Progress Report for the week preceding April 14 2020

I looked at the Graph embedding techniques available by going through some surveys. I came up with the following one-line intuitions behind some of the more used methods to figure out which one(s) would serve our purpose well. There are three broad categories:

1. Factorization based:
   1. Local Linear Embedding: Each node is a linear combination of its neighbour.
   2. Laplacian Eigenmaps: Keeps two nodes closer if weight is high.
   3. Graph Factorization: Factorizes adjacency matrix
   4. HOPE: Preserves higher order proximity by minimizing an objective function parameterized by Similarity matrix.
2. Random Walk based:
   1. Deepwalk: Preserves higher order proximity by maximizing the probability of observing the last ‘k’ nodes and the next ‘k’ nodes in a random walk.
   2. Node2vec: Maximizes the probability of occurrence of subsequent nodes in fixed length random walks.
3. Deep Learning based:
   1. Graph Convolution Networks (GCN): State of the art. Methods like DNGR (Deep Neural networks for learning Graph representation) or SDNE (Structural Deep network Embedding) uses global neighbourhood of each node as input and are hence computationally heavy making them infeasible for large sparse graphs. GCN solves this doing graph convolution operations on each node in an incrementally aggregated fashion.

A summary table of the above along with some other techniques is included below:

