Graph Neural Networks
Script based on CS224W

Unity05

December 24, 2021

1 Message Passing Graph Neural Networks

1.1 Introduction

The idea behind message passing GNNs is **k - hop neighborhood aggregation**. One single GNN layer can be looked at as one single hop.

A GNN layer mainly consists of two parts: **Message Computation** and **Aggregation**.

1.1.1 Message Computation

Each node computes a message based on it's embedding in the previous layer.

$$m_u^{(l)} = \phi^{(l)} \left(h_u^{(l-1)} \right)$$

$\quad m_u^{(l)} \quad \dots \quad$ message of node u in layer l
$\quad \phi^{(l)} \quad \dots \quad$ message computation function of layer l
$\quad h_u^{(l-1)} \quad \dots \quad$ node u's embedding in layer l - 1

1.1.2 Aggregation

Node v's new embedding is computed by aggregating its own message as well as all of its neighbor node's messages.

$$h_v^{(l)} = \sigma \left(\square^{(l)} \left(\{ m_u^{(l)} \mid u \in N(v) \}, m_v^{(l)} \right) \right)$$

$\quad \sigma \quad \dots \quad$ nonlinear activation function
$\quad h_v^{(l)} \quad \dots \quad$ node v's new embedding in layer l - 1
$\quad \square^{(l)} \quad \dots \quad$ aggregation function of layer l
$\quad m_u^{(l)} \quad \dots \quad$ message of node u in layer l
$\quad N(v) \quad \dots \quad$ neighborhood of node v

1.2 Popular GNN Layers

1.2.1 Graph Convolutional Networks (GCN)

Message Computation
Embeddings are passed through a linear layer (transformation with weight matrix). Normalized by node degrees.

$$m_u^{(l)} = \left(W^{(l)} \cdot h_u^{(l-1)} \right)$$

Aggregation

$$h_v^{(l)} = \sigma \left(\sum_{u \in N(v) \cup \{v\}} \frac{1}{\sqrt{deg(u)} \cdot \sqrt{deg(v)}} \cdot m_u^{(l)} \right)$$

1.2.2 Graph Attention Networks (GAT)

Message Computation
No difference to GCN.

$$m_u^{(l)} = \left(W^{(l)} \cdot h_u^{(l-1)} \right)$$

Aggregation
Weighted summation of messages normalized by attention weights.

$$h_v^{(l)} = \sigma \left(\sum_{u \in N(v) \cup \{v\}} \alpha_{vu} \cdot m_u^{(l)} \right)$$

Computation of α_{vu} with attention mechanism a:

$$\alpha_{vu} = \frac{exp(e_{vu})}{\sum_{k \in N(v)} exp(e_{vk})}$$

$$e_{vu} = a \left(W^{(l)} h_u^{(l-1)}, W^{(l)} h_v^{(l-1)} \right)$$

1.2.3 Graph Isomorphism Network (GIN)

We don't split up in Message Computation and Aggregation as MLP models both functions ϕ and f.

$$h_v^{(l)} = MLP^{(l)} \left(\left(1 + \epsilon^{(l)}\right) \cdot h_v^{(l-1)} + \sum_{u \in N(v)} h_u^{(l-1)} \right)$$

1.3 Expressiveness of GNNs

To assess the expressiveness of a GNN we have to look at its **computational graph**. It is important to keep in mind that the computational graph can only distinguish different node features not node IDs. Therefore, they are identical to subtrees rooted at the respective nodes.

Hence, we want GNNs to **map subtrees injectively**. This means we need to retain information about neighborhood structure. As the same feature yields to the same message we achieve this by using **injectiv aggregation functions**.

1.3.1 Mean - Pool

Cannot distinguish different multi-sets with the same embedding proportion.

1.3.2 Max - Pool

Cannot distinguish different sets with the same embeddings.

1.3.3 Injective Multi-Set Function

Any multi-set function can be expressed as:

$$\phi \left(\sum_{x \in X} f(x) \right)$$

According to the **universal approximation theorem**, any function ϕ and f can be modeled with an MLP. As MLPs can model compositions of functions, we end up with:

$$MLP \left(\sum_{x \in X} x \right)$$

Transferring this to the domain of GNNs it yields **GIN(1.2.3) as the most expressive GNN**.

2 Graph Generation

2.1 Network Properties

2.1.1 Degree Distribution

The degree distribution $P(k)$ is the probability that a random node has degree k. Computation:

$$N_k = \#\text{nodes with degree k}$$

$$P(k) = \frac{N_k}{N}$$

2.1.2 Clustering Coefficient

How connected the neighborhood of node i with degree k_i is. It basically measures, how many edges out of all possible edges exist.

$$C_i = \frac{2e_i}{k_i\,(k_i - 1)}$$

$$C = \frac{1}{N}\sum_{i}^{N} C_i$$

$$
\begin{array}{lll}
C_i & \ldots & \text{clustering coefficient} \\
e_i & \ldots & \text{total number of edges in neighborhood of node } i
\end{array}
$$

2.1.3 Connectivity

Connectivity is the size of the **largest connected component**. This largest component is called the **giant component**. Can be found by i.e. BFS.

2.1.4 Path Length

Diameter

$$max_{u,v} d(u, v)$$

$d(u, v)$ ...length of shortest path between u and v

Average Path Length

$$\bar{h} = \frac{1}{2E_{max}} \sum_{i,j \neq i} h_{ij}$$

E_{max} ... max number of edges $\left(\frac{n(n-1)}{2} \right)$

h_{ij} ... $d(i, j)$