

# **Principles/Social Media Mining**

**CIS 600**

## **Week 12: Community Detection**

**Edmund Yu, PhD**

**Associate Teaching Professor**

**esyu@syr.edu**

**November 12, 2020**

# Community

---

- ❖ **Community**: is formed by individuals such that those within a group interact with each other more frequently than with those outside the group
  - ❖ a.k.a. group, cluster, cohesive subgroup, module in different contexts
- ❖ Two types of groups in social media
  - ❖ **Explicit Groups**: formed by user subscriptions
  - ❖ **Implicit Groups**: implicitly formed by social interactions

# Network Modeling

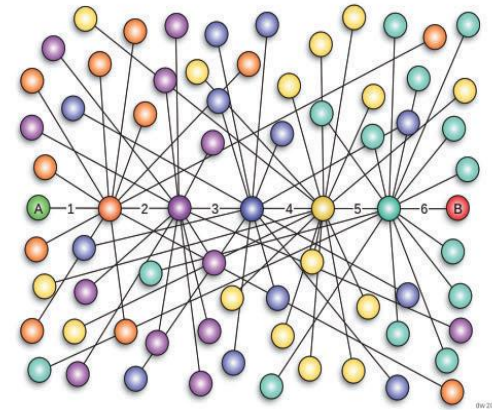
- ❖ Large Networks demonstrate **statistical patterns**

- ❖ Small-world effects (e.g., 6 degrees of separation)

- ❖ Power-law distribution (a.k.a. scale-free distribution)

$$f(x) = ax^k \quad f(cx) = a(cx)^k = c^k f(x) \propto f(x).$$

- ❖ **Community structure (high clustering coefficient)**



- ❖ Model the network dynamics

- ❖ Find a mechanism such that the statistical patterns observed in large-scale networks can be reproduced.

- ❖ Examples: random graph, preferential attachment process, Watts and Strogatz model

- ❖ Used for simulation to understand network properties

- ❖ Thomas Schelling's famous simulation: What could cause the segregation

- ❖ Network robustness under attack

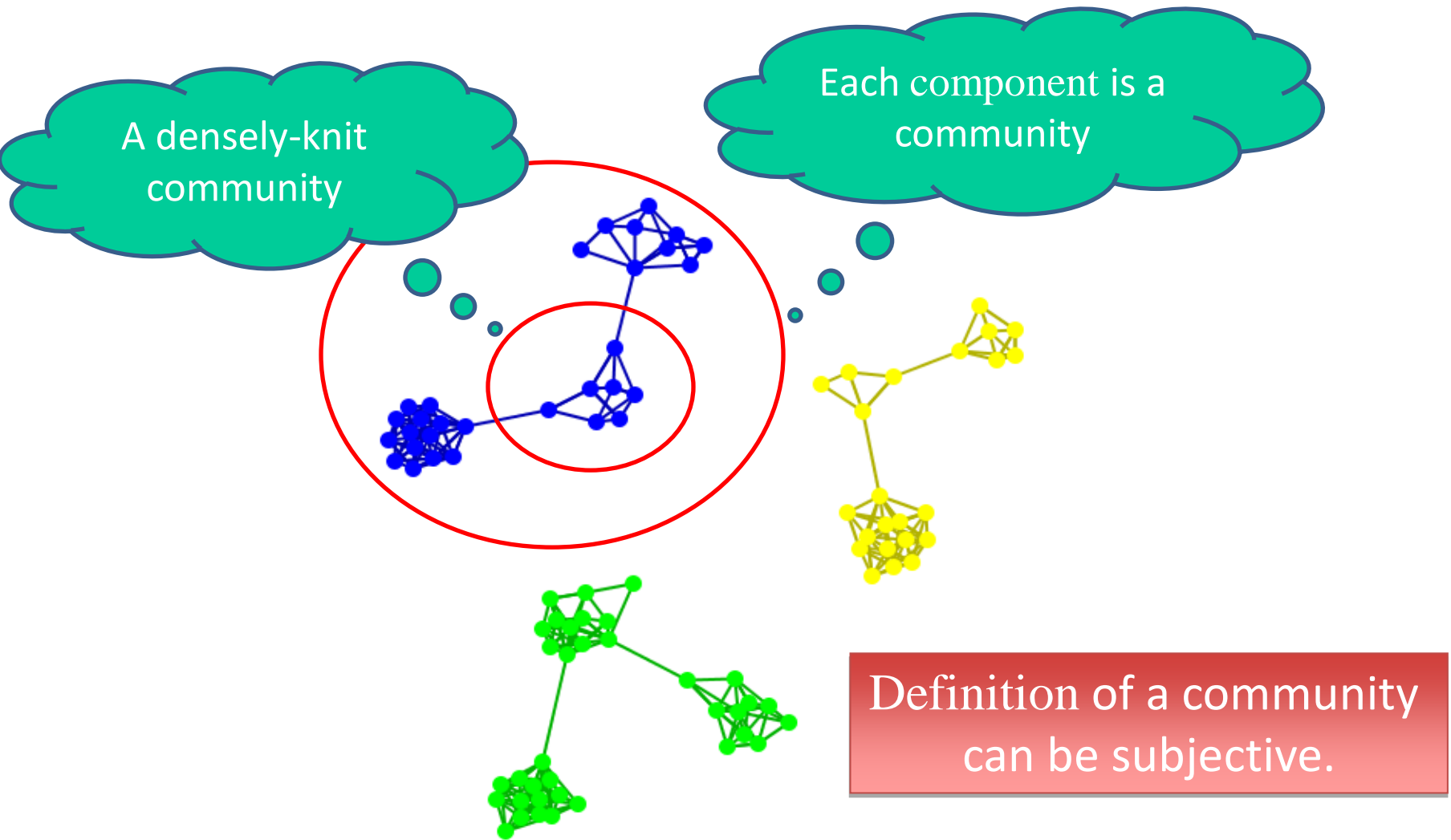
- ❖ Information diffusion within a given network structure

# Community Detection

---

- ❖ **Community detection**: is about discovering groups in a network where individuals' group memberships are not explicitly given
- ❖ The community structure, resulting from user interactions, provides rich information about the relationship between users
  - ❖ Can complement other kinds of information, e.g. user profile
  - ❖ Help network visualization and navigation
  - ❖ Provide basic information for other tasks, e.g. recommendation

# Subjectivity of Community Definition



# Taxonomy of Community Detection

---

- ❖ Roughly, community detection methods can be divided into 4 categories:



- ❖ **Node-Centric Community**

- ❖ **Each node** in a group satisfies certain properties

- ❖ Methods: cliques; k-cliques; k-clubs

- ❖ **Group-Centric Community**

- ❖ Consider the connections **within a group** as a whole. The group has to satisfy certain properties without zooming into node-level

- ❖ Methods: quasi-cliques

- ❖ **Network-Centric Community**

- ❖ Partition **the whole network** into several disjoint sets

- ❖ Clustering based on vertex similarity; latent space models; block models; spectral clustering; modularity maximization

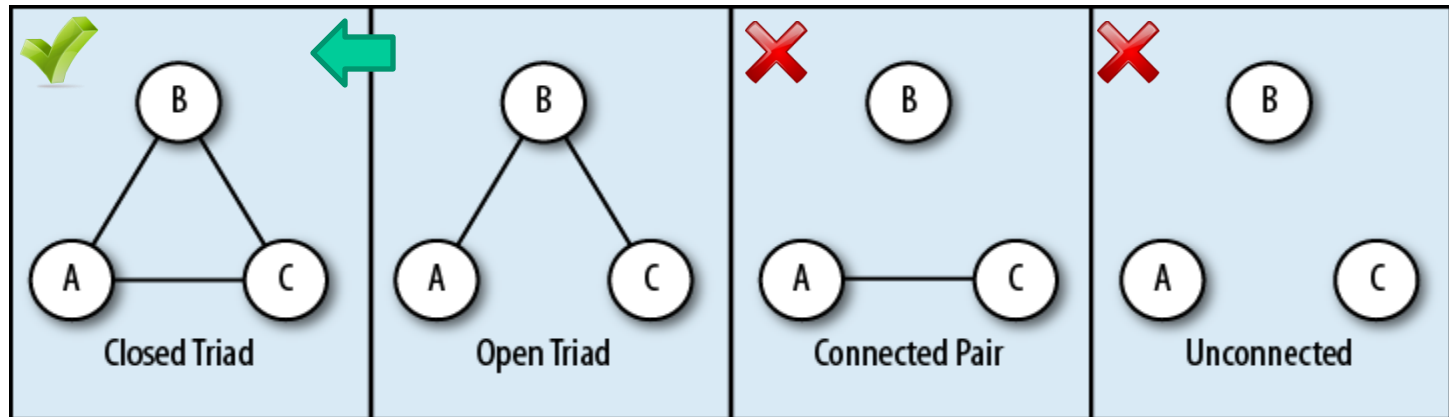
- ❖ **Hierarchy-Centric Community**

- ❖ Construct a **hierarchical structure** of communities

- ❖ Methods: Divisive clustering; Agglomerative clustering

# Node-Centric Community Detection

- ❖ Triads can be considered as smallest communities.
- ❖ A triad is simply three nodes interlinked in some way.
- ❖ All possible undirected triads are shown in the following figure.
- ❖ As you can see, only the first two have all of their nodes interconnected, and thus present a significant interest.



- ❖ If two people in a social network have a friend in common, then there is an increased likelihood that they will become friends themselves at some point in the future. ([triadic closure principle](#))

# Triads

---

- ❖ Triads represent perhaps the oldest piece of research in the entire field:
  - ❖ In 1908, Georg Simmel, a contemporary of Max Weber and a member of his intellectual circles, authored a piece called “*The Treatise on the Triad*”:
  - ❖ In a dyad (i.e., two nodes connected to each other):
    - ❖ Each person is able to retain their individuality while maintaining a close relationship.
    - ❖ Exchange of information and ideas happens, but at the same time it does not subjugate the individual to the group.
  - ❖ In a triad:
    - ❖ The third individual becomes at a source of balance (providing second opinions and calming nerves)
    - ❖ The third node is also a feedback loop —information from A can pass to B, then to C, and back to A (forming culture)



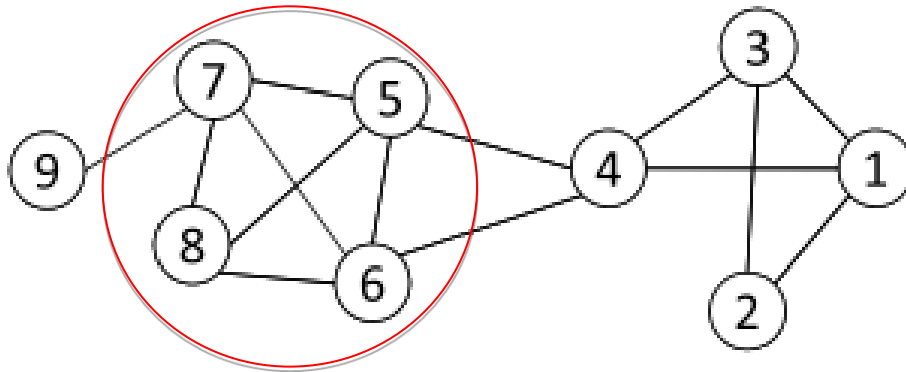
# Triads

---

- ❖ Another study in the same milieu was done by Newcomb, which was conducted in the early 1960s.
  - ❖ At the beginning of a semester, 17 students were recruited to live in a frat house for a semester, in exchange for their personal data.
  - ❖ Every week, researchers interviewed every one of the students asking them to rank their interactions with their fraternity brothers from 1 (best) to 16 (worst).
  - ❖ The findings of this study were:
    - ❖ **Asymmetric ties** (e.g., “I like you more than you like me”) were the least stable of all, lasting no more than two weeks
    - ❖ **Symmetric ties** (dyads where 2 people like each other about equally) were significantly more stable
    - ❖ **Triadic structures** were the most stable over time, with students smoothing over conflicts, organizing events (parties?) together, and overall defining the tone of interaction among the fraternity brothers.

# Cliques

- ❖ **Clique:** a maximum complete subgraph in which all nodes are connected to each other



Nodes 5, 6, 7 and 8 form a clique

- ❖ The word “maximum” means that no other nodes can be added to the clique without making it less connected.
- ❖ Essentially, a clique consists of several overlapping closed triads, and inherits many of the culture-generating, and amplification properties of closed triads.

# Finding the Maximum Clique

---

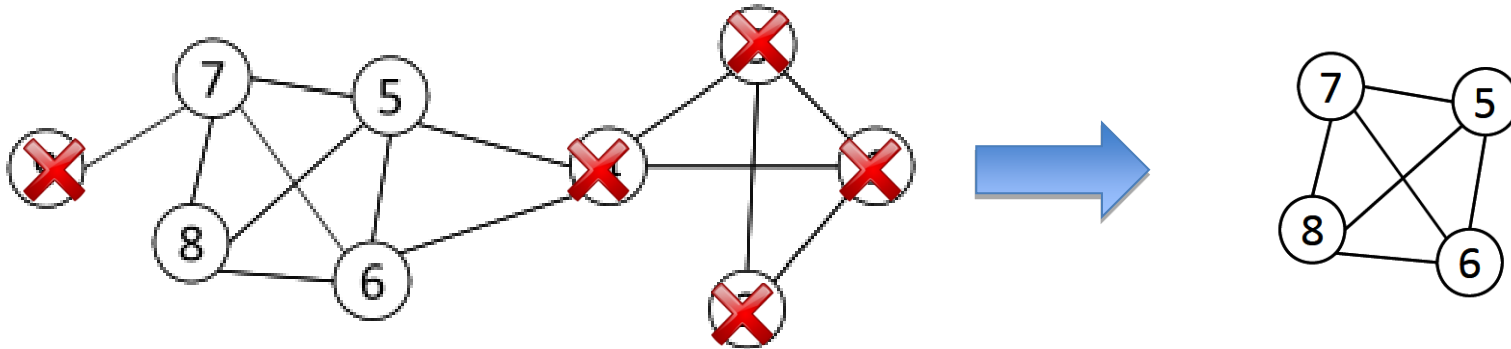
## Observation:

- ❖ In a clique of size  $k$ , each node maintains degree  $\geq k-1$ 
  - ❖ Nodes with degree  $< k-1$  will not be included in the maximum clique

## The Algorithm:

- ❖ Recursively apply the following **pruning** procedure
  - ❖ Sample a sub-network from the given network, and find a clique in the sub-network, say, by a greedy approach
  - ❖ Suppose the clique above is size  $k$ , in order to find out a larger  $(k+1)$  clique, all nodes with degree  $\leq k-1$  should be removed.
- ❖ Repeat until the network is small enough
- ❖ Note: Many nodes will be pruned as social media networks follow a power law distribution for node degrees

# Maximum Clique Example



- ❖ Suppose we sample a sub-network with nodes  $\{1-9\}$  and find a clique  $\{1, 2, 3\}$  of size 3 (this is  $k$ )
- ❖ In order to find a clique of size 4 (this is  $k+1$ ), we have to remove all nodes with degree  $\leq 2$ 
  - ❖ Remove nodes 2 and 9
  - ❖ Remove nodes 1 and 3
  - ❖ Remove node 4

### 3.9.2 networkx.algorithms.clique.find\_cliques

**find\_cliques** (*G*)

Returns all maximal cliques in an undirected graph.

For each node *v*, a *maximal clique for v* is a largest complete subgraph containing *v*. The largest maximal clique is sometimes called the *maximum clique*.

This function returns an iterator over cliques, each of which is a list of nodes. It is an iterative implementation, so should not suffer from recursion depth issues.

**Parameters** *G* (*NetworkX graph*) – An undirected graph.

**Returns** An iterator over maximal cliques, each of which is a list of nodes in *G*. The order of cliques is arbitrary.

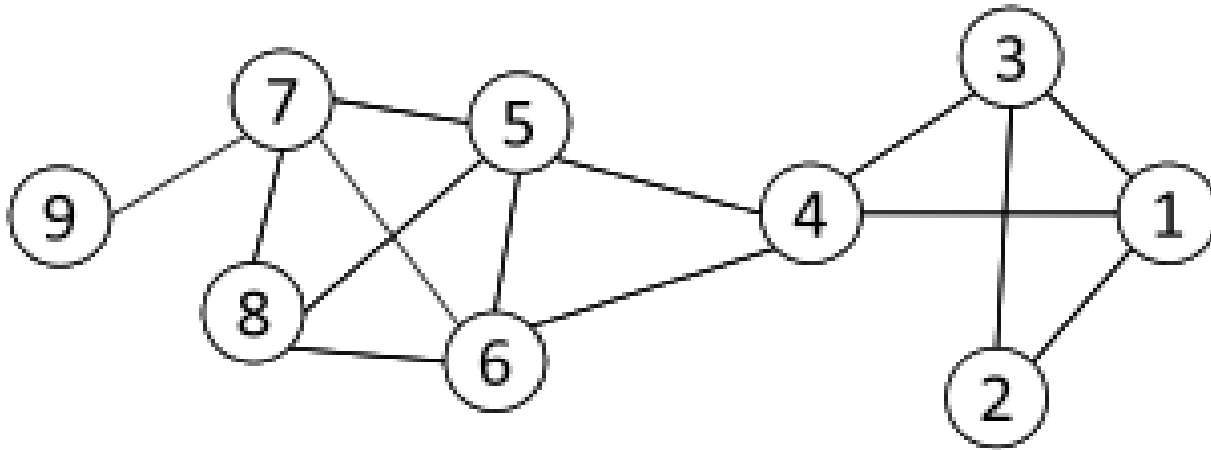
**Return type** iterator

---

<sup>1</sup> Yun Zhang, Abu-Khzam, F.N., Baldwin, N.E., Chesler, E.J., Langston, M.A., Samatova, N.F., “Genome-Scale Computational Approaches to Memory-Intensive Applications in Systems Biology”. *Supercomputing*, 2005. Proceedings of the ACM/IEEE SC 2005 Conference, pp. 12, 12–18 Nov. 2005. <<https://doi.org/10.1109/SC.2005.29>>.

# Cliques - Results

---



Clique 1 = [8, 5, 6, 7]

Clique 2 = [2, 1, 3]

Clique 3 = [4, 1, 3]

Clique 4 = [4, 5, 6]

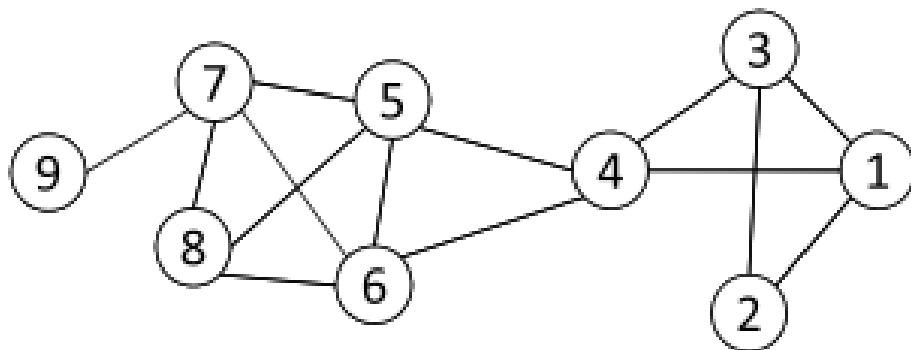
Clique 5 = [9, 7]

# Clique Percolation Method (CPM)

---

- ❖ Clique is a very strict definition, unstable
- ❖ Normally we use cliques as a **core** or a **seed** to find larger communities
- ❖ CPM is such a method to find **overlapping** communities
  - ❖ **Input**
    - ❖ A parameter  $k$ , and a network
  - ❖ **Procedure**
    - ❖ Find out all cliques of size  $k$  in a given network
    - ❖ Construct a **clique graph**: (each node in this graph is a clique)
      - ❖ Two cliques of size  $k$  are considered adjacent if they share  $k-1$  nodes
    - ❖ Each (connected) component in the clique graph forms a community

# CPM/Clique Graph Example



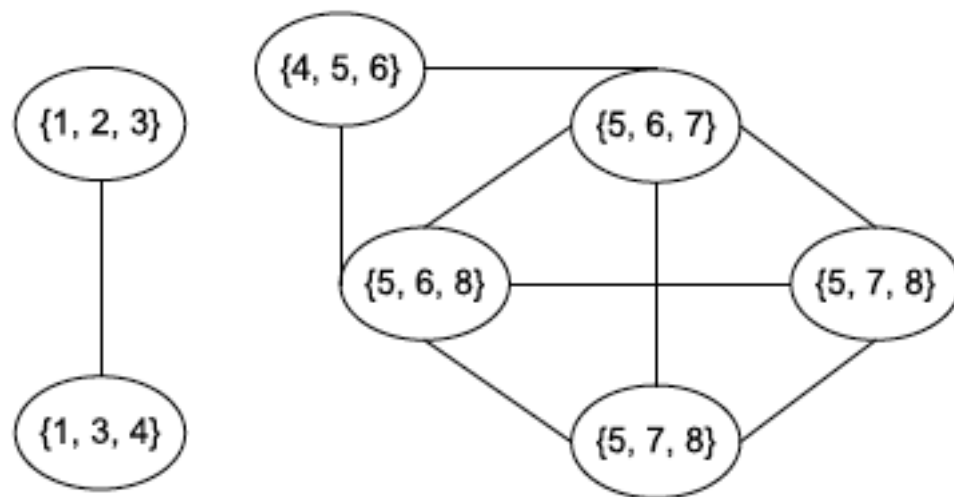
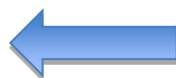
**Cliques of size 3:**

$\{1, 2, 3\}$ ,  $\{1, 3, 4\}$ ,  $\{4, 5, 6\}$ ,  
 $\{5, 6, 7\}$ ,  $\{5, 6, 8\}$ ,  $\{5, 7, 8\}$ ,  
 $\{6, 7, 8\}$



**Communities:**

$\{1, 2, 3, \underline{4}\}$   
 $\{\underline{4}, 5, 6, 7, 8\}$





### 3.13.3 K-Clique

---

<code>k_clique_communities(G, k[, cliques])</code>	Find k-clique communities in graph using the percolation method.
--	--

---

#### `networkx.algorithms.community.kclique.k_clique_communities`

**k\_clique\_communities** (*G*, *k*, *cliques=None*)

Find k-clique communities in graph using the percolation method.

A k-clique community is the union of all cliques of size *k* that can be reached through adjacent (sharing *k*-1 nodes) k-cliques.

#### Parameters

- **G** (*NetworkX graph*)
- **k** (*int*) – Size of smallest clique
- **cliques** (*list or generator*) – Precomputed cliques (use `networkx.find_cliques(G)`)

#### Returns

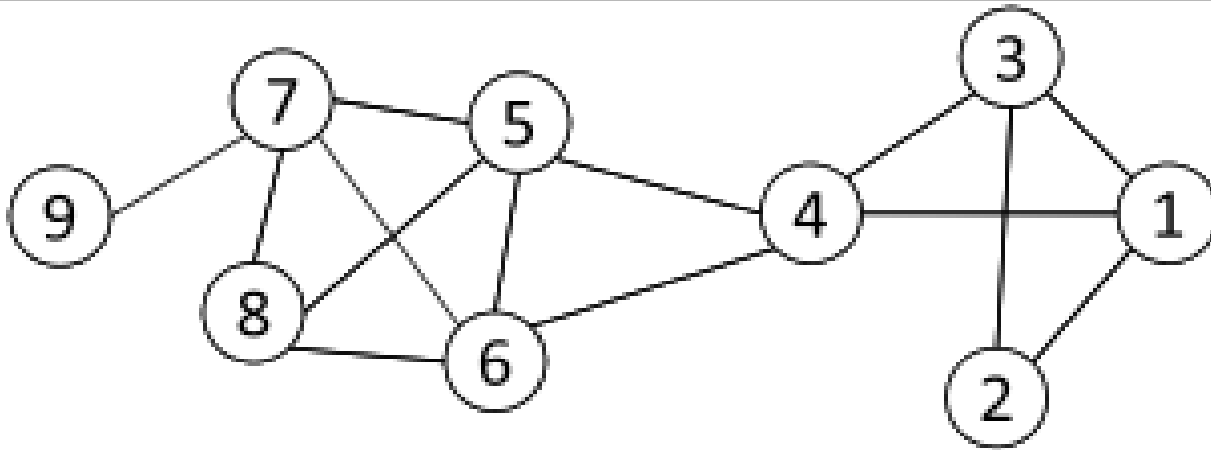
**Return type** Yields sets of nodes, one for each k-clique community.

#### Examples

```
>>> from networkx.algorithms.community import k_clique_communities
>>> G = nx.complete_graph(5)
>>> K5 = nx.convert_node_labels_to_integers(G, first_label=2)
>>> G.add_edges_from(K5.edges())
```

# k-Clique Community - Results

---



2\_clique community:

frozenset([1, 2, 3, 4, 5, 6, 7, 8, 9])

3\_clique communities:

frozenset([4, 5, 6, 7, 8])

frozenset([1, 2, 3, 4])

4\_clique community:

frozenset([8, 5, 6, 7])

# Taxonomy of Community Detection

---

- ❖ Roughly, community detection methods can be divided into 4 categories:

- ❖ **Node-Centric Community**

- ❖ **Each node** in a group satisfies certain properties

- ❖ Methods: cliques; k-cliques; k-clubs



- ❖ **Group-Centric Community**

- ❖ Consider the connections **within a group** as a whole. The group has to satisfy certain properties without zooming into node-level

- ❖ Methods: quasi-cliques

- ❖ **Network-Centric Community**

- ❖ Partition **the whole network** into several disjoint sets

- ❖ Clustering based on vertex similarity; latent space models; block models; spectral clustering; modularity maximization

- ❖ **Hierarchy-Centric Community**

- ❖ Construct a **hierarchical structure** of communities

- ❖ Methods: Divisive clustering; Agglomerative clustering

# Group-Centric Community Detection

---

❖ A group-centric criterion considers connections inside a group as whole.

❖ It is acceptable to have some nodes in the group to have low connectivity as long as the group overall satisfies certain requirements.

❖ E.g., the **group density**  $\geq$  a given threshold (see below)

❖ A subgraph  $G_s(V_s, E_s)$  is a  $\gamma$ -dense **quasi-clique** if

$$\frac{2|E_s|}{|V_s|(|V_s| - 1)} \geq \gamma$$

where the denominator is the maximum number of degrees.

❖ Clearly, the quasi-clique becomes a clique when  $\gamma = 1$ .

Note: the bins are width one, hence  $\text{len}(\text{lst})$  can be large ( $\text{Order}(\text{number\_of\_edges})$ )

### 4.1.3 `networkx.classes.function.density`

**density**(*G*)

Return the density of a graph.

The density for undirected graphs is

$$d = \frac{2m}{n(n-1)},$$

and for directed graphs is

$$d = \frac{m}{n(n-1)},$$

where *n* is the number of nodes and *m* is the number of edges in *G*.

#### Notes

The density is 0 for a graph without edges and 1 for a complete graph. The density of multigraphs can be higher than 1.

Self loops are counted in the total number of edges so graphs with self loops can have density higher than 1.

### 4.1.4 `networkx.classes.function.info`

**info**(*G*, *n=None*)

Print short summary of information for the graph *G* or the node *n*.

#### Parameters

- *G* (*Networkx graph*) – A graph
- *n* (*node (any hashable)*) – A node in the graph *G*

### 4.1.5 `networkx.classes.function.create_empty_copy`

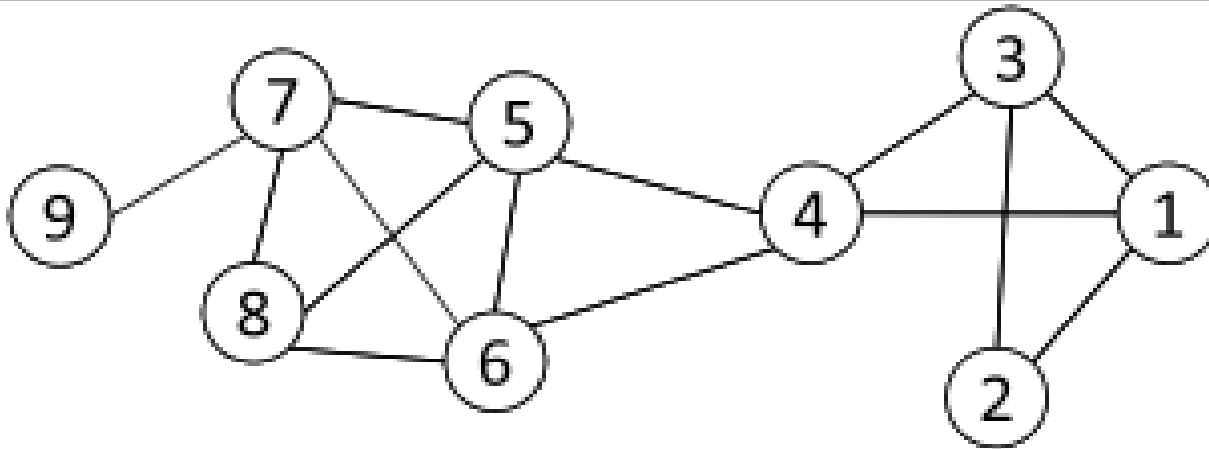
**create\_empty\_copy**(*G*, *with\_data=True*)

Return a copy of the graph *G* with all of the edges removed.

#### Parameters

# Density - Results

---



`frozenset([1, 2, 3, 4, 5, 6, 7, 8, 9])` → Density = 0.38888888888889

3\_clique communities:

`frozenset([4, 5, 6, 7, 8])` → Density = 0.8

`frozenset([1, 2, 3, 4])` → Density = 0.83333333333333

4\_clique communities :

`frozenset([5, 6, 7, 8])` → Density = 1

# Search for Quasi-Cliques

---

- ❖ It is not a trivial task to search for quasi-cliques.
  - ❖ So strategies similar to those of finding cliques should be exploited.
- ❖ In Abello et al. (2002), a greedy search followed by pruning is employed to find the maximal  $\gamma$ -dense quasi-clique in a network. ,
  - ❖ Though the solution returned by the algorithm does not guarantee to be optimal, it works reasonably well in most cases.

<

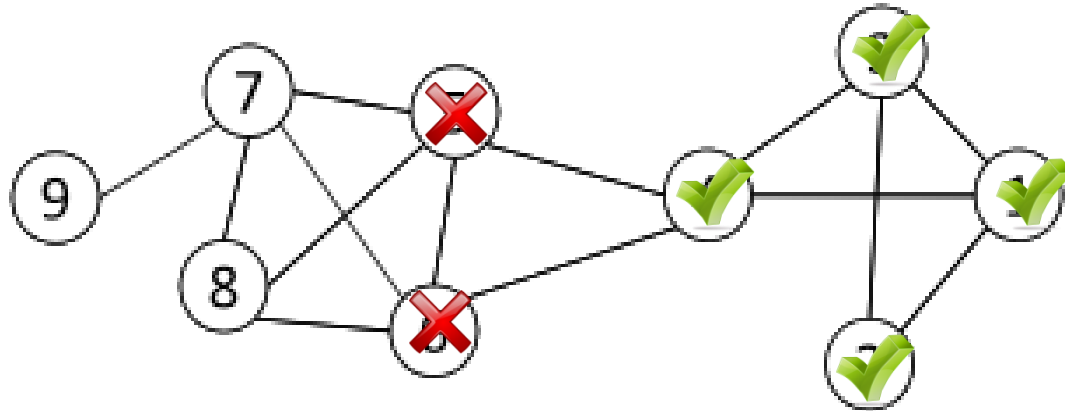
# Search for Quasi-Cliques

---

- ❖ The iterative algorithm consists of two steps - local search and heuristic pruning:
  - ❖ Local search: Sample a sub-network from the given network and search for a maximal quasi clique in the sub-network. A greedy approach is to aggressively expand a quasi-clique by encompassing those high-degree neighboring nodes until the density drops below  $\gamma$ .
  - ❖ Heuristic pruning: If we know a  $\gamma$ -dense quasi-clique of size  $k$ , then a heuristic is to prune those “peelable” nodes and their edges.
    - ❖ A node  $v$  is peelable if  $v$  and its neighbors all have degree less than  $k\gamma$  because it is less likely to contribute to a larger quasi-clique by including such a node.
- ❖ This process is repeated until the network is reduced to a reasonable size so that a maximal quasi clique can be found directly.

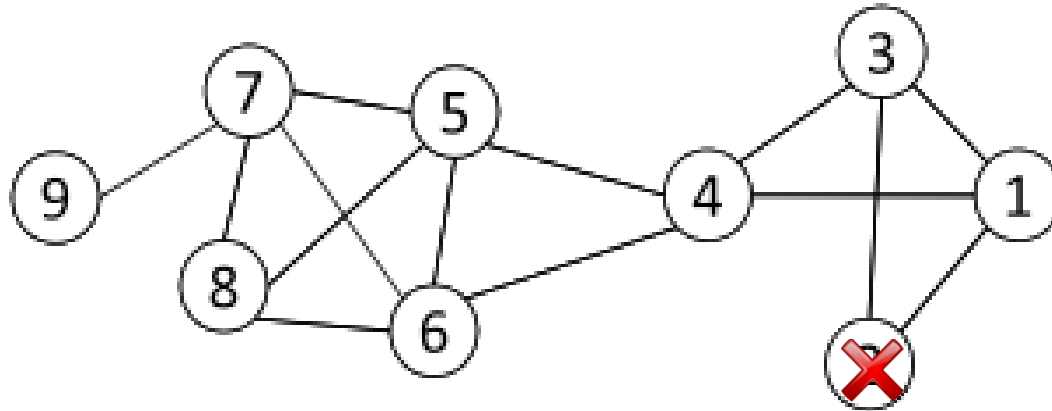


# Quasi-Clique Example – Greedy 1



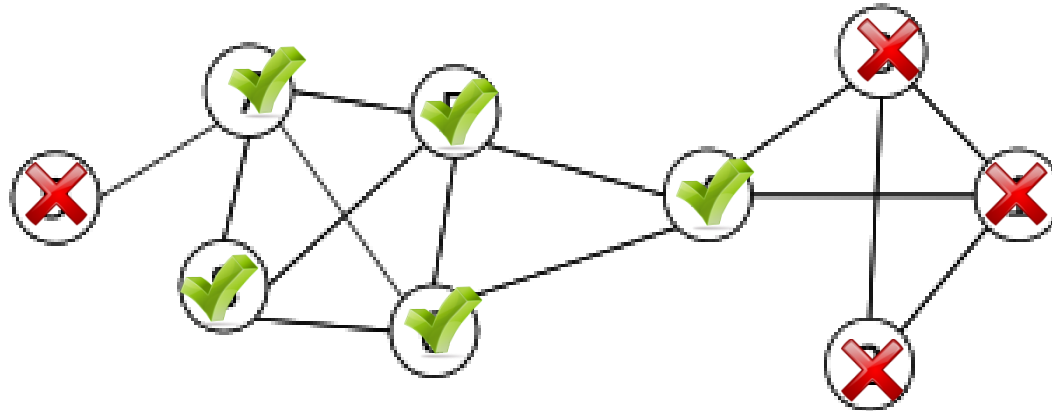
- ❖ Suppose we start with a clique  $\{1, 2, 3\}$ ;  $\gamma$  is set to 0.8
- ❖ In order to find a quasi-clique of size 4, we try node 4 first
- ❖  $\text{Density}([1, 2, 3, 4]) = 0.833333333333$
- ❖ In order to find a quasi-clique of size 5, we try node 5 first
- ❖  $\text{Density}([1, 2, 3, 4, 5]) = 0.6 < \mathbf{0.8}$
- ❖ Next, we try node 6
- ❖  $\text{Density}([1, 2, 3, 4, 6]) = 0.6 < \mathbf{0.8}$

# Quasi-Clique Example – Pruning 1



- ❖  $\gamma$  set to 0.8 as before;  $k = 4$ , so  $\gamma k = 3.2$
- ❖ Node 2 peeled
- ❖ Node 1 survives
- ❖ Node 3 survives
- ❖ Node 4 survives
- ❖ Node 5 survives
- ❖ Node 6 survives...


# Quasi-Clique Example – Greedy 2



- ❖ Since 2 is peeled, we start with a clique  $\{5, 6, 7, 8\}$ ;  $\gamma$  is set to 0.8 like before
- ❖ In order to find a quasi-clique of size 5, we add node 4
- ❖  $\text{Density}([4, 5, 6, 7, 8]) = 0.8$
- ❖ In order to find a quasi-clique of size 6, we add node 9
- ❖  $\text{Density}([4, 5, 6, 7, 8, 9]) = 0.6$
- ❖ If we tried node 9 first instead of 4,  $\text{Density}([5, 6, 7, 8, 9]) = 0.7$
- ❖ Then try node 3
- ❖  $\text{Density}([3, 4, 5, 6, 7, 8, 9]) = 0.6$
- ❖ Finally, try node 1
- ❖  $\text{Density}([1, 4, 5, 6, 7, 8, 9]) = 0.6$

# Taxonomy of Community Detection

---

- ❖ Roughly, community detection methods can be divided into 4 categories:
  - ❖ **Node-Centric Community**
    - ❖ **Each node** in a group satisfies certain properties
    - ❖ Methods: cliques; k-cliques; k-clubs
  - ❖ **Group-Centric Community**
    - ❖ Consider the connections **within a group** as a whole. The group has to satisfy certain properties without zooming into node-level
    - ❖ Methods: quasi-cliques
  -  ❖ **Network-Centric Community**
    - ❖ Partition **the whole network** into several disjoint sets
    - ❖ Clustering based on vertex similarity; latent space models; block models; spectral clustering; modularity maximization
  - ❖ **Hierarchy-Centric Community**
    - ❖ Construct a **hierarchical structure** of communities
    - ❖ Methods: Divisive clustering; Agglomerative clustering

# Network-Centric Community Detection

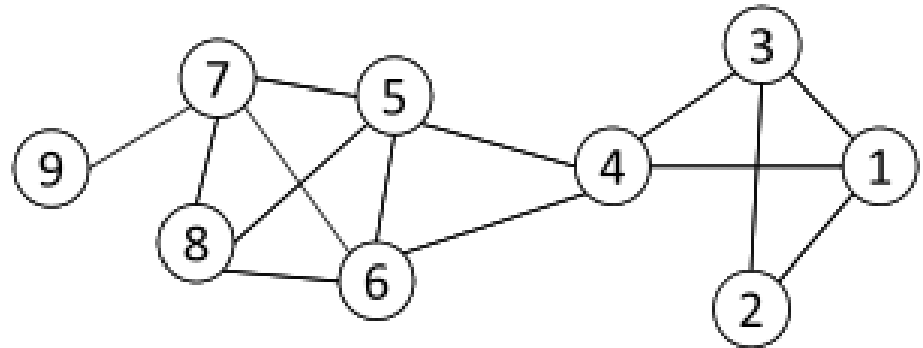
---

- ❖ Network-centric criterion needs to consider the connections within a network globally
- ❖ Goal: partition nodes of a network into disjoint sets
- ❖ Approaches:
  1. Clustering based on **node/vertex similarity**
  2. Latent space models
  3. Block model approximation
  4. Spectral clustering
  5. Modularity maximization

# Clustering based on Node Similarity

- ❖ Apply **k-means** or **similarity**-based clustering to nodes
- ❖ Node similarity is defined in terms of **the similarity of their neighborhood** (see next slide)
- ❖ A related concept is **structural equivalence**: two nodes are structurally equivalent iff they are connecting to the same set of actors

Nodes 1 and 3 are  
structurally equivalent;  
So are nodes 5 and 6.

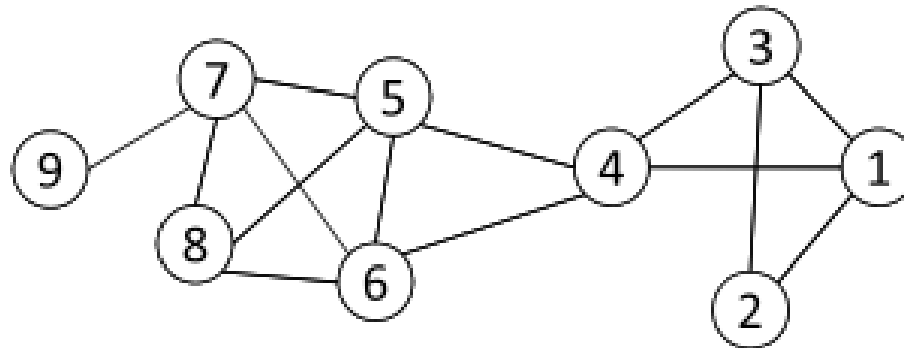


- ❖ However, structural equivalence is too restricted for practical use.

# Vertex/Node Similarity

❖ Jaccard Similarity  $Jaccard(v_i, v_j) = \frac{|N_i \cap N_j|}{|N_i \cup N_j|}$

❖ Cosine Similarity  $Cosine(v_i, v_j) = \frac{|N_i \cap N_j|}{\sqrt{|N_i| \cdot |N_j|}}$



$$Jaccard(4, 6) = \frac{|\{5\}|}{|\{1, 3, 4, 5, 6, 7, 8\}|} = \frac{1}{7}$$

$$cosine(4, 6) = \frac{1}{\sqrt{4 \cdot 4}} = \frac{1}{4}$$

# $k$ -Means

---

- ❖ The  $k$ -means clustering algorithm has been used frequently to find communities in networks.
  - ❖ Given a parameter  $k$ , the  $k$ -means clustering algorithm partitions data instances into  $k$  clusters.
  - ❖ Each cluster is associated with a **centroid** (center point).
  - ❖ The centroid of a cluster is typically computed as the mean of the data points assigned to the cluster, and each data point is assigned to the **closest** cluster.
  - ❖ The closeness of data points to a centroid depends on the definition of **distance** or **the inverse of similarity**.
  - ❖ In classical  $k$ -means, **Euclidean distance** is employed and a data instance is assigned to the centroid with minimum distance.



# *k*-Means

---

❖ Given the desired number of clusters  $k$ , the basic **k-means** algorithm is carried out in the following steps:

1: Select  $\underline{k}$  points as initial centroids;

2: **Repeat**

2.1: Form  $\underline{k}$  clusters by assigning all points to their closest centroids;

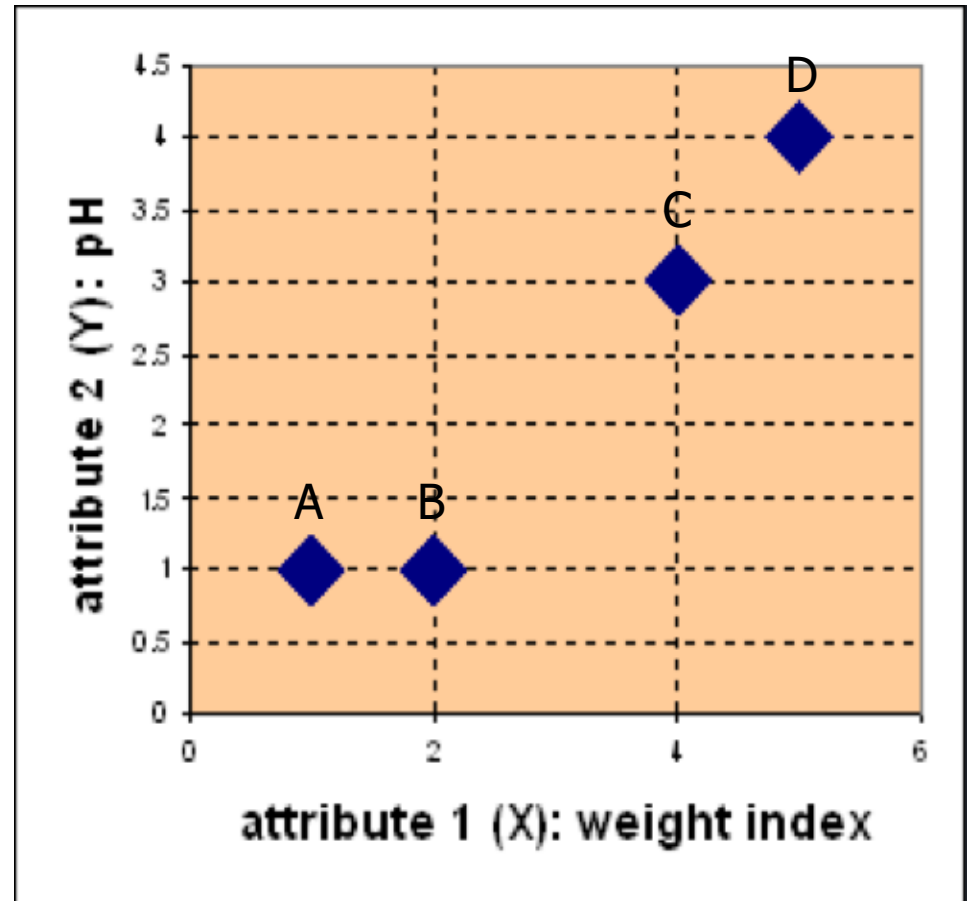
2.2: Recompute the centroid of each cluster;

3: **Until the centroids do not change.**

# A Simple Example

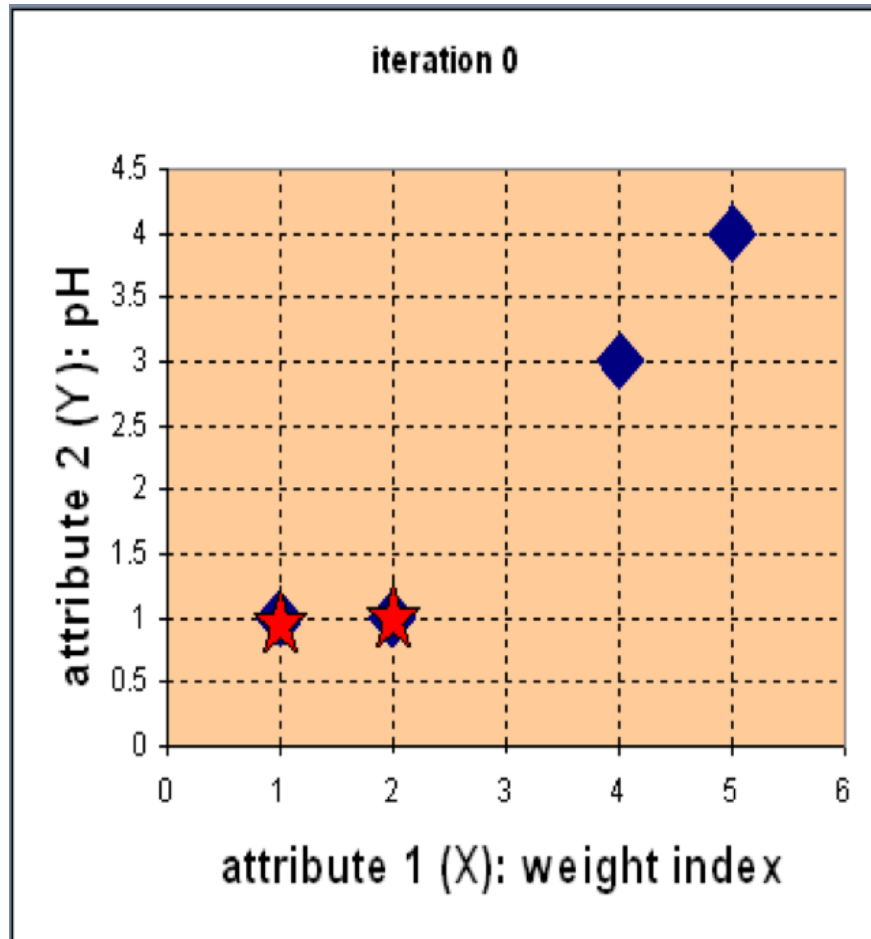
Suppose we have 4 types of medicines and each has two attributes (pH and weight index). Our goal is to group these objects into  $k=2$  group of medicine.

Medicine	Weight	pH-Index
A	1	1
B	2	1
C	4	3
D	5	4



# A Simple Example

❖ Step 1: Use initial seeds for partitioning



$$c_1 = A, c_2 = B$$

$$D^0 = \begin{bmatrix} 0 & 1 & 3.61 & 5 \\ 1 & 0 & 2.83 & 4.24 \end{bmatrix} \quad \begin{array}{l} c_1 = (1,1) \text{ group-1} \\ c_2 = (2,1) \text{ group-2} \end{array}$$

	A	B	C	D	
	1	2	4	5	X
	1	1	3	4	Y

$$d(D, c_1) = \sqrt{(5-1)^2 + (4-1)^2} = 5$$

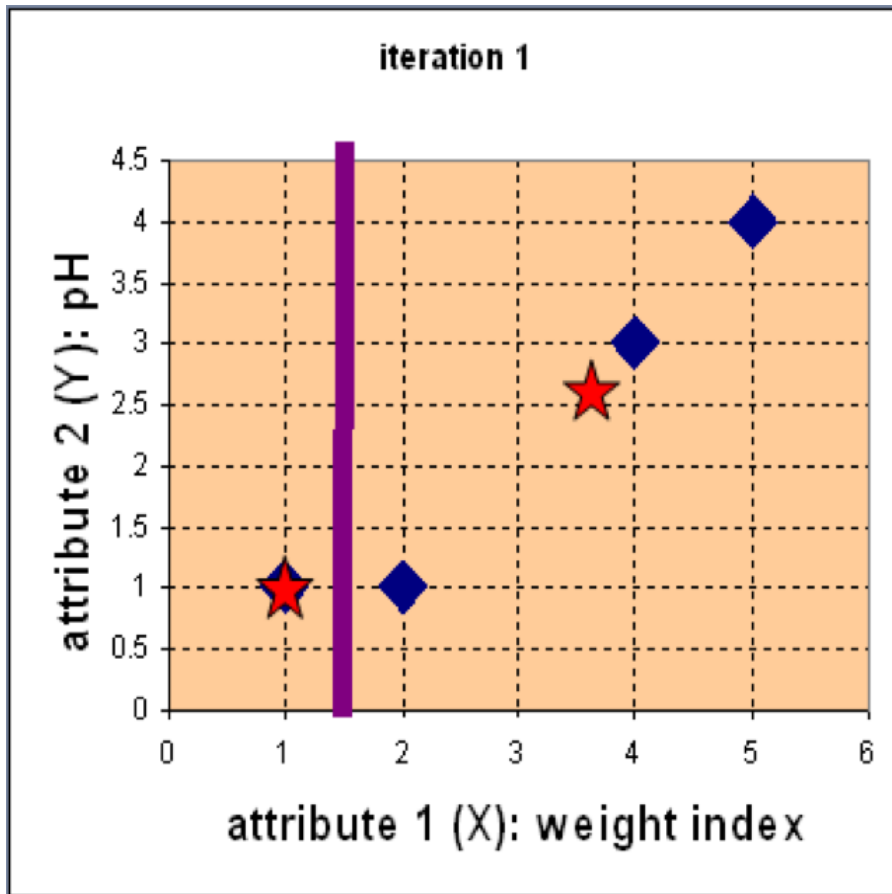
$$d(D, c_2) = \sqrt{(5-2)^2 + (4-1)^2} = 4.24$$

Euclidean distance

Assign each object to the closest cluster

# A Simple Example

- ❖ Step 2: Compute new centroids of the current partition



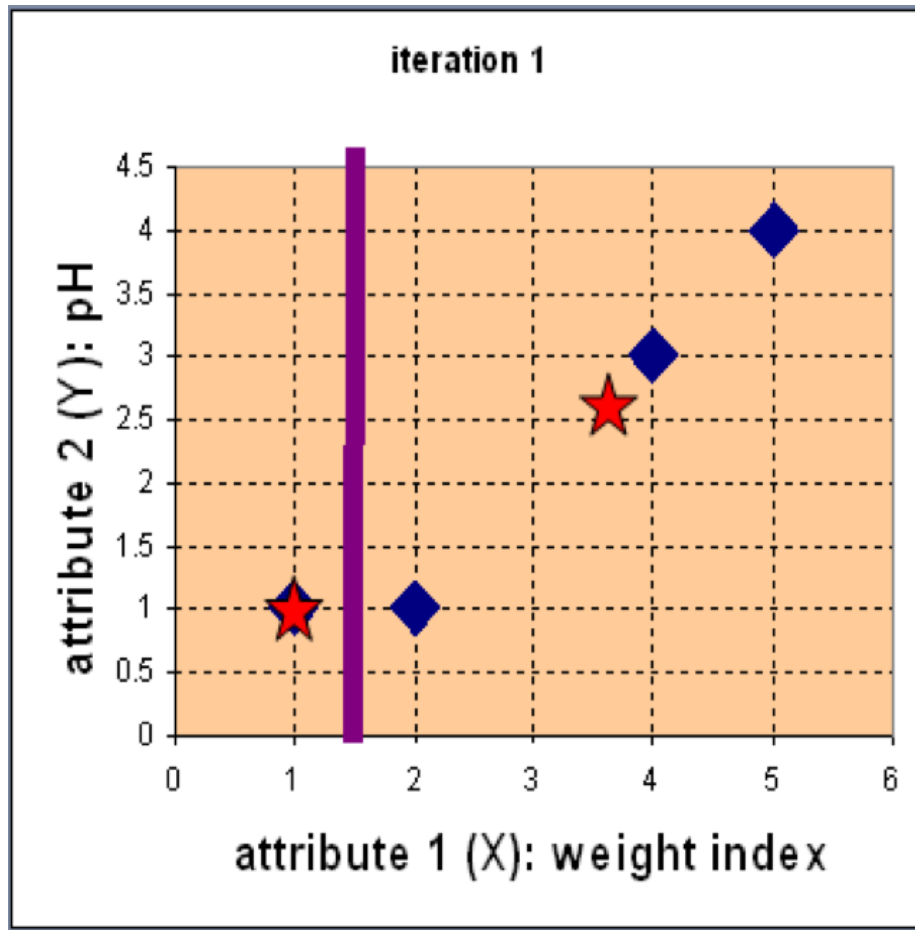
Knowing the members of each cluster, now we compute the new **centroid** of each group based on these new memberships.

$$c_1 = (1, 1)$$

$$\begin{aligned} c_2 &= \left( \frac{2 + 4 + 5}{3}, \frac{1 + 3 + 4}{3} \right) \\ &= (11/3, 8/3) \\ &= (3.67, 2.67) \end{aligned}$$

# A Simple Example

❖ Step 2: Renew membership based on new centroids



Compute the distance of all objects to the new centroids

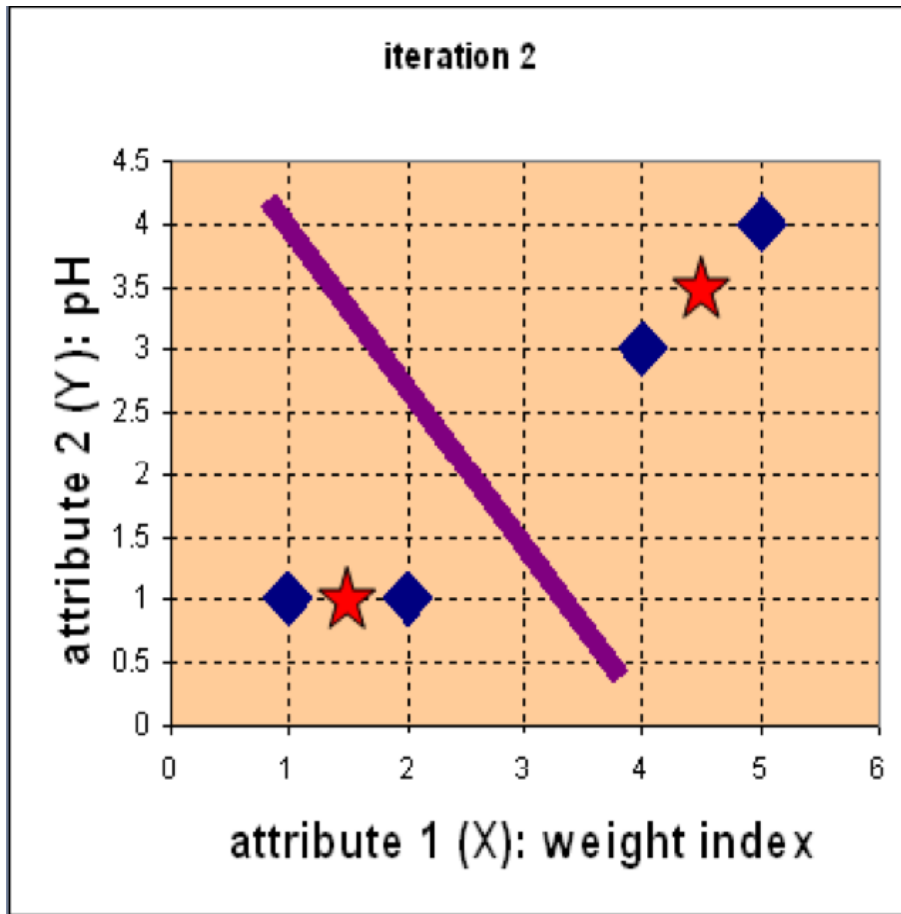
$$D^1 = \begin{bmatrix} 0 & 1 & 3.61 & 5 \\ 3.14 & 2.36 & 0.47 & 1.89 \end{bmatrix} \quad \begin{array}{l} \mathbf{c}_1 = (1, 1) \text{ group-1} \\ \mathbf{c}_2 = (\frac{11}{3}, \frac{8}{3}) \text{ group-2} \end{array}$$

	A	B	C	D	
X	1	2	4	5	
Y	1	1	3	4	

Assign the membership to objects

# A Simple Example

❖ Step 3: Repeat the first two steps until its convergence



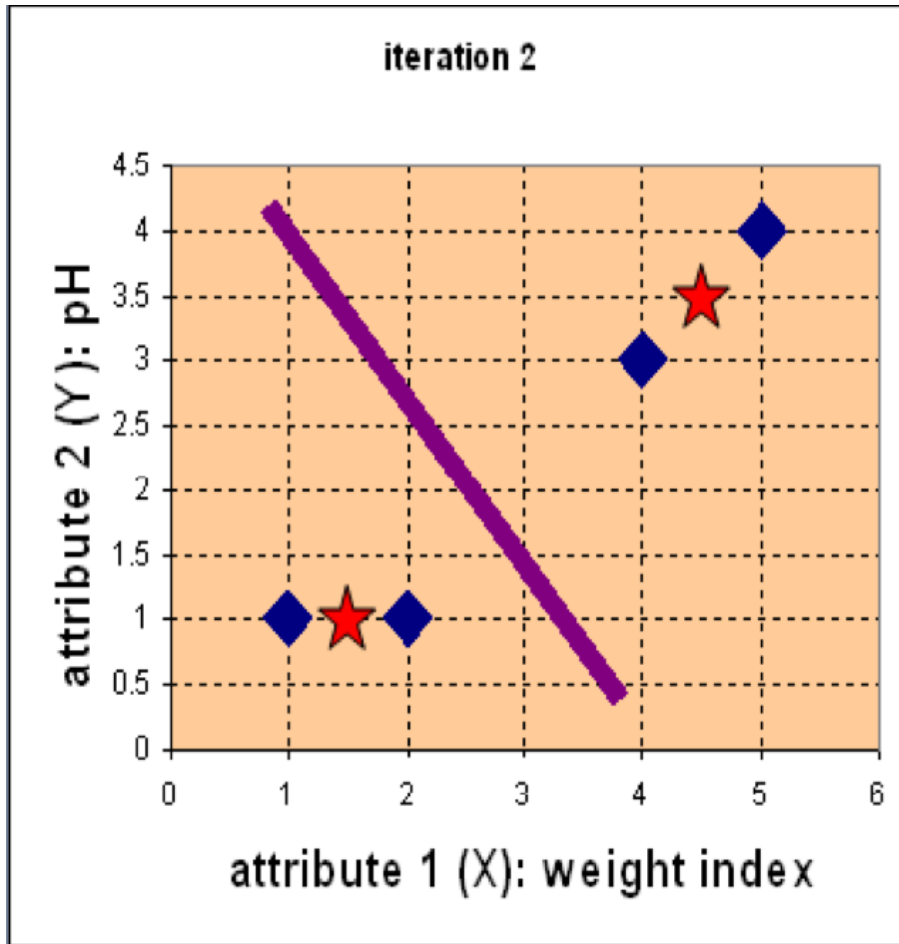
Knowing the members of each cluster, now we compute the new centroid of each group based on these new memberships.

$$c_1 = \left( \frac{1+2}{2}, \frac{1+1}{2} \right) = \left( 1\frac{1}{2}, 1 \right)$$

$$c_2 = \left( \frac{4+5}{2}, \frac{3+4}{2} \right) = \left( 4\frac{1}{2}, 3\frac{1}{2} \right)$$

# A Simple Example

❖ Step 3: Repeat the first two steps until its convergence



Compute the distance of all objects to the new centroids

$$D^2 = \begin{bmatrix} 0.5 & 0.5 & 3.20 & 4.61 \\ 4.30 & 3.54 & 0.71 & 0.71 \end{bmatrix} \quad \begin{array}{l} \mathbf{c}_1 = (1\frac{1}{2}, 1) \text{ group-1} \\ \mathbf{c}_2 = (4\frac{1}{2}, 3\frac{1}{2}) \text{ group-2} \end{array}$$

	A	B	C	D	
$\begin{bmatrix} 1 & 2 & 4 & 5 \end{bmatrix}$	1	2	4	5	X
$\begin{bmatrix} 1 & 1 & 3 & 4 \end{bmatrix}$	1	1	3	4	Y

Stop due to no new assignment

# $k$ -Means Clustering

The  $k$ -means clustering algorithm has been used frequently to find communities in networks in Chapters 3 and 4.  $k$ -means clustering finds clusters of data in attribute format. Given a parameter  $k$ , the  $k$ -means clustering algorithm partitions data instances into  $k$  clusters. Each cluster is associated with a centroid (center point). The centroid of a cluster is typically computed as the mean of the data points assigned to the cluster. And each data point is assigned to the closest cluster, or the distance between its centroid and the data point is the shortest. The closeness of data points to a centroid depends on the definition of distance or the inverse of similarity. In classical  $k$ -means, Euclidean distance is employed and a data instance is assigned to the centroid with minimum distance. As in Section 3.3.1, we can also assign a data instance to a centroid with maximum cosine similarity or Jaccard similarity. The basic  $k$ -means clustering algorithm (Tan et al., 2005) is listed in Figure C.1.

- 
- 1: select  $k$  points as initial centroids;
  - 2: **repeat**
  - 3:   form  $k$  clusters by assigning all points to their closest centroids;
  - 4:   recompute the centroid of each cluster;
  - 5: **until** the centroids do not change.
- 

**Figure C.1:** The  $k$ -Means Clustering Algorithm

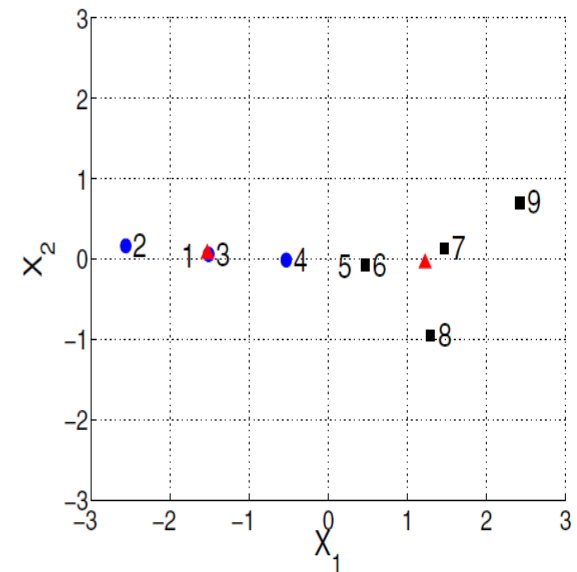
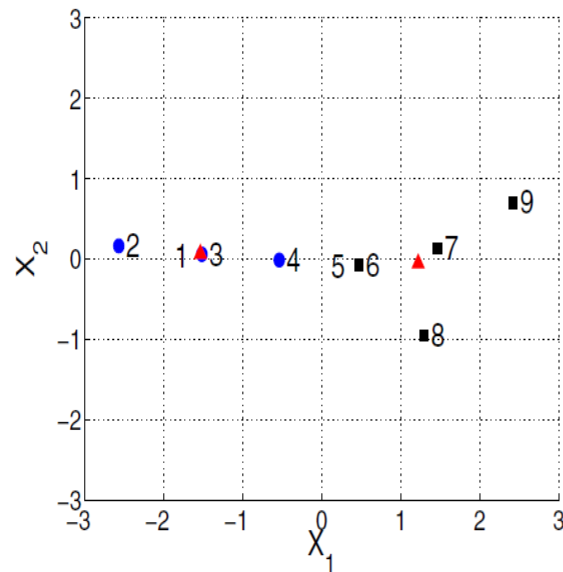
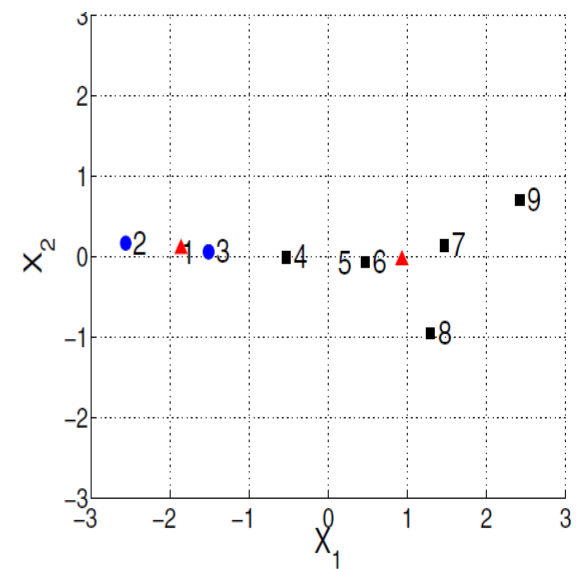
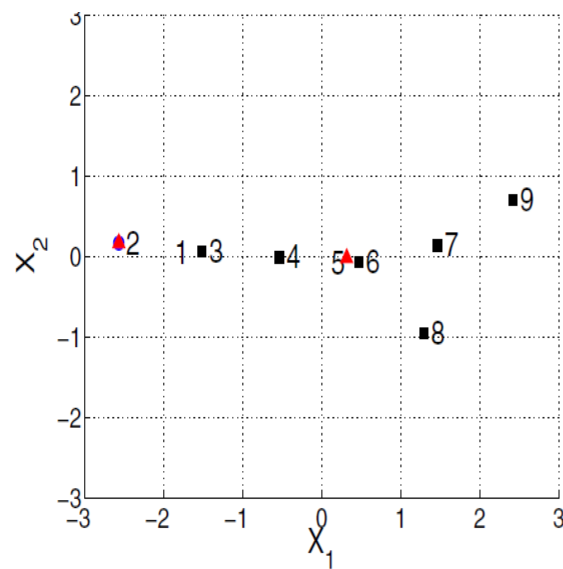


### Figure C.1: The $k$ -Means Clustering Algorithm

Here, we apply  $k$ -means clustering algorithm with  $k = 2$  to a data set  $X$  of 7 instances listed below:

$$X = \begin{array}{c|cc} & X_1 & X_2 \\ \hline 1 & -1.51 & 0.06 \\ 2 & -2.56 & 0.17 \\ 3 & -1.51 & 0.06 \\ 4 & -0.53 & -0.01 \\ 5 & 0.47 & -0.08 \\ 6 & 0.47 & -0.08 \\ 7 & 1.47 & 0.14 \\ 8 & 1.29 & -0.95 \\ 9 & 2.42 & 0.70 \end{array}$$

Each column above indicates one attribute ( $X_1$  and  $X_2$ , respectively). We start by randomly selecting two data points (say, #2 and #4) as the initial centroids:



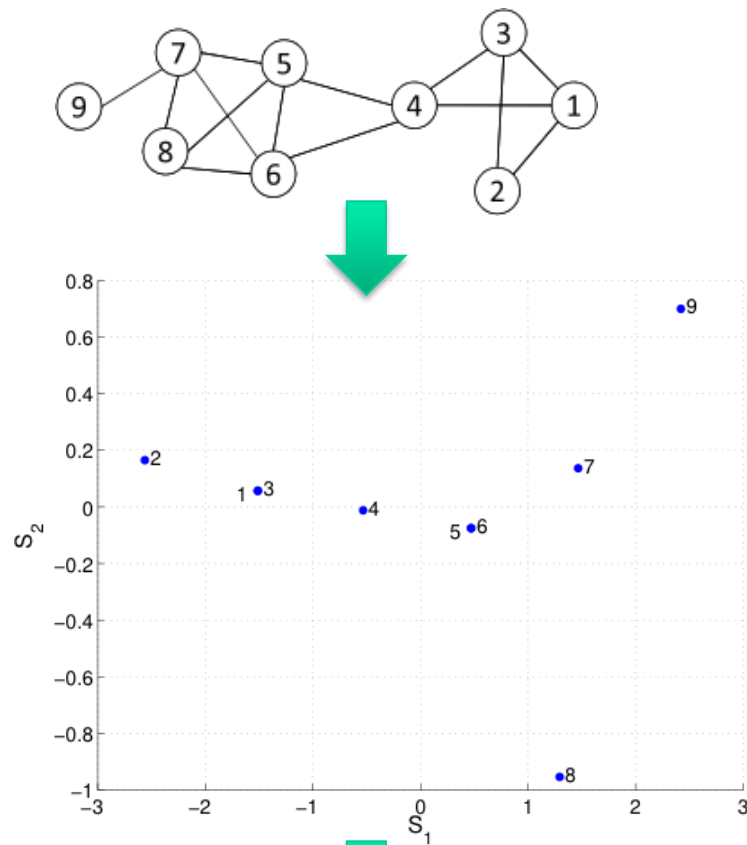
**Figure C.2:** Cluster Assignment and Corresponding Centroids in Each Iteration. Circles and squares each denote a cluster, and triangles are the centroids.

# Latent Space Models

- ❖ Map nodes into a low-dimensional space such that the proximity between nodes based on network connectivity is preserved in the new space, then apply k-means clustering
- ❖ A Representative Approach: **Multi-dimensional scaling (MDS)**
  - ❖ Given a network, construct a proximity matrix  $P$  (on next page) representing the pairwise distance between nodes (e.g., geodesic distance)
  - ❖ Let  $S \in R^{n \times l}$  denote the coordinates of nodes in the low-dimensional space
$$SS^T \approx -\frac{1}{2}(I - \frac{1}{n}\mathbf{1}\mathbf{1}^T)(P \circ P)(I - \frac{1}{n}\mathbf{1}\mathbf{1}^T) = \tilde{P}$$
  - ❖ **Objective function:**  $S = V\Lambda^{\frac{1}{2}}$
  - ❖ **Solution:**  $\min \|SS^T - \tilde{P}\|_F^2$
  - ❖  $V$  is the top  $\ell$  eigenvectors of  $\tilde{P}$ , and  $\Lambda$  is a diagonal matrix of top eigenvalues

$$\Lambda = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_\ell)$$

# Latent Space Models



Two communities:  
 {1, 2, 3, 4} and {5, 6, 7, 8, 9}

geodesic  
distance

$$P = \begin{bmatrix} 0 & 1 & 1 & 1 & 2 & 2 & 3 & 3 & 4 \\ 1 & 0 & 1 & 2 & 3 & 3 & 4 & 4 & 5 \\ 1 & 1 & 0 & 1 & 2 & 2 & 3 & 3 & 4 \\ 1 & 2 & 1 & 0 & 1 & 1 & 2 & 2 & 3 \\ 2 & 3 & 2 & 1 & 0 & 1 & 1 & 1 & 2 \\ 2 & 3 & 2 & 1 & 1 & 0 & 1 & 1 & 2 \\ 3 & 4 & 3 & 2 & 1 & 1 & 0 & 1 & 1 \\ 3 & 4 & 3 & 2 & 1 & 1 & 1 & 0 & 2 \\ 4 & 5 & 4 & 3 & 2 & 2 & 1 & 2 & 0 \end{bmatrix}$$

$$\tilde{P} = \begin{bmatrix} 2.46 & 3.96 & 1.96 & 0.85 & -0.65 & -0.65 & -2.21 & -2.04 & -3.65 \\ 3.96 & 6.46 & 3.96 & 1.35 & -1.15 & -1.15 & -3.71 & -3.54 & -6.15 \\ 1.96 & 3.96 & 2.46 & 0.85 & -0.65 & -0.65 & -2.21 & -2.04 & -3.65 \\ 0.85 & 1.35 & 0.85 & 0.23 & -0.27 & -0.27 & -0.82 & -0.65 & -1.27 \\ -0.65 & -1.15 & -0.65 & -0.27 & 0.23 & -0.27 & 0.68 & 0.85 & 1.23 \\ -0.65 & -1.15 & -0.65 & -0.27 & -0.27 & 0.23 & 0.68 & 0.85 & 1.23 \\ -2.21 & -3.71 & -2.21 & -0.82 & 0.68 & 0.68 & 2.12 & 1.79 & 3.68 \\ -2.04 & -3.54 & -2.04 & -0.65 & 0.85 & 0.85 & 1.79 & 2.46 & 2.35 \\ -3.65 & -6.15 & -3.65 & -1.27 & 1.23 & 1.23 & 3.68 & 2.35 & 6.23 \end{bmatrix}$$

$$V = \begin{bmatrix} -0.33 & 0.05 \\ -0.55 & 0.14 \\ -0.33 & 0.05 \\ -0.11 & -0.01 \\ 0.10 & -0.06 \\ 0.10 & -0.06 \\ 0.32 & 0.11 \\ 0.28 & -0.79 \\ 0.52 & 0.58 \end{bmatrix}, \quad \Lambda = \begin{bmatrix} 21.56 & 0 \\ 0 & 1.46 \end{bmatrix}, \quad S = V \Lambda^{1/2} = \begin{bmatrix} -1.51 & 0.06 \\ -2.56 & 0.17 \\ -1.51 & 0.06 \\ -0.53 & -0.01 \\ 0.47 & -0.08 \\ 0.47 & -0.08 \\ 1.47 & 0.14 \\ 1.29 & -0.95 \\ 2.42 & 0.70 \end{bmatrix}$$

# Block Models

Table 3.1: Adjacency Matrix

-	1	1	1	0	0	0	0	0
1	-	1	0	0	0	0	0	0
1	1	-	1	0	0	0	0	0
1	0	1	-	1	1	0	0	0
0	0	0	1	-	1	1	1	0
0	0	0	1	1	-	1	1	0
0	0	0	0	1	1	-	1	1
0	0	0	0	1	1	1	-	0
0	0	0	0	0	0	1	0	-



$$\min \|A - S\Sigma S^T\|_F^2$$

Table 3.2: Ideal Block Structure

1	1	1	1	0	0	0	0	0
1	1	1	1	0	0	0	0	0
1	1	1	1	0	0	0	0	0
1	1	1	1	0	0	0	0	0
0	0	0	0	1	1	1	1	1
0	0	0	0	1	1	1	1	1
0	0	0	0	1	1	1	1	1
0	0	0	0	1	1	1	1	1
0	0	0	0	1	1	1	1	1

- ❖ S is the community indicator matrix (group memberships)
- ❖ Relax S to be real values, then the optimal solution corresponds to the **top eigenvectors** of A

$$S = \begin{bmatrix} 0.20 & -0.52 \\ 0.11 & -0.43 \\ 0.20 & -0.52 \\ 0.38 & -0.30 \\ 0.47 & 0.15 \\ 0.47 & 0.15 \\ 0.41 & 0.28 \\ 0.38 & 0.24 \\ 0.12 & 0.11 \end{bmatrix}, \Sigma = \begin{bmatrix} 3.5 & 0 \\ 0 & 2.4 \end{bmatrix}.$$



Two communities:  
{1, 2, 3, 4} and {5, 6, 7, 8, 9}

**blockmodel** (*G, partitions, multigraph=False*)

Returns a reduced graph constructed using the generalized block modeling technique.

The blockmodel technique collapses nodes into blocks based on a given partitioning of the node set. Each partition of nodes (block) is represented as a single node in the reduced graph.

Edges between nodes in the block graph are added according to the edges in the original graph. If the parameter *multigraph* is *False* (the default) a single edge is added with a weight equal to the sum of the edge weights between nodes in the original graph. The default is a weight of 1 if weights are not specified. If the parameter *multigraph* is *True* then multiple edges are added each with the edge data from the original graph.

**Parameters** *G* : graph

A networkx Graph or DiGraph

**partitions** : list of lists, or list of sets

The partition of the nodes. Must be non-overlapping.

**multigraph** : bool, optional

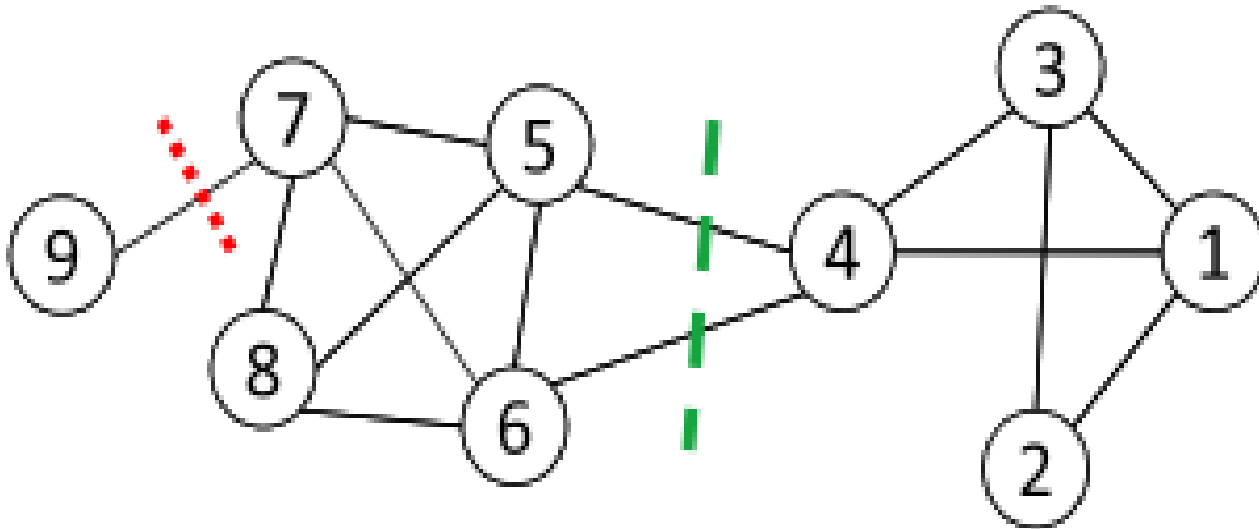
If *True* return a MultiGraph with the edge data of the original graph applied to each corresponding edge in the new graph. If *False* return a Graph with the sum of the edge weights, or a count of the edges if the original graph is unweighted.

**Returns** **blockmodel** : a Networkx graph object

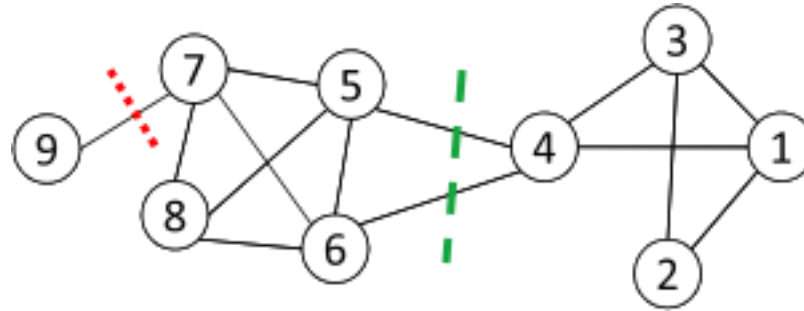
**This function doesn't partition the network for you**

# Spectral Clustering

- ❖ Spectral clustering (Luxburg, 2007) is derived from the problem of graph partition.
- ❖ Graph partition aims to find out a partition such that the **cut** (the total number of edges between two disjoint sets of nodes) is minimized.



# Ratio Cut & Normalized Cut



- ❖ **Minimum cut** often returns an imbalanced partition, with one set being a singleton, e.g. node 9
- ❖ Change the objective function to consider community size

$$\text{Ratio Cut}(\pi) = \frac{1}{k} \sum_{i=1}^k \frac{\text{cut}(C_i, \bar{C}_i)}{|C_i|},$$

$C_i$ : a community

$|C_i|$ : number of nodes in  $C_i$

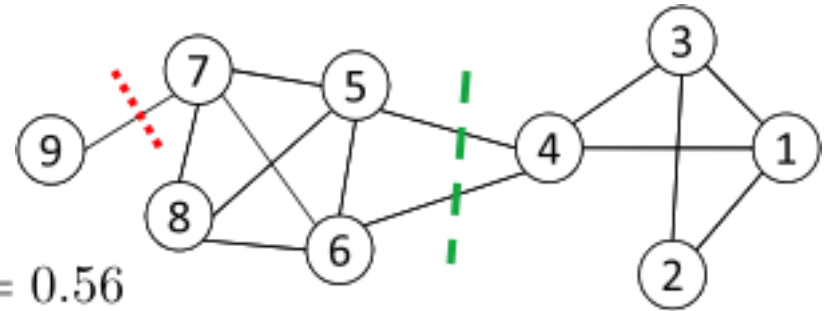
$\text{vol}(C_i)$ : sum of degrees in  $C_i$

$$\text{Normalized Cut}(\pi) = \frac{1}{k} \sum_{i=1}^k \frac{\text{cut}(C_i, \bar{C}_i)}{\text{vol}(C_i)}$$



# Ratio Cut & Normalized Cut Example

For partition in red:  $\pi_1$



$$\text{Ratio Cut}(\pi_1) = \frac{1}{2} \left( \frac{1}{1} + \frac{1}{8} \right) = 9/16 = 0.56$$

$$\text{Normalized Cut}(\pi_1) = \frac{1}{2} \left( \frac{1}{1} + \frac{1}{27} \right) = 14/27 = 0.52$$

For partition in green:  $\pi_2$

$$\text{Ratio Cut}(\pi_2) = \frac{1}{2} \left( \frac{2}{4} + \frac{2}{5} \right) = 9/20 = 0.45 < \text{Ratio Cut}(\pi_1)$$

$$\text{Normalized Cut}(\pi_2) = \frac{1}{2} \left( \frac{2}{12} + \frac{2}{16} \right) = 7/48 = 0.15 < \text{Normalized Cut}(\pi_1)$$

Both ratio cut and normalized cut prefer a balanced partition

# Spectral Clustering

---

- ❖ Both ratio cut and normalized cut can be reformulated as

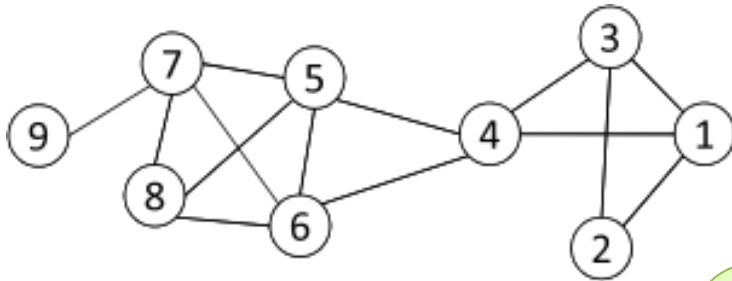
$$\min_{S \in \{0,1\}^{n \times k}} \text{Tr}(S^T \tilde{L} S)$$

where  $\tilde{L} = \begin{cases} D - A & \text{graph Laplacian for ratio cut} \\ I - D^{-1/2} A D^{-1/2} & \text{normalized graph Laplacian} \end{cases}$

$D = \text{diag}(d_1, d_2, \dots, d_n)$  A diagonal matrix of degrees

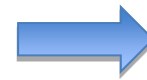
- ❖ Spectral relaxation:  $\min_S \text{Tr}(S^T \tilde{L} S) \quad \text{s.t.} \quad S^T S = I_k$
- ❖ Optimal solution: eigenvectors with the smallest eigenvalues

# Spectral Clustering Example



$$D = \text{diag}(3, 2, 3, 4, 4, 4, 4, 3, 1)$$

$$\tilde{L} = D - A = \begin{bmatrix} 3 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & 3 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & -1 & 4 & -1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 4 & -1 & -1 & -1 & 0 \\ 0 & 0 & 0 & -1 & -1 & 4 & -1 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & -1 & 4 & -1 & -1 \\ 0 & 0 & 0 & 0 & -1 & -1 & -1 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 \end{bmatrix}$$



$$S = \begin{bmatrix} 0.33 & -0.38 \\ 0.33 & -0.48 \\ 0.33 & -0.38 \\ 0.33 & -0.12 \\ 0.33 & 0.16 \\ 0.33 & 0.16 \\ 0.33 & 0.30 \\ 0.33 & 0.24 \\ 0.33 & 0.51 \end{bmatrix}$$

The 1<sup>st</sup> eigenvector means all nodes belong to the same cluster, no use

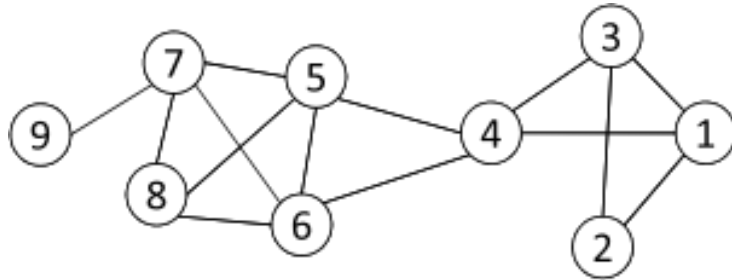


k-means

Two communities:  
{1, 2, 3, 4} and {5, 6, 7, 8, 9}

# Modularity Maximization

- ❖ Modularity measures the strength of a community partition by taking into account the degree distribution
- ❖ Given a network with  $m$  edges, the expected number of edges between two nodes with degrees  $d_i$  and  $d_j$  is  $d_i d_j / 2m$



The expected number of edges between nodes 1 and 2 is  
 $3 * 2 / (2 * 14) = 3/14$

- ❖ Strength of a community:  $\sum_{i \in C, j \in C} A_{ij} - d_i d_j / 2m$
- ❖ Modularity:  $Q = \frac{1}{2m} \sum_{\ell=1}^k \sum_{i \in C_{\ell}, j \in C_{\ell}} (A_{ij} - d_i d_j / 2m)$

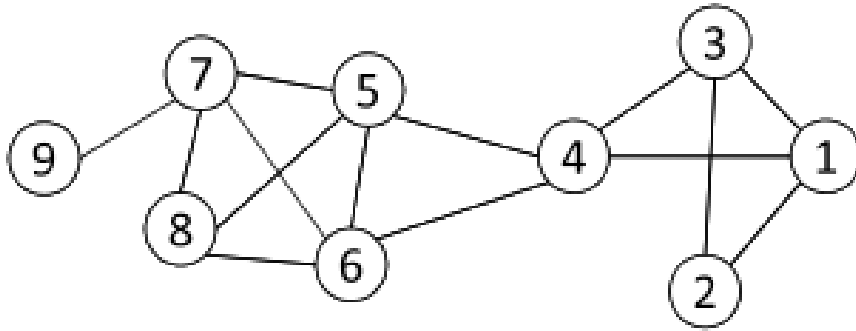
- ❖ A larger value indicates a good community structure

# Modularity Matrix

---

- ❖ Modularity matrix:  $B = A - \mathbf{d}\mathbf{d}^T/2m$  ( $B_{ij} = A_{ij} - d_i d_j / 2m$ )
- ❖ Similar to spectral clustering, modularity maximization can be reformulated as
$$\max Q = \frac{1}{2m} \text{Tr}(S^T B S) \quad s.t. \quad S^T S = I_k$$
- ❖ **Optimal solution: top eigenvectors of the modularity matrix**
- ❖ Apply k-means to S as a post-processing step to obtain community partition

# Modularity Maximization Example



Two Communities:  
 $\{1, 2, 3, 4\}$  and  $\{5, 6, 7, 8, 9\}$

$$B = \begin{bmatrix} -0.32 & 0.79 & 0.68 & 0.57 & -0.43 & -0.43 & -0.43 & -0.32 & -0.11 \\ 0.79 & -0.14 & 0.79 & -0.29 & -0.29 & -0.29 & -0.29 & -0.21 & -0.07 \\ 0.68 & 0.79 & -0.32 & 0.57 & -0.43 & -0.43 & -0.43 & -0.32 & -0.11 \\ 0.57 & -0.29 & 0.57 & -0.57 & 0.43 & 0.43 & -0.57 & -0.43 & -0.14 \\ -0.43 & -0.29 & -0.43 & 0.43 & -0.57 & 0.43 & 0.43 & 0.57 & -0.14 \\ -0.43 & -0.29 & -0.43 & 0.43 & 0.43 & -0.57 & 0.43 & 0.57 & -0.14 \\ -0.43 & -0.29 & -0.43 & -0.57 & 0.43 & 0.43 & -0.57 & 0.57 & 0.86 \\ -0.32 & -0.21 & -0.32 & -0.43 & 0.57 & 0.57 & 0.57 & -0.32 & -0.11 \\ -0.11 & -0.07 & -0.11 & -0.14 & -0.14 & -0.14 & 0.86 & -0.11 & -0.04 \end{bmatrix}$$

Modularity Matrix




$k$ -means

0.4384	-0.2709
0.3809	0.2671
0.4384	-0.2709
0.1716	0.6063
-0.2861	-0.3487
-0.2861	-0.3487
-0.3754	0.3355
-0.3421	0.1855
-0.1396	-0.1552

# Taxonomy of Community Detection

---

- ❖ Roughly, community detection methods can be divided into 4 categories:
  - ❖ **Node-Centric Community**
    - ❖ **Each node** in a group satisfies certain properties
    - ❖ Methods: cliques; k-cliques; k-clubs
  - ❖ **Group-Centric Community**
    - ❖ Consider the connections **within a group** as a whole. The group has to satisfy certain properties without zooming into node-level
    - ❖ Methods: quasi-cliques
  - ❖ **Network-Centric Community**
    - ❖ Partition **the whole network** into several disjoint sets
    - ❖ Clustering based on vertex similarity; latent space models; block models; spectral clustering; modularity maximization
  -  ❖ **Hierarchy-Centric Community**
    - ❖ Construct a **hierarchical structure** of communities
    - ❖ Methods: Divisive clustering; Agglomerative clustering

# Hierarchy-Centric Community Detection

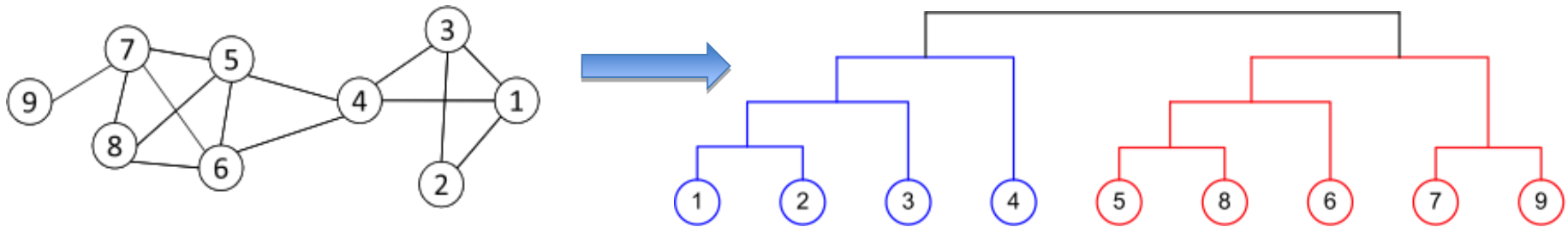
---

- ❖ All methods discussed up till now have considered communities at a single level.
  - ❖ In reality, communities may have hierarchies. Each community can have sub/super communities.
  - ❖ Hierarchical clustering deals with this scenario and generates community hierarchies.
- ❖ Initially  $n$  actors are considered as either  $1$  or  $n$  communities in hierarchical clustering
  - ❖ These communities are gradually
    - ❖ Merged (agglomerative hierarchical clustering algorithms), or
    - ❖ Split (divisive hierarchical clustering algorithms)



# Hierarchy-Centric Community Detection

- ❖ Goal: build a hierarchical structure of communities based on network topology
- ❖ Allow the analysis of a network at different resolutions (levels)
- ❖ Representative approaches:
  - ❖ Divisive Hierarchical Clustering (top-down)
  - ❖ Agglomerative Hierarchical clustering (bottom-up)



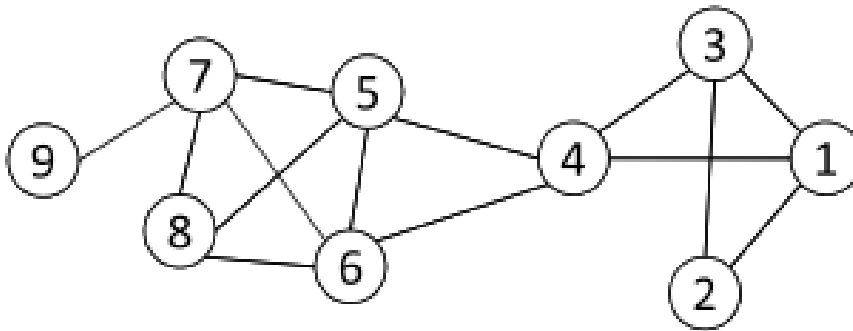
# Divisive Hierarchical Clustering

---

- ❖ Divisive clustering
  - ❖ Partition nodes into several sets
  - ❖ Each set is further divided into smaller ones
- ❖ One particular example: recursively cut the “weakest” link/tie
  1. Find the edge with the least strength
  2. Remove the edge and update the corresponding strength of each edge
- ❖ Recursively apply the above two steps until a network is decomposed into desired number of connected components.
- ❖ Each component forms a community

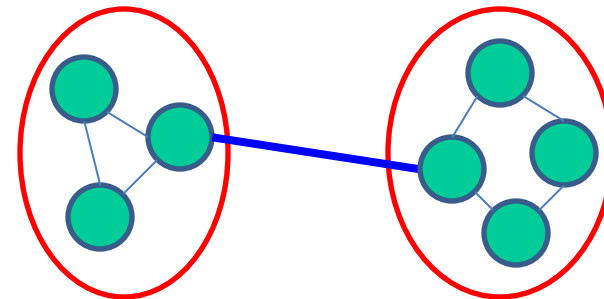
# Edge Betweenness

- ❖ The strength of a tie can be measured by **edge betweenness**
- ❖ **Edge betweenness**: the sum of the fraction of all-pairs' shortest paths that pass through this edge



The edge betweenness of  $e(1, 2)$  is 4 ( $=6/2 + 1$ ), as all the shortest paths from 2 to  $\{4, 5, 6, 7, 8, 9\}$  have to either pass  $e(1, 2)$  or  $e(2, 3)$ , and  $e(1,2)$  is the shortest path between 1 and 2

- ❖ The edge with higher betweenness tends to be the bridge between two communities.



### `networkx.algorithms centrality.edge_betweenness centrality`

**edge\_betweenness centrality**(*G*, *k=None*, *normalized=True*, *weight=None*, *seed=None*)

Compute betweenness centrality for edges.

Betweenness centrality of an edge  $e$  is the sum of the fraction of all-pairs shortest paths that pass through  $e$

$$c_B(e) = \sum_{s,t \in V} \frac{\sigma(s,t|e)}{\sigma(s,t)}$$

where  $V$  is the set of nodes,  $\sigma(s,t)$  is the number of shortest  $(s,t)$ -paths, and  $\sigma(s,t|e)$  is the number of those paths passing through edge  $e$ <sup>2</sup>.

#### Parameters

- **G** (*graph*) – A NetworkX graph.

---

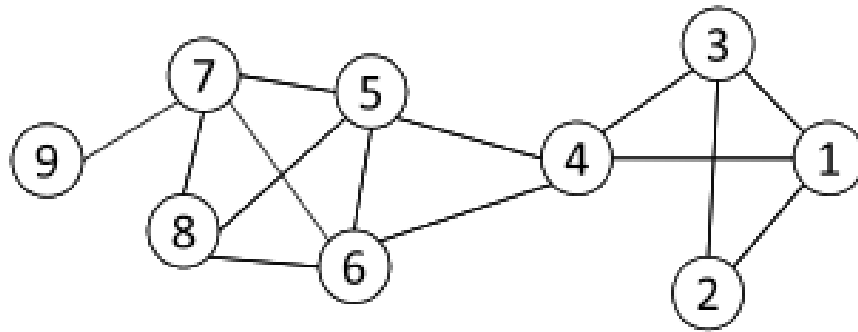
<sup>1</sup> Ulrik Brandes: A Faster Algorithm for Betweenness Centrality. Journal of Mathematical Sociology 25(2):163-177, 2001. <http://www.inf.uni-konstanz.de/algo/publications/b-fabc-01.pdf>

<sup>4</sup> Linton C. Freeman: A set of measures of centrality based on betweenness. Sociometry 40: 35–41, 1977 <http://moreno.ss.uci.edu/23.pdf>

<sup>3</sup> Ulrik Brandes and Christian Pich: Centrality Estimation in Large Networks. International Journal of Bifurcation and Chaos 17(7):2303-2318, 2007. <http://www.inf.uni-konstanz.de/algo/publications/bp-celn-06.pdf>

<sup>2</sup> Ulrik Brandes: On Variants of Shortest-Path Betweenness Centrality and their Generic Computation. Social Networks 30(2):136-145, 2008. <http://www.inf.uni-konstanz.de/algo/publications/b-vspbc-08.pdf>

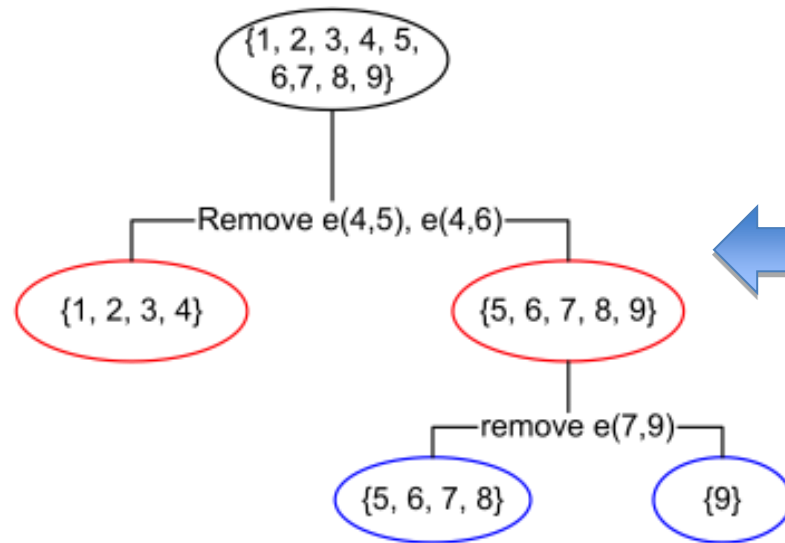
# Clustering Based on Edge Betweenness



Initial betweenness value

Table 3.3: Edge Betweenness

	1	2	3	4	5	6	7	8	9
1	0	4	1	9	0	0	0	0	0
2	4	0	4	0	0	0	0	0	0
3	1	4	0	9	0	0	0	0	0
4	9	0	9	0	10	10	0	0	0
5	0	0	0	10	0	1	6	3	0
6	0	0	0	10	1	0	6	3	0
7	0	0	0	0	6	6	0	2	8
8	0	0	0	0	3	3	2	0	0
9	0	0	0	0	0	0	8	0	0



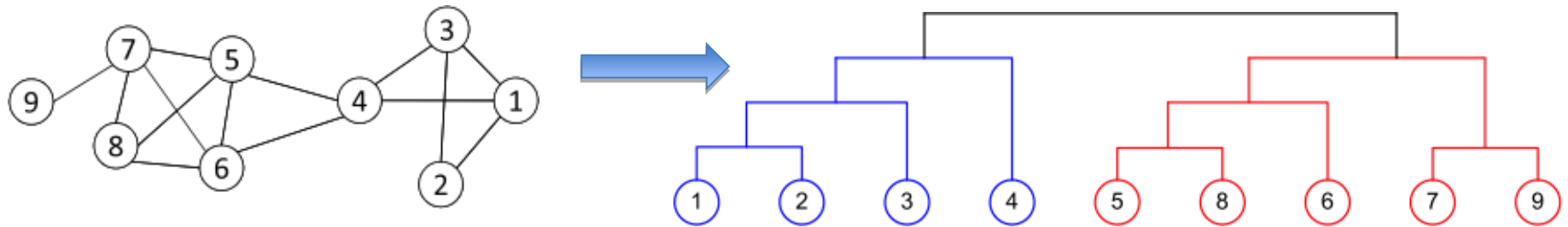
After remove  $e(4,5)$ , the betweenness of  $e(4, 6)$  becomes 20, which is the highest;

After remove  $e(4,6)$ , the edge  $e(7,9)$  has the highest betweenness value 4, and should be removed.

Idea: progressively removing edges with the highest betweenness

# Agglomerative Hierarchical Clustering

- ❖ Initialize each node as a community
- ❖ Merge communities successively into larger communities following a certain criterion (e.g., based on modularity increase)



**Dendrogram** according to Agglomerative Clustering based on Modularity

$$Q = \frac{1}{2m} \sum_{\ell=1}^k \sum_{i \in C_{\ell}, j \in C_{\ell}} (A_{ij} - d_i d_j / 2m)$$

Modularity

Strength of a community

# Agglomerative Hierarchical Clustering

---

- ❖ It is noticed that this algorithm incurs many imbalanced merges (i.e., a large community merges with a tiny community, such as the merge of node 4 with { 1, 2, 3}), resulting in a high computational cost ([Wakita and Tsurumi, 2007](#)).
- ❖ Hence, the merge criterion is modified accordingly by considering the size of communities. In the new scheme, communities of comparable sizes are joined first, leading to a more balanced hierarchical structure of communities and to improved efficiency.
- ❖ A state-of-the-art for such hierarchical clustering is the **Louvain method** ([Blondel et al., 2008](#)).
  - ❖ It starts from each node as a base community and determines which of its neighbor should be merged to based on the modularity criterion.
  - ❖ After the first scan, some base communities are merged into one.
  - ❖ Then, each community's degree and connection information is aggregated and treated as a single node for further merge.
  - ❖ This multi-level approach is extremely efficient to handle large-scale networks.

## Community detection

[home](#) | [search](#) | [documentation](#) »

[previous](#) | [modules](#) | [modules](#) | [index](#)

# community API

This package implements community detection.

---

`community.partition_at_level(dendrogram, level)`

Return the partition of the nodes at the given level

A dendrogram is a tree and each level is a partition of the graph nodes. Level 0 is the first partition, which contains the smallest communities, and the best is `len(dendrogram) - 1`. The higher the level is, the bigger are the communities

**Parameters:** **dendrogram** : list of dict

a list of partitions, ie dictionnaires where keys of the  $i+1$  are the values of the  $i$ .

**level** : int

the level which belongs to  $[0..len(dendrogram)-1]$

**Returns:** **partition** : dictionnary

A dictionary where keys are the nodes and the values are the set it belongs to

**Raises:** **KeyError** :

If the dendrogram is not well formed or the level is too high

### Table Of Contents

[community API](#)  
[Indices and tables](#)

### Previous topic

[Community detection for NetworkX's documentation](#)

### This Page

[Show Source](#)

### Quick search

Enter search terms or a module, class or function name.

See also

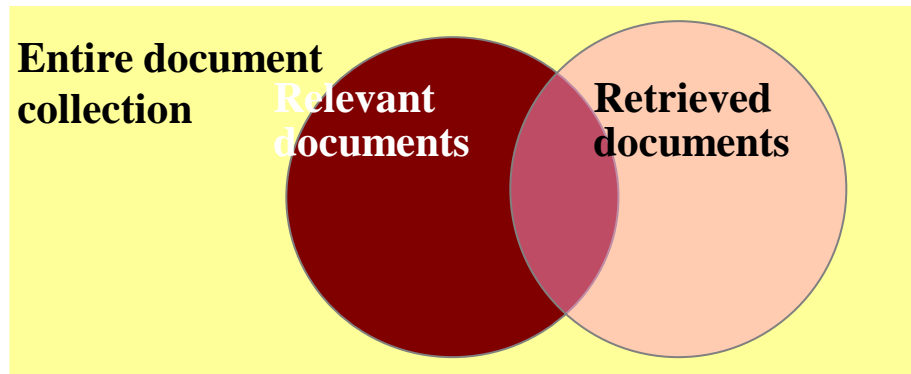


# Evaluating Community Detection

---

- ❖ For groups with clear definitions (easiest case)
  - ❖ E.g., Cliques, k-cliques, k-clubs, quasi-cliques
  - ❖ Verify whether extracted communities satisfy the definition
- ❖ For networks with ground truth information
  - ❖ Precision and Recall
  - ❖ Accuracy of pairwise community memberships
  - ❖ Normalized mutual information

# Precision and Recall: Retrieval



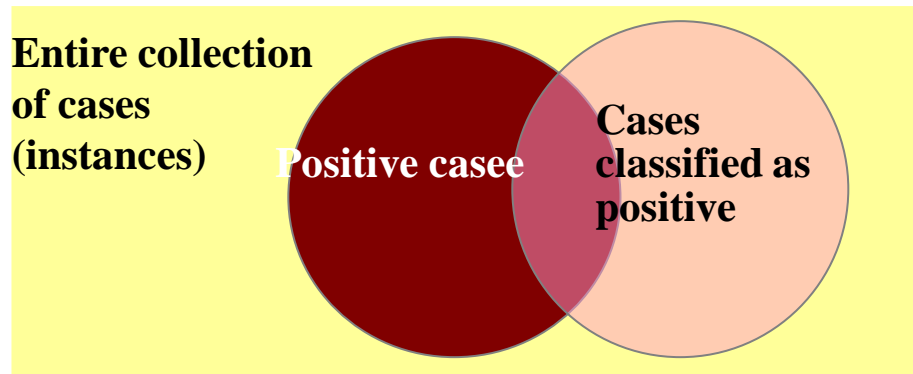
relevant	retrieved & relevant	Not retrieved & relevant
	retrieved & irrelevant	not retrieved but irrelevant
irrelevant	retrieved	not retrieved

$$\text{recall} = \frac{\text{Number of relevant documents retrieved}}{\text{Total number of relevant documents}}$$

$$\text{precision} = \frac{\text{Number of relevant documents retrieved}}{\text{Total number of documents retrieved}}$$

To consolidate Precision & Recall into 1 measure: **F = 2PR/(P+R)**

# Precision and Recall: Classification



Positive	True Positive	False Negative
	False Positive	True Negative
Negative	Classified as Positive	Classified as Negative

$$\text{recall} = \frac{\text{True positive cases}}{\text{Total number of positive cases}}$$

$$\text{precision} = \frac{\text{True positive cases}}{\text{Total number of cases classified as positive}}$$

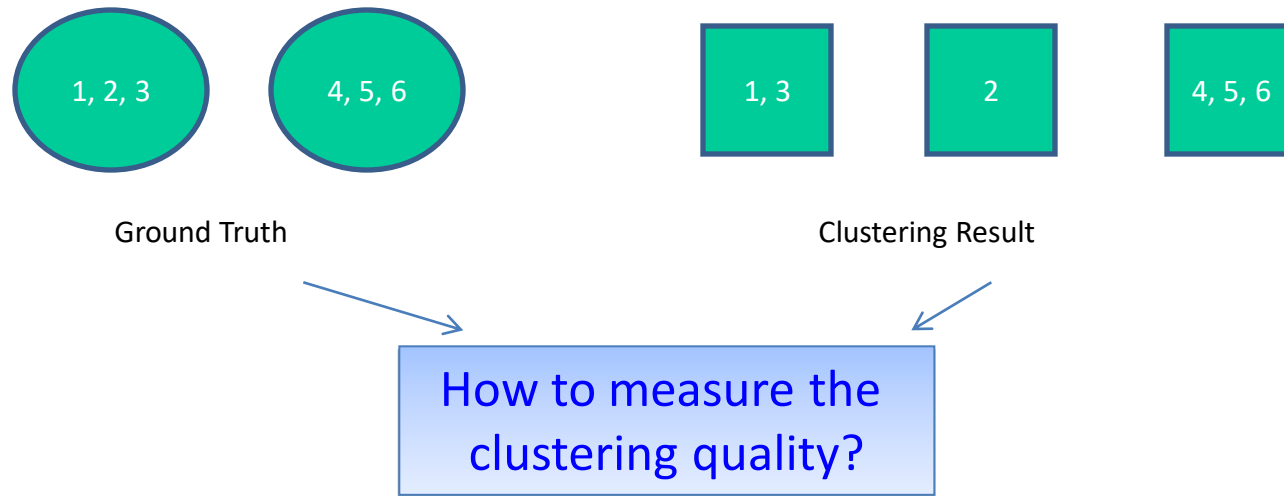
# Precision and Recall: Clustering

$$P = \frac{TP}{TP + FP}$$

$$R = \frac{TP}{TP + FN}$$

- **True Positive (TP) :**
  - when similar points are assigned to the same clusters
  - This is considered a correct decision.
- **True Negative (TN) :**
  - when dissimilar points are assigned to different clusters
  - This is considered a correct decision
- **False Negative (FN) :**
  - when similar points are assigned to different clusters
  - This is considered an incorrect decision
- **False Positive (FP) :**
  - when dissimilar points are assigned to the same clusters
  - This is considered an incorrect decision

# Measuring a Clustering Result



- ❖ The number of communities after grouping can be different from the ground truth
- ❖ No clear community correspondence between clustering result and the ground truth

# Pairwise Community Memberships

- ❖ Consider all the possible pairs of nodes and check whether they reside in the same community
- ❖ An **error** occurs *if*
  - ❖ Two nodes belonging to the **same** community are assigned to **different** communities after clustering
  - ❖ Two nodes belonging to **different** communities are assigned to the **same** community
- ❖ Construct a **contingency table or confusion matrix**

Clustering Result		Ground Truth	
		$C(v_i) = C(v_j)$	$C(v_i) \neq C(v_j)$
	$C(v_i) = C(v_j)$	a	b
	$C(v_i) \neq C(v_j)$	c	d

$$accuracy = \frac{a + d}{a + b + c + d} = \frac{a + d}{n(n - 1)/2}$$

# Accuracy Example



Ground Truth



Clustering Result

		Ground Truth	
		$C(v_i) = C(v_j)$	$C(v_i) \neq C(v_j)$
Clustering Result	$C(v_i) = C(v_j)$	4	0
	$C(v_i) \neq C(v_j)$	2	9

$$\text{Accuracy} = (4+9) / (4+2+9+0) = 13/15$$

# Normalized Mutual Information

---

- ❖ **Entropy**: the information contained in a distribution

$$H(X) = -\sum_{x \in X} p(x) \log p(x)$$

- ❖ **Mutual Information**: the shared information between two distributions

$$I(X; Y) = \sum_{y \in Y} \sum_{x \in X} p(x, y) \log \left( \frac{p(x, y)}{p_1(x)p_2(y)} \right)$$

- ❖ **Normalized Mutual Information** (between 0 and 1)

$$NMI(X; Y) = \frac{I(X; Y)}{\sqrt{H(X)H(Y)}}$$

- ❖ Consider a partition as a distribution (probability of one node falling into one community), we can compute the matching between the **clustering result** and the **ground truth**



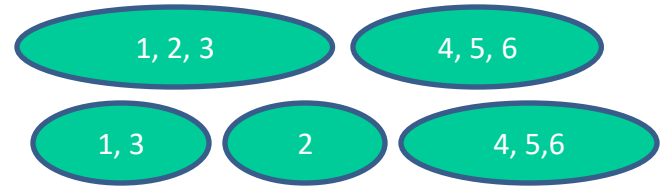
We can consider a partition as a probability distribution of one node falling into one community. Let  $\pi^a, \pi^b$  denote two different partitions of communities.  $n_{h,\ell}, n_h^a, n_\ell^b$  are, respectively, the number of actors simultaneously belonging to the  $h$ -th community of  $\pi^a$  and  $\ell$ -th community of  $\pi^b$ , the number of actors in the  $h$ -th community of partition  $\pi^a$ , and the number of actors in the  $\ell$ -th community of partition  $\pi^b$ . Thus,

$$\begin{aligned}
 H(\pi^a) &= \sum_h^{k^{(a)}} \frac{n_h^a}{n} \log \frac{n_h^a}{n}, \\
 H(\pi^b) &= \sum_\ell^{k^{(b)}} \frac{n_\ell^b}{n} \log \frac{n_\ell^b}{n}, \\
 I(\pi^a; \pi^b) &= \sum_h \sum_\ell \frac{n_{h,\ell}}{n} \log \left( \frac{\frac{n_{h,\ell}}{n}}{\frac{n_h^a}{n} \frac{n_\ell^b}{n}} \right).
 \end{aligned}$$

# NMI-Example

❖ Partition a: [1, 1, 1, 2, 2, 2]

❖ Partition b: [1, 2, 1, 3, 3, 3]



$n = 6$		$n_h^a$		$n_l^b$	$n_{h,l}$	$l=1$	$l=2$	$l=3$
$k^{(a)} = 2$	h=1	3	l=1	2	h=1	2	1	0
$k^{(b)} = 3$	h=2	3	l=2	1	h=2	0	0	3
			l=3	3				

contingency table or confusion matrix

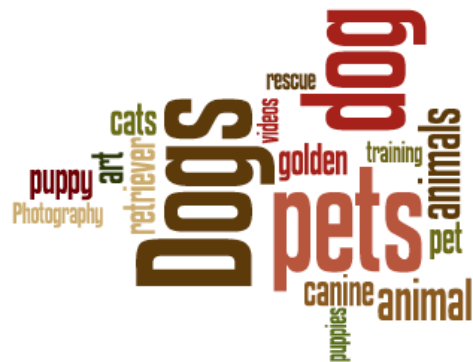
$$NMI(\pi^a, \pi^b) = \frac{\sum_{h=1}^{k^{(a)}} \sum_{\ell=1}^{k^{(b)}} n_{h,\ell} \log \left( \frac{n \cdot n_{h,\ell}}{n_h^{(a)} \cdot n_\ell^{(b)}} \right)}{\sqrt{\left( \sum_{h=1}^{k^{(a)}} n_h^{(a)} \log \frac{n_h^{(a)}}{n} \right) \left( \sum_{\ell=1}^{k^{(b)}} n_\ell^{(b)} \log \frac{n_\ell^{(b)}}{n} \right)}} = 0.8278$$

# Evaluation using Semantics

## ❖ For networks with semantics

- ❖ Networks come with semantic or attribute information of nodes or connections
- ❖ Human subjects can verify whether the extracted communities are coherent
- ❖ Evaluation is **qualitative**
- ❖ It is also intuitive and helps understand a community

An **animal** community



A **health** community



# Evaluation without Ground Truth

---

- ❖ For networks without ground truth or semantic information
- ❖ This is the most common situation
- ❖ An option is to resort to **cross-validation**
  - ❖ Extract communities from a (training) network
  - ❖ Evaluate the quality of the community structure on a network constructed from a different date or based on a related type of interaction
- ❖ Quantitative evaluation functions
  - ❖ Modularity (M. Newman. Modularity and community structure in networks. PNAS 06.)
  - ❖ Link prediction (the predicted network is compared with the true network)



# Community Detection

---

- ❖ Community detection is still an active and evolving field. For a comprehensive survey:
  - ❖ S. Fortunato. **Community detection in graphs.**  
*Physics Reports*, 486(3-5):75 – 174, 2009.
  - ❖ Downloadable from the SU library.

# Assignment 3 (Cancelled)

---

**The Data can be generated by the following Python code:**

```
friends_C, followers_C = get_friends_followers_ids(t,  
    screen_name='CongressListing', friends_limit=10000, followers_limit=0)  
  
friends_D, followers_D = get_friends_followers_ids(t, screen_name='TheDemocrats',  
    friends_limit=10000, followers_limit=0)  
  
DC = [val for val in friends_D if val in friends_C] #filtering  
  
friends_R, followers_R = get_friends_followers_ids(t, screen_name='GOP',  
    friends_limit=10000, followers_limit=0)  
  
RC = [val for val in friends_R if val in friends_C] #filtering  
  
G=nx.Graph()  
G.add_nodes_from(DC)  
G.add_nodes_from(RC)
```

# Assignment 3 (Cancelled)

---

```
# Add all the friends for all the nodes in the combined network
# You will hit the rate limit several times
try:
    for id in G.nodes():
        friends, followers = get_friends_followers_ids(t, user_id=id,
            friends_limit=10000, followers_limit=0)
        for f in friends:
            # only add edges to nodes already in G
            # no need to add_node(); no need to check duplicates
            if f in G.nodes():
                G.add_edge(id, f)
except: # protected users will be skipped
    pass
nx.write_adjlist(G, 'assignment3_2019.adjlist')
```

# Assignment 3 (Cancelled)

---

## Part 1: Find communities:

- ❖ Use one or more of the community detection methods you have learned to divide the entire network into 2 disjoint communities. Feel free to use any other method you have learned from other sources, but remember to cite your sources.
- ❖ Deliverables:
  - ❖ 2 lists of members of those 2 communities. Ids are fine.
  - ❖ A paragraph (at a minimum) explaining why you think your results are good, or are the best you can do



# Assignment 3 (Cancelled)

---

## Part 2: Evaluation:

- ❖ Evaluate your community detection results, namely, your way of dividing the network into 2 disjoint communities, by using one of the following evaluation methods:
  - ❖ The accuracy measure based on pair-wise comparisons
  - ❖ Normalized Mutual Information
- ❖ Deliverables:
  - ❖ Show me the number (and your program)

# Assignment 3 (Cancelled)

---

## Part 3: Find leaders:

- ❖ Find top 5 leaders of each community, using a combination of those 4 centrality measures (degree centrality, closeness centrality, betweenness centrality & eigenvector centrality). Feel free to use any other method you have learned from other sources, but remember to cite your sources.
- ❖ Deliverables:
  - ❖ 2 ranked lists of 5 leaders, one for each community, preferably in real names, but ids are fine
  - ❖ One paragraph (at a minimum) describing why you think your method is good, or is the best you can do
  - ❖ Plus the usual deliverables – code/documentation (for all 3 parts)

# Reading Assignments

---

- ❖ Textbook #2 : Social Media Mining: An Introduction. By Reza Zafarani, Mohammad Ali Abbasi, and Huan Liu, 2015, Cambridge. (Click its link in Blackboard.) **Chapter 6.**
- ❖ Textbook #4: Community Detection and Mining in Social Media, Tang & Liu, 2010, Morgan and Claypool Publishers. (Click its link in Blackboard.) **Chapter 3.**