Machine Learning on Graphs

COMP9312_23T2

UNSW SYDNEY

Outline

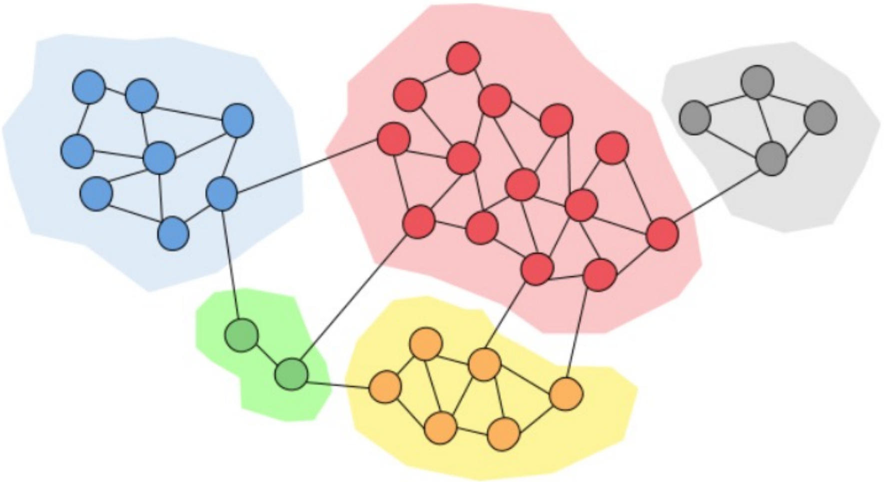
- Machine Learning on Graphs
- Node Feature Engineering

Data Structure & Algorithms

Case studies on Community Detection:

Connected Component, K-Core, K-Truss, Clique, …

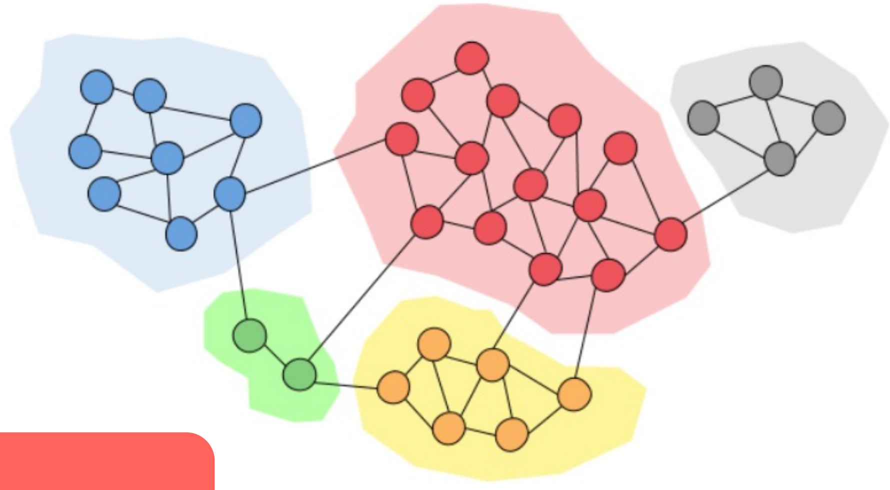
Clustering/partition algorithms, …



Step 1: Formulate problem → Step 2: Design Algorithm

It is hard in many applications.

Learning-based Algorithms

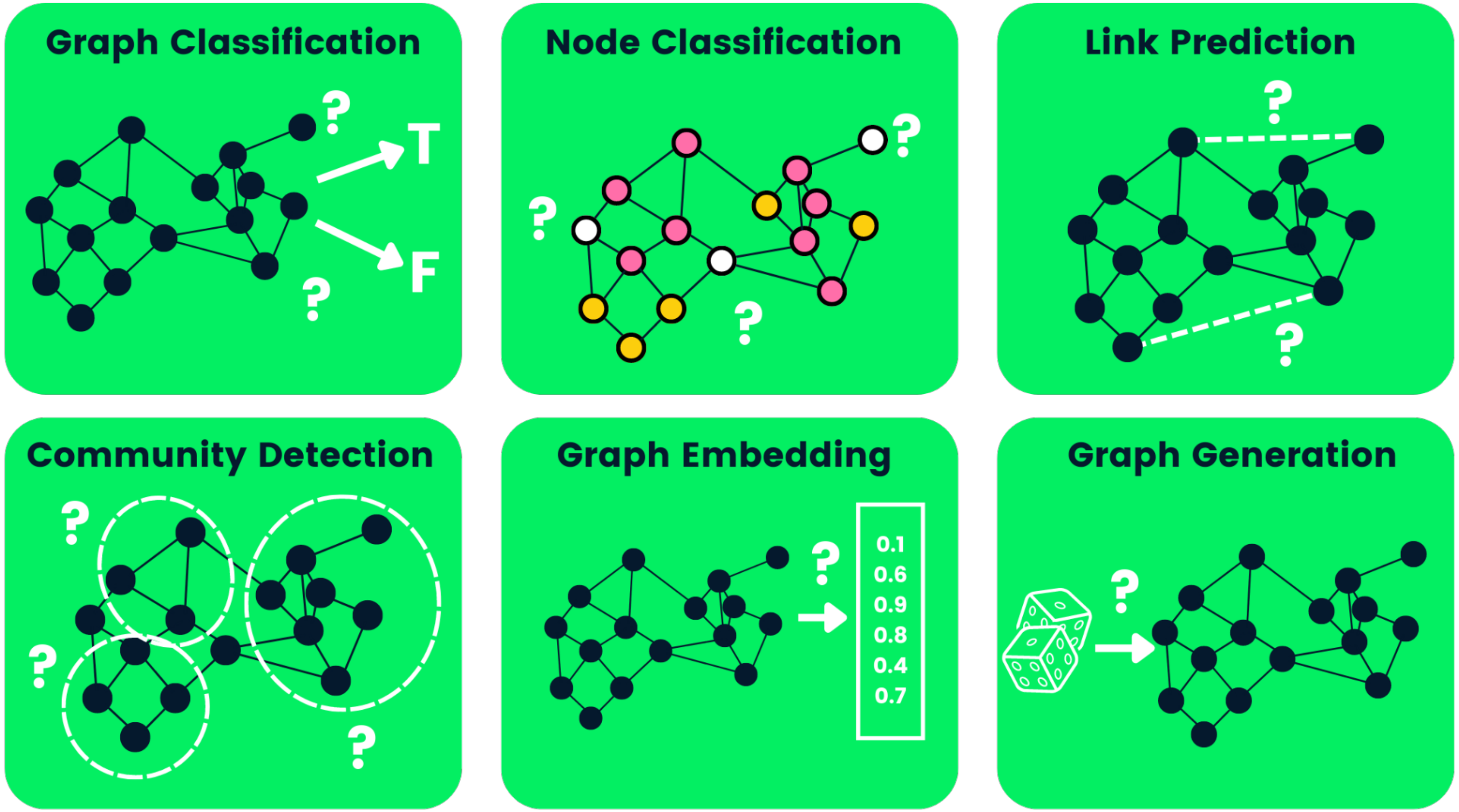


- It is hard to define a good community.
- It is not hard to judge a community.

Step 1: Decide parameters → Step 2: Define loss function → Step 3: Training

Efficiency VS Effectiveness

Application



https://www.datacamp.com/tutorial/comprehensive-introduction-graph-neural-networks-gnns-tutorial

Application

Node classification: Predict a property of a node

Example: Categorize online users / items

Link prediction: recommendation

Example: Knowledge graph completion

Graph classification: Categorize different graphs

Example: Molecule property prediction

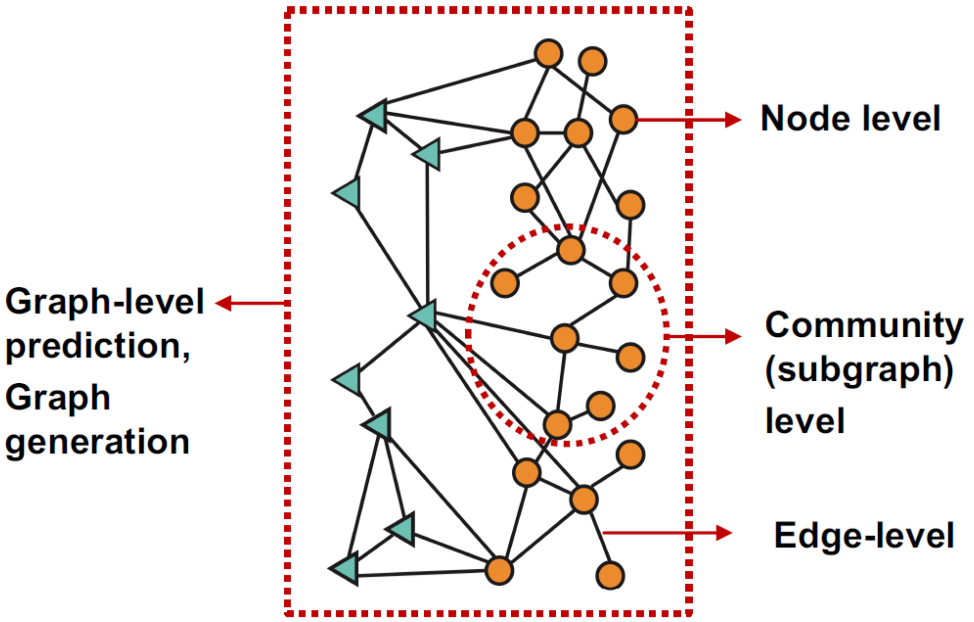
Clustering: Detect if nodes form a community

Example: Social circle detection

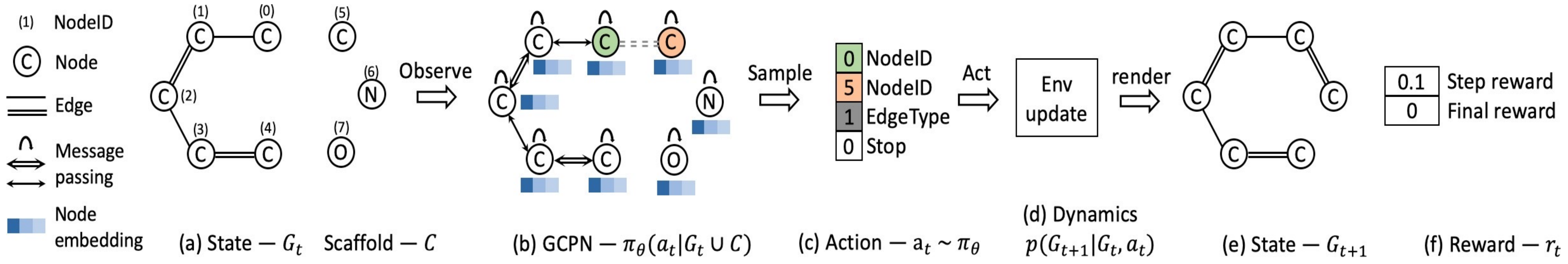
Other tasks:

Graph generation: Drug discovery

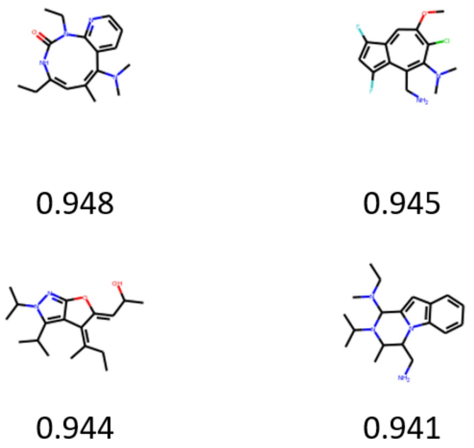
Graph evolution: Physical simulation



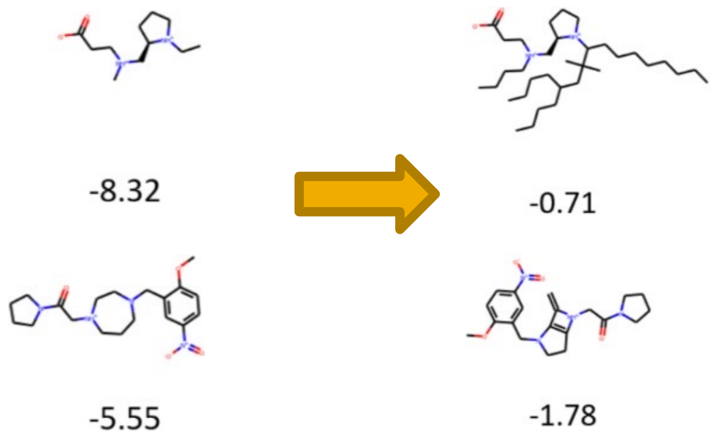
Application: Molecule Generation



Use case 1: Generate novel molecules with high drug likeness



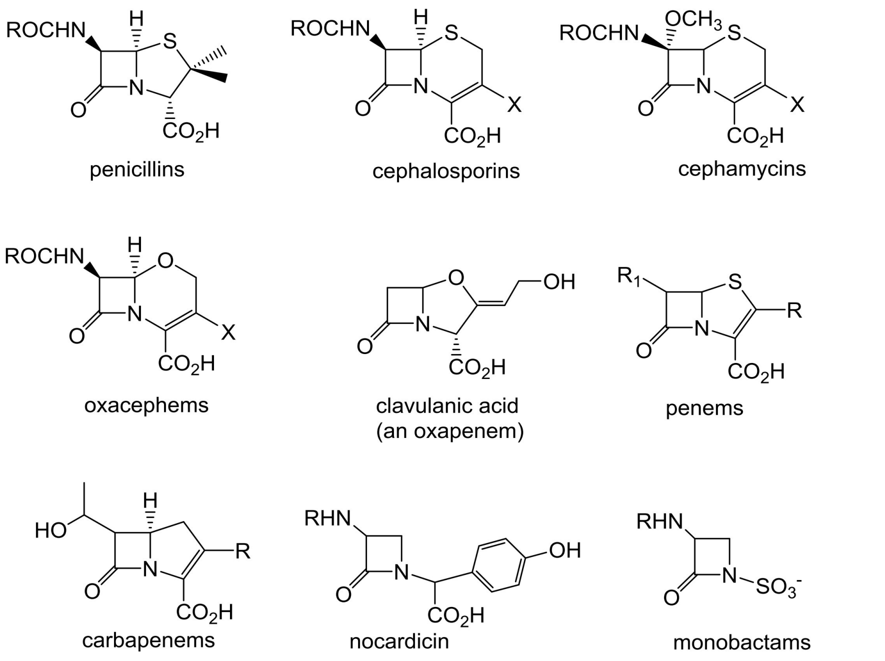
Use case 2: Optimize existing molecules to have desirable properties



Application: Drug Discovery

Antibiotics are small molecular graphs

- Nodes: Atoms
- Edges: Chemical bonds



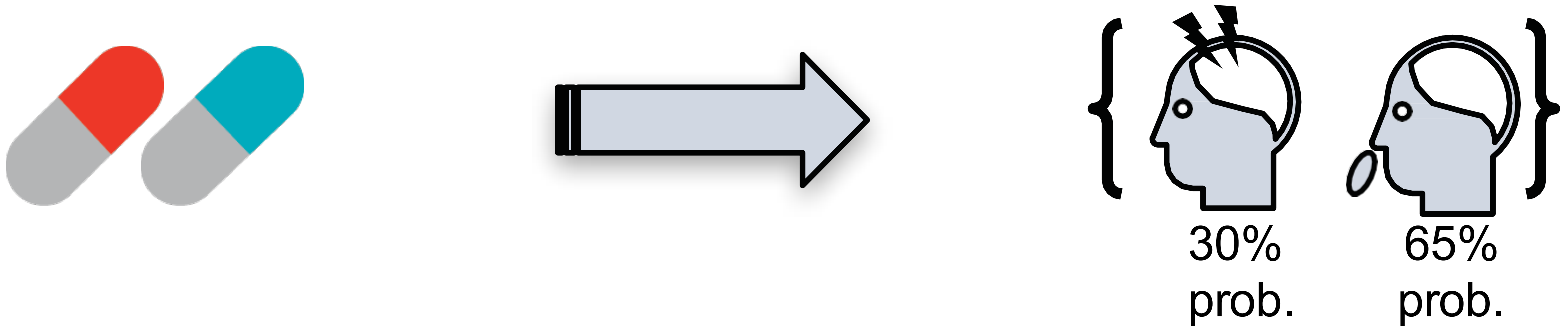
Konaklieva, Monika I. "Molecular targets of β-lactam-based antimicrobials: beyond the usual suspects." Antibiotics 3.2 (2014): 128-142.

Application: Drug Side Effects

Many patients take multiple drugs to treat complex or co-existing diseases:

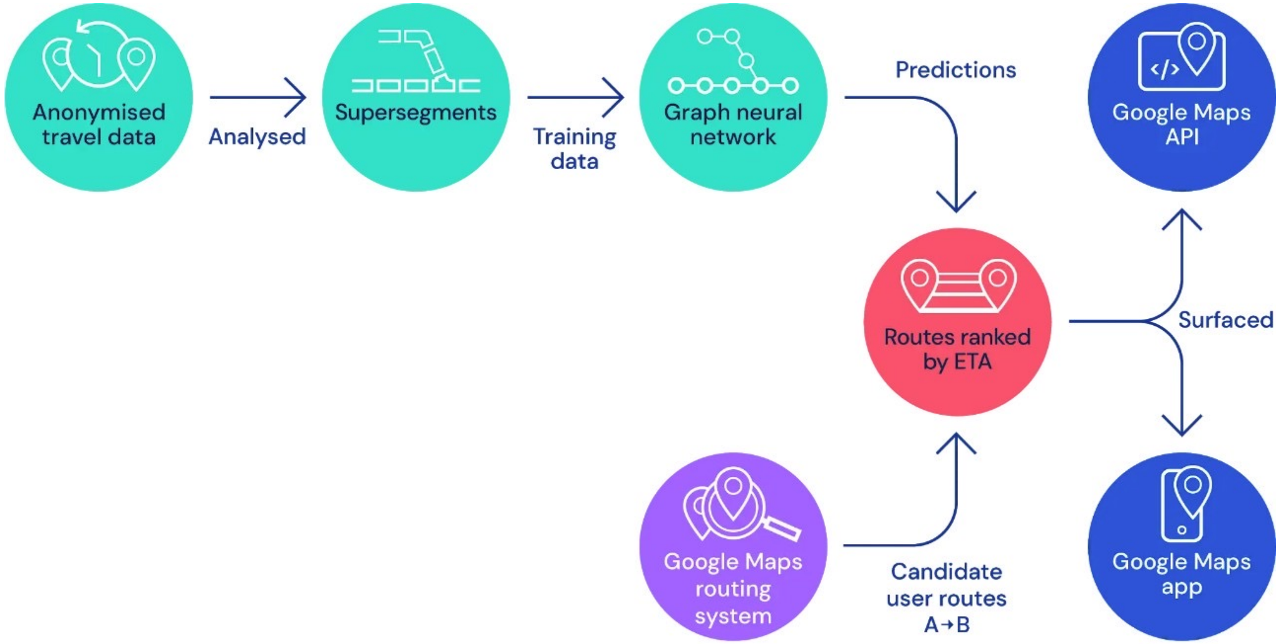
- 46% of people ages 70-79 take more than 5 drugs
- Many patients take more than 20 drugs to treat heart disease, depression, insomnia, etc.

Task: Given a pair of drugs predict adverse side effects



Application: Google Map

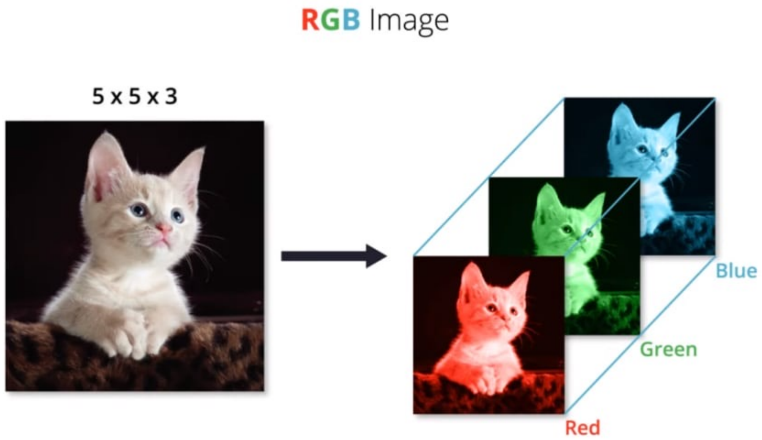
Predict via Graph Neural Networks

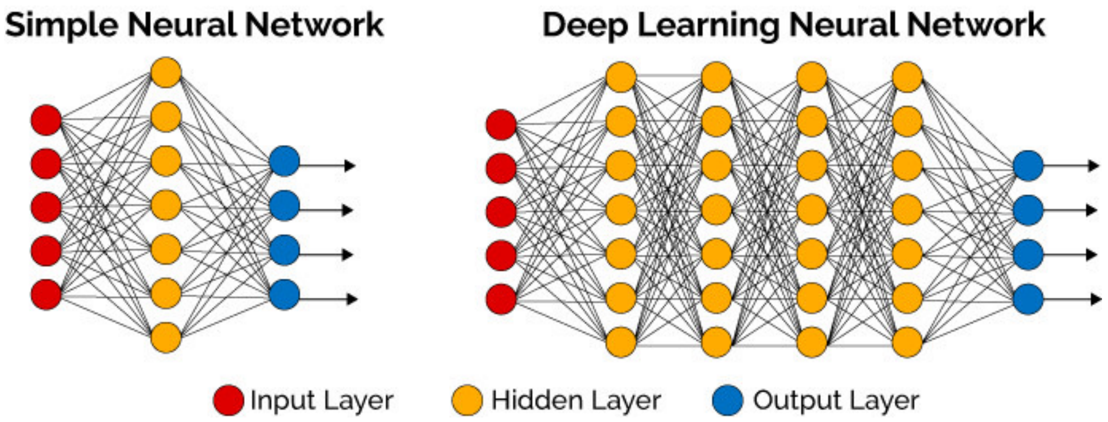


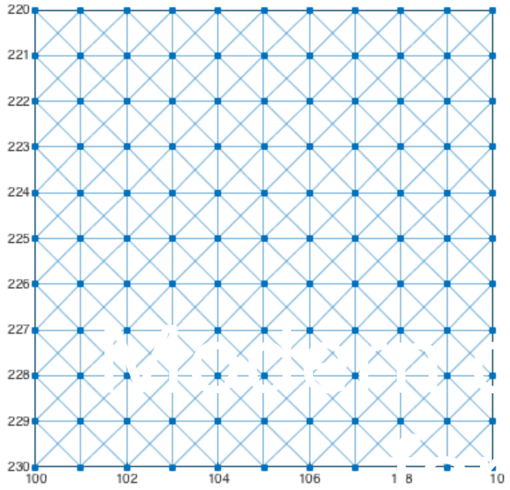
THE MODEL ARCHITECTURE FOR DETERMINING OPTIMAL ROUTES AND THEIR TRAVEL TIME.

Image credit: [DeepMind]

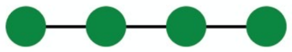
ML/DL on traditional data

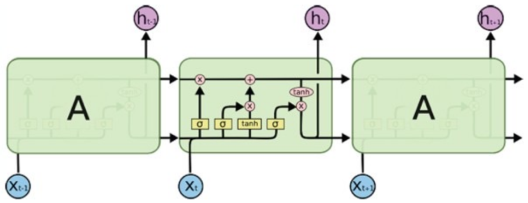






Images



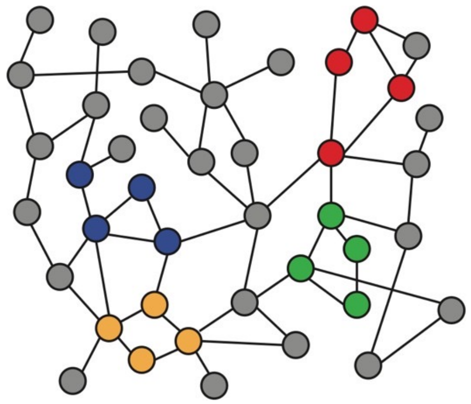


Text/Speech/Audio

Challenges

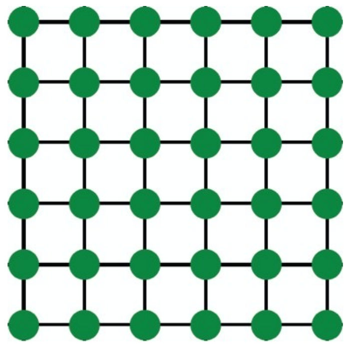
Graphs are complex

- Arbitrary size and complex topological structure (*i.e.*, no spatial locality like grids)
- No fixed node ordering or reference point
- Often dynamic and have multimodal features

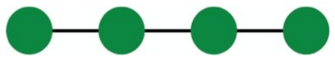


Graph

VS

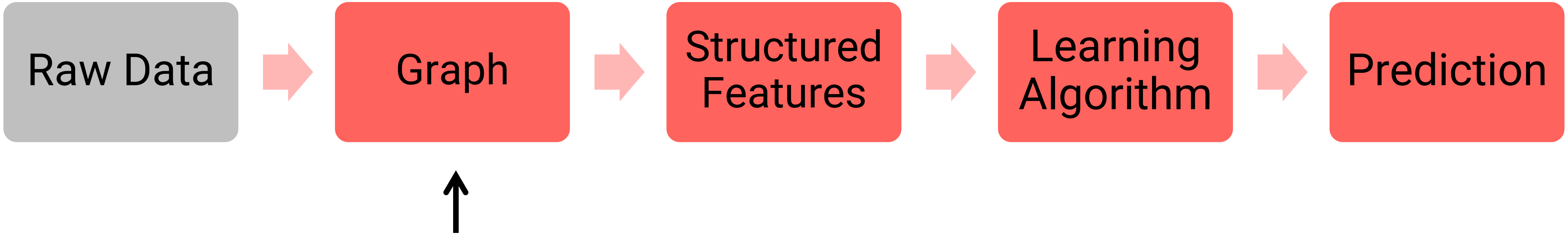


Image



Text

Graph Neural Networks



Adjacency Matrix? Adjacency List? CSR?

Machine learning needs features!

How to get features

1. **Feature Engineering**

   Covered in this topic

2. Graph Representation Learning

   Optional topic of node embedding

Raw Data → Graph → Structured Features → Learning Algorithm → Prediction

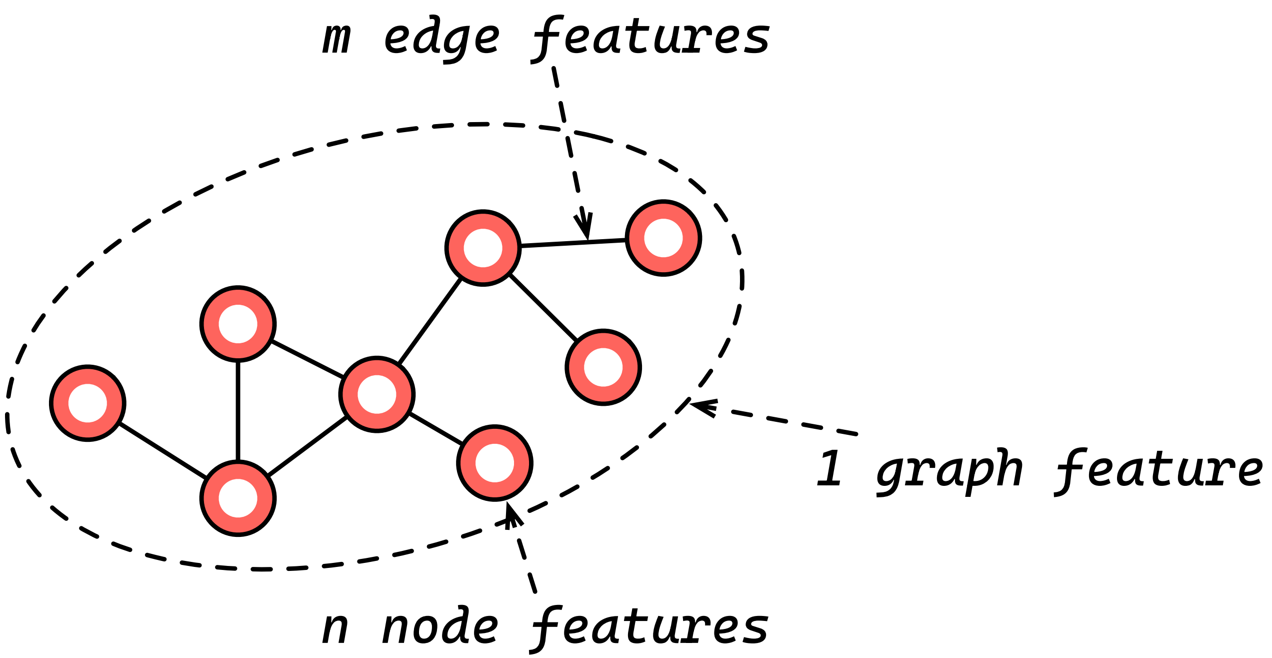
↑

Adjacency Matrix? Adjacency List? CSR?

Machine learning needs features!

Different types of graph features

- Node Level
- Edge Level
- Graph Level



Traditional ML on Graphs

Good features effectively represent the graph structure and achieve good performance.

1. Design features for **nodes**/edges/graphs.

2. Get features additional features from training data.

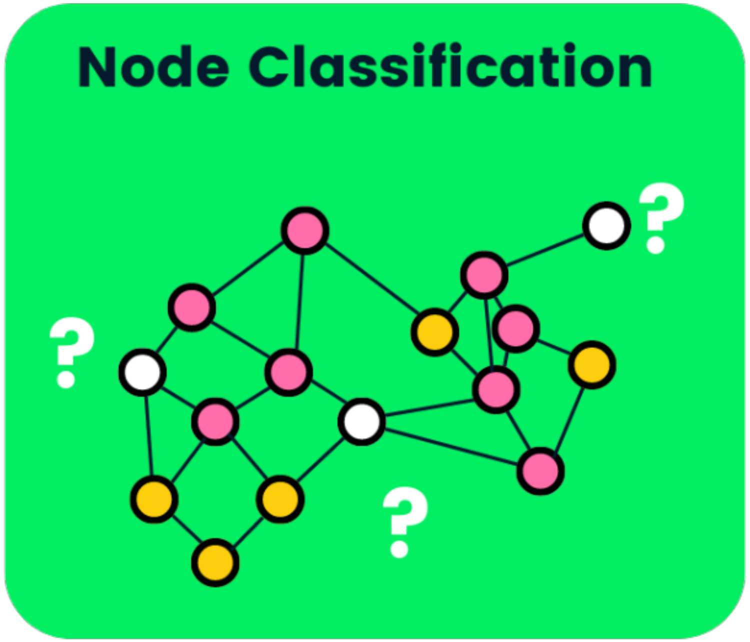
3. Use features to train parameters.

Testing: predict using the feature of query node/link/graph

Node-Level Features

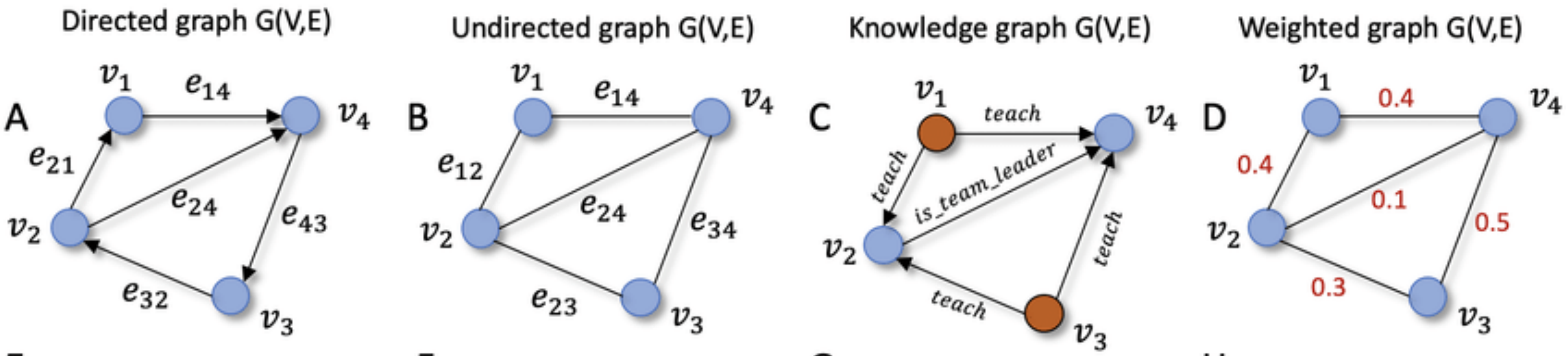
Goal:

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



A typical application: node classification

Adjacency Matrix?



E

	v_1	v_2	v_3	v_4
v_1	0	0	0	1
v_2	1	0	0	1
v_3	0	1	0	0
v_4	0	0	1	0

F

	v_1	v_2	v_3	v_4
v_1	0	1	0	1
v_2	1	0	1	1
v_3	0	1	0	1
v_4	1	1	1	0

G

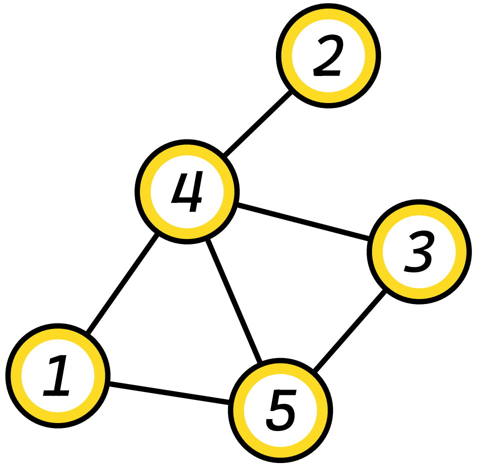
	v_1	v_2	v_3	v_4
v_1	0	1	0	1
v_2	0	0	0	1
v_3	0	1	0	1
v_4	0	0	0	0

H

	v_1	v_2	v_3	v_4
v_1	0	0.4	0	0.4
v_2	0.4	0	0.3	0.1
v_3	0	0.3	0	0.5
v_4	0.4	0.1	0.5	0

Not working for big graphs!

Adjacency List?

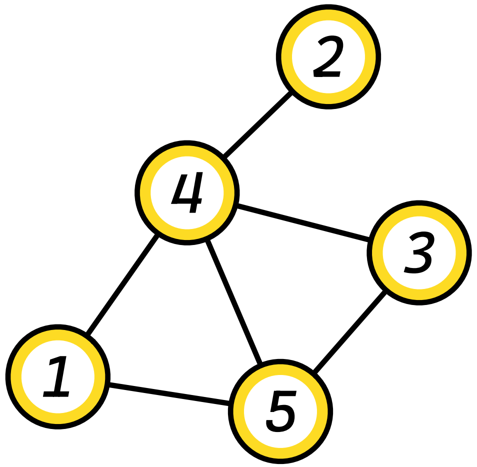


v1	4	5		
v2	4			
v3	4	5		
v4	1	2	3	5
v5	1	3	4	

Feature dimension need to be consistent

Adjacency List?

How about this?



v1	4	5	0	0
v2	4	0	0	0
v3	4	5	0	0
v4	1	2	3	5
v5	1	3	4	0

Feature dimension need to be consistent

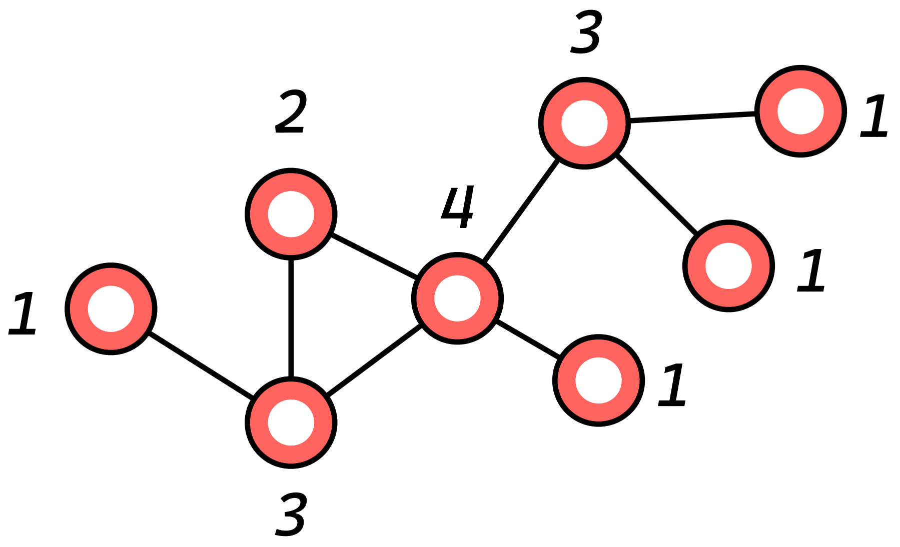
Node-Level Features: Overview

- Node degree
- Clustering coefficient
- Graphlets
- Node centrality
- …

Node Degree

Degree of a node: the number of neighbors.

Treat all neighbors equally.

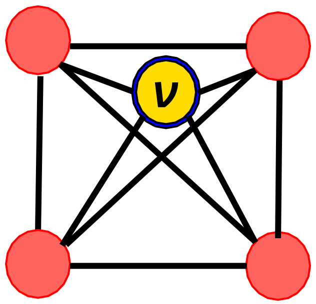


Node Centrality: Clustering Coefficient

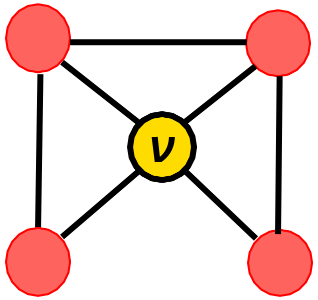
Measures how connected $v's$ neighboring nodes are:

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

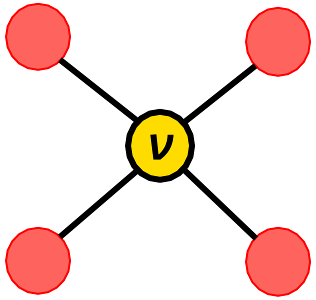
Can be also understand as #triangles/#possible triangles



$e_v = 1$



$e_v = 0.5$



$e_v = 0$

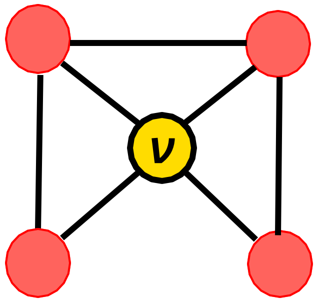
Ego-network:
the induced subgraph of the node and all its neighbors

Computing Clustering Coefficient

Can you design an algorithm to compute the clustering coefficient of all nodes in a graph with n nodes and m edges?

Node Features: Graphlets

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



$$e_v = 0.5$$

Three triangles in 6 possible triplets

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

Node Features: Graphlets

Graphlets are small subgraphs.

We aim to describe network structure around the node based on graphlets.

Analogy:

Degree

counts **#(edges)** that a node touches.

Clustering coefficient

counts **#(triangles)** that a node is involved.

Graphlet Degree Vector (GDV):

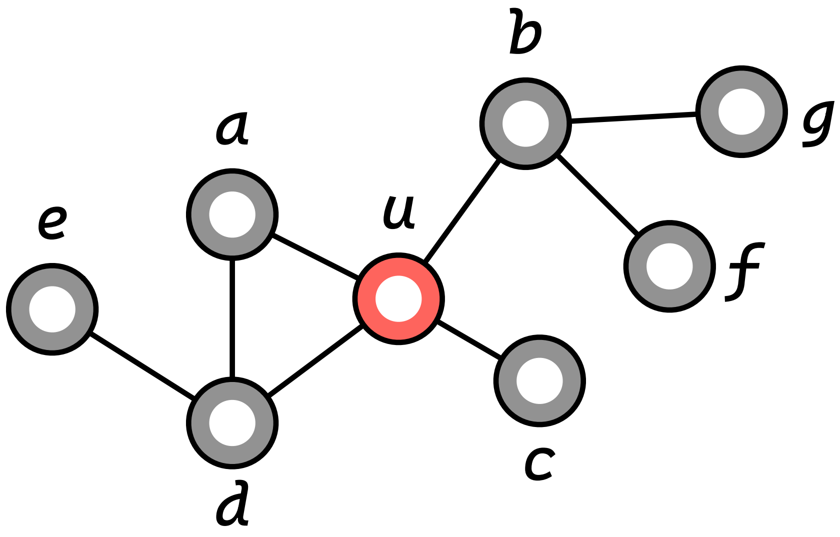
Graphlet-base features for nodes

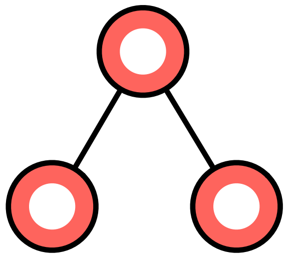
GDV counts **#(graphlets)** that a node is involved.

Node Features: Graphlets

How to represent a node by graphlets?

Let's start by considering (connected) graphlets with three nodes:





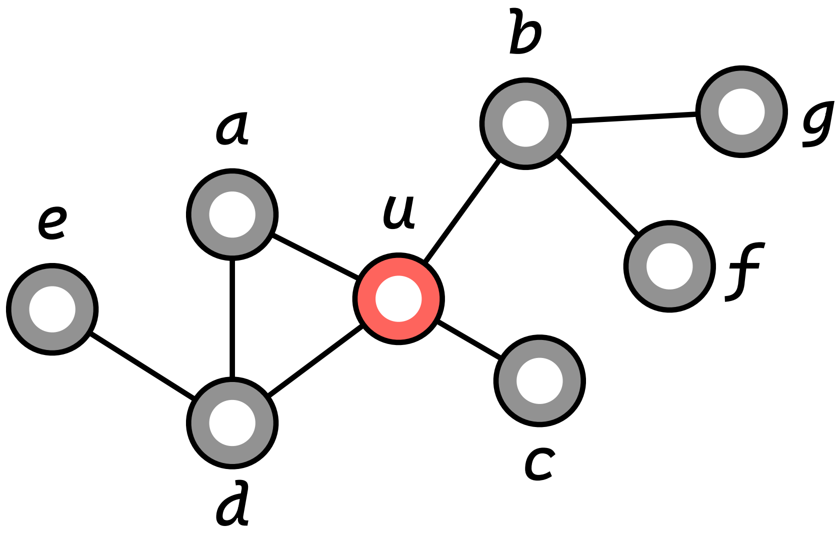
Choose a specific pattern (wedge)

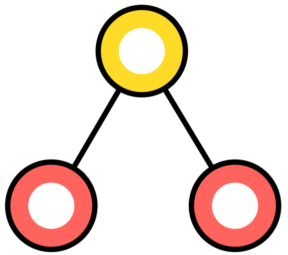
How many subgraphs containing u that are isomorphic to the pattern?

Node Features: Graphlets

How many subgraphs containing u that are isomorphic to the pattern?

11 after removing symmetric cases





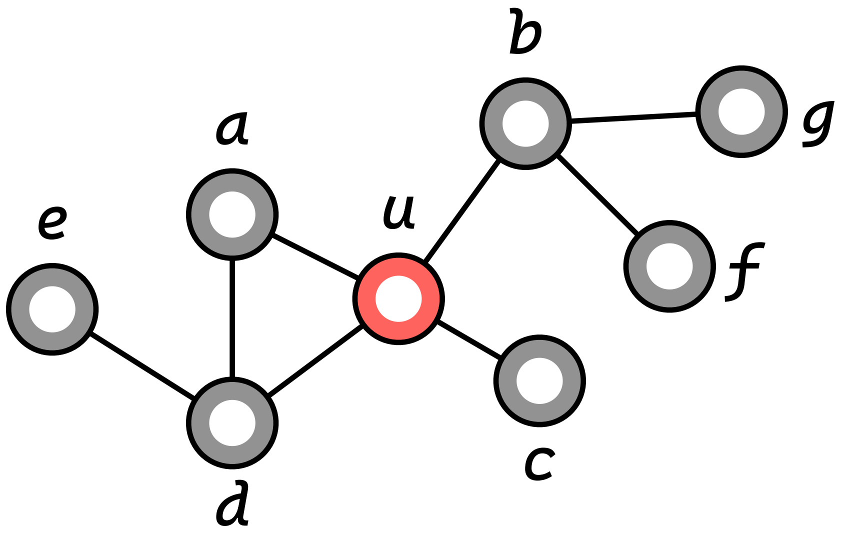
Choose a specific pattern (wedge)

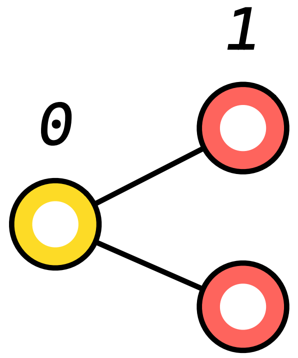
We use ***11*** as the feature of u

Node Features: Graphlets

Move forward by utilizing different types:

```
6 for type 0
5 for type 1
```



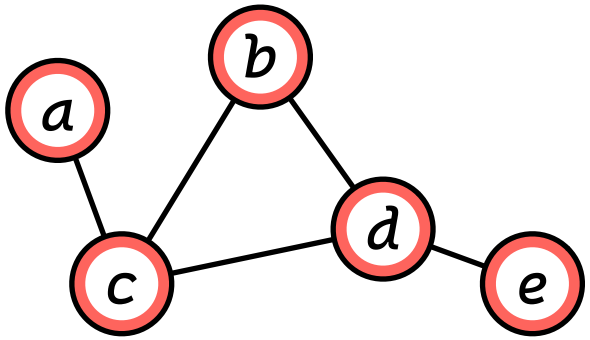


Choose a specific pattern (wedge)

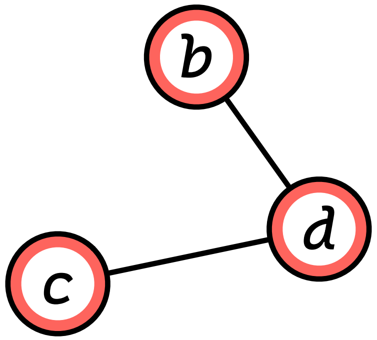
We use *[6,5]* as the feature of u

Node Features: Graphlets

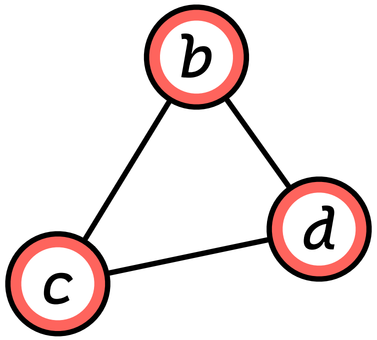
Move forward by only considering induced matching instances:



A graph



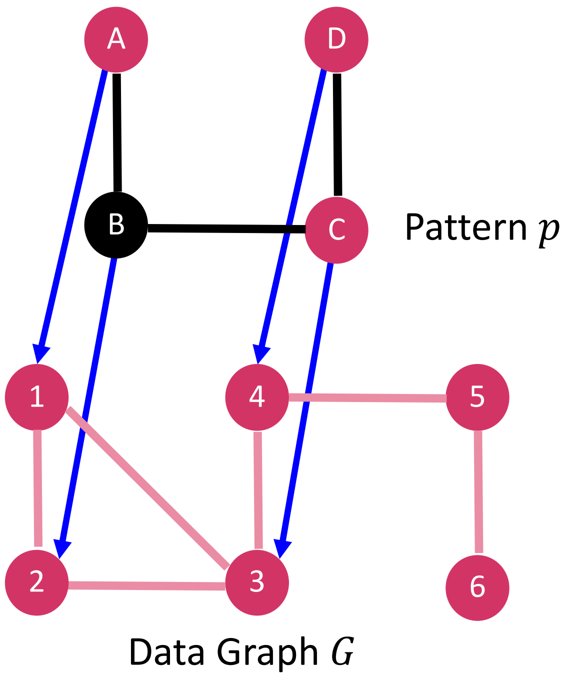
A non-induced subgraph



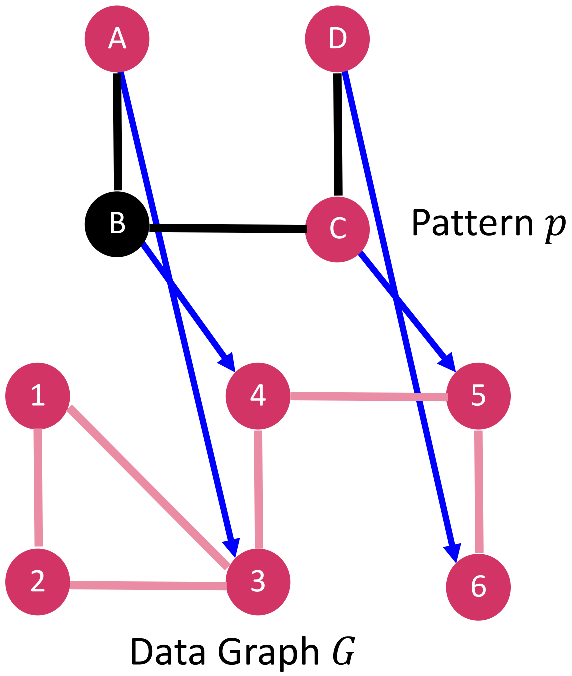
An induced subgraph

Node Features: Graphlets

Move forward by only considering induced matching instances:



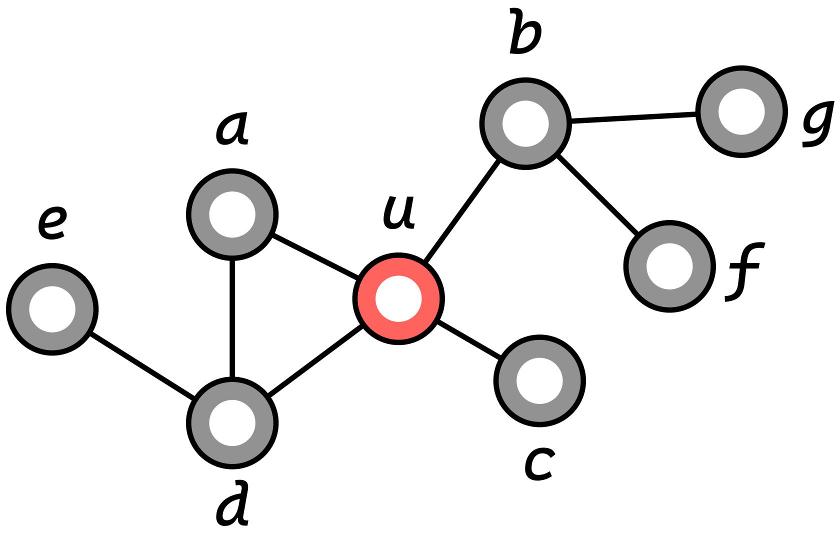
non-induced matching

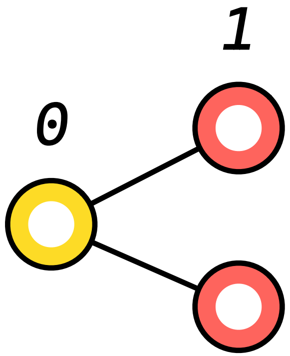


induced matching

Node Features: Graphlets

Move forward by only considering **induced** matching instances:





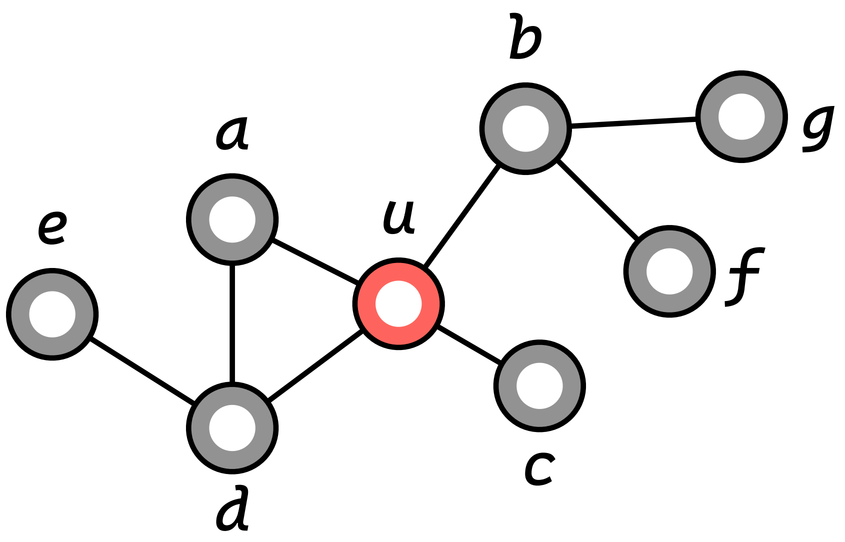
Choose a specific pattern (wedge)

We use *[5,3]* as the feature of u

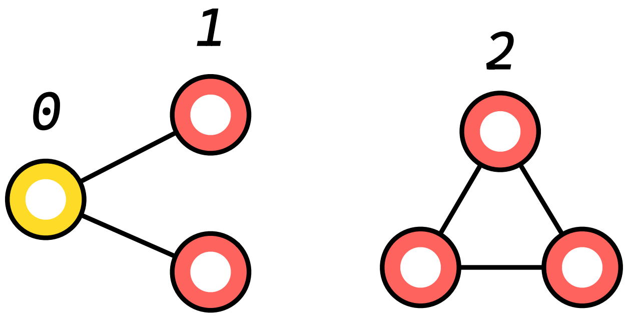
Node Features: Graphlets

Move forward by utilizing all 3-graphlets:

6 for type 0
5 for type 1
1 for type 2



Type 0:



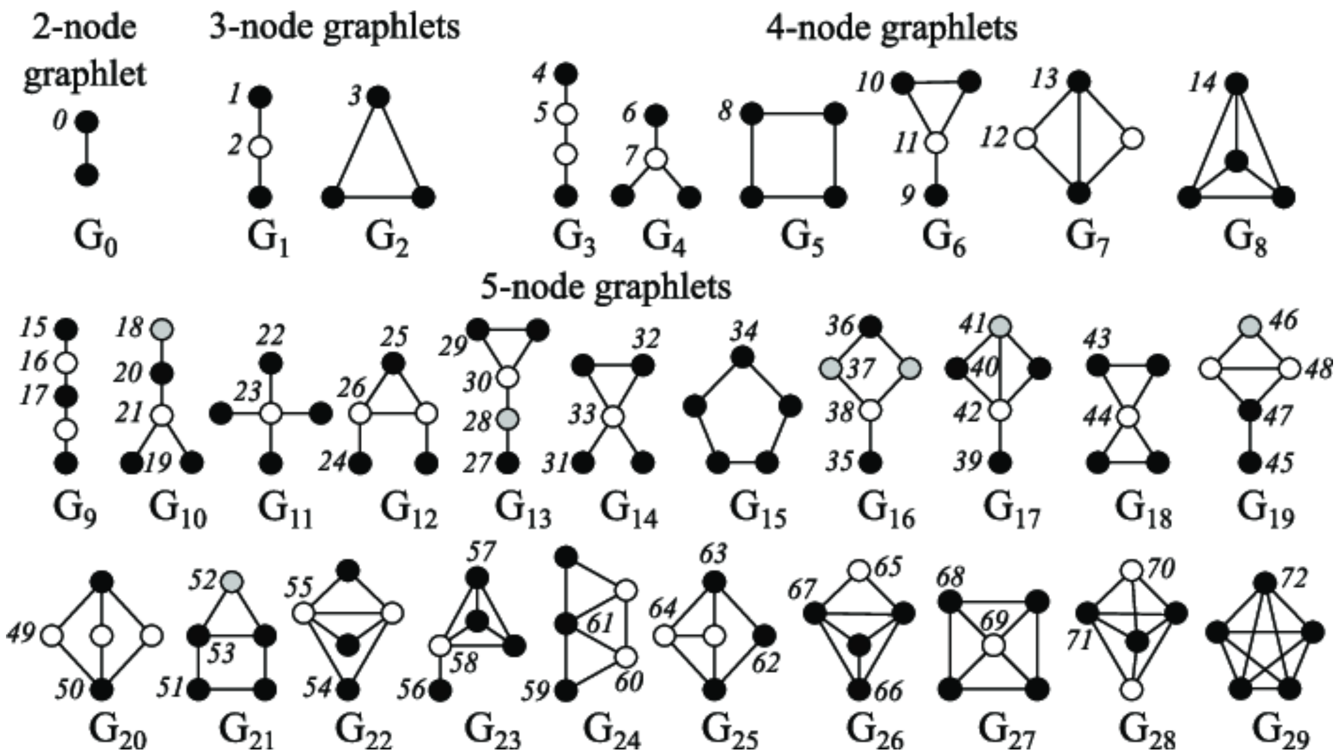
There are three types in all 3-graphlets.

We use *[5,3,1]* as the feature of u

Node Features: Graphlets

Consider all graphlets with <= 5 nodes

How many node roles in all connected non-isomorphic subgraphs?



There are **73** different graphlets of up to 5 nodes.

To get the node feature, compute the number of induced matching instances for each role id.

Node Features: Graphlets

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

Considering graphlets of size 2-5 nodes we get:

- Vector of 73 coordinates is a signature of a node that describes the topology of node's neighborhood

Graphlet degree vector provides a measure of a node's local network topology:

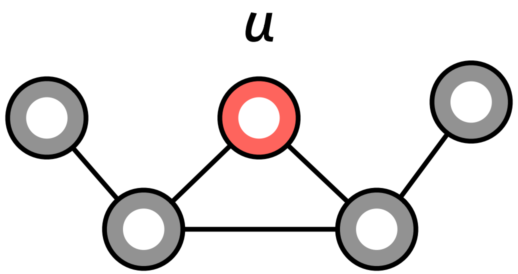
- Comparing vectors of two nodes provides a more detailed measure of local topological similarity than node degrees or clustering coefficient.

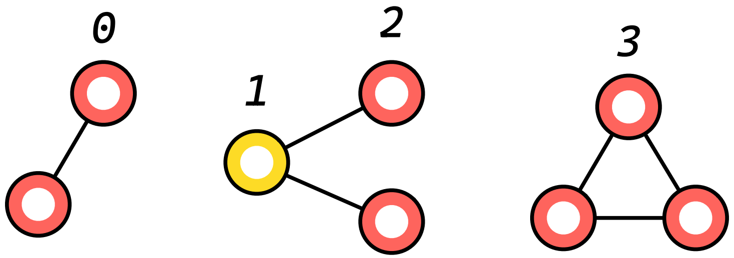
Usually, we only compute up to 4 or 5 nodes . . .

Node Features: Graphlets

More examples:

`GFV(u) = [2,0,2,1]`





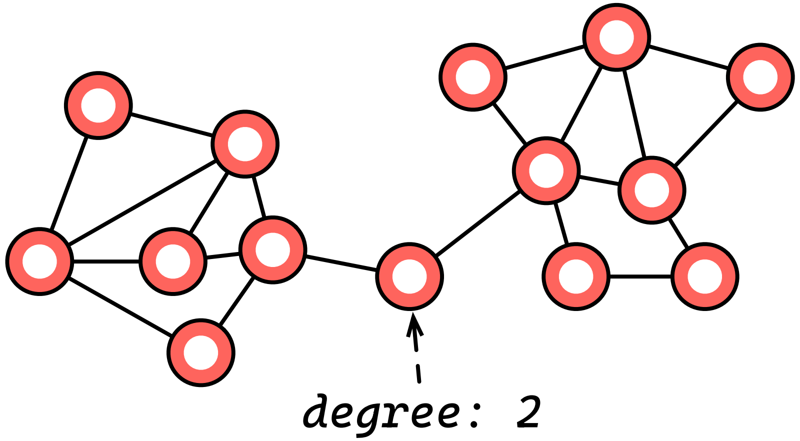
Node Centrality

Node degree counts neighbors without capturing their importance.

Node **centrality** takes the node importance in a graph into account

Different ways to model importance:

- PageRank
- Eigenvector centrality
- Betweenness centrality
- Closeness centrality
- many others…



Node Centrality: Eigenvector

Motivation

A node is important if surrounded by important neighbors.

We model the centrality of node v as the sum of the centrality of neighbors:

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

λ is normalization constant (it will turn out to be the largest eigenvalue of A)

The above equation models centrality in a recursive manner.
How do we solve it?

Node Centrality: Eigenvector

Optional

Math Warning!

Rewrite the recursive equation in the matrix form.

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

λ is normalization const (largest eigenvalue of A)

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

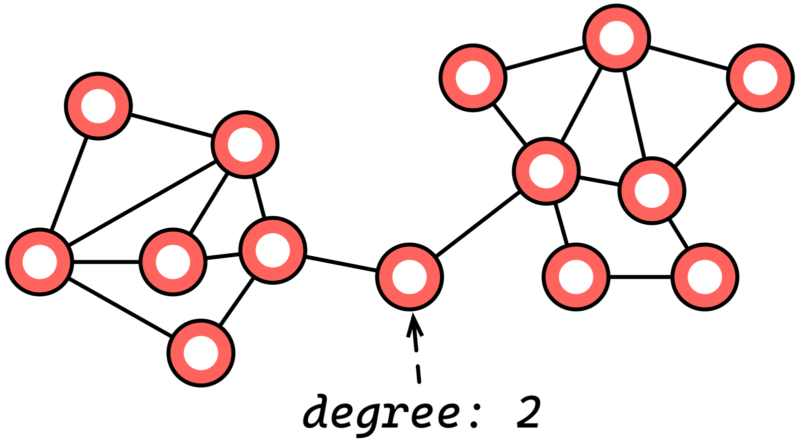
- We see that centrality c is the **eigenvector of $\boldsymbol{A}$**!
- The largest eigenvalue λ_{max} is always positive and unique (by Perron-Frobenius Theorem).
- The eigenvector $\boldsymbol{c}_{max}$ corresponding to λ_{max} is used for centrality.

Node Centrality: Betweenness

Betweenness centrality:

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

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

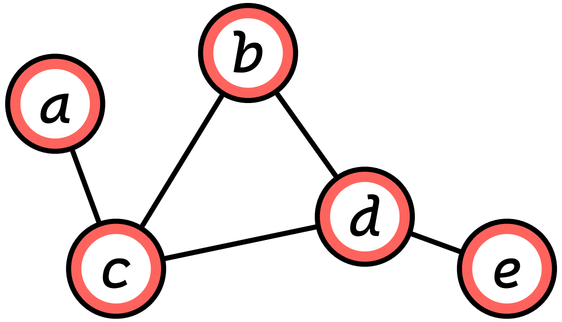


How to identity the bridge node

Node Centrality: Betweenness (cont)

Example:

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



$c_a = c_b = c_e = 0$

$c_c = 3$
(a-c-b, a-c-d, a-c-d-e)

$c_d = 3$
(a-c-d-e, b-d-e, c-d-e)

Computing Betweenness Centrality

Exact solution:

$O(nm)$ for unweighted graphs

$O(nm+n^2 \log n)$ for weighted graphs

https://kops.uni-konstanz.de/server/api/core/bitstreams/420590d1-3010-4eab-a585-6fa3eff46f9e/content

Approximate solution:

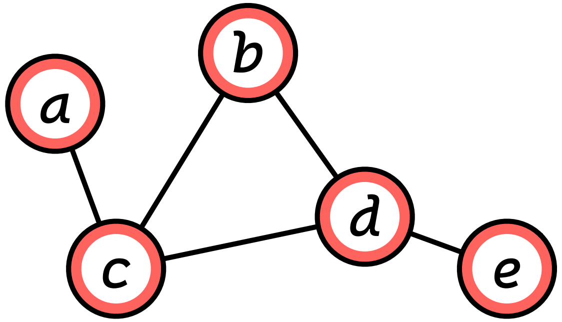
Sampling a set of shortest paths…

Node Centrality: Closeness

Closeness centrality:

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

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



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)

Computing Closeness Centrality

Can you design an algorithm to compute the closeness centrality of all nodes in a graph with n nodes and m edges?

Node-Level Feature: Summary

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

- Structure-based features:
  - Node degree
  - Clustering coefficient
  - Graphlet count vector

Node-level Feature: Summary

Importance-based features: capture the importance of a node in a graph

- ~~Node degree:~~
  - Simply counts the number of neighboring nodes
- Node centrality:
  - Model importance of neighbors 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

Node-level Feature: Summary

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.

Learning Outcome

- **Traditional ML Pipeline**
  - Hand-crafted feature + ML model
- **Hand-crafted node features for graph data**
  - Node degree, centrality, clustering coefficient, graphlets