GRAPH ANALYSIS AND LEARNING IN GRAPHS

MDI 341, MS Big Data, Télécom ParisTech

Aurélien Bellet

Inria Lille

March 3, 2020

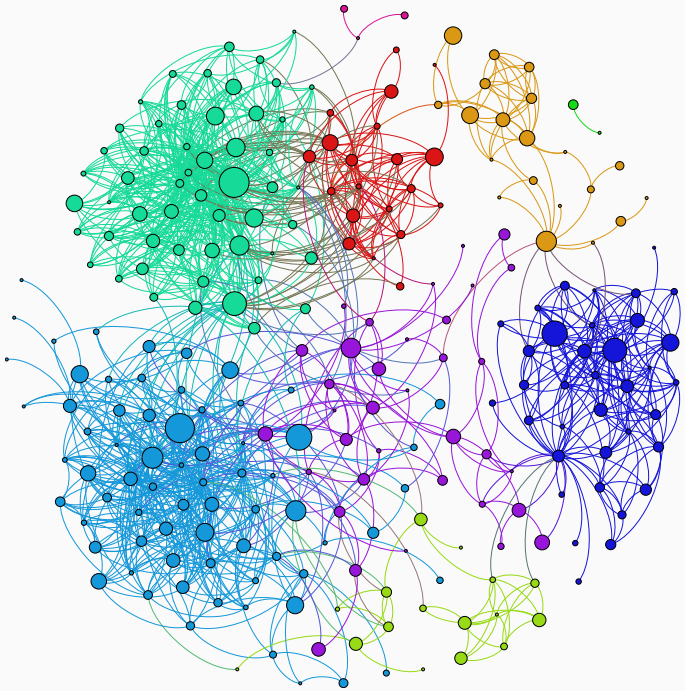
OUTLINE

1. Introduction
2. Basic Graph Notions
3. Graph Analysis
4. Learning in graphs

INTRODUCTION

Graph: collection of interconnected nodes

- Social networks

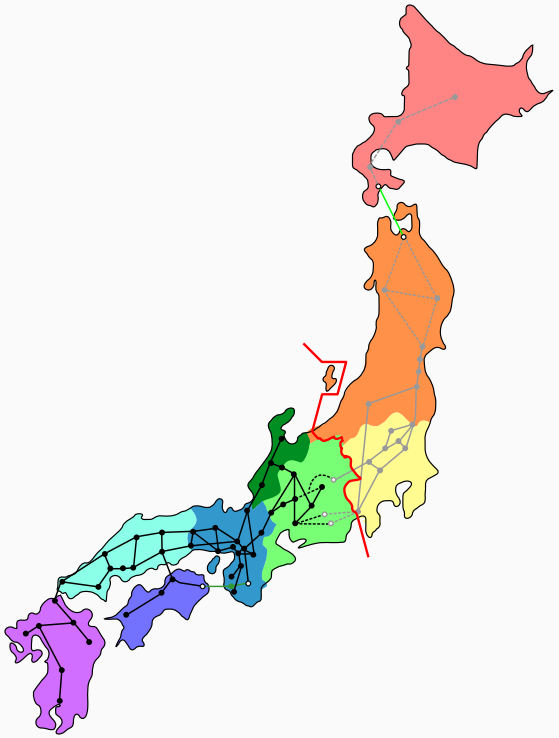


LinkedIn ego network

Credit: `http://allthingsgraphed.com`

Graph: collection of interconnected nodes

- Social networks
- Power grids



Japanese electrical network

Graph: collection of interconnected nodes

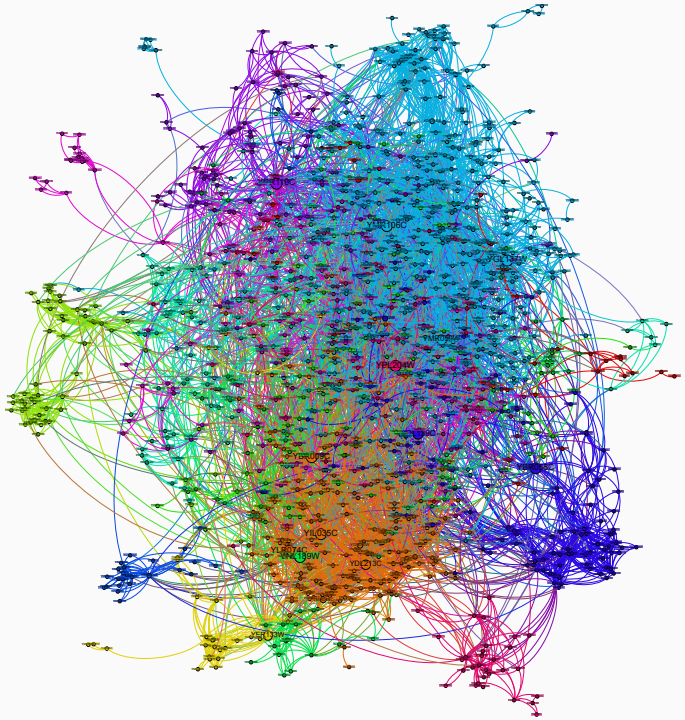
- Social networks
- Power grids
- Transportation networks

SNCF network

Credit: PouX / madcap, License: CC BY-SA 3.0

Graph: collection of interconnected nodes

- Social networks
- Power grids
- Transportation networks
- Biological networks

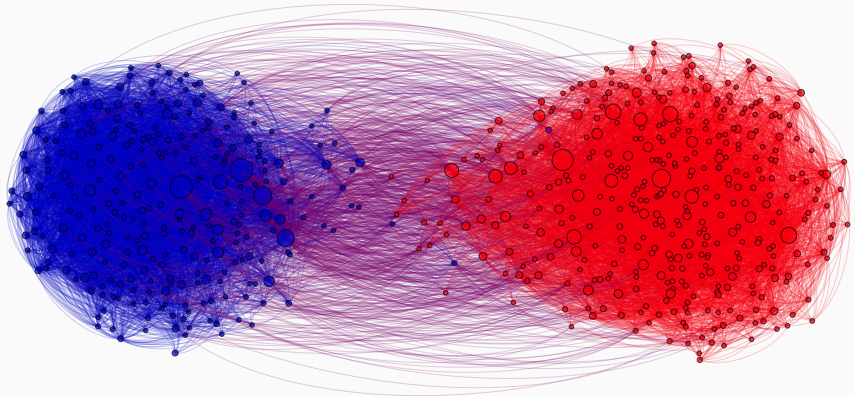


Protein-protein interaction network

Credit: `http://allthingsgraphed.com`

Graph: collection of interconnected nodes

- Social networks
- Power grids
- Transportation networks
- Biological networks
- Web pages



Hyperlinks between American political blogs
Credit: `http://allthingsgraphed.com`

Graph: collection of interconnected nodes

- Social networks
- Power grids
- Transportation networks
- Biological networks
- Web pages
- Graphs as an abstraction (e.g., similarity graphs)

Credit: Michal Valko

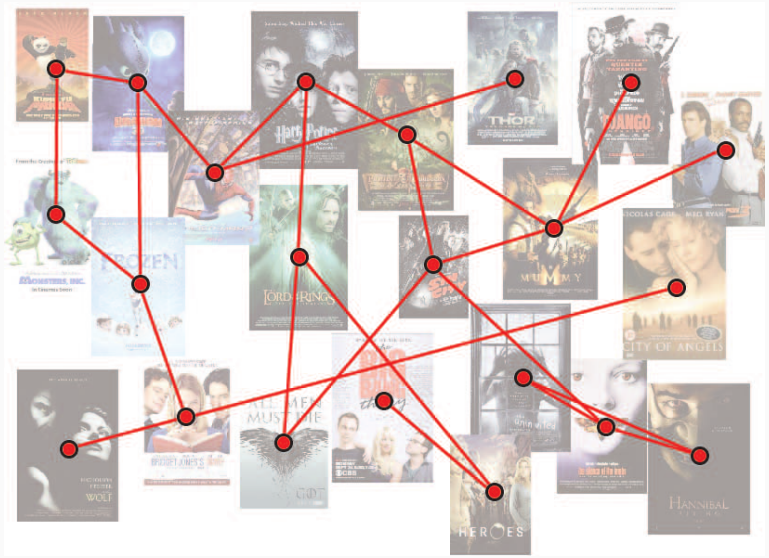
Graph: collection of interconnected nodes

- Social networks
- Power grids
- Transportation networks
- Biological networks
- Web pages
- Graphs as an abstraction (e.g., similarity graphs)

Credit: Michal Valko

Graph: collection of interconnected nodes

- Social networks
- Power grids
- Transportation networks
- Biological networks
- Web pages
- Graphs as an abstraction (e.g., similarity graphs)



Credit: Michal Valko

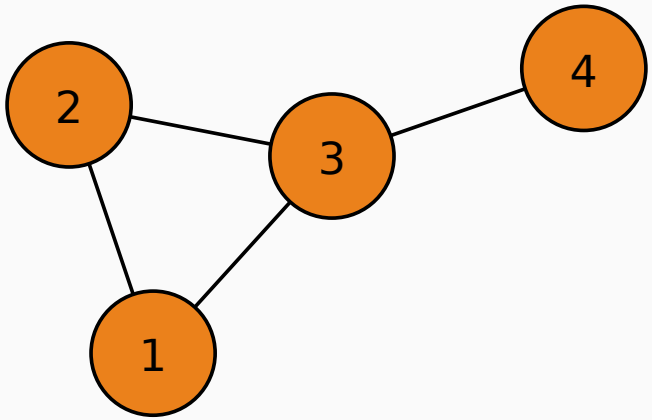
EXAMPLES OF RELEVANT TASKS

- **Graph analysis** (measuring networks)
  - Global study of connectivity and topology
  - Community detection
  - Identification of important (central) nodes
- **Learning in graphs**
  - Link prediction
  - Node classification
  - Graph embedding

BASIC GRAPH NOTIONS

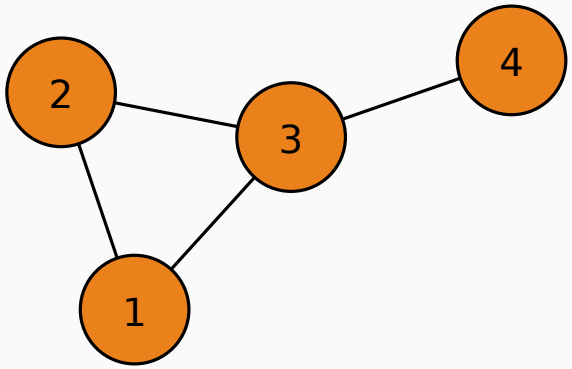
NOTATIONS

- A graph consists of a set of vertices (or nodes) and a set of edges
- More formally, a graph is denoted by $G = (V, E)$ where
  - $V = \{1, \ldots, n\}$ is the set of nodes
  - $E \subseteq V \times V$ is the set of edges
- An edge $(i, j) \in E$ links nodes i and j: we say they are adjacent or neighbors

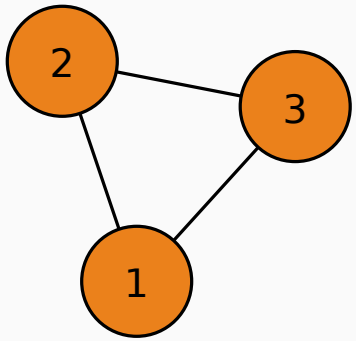


DEGREES

- The degree of a node is equal to its number of neighbors
- A graph is complete if there is an edge between every pair of vertices
- In a complete graph, all nodes thus have degree $n - 1$



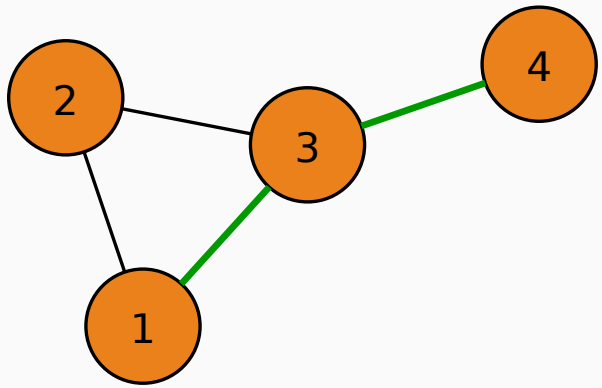
$d = [2, 2, 3, 1]$



$d = [2, 2, 2]$

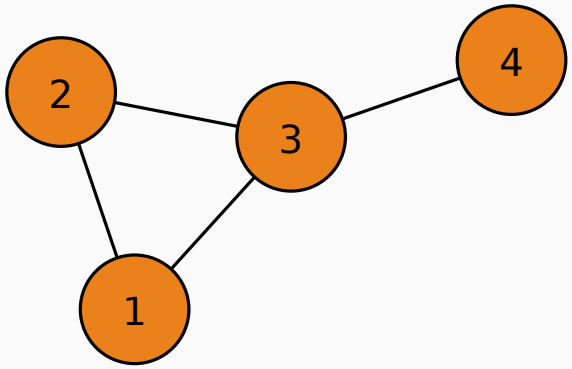
PATHS

- A path from node *i* to node *j* is a sequence of edges from *i* to *j*
- A cycle is a path that starts and ends at the same node
- The length of a path is the number of edges in the path
- A geodesic path is a shortest path between *i* and *j*
- The diameter of a graph is the length of the longest shortest path between any two nodes

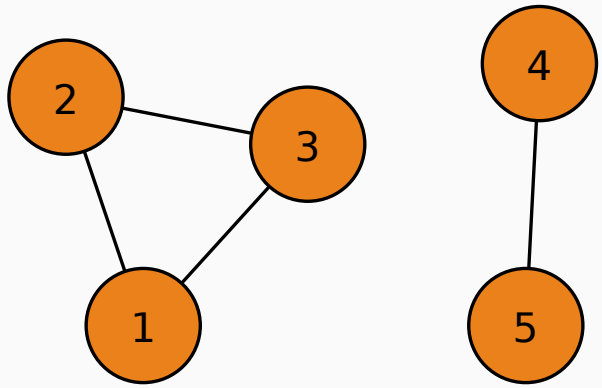


CONNECTIVITY

- All vertices which can be reached from each other by a path form a connected component
- A graph is connected if it has a single connected component



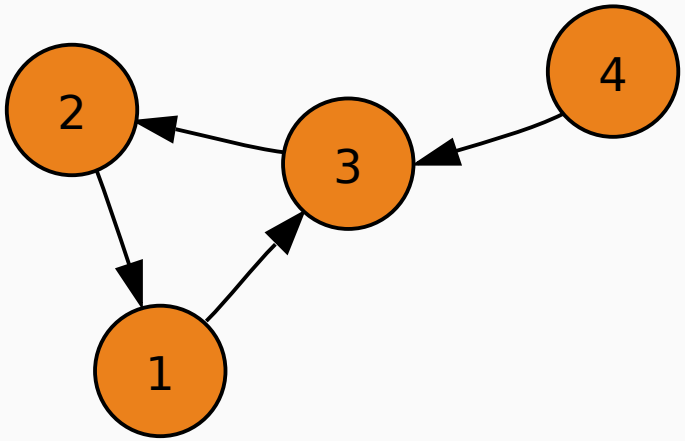
Connected graph



Graph with 2 connected components: $\{1, 2, 3\}$ and $\{4, 5\}$

DIRECTED GRAPH

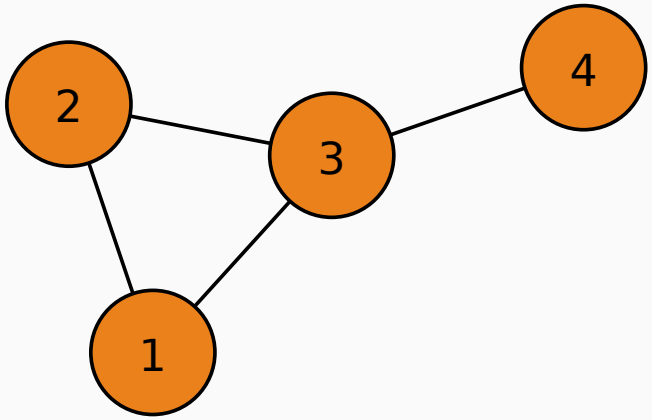
- Until now the graphs we considered were undirected
- In a directed graph, edges are ordered pairs
  - $(i, j) \in E$ points from i to j
  - In-degree of i: number of incoming edges to i
  - Out-degree of i: number of outgoing edges from i



$$\begin{aligned} d^{in} &= [1, 1, 2, 0]^T \\ d^{out} &= [1, 1, 1, 1]^T \end{aligned}$$

COMPUTER REPRESENTATIONS

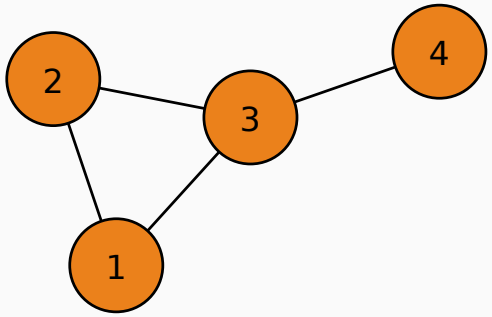
- Text representation: edge list



Corresponding file

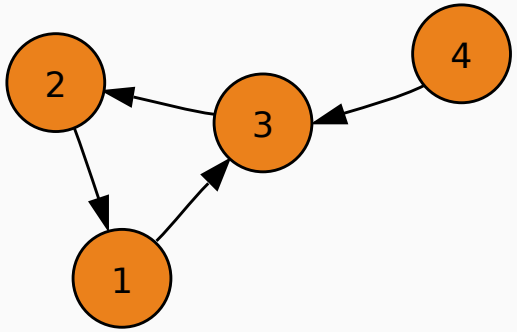
```
1 2
1 3
2 3
3 4
```

- Matrix representation: adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$
  - $A_{i,j} = 1$ if $(i, j) \in E$, else $A_{i,j} = 0$



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

(symmetric)

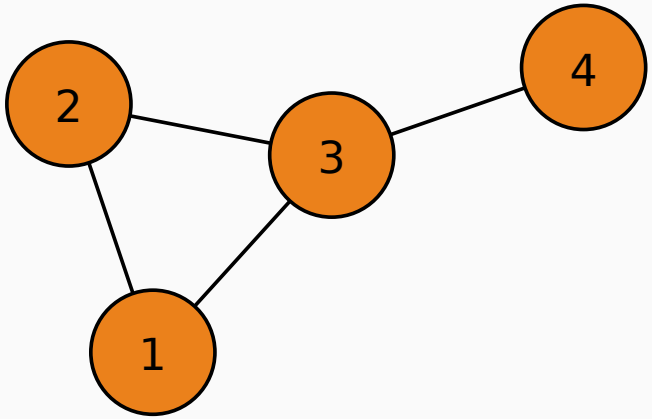


$$\boldsymbol{A} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

(not symmetric)

COMPUTER REPRESENTATIONS

- Linked list representation: adjacency lists



Adjacency lists

```
1: 2 -> 3
2: 1 -> 3
3: 1 -> 2 -> 4
4: 3
```

- The best representation depends on available memory and algorithm of interest

RICHER GRAPHS

- Weighted edges (e.g., distance, similarity score)
- Labels on nodes and/or edges
- Feature vectors associated with nodes and/or edges
- ...

SOME CLASSIC GRAPH PROBLEMS AND ALGORITHMS

- Find shortest paths from a node to all others
  - Algorithm: Dijkstra
  - Time complexity (adjacency lists): $O(|E| + |V| \log(|V|))$
- Graph traversal (visit all nodes of the graph)
  - Algorithm: depth-first search or breadth-first search
  - Time complexity: $O(|E| + |V|)$
  - Can be used to identify connected components
- Traveling Salesman Problem (TSP)
  - Algorithms: approximations and heuristics
  - Time complexity: NP-complete (exponential in graph size)
  - World TSP: `http://www.math.uwaterloo.ca/tsp/world/`

GRAPH ANALYSIS

FROM THE NETWORK TO INDIVIDUAL NODES

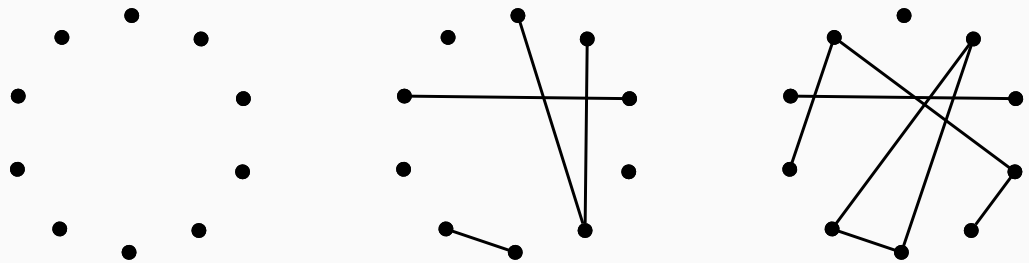
- We can analyze / measure the graph at different scales:
  - Global properties of the network
  - Communities (clusters of nodes)
  - Individual nodes
- In this part we will go from the global scale to the local scale

GLOBAL MEASURES OF NETWORKS

- Many descriptive measures are used to analyze the global properties of a network
  - Degree distribution
  - Clustering coefficient
  - “Small world” phenomena
  - ...
- We will illustrate some of them on two random graph models

ERDÖS–RÉNYI MODEL

- The Erdös–Rényi random graph model has two parameters
  - The number of nodes n
  - A probability $0 \leq p \leq 1$
- A random graph with n nodes is generated by drawing an edge between each pairs of nodes (i, j) independently with probability p
- This models graphs where nodes connect in a random and uniform way

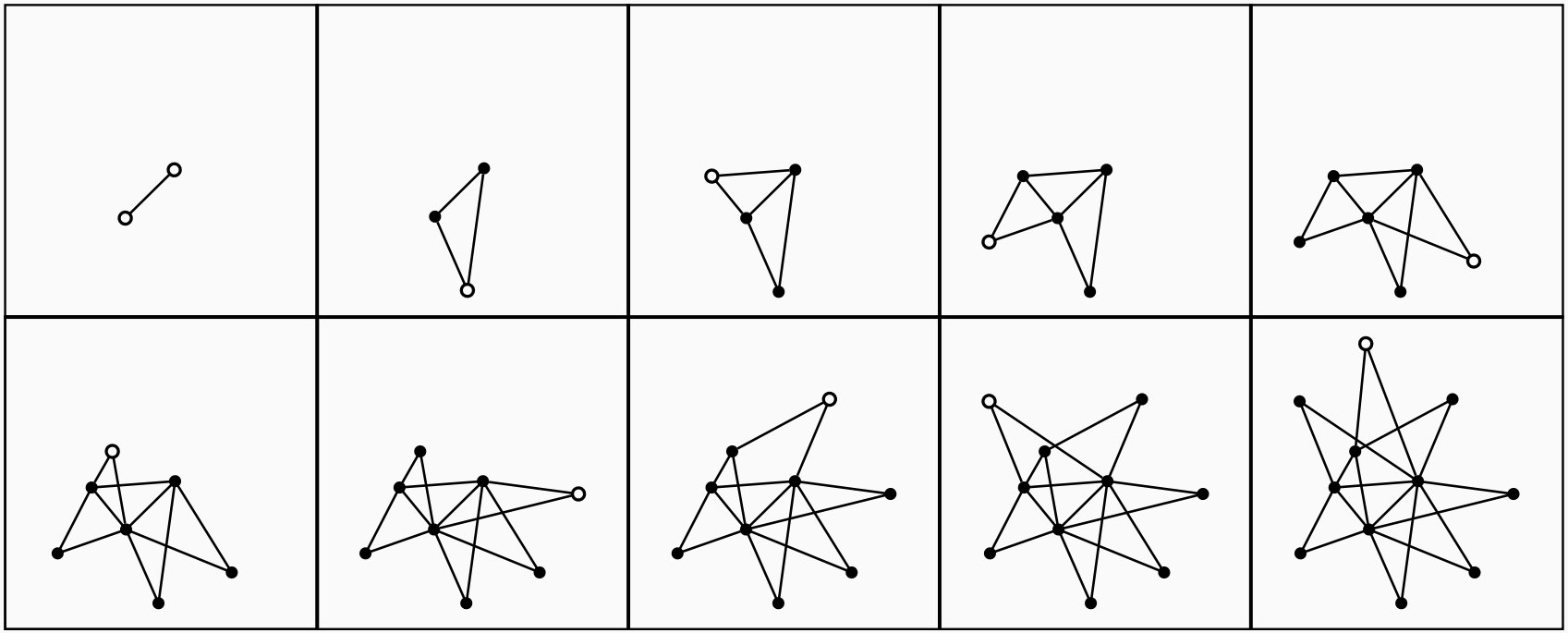


$p = 0$ $p = 0.1$ $p = 0.15$

BARABÁSI-ALBERT MODEL

- The Barabási-Albert random graph model has two parameters
  - An initial graph with n nodes
  - A probability $0 \leq p \leq 1$
- Nodes are added one at a time as follows
  1. With probability p, go to step 2, else go to step 3
  2. Connect new node to n existing nodes chosen uniformly at random
  3. Connect new node to n existing nodes with a probability proportional to their (in-)degree
- This models graphs with preferential attachment, often seen in real networks

BARABÁSI-ALBERT MODEL



DEGREE DISTRIBUTIONS

- Let p_k be the probability that a randomly selected node has degree k
- **Erdös–Rényi**: distribution of degree of a vertex is binomial

$$p_k = \binom{n-1}{k} p^k (1-p)^{n-1-k}$$

  - Highly concentrated around the mean
  - Probability of high degree nodes decreases exponentially fast

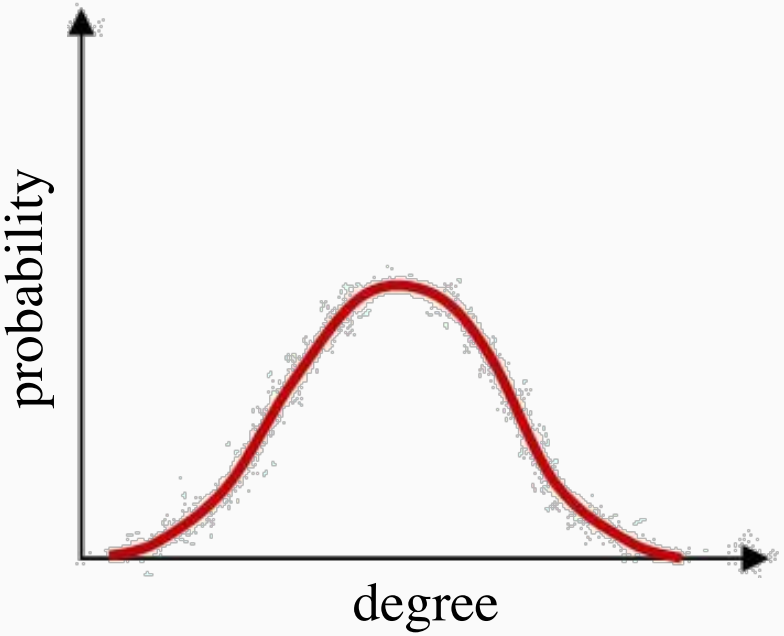
- **Barabási-Albert**: degree distribution follows a power law

$$p_k \propto k^{-\alpha}$$

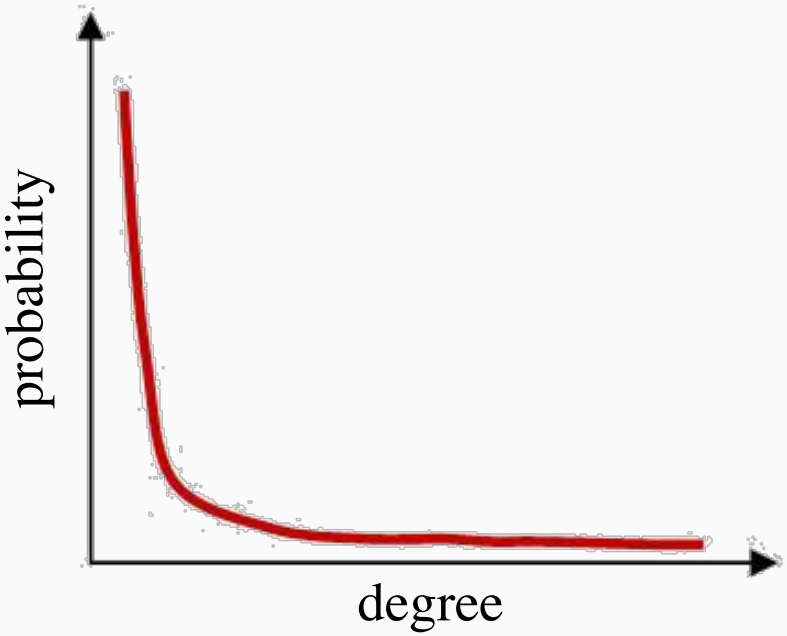
  - Heavy-tailed distribution: non-negligible fraction of high degree
  - Scale-free: average degree is not informative

DEGREE DISTRIBUTIONS

Erdös–Rényi

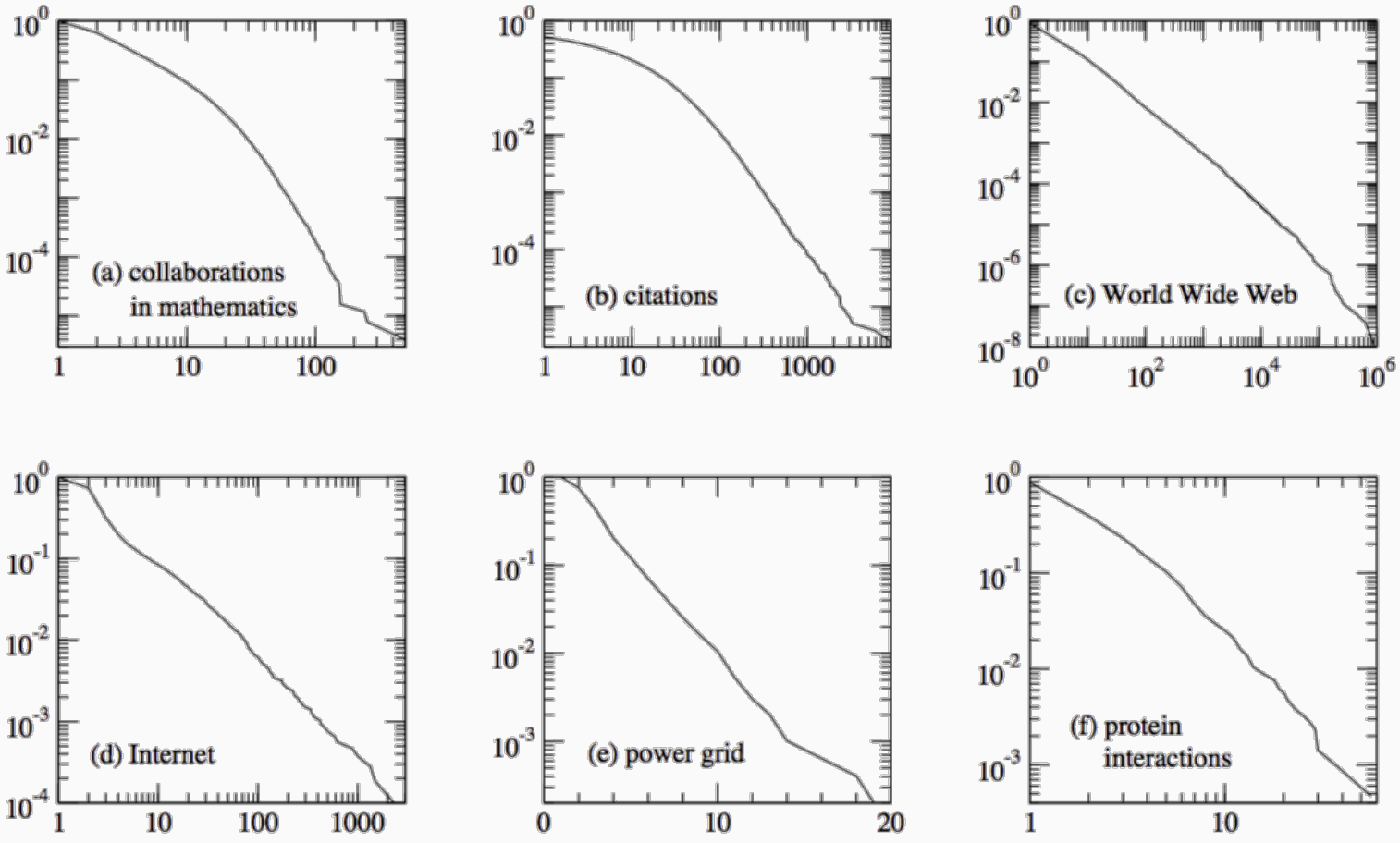


Barabási-Albert



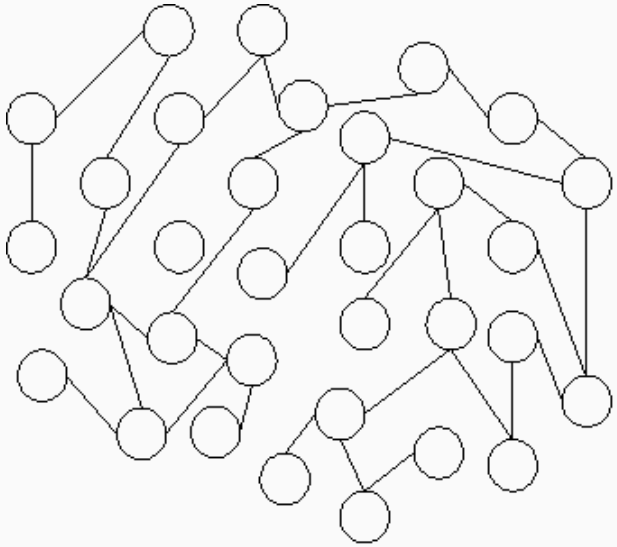
DEGREE DISTRIBUTIONS

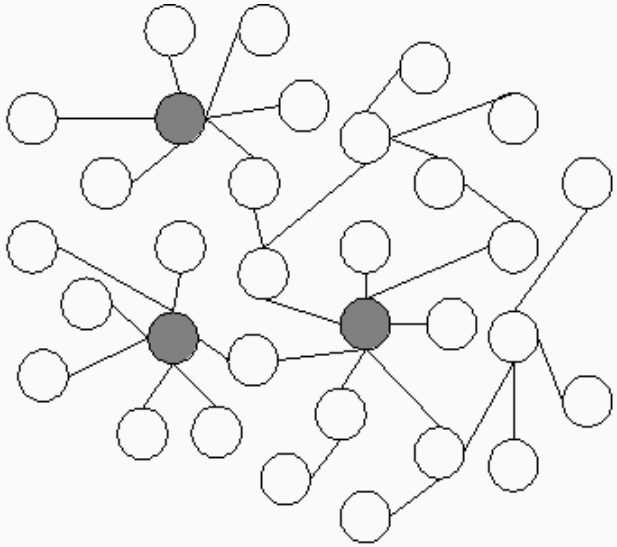
- Power law distributions give roughly a line in the log-log plot
- Many real networks have power law degree distributions



DEGREE DISTRIBUTIONS

- Which graph is uniformly random and which one is scale-free ?





MAXIMUM AND AVERAGE DEGREE

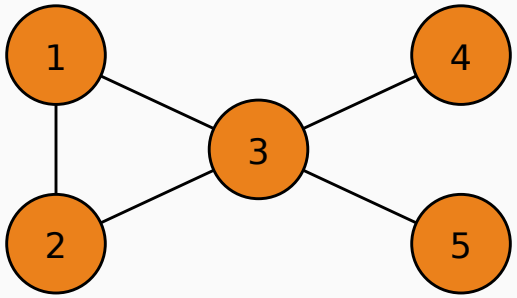
- Erdös–Rényi graphs
  - Average degree is np (in expectation)
  - Maximum degree highly concentrated around average degree
- For power law graphs (Barabási-Albert)
  - Average degree is a constant if $\alpha \geq 2$ (diverges if $\alpha < 2$)
  - Maximum degree is $O\left(n^{\frac{1}{\alpha-1}}\right)$

CLUSTERING COEFFICIENT

- A measure of how well nodes tend to cluster together
- The local clustering coefficient quantifies how close a node i and its neighbors are to being a complete graph

$$C_i = \frac{\text{triangles centered at node } i}{\text{triples centered at node } i}$$

- The global clustering coefficient $CC = \frac{1}{n}\sum_{i=1}^{n} C_i$ measures the density of triangles (local clusters) in the graph



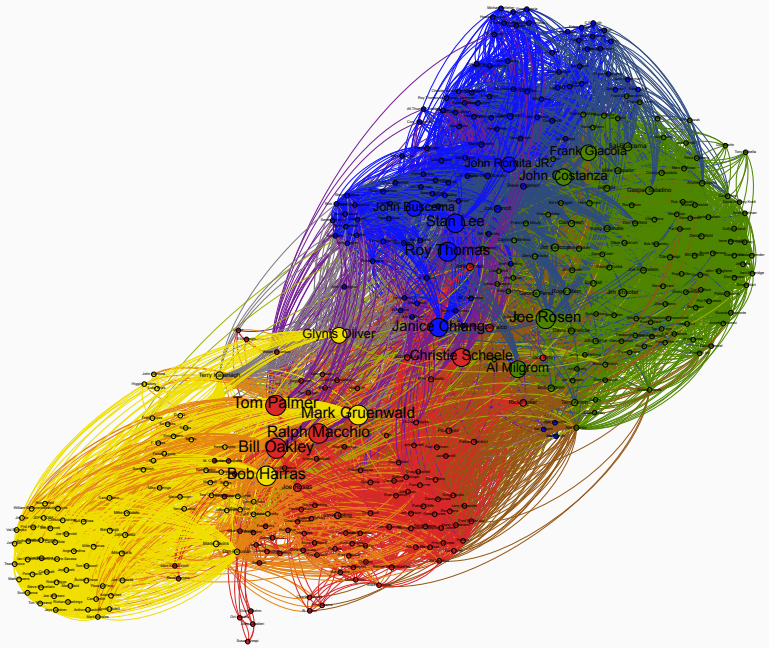
$$CC = \frac{1}{5}\left(1 + 1 + \frac{1}{6} + 0 + 0\right) = \frac{13}{30}$$

CLUSTERING COEFFICIENT

- For Erdös–Rényi random graphs, $\mathbb{E}[CC] = \mathbb{E}[C_i] = p$
  - Probability of two of your neighbors to also be neighbors is p, independently of the local structure
- For Barabási-Albert random graphs
  - CC approximately follows a power law in the number of nodes
  - Let $C(k)$ be the average clustering coefficient of nodes with degree k, then $C(k) \propto k^{-1}$ for Barabási-Albert
- More generally, a power law distribution for $C(k)$ indicates a hierarchical structure
  - Nodes with low degree are connected to other nodes in their community
  - Nodes with high degrees are linked to nodes in different communities

SMALL WORLD PHENOMENON

- Originates from Milgram's small world experiment in the 60's
- How to measure the small world phenomenon?
  - Average length of shortest paths
  - Diameter of the graph (longest shortest path)
  - Length distribution of all shortest paths
  - High clustering coefficient



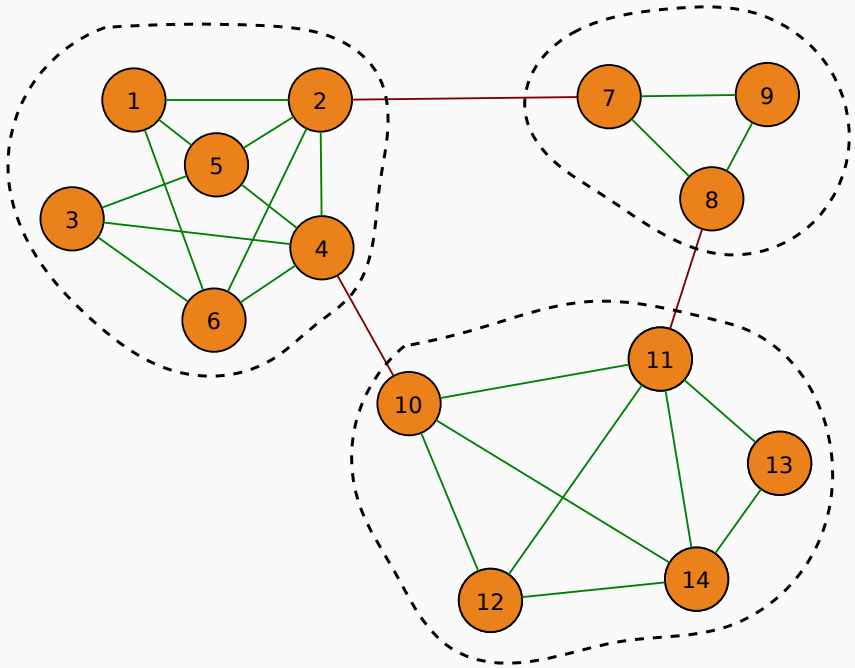
Marvel Comic Book Artist Collaboration Network from `http://allthingsgraphed.com`

COMMUNITY DETECTION

- We seek to partition the nodes of the graph into a set of groups (clusters) according to a certain quality criterion
- Applications: identify
  - social communities
  - clients with similar behavior
  - web pages about the same topic
  - proteins with strong interactions with each other
  - products frequently bought together
  - ...

WHAT IS A GOOD COMMUNITY?

- No universal definition: depends on the application and the network of interest
- General idea: a community is a set of nodes densely connected internally and/or sparsely connected externally



General approach to community detection

1. Define a quality criterion reflecting the desired properties of the communities
    - Many existing criteria (see next slide)
2. Design an algorithm to find the community optimizing the criterion
    - This is generally NP-difficult: for graphs with more than a few hundred nodes, only approximate solutions can be guaranteed

SOME QUALITY CRITERIA

Notations

n: number of nodes in the graph
m: number of edges in the graph
$S \subseteq V$: nodes in the community
n_S: number of nodes in S
m_S: number of edges in S
o_S: number of edges between S and $V \setminus S$

- Based on internal connections:
  - Internal density of edges: $\frac{m_S}{n_S(n_S-1)/2}$
  - Average internal degree: $\frac{2m_S}{n_S}$
- Based on external connections:
  - Expansion: $\frac{o_S}{n_S}$
  - Ratio cut: $\frac{o_S}{n_S(n-n_S)}$

SOME QUALITY CRITERIA

Notations

n: number of nodes in the graph
m: number of edges in the graph
$S \subseteq V$: nodes in the community
n_S: number of nodes in S
m_S: number of edges in S
o_S: number of edges between S and $V \setminus S$

- Based on both internal and external connections:
  - Conductance: $\frac{o_S}{2m_S+o_S}$
  - Normalized cut: $\frac{o_S}{2m_S+o_S} + \frac{o_S}{2(m-m_S)+o_S}$
  - Modularity: $\frac{1}{4}(m_S - \mathbb{E}[m_S])$

- The expectation $\mathbb{E}[m_S]$ is computed with respect to a random process which preserves the degree of each node
  - Each edge is split into two parts (one on each node)
  - Each part is combined to another randomly
- Modularity is then equal to:

$$\frac{1}{2m} \sum_{i,j \in V} \left(A_{ij} - \frac{d_i d_j}{2m} \right) \mathbb{I}\{c_i = c_j\}$$

- Finding the communities maximizing the modularity requires to consider an exponential number of groups $\rightarrow$ very costly even for graphs with a few hundred nodes

LOUVAIN METHOD

- At the beginning, each node has its own community
- The algorithm alternates between two phases until convergence:

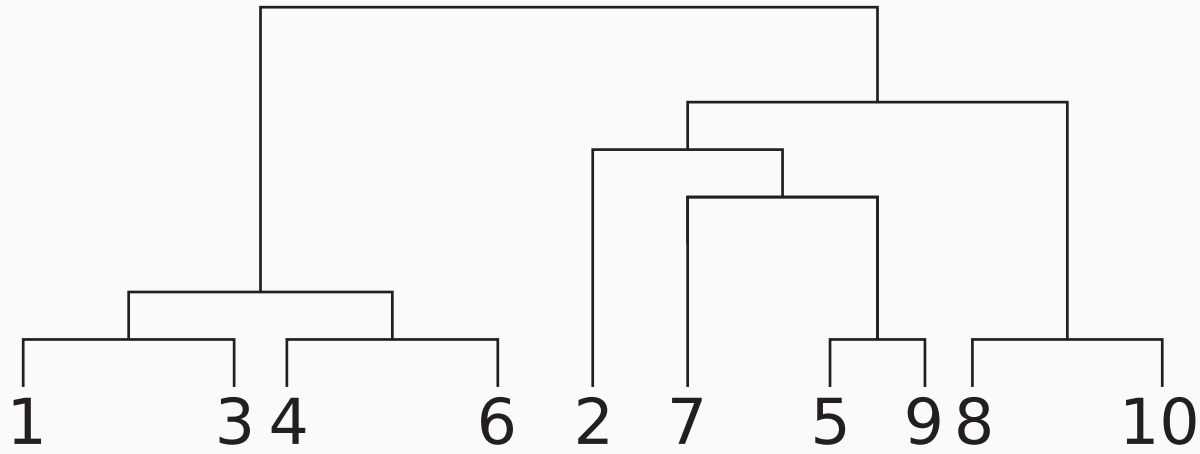
    1. **Optimize local modularity**
       For each node, we create a new community with the neighboring node maximizing the modularity. If no modularity improvement is possible, we keep the node alone.

    2. **Create a new weighted graph**
       The communities of phase 1 become the nodes of the graph. We create a self-loop on each community node weighted by the number of links within the community, and a link between pairs of communities weighted by the number of links between these communities.

- We obtain an approximate solution (no theoretical guarantee but good practical performance)

OTHER APPROACH: HIERARCHICAL CLUSTERING

- It is often relevant to analyze the community structure at different scales (cluster sizes)
- Goal: construct a hierarchy of clusters (represented as a dendrogram)



Hierarchical clustering: bottom-up approach

- In the bottom-up approach, we start with a cluster for each node (as in Louvain method)
- Greedy algorithm: at each iteration, we merge the two "closest" clusters
- We thus need to define a notion of dissimilarity between nodes and between sets of nodes

HIERARCHICAL CLUSTERING: DISTANCES

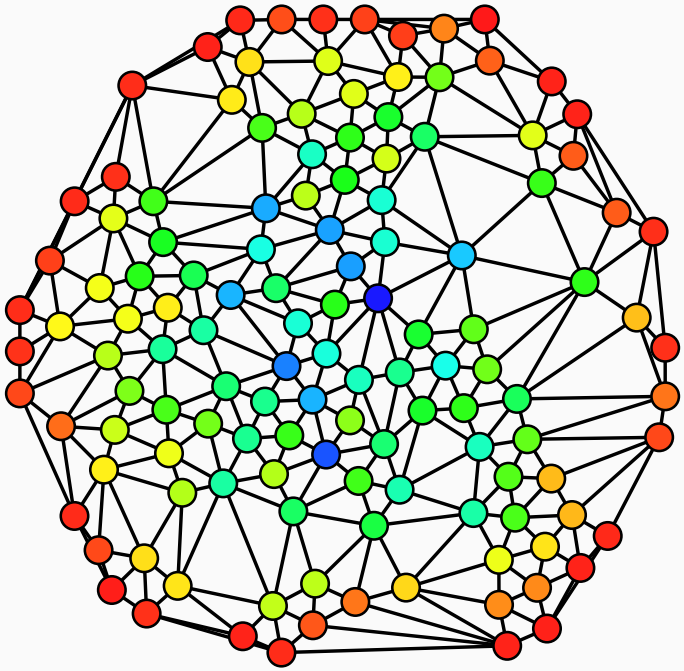
- A natural distance function $d(i, j)$ between 2 nodes i and j is the length of the shortest path between i and j
- Some popular dissimilarity measures between two clusters C_1 and C_2 (linkage criterion):
  - Minimum linkage: $D(C_1, C_2) = \min_{i \in C_1, j \in C_2} d(i, j)$
  - Maximum linkage: $D(C_1, C_2) = \max_{i \in C_1, j \in C_2} d(i, j)$
  - Average linkage: $D(C_1, C_2) = \frac{1}{|C_1||C_2|} \sum_{i \in C_1, j \in C_2} d(i, j)$
  - Centroid linkage: $D(C_1, C_2) = d(G_1, G_2)$ where G_1 and G_2 are the"centers" of C_1 and C_2

IDENTIFY CENTRAL NODES

- Goal: rank the nodes of the graph according to a centrality measure (importance)
- Applications: identify
  - influencers in a social network
  - important (“hub”) web pages
  - bottlenecks in transportation networks
  - products relevant for “loss leader” pricing strategies
  - ...

HOW TO DEFINE CENTRALITY ?

- Again, no universal definition
- Yet, a central notion: walks in graphs
  - A walk is a path which can go through the same node several times
  - Centrality measures vary with the type of walk considered and the way of counting them (number or length)



POPULAR CENTRALITY MEASURES

- Degree centrality: $C(x_i) = d_i$
  - Interpretation: number of walks of length 1 ending at node i
- Eigenvector centrality: $C(x_i) = v_i = \frac{1}{\lambda} \sum_{j=1}^{n} A_{ij} C(x_j)$
  - $\boldsymbol{v}$ satisfies $\boldsymbol{A}\boldsymbol{v} = \lambda \boldsymbol{v}$ where λ is the largest eigenvalue of $\boldsymbol{A}$
  - Interpretation: number of walks of infinite length ending at node i
  - More importance given to nodes with well-connected neighbors
  - Google PageRank is a variant of eigenvector centrality

POPULAR CENTRALITY MEASURES

- Closeness centrality: $C(x_i) = \frac{1}{\sum_{j \neq i} d(i,j)}$
  - $d(i,j)$: length of shortest path between nodes i and j
  - Interpretation: inversely proportional to the sum of lengths of the shortest paths to other nodes
- Betweenness centrality: $C(x_i) = \sum_{j \neq i \neq k} \frac{\sigma_{jk}(i)}{\sigma_{jk}}$
  - σ_{jk}: number of shortest paths between j and k
  - $\sigma_{jk}(i)$: number of shortest paths between j and k going through i
  - Interpretation: number of times the node acts as a "bridge" between two nodes

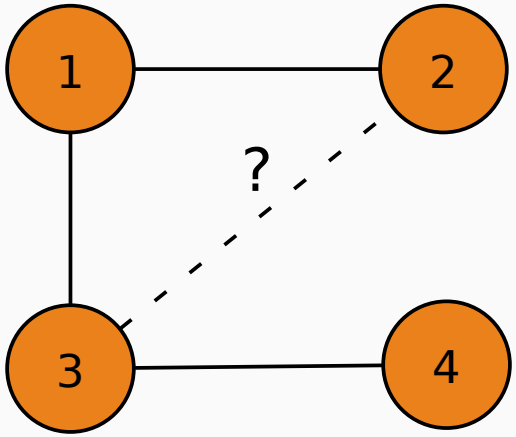
LEARNING IN GRAPHS

LEARNING IN GRAPHS

- So far we have focused on analyzing an observed graph
  - Global properties
  - Communities
  - Node centrality
- In this part we will learn from the observed graph to make predictions
- We will consider two tasks: link prediction and node labeling

LINK PREDICTION

- **Goal**: given a graph G, predict new edges
- These new edges can represent probable future interactions
  - E.g., two persons likely to become friend on Facebook
- They can also be missing edges (partially observed graph)
  - E.g., only a subset of protein-protein interactions are known
  - E.g., not all product combinations have been tried (to see whether they sell well together)



LINK PREDICTION: STANDARD APPROACH

- Use a similarity measure between pairs of nodes to rank potential edges
- Top-ranking edges are the more likely to be correct
- We can thus predict the top-k edges, or use a threshold
- This graph-based strategy can be easily combined with a content-based approach (when data is attached to nodes)

LINK PREDICTION: SIMILARITY SCORES

Notation

$\mathcal{N}(i)$: set of neighbors of node i

- Common neighbors: $S(i,j) = |\mathcal{N}(i) \cap \mathcal{N}(j)|$
- Jaccard coefficient: $S(i,j) = \frac{|\mathcal{N}(i) \cap \mathcal{N}(j)|}{|\mathcal{N}(i) \cup \mathcal{N}(j)|}$
  - Normalized version of common neighbors
- Adamic-Adar index: $S(i,j) = \sum_{k \in \mathcal{N}(i) \cap \mathcal{N}(j)} \frac{1}{\log |\mathcal{N}(k)|}$
  - More weight given to common neighbors of low degree
- Preferential attachment: $S(i,j) = |\mathcal{N}(i)| \cdot |\mathcal{N}(j)|$

LINK PREDICTION: SIMILARITY SCORES

- Some similarity scores also use the community information when available
  - More weight given to neighbors from the same community
- Additional data attached to nodes can be easily integrated as part of the score
  - E.g., cosine similarity between node feature vectors
  - E.g., classifier trained to predict the presence of an edge from data at two nodes

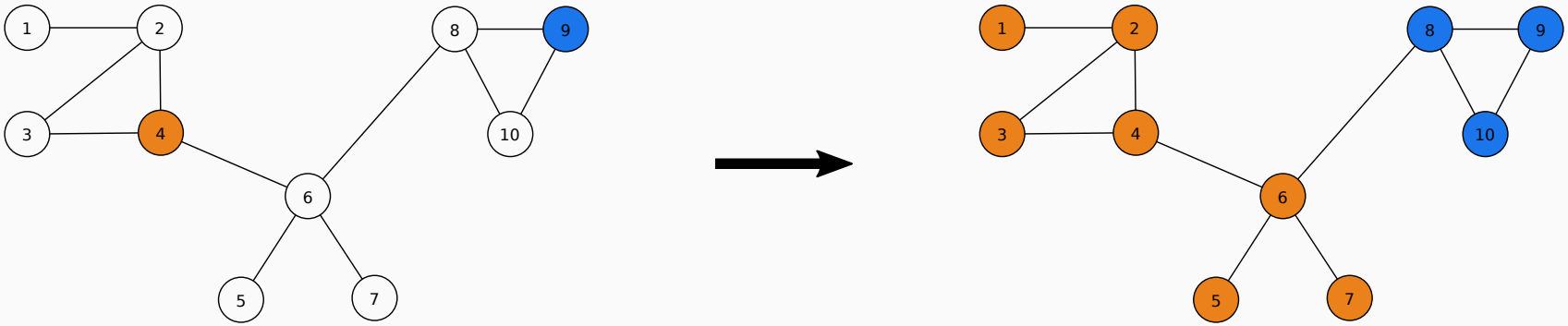
LINK PREDICTION: EVALUATION

- How to evaluate the accuracy in link prediction and perform model selection?
- Practical approach: hide a subset of node pairs and predict based on the rest of the graph
- Performance measures:
  - Dense graphs: proportion of correct predictions
  - Sparse graphs: Area under the ROC Curve (AUC)

$$AUC(S) = \frac{1}{|E^+||E^-|} \sum_{e^+} \sum_{e^-} \mathbb{I}\{S(e^+) > S(e^-)\}$$

NODE LABELING

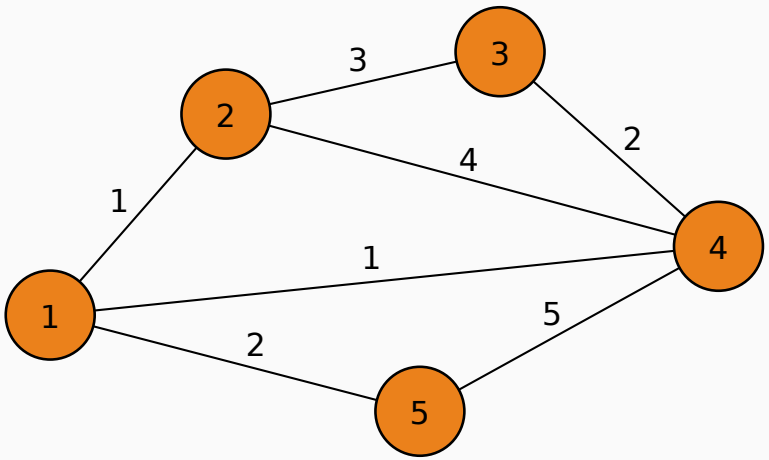
- **Goal**: given a graph where some nodes are labeled, predict missing node labels
- This is a semi-supervised learning problem
- **Central assumption**: correct labels are smooth on the graph
  - Classification: two neighbors tend to have the same label
  - Regression: two neighbors tend to have similar target values



GRAPH LAPLACIAN MATRIX

- For a graph $G = (V, E)$ we denote by

A	adjacency matrix
W	weight matrix
D	(diagonal) degree matrix
$L = D - W$	Laplacian matrix (symmetric)



$$L = \begin{pmatrix} 4 & -1 & 0 & -1 & -2 \\ -1 & 8 & -3 & -4 & 0 \\ 0 & -3 & 5 & -2 & 0 \\ -1 & -4 & -2 & 12 & -5 \\ -2 & 0 & 0 & -5 & 7 \end{pmatrix}$$

REFRESHER ON EIGENVECTORS AND EIGENVALUES

- Let $L \in \mathbb{R}^{n \times n}$ symmetric matrix
- A vector $v \in \mathbb{R}^n$ is an eigenvector of L of eigenvalue $\lambda \in \mathbb{R}$ if

$$Lv = \lambda v$$

- If (λ_1, v_1), (λ_2, v_2) are eigenpairs for L with $\lambda_1 \neq \lambda_2$ then $v_1 \perp v_2$, i.e., $v_1^T v_2 = 0$
- If (λ, v_1), (λ, v_2) are eigenpairs for L, then $(\lambda, v_1 + v_2)$ is also en eigenpair
- The multiplicity of eigenvalue λ is the dimension of the space of eigenvectors corresponding to λ
- L has n eigenvalues (counting possible multiplicities)

- $\boldsymbol{L}$ is symmetric positive semi-definite (PSD) since for any $\boldsymbol{f} \in \mathbb{R}^n$

$$\begin{aligned}
\boldsymbol{f}^T \boldsymbol{L} \boldsymbol{f} &= \boldsymbol{f}^T \boldsymbol{D} \boldsymbol{f} - \boldsymbol{f}^T \boldsymbol{W} \boldsymbol{f} \\
&= \sum_i d_i f_i^2 - \sum_i \sum_j w_{i,j} f_i f_j = \sum_i \sum_j w_{i,j} f_i^2 - \sum_i \sum_j w_{i,j} f_i f_j \\
&= \frac{1}{2} \left(\sum_i \sum_j w_{i,j} f_i^2 - 2 \sum_i \sum_j w_{i,j} f_i f_j + \sum_i \sum_j w_{i,j} f_i^2 \right) \\
&= \frac{1}{2} \sum_i \sum_j w_{i,j} (f_i - f_j)^2 \geq 0
\end{aligned}$$

- Since $\boldsymbol{L}$ is PSD, its eigenvalues satisfy $0 \leq \lambda_1 \leq \cdots \leq \lambda_n$
- We can easily see that $(0, \mathbf{1}_n)$ is an eigenpair for $\boldsymbol{L}$

Theorem

The multiplicity of eigenvalue 0 *of* $\boldsymbol{L}$ *is equal to the number of connected components of the graph. The eigenspace of* 0 *is spanned by the components' indicators.*

Proof.

If $(0, \boldsymbol{f})$ is an eigenpair, then $0 = \frac{1}{2}\sum_{i,j} w_{i,j}(f_i - f_j)^2$, hence $\boldsymbol{f}$ is constant on each connected component. If there are k connected components, $\boldsymbol{L}$ is k-block-diagonal:

$$\boldsymbol{L} = \begin{bmatrix} \boldsymbol{L}_1 & & & \\ & \boldsymbol{L}_2 & & \\ & & \ddots & \\ & & & \boldsymbol{L}_k \end{bmatrix}$$

The spectrum of block-diagonal matrices is the union of the spectra of $\boldsymbol{L}_i$ (padded with zeros). The theorem follows from the fact that for $i = 1, \ldots, k$, $(0, \mathbf{1}_{|V_i|})$ is an eigenpair for $\boldsymbol{L}_i$, where V_i is the set of nodes in the i^{th} connected component. □

SMOOTHNESS OF A GRAPH FUNCTION

- A graph function is a vector $\boldsymbol{f} \in \mathbb{R}^n$ assigning values to nodes

$$\boldsymbol{f} : V \to \mathbb{R}$$

- The smoothness of a graph function is given by the quadratic form of the Laplacian

$$S_G(\boldsymbol{f}) = \boldsymbol{f}^T \boldsymbol{L} \boldsymbol{f} = \frac{1}{2} \sum_{i,j} w_{i,j} (f_i - f_j)^2$$

- When $S_G(\boldsymbol{f})$ is small, $\boldsymbol{f}$ does not vary much in high density regions of the graph

SYMMETRIC NORMALIZED LAPLACIAN

- We can also consider a symmetric normalized Laplacian matrix

$$L_{sym} = D^{-1/2} L D^{-1/2} = I - D^{-1/2} W D^{-1/2}$$

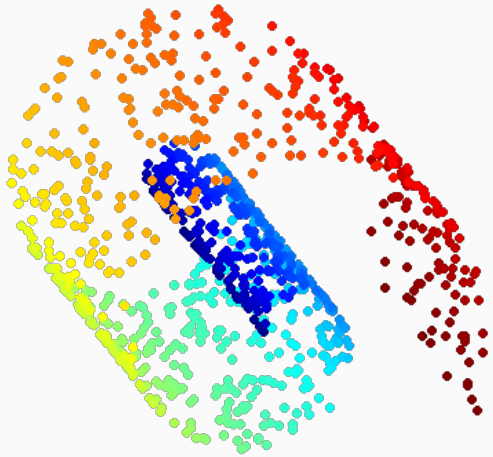
- Normalized variant of smoothness

$$f^T L_{sym} f = \frac{1}{2} \sum_{i,j} w_{i,j} \left(\frac{f_i}{\sqrt{d_i}} - \frac{f_j}{\sqrt{d_j}} \right)^2$$

- L_{sym} is also PSD, and $(0, D^{1/2} \mathbf{1}_n)$ is an eigenpair for L_{sym}

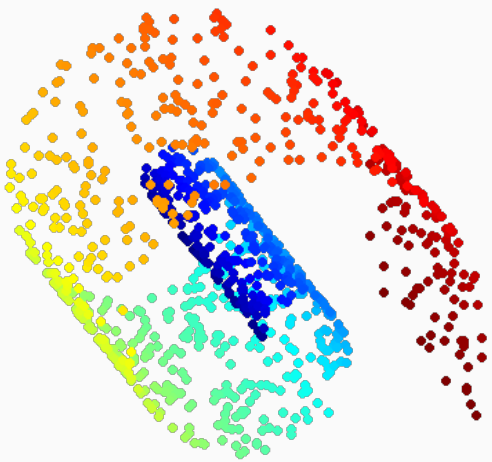
BEYOND EXPLICIT GRAPHS: DATA ON A MANIFOLD

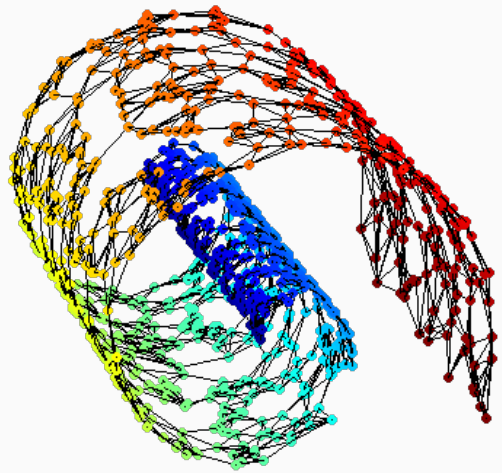
- There is a natural link between discrete representations (graphs) and continuous representations
- Metric space: distances between all points in the space are defined (e.g., Euclidean space)
- Manifold: every point has a neighborhood which is homeomorphic to the Euclidean space
  - Locally Euclidean (distance in small region is meaningful)
  - Global structure more complex (Euclidean distance between "distant" points is meaningless)



SIMILARITY GRAPHS AND MANIFOLD STRUCTURE

- Let $x_1, \ldots, x_n$ be a set of data points with similarity matrix S
  - $S_{i,j} \geq 0$: similarity score between x_i and x_j
  - Can use standard / handcrafted / learned similarity measure
- Similarity graph: $(x_i, x_j) \in E$ if $S_{i,j}$ large enough
- Such a graph can approximate the manifold structure!
- For this to work we must enforce locality (sparsify the graph)

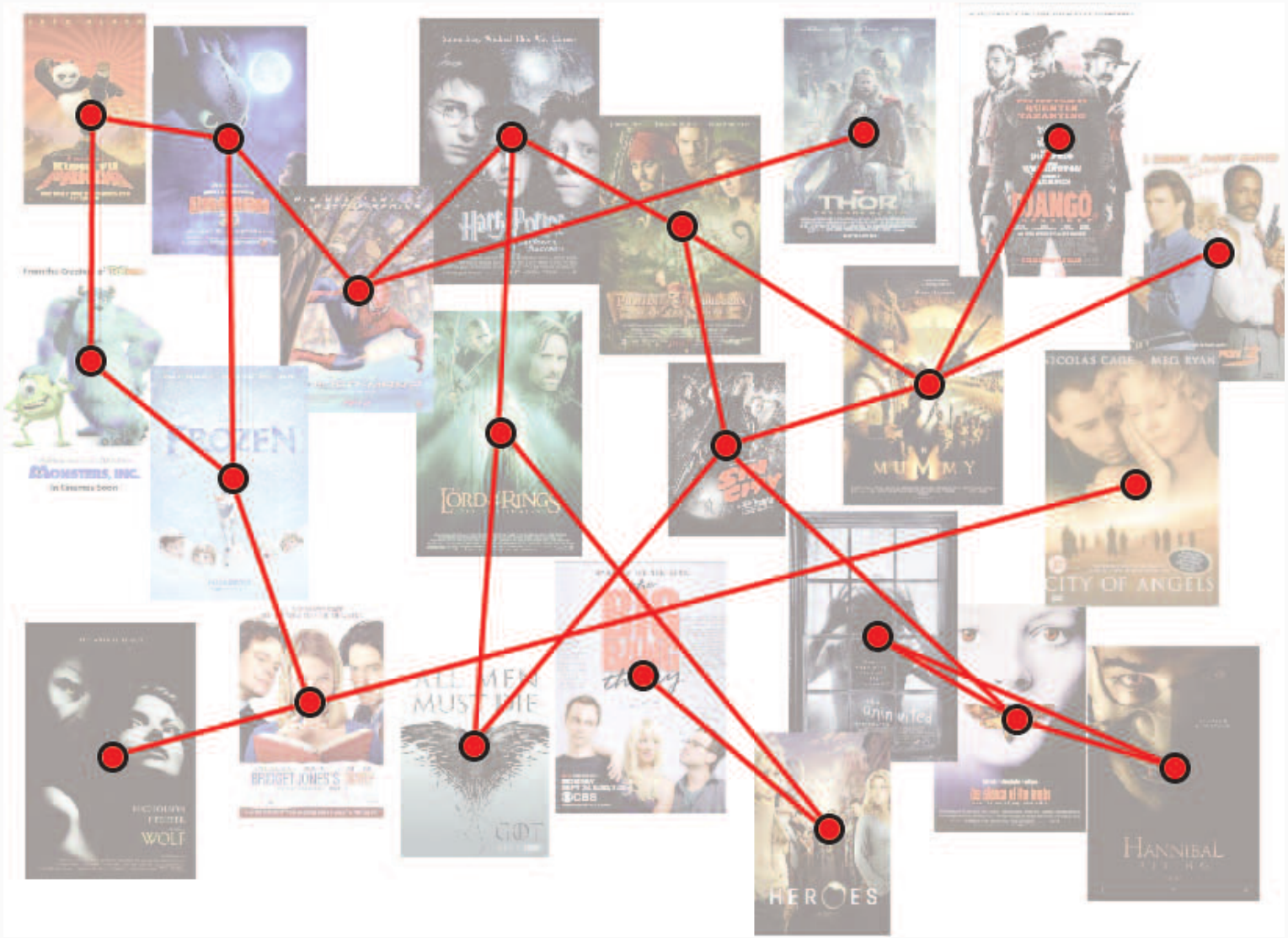




SPARSIFICATION OF SIMILARITY GRAPHS

- k-nearest neighbor graph
  - Connect i and j if x_i is among the k-nearest neighbors of v_j and/or x_j is among the k-nearest neighbors of v_i
- ϵ-neighborhood graph
  - Connect i and j if $S_{i,j}$ is larger than ϵ
- Exponential graph
  - Weight each edge $(i, j) \in V^2$ by $S_{i,j}$
  - Must use with fast-decaying similarity to enforce locality
  - Typical choice is the Gaussian kernel $S_{i,j} = \exp(-\gamma \|x_i - x_j\|^2)$
- Some issues
  - Little theoretical underpinning to guide graph construction
  - Must tune k, ϵ or γ to adjust locality
  - For efficiency reasons, we like to deal with sparse graphs

EXAMPLE: MOVIE SIMILARITY GRAPH



MANIFOLD REGULARIZATION

- Assume data lies on a nonlinear manifold $\mathcal{M} \subset \mathcal{X}$
- We want to learn a function $f : \mathcal{M} \to \mathbb{R}$ which varies smoothly in dense regions
- Natural choice is to enforce small gradient along $\mathcal{M}$ where the marginal probability density is large

$$\|f\|_I^2 = \int_{x \in \mathcal{M}} \|\nabla_{\mathcal{M}} f(x)\|^2 P(x) dx$$

- $P(x)$ is unknown but we can approximate it using n labeled/unlabeled points [Belkin et al., 2006]

$$\|f\|_I^2 \approx \frac{1}{n^2} \boldsymbol{f}^T \boldsymbol{L} \boldsymbol{f} = \frac{1}{2} \sum_{i,j} w_{i,j} (f_i - f_j)^2$$

under some conditions (appropriately scaled exponential graph)

MANIFOLD REGULARIZATION

- **Manifold regularization**: use the quadratic form of the Laplacian as regularizer for machine learning models
- Generic way to make use of unlabeled data in supervised learning algorithms $\rightarrow$ semi-supervised learning
- Smoothness assumption becomes the manifold assumption: points connected via a path through high density regions on the data manifold are likely to have a similar label
- Many successful algorithms: Laplacian eigenmaps, Laplacian SVMs, label propagation, online node labeling...

SEMI-SUPERVISED LABEL PROPAGATION: NOTATIONS

- Let $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_l, \boldsymbol{x}_{l+1}, \ldots, \boldsymbol{x}_n \in \mathbb{R}^p$
- We have labels $y_1, \ldots, y_l \in \{1, \ldots, C\}$ for the first l points
- Build exponential graph with $W_{i,j} = \exp(-\gamma \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2)$
- Define initial label matrix $\boldsymbol{Y} \in \mathbb{R}^{n \times C}$ such that

$$Y_{i,j} = \begin{cases} 1 & \text{if } \boldsymbol{x}_i \text{ has label } y_i = j \\ 0 & \text{otherwise} \end{cases}$$

- The algorithm will generate a prediction matrix $\boldsymbol{F} \in \mathbb{R}^{n \times C}$ from which we will predict the label of a node i using

$$\hat{y}_i = \arg\max_j F_{i,j}$$

SEMI-SUPERVISED LABEL PROPAGATION: FORMULATION

- The prediction matrix $\boldsymbol{F} \in \mathbb{R}^{n \times C}$ is the one minimizing the following objective function [Zhou et al., 2003]

$$\min_{\boldsymbol{F} \in \mathbb{R}^{n \times c}} \quad \frac{1}{2} \Bigg(\underbrace{\sum_{i,j=1} w_{i,j} \left\| \frac{\boldsymbol{F}_i}{\sqrt{d_i}} - \frac{\boldsymbol{F}_j}{\sqrt{d_j}} \right\|^2}_{\text{smoothness term}} + \mu \underbrace{\sum_{i=1}^{n} \|\boldsymbol{F}_i - \boldsymbol{Y}_i\|^2}_{\text{fit known labels}} \Bigg)$$

- Trade-off between two terms (ruled by $\mu \geq 0$)
  - Smoothing predictions with normalized Laplacian
  - Keeping accurate predictions for labeled points

SEMI-SUPERVISED LABEL PROPAGATION: SOLUTION

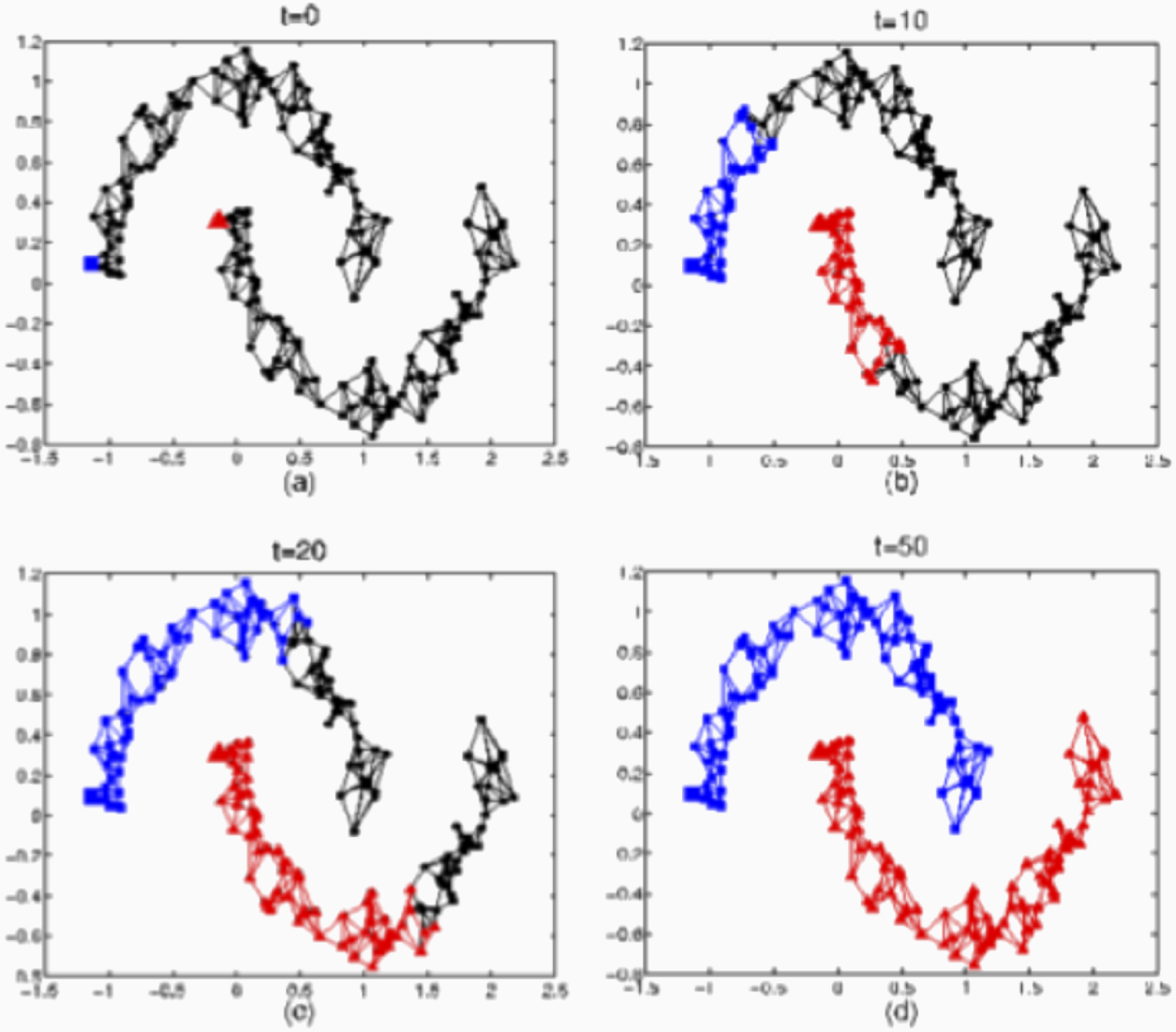
- The objective function is convex and quadratic, so there is a closed-form solution (found by setting the gradient to zero)

$$F^* = ((1-\alpha)I + L_{sym})^{-1}Y, \quad \text{with } \alpha = 1/(1+\mu)$$

- Inverting $(1-\alpha)I + L_{sym}$ is costly for large graphs
- Equivalent and cheaper iterative algorithm:
  1. Initialize $F(0) = Y$
  2. Iterate the following until convergence

$$F(t+1) = \alpha(I - L_{sym})F(t) + (1-\alpha)Y$$

Semi-supervised label propagation: illustration



References I

[Belkin et al., 2006] Belkin, M., Niyogi, P., and Sindhwani, V. (2006).
Manifold regularization: A geometric framework for learning from labeled and unlabeled examples.
Journal of Machine Learning Research, 7:2399–2434.

[Chung, 1997] Chung, F. R. K. (1997).
***Spectral Graph Theory*, volume 92.**
American Mathematical Society.

[Kolaczyk, 2009] Kolaczyk, E. D. (2009).
Statistical Analysis of Network Data: Methods and Models.
Springer.

[von Luxburg, 2007] von Luxburg, U. (2007).
A Tutorial on Spectral Clustering.
Statistics and Computing, 17(4):395–416.

[Zhou et al., 2003] Zhou, D., Bousquet, O., Lal, T. N., Weston, J., and Schölkopf, B. (2003).
Learning with Local and Global Consistency.
In *Advances in Neural Information Processing Systems 16*, pages 321–328.