Lecture 2: Traditional ML in Graphs

2.1 Traditional Feature-based Methods: Node-level

Review

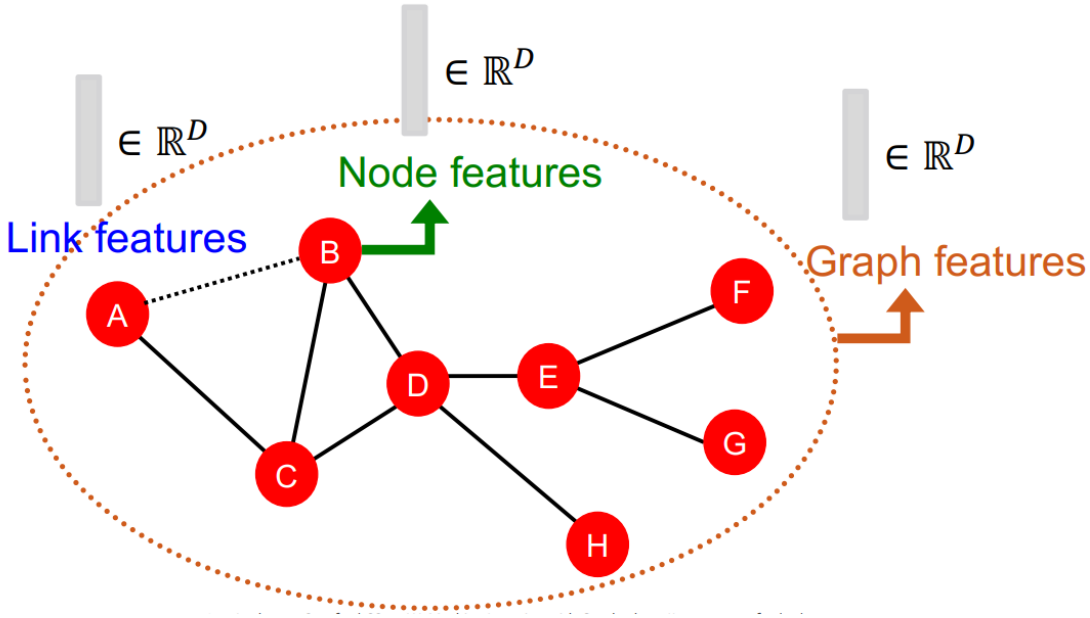
Traditional ML pipeline: design and obtain features for all training data regrading nodes, links and graphs, then train ML model to apply it to make a prediction.

Graph data itself has features, but we are also interested in features that describe its local structure features, that is, the feature describes the topology of the network and structure features can help to make more accurate predictions.

So generally there are two kinds of features:

- Structural features;
- Attributes and properties of nodes or relations.

- Design features for nodes/links/graphs
- Obtain features for all training data



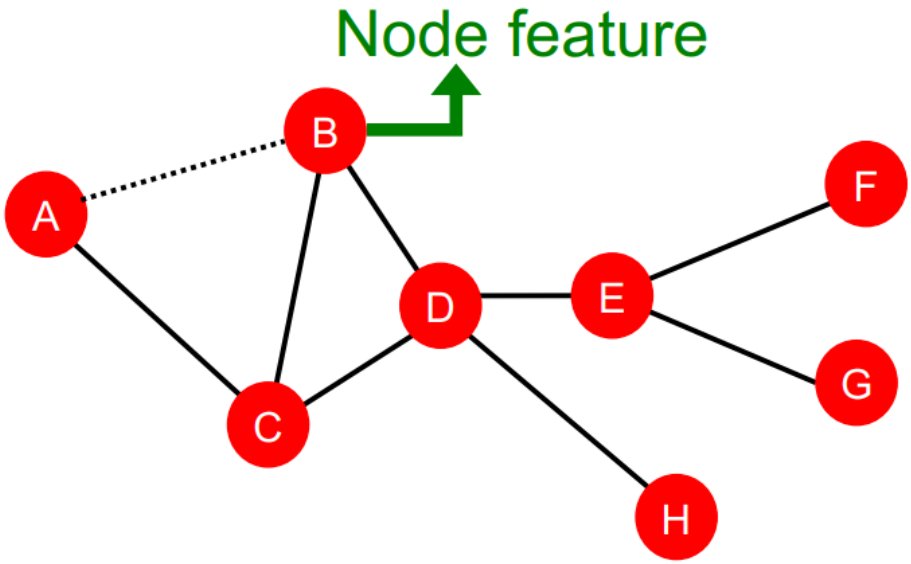
Design Choice

- Features: d-dimensional vectors;
- Objects: Nodes, edges, sets of nodes, entire graphs;
- Objective functions: what tasks are we aiming to solve?

Node-Level Features:

Goal: Characterize the structure and position of a node in the network:

- Node degree
- Node centrality
- Clustering coefficient
- Graphlets



Node Degree

The degree k_v of node v is the number of edges (neighboring nodes) the nodes has.

Node Centrality

- Why: node degree treat all neighboring nodes equally so it cannot capture their importance;
- Node Centrality c_v takes the node importance in a given graph into account;
- Different ways to model importance

1. Eigenvector

- **Eigenvector centrality:**
  - A node v is important if surrounded by important neighboring nodes $u \in N(v)$.
  - We model the centrality of node v as the sum of the centrality of neighboring nodes:

$$c_v = \frac{1}{\lambda} \sum_{u \in N(v)} c_u$$

λ is some positive constant

$$c_v = \frac{1}{\lambda} \sum_{u \in N(v)} c_u \quad \Longleftrightarrow \quad \lambda \boldsymbol{c} = \boldsymbol{A}\boldsymbol{c}$$

λ is some positive constant

- $\boldsymbol{A}$: Adjacency matrix $\boldsymbol{A}_{uv} = 1$ if $u \in N(v)$
- $\boldsymbol{c}$: Centrality vector

- We see that centrality is the **eigenvector**!

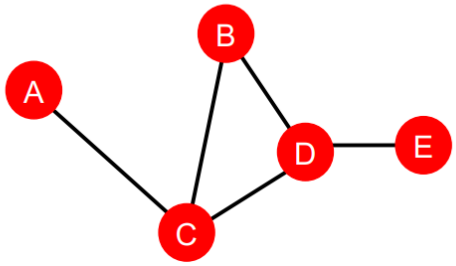
2. Betweenness

Betweenness centrality:

- A node is important if it lies on many shortest paths between other nodes.

$$c_v = \sum_{s \neq v \neq t} \frac{\#(\text{shortest paths betwen } s \text{ and } t \text{ that contain } v)}{\#(\text{shortest paths between } s \text{ and } t)}$$

- Example:



$c_A = c_B = c_E = 0$

$c_C = 3$

(A-C-B, A-C-D, A-C-D-E)

$c_D = 3$

(A-C-D-E, B-D-E, C-D-E)

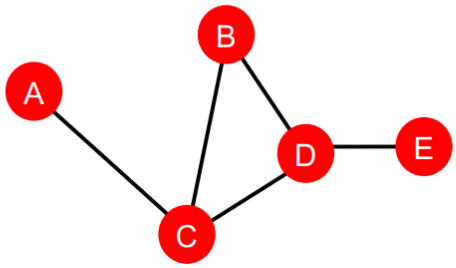
3. Closeness

Closeness centrality:

- A node is important if it has small shortest path lengths to all other nodes.

$$c_v = \frac{1}{\sum_{u \neq v} \text{shortest path length between } u \text{ and } v}$$

- **Example:**



$c_A = 1/(2+1+2+3) = 1/8$

(A-C-B, A-C, A-C-D, A-C-D-E)

$c_D = 1/(2+1+1+1) = 1/5$

(D-C-A, D-B, D-C, D-E)

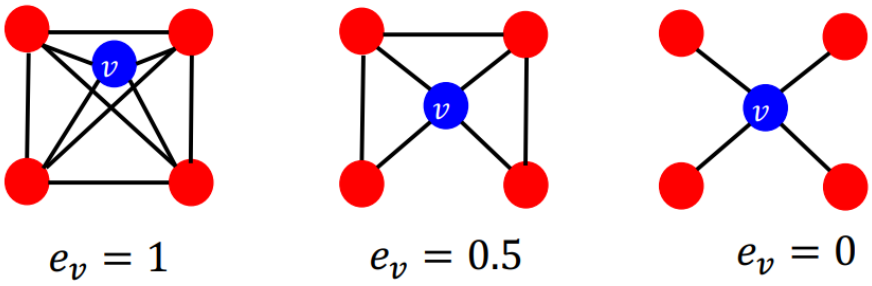
Clustering Coefficient

It measures how connected $v's$ neighboring nodes are:

$$e_v = \frac{\#(\text{edges among neighboring nodes})}{\binom{k_v}{2}} \in [0,1]$$

$\#(\text{node pairs among } k_v \text{ neighboring nodes})$

- **Examples:**

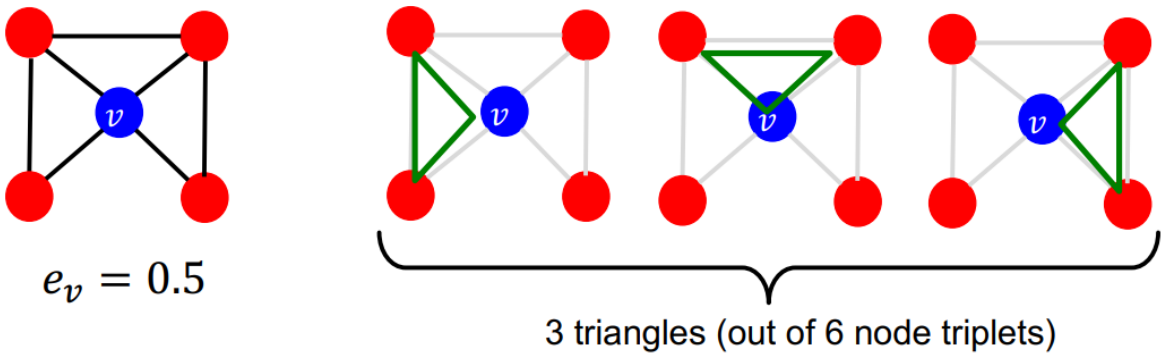


$e_v = 1$ $e_v = 0.5$ $e_v = 0$

The first example: $e_v = \frac{6}{6}$
The second example: $e_v = \frac{3}{6}$
The third example: $e_v = \frac{0}{6}$

Graphlets

Observation: Clustering coefficient counts the #(triangles) in the ego-network



$e_v = 0.5$

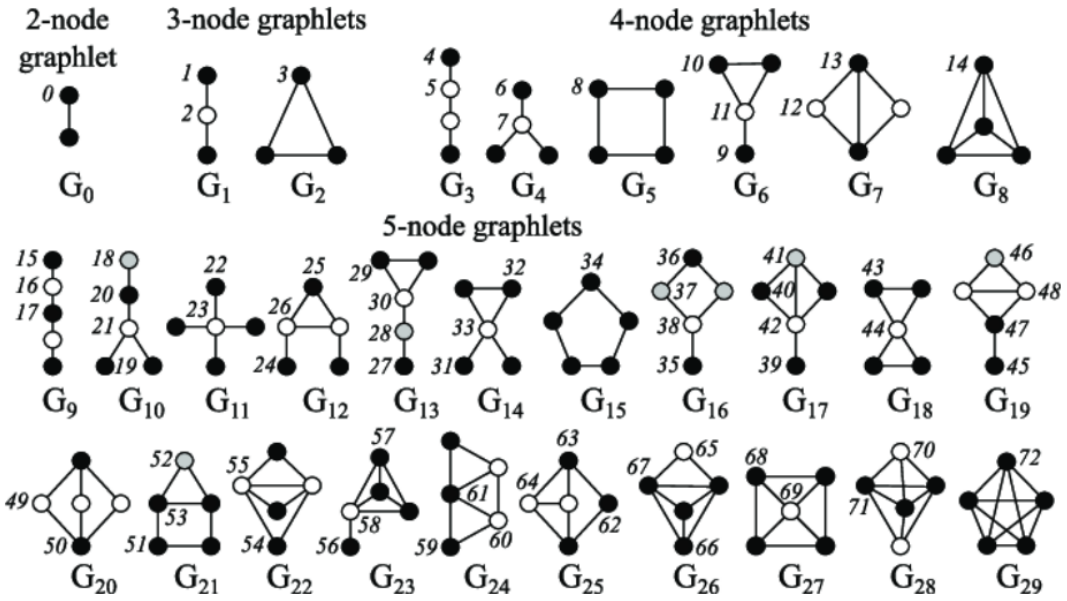
Ego-network of a given node is simply a network that is induced by the node itself and its neighbors. So it's basically degree 1 neighborhood network around a a given node.

We can generalize the above by counting #(pre-specified subgraphs, i.e., **graphlets**)

There are many such triangles in social networks, because it is conceivable that your friends may know each other through your introduction, thus constructing such a triangle/triple.

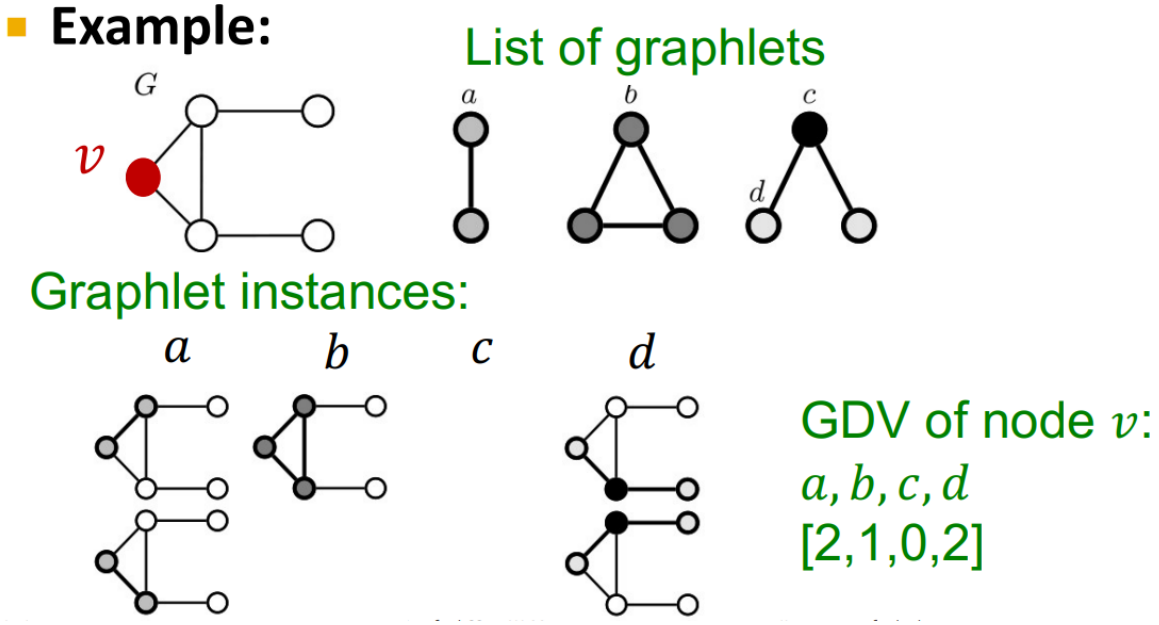
This triangle can be extended to some predefined subgraph pre-specified subgraph, such as the graphlet shown below:

Graphlets: Rooted connected non-isomorphic subgraphs:



- **Graphlet Degree Vector (GDV):** Graphlet-based features for nodes, it counts **graphlets** that a nods touches.;
- **Degree** counts **edges** that a node touches;
- **Clustering coefficient** counts **triangles** that a node touches

- **Example:**



Node-Level Summary

- Importance-based features:
  - Node degree
  - Different node centrality measures

  It can be used to predict celebrity users in a social network.
- Structure-based features:
  - Node degree
  - Clustering coefficient

- Graphlet count vector

It can be used to predict protein functionality in a PPI network.