Machine Learning in Graphs

Node & Link Features

Acknowledgement: Jure Leskovec, Stanford University

Overview

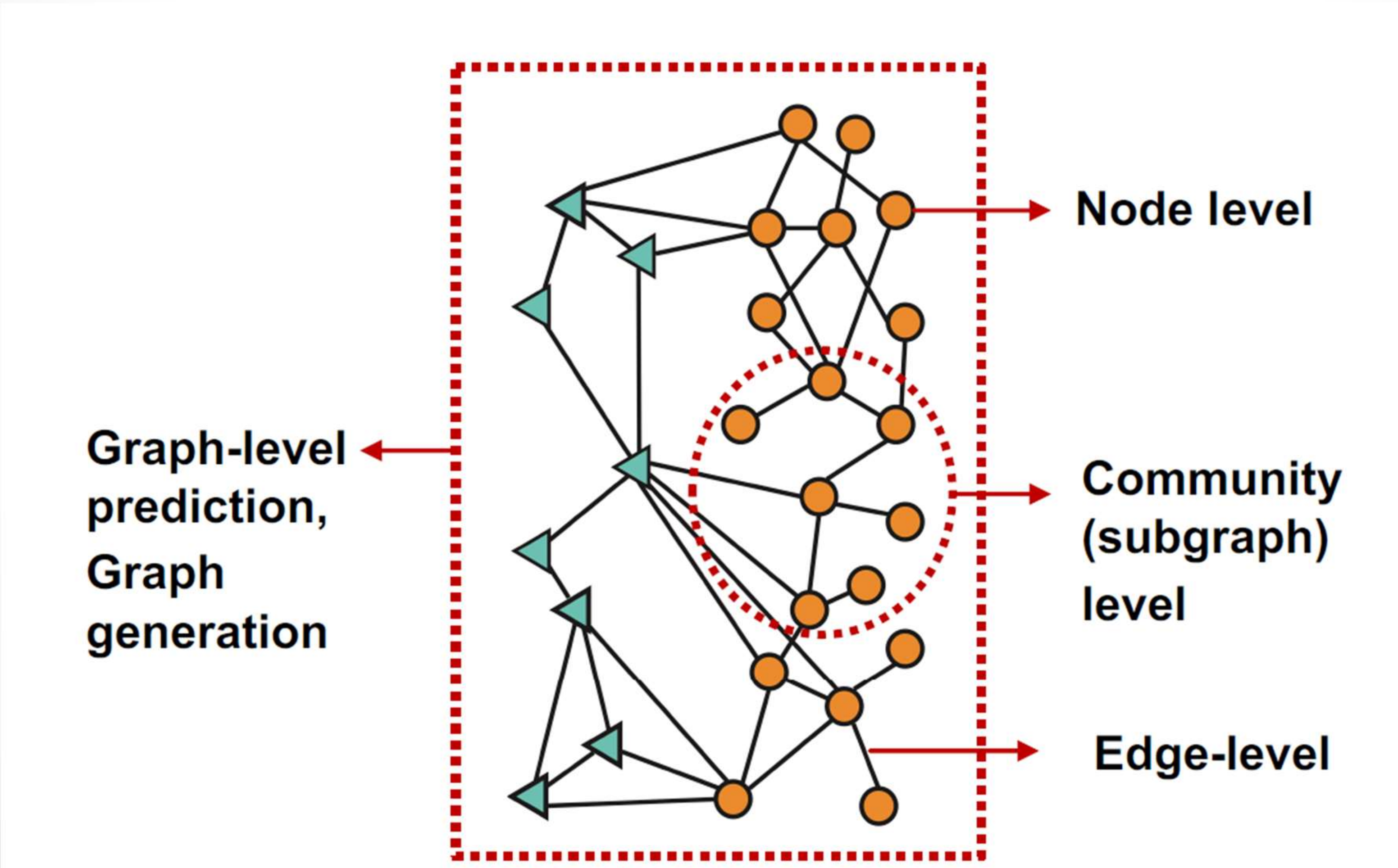
- Introduction
- Node-level Tasks and Features
- Link Prediction Task and Features

Overview

- Introduction
- Node-level Tasks and Features
- Link Prediction Task and Features

Graph Machine Learning - Applications

Different types of Tasks



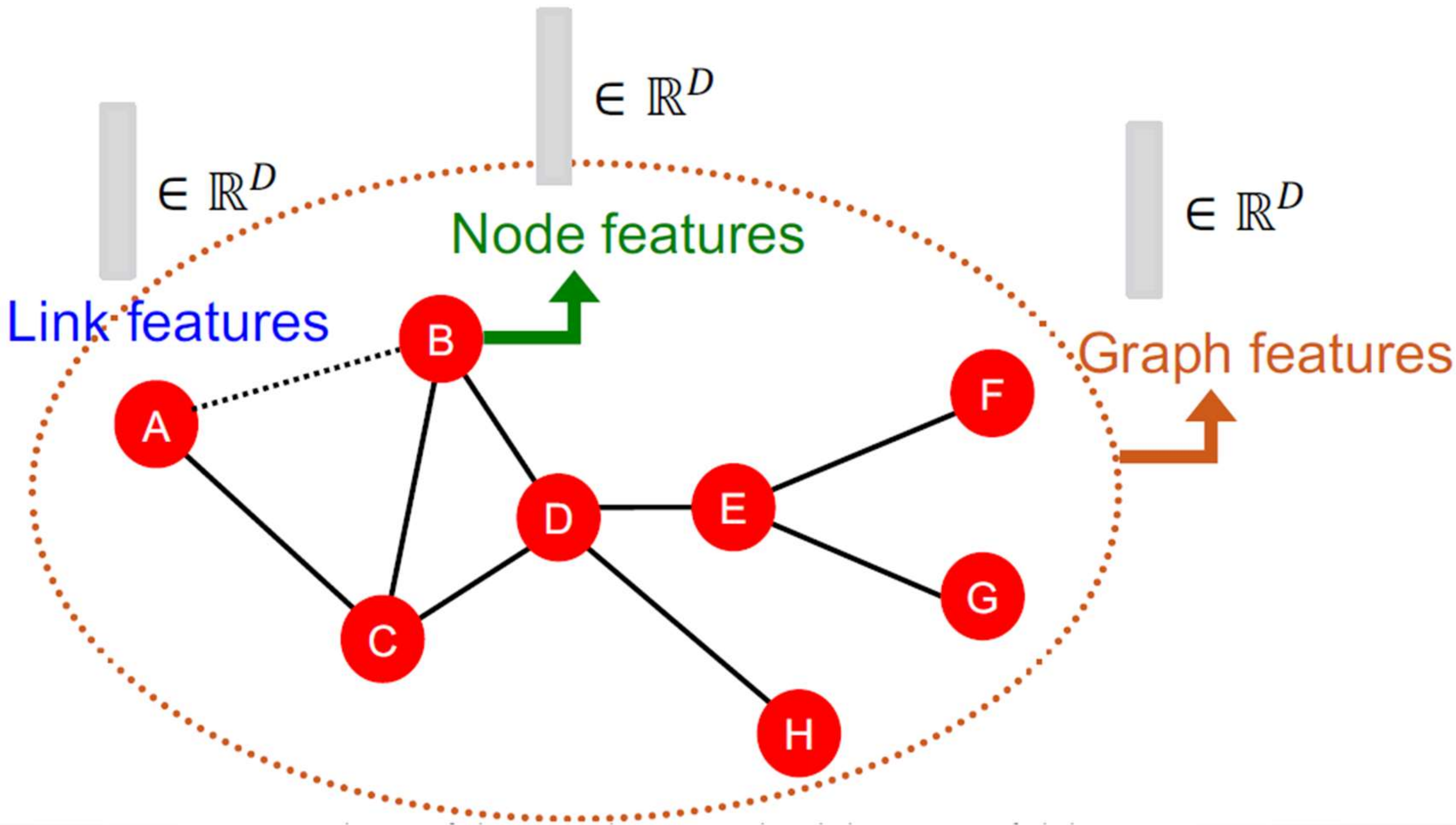
Classic Graph ML Tasks

- **Node classification**: Predict a property of a node
  - **Example:** Categorize online users / items
- **Clustering**: Detect if nodes form a community
  - **Example:** Social circle detection
- **Link prediction**: Predict whether there are missing links between two nodes
  - **Example:** Recommendation systems
- **Graph classification**: Categorize different graphs
  - **Example:** Molecule property prediction
- **Other tasks**:
  - **Graph generation**: Drug discovery
  - **Graph evolution**: Physical simulation

- *Only some of these tasks are related to Social Network Analysis – focus of this course*

Traditional ML Pipeline

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

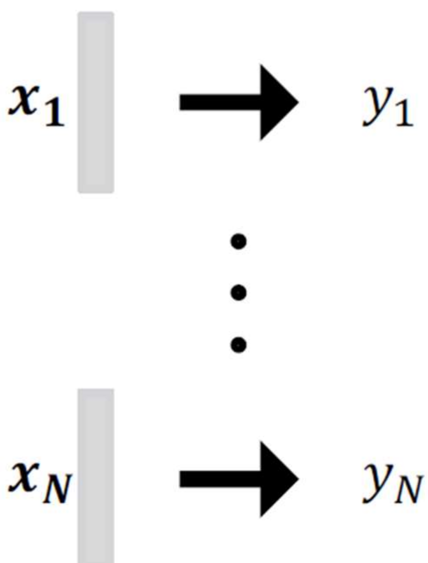


Features can be of two types:

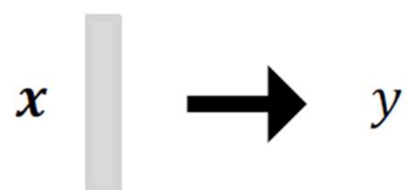
1) Attributes
2) Structural **(Focus of this lecture)**

Traditional ML Pipeline

- **Train an ML model:**
  - Random forest
  - SVM
  - Neural network, etc.

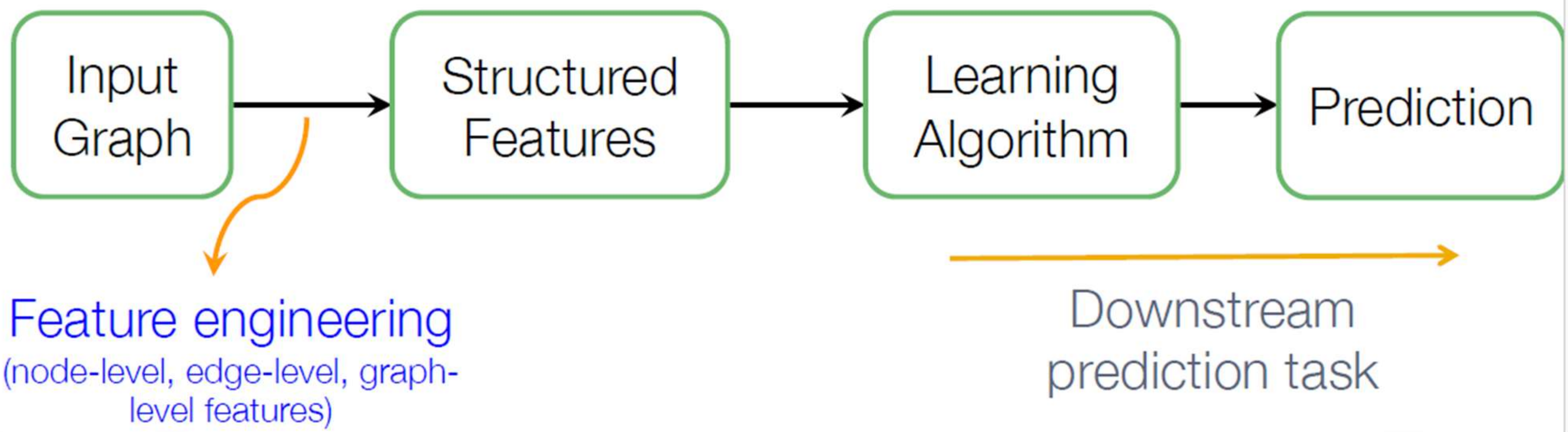


- **Apply the model:**
  - Given a new node/link/graph, obtain its features and make a prediction



Traditional ML Pipeline for Graphs

Given an input graph, extract node, link and graph-level features, learn a model (SVM, neural network, etc.) that maps features to labels.



Machine Learning in Graphs

- **Goal:** Make predictions for a set of objects
- **Design choices:**
  - **Features:** *d*-dimensional vectors
  - **Objects:** Nodes, edges, sets of nodes, entire graphs
  - **Objective function:** What task are we aiming to solve?
- **Feature design:**
  - Using effective features over graphs is the key to achieving good test performance.
  - Traditional ML pipeline uses hand-designed features.

- *For simplicity we focus on un-directed graphs*

Machine Learning in Graphs

Machine learning in graphs:

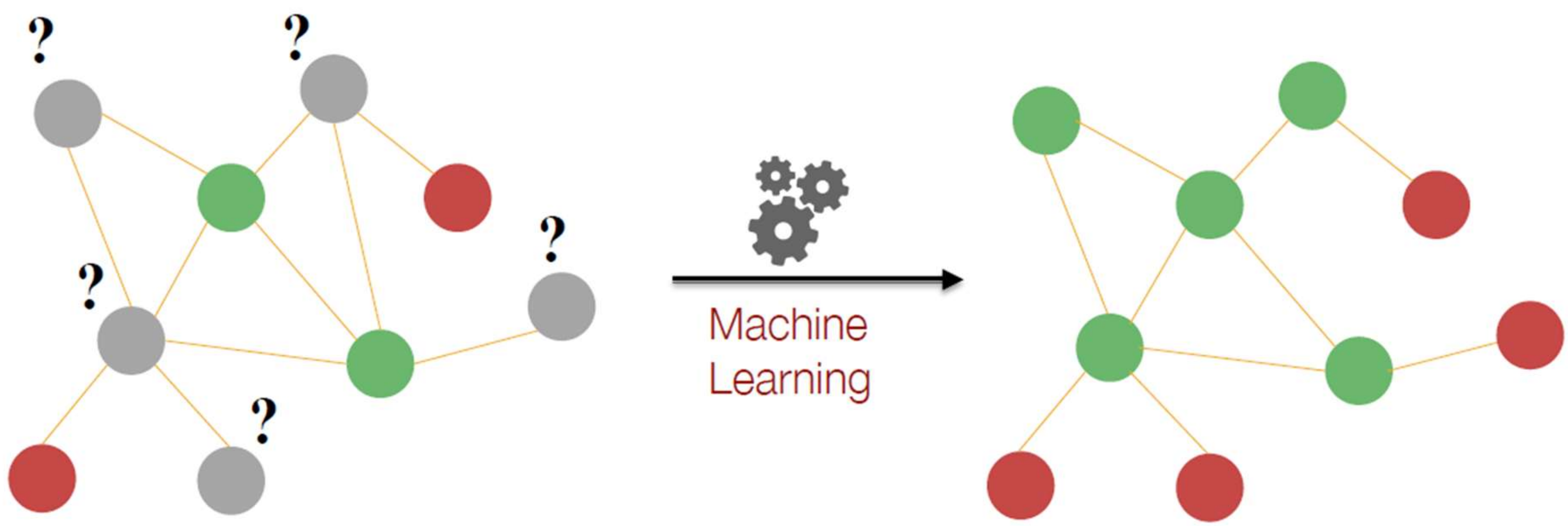
- Given: $G = (V, E)$
- Learn a function: $f : V \to \mathbb{R}$

How do we learn the function?

Overview

- Introduction
- Node-level Tasks and Features
- Link Prediction Task and Features

Node-level Tasks



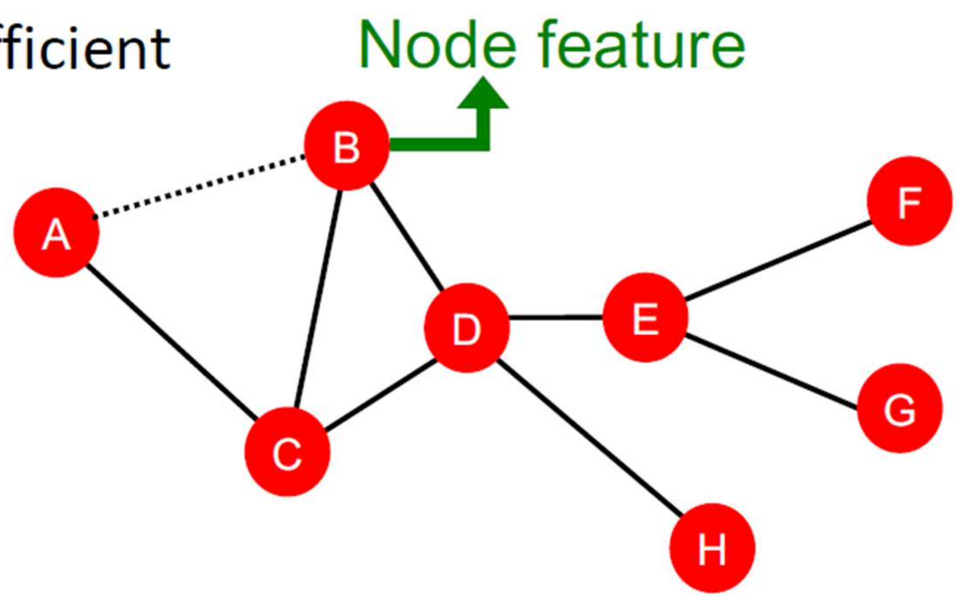
Node classification

ML needs features.

Node-level Features: Overview

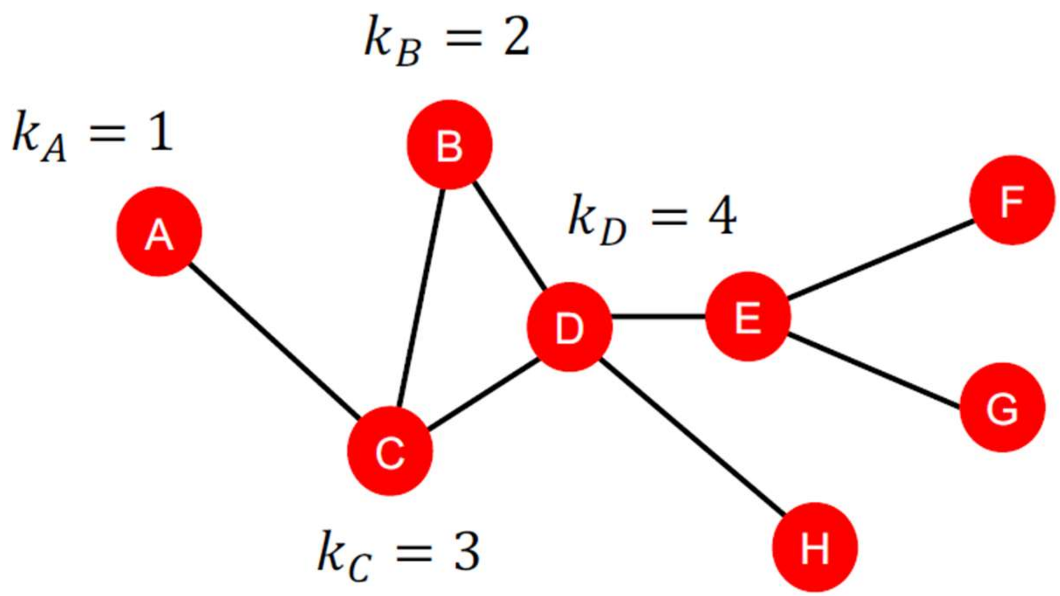
Goal: Characterize the structure and position of a node in the network:

- Node degree
- Node centrality
- Clustering coefficient
- Graphlets



Node Features: Node Degree

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



Node Features: Node Centrality

- Node degree counts the neighboring nodes without capturing their importance.
- Node centrality c_v takes the node importance in a graph into account
- **Different ways to model importance:**
  - Engienvector centrality
  - Betweenness centrality
  - Closeness centrality
  - and many others…

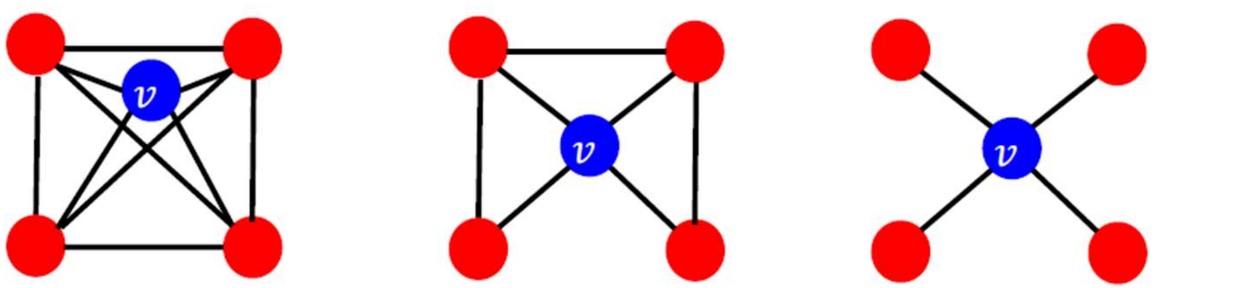
Clustering Coefficient

- Measures how connected $v's$ neighboring nodes are:

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

#(node pairs among k_v neighboring nodes)

- **Examples:**

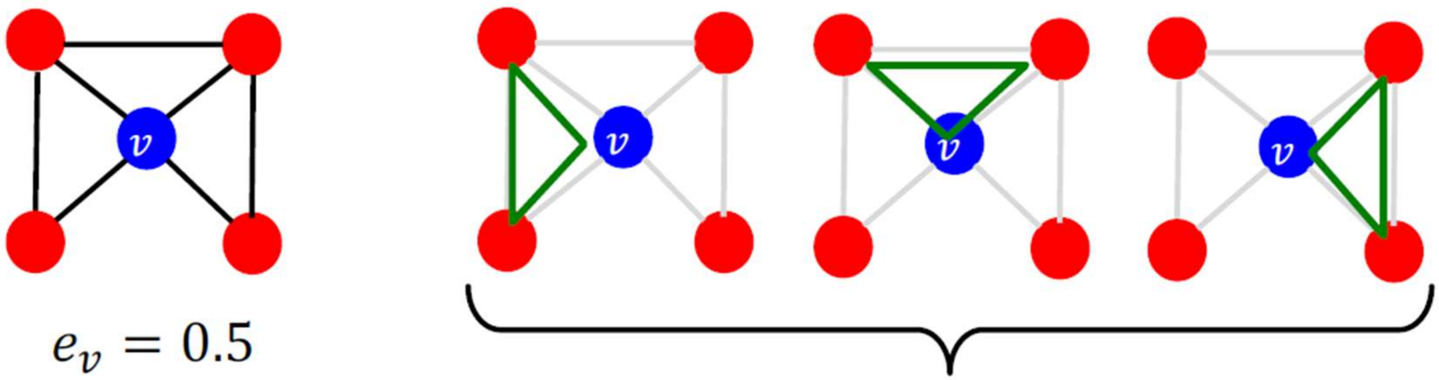


$e_v = 1$ $\quad$ $e_v = 0.5$ $\quad$ $e_v = 0$

Graphlets

Social Networks have a lot of triangles

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



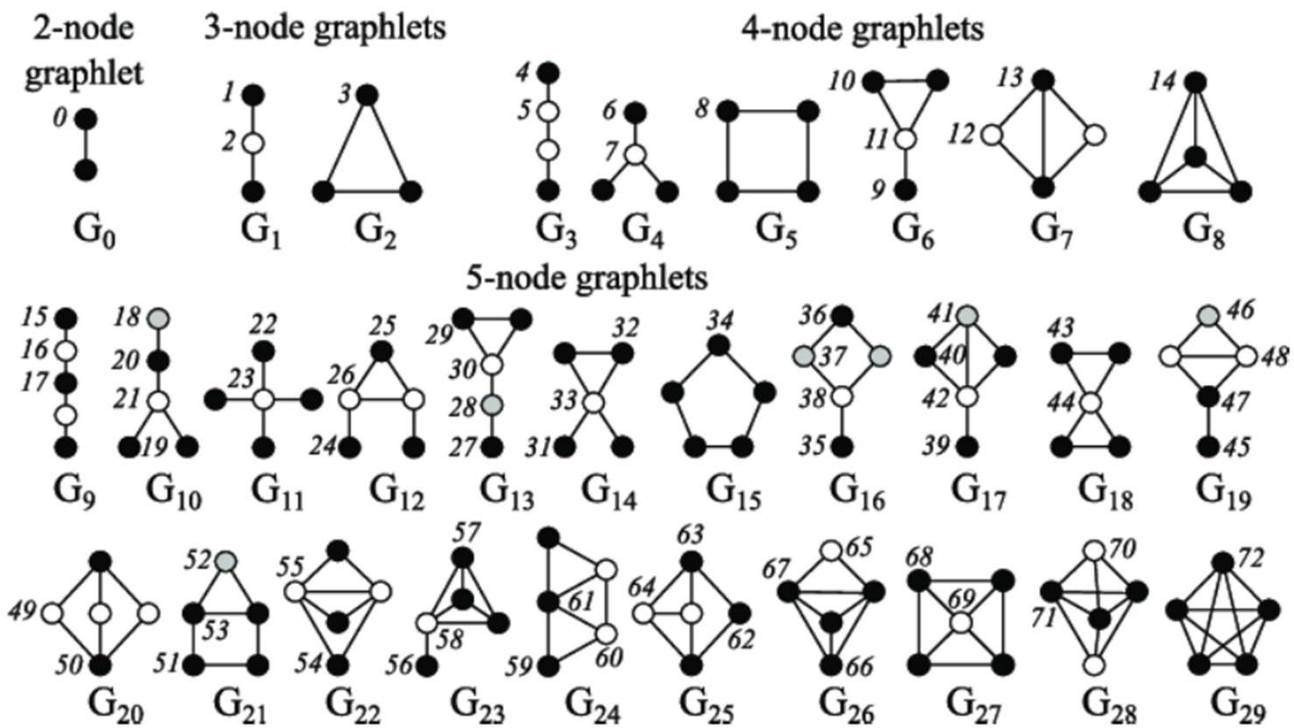
$e_v = 0.5$

3 triangles (out of 6 node triplets)

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

Graphlets

Graphlets: Rooted connected non-isomorphic subgraphs:



Graphlets

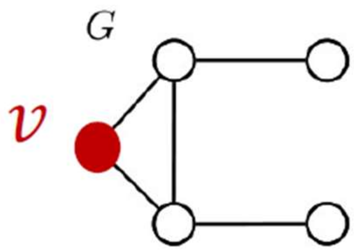
- **Graphlet Degree Vector (GDV):** Graphlet-base features for nodes

- **Degree** counts **#(edges)** that a node touches
- **Clustering coefficient** counts **#(triangles)** that a node touches.

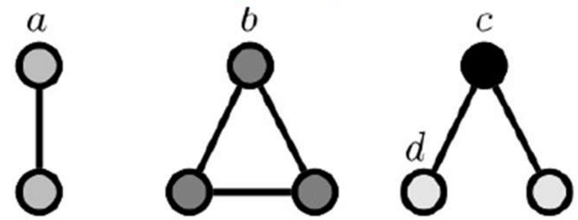
- **GDV** counts **#(graphlets)** that a node touches

Graphlets

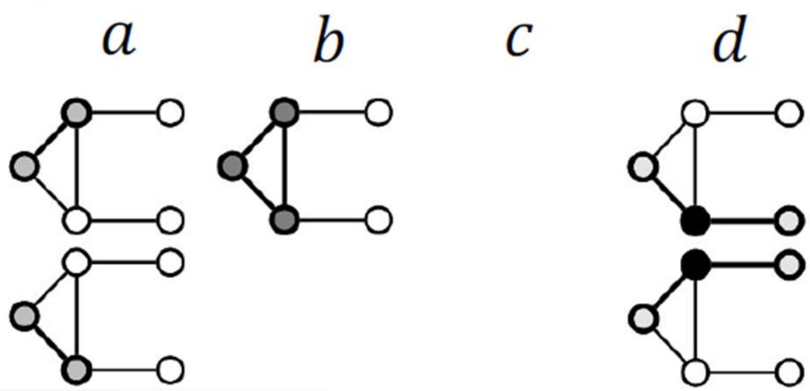
- **Graphlet Degree Vector (GDV)**: A count vector of graphlets rooted at a given node.
- **Example:**



List of graphlets



Graphlet instances:



GDV of node v:
a, b, c, d
[2,1,0,2]

Graphlets

- Considering graphlets on 2 to 5 nodes we get:
  - **Vector of 73 coordinates** is a signature of a node that describes the topology of node's neighborhood
  - Captures its interconnectivities out to a **distance of 4** hops
- Graphlet degree vector provides a measure of a **node's local network topology**:
  - Comparing vectors of two nodes provides a more detailed measure of local topological similarity than node degrees or clustering coefficient.

Node Level Features: Summary

- ❖ **We have introduced different ways to obtain node features.**
- ❖ **They can be categorized as:**
  - Importance-based features:
    - Node degree
    - Different node centrality measures
  - Structure-based features:
    - Node degree
    - Clustering coefficient
    - Graphlet count vector

Importance-based Features

- Capture the importance of a node is in a graph
  - Node degree:
    - Simply counts the number of neighboring nodes
  - Node centrality:
    - Models importance of neighboring nodes in a graph
    - Different modeling choices: eigenvector centrality,betweenness centrality, closeness centrality
- Useful for predicting influential nodes in a graph
- Example:
  - Predicting celebrity users in a social network

Structure-based Features

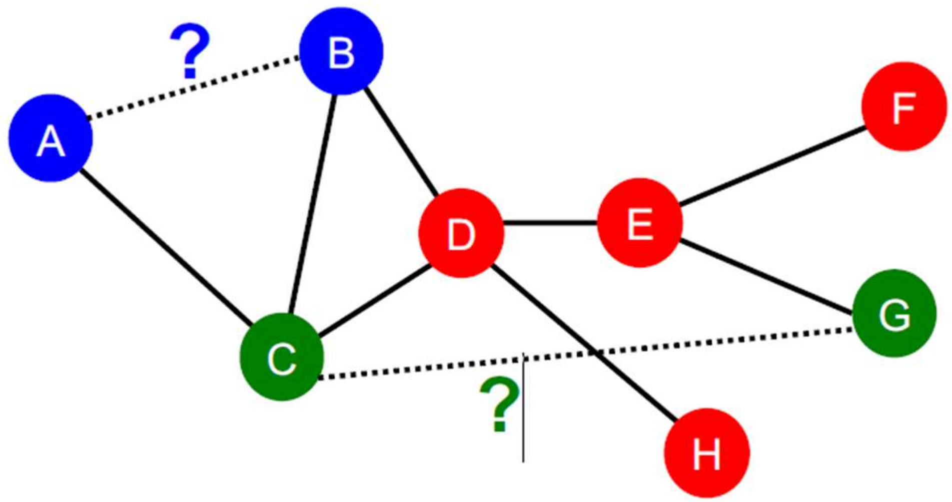
- Capture topological properties of local neighborhood around a node.
  - Node degree:
    - Counts the number of neighboring nodes
  - Clustering coefficient:
    - Measures how connected neighboring nodes are
  - Graphlet degree vector:
    - Counts the occurrences of different graphlets
- Useful for predicting a particular role a node plays in a graph
- Example:
  - Predicting protein functionality in a protein-protein interaction network.

Overview

- Introduction
- Node-level Tasks and Features
- Link Prediction Task and Features

Link-Level Prediction Task

- The task is to predict **new links** based on existing links.
- At test time, all node pairs (no existing links) are ranked, and top K node pairs are predicted.
- The key is to design features for **a pair of nodes**.

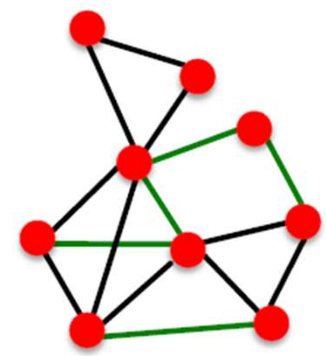


Just concatenating features of node 1 and node 2 will not be satisfactory - does not capture relationship between the 2 nodes

Link Prediction as a Task

Two formulations of the link prediction task:

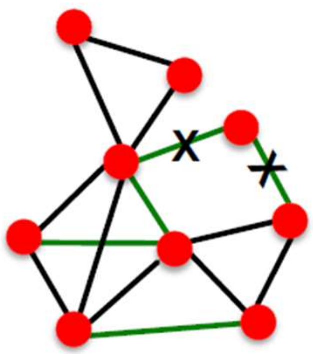
- **1) Links missing at random:**
  - Remove a random set of links and then aim to predict them
- **2) Links over time:**
  - Given $G[t_0, t_0']$ a graph on edges up to time t_0', **output a ranked list L** of links (not in $G[t_0, t_0']$) that are predicted to appear in $G[t_1, t_1']$
  - **Evaluation:**
    - $n = |E_{new}|$: # new edges that appear during the test period $[t_1, t_1']$
    - Take top n elements of L and count correct edges



1) Suitable for static networks (like Protein interaction networks)
2) Suitable for dynamic networks (like Social networks)

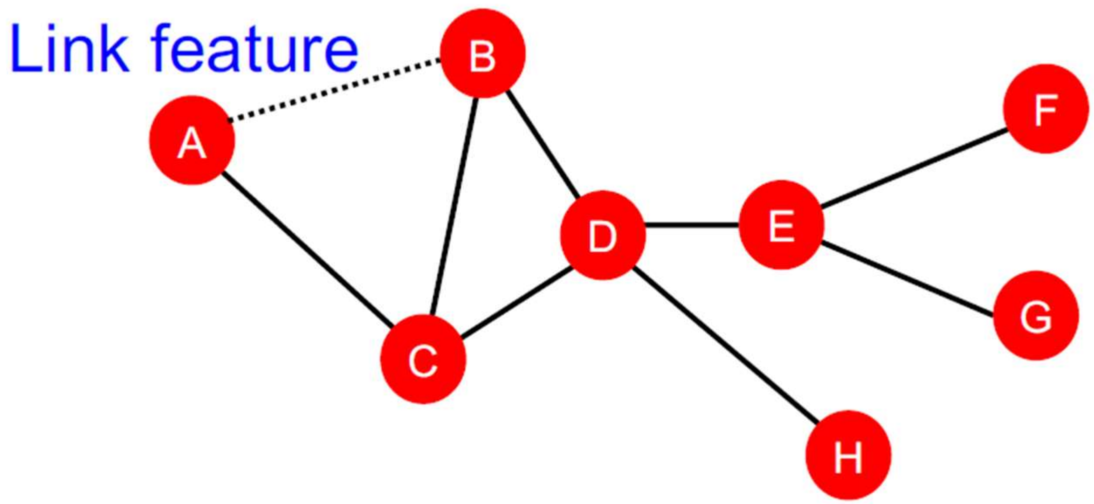
Link Prediction via Proximity

- **Methodology:**
  - For each pair of nodes *(x,y)* compute score *c(x,y)*
    - For example, *c(x,y)* could be the # of common neighbors of *x* and *y*
  - Sort pairs *(x,y)* by the decreasing score *c(x,y)*
  - **Predict top n pairs as new links**
  - **See which of these links actually appear in $G[t_1, t_1']$**



Link-Level Features: Overview

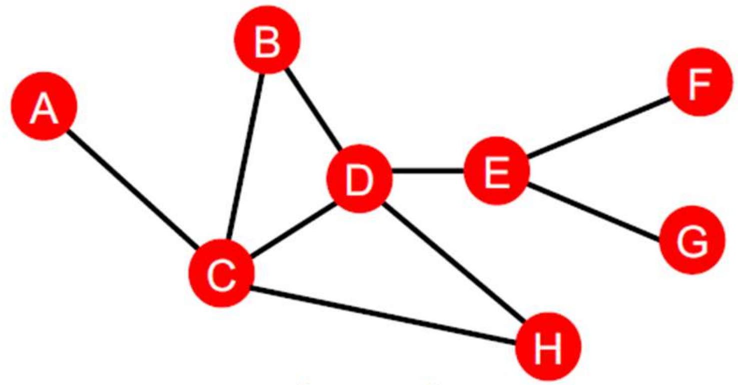
- Distance-based feature
- Local neighborhood overlap
- Global neighborhood overlap



Distance-based Features

Shortest-path distance between two nodes

- Example:



$$S_{BH} = S_{BE} = S_{AB} = 2$$
$$S_{BG} = S_{BF} = 3$$

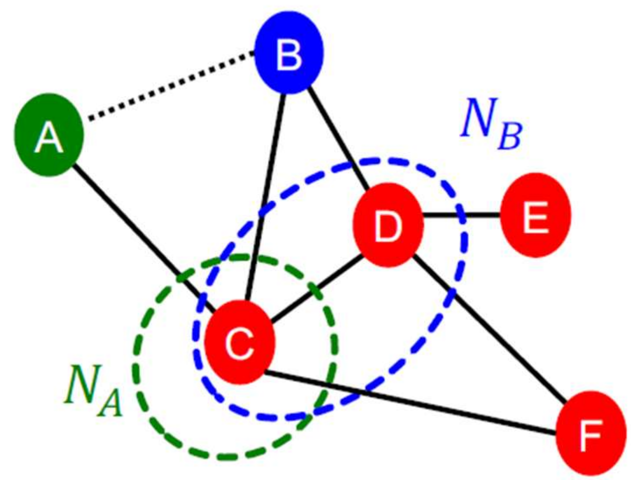
- However, this does not capture the degree of neighborhood overlap:
  - Node pair *(B, H)* has 2 shared neighboring nodes, while pairs *(B, E)* and *(A, B)* only have 1 such node.

Local Neighborhood Overlap

Captures # neighboring nodes shared between two nodes v_1 and v_2:

- **Common neighbors:** $|N(v_1) \cap N(v_2)|$
  - Example: $|N(A) \cap N(B)| = |\{C\}| = 1$
- **Jaccard's coefficient:** $\frac{|N(v_1) \cap N(v_2)|}{|N(v_1) \cup N(v_2)|}$
  - Example: $\frac{|N(A) \cap N(B)|}{|N(A) \cup N(B)|} = \frac{|\{C\}|}{|\{C, D\}|} = \frac{1}{2}$
- **Adamic-Adar index:**

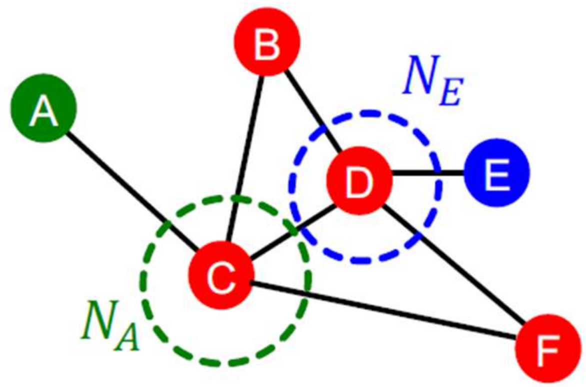
  $\sum_{u \in N(v_1) \cap N(v_2)} \frac{1}{\log(k_u)}$
  - Example: $\frac{1}{\log(k_C)} = \frac{1}{\log 4}$



- 1/log(Sum of degree of common neighbors)
- Importance of a neighbor decrease with increase in its degree
- Works well in Social Networks

Global Neighborhood Overlap

- **Limitation of local neighborhood features:**
  - Metric is always zero if the two nodes do not have any neighbors in common.



$$N_A \cap N_E = \phi$$
$$|N_A \cap N_E| = 0$$

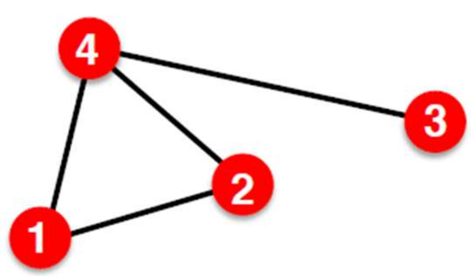
  - However, the two nodes may still potentially be connected in the future.
- **Global neighborhood overlap** metrics resolve the limitation by considering the entire graph.

Global Neighborhood Overlap

- **Katz index:** count the number of paths of all lengths between a given pair of nodes.
- **Q: How to compute #paths between two nodes?**
- Use **powers of the graph adjacency matrix**!

Power of Adjacency Matrices

- **Computing #paths between two nodes**
  - **Recall:** $A_{uv} = 1$ if $u \in N(v)$
  - Let $\boldsymbol{P}_{\boldsymbol{uv}}^{(\boldsymbol{K})}$ = #paths of length $\boldsymbol{K}$ between $\boldsymbol{u}$ and $\boldsymbol{v}$
  - We will show $\boldsymbol{P}^{(\boldsymbol{K})} = \boldsymbol{A}^{\boldsymbol{k}}$
  - $\boldsymbol{P}_{\boldsymbol{uv}}^{(\boldsymbol{1})}$ = #paths of length 1 (direct neighborhood) between u and $v = \boldsymbol{A}_{\boldsymbol{uv}}$



$\boldsymbol{P}_{\boldsymbol{12}}^{(\boldsymbol{1})} = \boldsymbol{A}_{\boldsymbol{12}}$

$$A = \begin{pmatrix} 0 & \boxed{1} & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

Power of Adjacency Matrices

- **How to compute $P_{uv}^{(2)}$?**
  - **Step 1:** Compute **#paths** of length 1 **between each of u's neighbor and v**
  - **Step 2**: **Sum up** these #paths across u's neighbors
  - $P_{uv}^{(2)} = \sum_i A_{ui} * P_{iv}^{(1)} = \sum_i A_{ui} * A_{iv} = A_{uv}^2$

Node **1**'s neighbors

#paths of length 1 between Node **1**'s neighbors and Node **2**

$P_{12}^{(2)} = A_{12}^2$

$$A^2 = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix} \times \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 2 & 1 & 1 & 1 \\ 1 & 2 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 3 \end{pmatrix}$$

Power of adjacency

Katz Index

- **Katz index:** count the number of paths of all lengths between a pair of nodes.

- How to compute #paths between two nodes?
- Use **adjacency matrix powers**!
  - $\boldsymbol{A}_{uv}$ specifies #paths of length 1 (direct neighborhood) between u and v.
  - $\boldsymbol{A}_{uv}^{2}$ specifies #paths of **length 2** (neighbor of neighbor) between u and v.
  - And, $\boldsymbol{A}_{uv}^{\boldsymbol{l}}$ specifies #paths of **length $\boldsymbol{l}$**.

Katz Index

- **Katz index** between v_1 and v_2 is calculated as

Sum over *all path lengths*

$$S_{v_1 v_2} = \sum_{l=1}^{\infty} \boxed{\beta^l} \boxed{A_{v_1 v_2}^l}$$

#paths of length l between v_1 and v_2

$0 < \beta < 1$: discount factor

- Katz index matrix is computed in closed-form:

$$\boldsymbol{S} = \sum_{i=1}^{\infty} \beta^i \boldsymbol{A}^i = \underbrace{(\boldsymbol{I} - \beta \boldsymbol{A})^{-1}}_{= \sum_{i=0}^{\infty} \beta^i \boldsymbol{A}^i} - \boldsymbol{I},$$

by geometric series of matrices

Link-Level Features: Summary

- **Distance-based features:**
  - Uses the shortest path length between two nodes but does not capture how neighborhood overlaps.
- **Local neighborhood overlap:**
  - Captures how many neighboring nodes are shared by two nodes.
  - Becomes zero when no neighbor nodes are shared.
- **Global neighborhood overlap:**
  - Uses global graph structure to score two nodes.
  - Katz index counts #paths of all lengths between two nodes.

References

- Social Network Analysis Tanmoy Chakraborty. Chapter 9