Recitation: Graph Neural Networks

- Quickly review GCN message passing process
- Graph Convolution layer forward
- Graph Convolution layer backward
- GCN code example
A single layer of GNN: Graph Convolution

Key idea: Node’s neighborhood defines a computation graph

- Learning a node feature by propagating and aggregating neighbor information!
- Node embedding can be defined by local network neighborhoods!
A single layer of GNN: Graph Convolution

Key idea: Generate node embedding based on local network neighborhoods

Considering 1 step of feature aggregation of the nearest neighbor

Processing information from A, C, D
A single layer of GNN: Graph Convolution

Key idea: Generate node embedding based on local network neighborhoods

Considering 1 step of feature aggregation of the nearest neighbor

Processing information from A, C, D

Now B have the information from its first nearest neighbors
A single layer of GNN: Graph Convolution

Key idea: Generate node embedding based on local network neighborhoods

Considering 1 step of feature aggregation of the nearest neighbor

Processing information from A, B, C, D

Also we don’t want to lose information from B itself
A single layer of GNN: Graph Convolution

Key idea: Generate node embedding based on local network neighborhoods

Considering 2 steps of feature aggregation of the nearest neighbor

Now B have the information from its first and second nearest neighbors
A single layer of GNN: Graph Convolution

Key idea: Generate node embedding based on local network neighborhoods

How to process and mix the information from neighbor?
A single layer of GNN: Graph Convolution

Key idea: Generate node embedding based on local network neighborhoods

How to process and mix the information from neighbor?

Apply Neural Networks
sum, product, mean, max, min etc.
A single layer of GNN: Graph Convolution

Key idea: Generate node embedding based on local network neighborhoods

How to process and mix the information from neighbor?

Apply Neural Networks

Mean (Traditional Graph Convolutional Neural Networks (GCN))

[Kipf and Welling, ICLR 2017]
A single layer of GNN: Graph Convolution

Key idea: Generate node embedding based on local network neighborhoods

During a single Graph Convolution layer, we apply the feature aggregation to every node in the graph at the same time (T)

Target node

Apply Neural Networks

Mean (Traditional Graph Convolutional Neural Networks (GCN))

[Kipf and Welling, ICLR 2017]
A single layer of GNN: Graph Convolution-Forward

Math for a single layer of graph convolution

\[
h^0_v = x_v
\]

\[
h^{t+1}_v = \sigma \left( \sum_{u \in N(v)} \frac{h^t_u}{|N(v)|} \right) + B \cdot h^t_v, \quad \forall t \in (0, \ldots, T-1)
\]

Node v feature at time(layer) \(t+1\)

Learnable weight

Non-linear activation (i.e. relu())

Average the neighbor node feature at time(layer) \(t\)

Learnable weight

Node v feature at time(layer) \(t\)

0\textsuperscript{th} layer embedding = node v initial feature

Number of time(layers)

Neural Networks

Mean

Node v feature at time(layer) \(t+1\)
A single layer of GNN: Graph Convolution-Forward

Matrix form for a single layer of graph convolution

\[
\begin{align*}
    h^{t+1}_v &= \sigma \left( W_k \sum_{u \in N(v)} \frac{h^t_u}{|N(v)|} + B_k h^t_v \right), \forall t \in (0, \ldots, T - 1)
\end{align*}
\]

We stack multiple \( h^t_v(1 \times F) \) together into \( H^t(N \times F) \)
A single layer of GNN: Graph Convolution-Forward

Matrix form for a single layer of graph convolution

\[
    h_{v}^{t+1} = \sigma \left( W_k \sum_{u \in N(v)} \frac{h_{u}^{t}}{|N(v)|} + B_k h_{v}^{t} \right), \forall t \in (0, \ldots, T - 1)
\]
A single layer of GNN: Graph Convolution-Forward

Matrix form for a single layer of graph convolution

\[
h_{v}^{t+1} = \sigma \left( W_k \sum_{u \in N(v)} \frac{h_u^t}{|N(v)|} + B_k h_v^t \right), \forall t \in (0, \ldots, T - 1)
\]

Noted that \( W^T \) is a learnable weight matrix.
A single layer of GNN: Graph Convolution-Forward

Matrix form for a single layer of graph convolution

\[
(1 \times F) \quad h_{v}^{t+1} = \sigma \left( W_k \sum_{u \in N(v)} \frac{h_{u}^{t}}{|N(v)|} + B_k h_{v}^{t} \right), \forall t \in (0, \ldots, T - 1)
\]

Why put \( W^T \) on the right hand site of \( H^t \)? Why not left? With a shape of \((N \times N)\)?
A single layer of GNN: Graph Convolution-Forward

Matrix form for a single layer of graph convolution

\[
h_v^{t+1} = \sigma \left( W_k \sum_{u \in N(v)} \frac{h_u^t}{|N(v)|} + B_k h_v^t \right), \forall t \in (0, \ldots, T - 1)
\]

What happen if we still put W on the left hand site?

Like this?

Seems like nothing goes wrong, the result matrix shape is still \((N \times F)\)?

What happen if we still put W on the left hand site?
A single layer of GNN: Graph Convolution-Forward

Matrix form for a single layer of graph convolution

\[
W_k \left( \sum_{u \in N(v)} \frac{h_u^t}{|N(v)|} \right) + B_k h_v^t, \forall t \in (0, \ldots, T - 1)
\]

\[
h_v^{t+1} = \sigma \left( W_k \left( \sum_{u \in N(v)} \frac{h_u^t}{|N(v)|} \right) + B_k h_v^t \right)
\]

Seems like nothing goes wrong, the result matrix shape is still \((N \times F)\)?

No, it’s wrong, because we are still mixing information among different nodes, which has the same function with adjacent matrix, feature within node does not receive any mixing.
A single layer of GNN: Graph Convolution-Forward

Matrix form for a single layer of graph convolution

\[
h^{t+1}_v = \sigma \left( W_k \sum_{u \in N(v)} \frac{h^t_u}{|N(v)|} + B_k h^t_v \right) + B_k h^t_v, \quad \forall t \in (0, \ldots, T - 1)
\]

Learnable weight is used to mix information along the feature within a single node

W term should be on the right hand site!
A single layer of GNN: Graph Convolution-Forward

Matrix form for a single layer of graph convolution

\[
h_v^{t+1} = \sigma \left( W_k \sum_{u \in N(v)} \frac{h_u^t}{|N(v)|} + B_k h_v^t \right) + 1, \forall t \in (0, \ldots, T - 1)
\]
A single layer of GNN: Graph Convolution-Forward

Matrix form for a single layer of graph convolution

\[
h^t_v + 1 = \sigma \left( W_k \sum_{u \in N(v)} \frac{h^t_u}{|N(v)|} + B_k h^t_v \right), \forall t \in (0, \ldots, T - 1)
\]

Noted that \(B^T\) is a learnable weight matrix

Self loop adjacent matrix is a diagonal matrix!
A single layer of GNN: Graph Convolution-Forward

Matrix form for a single layer of graph convolution

\[
\begin{aligned}
    h_v^{t+1} &= \sigma \left( W_k \sum_{u \in N(v)} \frac{h_u^t}{|N(v)|} + B_k h_v^t \right), \forall t \in (0, \ldots, T-1)
\end{aligned}
\]

Now let’s rewrite the scalar form above into matrix form

\[
\begin{aligned}
    H^{t+1} &= \sigma \left( D^{-1} A H^t W^T + A' H_v^t B^T \right)
\end{aligned}
\]

- Non-Linear Activation
- Aggregating neighbor node feature
- Aggregating self node feature

A single layer of GNN: Graph Convolution-Forward
Matrix form for a single layer of graph convolution

\[
h_{v}^{t+1} = \sigma \left( W_k \sum_{u \in N(v)} \frac{h_{u}^{t}}{|N(v)|} + B_k h_{v}^{t} \right), \forall t \in (0, \ldots, T - 1)
\]

\[
H^{t+1} = \sigma \left( D^{-1} \tilde{A} H^{t'} W'^T \right)
\]
A single layer of GNN: Graph Convolution-Forward

Matrix form for a single layer of graph convolution

\[ h_v^{t+1} = \sigma \left( W_k \sum_{u \in N(v)} \frac{h_u^t}{|N(v)|} + B_k h_v^t \right), \forall t \in (0, \ldots, T - 1) \]

\[ H^{t+1} = \sigma \left( D^{-1} \tilde{A} H^t \hat{W}^T \right) \]

Forward equation for GCN
A single layer of GNN: Graph Convolution-Backward
\[ H^{+1} = 6 \left( \frac{d}{dH} A H^T W^T \right) \]

\[ Y = X W \]

\[ \frac{\partial L}{\partial Y} \text{ known, } \frac{\partial L}{\partial x}, \frac{\partial L}{\partial w} ? \]

\[ \frac{\partial L}{\partial x} = \frac{\partial L}{\partial Y} W^T = \frac{\partial L}{\partial Y} X W^T \]

\[ \frac{\partial L}{\partial w} ? \]

\[ Y^T = (XW)^T \Rightarrow Y^T = W^T X^T \]

\[ \frac{\partial L}{\partial w} = \frac{\partial L}{\partial Y} W^T \theta^T = \frac{\partial L}{\partial Y} X \theta^T \]

\[ \frac{\partial L}{\partial w} = \left( \frac{\partial L}{\partial w^T} \right)^T = \left( \frac{\partial L}{\partial w} \right)^T = \left( \frac{\partial L}{\partial Y^T} X \right)^T \]
\[
H_{\text{th}}^{\text{th}} = \theta (D^{-1} A H^T W)_{\text{th}}^{\text{th}}
\]

Let \( \tilde{H} = D^{-1} A H^T W \)

Let \( H_0 = H^T W^T \)

\[
\frac{\partial L}{\partial H_{\text{th}}} = \frac{\partial L}{\partial \tilde{H}} \frac{\partial \tilde{H}}{\partial H_{\text{th}}}
\]

\[
\frac{\partial L}{\partial \tilde{H}} = \frac{\partial L}{\partial H^T W^T} \frac{\partial H^T W^T}{\partial \tilde{H}}
\]

\[
\frac{\partial L}{\partial H^T W^T} = \frac{\partial L}{\partial \tilde{H}} \frac{\partial \tilde{H}}{\partial H^T W^T} 
\]

\[
\frac{\partial L}{\partial \tilde{H}} = \frac{\partial L}{\partial H^T W^T} \frac{\partial H^T W^T}{\partial \tilde{H}}
\]

\[
\frac{\partial L}{\partial H^T} = (\frac{\partial L}{\partial H} \times (D^{-1} A))^T \times W^T 
\]

\[
\frac{\partial L}{\partial W^T} = (\frac{\partial L}{\partial H^T} \times (D^{-1} A))^T \times H^T
\]