$$

Assuming that the node degree distribution is uniform, we can get the approximation: $\xi \approx \bar{\xi} = \frac{1}{n}\sum_{i=1}^{n} {}^{\neg}D_{ii}$. Likewise, we can obtain an approximation of its filter function as a function of the average node degree by replacing ξ with $\bar{\xi}$ in the above equation, which proves the theorem.

□

D. Proof of Theorem 4

Proof. Let $\epsilon_1 \le \epsilon_2 \le \cdots \le \epsilon_n$ denote the eigenvalues of ${}^{\neg}D^{-\frac{1}{2}}{}^{\neg}A{}^{\neg}D^{-\frac{1}{2}}$ and $\omega_1 \le \omega_2 \le \cdots \le \omega_n$ denote the eigenvalues of ${}^{\neg}\tilde{D}^{-\frac{1}{2}}{}^{\neg}A{}^{\neg}\tilde{D}^{-\frac{1}{2}}$. It is easy to know $\epsilon_n = 1$. In addition, by choosing x such that $||x|| = 1$ and $y = {}^{\neg}D^{\frac{1}{2}}{}^{\neg}\tilde{D}^{-\frac{1}{2}}x$, we can get $||y||^2 = \sum_i \frac{{}^{\neg}D_{ii}}{{}^{\neg}D_{ii}+2}x_i^2$ and $\frac{\min_i {}^{\neg}D_{ii}}{2+\min_i {}^{\neg}D_{ii}} \le ||y||^2 \le \frac{\max_i {}^{\neg}D_{ii}}{2+\max_i {}^{\neg}D_{ii}}$.

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** $\hat{Y}^{(k)}$

First of all, we use the Rayleigh quotient to provide a higher bound to ω_n:

$$\begin{aligned}
\omega_n &= \max_{||x||=1} \left(x^T \neg\tilde{D}^{-\frac{1}{2}} \neg A \neg\tilde{D}^{-\frac{1}{2}} x \right) \\
&= \max_{||x||=1} \left(y^T \neg D^{-\frac{1}{2}} \neg A \neg D^{-\frac{1}{2}} y \right) \quad \text{(By replacing variable)} \\
&= \max_{||x||=1} \left(\frac{y^T \neg D^{-\frac{1}{2}} \neg A \neg D^{-\frac{1}{2}} y}{||y||^2} ||y||^2 \right) \\
&\because \max(\mathfrak{A}\mathfrak{B}) = \max(\mathfrak{A}) \max(\mathfrak{B}) \text{ if } \max(\mathfrak{A}) > 0, \forall \mathfrak{B} > 0; \\
&\text{and } \max_{||x||=1} \left({}^{y^T \neg D^{-\frac{1}{2}} \neg A \neg D^{-\frac{1}{2}} y}/_{||y||^2} \right) = \epsilon_n > 0: \\
&= \epsilon_n \max_{||x||=1} ||y||^2 \\
&\leq \frac{\max_i \neg D_{ii}}{2 + \max_i \neg D_{ii}} \quad \text{(See the properties of } ||y||^2\text{)}
\end{aligned} \tag{22}$$

Then, by rewriting matrix $\neg\tilde{D}^{-\frac{1}{2}}(\neg\tilde{D} - \neg\tilde{A})\neg\tilde{D}^{-\frac{1}{2}}$ as $I_n - \neg\tilde{D}^{-\frac{1}{2}}(2I_n - \neg A)\neg\tilde{D}^{-\frac{1}{2}}$, we can get:

$$\begin{aligned}
\neg\lambda_n &= \max_{||x||=1} x^T \left(I_n - 2\neg\tilde{D}^{-1} + \neg\tilde{D}^{-\frac{1}{2}} \neg A \neg\tilde{D}^{-\frac{1}{2}} \right) x \\
&\leq 1 - \min_{||x||=1} 2x^T \neg\tilde{D}^{-1} x + \max_{||x||=1} \left(x^T \neg\tilde{D}^{-\frac{1}{2}} \neg A \neg\tilde{D}^{-\frac{1}{2}} x \right) \\
&\because \omega_n \text{ is the largest eigenvalue of } \neg\tilde{D}^{-\frac{1}{2}} \neg A \neg\tilde{D}^{-\frac{1}{2}} \\
&= 1 - \frac{2}{2 + \max_i \neg D_{ii}} + \omega_n \\
&\leq 1 - \frac{2}{2 + \max_i \neg D_{ii}} + \frac{\max_i \neg D_{ii}}{2 + \max_i \neg D_{ii}} \\
&\text{(The higher bound of } \omega_n \text{ shown in Eq. 22)} \\
&= 2 - \frac{4}{2 + \max_i \neg D_{ii}}
\end{aligned} \tag{23}$$

□

E. Proof of Theorem 5

Proof. We can minimize the loss $\mathcal{Q}_{ggc}$ in Theorem 5 by the standard gradient descent:

$$\begin{aligned}
U^{(k+1)} &= U^{(k)} - \frac{\eta_{smo}}{2} \frac{\partial \bar{\mathcal{Q}}_{smo}}{\partial U^{(k)}} - \frac{\eta_{igc}}{2} \frac{\partial \mathcal{Q}_{igc}}{\partial U^{(k)}} \\
&= \frac{U^{(k)}}{2} - \frac{\eta_{smo}}{2} \frac{\partial \bar{\mathcal{Q}}_{smo}}{\partial U^{(k)}} + \frac{U^{(k)}}{2} - \frac{\eta_{igc}}{2} \frac{\partial \mathcal{Q}_{igc}}{\partial U^{(k)}} \\
&= \frac{U^{(k)} - \eta_{smo} \frac{\partial \bar{\mathcal{Q}}_{smo}}{\partial U^{(k)}}}{2} + \frac{U^{(k)} - \eta_{igc} \frac{\partial \bar{\mathcal{Q}}_{igc}}{\partial U^{(k)}}}{2}
\end{aligned} \tag{24}$$

where η_{smo} and η_{igc} are the learning rates. Those two parts in the last line of Eq. 24 are exactly the one half of the results of (lazy) graph convolution and IGC. Therefore, this theorem is proved. □

Algorithm 2 GGC

Require: Graph information (A and X), the max iteration number m, and moving out probability β
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 β 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 β
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 β 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:

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

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.

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

B. Derivative of Squared Loss

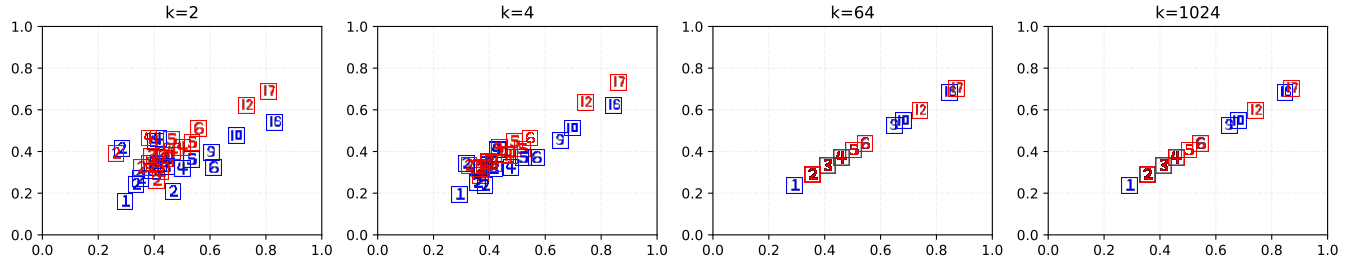
The squared loss of a single sample j is: $\mathcal{L}_j = \frac{1}{2}\sum_{k=1}^{c}(Y_{jk} - Z_{jk})^2$. Then, we can get the derivative of $\mathcal{L}_j$ w.r.t. U_{ji} as follows:

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

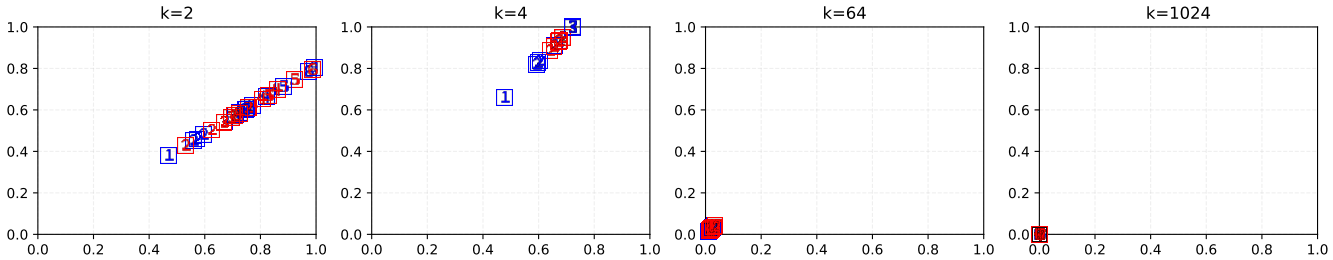
Considering all the samples together, we can finally get:

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

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



(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]	$[3.26e-20, 1.32e-20]$

C 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.