An introduction to Graph Neural Network (GNN)

Graph Neural Network

Graph Neural Network, as how it is called, is a neural network that can directly be applied to graphs. It provides a convenient way for node level, edge level, and graph level prediction task.

There are mainly three types of graph neural networks in the literature:

1. Recurrent Graph Neural Network
2. Spatial Convolutional Network
3. Spectral Convolutional Network

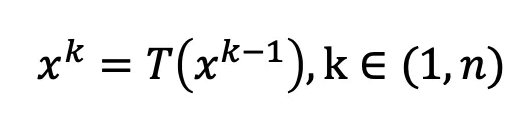
The intuition of GNN is that nodes are naturally defined by their neighbors and connections. To understand this we can simply imagine that if we remove the neighbors and connections around a node, then the node will lose all its information. Therefore, the neighbors of a node and connections to neighbors define the concept of the node.

Having this in mind, we then give every node a state (x) to represent its concept. We can use the node state (x) to produce an output (0), i.e. decision about the concept. The final state (x\_n) of the node is normally called “node embedding”. The task of all GNN is to determine the “node embedding” of each node, by looking at the information on its neighboring nodes.

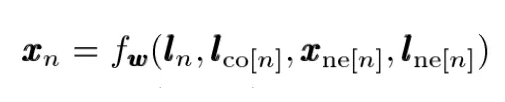
We will start with the most pioneer version of Graph Neural Network, Recurrent Graph Neural Network or RecGNN

Recurrent Graph Neural Network

As introduced in the original GNN paper, RecGNN is built with an assumption of Banach Fixed-Point Theorem.

Banach Fixed-Point Theorem states that: Let (X,d) be a complete metric space and let (T:X->X) be a contraction mapping. Then T has a unique fixed point (x\*) and for any x∈X the sequence T\_n(x)for n→∞ converges to (x\*). This means if we apply t on x for k times, x^k should be almost equal to x^(k-1), i.e.:

RecGNN defines a parameterized function f\_w:



Where l\_n, l\_co, x\_ne, l\_ne represents the features of the current node [n], the edges of the node [n], the state of the neighboring nodes, and the features of the neighboring nodes. (In the original paper, the author referred node feateatures as node