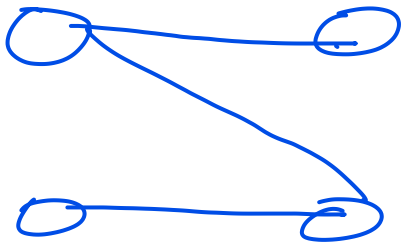
Graph: Graph is general language which describe/analyze relation/interaction amongentities

example:



→ Graph.

Many entity connected with relation

e.g: event graph

① event-graph.
② computer network.
③ ~~Da~~ Disease Pathways
④ Molecules, neurons, road network
⑤ Software → Code graph
⑥ knowledge graph
⑦ Citation network
⑧ social network etc.

Broadly dividing them there will be 2 Categories:

① Naturally Occuring graph
e.g Molecular structure
neurons
Protiens structure etc.

② Graph representation:
code graph
event graph.
knowledge graph etc.

In all this ~~the~~ main question is how

~~do~~ do we take advantage # of relational structure for better prediction?
Complex domain has rich relational structure which can be represented as relational graph.
We need to model this rich relationship structure for better prediction.

If we look at modern deep learning toolbox this there are very effective on grid or sequence kind of data.
Grid: Image
Sequence: Audio, text

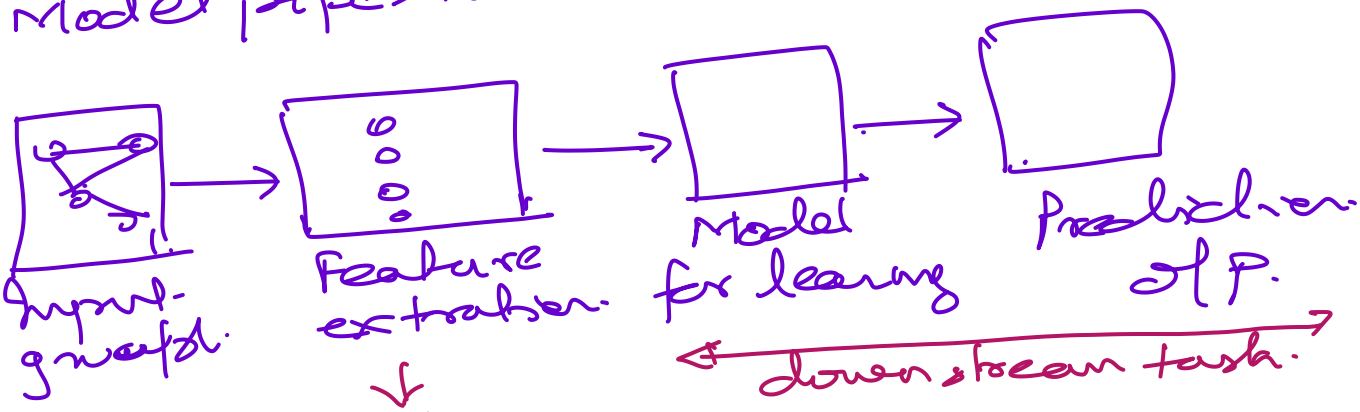
Entites which rich structural information can't be always represented as grid or sequence data. So these DL is not effective
We need more flexible DL which handles structural information ~~more~~ and applicable on these dataset.

Why Graph Deep learning is hard?
(1) No so special locality like grid/sequence.
(2) No starting reference point
(3) Multimodal feature and dynamic in nature.

We need to develop deep learning model which can handle above point.
These model usually perform following task:

1. Node level: node classification
2. Edge level
3. Graph level prediction / Graph generation
4. community / subgraph level task

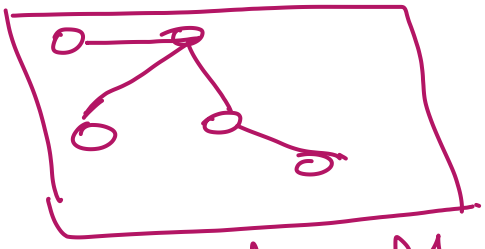
Model pipeline



↓

Automatic feature learning.

what is feature learning ?.



Input graph.

$f: u \rightarrow \mathbb{R}^d$.

d-dimension.

we define function f such that - if we create d-dimensional embedding for node such that similar node embedding will be closer.

There is two ways in which we can create feat node embedding.

1. Manually hand crafted: Traditional ML
2. Automatic learning: Called Representation learning.

Feature Generation:

1. Node level
2. link level
3. Graph level

There is two step in feature extraction:

1. Design the feature
2. Extract feature for training

model performance depends upon feature design. Traditional ML uses hand-designed features

Based on choice of task we need to design d-dimensional feature set.

Objects to generate d-dimensional feature set could be Nodes, edges, set of nodes, entire graphs.

Node level feature generation:

Given $G = (V, E)$

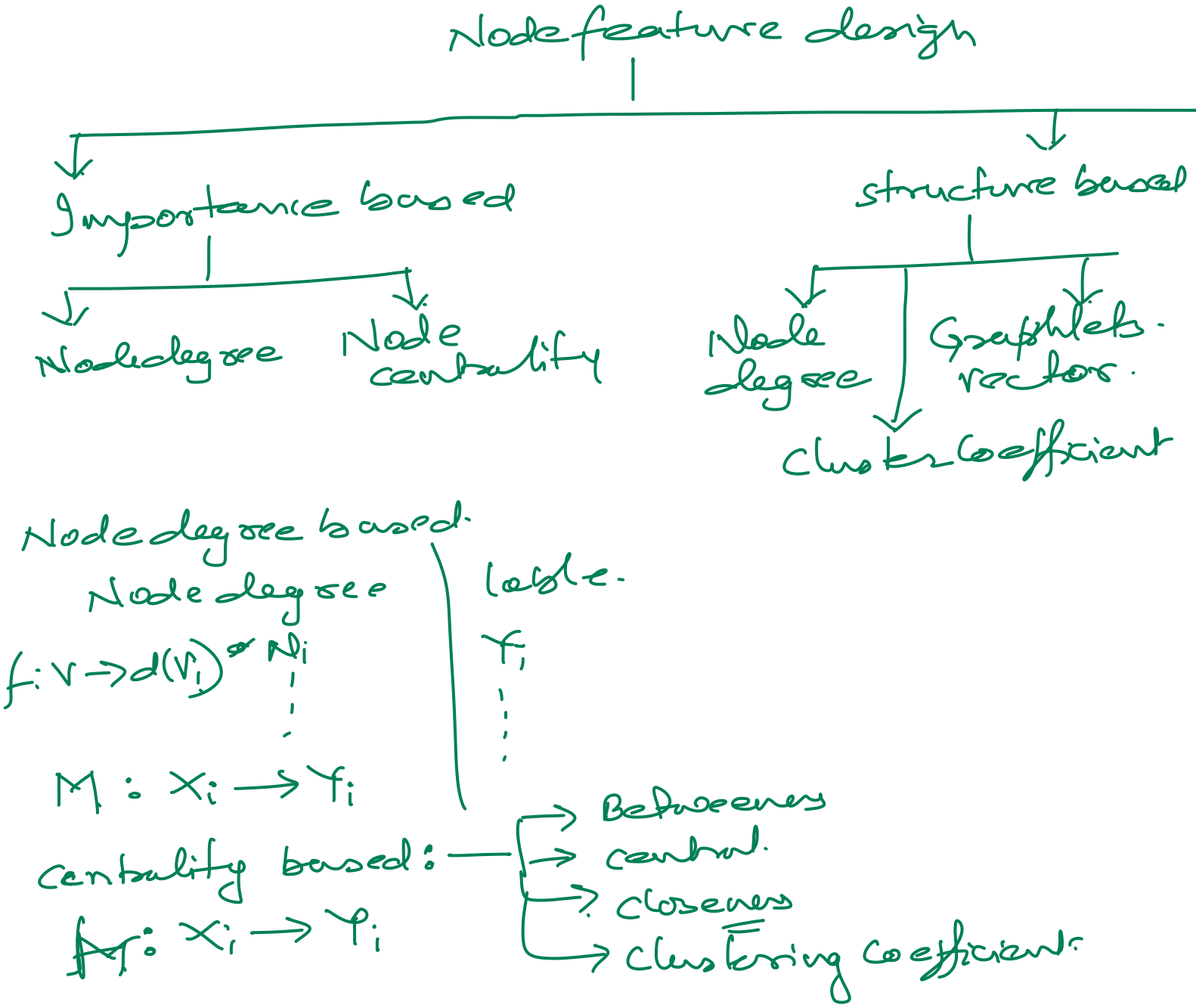
we need to learn a function

$$f: V \rightarrow \mathbb{R}^d$$

function f such that. It will take V(nodes) as input and generate d-dimensional real value vector.

Question: How to design this function:

when we say Node feature then that means how to convert relation/interaction/ structure into feature set. We do not mean node attribute or link attribute.



Node degree based.

Node degree

$f: V \to d(V_i)$ or N_i

$M : X_i \to Y_i$

Label.

Y_i

centrality based: $A : X_i \to Y_i$

- Betweenness centrality
- closeness
- clustering coefficient

Graphlet based are in other note.

Link level prediction task:

(A) Task is to predict link based on existing link.

(B) We need to design feature for pair of nodes

(C) At the testing time we need to take all node pair with not existing link and rank them. In this rank top k node pair will be predicted to have nodes

One of the way we can think of node features we take and concatinate that to form link feature but that will be very unsatisfactory because that will lose many important information related to link (Relationship) between two node.

Formulation of link predicting task can be think in two ways:

(1) For given network, remove random set of network and predict them back using ML. This is mostly static graph.

(2) Other formulation could be time evolving graph. eg citation network, social network etc where links evolves with time.

let say we take graph which evolves before time t_0 & t_0'. We take links and structure during this time and based on this we create rank list L of links that are predicted to appear between t_1 & t_1'.

We evaluate this model based on predicted and actual occuring links between t_1 & t_1'.

This kind of task is important for time evolving graph like social network, transactional network etc.

feature descriptor for pair of nodes:

One of the approach in designing feature descriptor is proximity based.

Let say we have pair of node (v_i, v_j)

we can define function like

$c(x, y) =$ no of common neighbour between (v_i, v_j)

The score c will be used used as basis for predicting links. e.g

sort c in decreasing order and select top k pair as prediction of links.

During test time (t_1, t_1') we actualy see which link actually apper and test the efficiency of model.

We will see 3 ways in which we can featurize the link prediction:

$v_1 = A$, $v_2 = B$

$N(A) = \{C\}$

$N(B) = \{C, D\}$

$u \in N(v_1) \cap N(v_2)$

$u \in \{C\} = \frac{1}{\log k_C} \Rightarrow k_C \Rightarrow$ degree of C

① Distance based: shortest path distance based. In this we define feature using pair of node and their hop distance.

But major problem in this is that this does not count the strength of relation with

neighbours like overlapping neighbours.
To overcome this we take overlapping neighbours as strength between pair of nodes.

$$\text{Common neigh} = N(v_i) \cap N(v_j)$$

for high degree nodes this may be dominating so we normalize this using Jaccard coeff.

$$J = \frac{|N(v_i) \cap N(v_j)|}{|N(v_i) \cup N(v_j)|}$$

There is other popular index is Adamic-Adar

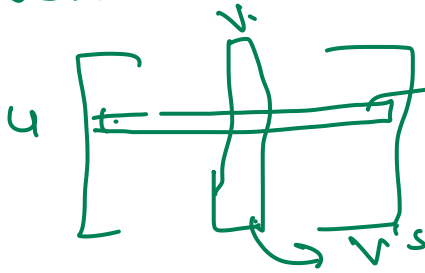
$$= \sum_{u \in (N(v_i) \cup N(v_j)} \frac{1}{\log(k_u)} \Rightarrow k_u = \text{degree of node } u.$$

This mean we will sum the inverse of log of degree 'u' node where u is common nodes between v_i & v_j.
This gives lower weightage to high degree common nodes.

But in local neighbour overlap is that- if nodes more than 2-hop away then there is will be situation where $|N(v_i) \cap N(v_j)| = 0$
But- there may be potential link between them
To overcome this limitation global neighbour -hood overlap concept used.
In this we take number of path of given length between two node. To find this lets understand what happen when we multiply Adjacency Matrix.

Txt- Trust deed
address proof
Id proof. Aeky form filling.

We know each element of A represent - path length '1' to reach that node and also this represents the neighbour of given node.



During multiplication common neighbour path gets added.

Suppose we need path length 2 between pair of node (u, v) then $u \to i \to v$ where i will be common neighbour of $u \& v$. So when we multiply two A then o/p simply add path length 2 between pairs.

$P_{uv}^{k} = A^{k}$ where P is power matrix of path length k between u, v. This will be equal to A^{k}.

For global neighbourhood over loops we take Katz index. This is count of all walk length between pair of nodes.

$$S = \sum_{l=1}^{\infty} \beta^{l} A^{l} = (I - \beta A)^{-1} - I$$

↓ GP series $\sum_{l=0}^{\infty} \beta^{l} A^{l}$.