Assignment 2

Chen Yuanzhong

April 12, 2025

1 Attention Time Complexity and Comparison of Self-Attention vs. Cross-Attention

1.1 Time Complexity of the Attention Operation

Consider n pairs of keys and values and m queries, each having dimension d. The attention operation involves:

1. **Score Computation:** Compute the dot product between each query and every key.
   - Each dot product: $\mathcal{O}(d)$.
   - For one query against n keys: $\mathcal{O}(nd)$.
   - For m queries: $\mathcal{O}(mnd)$.
2. **Softmax Normalization:** Apply a softmax to the resulting $m \times n$ matrix.
   - This requires $\mathcal{O}(mn)$ operations.
3. **Weighted Sum:** Multiply the $m \times n$ attention weights matrix with the $n \times d$ values matrix.
   - The multiplication takes $\mathcal{O}(mnd)$.

Combining these steps, the overall time complexity is:

$$\mathcal{O}(mnd)$$

1.2 Self-Attention vs. Cross-Attention

Self-Attention: In self-attention, the queries, keys, and values are derived from the same input. This mechanism allows each element in a sequence to attend to every other element, thereby capturing contextual dependencies within the sequence.

Cross-Attention: In cross-attention, the queries come from one source (e.g., the decoder) while the keys and values come from a different source (e.g., the encoder). This mechanism is used to fuse information from two different sequences.

1.3 Replacing Self-Attention with Cross-Attention in the Encoder

- **Design Purpose:**
  - Self-attention is specifically designed to capture the internal structure of a sequence.
  - Cross-attention is intended to integrate information between distinct sources.
- **Practicality:** Even if we consider self-attention as a special case of cross-attention where the queries, keys, and values are identical, replacing self-attention with cross-attention in the encoder does not provide any inherent benefit. The encoder's goal is to model intra-sequence relationships efficiently. Therefore, we **cannot** replace self-attention in the encoder with cross-attention.

2 Expressive Power of Graph Neural Networks

2.1 Number of GNN Layers Needed to Distinguish Nodes

Message Passing in GNN: In a GNN with sum aggregation and no nonlinearity:

- After 1 layer, each node aggregates information from its 1-hop neighbors.
- After 2 layers, each node has information from its 2-hop neighborhood.
- After 3 layers, each node has information from its 3-hop neighborhood.

Required Number of Layers: From the provided example, we can see that the two red points have different structures in their 3-hop neighborhoods. While the question mentions that the two points have different 4-hop structures, 3 hops is sufficient to distinguish them.

With fewer than 3 layers, the GNN cannot capture the structural differences that exist at the 3-hop distance. With 3 layers, the unique structural information from the 3-hop neighborhood can propagate to the red nodes, resulting in different embeddings.

Conclusion: Therefore, **3 layers** of message passing are needed for the GNN to distinguish between the two red nodes.

2.2 GNN for Node Classification on an Induced Cycle of Length 10

In this task, nodes are classified as positive if they belong to an induced cyclic subgraph (cycle) of length 10, and negative otherwise.

2.2.1 Example Graph

A simple diagram is shown below:

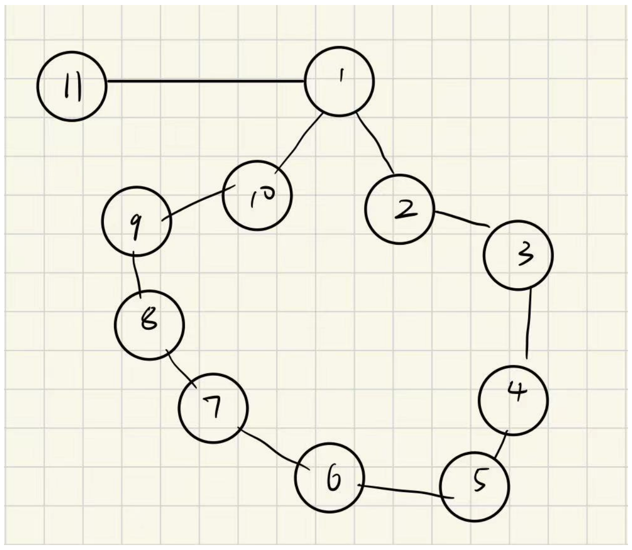

Figure 1: A graph with a positive node (node 1) that belongs to an induced cycle of length 10

In the above graph, node 1 is positive.

2.2.2 Why Fewer Than 5 Layers Are Insufficient

A GNN with L layers aggregates information from nodes at most L hops away. Thus, if a GNN has fewer than 5 layers, each node's receptive field is its L-hop neighborhood with $L \leq 4$.

Observation: In a cycle of length 10, the distance between some pairs of nodes is greater than 4.

Proof Sketch: Assume there exists a GNN with 4 layers which computes node embeddings from only the 4-hop neighborhoods.

1. For a node v_0 on the 10-cycle, its 4-hop subgraph is isomorphic to a simple path that looks similar to the 4-hop neighborhood of a node in a non-cyclic or a longer cyclic graph.
2. Due to the limited receptive field, the GNN cannot detect the global cycle structure because key connectivity (information about nodes at a distance of 5 or more hops) is missing.
3. Consequently, nodes that are part of an induced cycle of length 10 and nodes that are not (but have similar local structures up to 4 hops) may receive identical or indistinguishable embeddings.

Thus, a 4-layer GNN (or any GNN with fewer than 5 layers) is unable to perfectly classify nodes based on whether they lie on an induced cycle of length 10 because it cannot access the full cycle structure.

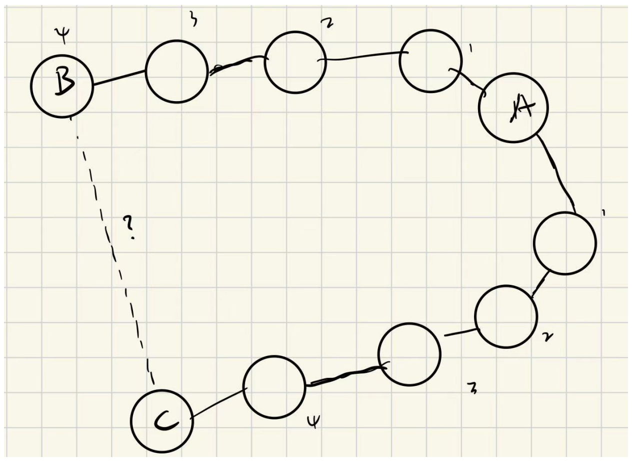

Figure 2: Example graph showing nodes B and C where a potential edge would form a cycle

Example: For the above graph, if we only use 4 layers, we cannot get the information between B and C, that is, we do not know whether there is an edge between B and C that would make them part of a cycle.

Conclusion: At least **5 layers** of message passing are required so that the receptive field of each node covers the entire cycle. A 5-layer GNN will have a 5-hop neighborhood that captures the necessary connectivity (including the edge that closes the 10-cycle) to distinguish nodes on the induced cycle from others.

3 Random Walk Aggregation and Message Passing

3.1 Random Walk and Mean Aggregation

3.1.1 Standard Mean Aggregation

Consider the standard mean aggregation:

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

This is equivalent to taking a one-step uniform random walk from node i. The corresponding transition matrix is given by:

$$P = D^{-1}A,$$

where A is the adjacency matrix with entries A_{ij} and D is the diagonal degree matrix defined by $D_{ii} = \sum_j A_{ij}$.

3.1.2 Mean Aggregation with Skip Connection

Now, suppose we include a skip connection in the aggregation:

$$h_i^{(l+1)} = \frac{1}{2}\,h_i^{(l)} + \frac{1}{2}\frac{1}{|N(i)|}\sum_{j\in N(i)} h_j^{(l)}.$$

In matrix form, this update becomes:

$$h^{(l+1)} = \frac{1}{2}I\,h^{(l)} + \frac{1}{2}D^{-1}A\,h^{(l)}.$$

Thus, the corresponding effective transition matrix is:

$$Q = \frac{1}{2}I + \frac{1}{2}D^{-1}A.$$

3.2 Expressive Power of a GNN for BFS

We assume that each node i has a 1-dimensional embedding $h_i^{(t)} \in \{0,1\}$ at step t. Initially, the source node has

$$h_i^{(0)} = 1,$$

and all other nodes have

$$h_i^{(0)} = 0.$$

3.2.1 Update Rule

At every step, a node becomes *visited* (i.e., its embedding becomes 1) if it was already visited or if at least one of its neighbors was visited in the previous step. This update rule can be written as:

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

3.2.2 Message and Aggregation Functions

A GNN can be designed to mimic the BFS update rule by choosing the following functions:

Message Function: For every directed edge from node j to node i, define the message as:

$$m_{j\to i}^{(t)} = h_j^{(t)}.$$

Aggregation Function: At node i, aggregate the messages from its neighbors by summing them:

$$a_i^{(t)} = \sum_{j\in N(i)} m_{j\to i}^{(t)}.$$

To incorporate the node's own previous state and ensure the output is binary, use a *clamping* function:

$$h_i^{(t+1)} = \min\Big\{1,\ h_i^{(t)} + a_i^{(t)}\Big\}.$$

This aggregation exactly implements our update rule, since if either $h_i^{(t)} = 1$ or at least one neighbor sends a 1 (so that $a_i^{(t)} \geq 1$), then $h_i^{(t+1)}$ becomes 1.

Summary:

- **Update Rule:**
$$h_i^{(t+1)} = \min\Big\{1,\ h_i^{(t)} + \sum_{j \in N(i)} h_j^{(t)}\Big\}.$$

- **Message Function:** $m_{j\to i}^{(t)} = h_j^{(t)}$.

- **Aggregation Function:** $a_i^{(t)} = \sum_{j \in N(i)} m_{j \to i}^{(t)}$, followed by $h_i^{(t+1)} = \min\{1,\, h_i^{(t)} + a_i^{(t)}\}$.

This design ensures that the GNN learns to mark a node as visited at time step $t + 1$ if it or any of its neighbors was visited at time step t, perfectly mimicking the breadth-first search process.