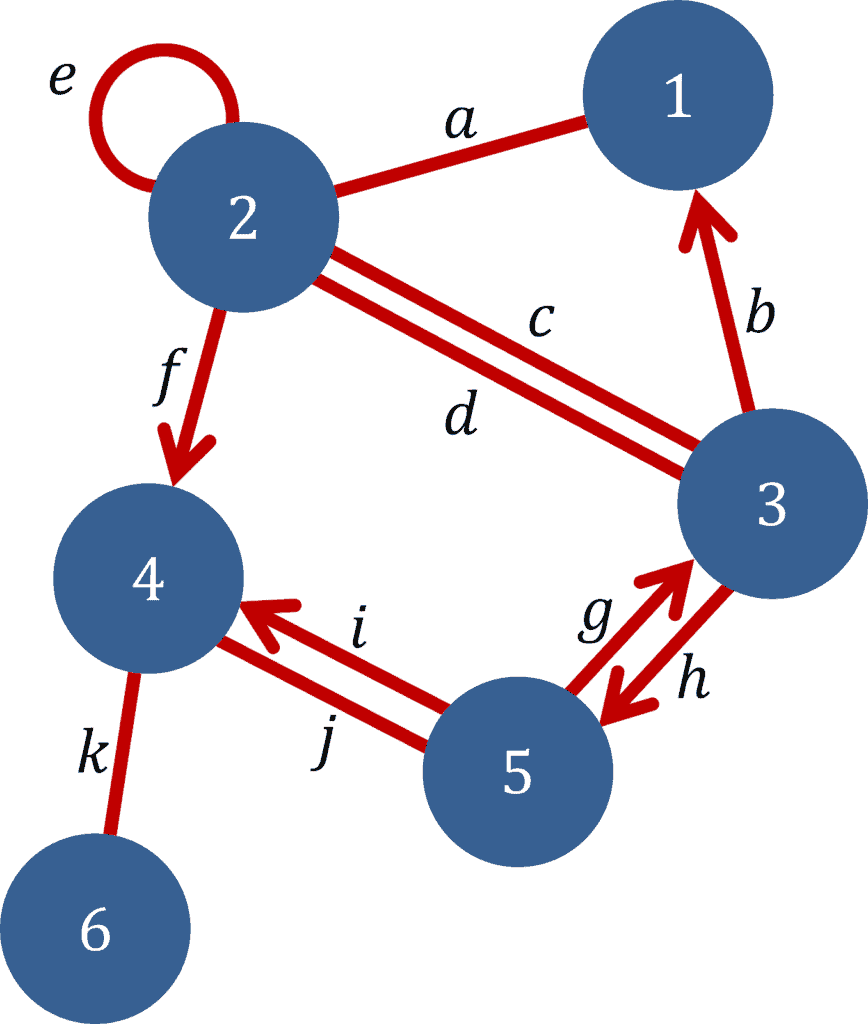
The Essential Guide to GNN

* Graph data works in the non-euclidean space.
* Graph analysis is aimed at node classification, link prediction, and clustering.



* 1-2 is undirected, 2-4 is a directed node
* Used widely in social network analysis
* Graph Neural Networks are a special class of neural networks that are capable of working with data that is represented in graph form.
* The nodes in the graphs aren’t represented in any order and dependency information between two nodes is represented by edges.

Types of Graph Neural Networks

* Graph Convolutional Networks (GCNs)- Graph Convolutional Networks (GCNs) utilize the same convolution operation as in normal Convolutional Neural Networks. GCNs learn features through the inspection of neighboring nodes. They are usually made up of a Graph convolution, a linear layer, and non-linear activation. GNNs work by aggregating vectors in the neighborhood, passing the result to a dense neural net layer, and finally applying non-linearity.

A Gentle Introduction to Graph Neural Networks

Images as graphs

Images are modelled as graphs with regular structure, where each pixel represents a node and is connected via an edge to adjacent pixels. Each non-border pixel has exactly 8 neighbors, and the information stored at each node is a 3-dimensional vector representing the RGB value of the pixel.

A screenshot of a computer

Description automatically generated

Text as graphs

We can digitize text by associating indices to each character, word, or token, and representing text as a sequence of these indices. This creates a simple directed graph, where each character or index is a node and is connected via an edge to the node that follows it.

* Graph valued data such as molecular structure with variable number of neighbouring nodes is hard to model other than in graphs.
* Social network analysis is even more wild (meta is not playing around)
* Unlike image and text data, social networks do not have identical adjacency matrices.
* Computer vision generates a graph by taking objects as nodes and identifies the relationship between them which will be the edges of the graph.
* There are three general types of prediction tasks on graphs: graph-level, node-level, and edge-level.
* In a graph-level task, we predict a single property for a whole graph. (This is analogous to image classification problems)
* Node-level tasks are concerned with predicting the identity or role of each node within a graph. (node-level prediction problems are analogous to image segmentation)
* For an edge-level task, we want to predict the property or presence of edges in a graph. Nodes represent the objects in the image, we wish to predict which of these nodes share an edge or what the value of that edge is. (image scene understanding)
* A problem with graphs is that there are many adjacency matrices that can encode the same connectivity, and there is no guarantee that these different matrices would produce the same result in a deep neural network.
* A graph of 4 nodes connected to each other can have 24 different adjacency matrices representing the same data (in different order of permutation).
* One elegant and memory-efficient way of representing sparse matrices is as adjacency lists. These describe the connectivity of edge **ek** between nodes **ni** and **nj** as a tuple (i,j) in the **k-th** entry of an adjacency list. Since we expect the number of edges to be much lower than the number of entries for an adjacency matrix (**n^2nodes**), we avoid computation and storage on the disconnected parts of the graph.

Graph Neural Networks

* GNNs using the “message passing neural network” framework takes graphs as input, with information loaded into its nodes, edges and global-context, and progressively transform these embeddings, without changing the connectivity of the input graph.
* Pooling –

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

* Message passing  
  Message passing works in three steps:

1. For each node in the graph, gather all the neighbouring node embeddings (or messages).
2. Aggregate all messages via an aggregate function (like sum).
3. All pooled messages are passed through an update function, usually a learned neural network.

* By stacking message passing GNN layers together, a node can eventually incorporate information from across the entire graph: after three layers, a node has information about the nodes three steps away from it.
* Problem – If a node N, which is important for updating the embeddings of the graph if far away and we perform message passing k times upto k layers, and the node is further away from k. In this case, we miss the node N.
* One solution – is by using the global representation of a graph (U) which is sometimes called a master node or context vector. This global context vector is connected to all other nodes and edges in the network, and can act as a bridge between them to pass information, building up a representation for the graph as a whole. This creates a richer and more complex representation of the graph than could have otherwise been learned.
* The performance of graphs depends on the message passing among entities or group of entities(nodes, edges, nodes & edges, global), aggregator function(mean, sum, max), edge embedding, node embeddings, number of layers, etc.
* Having inductive biases if the graph is doing some specialised computation can help improve key results.
* Sampling graphs can be done using various algos and is still a hot area of research.
* You can traverse and pass messages in graphs using matrix multiplications
* Transformers can be viewed as GNNs with an attention mechanism

A screenshot of a computer

Description automatically generated

Recommendation System using GCNs

Action plan –

* First gather a dataset, using Movielens dataset with 9000 movies and 600 users
* Step 1 is to preprocess the data and create a graph, define a threshold rating of a movie say 4 as positive interaction. Any interaction above 4 is considered an edge.
* I cant use RMSE to compute loss as my dataset as I don’t have actual ground truth labels so I’m using BPR (Bayesian personalized rating) loss, a pairwise objective which encourages the predictions of positive samples to be higher than negative samples for each user.

