Mid-term Review

CS 579: Online Social Network Analysis

Introduction

Kai Shu

Spring 2022

Definition

Social Media is the use of electronic and Internet tools for the purpose of sharing and discussing information and experiences with other human beings in more efficient ways.

Main Characteristics of Social Media

- Participation
  - social media encourages contributions and feedback from everyone who is interested. It blurs the line between media and audience.
- Openness
  - most social media services are open to feedback and participation. They encourage voting, comments and the sharing of information. There are rarely any barriers to accessing and making use of content – password-protected content is frowned on.
- Conversation
  - whereas traditional media is about "broadcast" (content transmitted or distributed to an audience) social media is better seen as a two-way conversation.
- Community
  - social media allows communities to form quickly and communicate effectively. Communities share common interests, such as a love of photography, a political issue or a favorite TV show.
- Connectedness
  - most kinds of social media thrive on their connectedness, making use of links to other sites, resources and people.

Social Media Mining *is the process of representing, analyzing, and extracting meaningful patterns from social media data*

CS 579: Online Social Network Analysis

Chapter 2 Graph Essentials

Kai Shu

Spring 2022

Social Networks and Social Network Analysis

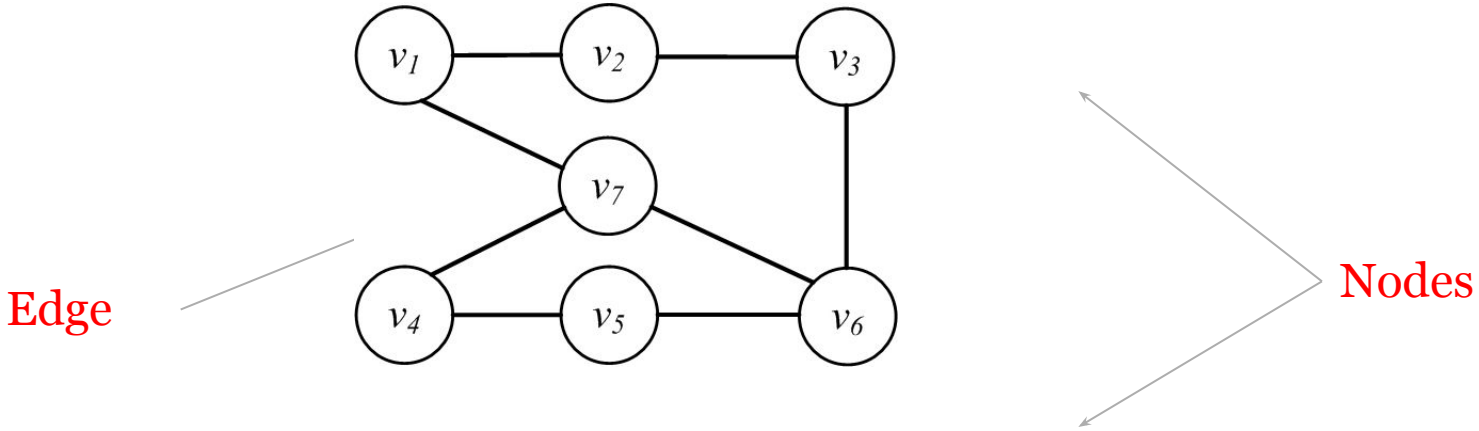
- A social network
  - A network where elements have a social structure
    - A set of actors (such as individuals or organizations)
    - A set of ties (connections between individuals)
- Examples of social networks:
  - Our family networks, our friendship networks, our colleagues, etc.
- To analyze these networks, we use Social Network Analysis (SNA)
- Social Network Analysis is an interdisciplinary field from social sciences, statistics, physics, graph theory, complex networks, computer science, …

Graph Basics

Nodes and Edges

A network is a graph, or a collection of *points* connected by *lines*

- Points are referred to as **nodes, actors,** or **vertices**
- Connections are referred to as **edges** or **ties**



Nodes or Actors

- In a *friendship* graph, *nodes* are people and any pair of people connected denotes the *friendship* between them
- Depending on the context, these nodes are called nodes, or actors
  - In a web graph, "nodes" represent sites and the connection between nodes indicates web-links between them
  - In a social setting, these nodes are called actors

$$V = \{v_1, v_2, \ldots, v_n\}$$

  - The size of the graph is |V|= **n**

Edges or Ties

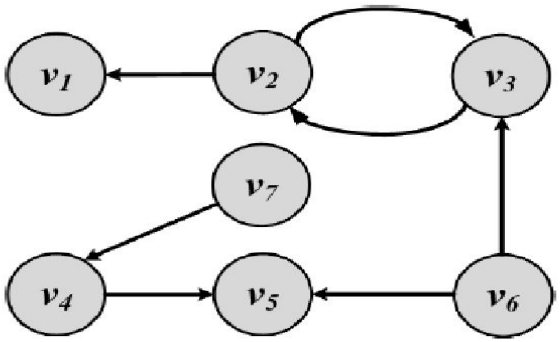
- Edges connect nodes and are also known as **ties** or **relationships**
- In a social setting, where *nodes* represent **social entities** such as people, *edges* indicate internode relationships and are therefore known as *relationships* or (social) *ties*

$$E = \{e_1, e_2, \ldots, e_m\}$$

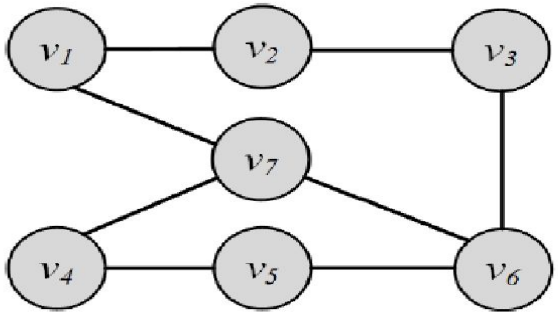
- Number is edges (size of the edge-set) is denoted as $|E|=\mathbf{m}$

Directed Edges and Directed Graphs

- Edges can have directions. A directed edge is sometimes called an *arc*



(a) Directed Graph



(b) Undirected Graph

- Edges are represented using their end-points (or end-nodes), **e(v2,v1)**. In undirected graphs, both representations are the same, i.e., **e(v1,v2)=e(v2,v1)**

Neighborhood and Degree (In-degree and out-degree)

- For any node v, the set of nodes it is connected to via an edge is called its neighborhood and is represented as ***N(v)***

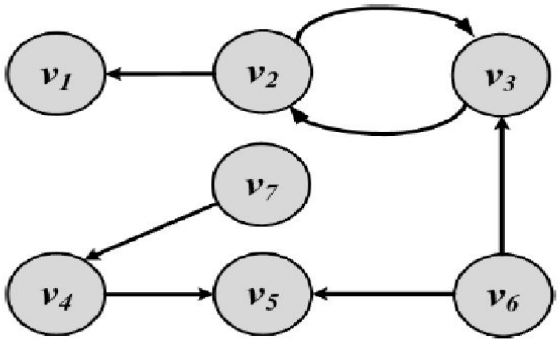
- The number of edges connected to one node is the **degree** of that node (the size of its neighborhood)
  - Degree of a node i is usually presented using notation d_i
  - In case of directed graphs,

d_i^{in} • In-degrees is the number of edges pointing towards a node

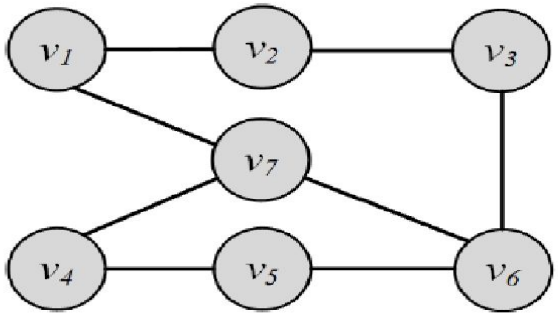
d_i^{out} • Out-degree is the number of edges pointing away from a node

Let's look at some examples

- N(v1) = ?
- degree of node v1?
- in-degree and out-degree of v1?



(a) Directed Graph



(b) Undirected Graph

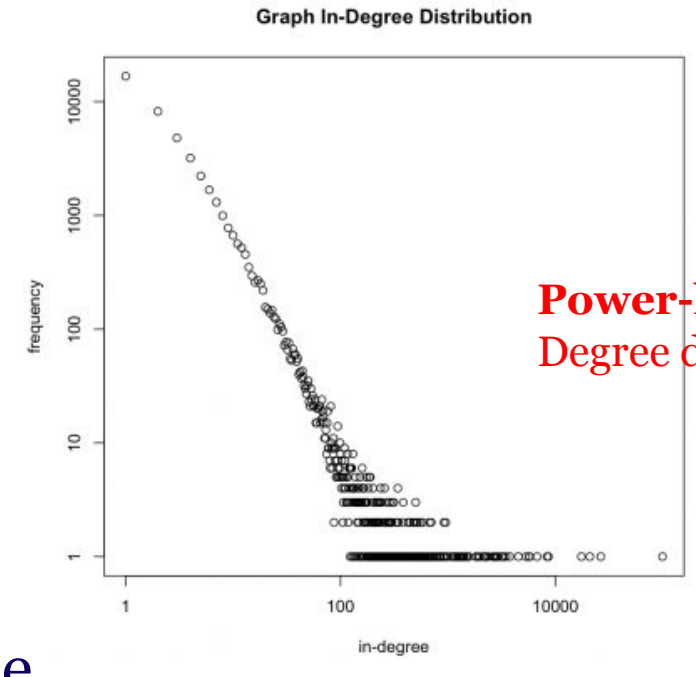
Degree Distribution

When dealing with very large graphs, how nodes' degrees are distributed is an important concept to analyze and is called ***Degree Distribution***

- Degree sequence

$$\pi(d) = \{d_1, d_2, \ldots, d_n\}$$

- Let's revisit Slide 18(a)



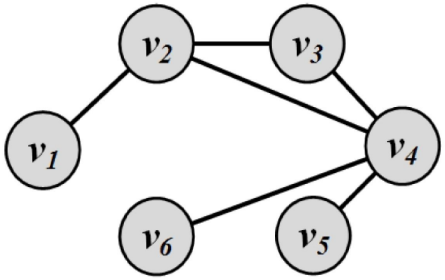
Power-law
Degree distribution

Degree distribution histogram

– The x-axis represents the degree and the y-axis represents the number of nodes having that degree

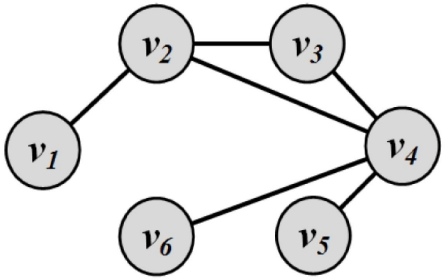
Graph Representation

- Adjacency Matrix
- Adjacency List
- Edge List



Adjacency Matrix

$$A_{ij} = \begin{cases} 1, \text{ if there is an edge between nodes } v_i \text{ and } v_j \\ 0, \text{ otherwise} \end{cases}$$



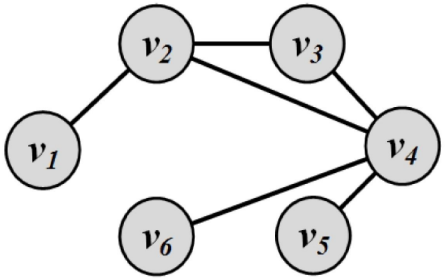
	v_1	v_2	v_3	v_4	v_5	v_6
v_1	0	1	0	0	0	0
v_2	1	0	1	1	0	0
v_3	0	1	0	1	0	0
v_4	0	1	1	0	1	1
v_5	0	0	0	1	0	0
v_6	0	0	0	1	0	0

- Diagonal Entries are self-links or loops

Social media networks have very sparse adjacency matrices

Adjacency List

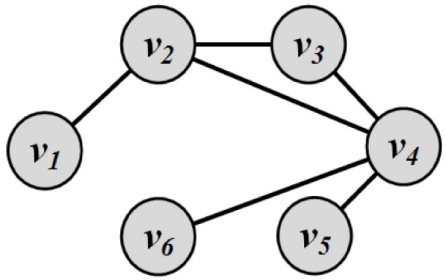
- In an adjacency list for every node, we maintain a list of all the nodes that it is connected to
- The list is usually **sorted** based on the node order or other preferences



Node	Connected To
v_1	v_2
v_2	v_1, v_3, v_4
v_3	v_2, v_4
v_4	v_2, v_3, v_5, v_6
v_5	v_4
v_6	v_4

Edge List

- In this representation, each element is an edge and is usually represented as *(u, v)*, denoting that node *u* is connected to node *v* via an edge



(v_1, v_2)
(v_2, v_3)
(v_2, v_4)
(v_3, v_4)
(v_4, v_5)
(v_4, v_6)

Types of Graphs

- **Null, Empty, Directed/Undirected/Mixed, Simple/Multigraph, Weighted, Webgraph, Signed Graph**

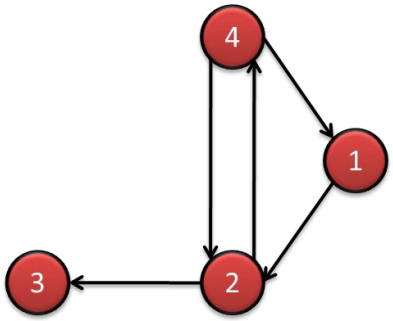
Null Graph and Empty Graph

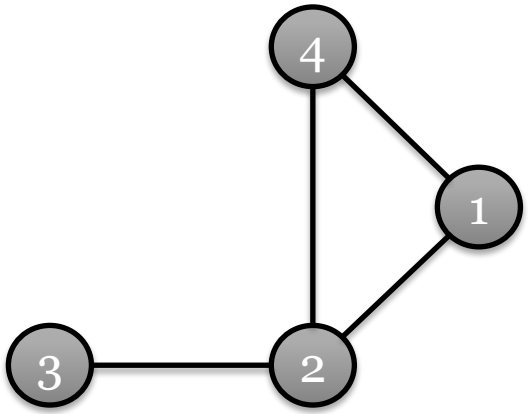
- A **null graph** is one where the node set is empty (there are no nodes)
  - Since there are no nodes, there are also no edges

$$G(V, E),\ V = E = \varnothing$$

- An **empty graph** or **edge-less graph** is one where the edge set is empty, $E = \varnothing$
  - The node set can be non-empty.
  - A null-graph is an empty graph.

Directed/Undirected/Mixed Graphs

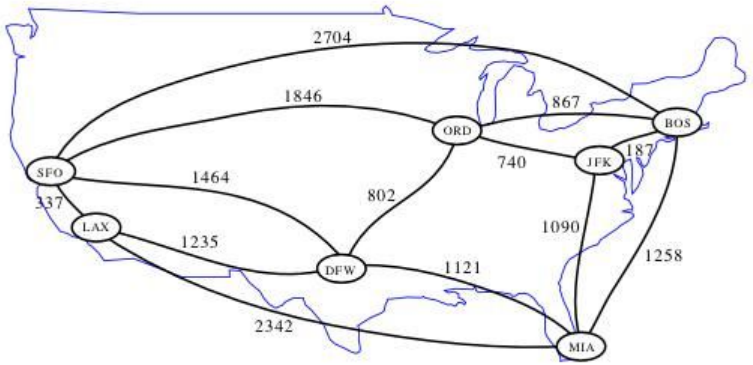




- The adjacency matrix for directed graphs is not symmetric ($A \neq A^T$)
  - ($A_{ij} \neq A_{ji}$) is not true for all pairs i,j
- The adjacency matrix for undirected graphs is symmetric ($A = A^T$)

Weighted Graph

- A weighted graph is one where edges are associated with **weights**
  - For example, a graph could represent a map where nodes are cities and edges are routes between them
  - The weight associated with each edge could represent the distance between these cities



G(V, E, W)

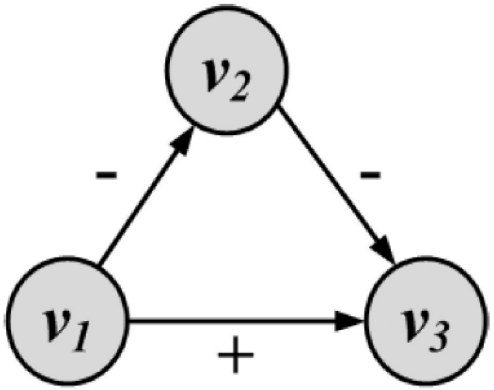
$$A_{ij} = \begin{cases} w, w \in R \\ 0, \text{There is no edge between i and j} \end{cases}$$

Webgraph

- A webgraph is a way of representing how internet sites are connected on the web
- In general, a webgraph is a *directed multigraph*
  - Nodes represent sites and edges represent links between sites
  - Two sites can have multiple links pointing to each other and can have loops (i.e., links pointing to themselves)

Signed Graph

- When weights are binary (0/1, -1/1, +/-) we have a signed graph



- It is used to represent friends or foes
- It is also used to represent social status

Connectivity in Graphs

- **Adjacent nodes/Edges, Walk/Path/Trail/Tour/Cycle**

Adjacent nodes and Incident Edges

Two *nodes* are adjacent if they are connected via an edge

Two *edges* are incident, if they share one end-node

An edge in a graph can be **traversed** when one starts at one of its end-nodes, moves along the edge, and stops at its other end-node

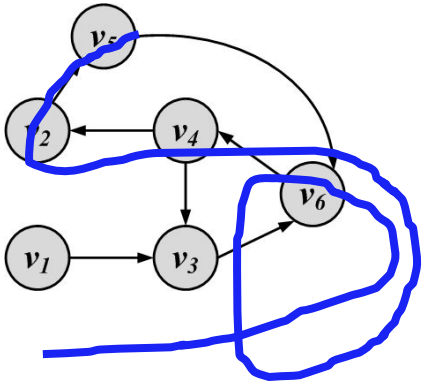
Walk, Trail, Tour, Path, and Cycle

Walk: A walk is a sequence of *incident* edges visited one after another

- **Open walk**: A walk does not end where it starts
- **Close walk**: A walk returns to where it starts

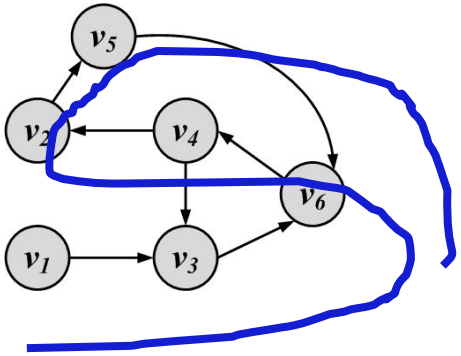
- Representing a walk:
  - A sequence of edges: $e_1, e_2, \ldots, e_n$
  - A sequence of nodes: $v_1, v_2, \ldots, v_n$
- *Length* of a walk: the number of visited edges

Length of walk= 8



Trail and Tour

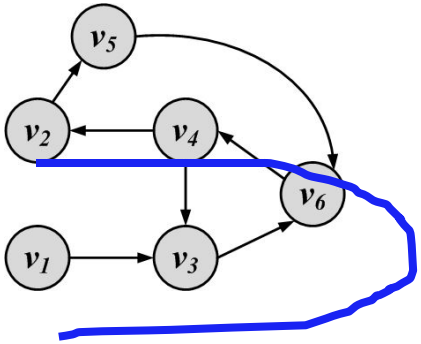
- A *trail* is a walk where **no edge is visited more than once** and all walk edges are distinct
- A closed trail (one that ends where it starts) is called a **tour** or **circuit**



Path

- A walk where **nodes and edges are distinct** is called a **path** and a closed path is called a **cycle**
- The length of a *path* or cycle is the number of edges visited in the path or cycle

Length of path= 4



Random walk

- A walk that the next node in each step is selected **randomly** among the neighbors
  - The **weight** of an edge can be used to define the probability of visiting it
  - For all edges that start at v_i the following equation holds

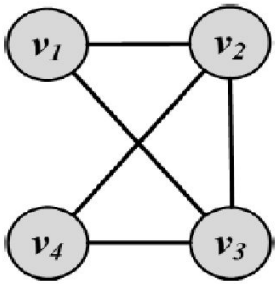
$$\sum_x w_{i,x} = 1, \forall i, j \;\; w_{i,j} \geq 0$$

Connectivity

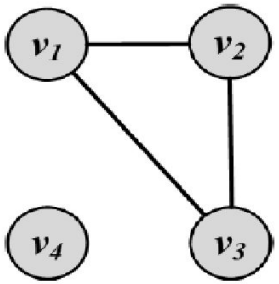
- **A node v_i is connected** to another node v_j (or reachable from v_j) if it is adjacent to it or there exists a path from v_i to v_j.

- **A graph is connected** if there exists a path between any pair of nodes in it
  - In a directed graph, **a graph is strongly connected** if there exists a directed path between any pair of nodes
  - In a directed graph, **a graph is weakly connected** if there exists a path between any pair of nodes, without following the edge directions
- A graph is **disconnected** if it is not connected.

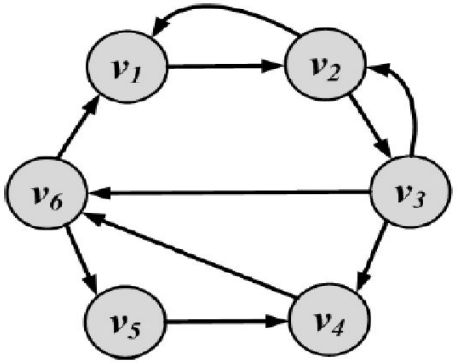
Connectivity: Example



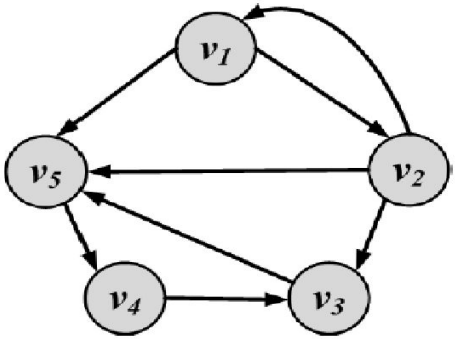
Connected



Disconnected

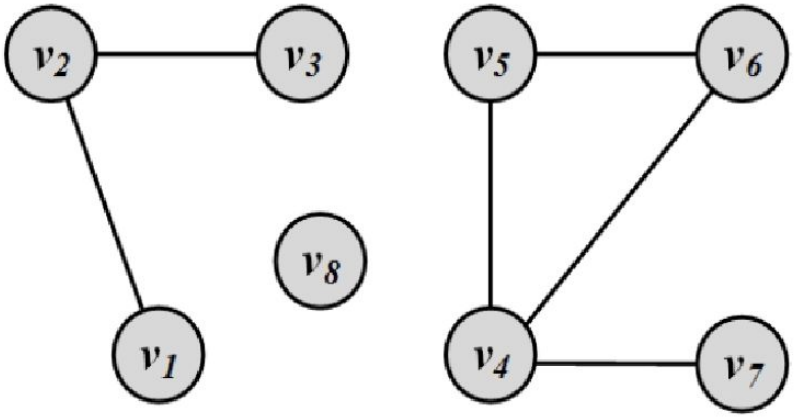


Strongly connected

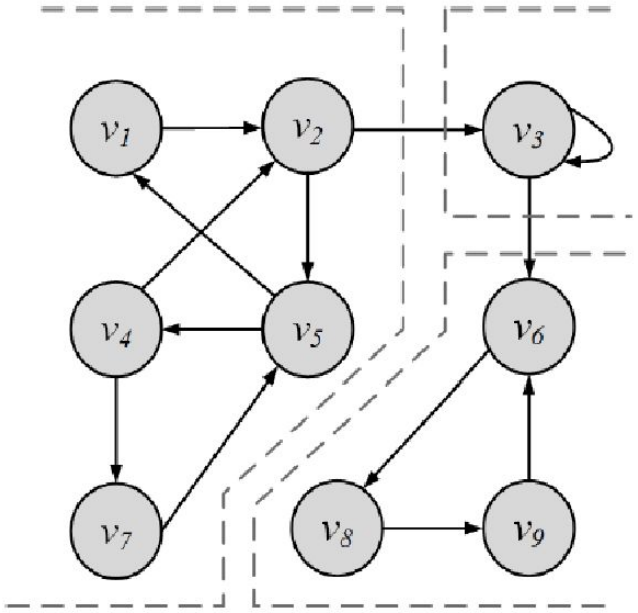


Weakly connected

Component Examples



3 components



3 Strongly-connected components

Shortest Path

- **Shortest Path** is the path between **two** nodes that has the shortest length.
  - We denote the length of the shortest path between nodes v_i and v_j as $l_{i,j}$

- The concept of the neighborhood of a node can be generalized using shortest paths. An **n-hop neighborhood** of a node is the set of nodes that are within n hops distance from the node.

Diameter

- The diameter of a graph is the length of the ***longest*** shortest path between *any* pairs of nodes in the graph

$$diameter_G = \max_{(v_i, v_j) \in V \times V} l_{i,j}.$$

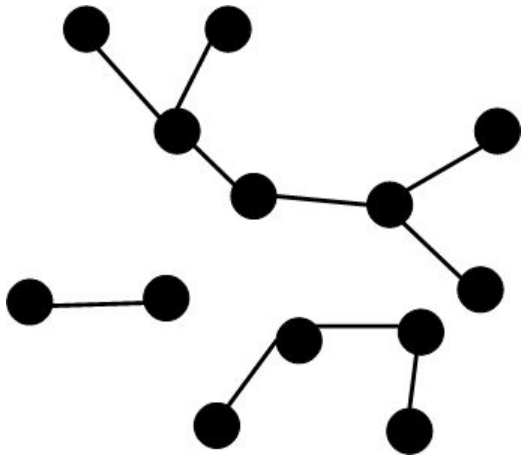
Special Graphs

Trees and Forests

- **Trees** are special cases of undirected graphs
- A tree is a graph structure that has no cycle in it
- In a tree, there is *exactly one path* between any pair of nodes
- In a tree:

$$|V| = |E| + 1$$

- A set of disconnected trees is called a **forest**

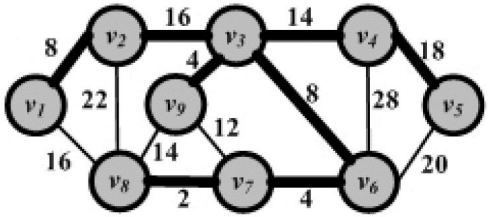


A forest containing 3 trees

Spanning Trees

- For any connected graph, the spanning tree is a subgraph and a tree that includes ***all*** the nodes of the graph
- There may exist *multiple* spanning trees for a graph.
- For a weighted graph and one of its spanning tree, the weight of that spanning tree is the summation of the edge weights in the tree.
- Among the many spanning trees found for a weighted graph, the one with the minimum weight is called the

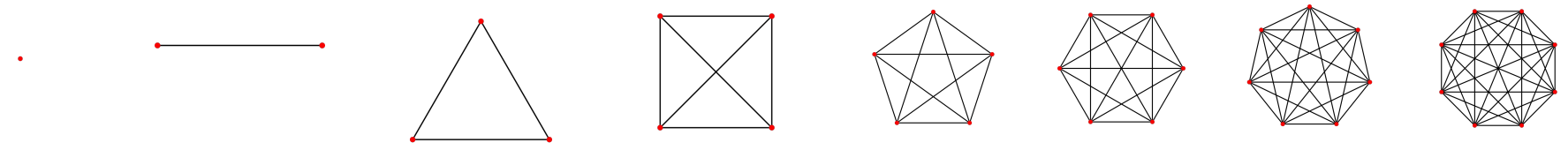
minimum spanning tree (MST)



Complete Graphs

- A complete graph is a graph where for a set of nodes *V*, all possible edges exist in the graph
- In a complete graph, any pair of nodes are connected via an edge

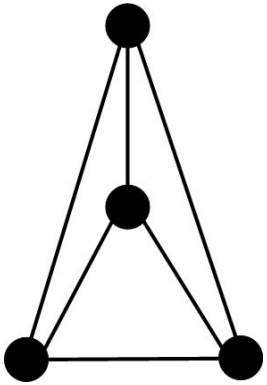
$$E = \binom{|V|}{2}$$



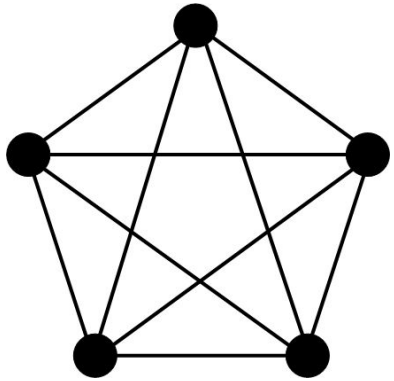
|v|=1 |v|=2 |v|=3 |v|=4 |v|=5 |v|=6 |v|=7 |v|=8

Planar Graphs

- A graph that can be drawn in such a way that **no two edges** cross each other (other than the endpoints) is called planar



Planar
Graph

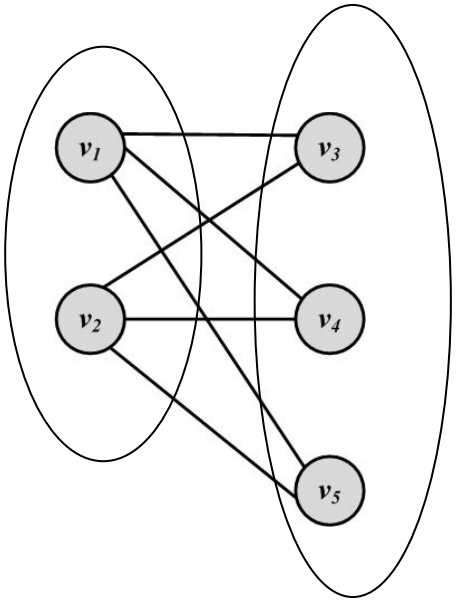


Non-planar
Graph

Bipartite Graphs

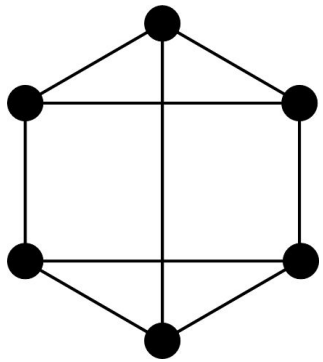
- A bipartite graph G(V; E) is a graph where the node set can be partitioned into **two sets** such that, for all edges, one end-point is in one set and the other end-point is in the other set.

$$\begin{cases} V = V_L \cup V_R, \\ V_L \cap V_R = \emptyset, \\ E \subset V_L \times V_R \end{cases}$$



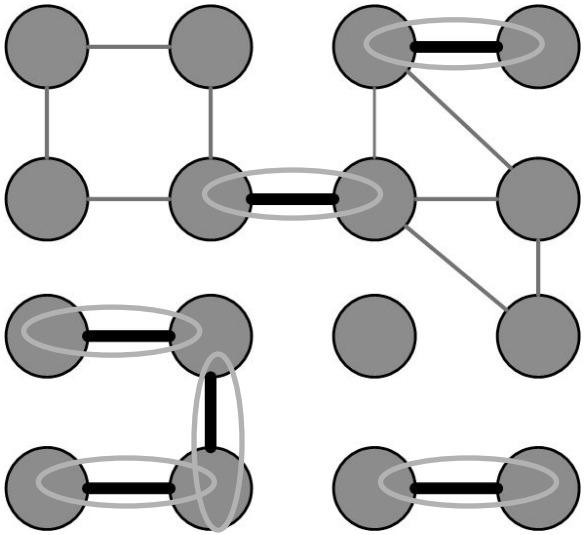
Regular Graphs

- A regular graph is one in which all nodes have the **same** degree
- Regular graphs can be connected or disconnected
- In a ***k***-regular graph, all nodes have degree k
- Complete graphs are examples of regular graphs



Bridges (cut-edges)

- Bridges are edges whose **removal** will **increase** the number of connected components



Graph Algorithms

Graph/Network Traversal Algorithms

Graph/Tree Traversal

- Consider a social media site that has many users and we are interested in surveying the site and computing the average age of its users. The traversal technique should guarantee that
- 1. All users are visited; and
- 2. No user is visited more than once

- There are two main techniques:
  - **Depth-First Search (DFS)**
  - **Breadth-First Search (BFS)**

Depth-First Search (DFS)

- Depth-First Search (DFS) starts from a node v_i, selects one of its neighbors v_j form $N(v_i)$ and performs Depth-First Search on v_j before visiting other neighbors in $N(v_i)$

- The algorithm can be used both for trees and graphs

- The algorithm can be implemented using a stack structure

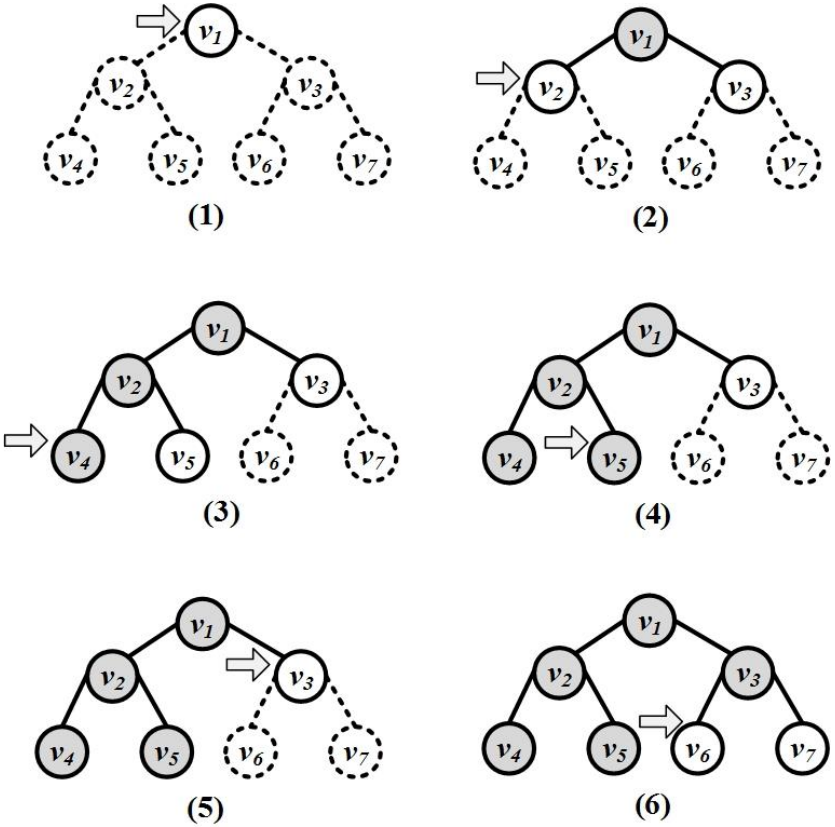
DFS Algorithm

Algorithm 2.2 Depth-First Search (DFS)

Require: Initial node v, graph/tree $G{:}(V, E)$, stack S

```
1: return An ordering on how nodes in G are visited
2: Push v into S;
3: visitOrder = 0;
4: while S not empty do
5:    node = pop from S;
6:    if node not visited then
7:       visitOrder = visitOrder +1;
8:       Mark node as visited with order visitOrder; //or print node
9:       Push all neighbors/children of node into S;
10:   end if
11: end while
12: Return all nodes with their visit order.
```

Depth-First Search (DFS): An Example



Breadth-First Search (BFS)

- BFS starts from a node, visits all its **immediate** neighbors first, and then moves to the second level by traversing their neighbors.
- The algorithm can be used both for trees and graphs
  - The algorithm can be implemented using a queue structure

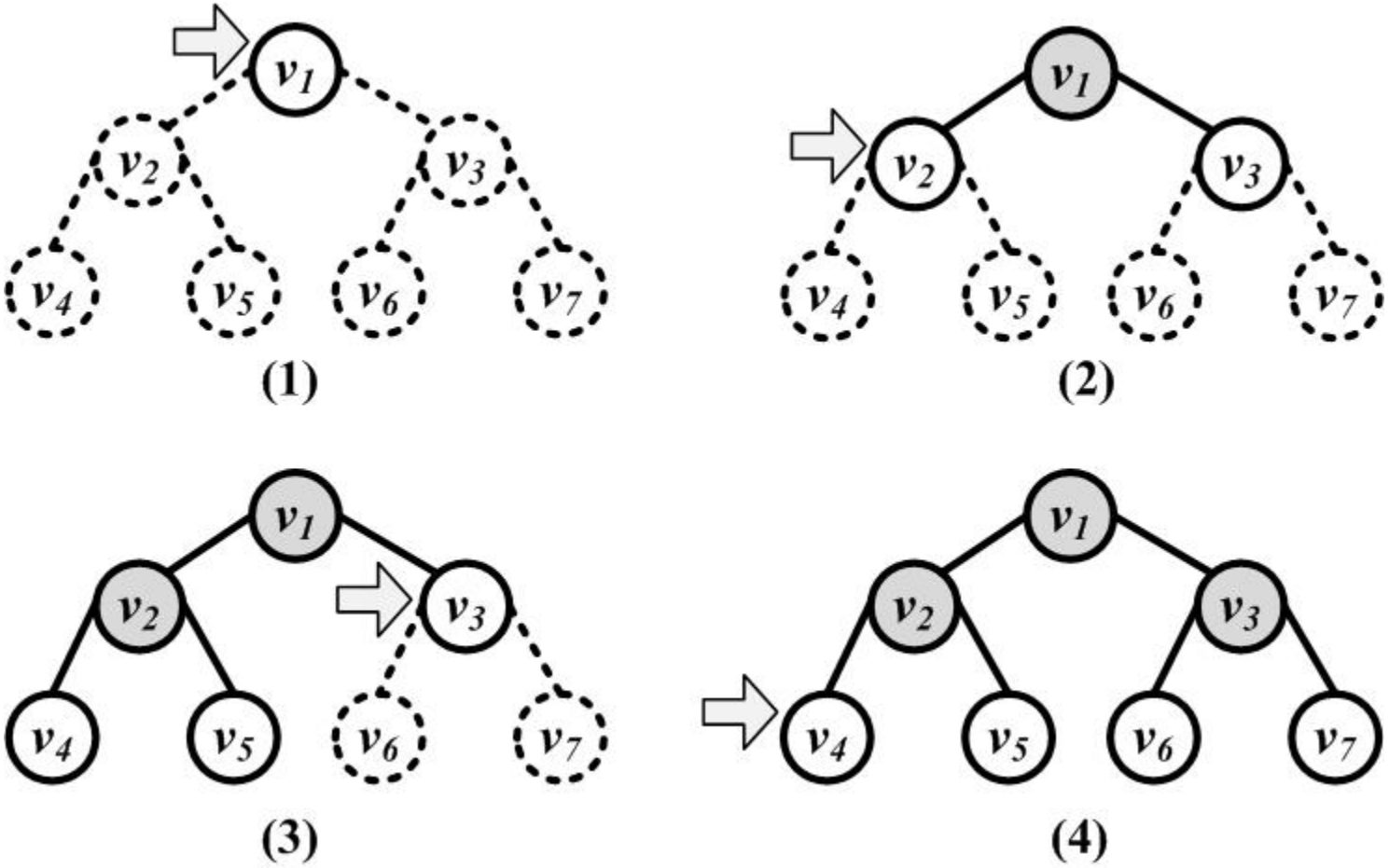
BFS Algorithm

Algorithm 2.3 Breadth-First Search (BFS)

Require: Initial node v, graph/tree $G(V, E)$, queue Q

```
1: return An ordering on how nodes are visited
2: Enqueue v into queue Q;
3: visitOrder = 0;
4: while Q not empty do
5:    node = dequeue from Q;
6:    if node not visited then
7:       visitOrder = visitOrder +1;
8:       Mark node as visited with order visitOrder; //or print node
9:       Enqueue all neighbors/children of node into Q;
10:   end if
11: end while
```

Breadth-First Search (BFS)



Shortest Path

When a graph is connected, there is a chance that multiple paths exist between any pair of nodes

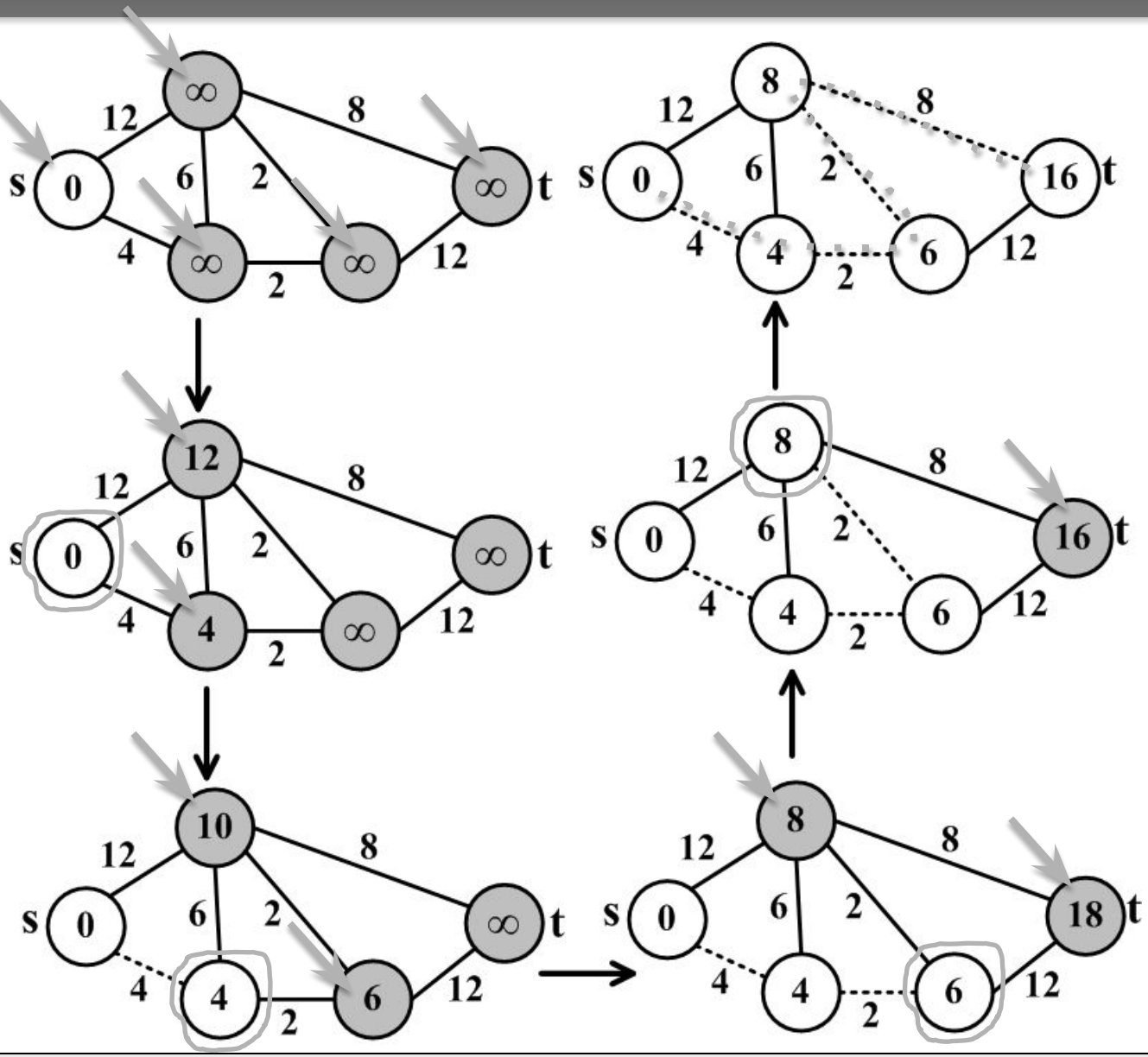
- In many scenarios, we want the shortest path between two nodes in a graph

- **Dijkstra's Algorithm**
  - It is designed for weighted graphs with non-negative edges
  - It finds shortest paths that start from a provided node *s* to all other nodes
  - It finds both shortest paths and their respective lengths

Dijkstra's Algorithm: Finding the shortest path

1. Initiation:
   - Assign zero to the source node and infinity to all other nodes
   - Mark all nodes unvisited
   - Set the source node as current
2. For the current node, consider all of its unvisited neighbors and calculate their *tentative* distances
   - If tentative distance (current node's distance + edge weight) is smaller than neighbor's distance, then Neighbor's distance = tentative distance
3. After considering all of the neighbors of the current node, mark the current node as visited and remove it from the *unvisited set*
   - A visited node will never be checked again and its distance recorded now is final and minimal
4. If the destination node has been marked visited or if the smallest tentative distance among the nodes in the *unvisited set* is infinity, then stop
5. Set the unvisited node marked with the smallest tentative distance as the next "current node" and go to step 2

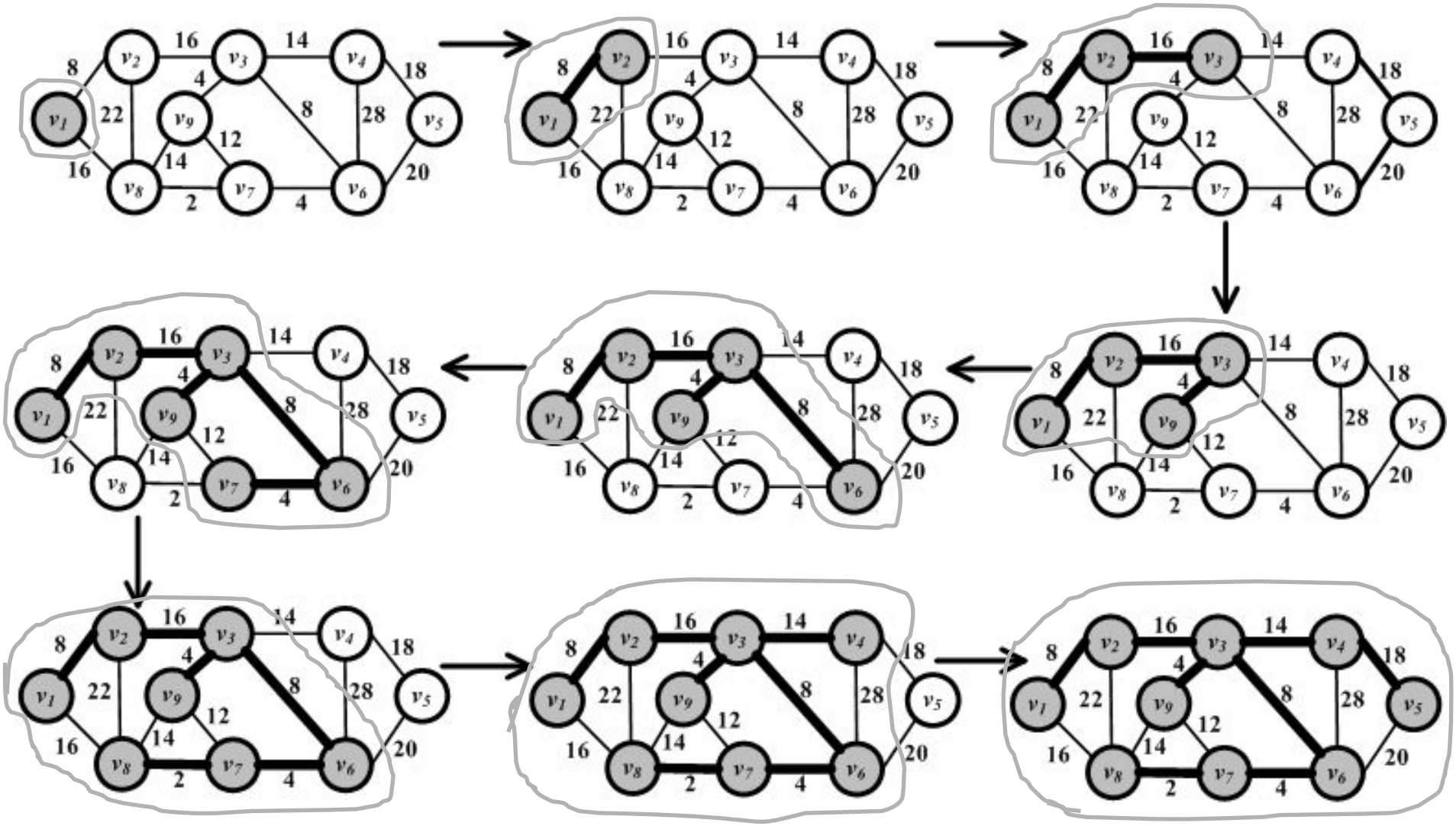
Dijkstra's Algorithm Execution Example



Prim's Algorithm: Finding Minimum Spanning Tree

- Find minimal spanning trees in a weighted graph
  - Start by selecting a random node and adding it to the spanning tree
  - Grow the spanning tree by selecting edges which have one endpoint in the existing spanning tree and one endpoint among the nodes that are not selected yet. Among the possible edges, the one with the **minimum** weight is added to the set (along with its end-point)
  - Iterate until the graph is fully spanned

Prim's Algorithm Execution Example



CS 579: Online Social Network Analysis

Network Measures

Kai Shu

Reading: Chapter 3

Why Do We Need Measures?

- Who are the central figures (influential individuals) in the network?
  - **Centrality**
- What interaction patterns are common in friends?
  - **Reciprocity and Transitivity**
  - **Balance and Status**
- Who are the like-minded users and how can we find these similar individuals?
  - **Similarity**
- To answer these and similar questions, one first needs to define measures for quantifying **centrality**, **level of interactions**, and **similarity**, among others.

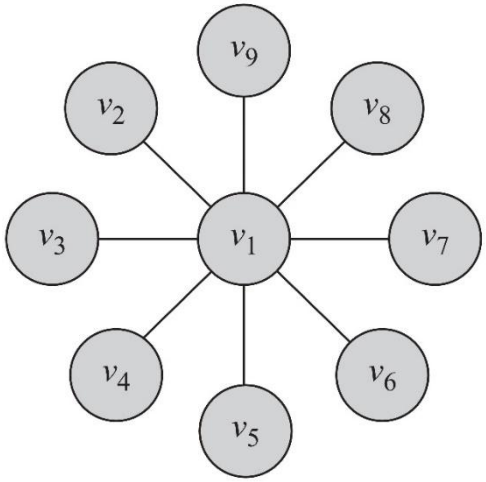
Degree Centrality

- **Degree centrality**: ranks nodes with more connections higher in terms of centrality

$$C_d(v_i) = d_i$$

- d_i is the degree (number of friends) for node v_i
  - i.e., the number of length-1 paths (can be generalized)

In this graph, degree centrality for node v_1 is d_1=8 and for all others is $d_j = 1, j \neq 1$



Degree Centrality in Directed Graphs

- In directed graphs, we can either use the in-degree, the out-degree, or the combination as the degree centrality value
- In practice, mostly in-degree is used.

$$C_d(v_i) = d_i^{\text{in}} \quad (prestige)$$

$$C_d(v_i) = d_i^{\text{out}} \quad (gregariousness)$$

$$C_d(v_i) = d_i^{\text{in}} + d_i^{\text{out}}$$

d_i^{out} is the number of outgoing links for node v_i

Normalized Degree Centrality

- Normalized by the maximum possible degree

$$C_d^{\text{norm}}(v_i) = \frac{d_i}{n-1}$$

- Normalized by the maximum degree

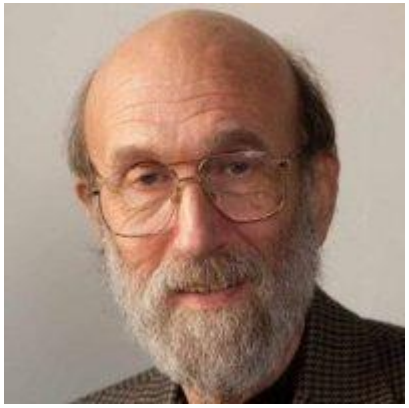
$$C_d^{\max}(v_i) = \frac{d_i}{\max_j d_j}$$

- Normalized by the degree sum

$$C_d^{\text{sum}}(v_i) = \frac{d_i}{\sum_j d_j} = \frac{d_i}{2|E|} = \frac{d_i}{2m}$$

Eigenvector Centrality

- Having more friends does not by itself guarantee that someone is more important
  - Having more **important friends** provides a stronger signal



Phillip Bonacich

- Eigenvector centrality generalizes degree centrality by incorporating the **importance of the neighbors** (undirected)

- For directed graphs, we can use incoming or outgoing edges

Formulation

- Let's assume the eigenvector centrality of a node is $c_e(v_i)$ (unknown)

- We would like $c_e(v_i)$ to be higher when important neighbors (node v_j with higher $c_e(v_j)$) point to us
  - Incoming or outgoing neighbors?
  - For incoming neighbors $A_{j,i} = 1$

- We can assume that v_i's centrality is the summation of its neighbors' centralities

$$c_e(v_i) = \sum_{j=1}^{n} A_{j,i} c_e(v_j)$$

- Is this summation bounded?
  - We have to normalize!
    λ: some fixed constant

$$c_e(v_i) = \frac{1}{\lambda} \sum_{j=1}^{n} A_{j,i} c_e(v_j)$$

Eigenvector Centrality (Matrix Formulation)

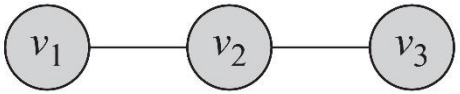
- Let $\mathbf{C}_e = (C_e(v_1), C_e(v_2), \ldots, C_e(v_n))^T$

$$\rightarrow \quad \lambda \mathbf{C}_e = A^T \mathbf{C}_e$$

- This means that $\boldsymbol{C_e}$ is an eigenvector of adjacency matrix A^T (or A when undirected) and λ is the corresponding eigenvalue

- Which eigenvalue-eigenvector pair to choose?
  - We prefer centrality values to be positive for convenient comparison

Eigenvector Centrality: Example 1



$$\lambda \mathbf{C}_e = A\mathbf{C}_e \qquad (A - \lambda I)\mathbf{C}_e = 0 \qquad \mathbf{C}_e = [u_1 \; u_2 \; u_3]^T$$

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 0-\lambda & 1 & 0 \\ 1 & 0-\lambda & 1 \\ 0 & 1 & 0-\lambda \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$det(A - \lambda I) = \begin{vmatrix} 0-\lambda & 1 & 0 \\ 1 & 0-\lambda & 1 \\ 0 & 1 & 0-\lambda \end{vmatrix} = 0$$

$$(-\lambda)(\lambda^2 - 1) - 1(-\lambda) = 2\lambda - \lambda^3 = \lambda(2 - \lambda^2) = 0$$

Eigenvalues are

$$(-\sqrt{2}, 0, +\sqrt{2})$$

Largest Eigenvalue

Corresponding eigenvector (assuming $\mathbf{C_e}$ has norm 1)

$$\begin{bmatrix} 0-\sqrt{2} & 1 & 0 \\ 1 & 0-\sqrt{2} & 1 \\ 0 & 1 & 0-\sqrt{2} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\mathbf{C}_e = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} 1/2 \\ \sqrt{2}/2 \\ 1/2 \end{bmatrix}$$

Katz Centrality

- A major problem with eigenvector centrality arises when it deals with directed graphs
- Centrality only passes over *outgoing* edges and in special cases such as when a node is in a directed acyclic graph centrality becomes zero
  - The node can have many edge connected to it

Elihu Katz

- To resolve this problem we add bias term β to the centrality values for all nodes

Eigenvector Centrality

$$C_{\text{Katz}}(v_i) = \alpha \sum_{j=1}^{n} A_{j,i} C_{\text{Katz}}(v_j) + \beta$$

Katz Centrality, cont.

$$C_{\text{Katz}}(v_i) = \alpha \sum_{j=1}^{n} A_{j,i} C_{\text{Katz}}(v_j) + \beta$$

Controlling term

Bias term

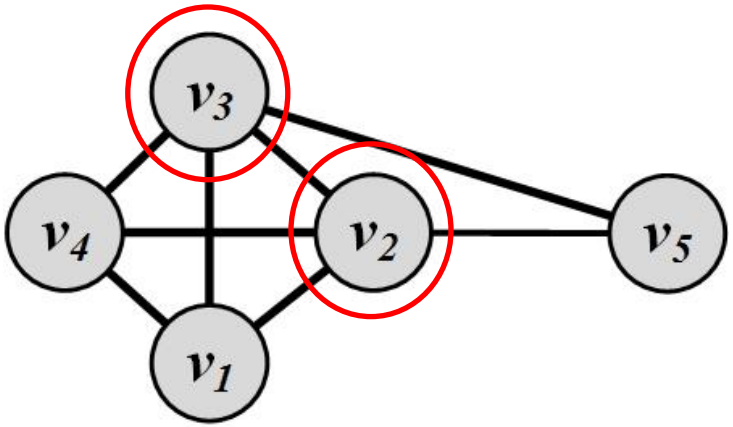
Rewriting equation in a vector form

$$\mathbf{C}_{\text{Katz}} = \alpha A^T \mathbf{C}_{\text{Katz}} + \beta \mathbf{1}$$

vector of all 1's

Katz centrality: $\mathbf{C}_{\text{Katz}} = \beta(\mathbf{I} - \alpha A^T)^{-1} \cdot \mathbf{1}$

Katz Centrality Example



$$A = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{bmatrix} = A^T$$

- The Eigenvalues are -1.68, -1.0, -1.0, 0.35, 3.32
- We assume α=0.25 < 1/3.32 and $\beta = 0.2$

$$\mathbf{C}_{Katz} = \beta(\mathbf{I} - \alpha A^T)^{-1} \cdot \mathbf{1} = \begin{bmatrix} 1.14 \\ 1.31 \\ 1.31 \\ 1.14 \\ 0.85 \end{bmatrix}$$

Most important nodes!

PageRank

- Problem with Katz Centrality:
  - In directed graphs, once a node becomes an authority (high centrality), it passes **all** its centrality along **all** of its out-links
- This is less desirable since not everyone known by a well-known person is well-known
- **Solution?**
  - We can divide the value of passed centrality by the number of outgoing links, i.e., out-degree of that node
  - Each connected neighbor gets a fraction of the source node's centrality

PageRank, cont.

$$C_p(v_i) = \alpha \sum_{j=1}^{n} A_{j,i} \frac{C_p(v_j)}{d_j^{\text{out}}} + \beta$$

What if the degree is zero?

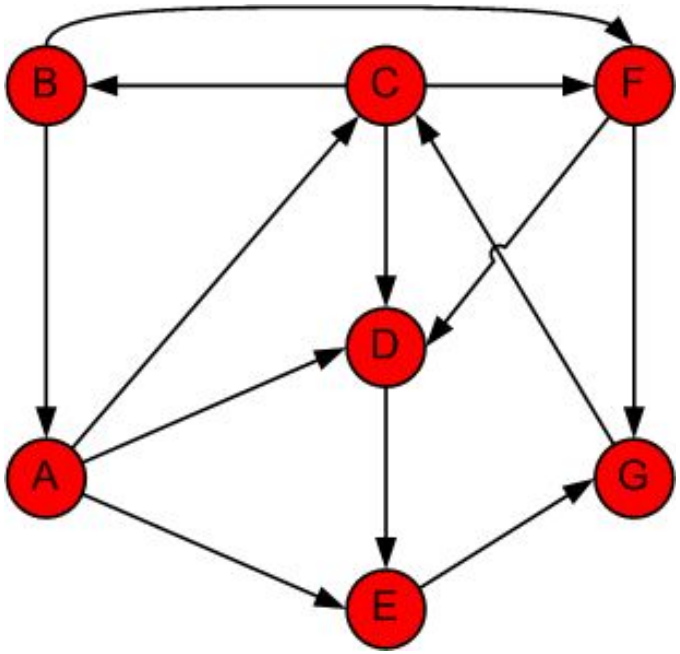
$$\begin{cases} d_j^{out} > 0 \\ D = diag(d_1^{\text{out}}, d_2^{\text{out}}, \ldots, d_n^{\text{out}}) \end{cases}$$

$$\mathbf{C}_p = \alpha A^T D^{-1} \mathbf{C}_p + \beta \mathbf{1}$$

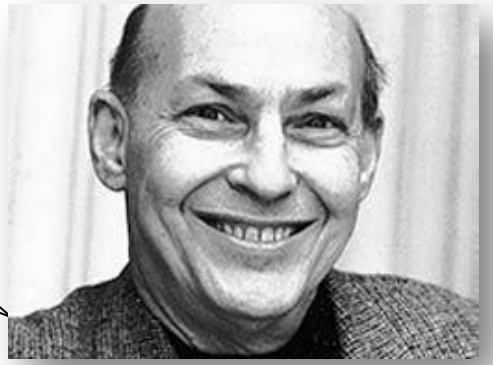
$$\mathbf{C}_p = \beta (\mathbf{I} - \alpha A^T D^{-1})^{-1} \cdot \mathbf{1}$$

Similar to Katz Centrality, in practice, $\boldsymbol{\alpha} < \mathbf{1}/\boldsymbol{\lambda}$, where λ is the largest eigenvalue of $A^T D^{-1}$. In undirected graphs, the largest eigenvalue of $A^T D^{-1}$ is $\boldsymbol{\lambda} = 1$; therefore, $\boldsymbol{\alpha} < \mathbf{1}$.

PageRank Example – Alternative Approach [Markov Chains]



"You don't understand anything until you learn it more than one way"



Marvin Minsky (1927-2016)

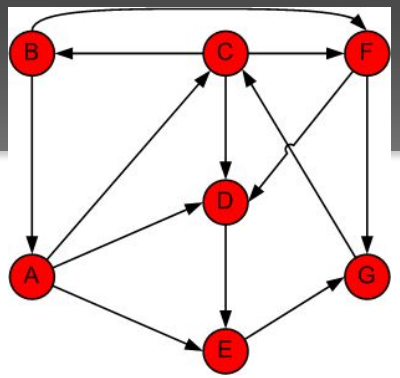
Using Power Method

α=1 and β =0?

$$C_p(v_i) = \alpha \sum_{j=1}^{n} A_{j,i} \frac{C_p(v_j)}{d_j^{\text{out}}} + \beta$$

Step	A	B	C	D	E	F	G
0	1/7	1/7	1/7	1/7	1/7	1/7	1/7
1	B/2	C/3	A/3 + G	A/3 + C/3 + F/2	A/3 + D	C/3 + B/2	F/2 + E
	0.071	0.048	0.190	0.167	0.190	0.119	0.214

PageRank: Example

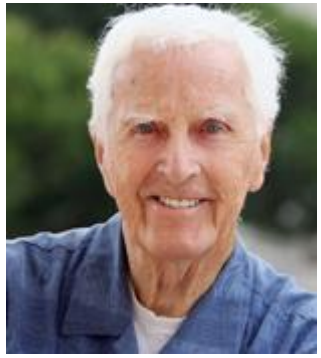


Step	A	B	C	D	E	F	G	Sum
1	0.143	0.143	0.143	0.143	0.143	0.143	0.143	1.000
2	0.071	0.048	0.190	0.167	0.190	0.119	0.214	1.000
3	0.024	0.063	0.238	0.147	0.190	0.087	0.250	1.000
4	0.032	0.079	0.258	0.131	0.155	0.111	0.234	1.000
5	0.040	0.086	0.245	0.152	0.142	0.126	0.210	1.000
6	0.043	0.082	0.224	0.158	0.165	0.125	0.204	1.000
7	0.041	0.075	0.219	0.151	0.172	0.115	0.228	1.000
8	0.037	0.073	0.241	0.144	0.165	0.110	0.230	1.000
9	0.036	0.080	0.242	0.148	0.157	0.117	0.220	1.000
10	0.040	0.081	0.232	0.151	0.160	0.121	0.215	1.000
11	0.040	0.077	0.228	0.151	0.165	0.118	0.220	1.000
12	0.039	0.076	0.234	0.148	0.165	0.115	0.223	1.000
13	0.038	0.078	0.236	0.148	0.161	0.116	0.222	1.000
14	0.039	0.079	0.235	0.149	0.161	0.118	0.219	1.000
15	0.039	0.078	0.232	0.150	0.162	0.118	0.220	1.000
Rank	**7**	**6**	**1**	**4**	**3**	**5**	**2**	

Centrality in terms of how you connect others
(information broker)

Betweenness Centrality

Another way of looking at centrality is by considering how important nodes are in connecting other nodes



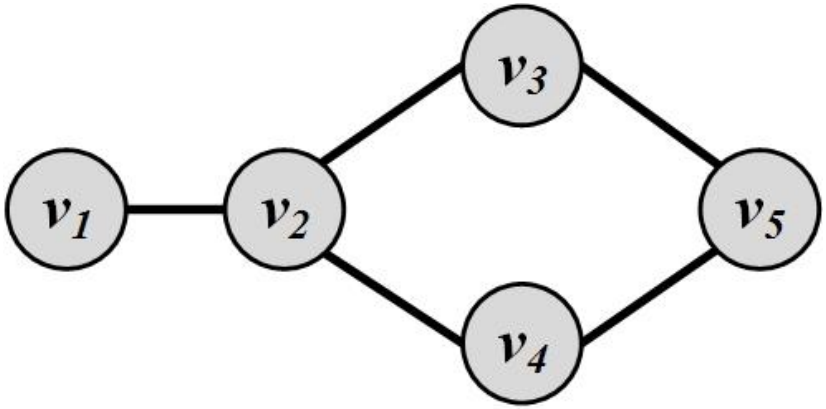
Linton Freeman

$$C_b(v_i) = \sum_{s \neq t \neq v_i} \frac{\sigma_{st}(v_i)}{\sigma_{st}}$$

σ_{st} The number of shortest paths from vertex s to t – *a.k.a.* information pathways

$\sigma_{st}(v_i)$ The number of **shortest paths** from s to t that pass through v_i

Betweenness Centrality: Example 1



$$C_b(v_2) = 2 \times (\underbrace{(1/1)}_{s=v_1,t=v_3} + \underbrace{(1/1)}_{s=v_1,t=v_4} + \underbrace{(2/2)}_{s=v_1,t=v_5} + \underbrace{(1/2)}_{s=v_3,t=v_4} + \underbrace{0}_{s=v_3,t=v_5} + \underbrace{0}_{s=v_4,t=v_5})$$

$$= 2 \times 3.5 = 7,$$

$$C_b(v_3) = 2 \times (\underbrace{0}_{s=v_1,t=v_2} + \underbrace{0}_{s=v_1,t=v_4} + \underbrace{(1/2)}_{s=v_1,t=v_5} + \underbrace{0}_{s=v_2,t=v_4} + \underbrace{(1/2)}_{s=v_2,t=v_5} + \underbrace{0}_{s=v_4,t=v_5})$$

$$= 2 \times 1.0 = 2,$$

$$C_b(v_4) = C_b(v_3) = 2 \times 1.0 = 2,$$

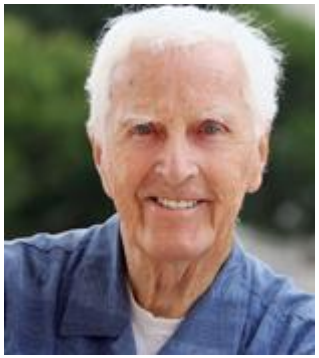
$$C_b(v_5) = 2 \times (\underbrace{0}_{s=v_1,t=v_2} + \underbrace{0}_{s=v_1,t=v_3} + \underbrace{0}_{s=v_1,t=v_4} + \underbrace{0}_{s=v_2,t=v_3} + \underbrace{0}_{s=v_2,t=v_4} + \underbrace{(1/2)}_{s=v_3,t=v_4})$$

$$= 2 \times 0.5 = 1,$$

Centrality in terms of how fast you can reach others

Closeness Centrality

- The intuition is that influential/central nodes can quickly reach other nodes



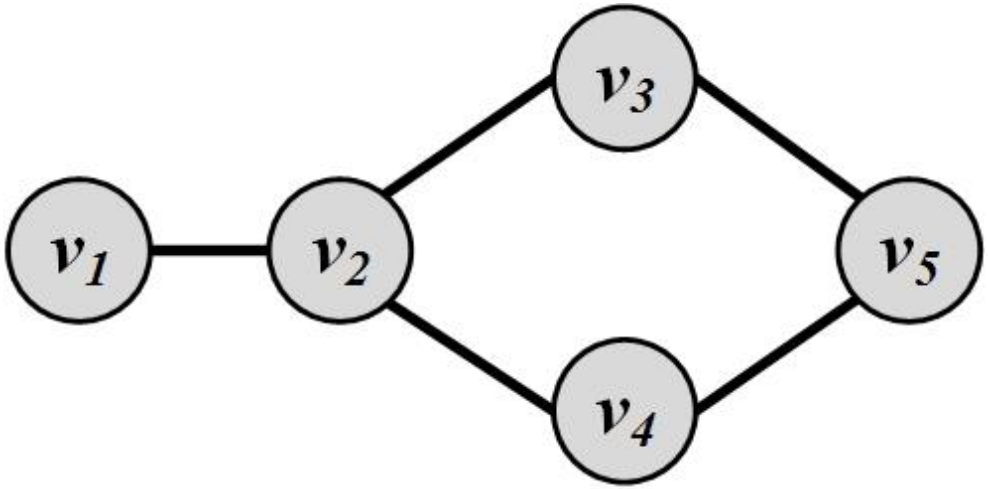
Linton Freeman

- These nodes should have a smaller average shortest path length to others

Closeness centrality: $C_c(v_i) = \frac{1}{\bar{l}_{v_i}}$

$$\bar{l}_{v_i} = \frac{1}{n-1} \sum_{v_j \neq v_i} l_{i,j}$$

Closeness Centrality: Example 1



$$
\begin{aligned}
 & C_c(v_1) = 1 \,/\, (\, (1+2+2+3)/4 \,) = 0.5, \\
 & C_c(v_2) = 1 \,/\, (\, (1+1+1+2)/4 \,) = 0.8, \\
C_c(v_3) = \quad & C_b(v_4) = 1 \,/\, (\, (1+1+2+2)/4 \,) = 0.66, \\
 & C_c(v_5) = 1 \,/\, (\, (1+1+2+3)/4 \,) = 0.57.
\end{aligned}
$$

Centrality for a group of nodes

Group Centrality

- All centrality measures defined so far measure centrality for a single node. These measures can be generalized for a group of nodes.

- A simple approach is to replace all nodes in a group with a super node
  - The group structure is disregarded.

- Let S denote the set of nodes in the group and $V - S$ the set of outsiders

III. Group Closeness Centrality

$$C_c^{\text{group}}(S) = \frac{1}{\bar{l}_S^{\text{group}}}$$

– It is the average distance from non-members to the group

$$\bar{l}_S^{\text{group}} = \frac{1}{|V-S|} \sum_{v_j \notin S} l_{S,v_j}$$

$$l_{S,v_j} = \min_{v_i \in S} l_{v_i,v_j}$$

- One can also utilize the *maximum distance* or the *average distance*

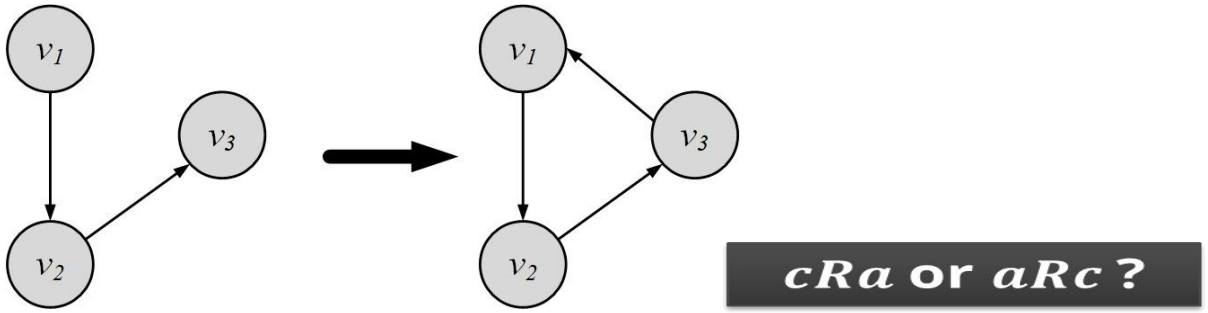
Friendship Patterns

- Transitivity/Reciprocity
- Status/Balance

I. Transitivity and Reciprocity

Transitivity

- Mathematic representation:
  - For a transitive relation R: $aRb \wedge bRc \rightarrow aRc$



- In a social network:
  - ***Transitivity is when a friend of my friend is my friend***
  - Transitivity in a social network leads to a denser graph, which in turn is closer to a complete graph
  - We can determine how close graphs are to the complete graph by measuring transitivity

[Global] Clustering Coefficient

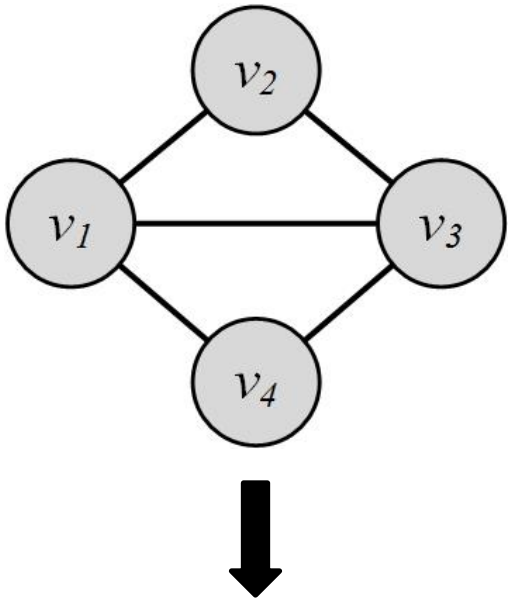
- **Clustering coefficient** measures transitivity in **undirected** graphs
  - Count paths of length two and check whether the third edge exists

$$C = \frac{|\text{Closed Paths of Length 2}|}{|\text{Paths of Length 2}|}$$

When counting triangles, since every triangle has 6 closed paths of length 2

$$C = \frac{(\text{Number of Triangles}) \times 6}{|\text{Paths of Length 2}|}$$

[Global] Clustering Coefficient: Example



$$C = \frac{\text{(Number of Triangles)} \times 3}{\text{Number of Connected Triples of Nodes}}$$

$$= \frac{2 \times 3}{2 \times 3 + \underbrace{2}_{v_2 v_1 v_4, v_2 v_3 v_4}} = 0.75.$$

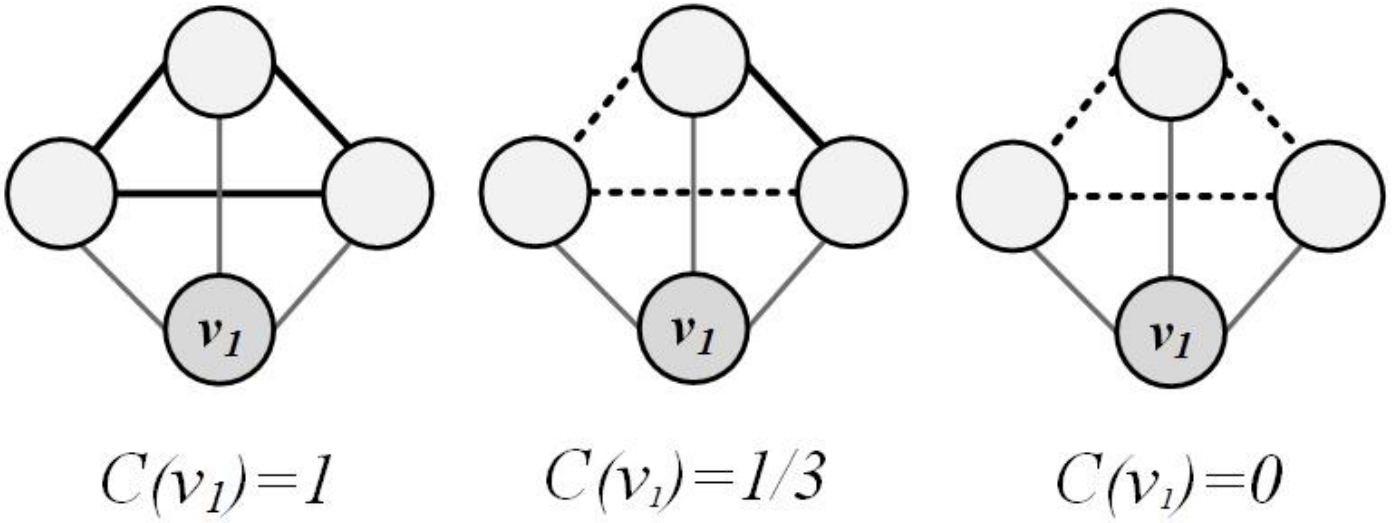
Local Clustering Coefficient

- Local clustering coefficient measures transitivity at the node level
  - Commonly employed for undirected graphs
  - Computes how strongly neighbors of a node v (nodes adjacent to v) are themselves connected

$$C(v_i) = \frac{\text{Number of Pairs of Neighbors of } v_i \text{ That Are Connected}}{\text{Number of Pairs of Neighbors of } v_i}.$$

In an **undirected** graph, the denominator can be rewritten as: $\binom{d_i}{2} = d_i(d_i - 1)/2$

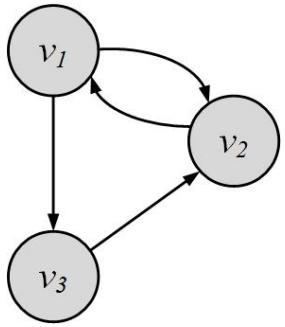
Local Clustering Coefficient: Example



- Thin lines depict connections to neighbors
- Dashed lines are the missing link among neighbors
- Solid lines indicate connected neighbors
  - When none of neighbors are connected $C = 0$
  - When all neighbors are connected $C = 1$

Reciprocity

If you become my friend, I'll be yours

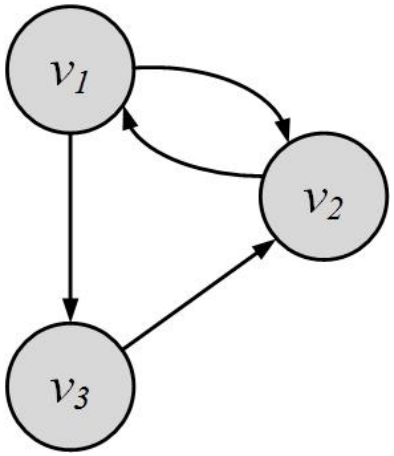


- Reciprocity is simplified version of transitivity
  - It considers closed loops of length 2
- If node v is connected to node u,
  - u by connecting to v, exhibits reciprocity

$$
\begin{aligned}
R &= \frac{\sum_{i,j,i<j} A_{i,j} A_{j,i}}{|E|/2}, \\
&= \frac{2}{|E|} \sum_{i,j,i<j} A_{i,j} A_{j,i}, \\
&= \frac{2}{|E|} \times \frac{1}{2} \mathrm{Tr}(A^2), \\
&= \frac{1}{|E|} \mathrm{Tr}(A^2), \\
&= \frac{1}{m} \mathrm{Tr}(A^2).
\end{aligned}
$$

$$\mathrm{Tr}(A) = A_{1,1} + A_{2,2} + \cdots + A_{n,n} = \sum_{i=1}^{n} A_{i,i}$$

Reciprocity: Example



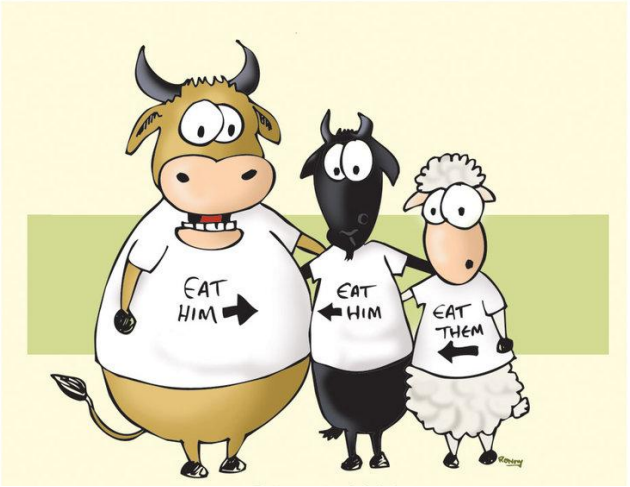
$$A = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

Reciprocal nodes: v_1, v_2

$$R = \frac{1}{m}\mathrm{Tr}(A^2) = \frac{1}{4}\mathrm{Tr}\left(\begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix}\right) = \frac{2}{4} = \frac{1}{2}.$$

II. Balance and Status

- **Measuring consistency in friendships**



Social Balance Theory

Social balance theory

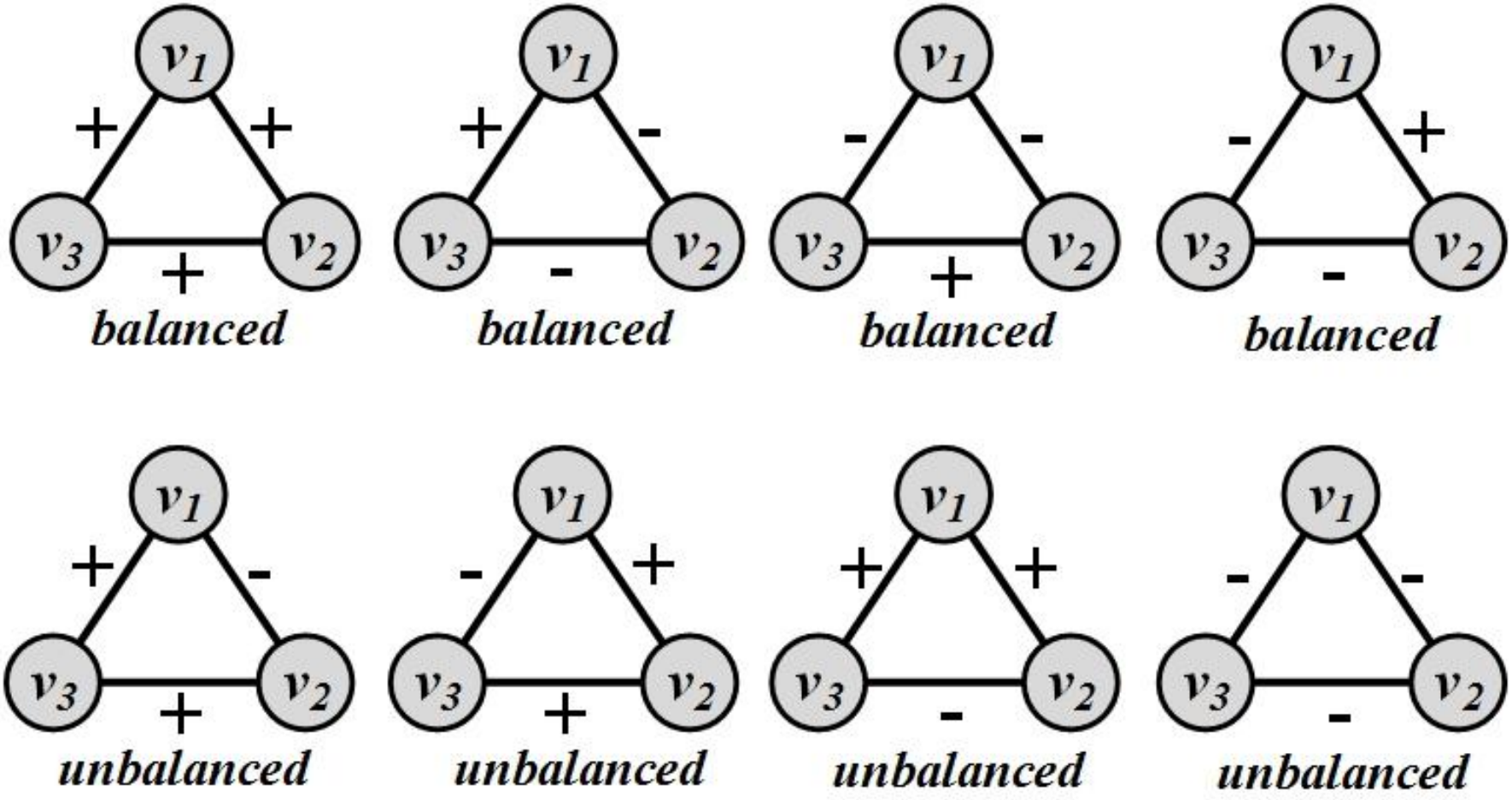
- Consistency in friend/foe relationships among individuals
- Informally, friend/foe relationships are consistent when

$$\begin{aligned}&\textit{The friend of my friend is my friend},\\&\textit{The friend of my enemy is my enemy},\\&\textit{The enemy of my enemy is my friend},\\&\textit{The enemy of my friend is my enemy}.\end{aligned}$$

- In the network
  - Positive edges demonstrate friendships ($w_{ij} = 1$)
  - Negative edges demonstrate being enemies ($w_{ij} = -1$)
- Triangle of nodes i, j, and k, is balanced, if and only if
  - w_{ij} denotes the value of the edge between nodes i and j

$$w_{ij}w_{jk}w_{ki} \geq 0.$$

Social Balance Theory: Possible Combinations



For any cycle, if the multiplication of edge values become positive, then the cycle is socially balanced

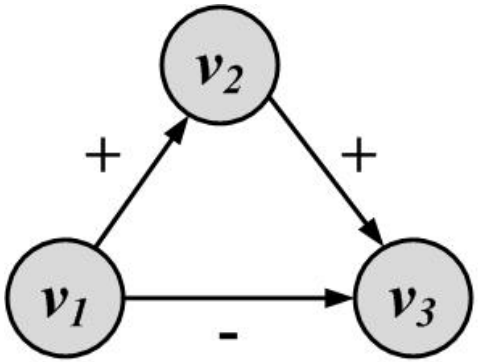
Social Status Theory

- **Status:** how prestigious an individual is ranked within a society

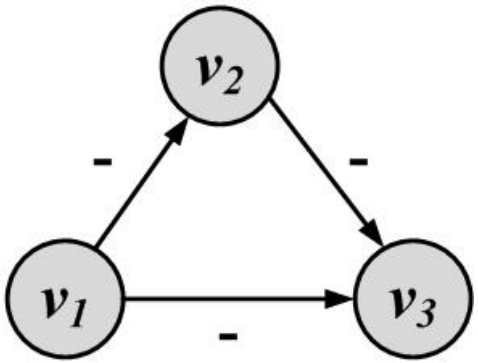
- **Social status theory:**
  - How consistent individuals are in assigning status to their neighbors
  - Informally,

If X has a higher status than Y and Y has a higher status than Z, then X should have a higher status than Z.

Social Status Theory: Example



Unstable configuration



Stable configuration

- A directed **'+'** edge from node X to node Y shows that Y has a higher status than X and a **'-'** one shows vice versa

Similarity

How similar are two nodes in a network?

- **Structural Equivalence**
- **Regular Equivalence**

Structural Equivalence

- **Structural Equivalence:**
  - We look at the neighborhood shared by two nodes;
  - The size of this shared neighborhood defines how similar two nodes are.

- ***Example:***
  - *Two brothers have in common*
    - *sisters, mother, father, grandparents, etc.*
  - *This shows that they are similar,*
  - *Two random male or female individuals do not have much in common and are dissimilar.*

Structural Equivalence: Definitions

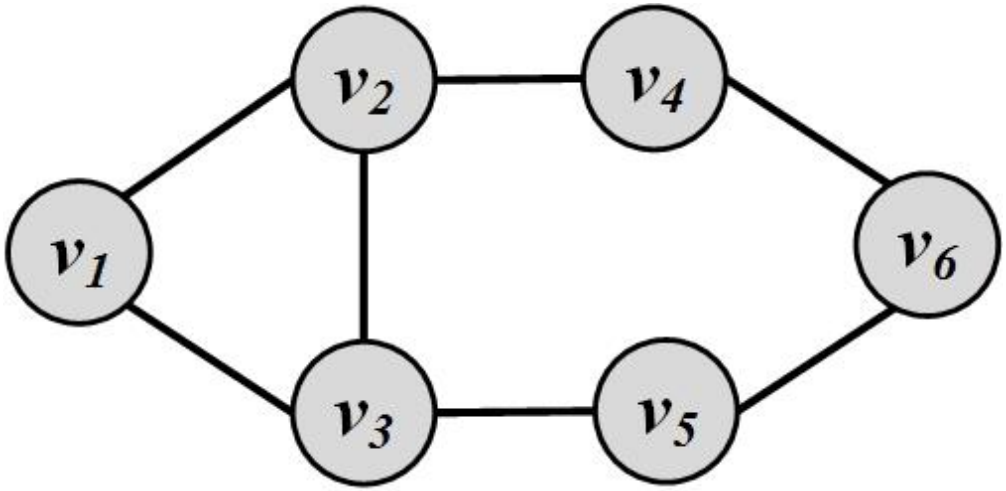
- **Vertex similarity**: $\sigma(v_i, v_j) = |N(v_i) \cap N(v_j)|$

Jaccard Similarity: $\sigma_{Jaccard}(v_i, v_j) = \frac{|N(v_i) \cap N(v_j)|}{|N(v_i) \cup N(v_j)|}$

Cosine Similarity: $\sigma_{Cosine}(v_i, v_j) = \frac{|N(v_i) \cap N(v_j)|}{\sqrt{|N(v_i)||N(v_j)|}}$

- The neighborhood $N(v)$ often excludes the node itself v.
  - **What can go wrong?**
    - Connected nodes not sharing a neighbor will be assigned zero similarity
  - **Solution:**
    - We can assume nodes are included in their neighborhoods

Similarity: Example



$$\sigma_{\text{Jaccard}}(v_2, v_5) = \frac{|\{v_1, v_3, v_4\} \cap \{v_3, v_6\}|}{|\{v_1, v_3, v_4, v_6\}|} = 0.25$$

$$\sigma_{\text{Cosine}}(v_2, v_5) = \frac{|\{v_1, v_3, v_4\} \cap \{v_3, v_6\}|}{\sqrt{|\{v_1, v_3, v_4\}||\{v_3, v_6\}|}} = 0.40.$$

CS 579: Online Social Network Analysis

Chapter 4 Network Models

Kai Shu

Spring 2022

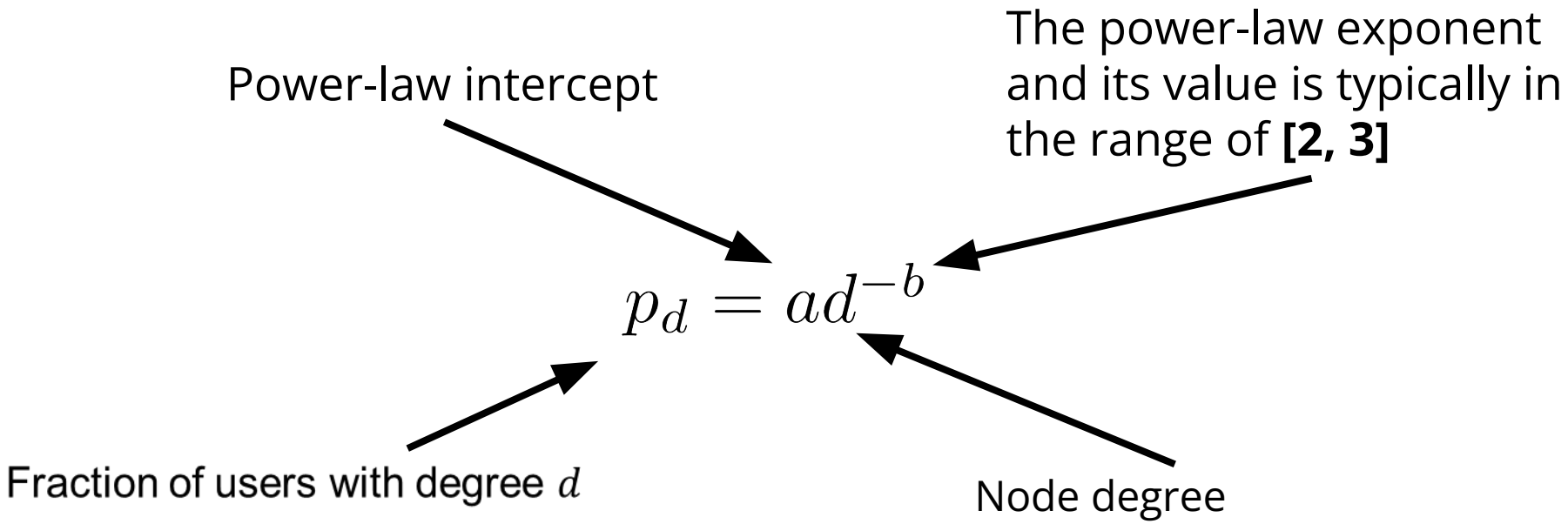
Properties of Real-World Networks

Power-law Distribution
High Clustering Coefficient
Small Average Path Length

Degree Distribution

Power Law Distribution

- When the frequency of an event changes as a power of an attribute
  - The frequency follows a **power-law**



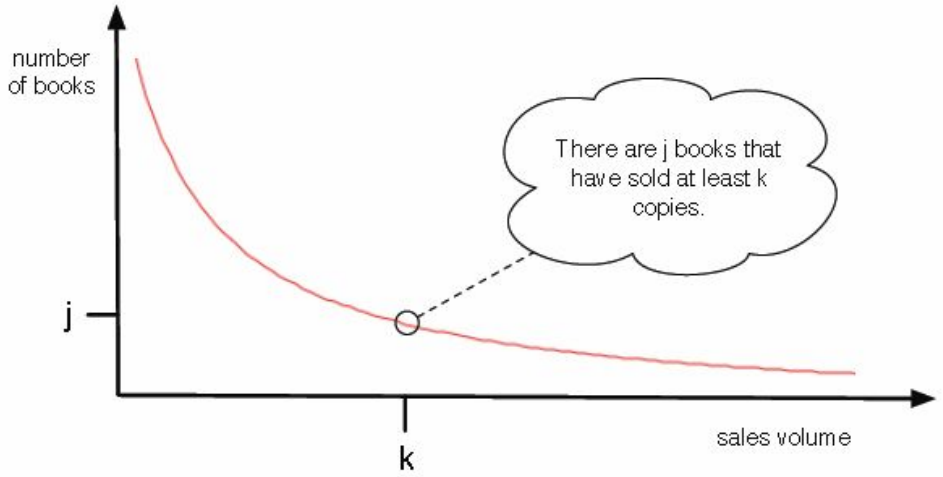
$$\ln p_d = -b \ln d + \ln a$$

The Long Tail

- In a company, are most sales being generated by a small set of items that are enormously popular, or by a much larger population of items that are each individually less popular?

The total sales volume of unpopular items, taken together, can be very significant.

- 57% of Amazon's sales is from the long tail



Clustering Coefficient

Clustering Coefficient

- In real-world networks, friendships are highly transitive, i.e., friends of an individual are often friends with one another
  - These friendships form triads -> **high** average [local] clustering coefficient

- In May 2011, Facebook had an average clustering coefficient of 0.5 for individuals who had 2 friends.

Web	Facebook	Flickr	LiveJournal	Orkut	YouTube
0.081	0.14 (with 100 friends)	0.31	0.33	0.17	0.13

Average Path Length

The Average Shortest Path

- In real-world networks, any two members of the network are usually connected via short paths. In other words, the average path length is **small**
  - Six degrees of separation:
    - **Stanley Milgram** in the well-known small-world experiment conducted in the 1960's conjectured that people around the world are connected to one another via a path of at most 6 individuals
  - Four degrees of separation:
    - **Lars Backstrom et al.** in May 2011, the average path length between individuals in the Facebook graph was 4.7. (4.3 for individuals in the US)

Web	Facebook	Flickr	LiveJournal	Orkut	YouTube
16.12	4.7	5.67	5.88	4.25	5.10

The Average Shortest Path in Sample Networks

	Network	Type	n	m	ℓ
Social	Film actors	Undirected	449 913	25 516 482	3.48
	Company directors	Undirected	7 673	55 392	4.60
	Math coauthorship	Undirected	253 339	496 489	7.57
	Physics coauthorship	Undirected	52 909	245 300	6.19
	Biology coauthorship	Undirected	1 520 251	11 803 064	4.92
	Telephone call graph	Undirected	47 000 000	80 000 000	
	Email messages	Directed	59 812	86 300	4.95
	Email address books	Directed	16 881	57 029	5.22
	Student dating	Undirected	573	477	16.01
	Sexual contacts	Undirected	2 810		
Information	WWW nd.edu	Directed	269 504	1 497 135	11.27
	WWW AltaVista	Directed	203 549 046	1 466 000 000	16.18
	Citation network	Directed	783 339	6 716 198	
	Roget's Thesaurus	Directed	1 022	5 103	4.87
	Word co-occurrence	Undirected	460 902	16 100 000	
Technological	Internet	Undirected	10 697	31 992	3.31
	Power grid	Undirected	4 941	6 594	18.99
	Train routes	Undirected	587	19 603	2.16
	Software packages	Directed	1 439	1 723	2.42
	Software classes	Directed	1 376	2 213	5.40
	Electronic circuits	Undirected	24 097	53 248	11.05
	Peer-to-peer network	Undirected	880	1 296	4.28
Biological	Metabolic network	Undirected	765	3 686	2.56
	Protein interactions	Undirected	2 115	2 240	6.80
	Marine food web	Directed	134	598	2.05
	Freshwater food web	Directed	92	997	1.90
	Neural network	Directed	307	2 359	3.97

l: average path length

Source: M. E. J Newman

Network Models

- Model-Driven Models!

Random graphs
Small-World Model
Preferential Attachment

Random Graphs

Random Graphs

- We start with the most basic assumption on how friendships are formed.

A Random Graph's main assumption:
Edges (i.e., friendships) between nodes (i.e., individuals) are formed randomly.

We discuss two random graph models $G(n,p)$ and $G(n,m)$

Random Graph Model - $G(n,p)$

- Consider a graph with a fixed number of nodes n
- Any of the $\binom{n}{2}$ edges can be formed independently, with probability p
- The graph is called a $G(n,p)$ *random graph*

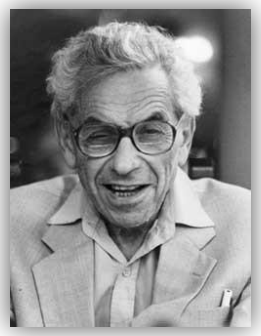
Proposed independently by Edgar Gilbert and by Solomonoff and Rapoport.

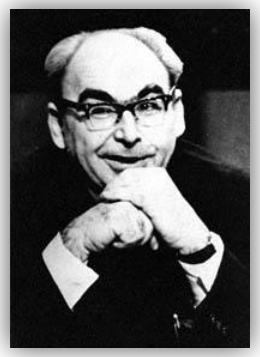
Random Graph Model - $G(n, m)$

- Assume both number of nodes n and number of edges m are fixed.
- Determine which m edges are selected from the set of possible edges
- Let Ω denote the set of graphs with n nodes and m edges
  - There are $|\Omega|$ different graphs with n nodes and m edges

$$|\Omega| = \binom{\binom{n}{2}}{m}$$

- To generate a random graph, we uniformly select one of the $|\Omega|$ graphs (the selection probability is $1/|\Omega|$)





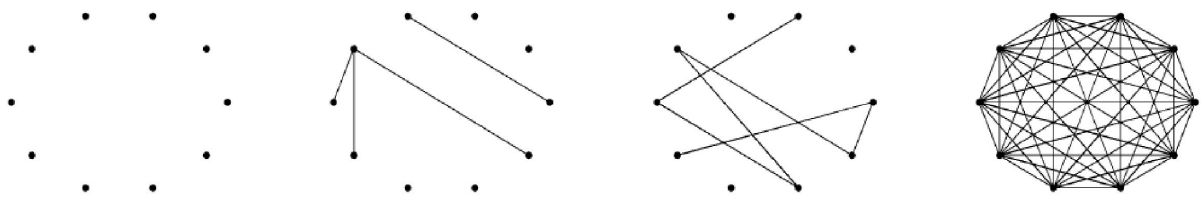
This model was first proposed by *Paul Erdös* and *Alfred Rényi*

The Giant Component

- In random graphs, as we increase p, a large fraction of nodes start getting connected
  - i.e., we have a path between any pair
- This large fraction forms a connected component:
  - **Largest connected component**, also known as the **Giant component**

- In random graphs:
  - $p = 0$
    - the size of the giant component is 0
  - $p = 1$
    - the size of the giant component is n

The Formation of a Giant Component



Probability (p)	0.0	0.055	0.11	1.0
Average node degree (c)	0.0	0.8	~1	n-1 = 9
Diameter	0	2	6	1
Giant component size	0	4	7	10
Average path length	0.0	1.5	2.66	1.0

Properties of Random Graphs

Degree Distribution

- When computing degree distribution, we estimate the probability of observing $P(d_v = d)$ for node v

- For a random graph generated by $G(n, p)$, this probability is

$$P(d_v = d) = \binom{n-1}{d} p^d (1-p)^{n-1-d}$$

- This is a binomial degree distribution. In the limit, this will become the Poisson degree distribution

Global Clustering Coefficient

The global clustering coefficient of a random graph generated by $G(n, p)$ is p

Proof

- The global clustering coefficient defines the probability of two neighbors of the same node being connected.
- In a random graph, for any two nodes, this probability is the same
  - Equal to the generation probability p that determines the probability of two nodes getting connected

- Observation: Local and Global clustering coefficients are the same

Modeling with Random Graphs

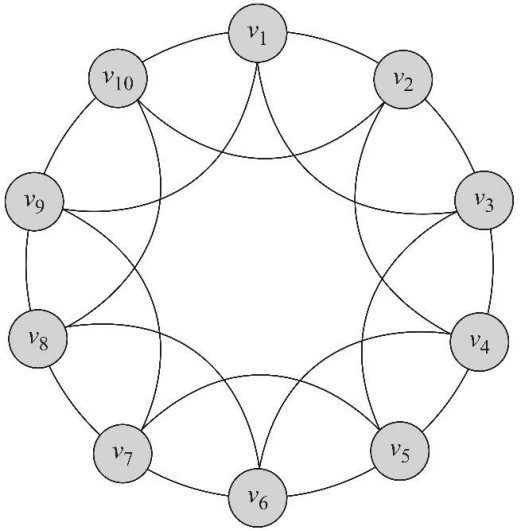
- Compute the average degree c in the real-world graph
- Compute p using $c/(n-1) = p$
- Generate the random graph using p

- How representative is the generated graph?
  - **[Degree Distribution]** Random graphs do not have a power-law degree distribution
  - **[Average Path Length]** Random graphs perform well in modeling the average path lengths
  - **[Clustering Coefficient]** Random graphs drastically underestimate the clustering coefficient

Small-World Model

Small-world Model

- In real-world interactions, many individuals have a limited and often at least, a fixed number of connections
- In graph theory terms, this assumption is equivalent to embedding users in a regular network
- A regular (ring) lattice is a special case of regular networks where there exists a certain pattern on how ordered nodes are connected to one another
- In a regular lattice of degree c, nodes are connected to their previous $c/2$ and following $c/2$ neighbors
- Formally, for node set V=$\{v_1, v_2, v_3, \ldots, v_n\}$, an edge exists between node i and j if and only if



$$0 \leq \min(n - |i - j|, |i - j|) \leq c/2$$

Generating a Small-World Graph

- The lattice has a **high**, but **fixed**, clustering coefficient
- The lattice has a **high** average path length

- In the small-world model, a parameter $0 \leq \beta \leq 1$ controls randomness in the model
  - When β is 0, the model is basically a regular lattice
  - When $\beta = 1$, the model becomes a random graph
- The model starts with a regular lattice and starts adding random edges [through **rewiring**]
  - **Rewiring:** take an edge, change one of its end-points randomly

Regular Lattice vs. Random Graph

- Regular Lattice:
  - Clustering Coefficient (**high**):

$$\frac{3(c-2)}{4(c-1)} \approx \frac{3}{4}$$

  - Average Path Length (**high**): $n/2c$

- Random Graph:
  - Clustering Coefficient (**low**): p
  - Average Path Length (**ok!**) : $\ln |V| / \ln c$

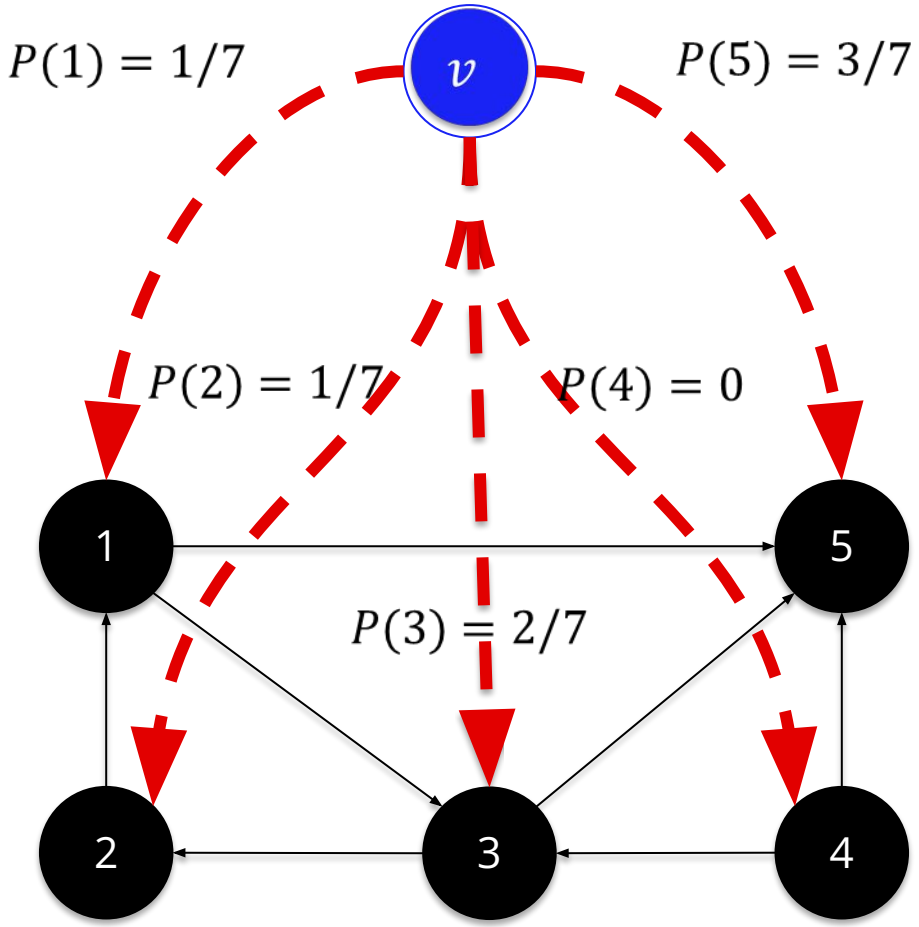
Preferential Attachment Model

Preferential Attachment: Example

- Node v arrives

$$P(v_i) = \frac{d_i}{\sum_j d_j}$$

- $P(1) = 1/7$
- $P(2) = 1/7$
- $P(3) = 2/7$
- $P(4) = 0$
- $P(5) = 3/7$



Modeling Real-World Networks w *Preferential Attachment*

- Similar to random graphs, we can simulate real-world networks by generating a preferential attachment model by setting the expected degree m

Algorithm 4.2 Preferential Attachment

Require: Graph $G(V_0, E_0)$, where $|V_0| = m_0$ and $d_v \geq 1 \; \forall \; v \in V_0$, number of expected connections $m \leq m_0$, time to run the algorithm t

1: **return** A scale-free network
2: `//Initial graph with` m_0 `nodes with degrees at least 1`
3: $G(V, E) = G(V_0, E_0)$;
4: **for** 1 to t **do**
5: $V = V \cup \{v_i\}$; `//` `add new node` v_i
6: **while** $d_i \neq m$ **do**
7: Connect v_i to a random node $v_j \in V, i \neq j$ (i.e., $E = E \cup \{e(v_i, v_j)\}$) with probability $P(v_j) = \frac{d_j}{\sum_k d_k}$.
8: **end while**
9: **end for**
10: Return $G(V, E)$

Properties of the Preferential Attachment Model

Real-World Networks and Simulated Graphs

	Original Network				Simulated Random Graph	
Network	*Size*	*Average Degree*	*Average Path Length*	*C*	*Average Path Length*	*C*
Film Actors	225,226	61	3.65	0.79	2.99	0.00027
Medline Coauthorship	1,520,251	18.1	4.6	0.56	4.91	1.8×10^{-4}
E.Coli	282	7.35	2.9	0.32	3.04	0.026
C.Elegans	282	14	2.65	0.28	2.25	0.05

			al Network		Simulated Graph	
		Degree	*Average Path Length*	*C*	*Average Path Length*	*C*
Film Actors	225,226	61	3.65	0.79	4.2	0.73
Medline Coauthorship	1,520,251	18.1	4.6	0.56	5.1	0.52
E.Coli	282	7.35	2.9	0.32	4.46	0.31
C.Elegans	282	14	2.65	0.28	3.49	0.37

Random Small World Preferential Attachment?

	Original Network				Simulated Graph	
Network	*Size*	*Average Degree*	*Average Path Length*	*C*	*Average Path Length*	*C*
Film Actors	225,226	61	3.65	0.79	4.90	≈ 0.005
Medline Coauthorship	1,520,251	18.1	4.6	0.56	5.36	≈ 0.0002
E.Coli	282	7.35	2.9	0.32	2.37	0.03
C.Elegans	282	14	2.65	0.28	1.99	0.05

CS 579: Online Social Network Analysis

Data Mining Essentials

Kai Shu

Reading: Chapter 5

Data Mining

The process of discovering hidden patterns in large data sets

It utilizes methods at the intersection of artificial intelligence, machine learning, statistics, and database systems

- *Extracting or "mining" knowledge from large amounts of data, or big data*
- Data-driven discovery and modeling of hidden patterns in big data
- Extracting implicit, previously unknown, unexpected, and potentially useful information/knowledge from data

Data

Data Types + Permissible Operations (statistics)

- **Nominal** (categorical)
  - **Operations**: Mode (most common feature value), Frequency, Equality Comparison
  - E.g., {dog, cat, snake, bird}
- **Ordinal**
  - Feature values have an intrinsic order to them, but the difference is not defined
  - **Operations**: same as nominal, feature value rank
  - E.g., {small, medium, large, x-large}
- **Interval**
  - **Operations:** Addition and subtractions are allowed whereas divisions and multiplications are not
  - E.g., year, temperature (F/C)
- **Ratio**
  - **Meaningful zero point**
  - **Operations:** divisions and multiplications are allowed
  - E.g., Height, weight, money quantities

Data Mining Algorithms

- **Supervised Learning Algorithm**

Class attribute is available

  - **Classification (class attribute is discrete)**
    - Assign data into predefined classes
      - Spam Detection, fraudulent credit card detection
  - **Regression (class attribute takes real values)**
    - Predict a real value for a given data instance
      - Predict the price for a given house

- **Unsupervised Learning Algorithm**

Class attribute is **not** available

  - **Clustering**
    - Group similar items together into some clusters
      - Detect communities in a given social network

Supervised Learning

Supervised Learning Algorithms

- Classification
  - *k*-Nearest Neighbor Classifier
  - Decision tree learning
  - Naive Bayes Classifier
  - Classification with Network information
- Regression
  - Linear Regression
  - Logistic Regression

k-Nearest Neighbor Classifier

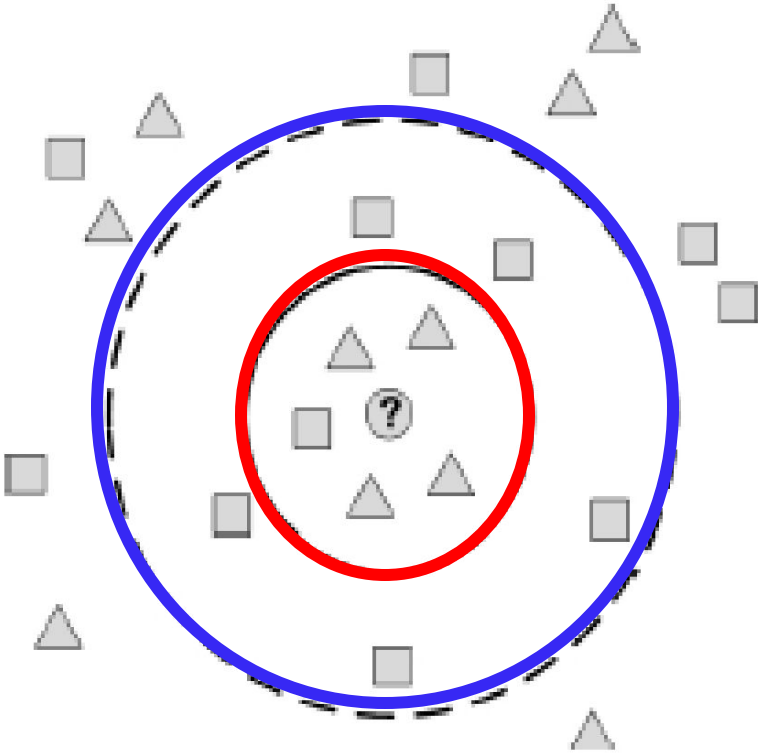
- *k*-Nearest Neighbors or kNN, as the name suggests, utilizes the neighbors of an instance to perform classification.

- To determine the neighbors of an instance, we need to measure its distance to all other instances based on some distance metric. Commonly, Euclidean distance is employed

- The instance being classified is assigned the label (class attribute value) that the majority of its *k* neighbors are assigned

- When $k = 1$, the closest neighbor's label is used as the predicted label for the instance being classified

k-NN example



- When k=5, the predicted label is: triangle
- When k=9, the predicted label is: square

Supervised Learning Algorithms

- Classification
  - *k*-Nearest Neighbor Classifier
  - Decision tree learning
  - Naive Bayes Classifier
  - Classification with Network information
- Regression
  - Linear Regression
  - Logistic Regression

Decision Tree

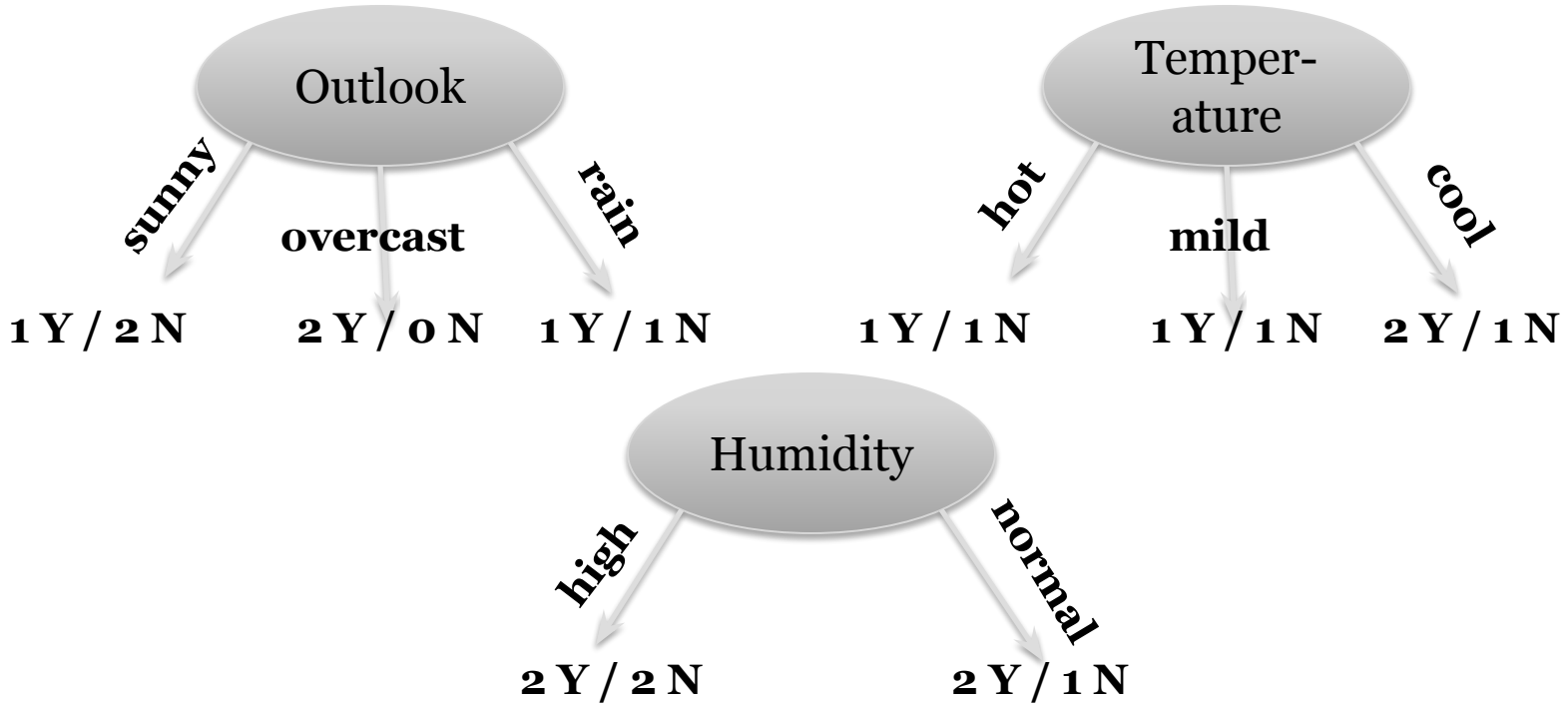
- A decision tree is learned from the dataset (training data with known classes) and later applied to predict the class attribute value of new data (test data with unknown classes) where only the feature values are known

Decision Tree Construction

- Decision trees are constructed recursively from training data using a top-down greedy approach in which features are sequentially selected.
- After selecting a feature for each node, based on its values, different branches are created.
- The training set is then partitioned into subsets based on the feature values, each of which fall under the respective feature value branch; the process is continued for these subsets and other nodes.
- When selecting features, we ***prefer features*** *that partition the set of instances into* ***subsets*** *that are more* ***pure***. A pure subset has instances that all have the same class attribute value.

Decision Tree Construction

No.	Outlook (O)	Temperature (T)	Humidity (H)	Play Golf (PG)
1	sunny	hot	high	N
2	sunny	mild	high	N
3	overcast	hot	high	Y
4	rain	mild	high	Y
5	sunny	cool	normal	Y
6	rain	cool	normal	N
7	overcast	cool	normal	Y



Decision Tree Construction

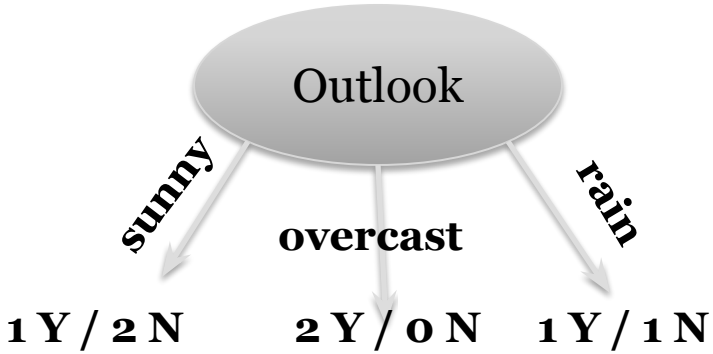
- To measure purity we can use entropy. Over a subset of training instances, T, with a binary class attribute (values in {+,-}), the entropy of T is defined as:

$$entropy(T) = -p_+ \log p_+ - p_- \log p_-,$$

  - p_+ is the proportion of positive examples in D
  - p_- is the proportion of negative examples in D
- We can use this to measure the quality of the leaves along a split.

Entropy is a measure of disorder

Entropy Example



H(sunny) = $-1/3*\log_2(1/3) - 2/3*\log_2(2/3) \sim= 0.92$

H(rain) = $-1/2*\log_2(1/2) - 1/2*\log_2(1/2) = 1$

H(overcast) = $-2/2*\log_2(2/2) - 0/2*\log_2(0/2) = 0$

- We have measured each *value* in the attribute.
- We need to aggregate these to measure the quality of *splitting on this attribute overall*.

Information Gain

- Entropy does not tell the whole story.
- We want many items in pure sets.
- Information Gain can tell us how much our knowledge has improved after the split, i.e. our reduction in entropy

What's the pureness if I do no split the node?

What's the pureness if I split based on attribute A?

$$IG(A, S) = H(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} H(S_v)$$

Attribute

IG(outlook, play golf) = Entropy(play golf) – Entropy(outlook, play golf)

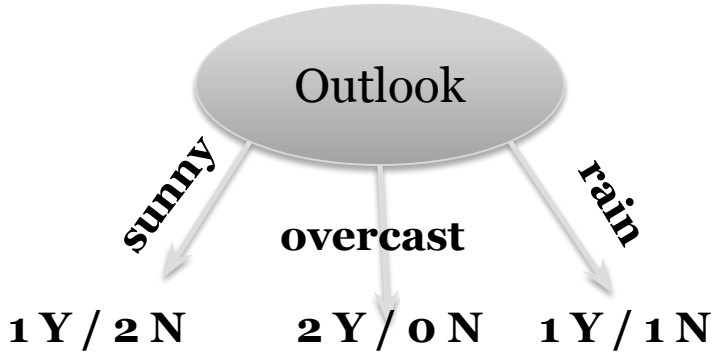
Entropy(outlook, play golf) = P(sunny)*H(outlook = sunny) + P(overcast)*H(outlook = overcast) + P(rain)*H(outlook = rain)

Information Gain

No.	Outlook (O)	Temperature (T)	Humidity (H)	Play Golf (PG)
1	sunny	hot	high	N
2	sunny	mild	high	N
3	overcast	hot	high	Y
4	rain	mild	high	Y
5	sunny	cool	normal	Y
6	rain	cool	normal	N
7	overcast	cool	normal	Y

$$IG(A,S) = H(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} H(S_v)$$

H(S) = $-4/7*\log_2(4/7) - 3/7*\log_2(3/7) \sim= 0.99$

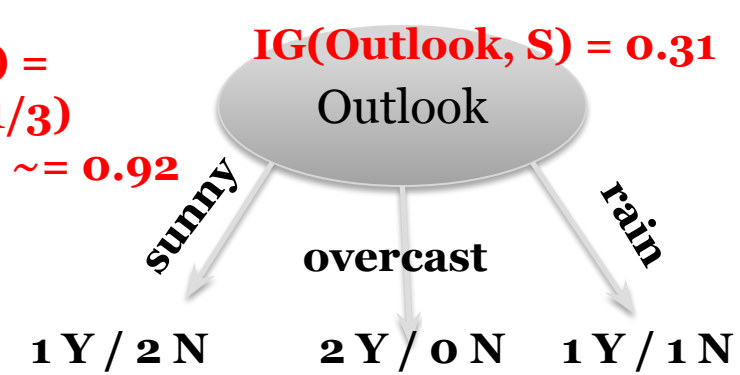


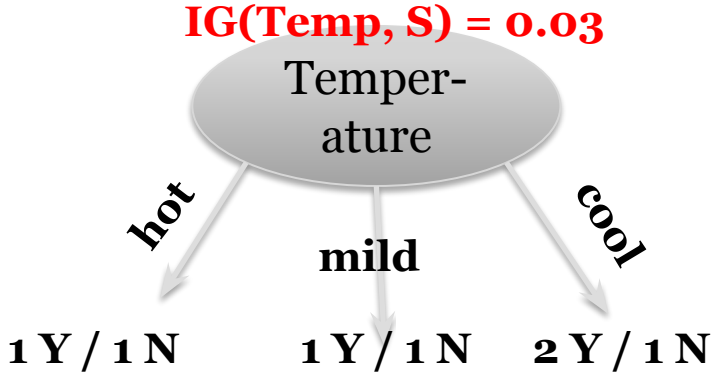
IG(A, S) = H(S) – [
3/7 * H(Outlook = Sunny) +
2/7 * H(Outlook = Overcast) +
2/7 * H(Outlook = Rain)]
= 0.99 – [3/7*.92 + 2/7*0 + 2/7*1]
= 0.31

Decision Tree Construction

No.	Outlook (O)	Temperature (T)	Humidity (H)	Play Golf (PG)
1	sunny	hot	high	N
2	sunny	mild	high	N
3	overcast	hot	high	Y
4	rain	mild	high	Y
5	sunny	cool	normal	Y
6	rain	cool	normal	N
7	overcast	cool	normal	Y

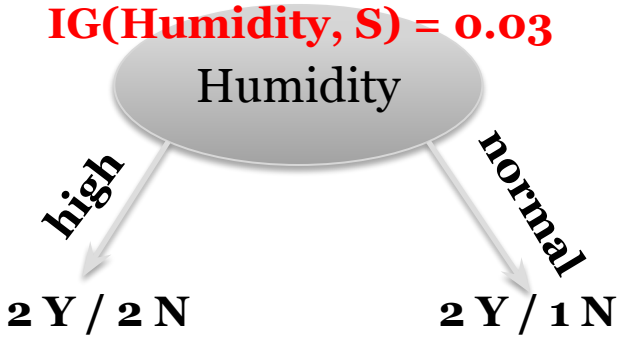
H(sunny) = -1/3*log$_2$(1/3) – 2/3*log$_2$(2/3) ~= 0.92





$$entropy(T) = -p_+ \log p_+ - p_- \log p_-,$$

$$IG(A,S) = H(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} H(S_v)$$



Naive Bayes Classifier

- First, we should understand Bayes' Theorem.

$$P(A|B)$$

- Probability that A occurs *given that* B occurred.

$$P(A \cap B) = P(A)P(B|A)$$

$$P(A \cap B) = P(B)P(A|B)$$

$$P(B)P(A|B) = P(A)P(B|A)$$

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Naive Bayes Classifier

For two random variables X and Y, Bayes theorem states that,

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

class variable

the instance features

Then class attribute value for instance X

$$\arg\max_{y_i} P(y_i|X)$$

We assume that features are independent given the class attribute

$$P(y_i|X) = \frac{P(X|y_i)P(y_i)}{P(X)}$$

$$P(X|y_i) = \Pi_{j=1}^{n} P(x_j|y_i) \Rightarrow P(y_i|X) = \frac{(\Pi_{j=1}^{n} P(x_j|y_i))P(y_i)}{P(X)}$$

NBC: An Example

No.	Outlook (O)	Temperature (T)	Humidity (H)	Play Golf (PG)
1	sunny	hot	high	N
2	sunny	mild	high	N
3	overcast	hot	high	Y
4	rain	mild	high	Y
5	sunny	cool	normal	Y
6	rain	cool	normal	N
7	overcast	cool	normal	Y
8	sunny	mild	high	?

$$
\begin{aligned}
(PG = Y|i_8) &= \frac{P(i_8|PG = Y)P(PG = Y)}{P(i_8)} \\
&= P(O = Sunny, T = mild, H = high|PG = Y) \\
&\quad \times \frac{P(PG = Y)}{P(i_8)} \\
&= P(O = Sunny|PG = Y) \times P(T = mild|PG = Y) \\
&\quad \times P(H = high|PG = Y) \times \frac{P(PG = Y)}{P(i_8)} \\
&= \frac{1}{4} \times \frac{1}{4} \times \frac{2}{4} \times \frac{\frac{4}{7}}{P(i_8)} = \frac{1}{56P(i_8)}.
\end{aligned}
$$

$$
\begin{aligned}
P(PG = N|i_8) &= \frac{P(i_8|PG = N)P(PG = N)}{P(i_8)} \\
&= P(O = Sunny, T = mild, H = high|PG = N) \\
&\quad \times \frac{P(PG = N)}{P(i_8)} \\
&= P(O = Sunny|PG = N) \times P(T = mild|PG = N) \\
&\quad \times P(H = high|PG = N) \times \frac{P(PG = N)}{P(i_8)} \\
&= \frac{2}{3} \times \frac{1}{3} \times \frac{2}{3} \times \frac{\frac{3}{7}}{P(i_8)} = \frac{4}{63P(i_8)}.
\end{aligned}
$$

$$P(X|y_i) = \Pi_{j=1}^{n} P(x_j|y_i)$$

$$\frac{1}{56P(i_8)} < \frac{4}{63P(i_8)} \rightarrow Play\ Golf = N$$

Regression

Regression

- In classification, class values or labels are categories {Play Golf, Don't Play Golf}
- In regression, the class attribute takes real values

$$y \approx f(X)$$

Class attribute(Dependent variable)
$y \in R$

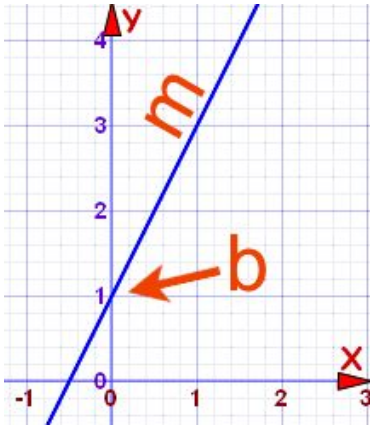
Features (Regressors)
$x_1, x_2, \ldots, x_m$

Regression finds the relation between y and the vector $(x_1, x_2, \ldots, x_m)$

Linear Regression

In linear regression, we assume the relation between the class attribute y and feature set $\boldsymbol{x}$ to be linear:

$$y = mx + b$$



$$y = \sum_{i=0}^{n} x_i w_i \qquad x_0 = 1$$

where $\boldsymbol{w}$ represents the vector of regression coefficients

- Regression can be solved by estimating *the weights* from the training data
  - Least squares is often used to solve the problem

Unsupervised Learning

Unsupervised Learning

Unsupervised division of instances into groups of similar objects

- Clustering is a form of **unsupervised learning**
  - The clustering algorithms do not have examples showing how the samples should be grouped together (unlabeled data)
- Clustering algorithms group together **similar items**

Similarity Measures: More Definitions

Measure Name	Formula	Type	Description
Mahalanobis	$d(X,Y) = \sqrt{(X-Y)^T Co^{-1}(X-Y)}$	Dissimilarity	X, Y are features vectors and *Co* is the Covariance matrix of the dataset
Manhattan	$d(X,Y) = \sum_i \lvert x_i - y_i \rvert$	Dissimilarity	X, Y are features vectors
L_p-norm	$d(X,Y) = (\sum_i \lvert x_i - y_i \rvert^n)^{\frac{1}{n}}$	Dissimilarity	X, Y are features vectors
Cosine	$c(X,Y) = \frac{X.Y}{\lvert X \rvert \lvert Y \rvert}$	Similarity	X, Y are features vectors and '.' represents the inner product

Once a distance measure is selected, instances are grouped using it.

Clustering

- Clusters are usually represented by compact and abstract notations.
- “Cluster centroids” are one common example of this abstract notation.
- Partitional Algorithms
  - Partition the dataset into a set of clusters
  - Each instance is assigned to a cluster exactly once and no instance remains unassigned to clusters.
  - k-Means

k-Means

The algorithm is the most commonly used clustering algorithm.

Algorithm 2 *K*-Means Algorithm

Require: A Dataset of Real-Value Attributes, *K* (number of Clusters)

1: **return** A Clustering of Data into *K* Clusters
2: Consider *K* random points in the data space as the initial cluster centroids.
3: **while** centroids have not converged **do**
4: Assign each data point to the cluster which has the closest cluster centroid.
5: If all data points have been assigned then recalculate the cluster centroids by averaging datapoints inside each cluster
6: **end while**

k-Means

- As an alternative, k-means implementations try to minimize an objective function. A well-known objective function in these implementations is the squared distance error:

$$\sum_{i=1}^{k} \sum_{j=1}^{n(i)} \|x_j^i - c_i\|^2,$$

- where x_j^i is the jth instance of cluster i, n(i) is the number of instances in cluster i, and ci is the centroid of cluster i.

- The process stops when the difference between the objective function values of two consecutive iterations of the k-means algorithm is bounded by some small value .

k-Means

- Finding the global optimum of the *k* partitions is computationally expensive (NP-hard).
- This is equivalent to finding the optimal centroids that minimize the objective function

- However, there are efficient heuristic algorithms that are commonly employed and converge quickly to an optimum that might not be global.
  - running k-means multiple times and selecting the clustering assignment that is observed most often or is more desirable based on an objective function, such as the squared error.

Evaluation with Ground Truth

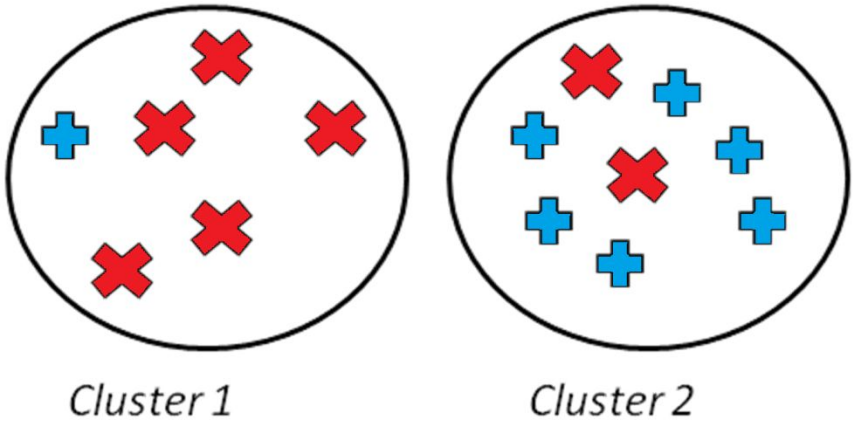
When ground truth is available, the evaluator has prior knowledge of what a clustering should be

- That is, we know the correct clustering assignments.

- We will discuss these methods in community analysis chapter

Evaluating the Clusterings

When we are given objects of two different kinds, the perfect clustering would be that objects of the same type are clustered together.



- Evaluation with ground truth
- Evaluation without ground truth

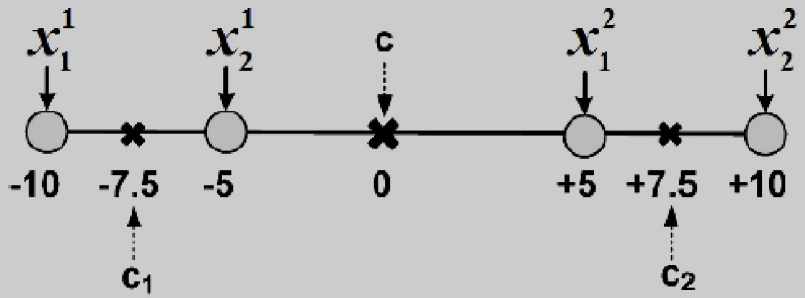
Evaluation without Ground Truth

- Cohesiveness
  - In clustering, we are interested in clusters that exhibit cohesiveness.
  - In cohesive clusters, instances inside the clusters are close to each other.
- Separateness
  - We are also interested in clusterings of the data that generates clusters that are well separated from one another

Cohesiveness

- Cohesiveness
  - In statistical terms, this is equivalent to having a small standard deviation, i.e., being close to the mean value.
  - In clustering, this translates to being close to the centroid of the cluster

$$cohesiveness = \sum_{i=1}^{k} \sum_{j=1}^{n(i)} dist(x_j^i, c_i)^2$$



$$cohesiveness = |-10-(-7.5)|^2+|-5-(-7.5)|^2+|5-7.5)|^2+|10-7.5|^2 = 25. \quad (5.59)$$

Separateness

- Separateness
  - We are also interested in clusterings with clusters that are well separated from one another.
  - In statistics, separateness can be measured by standard deviation.
  - Standard deviation is maximized when instances are far from the mean.
  - In clustering terms, this is equivalent to cluster centroids being far from the mean of the entire dataset

$$separateness = \sum_{i=1}^{k} dist(c, c_i)^2$$

Silhouette Index

- For any instance x that is a member of cluster C
- Compute the within-cluster average distance

$$a(x) = \frac{1}{|C| - 1} \sum_{y \in C, y \neq x} \|x - y\|^2.$$

- Compute the average distance between x and instances in cluster G that is closest to x in terms of the average distance between x and members of G

$$b(x) = \min_{G \neq C} \frac{1}{|G|} \sum_{y \in G} \|x - y\|^2.$$