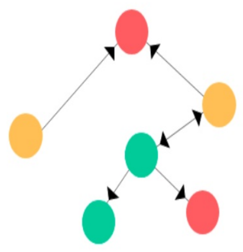
Graph Neural Network

Vishal Turkar and Subhalingam D

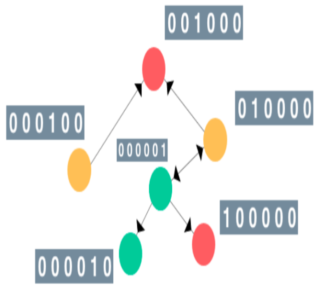
3 June 2020

1 Introduction

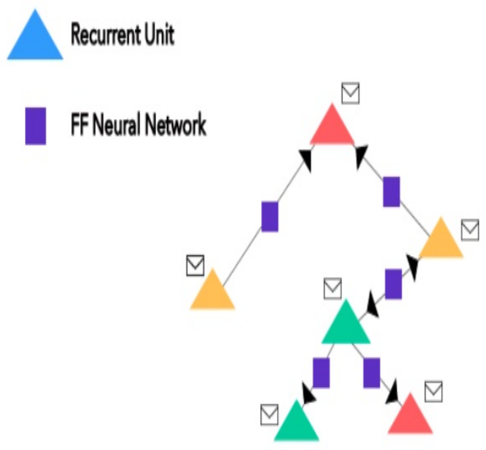
The aim is to read and understand Graph Neural Network(GNN) and conclude if it will be useful for our application. Let us suppose a arbitrary graph G with the below structure:



For simplicity's sake, let's assume that the feature vector is a one-hot-encoding of the current node's index. Likewise, the label (or class) could be the colour of the node (green, red, and yellow as shown above). It'd look something like this:

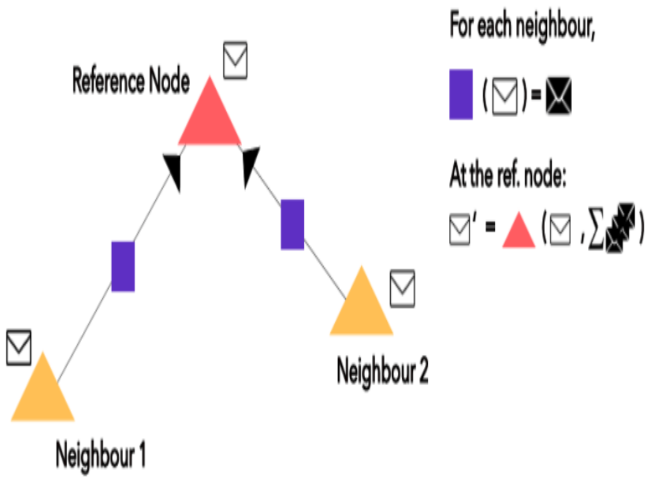


Here for simplicity purpose one-hot-encoding is used but we should use features that uniquely identify the node. Once we have one-hot-encodings (or embeddings) of the nodes, all the nodes are converted into neural network architecture (here into recurrent units) and all the edges house simple feed-forward neural networks. It will look like this:



1.1 Message Passing

Now since the conversion of nodes and edges are completed, the graph performs Message Passing between the nodes. This process is also called Neighbourhood Aggregation because it involves pushing messages (the embeddings) from surrounding nodes around a given reference node, through the directed edges. In terms of GNNs, for a single reference node, the neighbouring nodes pass their messages (embeddings) through the edge neural networks into the recurrent unit on the reference node. The new embedding of the reference recurrent unit is updated by applying said recurrent function on the current embedding and a summation of the edge neural network outputs of the neighbouring node embeddings. To visualise above process observe this figure:



The violet square is a simple feed-forward NN applied on the embeddings (white envelopes) from the neighbouring nodes. The recurrent function (red triangle) applied to the current embedding (white envelope) and summation of edge neural network outputs (black envelopes) to obtain the new embedding (white envelope prime).

This process is performed, in parallel, on all nodes in the network as embeddings in layer L+1 depend on embeddings in layer L. Which is why, in practice, we don't need to 'move' from one node to another to carry out Message Passing.

Once we perform the Neighbourhood Aggregation/Message Passing procedure a few times, we obtain a completely new set of embeddings for each nodal recurrent unit. Through the rounds of Message Passing, the nodes know more about their own information (features) and that of neighbouring nodes. This creates an even more accurate representation of the entire graph. For further processing in higher layers of a pipeline, or simply to represent the graph, you can take all the embeddings and sum them up together to get vector H that represents the whole graph. Using H is better than using an adjacency matrix because these matrices don't represent the features or unique aspects of the graph despite any graph contortion — simply the edge connections between nodes.

1.2 Conclusion

If we used the Statistics-based Approach which was discussed in the previous paper, there would be need of Adjacency matrix bur this matrix won't give a detailed information related to the nodes. In place of that we could use the final vector representation H so as to get better understanding of the knowledge level of a student.