CSE8803/CX4803

Machine Learning in Computational Biology

Lecture 14a:
Network basics

Yunan Luo

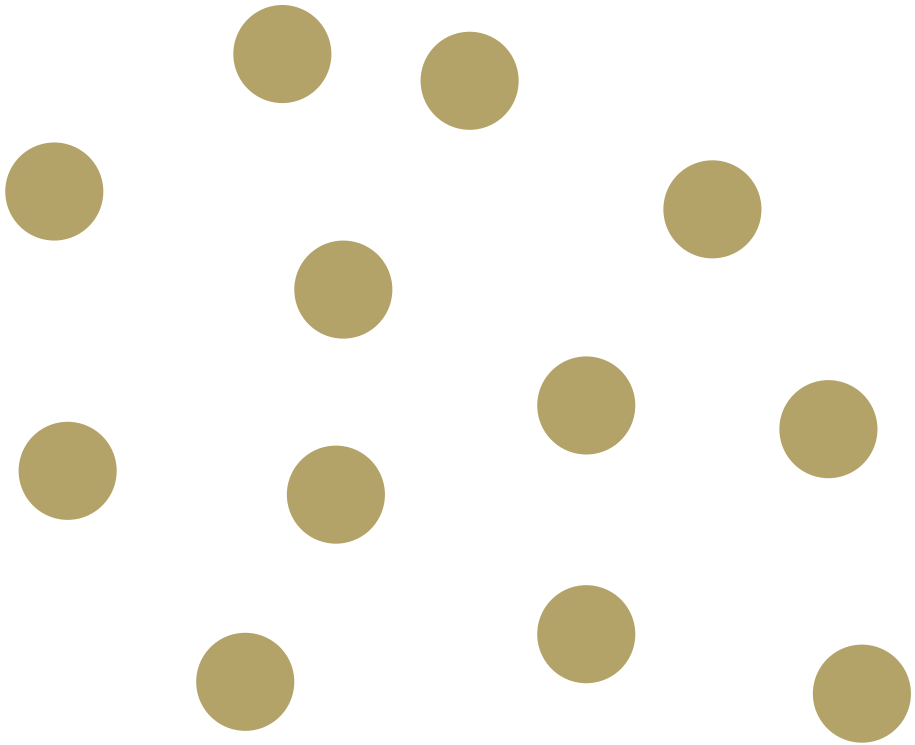
Learning from network data

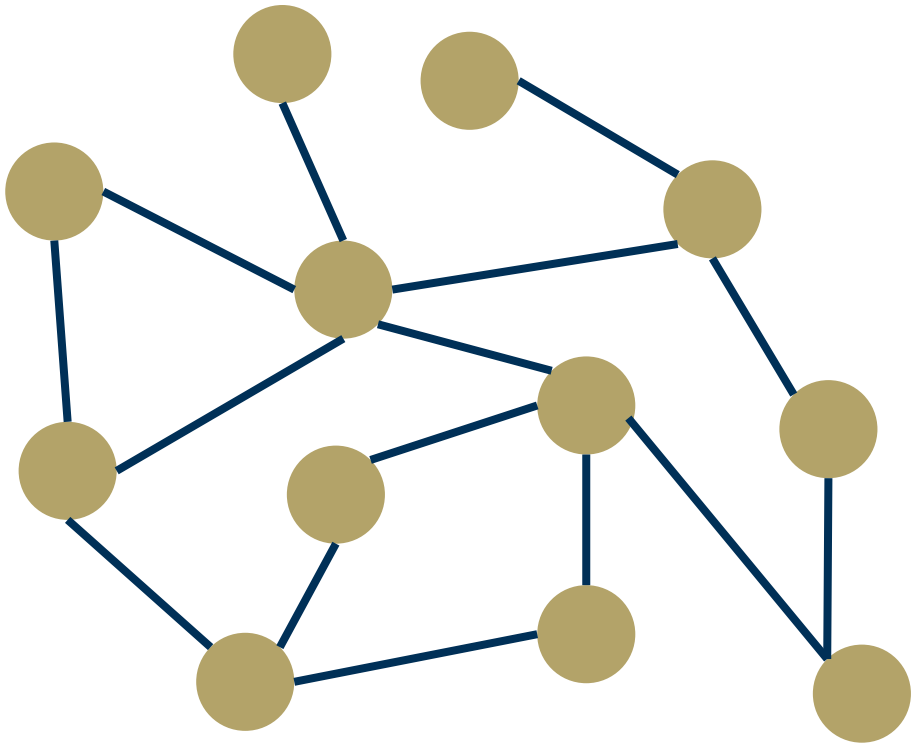
1. Learning from sequence data
2. Learning from high-dim data
3. Learning from structure data
4. **Learning from network data**

Date	Topic	Contents
1/10/2022	Introduction	Course intro & how to present papers
1/12/2022	Learning from sequence data	Dynamic programming & sequence alignment I
1/17/2022		No class (MLK Day)
1/19/2022		Sequence alignment II
1/24/2022		HMM & gene/motif finding
1/26/2022		HMM & Profile HMM
1/31/2022		Deep learning for DNA/protein sequence
2/2/2022	Learning from high-dim data	Learn from high-dim data: PCA, autoencoder & VAE
2/7/2022		Learn from high-dim data: MDS, tSNE, UMAP
2/9/2022		Clustering I
2/14/2022		Clustering II
2/16/2022		Clustering III
2/21/2022	Phase 1 presentations	Student presentation 1-3
2/23/2022		Student presentation 4-6
2/28/2022	Learning from structure data	RNA structure prediction
3/2/2022		Deep learning for structures (protein structure prediction)
3/7/2022	Phase 2 presentations	Student presentation 7-9
3/9/2022	Learning from network data	Network basics & traditinal ML for graphs
3/14/2022		Network embeddings
3/16/2022	Phase 3 presentations	Student presentation 10-12
3/21/2022	Spring break	No class (Spring Break)
3/23/2022		No class (Spring Break)
3/28/2022	Learning from network data	Graphical Models
3/30/2022		Deep learning for networks (graph neural networks)
4/4/2022	Phase 4 presentations	Student presentation 13-15
4/6/2022		Student presentation 15-18
4/11/2022		Student presentation 18-21
4/13/2022		Student presentation 22-24
4/18/2022		Student presentation 25-27
4/20/2022		Student presentation 28-30
4/25/2022		Student presentation 31-33

Network (graph) basics

Slides adapted from CS224W, Jure Leskovec





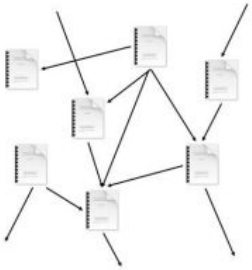
Networks (graphs)

Image credit: Medium

Social networks

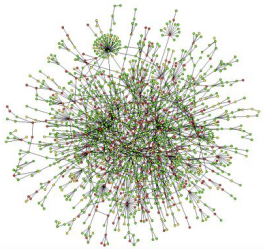
Image credit: Missoula Current News

Internet

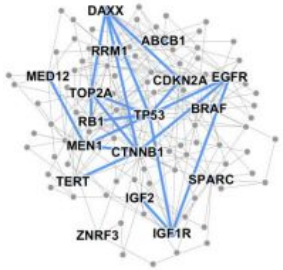


Citation networks

Underground networks



Protein-protein interaction networks



Disease pathways

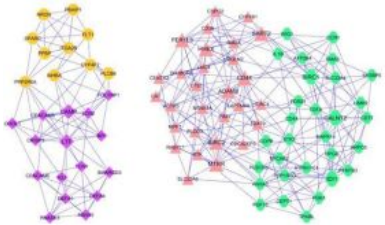


Image credit: ese.wustl.edu

Gene regulatory networks

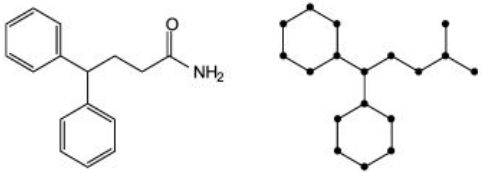
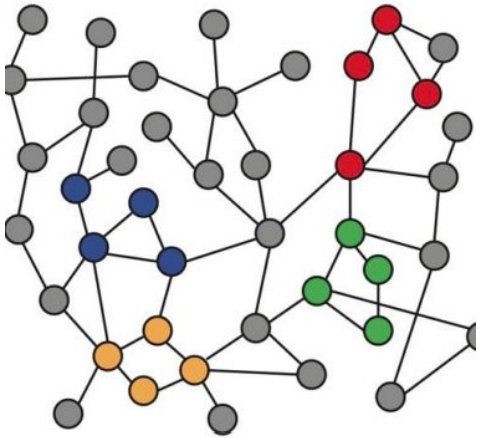


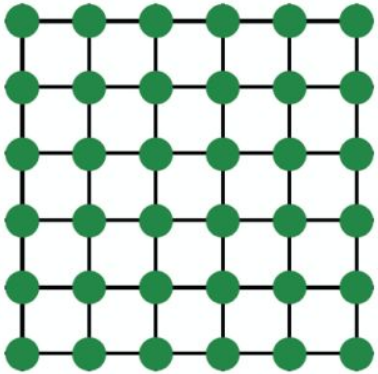
Image credit: MDPI

Molecules

Network is a general data representation

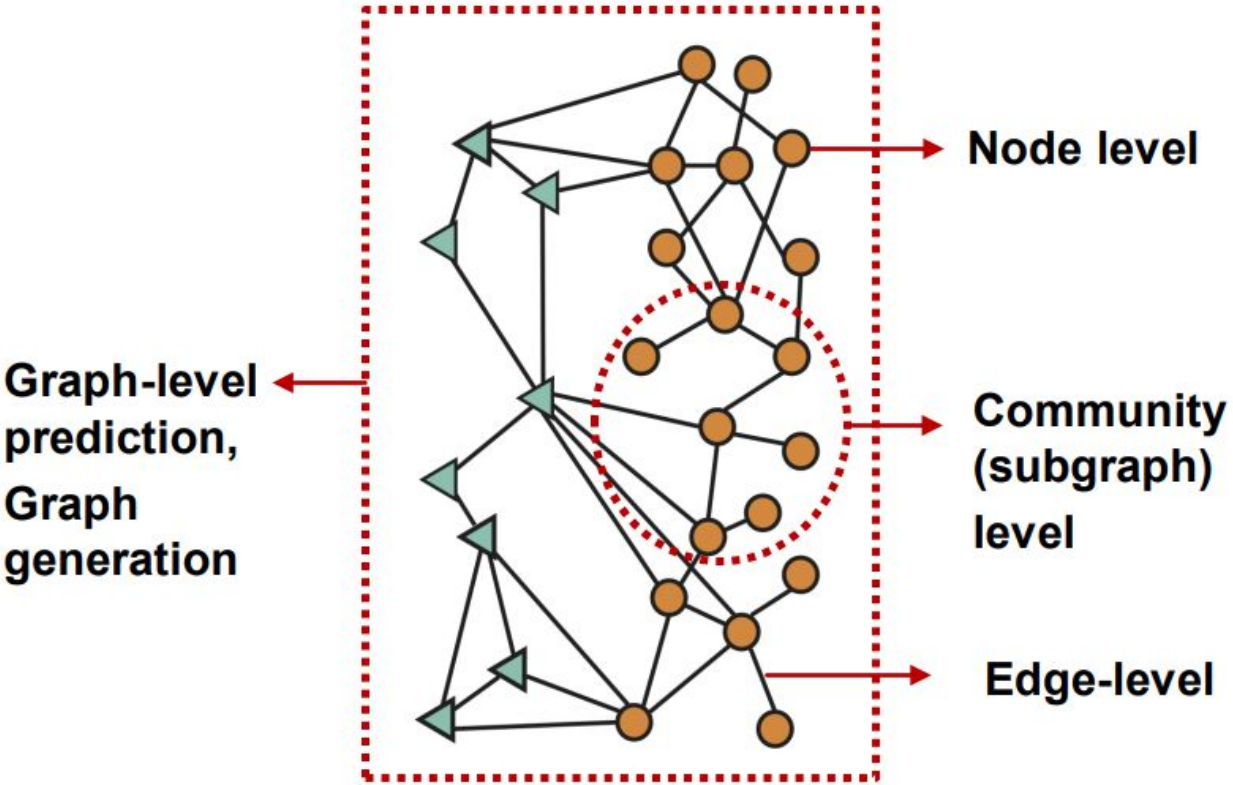


Special cases



Networks **Images** **Text**

Different types of ML tasks on graphs



Classic graph ML tasks

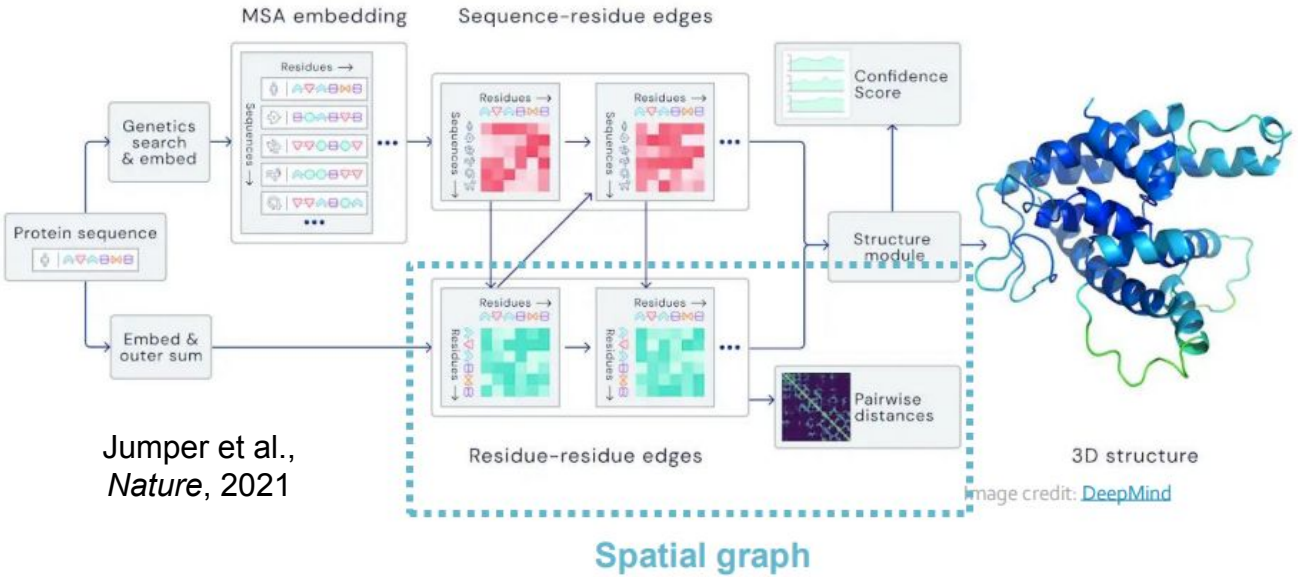
- **Node classification**: predict a property of a node
  - **Example**: Protein function prediction
- **Link prediction**: Predict whether a link exists between two nodes
  - **Example**: Recommendation (user <-> item), drug-target interaction prediction
- **Graph classification**: categorize different graphs
  - **Example**: Molecular property prediction
- **Clustering**: detect if nodes form a community
  - **Example**: Social circle detection, disease pathway detection
- **Graph generation**: generate new graphs
  - **Example**: drug design

These graph ML tasks lead to high-impact applications in real world

Example 1: node-level ML tasks

Protein structure prediction

- **Nodes**: amino acids in a protein sequence
- **Edges**: proximity (distance) between amino acids (residues)

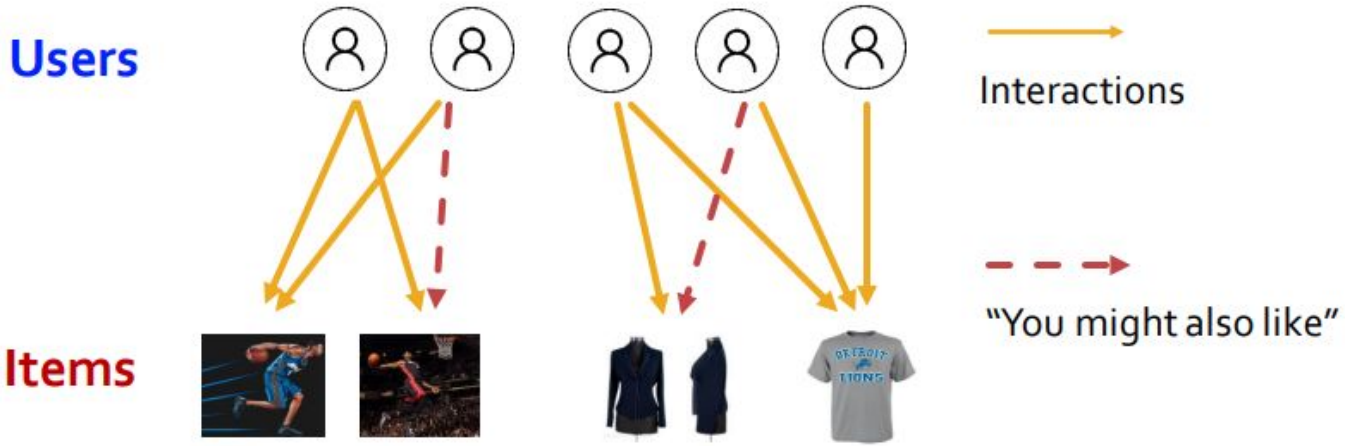


Jumper et al., *Nature*, 2021

Example 2: Edge-level ML tasks

Recommender Systems

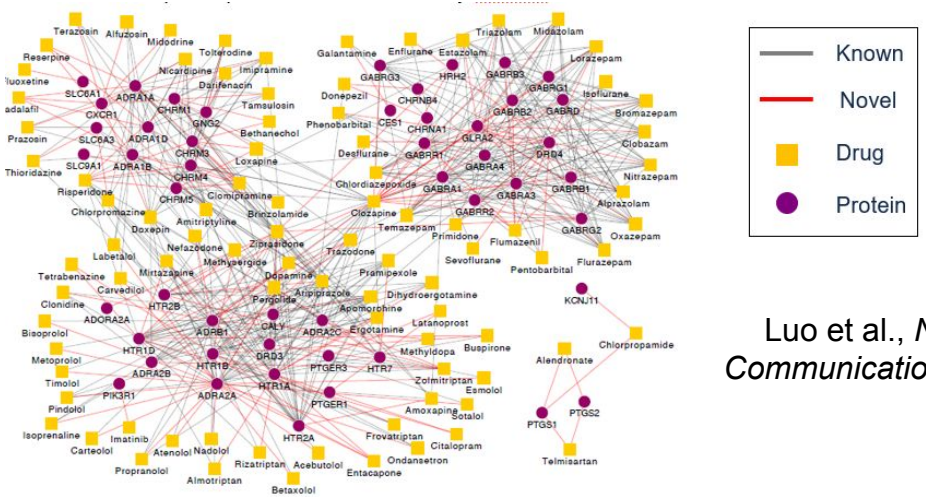
- **Nodes**: uses and items
- **Edges**: user-item interactions
- **Goal**: Recommend items users might like



Example 3: Edge-level ML tasks

Drug-target interaction

- **Nodes**: drugs and proteins (targets)
- **Edges**: drug-target interactions
- **Goal**: Predict unknown interactions between drugs and targets



Luo et al., *Nature Communications*, 2017

Example 4: Subgraph-level ML tasks

Traffic prediction

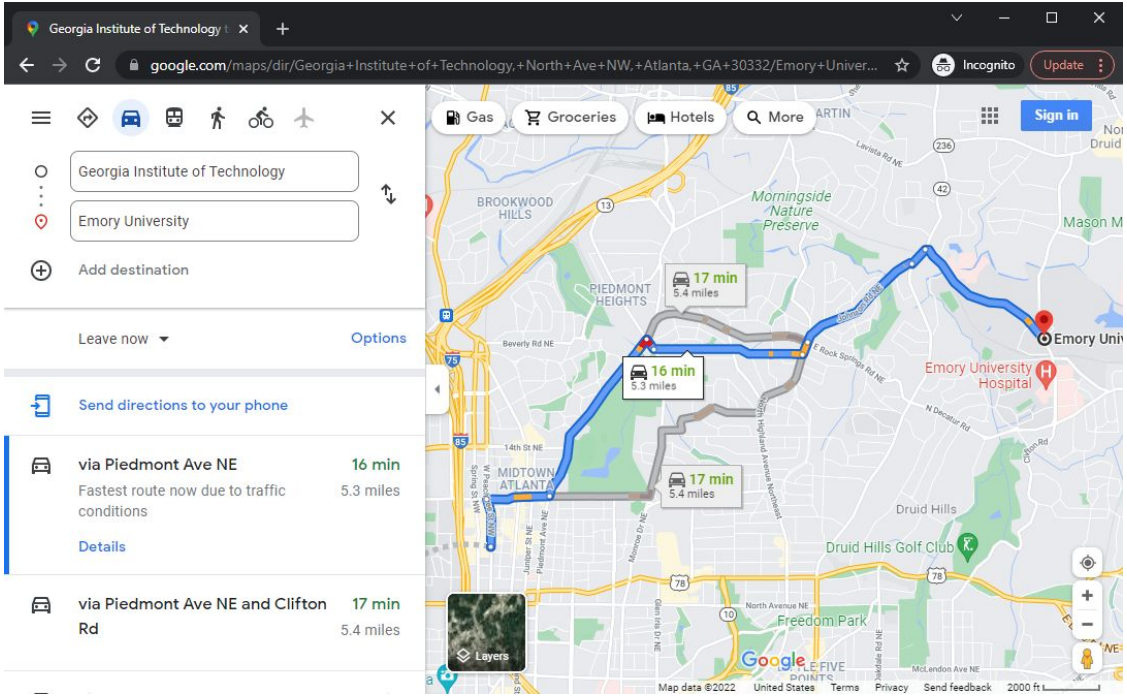
- **Nodes**: road segments
- **Edges**: connectivity between road segments
- **Goal**: predict time of arrival (ETA)

BLOG POST RESEARCH

03 SEP 2020

Traffic prediction with advanced Graph Neural Networks

DeepMind Blog

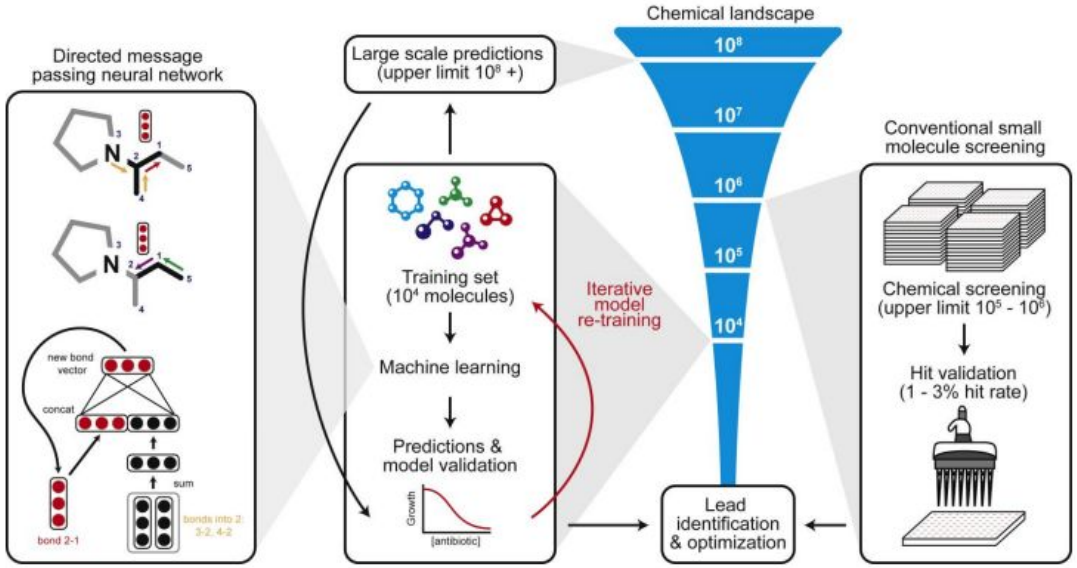


Deployed in Google Maps

Example 5: Graph-level ML tasks

Antibiotics discovery

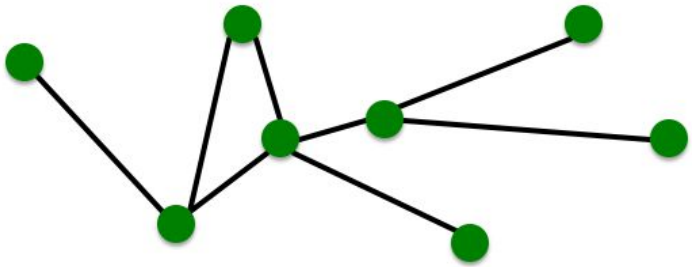
- **Nodes**: atoms
- **Edges**: bonds
- **Goal**: Predict novel antibiotics with a desired property



Stokes et al., *Cell*, 2020

What is a network (graph)?

Graph: $G = (V, E)$



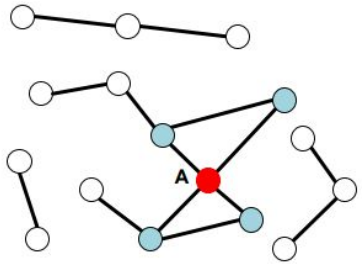
Objects: nodes, vertices — ***V***

Interactions: links, edges — ***E***

System: network, graph — ***G*(V, *E*)**

Node degree

Undirected

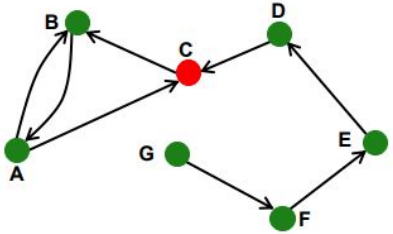


Node degree, k_i: the number of edges adjacent to node ***i***

$k_A = 4$

Avg. degree: $\bar{k} = \langle k \rangle = \frac{1}{N}\sum_{i=1}^{N} k_i = \frac{2E}{N}$

Directed



Source: Node with $k^{in} = 0$
Sink: Node with $k^{out} = 0$

In directed networks we define an **in-degree** and **out-degree.**
The (total) degree of a node is the sum of in- and out-degrees.

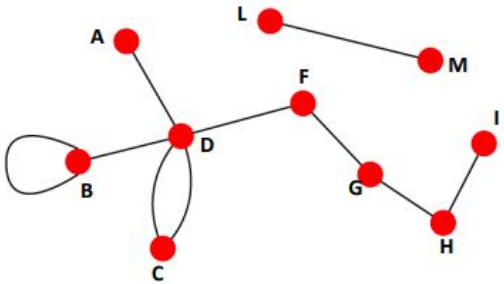
$k_C^{in} = 2 \qquad k_C^{out} = 1 \qquad k_C = 3$

$\bar{k} = \frac{E}{N}$ $\qquad\qquad$ $\overline{k^{in}} = \overline{k^{out}}$

Directed vs. Undirected Graphs

Undirected graphs

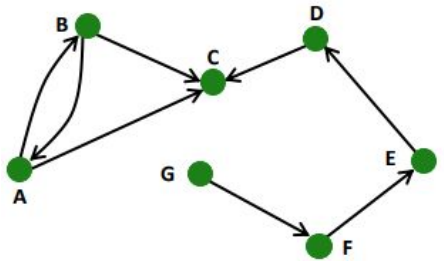
- **Links**: undirected (symmetrical, reciprocal)



- **Examples**:
  - Collaborations
  - Friendship on Facebook

Directed graphs

- **Links**: directed (arcs)



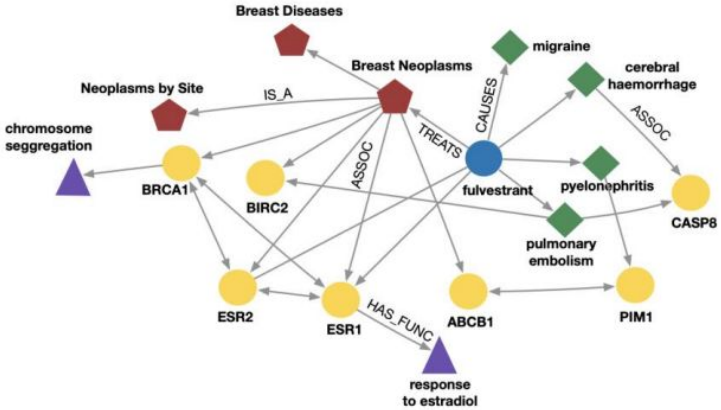
- **Examples**:
  - Paper citation
  - Flights between airports

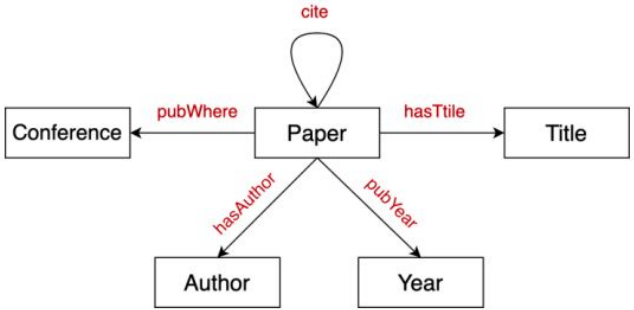
Heterogeneous graphs

Heterogeneous graph: $G = (V, E, T, R)$

- Nodes with node types $v_i \in V$
- Edges with relation types $(v_i, r, v_j) \in V$
- Node type $T(v_i)$
- Relation type $r \in \mathbf{R}$

Many graphs are heterogeneous graphs





Biomedical Knowledge Graphs

Example node: Migraine
Example edge: (fulvestrant, Treats, Breast Neoplasms)
Example node type: Protein
Example edge type (relation): Causes

Academic Graphs

Example node: ICML
Example edge: (GraphSAGE, NeurIPS)
Example node type: Author
Example edge type (relation): pubYear

Representing graphs: adjacency matrix

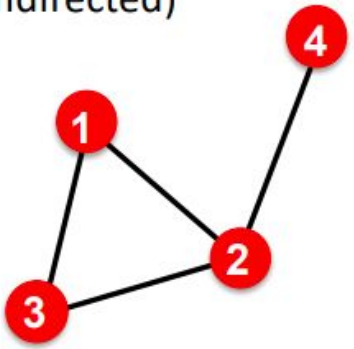
$A_{ij} = 1$ if there is a link from node i to node j

$A_{ij} = 0$ otherwise

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

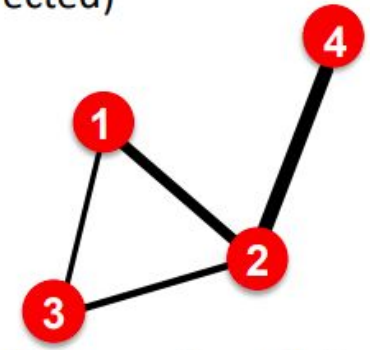
Weighted adjacency matrix

- **Unweighted**
  (undirected)



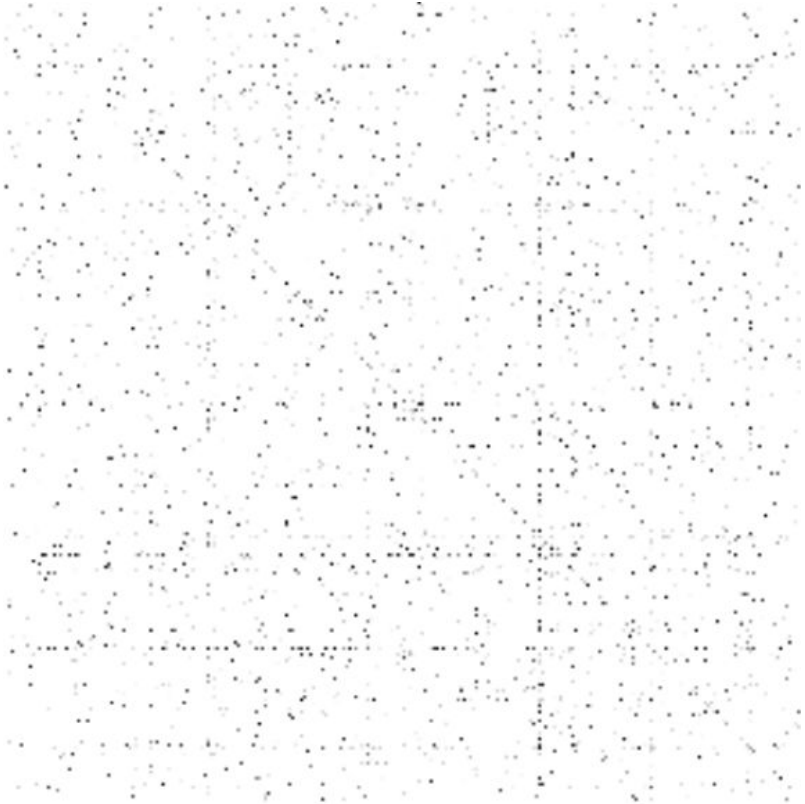
$$A_{ij} = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

- **Weighted**
  (undirected)



$$A_{ij} = \begin{pmatrix} 0 & 2 & 0.5 & 0 \\ 2 & 0 & 1 & 4 \\ 0.5 & 1 & 0 & 0 \\ 0 & 4 & 0 & 0 \end{pmatrix}$$

Adjacency matrices are sparse



Most real-world networks are sparse

Number of edges

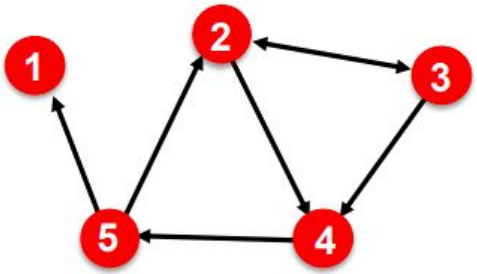
Average degree

NETWORK	NODES	LINKS	DIRECTED/ UNDIRECTED	N	E	<k>
Internet	Routers	Internet connections	Undirected	192,244	609,066	6.33
WWW	Webpages	Links	Directed	325,729	1,497,134	4.60
Power Grid	Power plants, transformers	Cables	Undirected	4,941	6,594	2.67
Phone Calls	Subscribers	Calls	Directed	36,595	91,826	2.51
Email	Email Addresses	Emails	Directed	57,194	103,731	1.81
Science Collaboration	Scientists	Co-authorship	Undirected	23,133	93,439	8.08
Actor Network	Actors	Co-acting	Undirected	702,388	29,397,908	83.71
Citation Network	Paper	Citations	Directed	449,673	4,689,479	10.43
E. Coli Metabolism	Metabolites	Chemical reactions	Directed	1,039	5,802	5.58
Protein Interactions	Proteins	Binding interactions	Undirected	2,018	2,930	2.90

Consequence: Adjacency matrix is filled with zeros
(Density of the matrix $[E/N^2]$: WWW=1.51×10^{-5})

Representing graphs: edge list

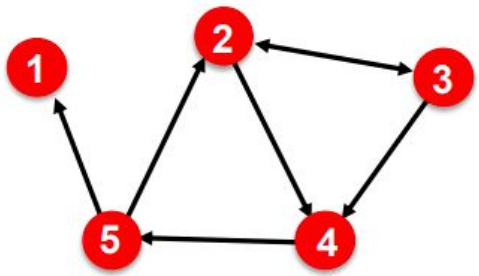
- Represent graph as a list of edges:



- (2, 3)
- (2, 4)
- (3, 2)
- (3, 4)
- (4, 5)
- (5, 2)
- (5, 1)

Representing graphs: adjacency list

- Represent graph as a list of (**key**: **value**) pairs:
  - (Node: its neighboring nodes)



- 1:
- 2: 3, 4
- 3: 2, 4
- 4: 5
- 5: 1, 2

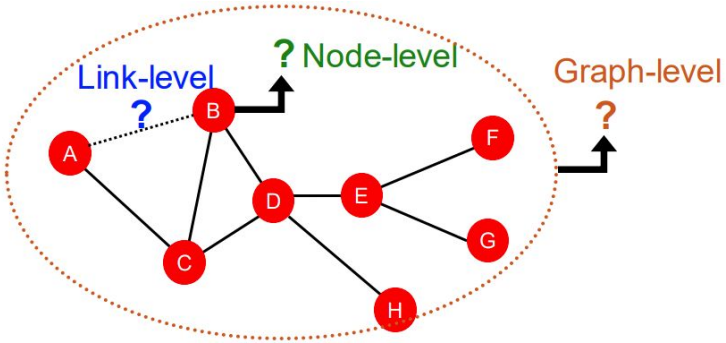
Traditional machine learning for graphs

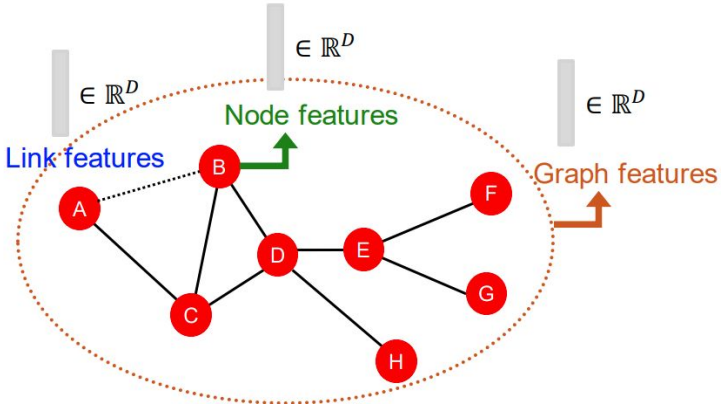
(for simplicity, we focus on undirected graphs in the lecture)

Machine learning tasks on graphs

- **Node-level** prediction
- **Link-level** prediction
- **Graph-level** prediction

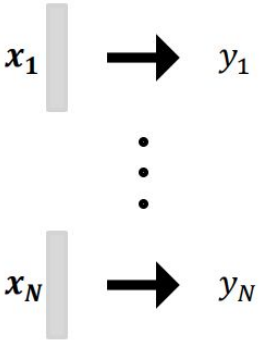
Design features for
nodes / links / graphs



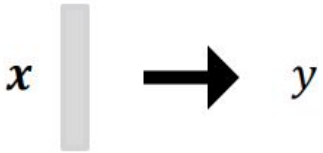


Traditional machine learning pipeline

- **Train an ML model**:
  - Obtain features of training data (node/edge/graph)
  - Train a ML model (SVM, NN, etc)



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



Traditional ML focuses on (manually) designing effective features over graphs