Assignment 2

KANG, Yi Chen (yckang@connect.ust.hk)

1 Attention and Transformer

1.1 Attention Operation

From matrix representation, Q is the matrix of queries, K is the matrix of keys and V is the matrix of values. The attention score (similarity between a query and all keys) can be computed using a dot product:

$$A = QK^T$$

In terms of time complexity:

- The matrix multiplication QK^T involves computing the dot product of m queries with n keys, dimension of d. The time complexity is $O(\ nmd)$.
- On the softmax operation, each query involves $O(n)$ operations, all queries require $O(mn)$ computation.
- For each query, weighted sum of values can be computed with attention scores, which requires matrix multiplication AV, the time complexity is $O(mnd)$

Therefore, the overall time complexity is $O(mnd)$.

1.2 Self-attention and Cross-attention

1. **Self-attention and cross-attention difference**
   - Self-attention: both the keys and queries are from same source, with the same sets of inputs.
   - Cross-attention: involves queries from one source (eg: decoder input) and keys from another source (eg: encoder output), it allows the decoder to attend to relevant parts of the encoder's output while generating its own output

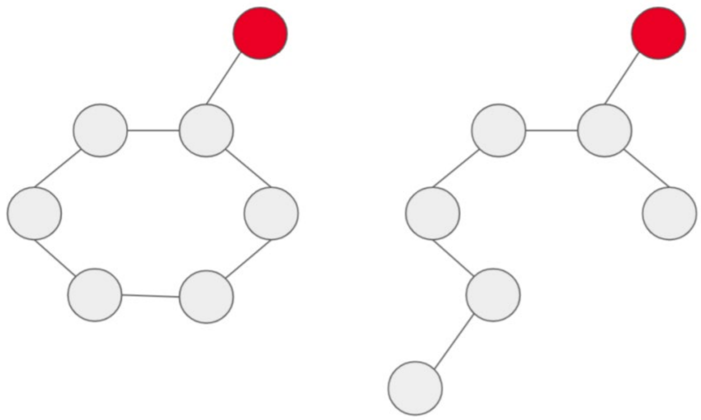
2. **Replacing self-attention (in encoder) with cross-attention**
   Not possible and feasible, as encoder's purpose is to capture dependencies within the input sequence itself. Cross-attention depends on 2 different inputs (eg: encoder to decoder), therefore cannot serve as the same purpose as self-attention.

2 Graph Neural Networks

2.1 Effect of Depth of Expressiveness

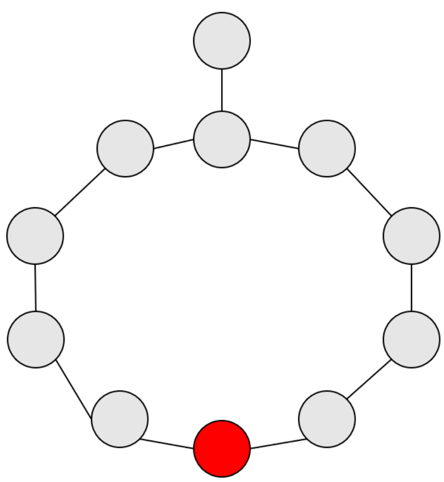
1. **Distinguishing node embeddings based on depth**

   For the following graph: 3 hops are needed to distinguish between the two nodes because the third hop uncovers the structural difference (cycle vs tree).



2. **GNNs and cyclic subgraph classification**

   Example:



   A GNN with fewer than 5 layers cannot perfectly classify (cannot guarantee it is true).

   Each message passing later captures information from a 1-hop neighborhood. To distinguish nodes in a 10-cycle, the GNN needs at least 5 layers, as it needs to capture information from at least 5-hops (half cycle), to differentiate nodes inside the cycle from those outside.

2.2 Relation to Random Walk

1. **Transition to random walk**

   When perform message passing in a GNN using the mean aggregator, each node aggregates the average of its neighbors' embeddings. Given embedding $h_i^{(l)}$ of node i in layer l, the update rule for layer $l+1$ is:

$$h_i^{(l+1)} = \frac{1}{|N_i|}\sum_{j \in N_i} h_j^{(l)}$$

   This is equivalent to a random walk, where the probability of moving from node i to node j is proportional to the adjacency matrix A, normalized by the degree matrix D. Transition matrix T is given as:

$$T = (AD^{-1})^T = D^{-1}A$$

   where D^{-1} normalizes the adjacency matrix A by the degree of each node. This transition matrix represents a uniform random walk, where each node distributes equal probability to all its neighbors.

2. **Skip connection in aggregation**

   When adding a skip connection, it allows a node to retain part of its previous representation while also aggregating information from its neighbors, the update rule becomes:

$$h_i^{(l+1)} = \frac{1}{2} h_i^{(l)} + \frac{1}{2}\left(\frac{1}{|N_i|}\sum_{j \in N_i} h_j^{(l)}\right)$$

   For this case, the corresponding transition matrix is now a weighted combination of the identity matrix I (keep the previous embedding) and the original transition matrix T.

$$T' = \frac{1}{2}(I + T) = \frac{1}{2} I + \frac{1}{2} D^{-1}A$$

   This indicates that the node's new embedding is influenced by both its own previous embedding and by the embeddings of its neighbors.

2.3 Learning BFS with CNN

1. **BFS update rule**

   In BFS traversal, a node is visited (1) or not visited (0). The GNN's goal is to propagate reachability information through the graph. At each step t, a node's embedding is updated based on whether it or any of its neighbors were reached at the previous step, the update rule can be defined as:

$$h_i^{(t)} = \min\left(1, h_i^{(t-1)} + \sum_{j \in N(i)} h_j^{(t-1)}\right)$$

   where $h_i^{(t)}$ is the state of node i at time step t (1 if visited, 0 if not). The result is clamped to 1 using the min function, ensuring that the state remains binary (0 or 1).

2. **Aggregation method**

   The message function MSG describes what information each node sends to its neighbors, for this case, an identity function is used:

$$MSG(i) = h_i^{(t-1)}$$

   Which means that each node i passes its current state to all its neighbors $j \in N(i)$.

   For the aggregation function AGG, it determines how a node aggregates the messages it receives from its neighbors, for this case a clamped sum is selected as the aggregation function.

$$AGG(i) = \min\left(1, \sum_{j \in N(i)} h_j^{(t-1)}\right)$$

   This function sums the states of all neighbors j of node i, then clamps the result to 1. It ensures that the state remains binary (0 or 1), representing whether the node has been visited.