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 it’s 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)

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

\[
h_v^0 = x_v \quad (1 \times F)
\]

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

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

Number of time(layers)

0th layer embedding = node v initial feature

Neural Networks

Mean
A single layer of GNN: Graph Convolution-Forward

Matrix form for a single layer of graph convolution

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

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

Matrix form for a single layer of graph convolution

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

\[
D^{-1}(N \times N) = \begin{bmatrix}
1 & \cdots & 0 \\
\vdots & \ddots & \vdots \\
0 & \cdots & 1/|N(v)|
\end{bmatrix}
\]

\[
A(N \times N) \times \begin{bmatrix}
1 & \cdots & 0 \\
\vdots & \ddots & \vdots \\
0 & \cdots & 1/|N(v)|
\end{bmatrix} \times N
\]

\[
H^{t}(N \times F)
\]
A single layer of GNN: Graph Convolution-Forward

Matrix form for a single layer of graph convolution

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

\[
D^{-1}(N \times N)
\]

\[
\begin{bmatrix}
\frac{1}{|N(v)|} & \cdots & 0 \\
\vdots & \ddots & \vdots \\
0 & \cdots & \frac{1}{|N(v)|}
\end{bmatrix}
\]

\[
A(N \times N)
\]

\[
H^t(N \times F)
\]

\[
W^T(F \times F)
\]

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

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

Why put \( W^T \) on the right hand side 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

\[
(1 \times F)
\begin{pmatrix}
W_k \\
\sum_{u \in N(v)} \frac{h^t_u}{|N(v)|}
\end{pmatrix}
\begin{pmatrix}
1 \\
N(v)
\end{pmatrix}
(1 \times F)
+ B_k h^t_v
, \forall t \in (0, \ldots, T - 1)
\]

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

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

\[ \begin{bmatrix}
W_{11} & \cdots & W_{1n} \\
\vdots & \ddots & \vdots \\
W_{n1} & \cdots & W_{nn}
\end{bmatrix} \times \begin{bmatrix}
h^t_1 \\
h^t_2 \\
\vdots \\
h^t_N
\end{bmatrix} = \begin{bmatrix}
h^t_{11} \\
h^t_{21} \\
\vdots \\
h^t_{N1}
\end{bmatrix} \]

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

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

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

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

Matrix form for a single layer of graph convolution

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

- **Self loop adjacent matrix is a diagonal matrix!**
- **Noted that** \( B^{T} \) **is a learnable weight matrix**

![Diagram showing the matrix form and self loop adjacency]
A single layer of GNN: Graph Convolution-Forward

Matrix form for a single layer of graph convolution

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

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

\[
H^{t+1} = \sigma \left( D^{-1} A H^{t} W^{T} + A' H_{v}^{t} B^T \right)
\]

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

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

\[
    H^{t+1} = \sigma \left( D^{-1} \hat{A} H^{t'} W^{t'} T \right)
\]

\[
    \begin{align*}
    &\begin{bmatrix}
    A(N \times N) \\
    \end{bmatrix} \\
    &\begin{bmatrix}
    A \quad B \quad C \quad D \quad E \quad F \\
    A \quad B \quad C \quad D \quad E \quad F \\
    \hline
    A \quad B \quad C \quad D \quad E \quad F \\
    A \quad B \quad C \quad D \quad E \quad F \\
    \end{bmatrix}
    + \begin{bmatrix}
    A'(N \times N) \\
    \end{bmatrix} \\
    &\begin{bmatrix}
    A \quad B \quad C \quad D \quad E \quad F \\
    A \quad B \quad C \quad D \quad E \quad F \\
    \hline
    A \quad B \quad C \quad D \quad E \quad F \\
    A \quad B \quad C \quad D \quad E \quad F \\
    \end{bmatrix}
    = \begin{bmatrix}
    \hat{A}(N \times N) \\
    \end{bmatrix} \\
    &\begin{bmatrix}
    A \quad B \quad C \quad D \quad E \quad F \\
    A \quad B \quad C \quad D \quad E \quad F \\
    \hline
    A \quad B \quad C \quad D \quad E \quad F \\
    A \quad B \quad C \quad D \quad E \quad F \\
    \end{bmatrix}
    \end{align*}
\]
A single layer of GNN: Graph Convolution-Forward

Matrix form for a single layer of graph convolution

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

\[
H^{t+1} = \sigma (D^{-1} \hat{A} H^{t'} W'T)
\]

Forward equation for GCN
A single layer of GNN: Graph Convolution-Backward

First, let’s recall…

\[ Y = X \odot W \]

4x2 \hspace{1cm} 4x3 \hspace{1cm} 3x2

How to express \( \frac{dL}{dX} \) and \( \frac{dL}{dW} \) with \( \frac{dL}{dY} \)?
A single layer of GNN: Graph Convolution-Backward

First, let’s recall…

\[ Y = X \, @ \, W \]

4x2  4x3  3x2

How to express \( \frac{dL}{dX} \) and \( \frac{dL}{dW} \) with \( \frac{dL}{dY} \)?

\[
\frac{dL}{dX} = \frac{dL}{dY} \, @ \, \frac{dY}{dX} \\
= \frac{dL}{dY} \, @ \, W^T \\
4x2 \quad 2x3
\]
First, let’s recall...

\[
Y = X \@ W
\]
4x2 4x3 3x2

How to express \(\frac{dL}{dX}\) and \(\frac{dL}{dW}\) with \(\frac{dL}{dY}\)?

\[
\frac{dL}{dX} = \frac{dL}{dY} @ \frac{dY}{dX}
= \frac{dL}{dY} @ W^T
\]
4x2 2x3

What about \(\frac{dL}{dW}\)?

Notice \(Y^T = W^T @ X^T\)

So \(\frac{dL}{dW} = ((dL/dW)^T)^T = (dL/dW^T)^T\)

\[
\frac{dL}{dW} = (\frac{dL}{dY^T} @ X)^T
\]
2x4 4x3

So \(\frac{dL}{dW^T} = \frac{dL}{dY^T} @ \frac{dY^T}{dW^T}\)

\[
\frac{dL}{dW^T} = \frac{dL}{dY^T} @ (X^T)^T
= \frac{dL}{dY^T} @ X
\]
A single layer of GNN: Graph Convolution-Backward

Now, let’s derive the backward equation

\[ H^{t+1} = \sigma(D^{-1}\hat{A}H^tW'^T) \]

NxF \quad NxN \quad NxN \quad NxF \quad FxF

Let’s define

\[ H^\sim = D^{-1}A^\sim H^tW'^T \] (what’s inside the brackets)
\[ H_0^\sim = H^tW'^T \]

Want to derive:

\[ \frac{dL}{dH^t} \]
\[ \frac{dL}{dW'^T} \]
A single layer of GNN: Graph Convolution-Backward

Let’s draw the computational graph

With these definitions
\[ H^\sim = D^{-1}A^\wedge H'W'^T \] (what’s inside the brackets)
\[ H_0^\sim = H'W'^T \]

Want to derive:
\[ \frac{dL}{dH'} \]
\[ \frac{dL}{dW'^T} \]
A single layer of GNN: Graph Convolution-Backward

\[
\frac{dL}{dH^\sim} = \frac{dL}{dH^{t+1}} \ast \frac{dH^{t+1}}{dH^\sim}
\]

\[
\frac{dL}{dH_0^\sim} = (\frac{dL}{dH^\sim} @ (D^{-1}A^\wedge))^T
\]

(recall that \(H^\sim = (D^{-1}A^\wedge)H_0^\sim\))

Now also recall that \(H_0^\sim = H^tW'^T\)

\[
\frac{dL}{dH^t'} = (\frac{dL}{dH^\sim} @ (D^{-1}A^\wedge))^T @ W'
\]

\[
\frac{dL}{dW'^T} = (\frac{dL}{dH^\sim} @ (D^{-1}A^\wedge) @ H^t')^T
\]