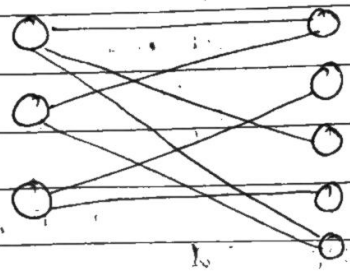
DATE : / /
PAGE NO. :

graph parameters

Bipartite graphs:-

A graph 'G' is bipartite if its vertices can be partitioned into two disjoint sets L and R such that every edge of G connects a vertex in L to a vertex in R i.e no edge connects two vertices from the same part.



Theorem : A graph is bipartite if and only if it has no cycles of odd length.

Matchings :-

- A matching in a graph is a set of edges without common vertices.
- A maximal matching is a matching which cannot be extended to a larger matching.
- A maximum matching is a matching of the largest size.

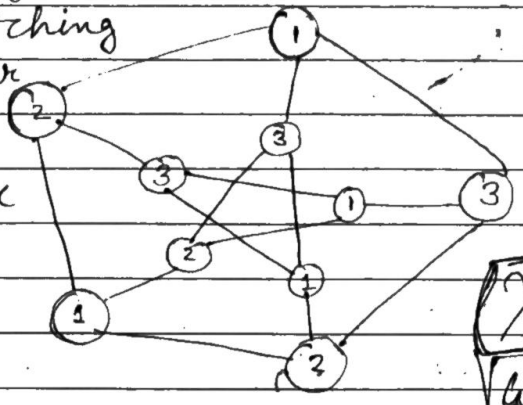
Hall theorem:-

Set of vertices is a vertex cover if and only if its complement is an independent set.
The vertices of any maximal matching form a vertex cover.

DATE: / /
PAGE:

Four Color theorem - every map can be colored with 4 colors

In a bipartite graph, the number of edges in a maximum matching equals the number of vertices in a Minimum Vertex cover.

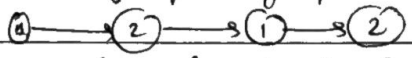


∴ chromatic no $\chi = 3$

$$\chi(G) \cdot \alpha(G) \geq n$$

$$\omega(G) = \alpha(G')$$

Chromatic no of Full graphs K_n is n.
chromatic no of path graphs P_n is 2 when $n > 1$



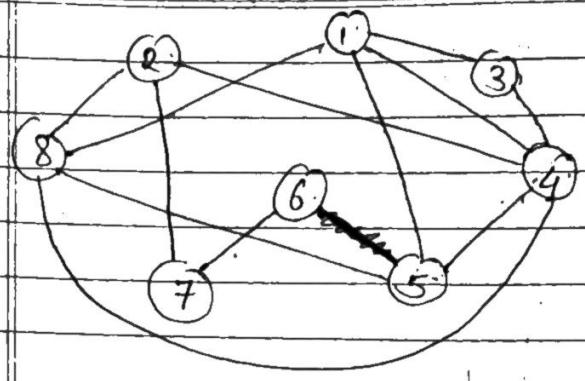
Chromatic no of cycle graph C_n
if n is even $\chi = 2$
if n is odd $\chi = 3$; $n > 2$

chromatic no of bipartite graph is 2 with atleast 1 edge
Every planar graph can be colored with 4 colors.
A graph G of max degree Δ can be colored with $\Delta + 1$.
A graph G of max degree Δ can be colored with Δ colors, unless G is full (K_n) or a cycle of odd length C_{2k+1}.

Clique No. $\omega(G)$

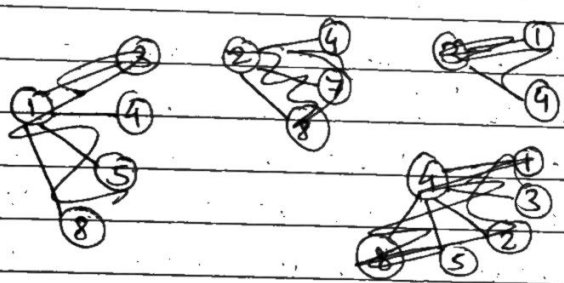
A Clique of a graph is a set of vertices such that every two vertices are connected by an edge. The maximal clique is a clique which is not contained in any other clique. The clique no of a graph G is the no of vertices in its maximum clique. The maximum clique is a clique such that there are no cliques with more vertices.

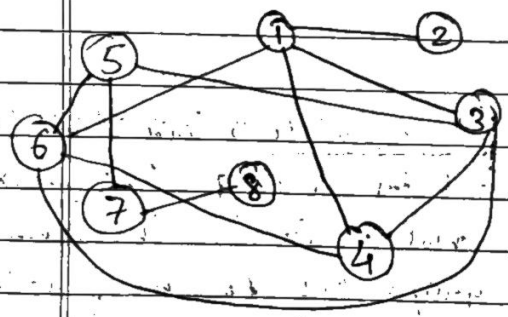
DATE: / /
PAGE NO.:



(1, 4, 5) is a clique
(1, 3, 4) is a maximal clique.
(1, 4, 5, 8) is a maximum clique.

Independent set of a graph is a set of vertices such that no two vertices are connected by an edge.
A maximal independent set is an IS which is not contained in any other IS (cannot be extended to a larger IS)
A maximum IS is an IS such that there are no IS with more vertices.
The independence no. $\alpha(G)$ of a graph G is the no of vertices in its max IS





(2, 5, 4) IS
(2, 7, 4) maximal IS
(2, 8, 5, 4) maximum IS

Imp: Each color forms an independent set.
n vertices can be partioned into χ IS.

DATE : / /
PAGE NO. :

$\chi(G) \cdot \alpha(G) \geq n$

each IS is of size $\leq \alpha(G)$

vertex cover of a graph G is a set of vertices C such that every edge of G is connected to some vertex in C

minimum vertex cover $\beta(G)$.

Note: A set of vertices is a vertex cover if and only if its complement is an independent set.

$$\beta(G) + \alpha(G) = n$$

Konig's Theorem – The vertices of any maximal matching form a vertex cover.

DATE : / /

PAGE NO. :

$$C_i = \frac{2e_i}{k_i(k_i - 1)}$$

$$\text{average path length} = \frac{\sum d(x,y)}{2E_{max}}$$

$$C_{close}(x) = \frac{1}{\sum_{y=1}^{N} d(x,y)}$$

$$C_{bet}(x) = \sum_{u,v \neq x} \frac{\sigma_{u,v}(x)}{\sigma_{u,v}}$$

there should be atleast one path from u to v

DATE : / /
PAGE NO. :

GAT and RNN

Graph Attention network (GAT) is a type of neural network that is designed for processing data that is represented as a graph. It uses attention mechanisms to aggregate information from the neighboring nodes in the graph and produce a node level representations. The attention mechanism allows the GAT to focus on the most relevant information from the neighboring nodes.

GAT can be useful to analyze social networks by representing the nodes as ~~individuals~~ users in the network and the edges as the relationships between them. GAT can then learn to extract features from network and use attention mechanisms to focus on the most relevant relationships for a particular task, such as predicting which users are likely to be friends.

RNN (Recurrent neural networks) is a type of neural network that is designed for processing sequential data, where the order of inputs matters. It maintains an internal state that captures the content of the previous inputs, and uses it to make predictions about the current input.
RNN on the other hand, can be used to analyze social network data that evolves over time. For example in a dynamic social network where the relationships between individuals change over time, an RNN can be used to capture the temporal dependencies between the network states at different time steps. This can be useful for predicting how the network will evolve in the future or

DATE : / /
PAGE NO. :

detecting anomalous bhavior in the network

In combination, both GAT and RNN can be used to analyze both the structure and dynamics of social network.

While both GAT and node embedding in the GNNs involve learning low dimensional representation of nodes in a graph, they differ in their approach to aggregation, training objective and application domains.

~~Node embedding~~

① GAT uses attention mechanisms to weight the importance of the neighboring nodes when computing the node level representations, whereas GNNs typically use graph convolutional layers to aggregate information from the neighboring nodes.

② Node embedding in GNNs typically involves training the model to minimize a loss function that captures the similarity between the predicted and actual node embeddings, whereas in GAT the attention mechanism is trained end-to-end with the downstream task, such as node classification or link prediction.

DATE: / /
PAGE NO.

Recurrent neural network

graph RNN is a generative model for graphs. It is used to generate realistic graphs.

The key idea is that the graph is generated via sequential adding of nodes and edges.

Model graph as a sequences — Graph G with node ordering π can be uniquely mapped into a sequence of node and edge additions S^{π}

A graph is set of nodes and objects and a graph does not have any node ordering by itself but given a node ordering π that determines in which the nodes appears in the graph then the sequence to generate the graph is uniquely defined.

We can say that given a node ordering π along with the graph the problem of generating a graph reduces to a simple sequence generation problem.

The sequence S^{π} has two levels. There is a sequence of adding nodes and there is a sequence of addding edges. At node level sequence, at each step a new node is added and after a node is added the edges are added for that node. So the two levels are the high level - node level sequence and the low level - edge level sequence.

In the edge level sequence a binary sequence is generated that says "connect" for "1" and "don't connect" for "0".

DATE: / /
PAGE NO.:

for all the previous nodes present in the graph till that level.

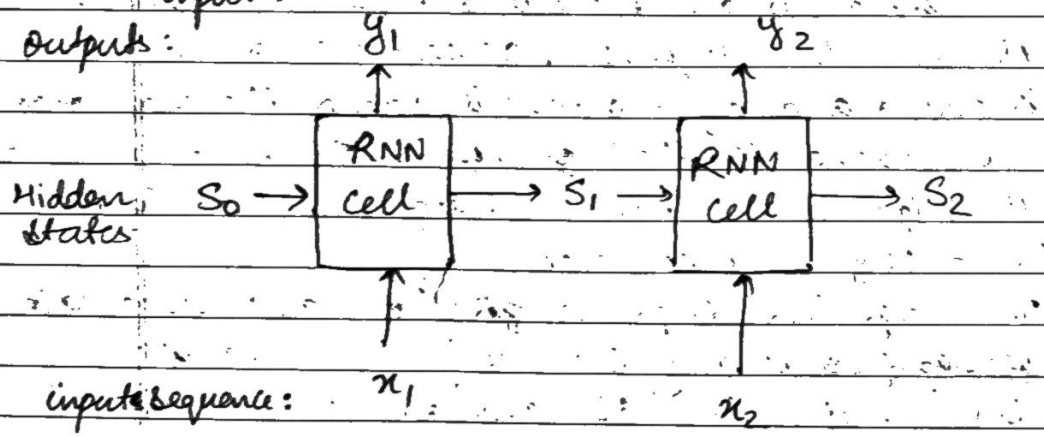
To model this two-level sequence problem we are going to use the nested Recurrent neural network.

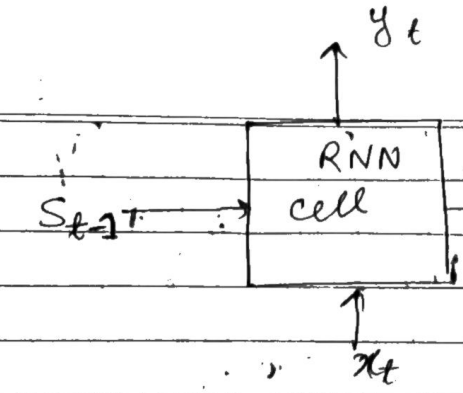
What are recurrent neural networks?

Recurrent neural networks is a type of neural network that are designed for sequential data. And it takes ~~an input sequence~~ sequence as an input to update its hidden states ~~and based on the hidden state~~ and the hidden state summarizes all the information input to the RNN and the update is conducted via RNN cells. For every time step we have a new RNN cell.

or

RNN is a type of neural network that is designed for processing sequential data, where the order of input matters. It maintains a hidden state that captures the content of the previous inputs and uses it to make predictions about the current input.





where, y_t = Output of RNN at step t
x_t = input to RNN at step t
S_{t-1} = state of RNN at step t
S_t = state of RNN after step t

The RNN cell will do :- ① Update hidden state

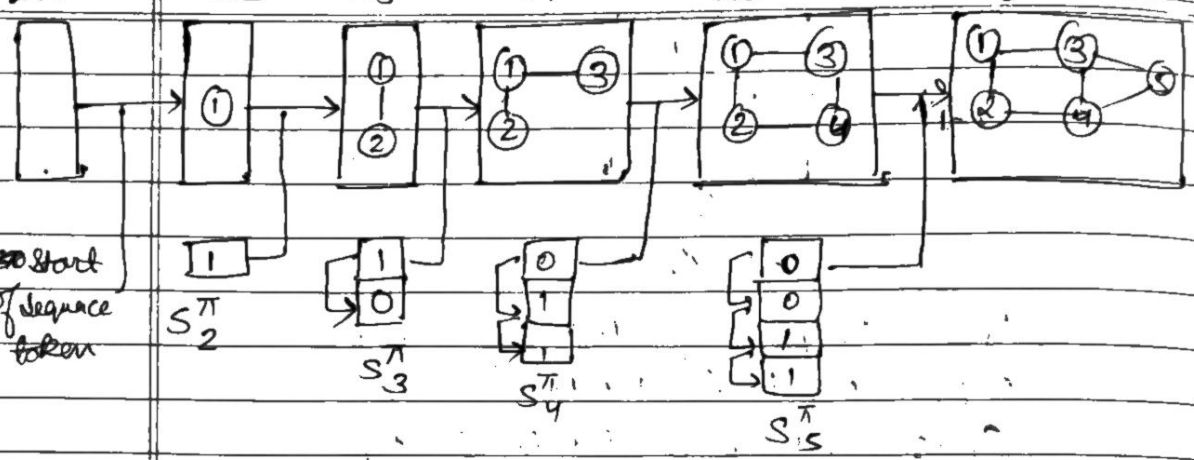
$$S_t = \sigma(W \cdot x_t + U \cdot S_{t-1})$$

② Output prediction.

$$y_t = V S_t$$

where, W, U, V are trainable parameters that can be a vector or matrix.

Now, GraphRNN has a node-level RNN and an edge level RNN. The node-level RNN generates the initial state for edge-level RNN and edge-level RNN sequentially predict if the new node will connect to each of the previous node. After that it will pass the state back to the node level RNN to generate the new node, update the hidden state and push down to the edge level RNN.



Ques: How to use RNN to generate sequences?

=> We are basically going to take the output of the previous RNN cell and put it as an input to the next cell

$$x_{t+1} = y_t$$

Ques: How to initialize the input sequence?

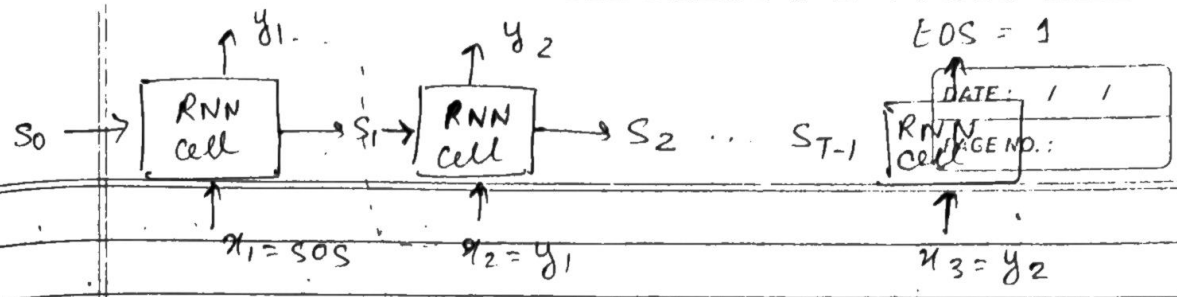
=> We initialize it to have a special token and train the neural network that when it gets this special token SOS as initial input, the network starts generating the output. It might be a vector of all zeros or all ones.

Ques: When to stop generation?

=> We stop the generation when the end of sequence token is produced. Use end of sequence token (EOS) as an extra RNN output

- If output EOS = 0, RNN will continue generation.
- If output EOS = 1, RNN will stop generation.

EOS is produced by edge level RNN, which states that no edges are connected between the new node and the previous nodes. Hence, stop the RNN generation.



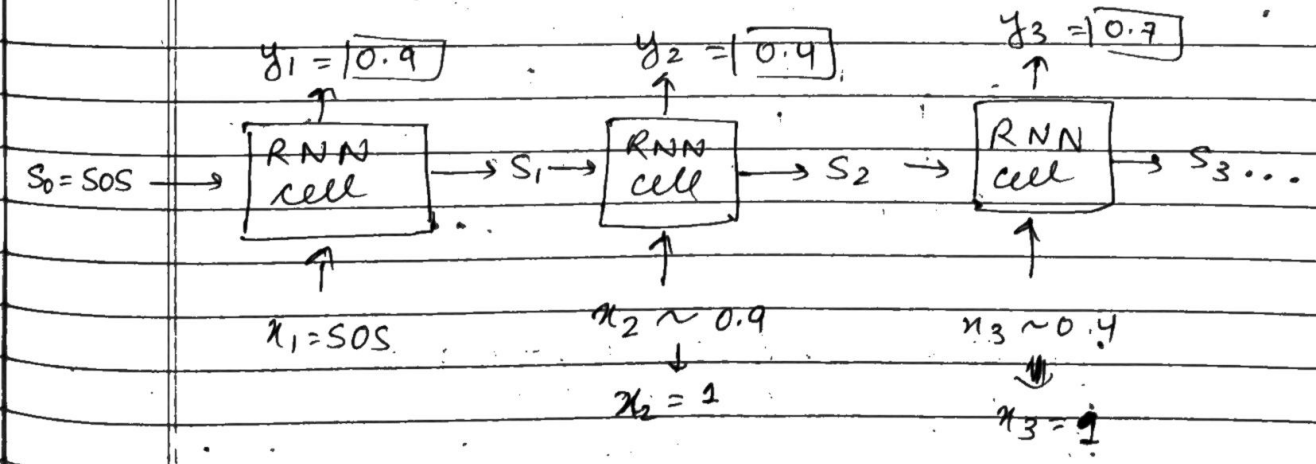
Problem with model is that it is deterministic and generating edges and nodes in a deterministic way. We want a stochastic model that will have randomness.

We are using now the RNN to ~~produce~~ model the product of conditional distributions.

$$\prod_{t=1}^{n} P_{model}(x_t \mid x_1, \ldots x_{t-1}; \theta)$$

sample x_{t+1} from $y_t \Rightarrow x_{t+1} \sim y_t$

Each step of RNN outputs a probability, P, of a single edge (edge level) and based on that ~~it takes value 1 with probability p and 0 with probability (1-p)~~ ~~flip a coin~~ ~~0 or 1~~ that will be the input to the next step. Instead of generating an edge, RNN generates the probability and then we have a stochastic event with the output as bias that ~~feeds~~ gives 0 or 1 which goes as input to the next step. Here, y_t will be a scalar, a bernaulli distribution.

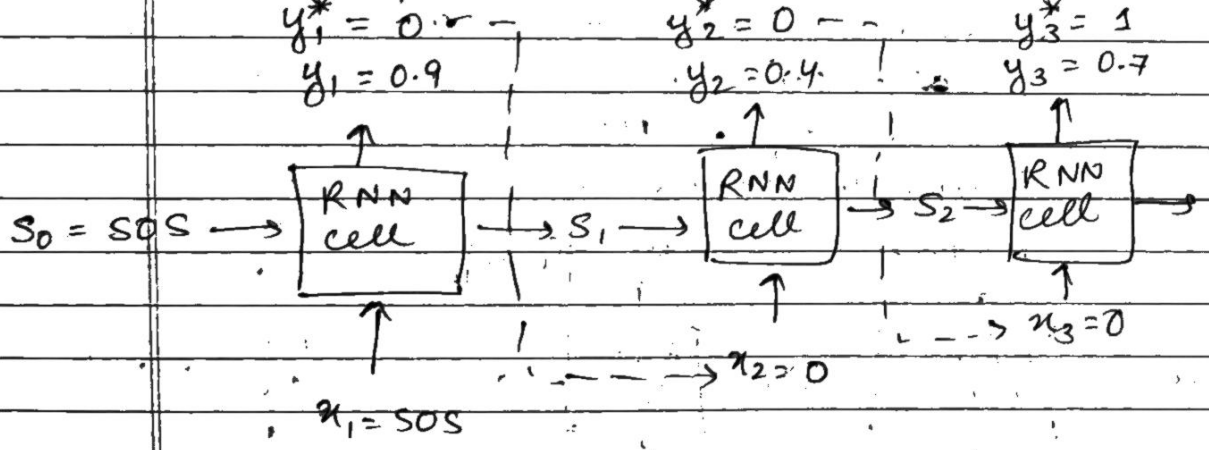


DATE : / /
PAGE NO.

Start the RNN it will output a perbability and based on this probability it will decide whether there will be an edge or not and whatever will be the output it will be sent to the next cell ~~where~~.

Ques: How do we use the training data $x_1, x_2 \ldots, x_n$?

=> Here the training data is the training graphs. In order to train the model we are assuming the graph that was given to us ~~we have~~ ~~the~~ ~~given~~ and we have already observed its sequence of zeros and ones depicted by y^*
we use the principle of "Teacher Forcing" that replaces the input and the output by the real sequence.



And no stochastic event will occur, only forcing the correct inputs will be done during the training. This is called Teacher Forcing.

DATE : / /
PAGE NO. :

Loss :- Binary cross entropy loss fection.

$$L = -[y_i^* \log(y_i) + (1 - y_i^*) \log(1 - y_i)]$$

if $y_i^* = 1$, we minimize $-\log(y_i)$; making y_i higher. Generate a probability close to 1.

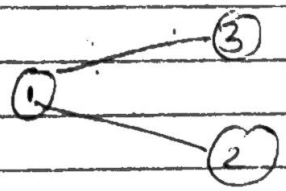
if $y_i^* = 0$, we minimize $-\log(1 - y_i)$, making y_i lower. Generate a probability close to 0.

This way y_i is fitting the data samples y_i^*

y_i is computed by RNN, this loss will adjust RNN parameters accordingly, using back propagation.

matrices W, U & V

node-level RNN →



0	1	1	edge level RNN
1	0	0	
1	0	0	

(4) Isolated node giving signle of F.O.S

DATE : / /
PAGE NO. :

The Strength of Weak Ties :-

In graph theory, a weak tie is typically referred to as a "bridge" or "weak bridge" edge. An edge is said to be a bridge edge if its removal would disconnect the graph into two or more connected components. A weak bridge edge is an edge that connects two components, but whose removal would not affect the connectedness of either component.

The concept of weak ties in sociology and graph theory are related and depicts a connection between individuals or nodes that are not strong and direct. They are often used to analyze the resilience of networks and the affects of edge removal on network structure

"The Strength of weak ties" is an idea that weak ties between nodes can be more valuable than strong ties for accessing information that are not available within an immediate social circle.

In SNA weak ties can be identified and analyzed using a variety of metrics, measures of centrality and clustering. One important metric for measuring the strength of weak ties is "betweenness centrality" which measures the extent to which an individual lies on the shortest path between other individuals in the

DATE : / /
PAGE NO. :

network. Individuals with high betweenness centrality are often "bridge" or "broker" nodes who connect otherwise disconnected parts of the network and have the potential to play important roles in facilitating information flow and access to resources.

Another important concepts related to the strength of weak ties in SNA is "structural holes", which refer to the gaps or holes in a network that exists between groups or clusters of individuals who are not directly connected. Structural holes create an opportunity for nodes with weak ties to bridge these gaps and access information that are not available within their immediate social circle.

→ Suicide with less clustering coefficient.

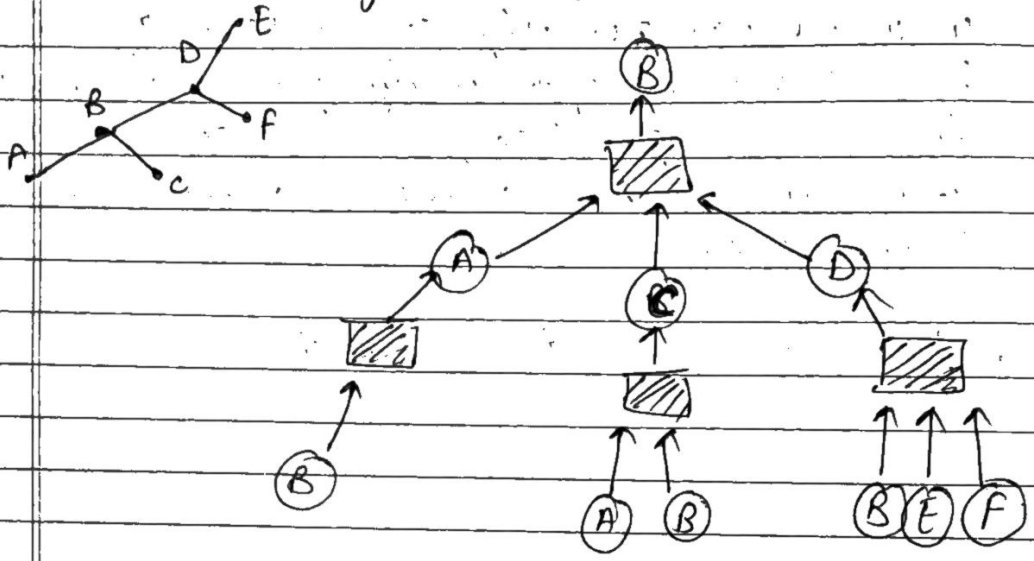
DATE : / /
PAGE NO.

Graph Sage

Graph Sage Neighbor sampling: Scaling up GNNs

GNN generate node embeddings via neighborhood aggregation i.e they aggregate feature information from the neighbors and create a message and pass it on.

Observation :- A 2 layer GNN generates embedding of node "B" using 2-hop neighborhood structure and features.



More generally, we can say that A K-layer GNN generates embedding of a node using K-hop neighborhood structure and features.

This means that one needs only little memory as one need equal to K hops neighbors of the target node to generate embeddings and the rest of the network can be ignored. Because the rest of the network does not affect the embedding of this node of interest. This means that

DATE : / /
PAGE NO. :

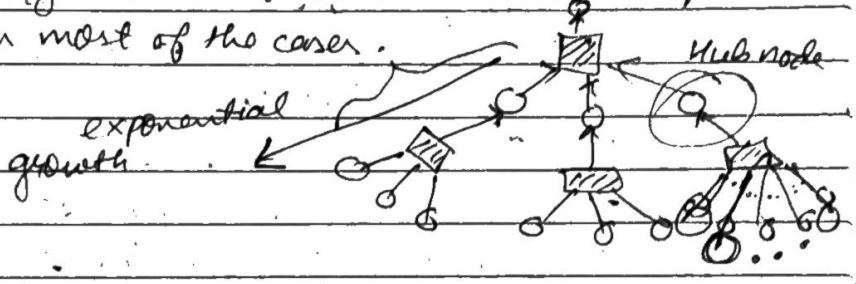
~~we can generate minibatches of the~~ ~~computational graphs~~ we can generate a minibatch of M different nodes using their computational graphs. And then these M computational graphs are put into GPU memory so that we can compute the loss over this mini batch of M computation graphs.

Instead of putting nodes into the minibatches we are putting computational graphs or in other words K-hop neighborhoods.

Issues :-

1. Computation graph becomes exponentially large with respect to the layer size K.
2. Computation graph explodes when it hits a hub node (high degree node).

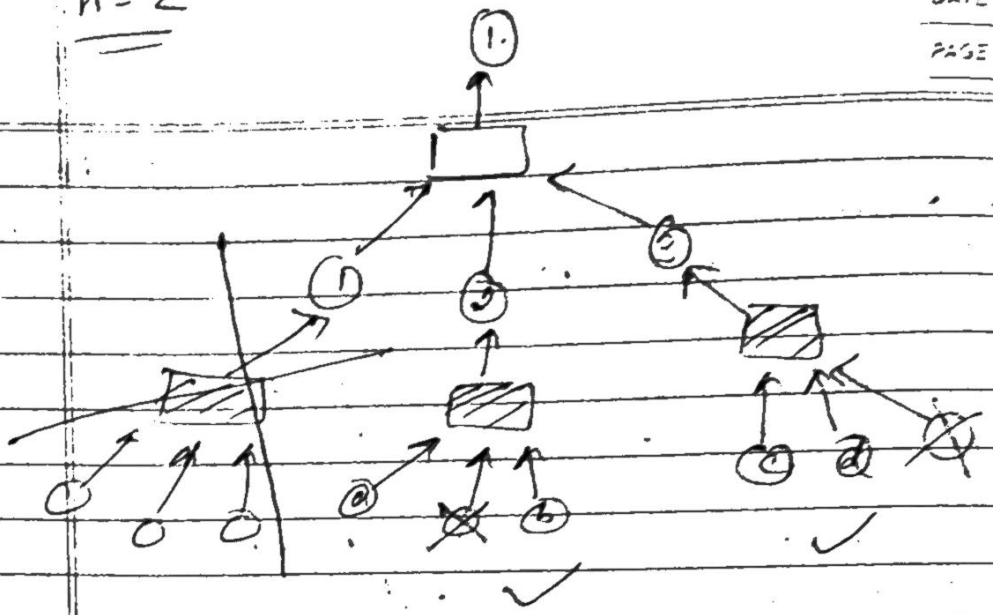
Because of this you cannot take the entire K-hop neighborhood in most of the cases.



Solution

Neighbor sampling — Construct the computational graph by (randomly) sampling at most H neighbors at each hop. i.e that every node in the computational graph is going to ~~at most~~ aggregate from at most H neighbouring nodes.

H = 2



Consider only (2) and (3) for embeddings for node (1) in hops and nodes a, b, c & d for hop-2.
This is called pruned computational graph and is used to more efficiently compute node embeddings.

Remark 1: Trade off in sampling number H:
⇒ Smaller H leads to more efficient neighbor aggregation, but results in more unstable training due to the larger variance in neighbor aggregation.

Remark 2: Computational time
⇒ Even with neighbor sampling, the size of the computational graph is still exponential with respect to number of GNN layers K.
⇒ Increasing one GNN layer would make computation H times more expensive

Remark 3: How to sample the nodes:
⇒ Random sampling: fast but many times not optimal (may sample many "unimportant" nodes)

DATE : / /
PAGE NO. :

=> Random walk with restarts :
Natural graphs are "scale-free", sampling random neighbors, samples many low degree "leaf" nodes
Strategy to sample important nodes :
- Compute random walk with restarts score R_i starting at the node of interest.
- At each level sample H neighbors i with the highest R_i

Summary

A computational graph is constructed for each node in a mini-batch.
In neighbor sampling, the computational graph is pruned/sub-sampled to increase computational efficiency.
The pruned computational graph is used to generate a node embedding.
However, computational graphs can still become large, especially for GNNs with many message-passing layers. If batch size is small the gradient is less reliable. and if pruning is too much then also gradient are less reliable. Balance between pruning factor H and batch size should be maintained for computational graphs.

DATE : / /
PAGE NO. :

GAT (graph attention network).

Ques Explain the concept of attention mechanisms in graph neural networks. How do they work and how do they improve performance of GNNs on various graph based tasks?

=> (1) Attention mechanisms in graph neural networks (GNNs) allow the network to selectively focus on different parts of the input graph during each layer of computation (2) This attention mechanism provides a way for the network to weigh the importance of different nodes or edges in the graph, based on their relevance to the task at hand. The basic idea of attention (3) in GNNs is to compute an attention score for each node or edge in the graph, which reflects its importance for the current task (4) The attention scores are then used to compute weighted sums of the features of neighboring nodes or edges, which are then combined with the current node or edge feature to produce a new representation.

(5) The attention score for each node or edge can be computed in different ways, depending on the specific attention mechanism used. One common approach is to compute the attention score based on the similarity between the features of the current node or edge and its neighbors (2) This is often done using a dot product or a multilayer perceptron (MLP) applied to the concatenation of the features. The

(7) Attention mechanism has been widely used in NLP tasks such as sentiment analysis and question answering.

DATE : / /
PAGE NO. :

(8) Attention mechanism in GNNs have been shown to improve performance on a wide range of graph based tasks, including node classification, link prediction, and graph classification. By selectively attending to the most relevant parts of the graph, GNNs can capture more nuanced relationships between nodes and edges, and produce more accurate predictions. Attention mechanisms also make it possible to interpret the decisions made by the network, by visualizing the attention scores assigned to each node or edge.

Que. Compare and contrast the performance of GCN and GAT on node classification and link prediction tasks. What are some advantages and disadvantages of each approach.

=> GCNs and GATs are two popular types of GNNs commonly used for node classification and link prediction tasks. or

Similarities :-

- Both GCNs and GATs are based on the idea of propagating information through the graph, by aggregating information from neighboring nodes.
- Both methods use a layer-wise approach, where each layer updates the node features based on the features of its neighbors and its own current feature representation.
- Both GCNs and GATs can handle graph with variable size and structures.

DATE : / /
PAGE NO. :

Differences :-

- GCNs use a fixed weight matrix to aggregate information from the neighbors of a node, while GATs use an attention mechanism to learn a dynamic weighting for each neighbor based on the current node representation.
- GATs are able to selectively attend to different parts of the graph, whereas GCNs treat all nodes equally.
- GATs are computationally more expensive than GCNs due to use of the attention mechanism.

Regarding performance GAT's have been shown to outperform GCNs in several node classification and link prediction tasks. But GATs are slower and less scalable. GATs are also more sensitive to hyperparameters.

GCNs are computationally efficient and easy to implement, but may struggle with capturing fine-grained information from the graph.