Unit V - Graph Neural Networks

▼ **Geometric Deep learning**

A niche deep learning that aims to generalize neural network models to non euclidian domains such as graphs and manifolds.

▼ **Word2Vec skipgram and network embedding**

Skipgram architecture

- Input - One hot encoded centre words
- Output - Context word
- Activation function for hidden layer is identity function and for output layer is softmax function
- Input dimension = Output dimension is 1xV
- Word embedding is weight matrix VxN where N is chosen embedding length
- Encoder merely provides a lookup function

Training optimization:

1. Hierarchical softmax
2. Subsampling strategy - looks at the frequency of the word in the corpus and decides whether to include that in the vocabulary for future training.
3. Negative sampling

▼ **Deepwalk**

1. Random walk based embedding uses random walks to generate sequences of nodes for training purposes.
2. Once the sequence of nodes are generated, they are used as an input to a skipgram with negative sampling model.

▼ **Node2vec**

Goal is to encode nodes so that similarity in the embedding space approximates similarity in the original network.

1. Define an encoder - mapping from node to embedding
2. Define a node similarity function
3. Optimize parameters so that similarity ~ embedding similarity

→ Encoder maps each node to a low dimensionality vector

→ Similarity function specifies how relationships in the vector space map to relationships in the original network.

N(bfs) - Local microscopic view

N(dfs) - Global macroscopic view

```
Node2vec algo

-Compute random walk probabilities
-Simulate r random walks of length l from each node u
-Optimize node2vec using SGD

Linear time complexity
All three steps are individually parallelizable
```

Shallow encoding - encoder is just an embedding lookup

Limitations of shallow embedding -

1. O(|V|) parameters are needed
2. Inherently transductive
3. Does not incorporate node features

▼ Graph convolution

Generate node embeddings based on local network neighbourhoods

Network neighbourhood defines a computation graph

Average messages from neighbours → apply neural network

Aggregate - takes an input of the embedding of the neighbouring nodes in the computation graph and generates a message based on this information. This function takes set as input

Update - The above message combined with it's own previous embedding is used to update it's embedding

Both the functions are differentiable functions for backpropagation needs

Transformation - trainable weight matrix with which individual node message is formed

Self loop doesn't require an explicit update, the bias term is integrated within aggregate.

▼ **Permutation invariance and equivariance in GNN**

Invariant to translation means that a translation of input features does not change the outputs at all.

Equivariance to translation means that a translation of input features results in an equivalent translation of the outputs.

> 💡 Graph NN MUST have permutation invariance

▼ **Aggregate and Update**

Aggregate is a set operation → has to be permutation invariant

Normalisation approaches -

1. Simple average
2. Set pooling

3. Symmetric normalization

4. Attention

▼ **GCN**

GCN normalization - symmetric normalization while adding self loop.

▼ **GraphSAGE**

Aggregation + Update (Concat)

- mean
- pool
- lstm
- l2 normalization

▼ **GNN based modelling**

Receptive field in a GNN - set of nodes that determine the embedding of a node of interest.

How to make a GNN more expressive?

- Increase the expressive power within each GNN layer
- Add layers that do not pass messages
- Add skip connections in GNN