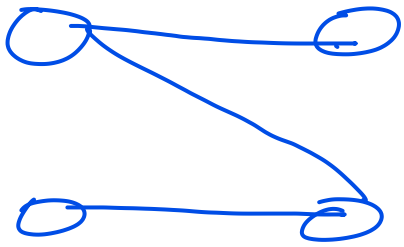
Graph: Graph is general language which describe/analyze relation/interaction amongentities

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

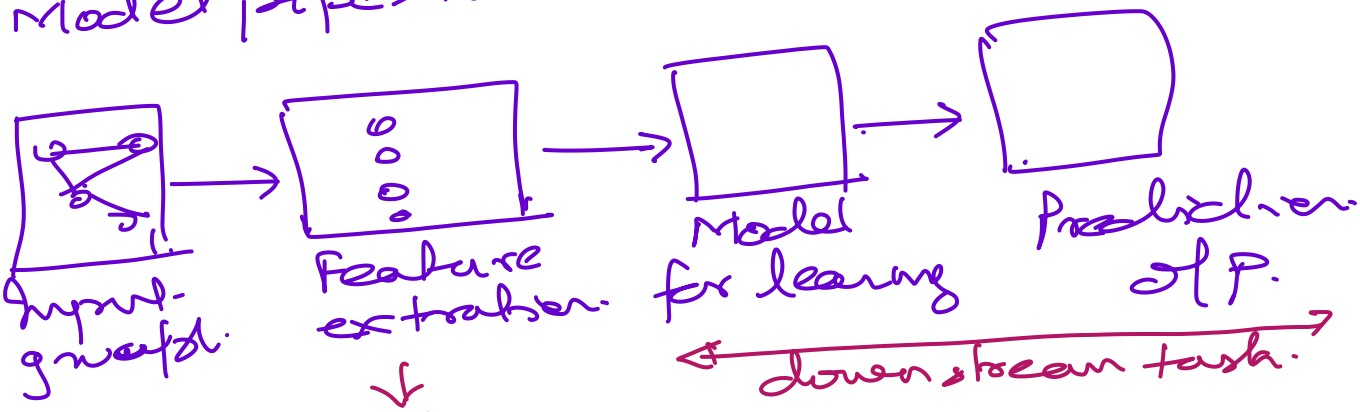
Entites which rich structural information can't be always represented as grid or sequence data. So these DL is not effective
We need more flexible DL which handles structural information ~~more~~ and applicable on these dataset.

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

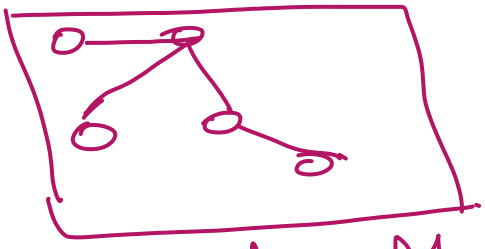
Model pipeline



↓

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 feature design

- Importance based
  - Node degree
  - Node centrality
- structure based
  - Node degree
  - Graphlets. vectors.
  - Cluster coefficient

Node degree based.

Node degree

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

$M : X_i \to Y_i$

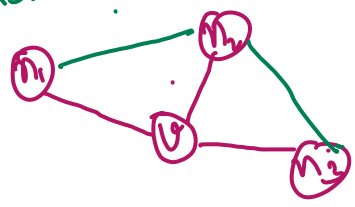
table.

Y_i

centrality based: $\to$ Betweeness centrality
$\to$ closeness
$\to$ Clustering coefficient.

$f: X_i \to Y_i$

Cluster coefficient: This looks how well neighbours of given node connected to each other.

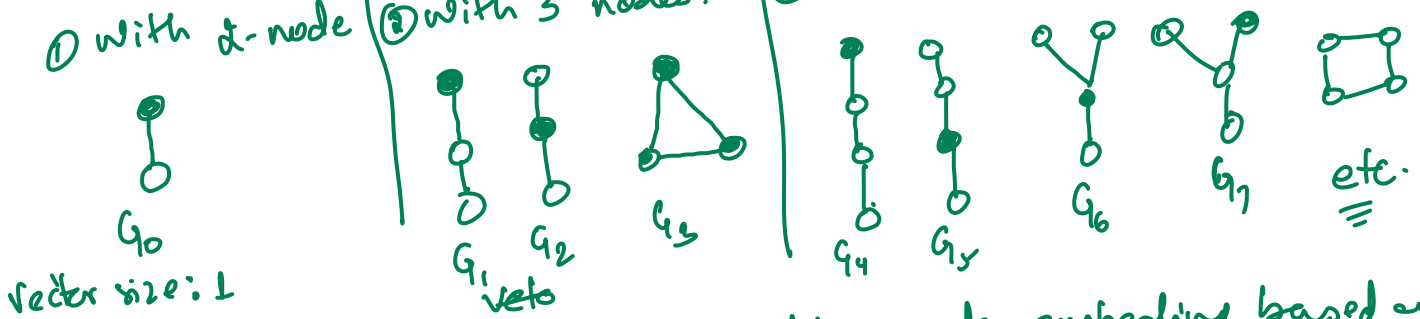


→ In this if all 3 nodes will be connected then total 3 triangle will be formed but here 2 triangle so $\frac{2}{3}$ is cc for node (v)

Node local topological signature is best described by Graphlets.

Graphlets are non isomorphic graph that can be formed by given number of nodes. In each sub-graph we can also indicate non-isomorphic point of interest.

① With 1-node ② With 3 nodes: ③ With 4-nodes



Vector size: 1

Vector size: 2

Then for given node where we need to create embedding based on graphlets.

Vector size for 3 node: 3
Vector size for 4 node: 15
Vector size for 5 node: 73

(Explain example from ppt.)
GDV

link feature set generation done in other note.

Graph feature set generation: Graph kernal

Our aim is to produce feature set that characterize entire graph. For this we use Graph kernels instead of graph feature vector.

Let $\phi(G)$ & $\phi(G')$ are two feature vector of graph.

Then $K(G,G') = \phi(G)^T \cdot \phi(G')$ ⇒ Dot product of two feature vector is called kernal. Here kernal measures similarity of two feature vector. Here we do not explicitly need to define ϕ to get kernal. There are method define to get graph kernal. Once kernal is known then we can use kernal SVM ML method for prediction task.

Graph kernal measures the similarity between two graph. To find graph kernal we need graph features.

Graph feature could be understood as bag-of-word concept for text data. In graph we can take node or node degree as bag of words but this lacks many structural locality missing in this feature set. So we need bag-of-"something" which can be more sophisticated than degree and used for feature generation.

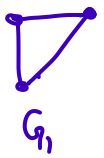
There are many ways where "something" can be used e.g

① Graphlet kernel
② Weisfeiler-Lehman kernel } discuss in detail

others could be
Random-Walk kernel
shortest-path graph kernel etc.

Graphlet kernel: This graphlet kernel is slightly different than node level graphlet. Two main difference:
① All nodes of graphlet need not be connected
② Graphlets are not rooted means making same node as root and based on that graphlets counted.

Example:
For 3-node



} But for node level feature we need to consider one node as root for which need to count orientation.

We count no. of graphlets present in graph to make feature vector

$$f(G)_i = \#(g_i \subseteq G) \text{ for } i = 1, 2, 3 \ldots n_k$$

From this feature vector we generate graph kernel

$$K(G, G') = f(G) \cdot f(G')$$

In case two graph is different size then data will be poorly skewed. we can handle this situation by normalization.

$$h_G = \frac{f(G)}{\text{sum}(f_G)}, \quad K(G, G') = h(G)^T \cdot h(G')$$

Limitation: ① Counting graphlets is expensive
② Subgraph isomorphism is NP-hard problem. Graphlet matching with graph is subgraph isomorphism.

Weisefeiler-Lehman kernel: This is more efficient- than graphlet counting.

Iteratively enhance the node features using hash table.

step-details in seperate document.

Based on refinement- $\phi(G)$ vector defined.

$K(G, G') = \phi(G) \cdot \phi(G') \rightarrow$ WL kernel

In this graph kernel creation is linear time complexity.