Dissecting the Diffusion Process in Linear Graph Convolutional Networks

Yifei Wang, Yisen Wang, Jiansheng Yang, Zhouchen Lin (Peking University)

https://yifeiwang77.github.io

NeurIPS 2021
Background

• Linear GCNs achieve comparable performance to nonlinear ones

• SGC (Simple Graph Convolution)
  • Given input $X$, label $Y$, adjacency matrix $A$, SGC predicts with
    \[
    \hat{Y}_{SGC} = \text{softmax}(S^KX\Theta)
    \]
  
  • 1) $K$ propagation steps (core)
    \[
    X^{(k)} \leftarrow SX^{(k-1)}, \text{ where } S = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}} \implies X^{(K)} = S^KX
    \]
  
  • 2) linear classification
    \[
    \hat{Y}_{SGC} = \text{softmax}(X^{(K)\Theta})
    \]

• advantages: memory and parameter efficiency (preprocessed features)

• disadvantages: over-smoothing, inferior performance

Equivalence between SGC and Graph Heat Equation

• Key Insight from a continuous perspective
  • SGC’s propagation = a (coarse) discretization of the graph diffusion equation

• Graph Heat Equation (GHE)
\[
\begin{cases}
\frac{dX_t}{dt} = -LX_t \\
X_0 = X
\end{cases}
\]

• where \( L = I - S \) is the graph Laplacian
Equivalence between SGC and Graph Heat Equation

• Key Insight from a continuous perspective
  • SGC’s propagation = a (coarse) discretization of the graph diffusion equation

• Graph Heat Equation (GHE)
  \[
  \begin{align*}
  \frac{dX_t}{dt} &= -LX_t \\
  X_0 &= X
  \end{align*}
  \]
  • where \( L = I - S \) is the graph Laplacian

• Discretization
  • Applying the forward Euler method with time interval \( \Delta t \)

  Euler: \[
  \hat{X}_{t+\Delta t} = \hat{X}_t - \Delta t L \hat{X}_t = \hat{X}_t - \Delta t (I - S) \hat{X}_t = [(1 - \Delta t)I + \Delta t S] \hat{X}_t
  \]

  SGC propagation: \[
  X^{(k)} \leftarrow S X^{(k-1)}
  \]
  • Thus, SGC is the Euler discretization of GHE with step size \( \Delta t = 1 \)
Revealing SGC’s Fundamental Limitations

- Limitations
  - 1. Oversmoothing (asymptotic)
    - SGC will oversmooth with increasing propagation steps $K = T \to \infty$
    - We provide a continuous characterization of this phenomenon

**Theorem 1** (Oversmoothing from a spectral view). Assume that the eigendecomposition of the Laplacian matrix as $L = \sum_{i=1}^{n} \lambda_{i} u_{i} u_{i}^{\top}$, with eigenvalues $\lambda_{i}$ and eigenvectors $u_{i}$. Then, the heat equation (Eq. (4)) admits a closed-form solution at time $t$, known as the heat kernel $H_{t} = e^{-tL} = \sum_{i=1}^{n} e^{-\lambda_{i}t} u_{i} u_{i}^{\top}$. As $t \to \infty$, $H_{t}$ asymptotically converges to a non-informative equilibrium as $t \to \infty$, due to the non-trivial (i.e., positive) eigenvalues vanishing:

$$\lim_{t \to \infty} e^{-\lambda_{i}t} = \begin{cases} 0, & \text{if } \lambda_{i} > 0 \\ 1, & \text{if } \lambda_{i} = 0, \ i = 1, \ldots, n. \end{cases}$$ (7)
Revealing SGC’s Fundamental Limitations

• Limitations
  • 1. Oversmoothing (asymptotic)
  • 2. Numerical Error
    • Consequence by adopting a fixed time interval $\Delta t = 1$

**Theorem 2** (Numerical errors). *For the initial value problem in Eq. (4) with finite terminal time $T$, the numerical error of the forward Euler method in Eq. (5) with $K$ steps can be upper bounded by*

$$
\|e_T^{(K)}\| \leq \frac{T\|L\|\|X_0\|}{2K} \left(e^{T\|L\|} - 1\right).
$$

• As $T=K$, the upper bound reduces to $c \cdot \left(e^{T\|L\|} - 1\right)$
• The numerical error increases exponentially with more propagation steps $K=T$
Revealing SGC’s Fundamental Limitations

• Limitations
  
  • 1. Oversmoothing (asymptotic)
  • 2. Numerical Error
  • 3. Learning Risks

  • The two above issues will finally lead to a large learning risk

Theorem 3 (Learning risks). Consider a simple linear regression problem \((X, Y)\) on graph, where the observed input features \(X\) are generated by corrupting the ground truth features \(X_c\) with the following inverse graph diffusion with time \(T^*\):

\[
\frac{d\tilde{X}_t}{dt} = L\tilde{X}_t, \quad \text{where} \quad \tilde{X}_0 = X_c \quad \text{and} \quad \tilde{X}_{T^*} = X.
\] (9)

Denote the population risk with ground truth features as \(R(W) = \mathbb{E} \|Y - X_cW\|_2^2\) and that of Euler method applied input \(X\) (Eq. (5)) as \(\hat{R}(W) = \mathbb{E} \|Y - [S(\Delta t)]^K XW\|_2^2\). Supposing that \(\mathbb{E}\|X_c\|_2^2 = M < \infty\), we have the following upper bound:

\[
\hat{R}(W) \leq R(W) + \|W\|_2 \left( M \left\| e^{T^*L} \right\|_2^2 \left\| e^{-T^*L} - e^{-\hat{T}L} \right\|_2^2 + \mathbb{E} \left\| e^{(K)} \right\|_2^2 \right).
\] (10)

To minimize the risk, we need

1) the optimal terminal time
2) minimized numerical errors
Revealing SGC’s Fundamental Limitations

• Limitations
  • 1. Oversmoothing (asymptotic)
  • 2. Numerical Error
  • 3. Learning Risks
    • The two above issues will finally lead to a large learning risk

**Theorem 3** (Learning risks). Consider a simple linear regression problem \((X, Y)\) on graph, where the observed input features \(X\) are generated by corrupting the ground truth features \(X_c\) with the following inverse graph diffusion with time \(T^*\):

\[
\frac{d\tilde{X}_t}{dt} = L\tilde{X}_t, \quad \text{where } \tilde{X}_0 = X_c \text{ and } \tilde{X}_{T^*} = X. \quad (9)
\]

Denote the population risk with ground truth features as \(R(W) = \mathbb{E}\|Y - X_cW\|^2\) and that of Euler method applied input \(X\) (Eq. (5)) as \(\hat{R}(W) = \mathbb{E}\|Y - [S^{(\Delta t)}]^K XW\|^2\). Supposing that \(\mathbb{E}\|X_c\|^2 = M < \infty\), we have the following upper bound:

\[
\hat{R}(W) \leq R(W) + \|W\|^2 \left( M \left\| e^{T^*L} \right\|^2 \left\| e^{-T^*L} - e^{-\hat{T}L} \right\|^2 + \mathbb{E} \left\| e^{(K')} \right\|^2 \right). \quad (10)
\]

To minimize the risk, we need:

1) the optimal terminal time \(T^*\)
2) minimized numerical errors

**Ideal:** real-value

**SGC:** integer
Revealing SGC’s Fundamental Limitations

• Limitations
  • 1. Oversmoothing (asymptotic)
  • 2. Numerical Error
  • 3. Learning Risks
    • The two above issues will finally lead to a large learning risk

**Theorem 3 (Learning risks).** Consider a simple linear regression problem \((\mathbf{X}, \mathbf{Y})\) on graph, where the observed input features \(\mathbf{X}\) are generated by corrupting the ground truth features \(\mathbf{X}_c\) with the following inverse graph diffusion with time \(T^*\):

\[
\frac{d\tilde{\mathbf{X}}_t}{dt} = \mathbf{L}\tilde{\mathbf{X}}_t, \text{ where } \tilde{\mathbf{X}}_0 = \mathbf{X}_c \text{ and } \tilde{\mathbf{X}}_{T^*} = \mathbf{X}.
\]

Denote the population risk with ground truth features as \(R(\mathbf{W}) = \mathbb{E}\|\mathbf{Y} - \mathbf{X}_c\mathbf{W}\|^2\) and that of Euler method applied input \(\mathbf{X}\) (Eq. (5)) as \(\hat{R}(\mathbf{W}) = \mathbb{E}\|\mathbf{Y} - [\mathbf{S}^{(\Delta t)}]^{K}\mathbf{X}\mathbf{W}\|^2\). Supposing that \(\mathbb{E}\|\mathbf{X}_c\|^2 = M < \infty\), we have the following upper bound:

\[
\hat{R}(\mathbf{W}) \leq R(\mathbf{W}) + \|\mathbf{W}\|^2 \left( M \left\| e^{T^*\mathbf{L}} \right\|^2 \left\| e^{-T^*\mathbf{L}} - e^{-\hat{T}^*\mathbf{L}} \right\|^2 + \mathbb{E}\left\| e^{(K^*)\mathbf{L}} \right\|^2 \right) .
\]

To minimize the risk, we need

1) the optimal terminal time
2) minimized numerical errors

**Ideal:** \(\Delta t \rightarrow 0\)

SGC: fixed step size \(\Delta t = 1\)
A Simple Fix to All These Limitations!

Decoupling $T$ (terminal time) and $K$ (propagation steps)

• We take $K$ and $T$ as two free parameters
  • 1. Flexibly choose $T$ (real-valued) for an optimal tradeoff of smoothing
  • 2. Given a fixed optimal $T^*$, we can increase $K$ for better precision without oversmoothing
A Simple Fix to All These Limitations!

**Decoupling T (terminal time) and K (propagation steps)**

- We take K and T as two free parameters
  - 1. Flexibly choose T (real-valued) for an optimal tradeoff of smoothing
  - 2. Given a **fixed** optimal T*, we can increase K for better precision without oversmoothing
- Decoupled Graph Convolution (DGC)
  \[
  \hat{Y}_{\text{DGC}} = \text{softmax}\left(\hat{X}_T \Theta\right), \text{ where } \hat{X}_T = \text{ode}_{\text{int}}(X, \Delta t, K)
  \]
  - where \text{ode}_{\text{int}}(X, \Delta t, K) denotes the numerical integration with step size \Delta t for K steps
A Simple Fix to All These Limitations!

**Decoupling** $T$ (terminal time) and $K$ (propagation steps)

- We take $K$ and $T$ as two free parameters
  - 1. Flexibly choose $T$ (real-valued) for an optimal tradeoff of smoothing
  - 2. Given a **fixed** optimal $T^*$, we can increase $K$ for better precision without oversmoothing

- Decoupled Graph Convolution (DGC)
  \[
  \hat{Y}_{\text{DGC}} = \text{softmax}(\hat{X}_T \Theta), \text{ where } \hat{X}_T = \text{ode}_{\text{int}}(X, \Delta t, K)
  \]
  
  - where $\text{ode}_{\text{int}}(X, \Delta t, K)$ denotes the numerical integration with step size $\Delta t$ for $K$ steps

- DGC-Euler with forward Euler scheme and step size $\Delta t = T/K$
  \[
  \hat{X}_T = \left[S^{(T/K)}\right]^K X, \text{ where } S^{(T/K)} = (1 - T/K) \cdot I + (T/K) \cdot S
  \]
A Simple Fix to All These Limitations!

**Decoupling** $T$ (terminal time) and $K$ (propagation steps)

- We take $K$ and $T$ as two free parameters
  - 1. Flexibly choose $T$ (real-valued) for an optimal tradeoff of smoothing
  - 2. Given a **fixed** optimal $T^*$, we can increase $K$ for better precision without oversmoothing

- Decoupled Graph Convolution (DGC)
  
  $$\hat{Y}_{\text{DGC}} = \text{softmax}\left(\hat{X}_T\Theta\right), \text{ where } \hat{X}_T = \text{ode}_{\text{int}}(X, \Delta t, K)$$

  - where $\text{ode}_{\text{int}}(X, \Delta t, K)$ denotes the numerical intergration with step size $\Delta t$ for $K$ steps

- DGC-Euler with forward Euler scheme and step size $\Delta t = T/K$

- DGC-RK with the 4\textsuperscript{th}-order Runge-Kutta (RK) method
  
  $$\hat{X}_{t+\Delta t} = \hat{X}_t + \frac{1}{6} \Delta t (R_1 + 2R_2 + 2R_3 + R_4) \triangleq S_{\text{RK}}^{(\Delta t)} \hat{X}_t$$
Verifying the Benefits of DGC

• Theoretical Benefits
  • Comparing SGC to DGC

<table>
<thead>
<tr>
<th>Aspects</th>
<th>SGC (Wu et al. 2019)</th>
<th>DGC-Euler (ours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asymptotic</td>
<td>Over-smoothing as T=K</td>
<td>A fixed T with optimal tradeoff</td>
</tr>
<tr>
<td>Numerical error</td>
<td>Exponentially large when K increases</td>
<td>With fixed T, increasing K leads to smaller numerical error</td>
</tr>
<tr>
<td>Learning Risk</td>
<td>Deviation from optimal T + Large numerical error</td>
<td>Reach optimal real-valued T + minimized numerical error with large K</td>
</tr>
</tbody>
</table>
Verifying the Benefits of DGC

- Theoretical Benefits
- Empirical Evidence

**T:** Either a smaller $T$ or a larger $T$ mixes the features up. An **optimal $T$** implies better separable features.

**K:** With fixed optimal $T$, too large step size $\Delta t$ (small $K$) leads to feature collapse, and **large $K$** makes features separable!

Figure 1: Input feature visualization of our DGC-Euler model with t-SNE [19] on the Cora dataset. Each point represents a node in the graph and its color denotes the class of the node.
Experiments

- Performance on Semi-supervised Node Classification

<table>
<thead>
<tr>
<th>Type</th>
<th>Method</th>
<th>Cora</th>
<th>Citeseer</th>
<th>Pubmed</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>81.5</td>
<td>70.3</td>
<td>79.0</td>
</tr>
<tr>
<td>Non-linear</td>
<td>GCN [8]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>GAT [20]</td>
<td>83.0 ± 0.7</td>
<td>72.5 ± 0.7</td>
<td>79.0 ± 0.3</td>
</tr>
<tr>
<td></td>
<td>GraphSAGE [6]</td>
<td>82.2</td>
<td>71.4</td>
<td>75.8</td>
</tr>
<tr>
<td></td>
<td>JKNet [26]</td>
<td>81.1</td>
<td>69.8</td>
<td>78.1</td>
</tr>
<tr>
<td></td>
<td>APPNP [9]</td>
<td>83.3</td>
<td>71.8</td>
<td>80.1</td>
</tr>
<tr>
<td></td>
<td>GWNN [25]</td>
<td>82.8</td>
<td>71.7</td>
<td>79.1</td>
</tr>
<tr>
<td></td>
<td>GraphHeat [24]</td>
<td>83.7</td>
<td>72.5</td>
<td>80.5</td>
</tr>
<tr>
<td></td>
<td>CGNN [23]</td>
<td>84.2 ± 0.6</td>
<td>71.8 ± 0.7</td>
<td>76.8 ± 0.6</td>
</tr>
<tr>
<td></td>
<td>GCDE [16]</td>
<td>83.8 ± 0.5</td>
<td>72.5 ± 0.5</td>
<td>79.9 ± 0.3</td>
</tr>
<tr>
<td></td>
<td>Label Propagation [29]</td>
<td>45.3</td>
<td>68.0</td>
<td>63.0</td>
</tr>
<tr>
<td>Linear</td>
<td>DeepWalk [15]</td>
<td>70.7 ± 0.6</td>
<td>51.4 ± 0.5</td>
<td>76.8 ± 0.6</td>
</tr>
<tr>
<td></td>
<td>SGC [22]</td>
<td>81.0 ± 0.0</td>
<td>71.9 ± 0.1</td>
<td>78.9 ± 0.0</td>
</tr>
<tr>
<td></td>
<td>SGC-PairNorm [28]</td>
<td>81.1</td>
<td>70.6</td>
<td>78.2</td>
</tr>
<tr>
<td></td>
<td>DGC (ours)</td>
<td>83.5 ± 0.0</td>
<td>74.5 ± 0.2</td>
<td>80.2 ± 0.1</td>
</tr>
</tbody>
</table>
Experiments

• Performance on Semi-supervised Node Classification

Table 2: Test accuracy (%) of semi-supervised node classification on citation networks.

<table>
<thead>
<tr>
<th>Type</th>
<th>Method</th>
<th>Cora</th>
<th>Citeseer</th>
<th>Pubmed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-linear</td>
<td>GCN [8]</td>
<td>81.5</td>
<td>70.3</td>
<td>79.0</td>
</tr>
<tr>
<td></td>
<td>GAT [20]</td>
<td>83.0 ± 0.7</td>
<td>72.5 ± 0.7</td>
<td>79.0 ± 0.3</td>
</tr>
<tr>
<td></td>
<td>GraphSAGE [6]</td>
<td>82.2</td>
<td>71.4</td>
<td>75.8</td>
</tr>
<tr>
<td></td>
<td>JKNet [26]</td>
<td>81.1</td>
<td>69.8</td>
<td>78.1</td>
</tr>
<tr>
<td></td>
<td>APPNP [9]</td>
<td>83.3</td>
<td>71.8</td>
<td>80.1</td>
</tr>
<tr>
<td></td>
<td>GWNN [25]</td>
<td>71.7</td>
<td>79.1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SGC</td>
<td>72.5</td>
<td>80.5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SGC-Pooling [29]</td>
<td>71.8 ± 0.7</td>
<td>76.8 ± 0.6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DeepWalk [15]</td>
<td>70.7 ± 0.6</td>
<td>51.4 ± 0.5</td>
<td>76.8 ± 0.6</td>
</tr>
<tr>
<td>Linear</td>
<td>SGC [22]</td>
<td>81.0 ± 0.0</td>
<td>71.9 ± 0.1</td>
<td>78.9 ± 0.0</td>
</tr>
<tr>
<td></td>
<td>SGC-PairNorm [28]</td>
<td>81.1</td>
<td>70.6</td>
<td>78.2</td>
</tr>
<tr>
<td></td>
<td>DGC (ours)</td>
<td>83.5 ± 0.0</td>
<td>74.5 ± 0.2</td>
<td>80.2 ± 0.1</td>
</tr>
</tbody>
</table>

Improves SGC by a large margin
Experiments

- Performance on Semi-supervised Node Classification

<table>
<thead>
<tr>
<th>Type</th>
<th>Method</th>
<th>Cora</th>
<th>Citeeseer</th>
<th>Pubmed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-linear</td>
<td>GCN [8]</td>
<td>81.5</td>
<td>70.3</td>
<td>79.0</td>
</tr>
<tr>
<td></td>
<td>GAT [20]</td>
<td>83.0 ± 0.7</td>
<td>72.5 ± 0.7</td>
<td>79.0 ± 0.3</td>
</tr>
<tr>
<td></td>
<td>GraphSAGE [6]</td>
<td>82.2</td>
<td>71.4</td>
<td>75.8</td>
</tr>
<tr>
<td></td>
<td>JKNNet [26]</td>
<td>81.1</td>
<td>69.8</td>
<td>78.1</td>
</tr>
<tr>
<td></td>
<td>APPNP [9]</td>
<td>83.3</td>
<td>71.8</td>
<td>80.1</td>
</tr>
<tr>
<td></td>
<td>GWWN [25]</td>
<td></td>
<td>71.7</td>
<td>79.1</td>
</tr>
<tr>
<td></td>
<td>GPC-Stacking [29]</td>
<td>76.0</td>
<td>71.8 ± 0.7</td>
<td>76.8 ± 0.6</td>
</tr>
<tr>
<td></td>
<td>DeepWalk [15]</td>
<td>70.7 ± 0.6</td>
<td>51.4 ± 0.5</td>
<td>76.8 ± 0.6</td>
</tr>
<tr>
<td>Linear</td>
<td>SGC [22]</td>
<td>81.0 ± 0.0</td>
<td>71.9 ± 0.1</td>
<td>78.9 ± 0.0</td>
</tr>
<tr>
<td></td>
<td>SGC-PairNorm [28]</td>
<td>81.1</td>
<td>70.6</td>
<td>78.2</td>
</tr>
<tr>
<td></td>
<td>DGC (ours)</td>
<td>83.5 ± 0.0</td>
<td>74.5 ± 0.2</td>
<td>80.2 ± 0.1</td>
</tr>
</tbody>
</table>

Comparable to SOTA nonlinear GCNs!

Implements SGC by a large margin
Experiments

- Performance on Semi-supervised Node Classification
- Performance on Fully-supervised Node Classification

Table 3: Test accuracy (%) of fully-supervised node classification on citation networks.

<table>
<thead>
<tr>
<th>Type</th>
<th>Method</th>
<th>Cora</th>
<th>Citeseer</th>
<th>Pubmed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-linear</td>
<td>GCN [8]</td>
<td>85.77</td>
<td>73.58</td>
<td>88.13</td>
</tr>
<tr>
<td></td>
<td>GAT [20]</td>
<td>86.37</td>
<td>74.32</td>
<td>87.62</td>
</tr>
<tr>
<td></td>
<td>JK-MaxPool [26]</td>
<td>89.6</td>
<td>77.7</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>JK-Concat [26]</td>
<td>89.1</td>
<td>78.3</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>JK-LSTM [26]</td>
<td>85.8</td>
<td>74.7</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>APPNP [9]</td>
<td>90.21</td>
<td>79.8</td>
<td>86.29</td>
</tr>
<tr>
<td>Linear</td>
<td>SGC [22]</td>
<td>85.82</td>
<td>78.08</td>
<td>83.27</td>
</tr>
<tr>
<td></td>
<td>DGC (ours)</td>
<td><strong>88.2 ± 0.1</strong></td>
<td><strong>79.0 ± 0.2</strong></td>
<td><strong>88.7 ± 0.0</strong></td>
</tr>
</tbody>
</table>
Experiments

• Performance on Semi-supervised Node Classification
• Performance on Fully-supervised Node Classification
• Performance on Large Scale Datasets

Table 3: Test accuracy (%) of fully-supervised node classification on citation networks. OOM: out of memory.

<table>
<thead>
<tr>
<th>Type</th>
<th>Method</th>
<th>Cora</th>
<th>Citeseer</th>
<th>Pubmed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-linear</td>
<td>GCN [8]</td>
<td>85.77</td>
<td>73.58</td>
<td>88.13</td>
</tr>
<tr>
<td></td>
<td>GAT [20]</td>
<td>86.37</td>
<td>74.32</td>
<td>87.62</td>
</tr>
<tr>
<td></td>
<td>JK-MaxPool [26]</td>
<td>89.6</td>
<td>77.7</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>JK-Concat [26]</td>
<td>89.1</td>
<td>78.3</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>JK-LSTM [26]</td>
<td>85.8</td>
<td>74.7</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>APPNP [9]</td>
<td>90.21</td>
<td>79.8</td>
<td>86.29</td>
</tr>
<tr>
<td>Linear</td>
<td>SGC [22]</td>
<td>85.82</td>
<td>78.08</td>
<td>83.27</td>
</tr>
<tr>
<td></td>
<td><strong>DGC (ours)</strong></td>
<td><strong>88.2 ± 0.1</strong></td>
<td><strong>79.0 ± 0.2</strong></td>
<td><strong>88.7 ± 0.0</strong></td>
</tr>
</tbody>
</table>

Table 4: Test accuracy (%) comparison with inductive methods on a large scale dataset, Reddit. Reported results are averaged over 10 runs. OOM: out of memory.

<table>
<thead>
<tr>
<th>Type</th>
<th>Method</th>
<th>Acc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-linear</td>
<td>GCN [8]</td>
<td>OOM</td>
</tr>
<tr>
<td></td>
<td>FastGCN [3]</td>
<td>93.7</td>
</tr>
<tr>
<td></td>
<td>GraphSAGE-GCN [6]</td>
<td>93.0</td>
</tr>
<tr>
<td></td>
<td>GraphSAGE-mean [6]</td>
<td>95.0</td>
</tr>
<tr>
<td></td>
<td>GraphSAGE-LSTM [6]</td>
<td>95.4</td>
</tr>
<tr>
<td></td>
<td>APPNP [9]</td>
<td>95.0</td>
</tr>
<tr>
<td>Linear</td>
<td>RandDGI [21]</td>
<td>93.3</td>
</tr>
<tr>
<td></td>
<td>SGC [22]</td>
<td>94.9</td>
</tr>
<tr>
<td></td>
<td><strong>DGC (ours)</strong></td>
<td><strong>95.8</strong></td>
</tr>
</tbody>
</table>
Experiments

• Empirical Understandings of DGC
  • Left: over-smoothing with increasing steps
  • Middle: robustness to feature noise
  • Right: computation time
Experiments

• Empirical Understandings of DGC
  • Left: graph Laplacian
  • Middle: numerical scheme
  • Right: terminal time
Takeaways

- The diffusion process can be understood through continuous PDEs
- This perspective inspires us to design more accurate and robust (linear) GCNs by simply decoupling $T$ and $K$
- A properly designed linear GCN is comparable to SOTA nonlinear ones
- We should propose new alternatives that can truly benefit from nonlinear architectures
Thanks!

Q & A

Find more stuff about this work at https://yifeiwang77.github.io/
Contact:
yifei_wang AT pku.edu.cn; yisen.wang AT pku.edu.cn