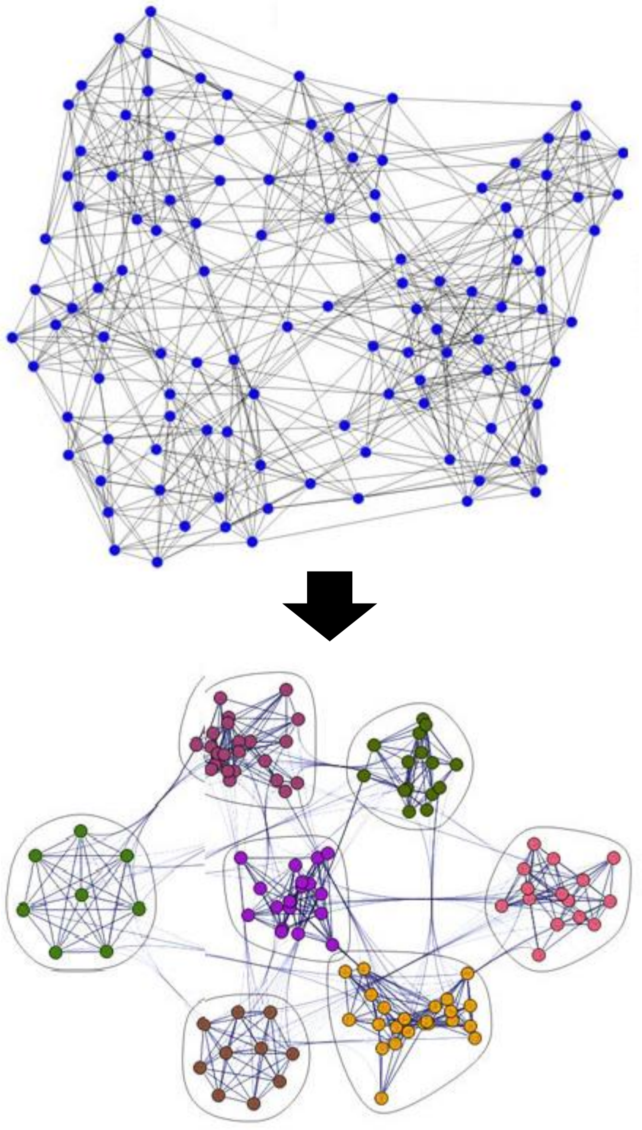
Big Data on Social Media Mining and Analytics -Community Analysis

April 24, 2015



1. How can we detect communities?

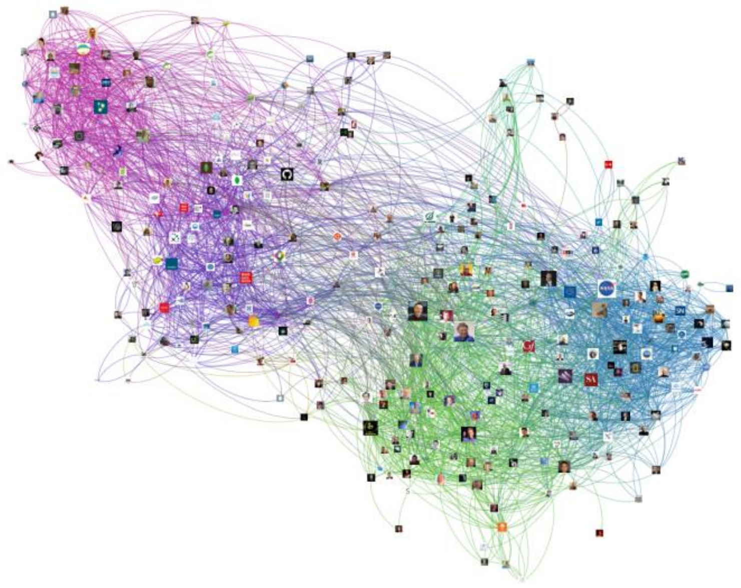
2. How do communities evolve and how can we study evolving community?

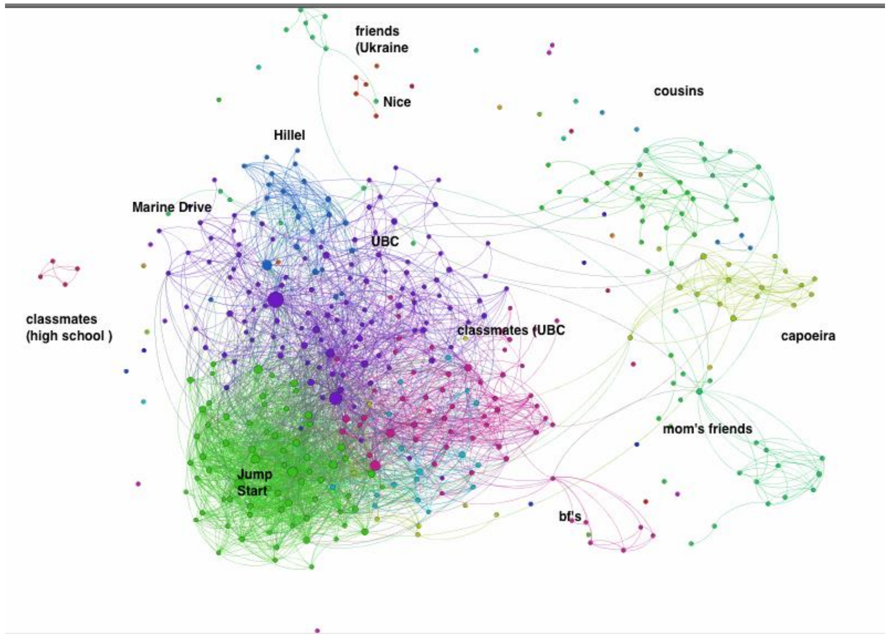
3. How can we evaluate detected communities?

Social Media Communities

Broadly speaking, a real-world community is a body of individuals with common economic, social, or political interests/characteristics, often living in relatively close proximity.

The formation of any community requires (1) a set of at least two nodes sharing some interest and (2) interactions with respect to that interest.





Social Communities

Explicit (emic) **communities**

formed by user subscriptions

Implicit (etic) **communities**

Individuals tacitly interact with others in the form of an unacknowledged community.

We may see *group*, *cluster*, *cohesive subgroup*, or *module* in different contexts instead of “community”

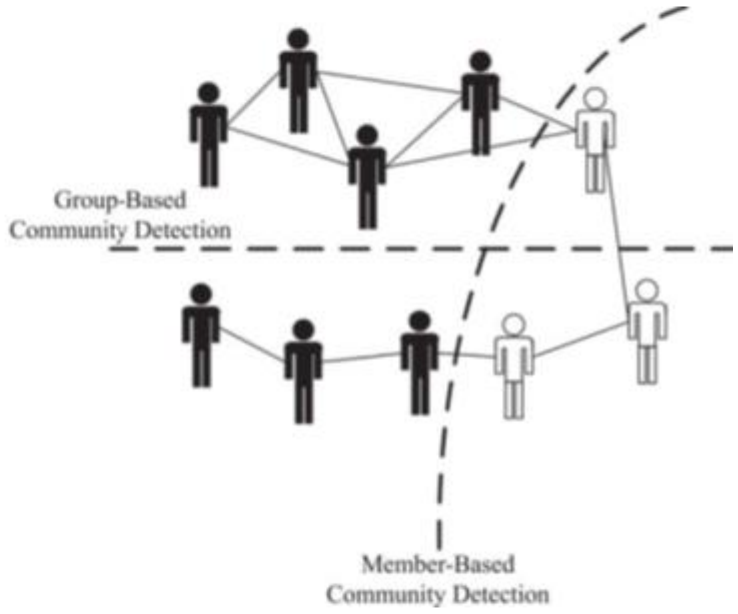
Communities in social media are more or less representatives of communities in the real world.

Community Detection

In contrast to explicit communities, in many social media sites, implicit communities and their members are obscure to many people. Community detection finds these implicit communities.

Formally, for a graph *G(V, E)*, the task of community detection is to find a set of communities $\{C_i\}_{i=1}^{n}$ in a *G*.

Definition



- Community Detection is the process of finding clusters ("communities") of nodes with strong internal connections and weak connections between different clusters
- An ideal decomposition of a large graph is into completely disjoint communities (groups of particles) where there are no interactions between different communities.
- In practice, the task is to find a partition into communities which are maximally decoupled.

Member-Based Community Detection

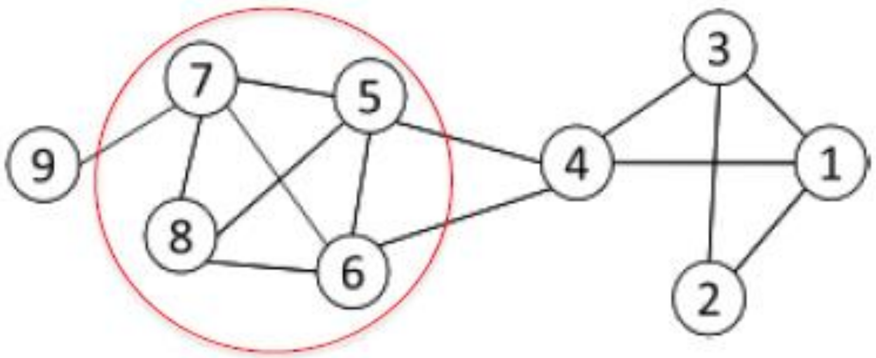
Methods that concentrate on properties of nodes and in most cases assume that nodes with similar characteristics represent a community.

- Node Characteristics:
  - **Degree:** node with same (or similar) degree are in one community
    - cliques
  - **Reachability:** nodes that are close (small shortest path) are in the same community
    - k-clique, k-club, and k-clan
  - **Similarity:** similar nodes are in the same community

Node-Degree

The most common subgraph searched for in networks based on *node degrees* is a ***clique***.

Clique: a maximum complete subgraph in which all pairs of nodes inside the subgraph are connected.



Nodes 5, 6, 7 and 8 form a clique

Node-Degree

To find communities, we can search for the maximum clique (the one with the largest number of vertices) or for all maximal cliques (cliques that are not subgraphs of a larger clique; i.e., cannot be expanded further).

However, both problems are NP-hard, as is verifying whether a graph contains a clique larger than size k.

To overcome these theoretical barriers,

(1) use brute force

(2) add some constraints such that the problem is relaxed and polynomially solvable.

Relaxed: k-plex; $d_v \geq |V| - k, \forall v \in V,$

(3) use cliques as the seed or core of a larger community.

CPM is a method to find communities with overlap

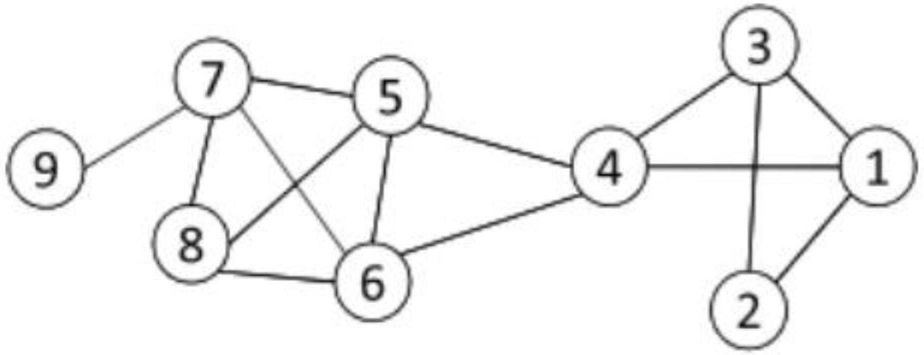
Clique Percolation Method (CPM): Algorithm

When using cliques as a seed or core of a community, we assume communities are formed from a set of cliques (small or large) in addition to edges that connect these cliques.

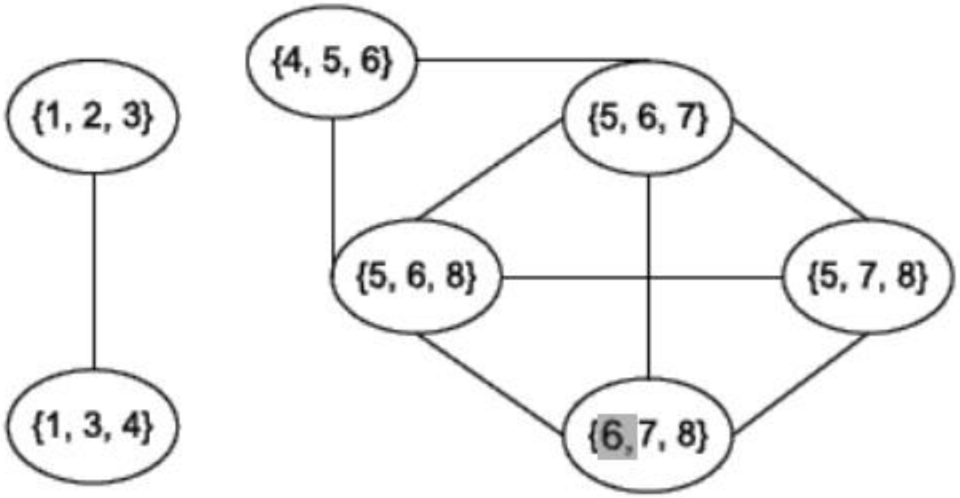
- **Input**
  - A parameter k, and a network

- **Procedure**
  - Find out all cliques of size k in the given network
  - Construct a clique graph where all cliques are represented as nodes
    - Two cliques are adjacent if they share ***k-1*** nodes
  - Each connected components in the clique graph form a community

CPM



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}



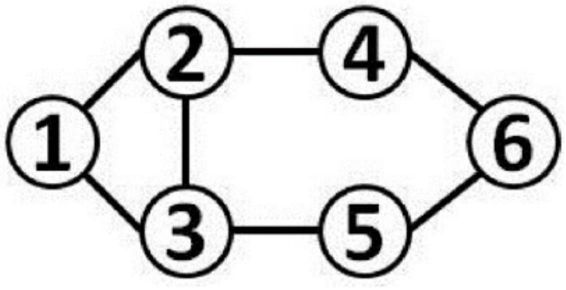
Communities:
{1, 2, 3, 4}
{4, 5, 6, 7, 8}

Node-Reachability

Any node in a group should be reachable in k hops

•k-**Clique**: a **maximal** subgraph in which the shortest path between any nodes <= k.
•k-**Club**: it follows the same definition as k-clique with an additional constraint that nodes on the shortest paths *should be part of the subgraph*.
•k-**Clans:** it is a k-clique where for all shortest paths within the subgraph the distance is equal or less than k. All k-clans are k-cliques, but not vice versa.

$$k\text{-Clans} = k\text{-Cliques} \cap k\text{-Clubs}.$$

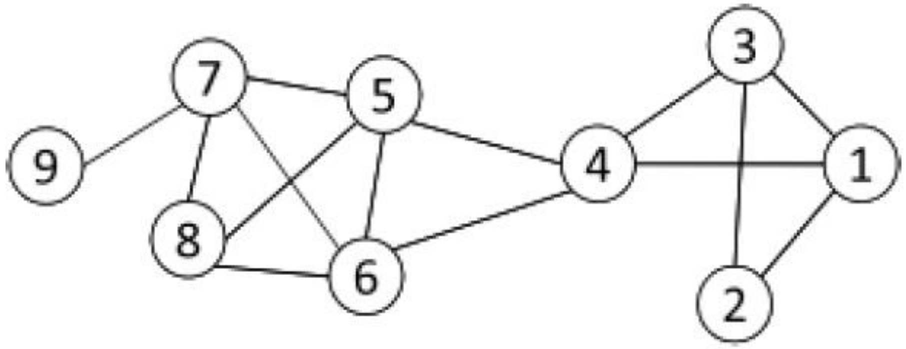


Cliques: {1, 2, 3}
2-cliques: {1, 2, 3, 4, 5}, {2, 3, 4, 5, 6}
2-clubs: {1,2,3,4}, {1, 2, 3, 5}, {2, 3, 4, 5, 6}
2-clans: {2, 3, 4, 5, 6}

Node Similarity

- Apply k-means or similarity-based clustering to nodes
- Vertex similarity is defined in terms of the similarity of their neighborhood
- **Structural equivalence**: two nodes are structurally equivalent if they are connecting to the same set of actors

Nodes 1 and 3 are structurally equivalent, So are nodes 5 and 7.



- Structural equivalence is too restrict for practical use.

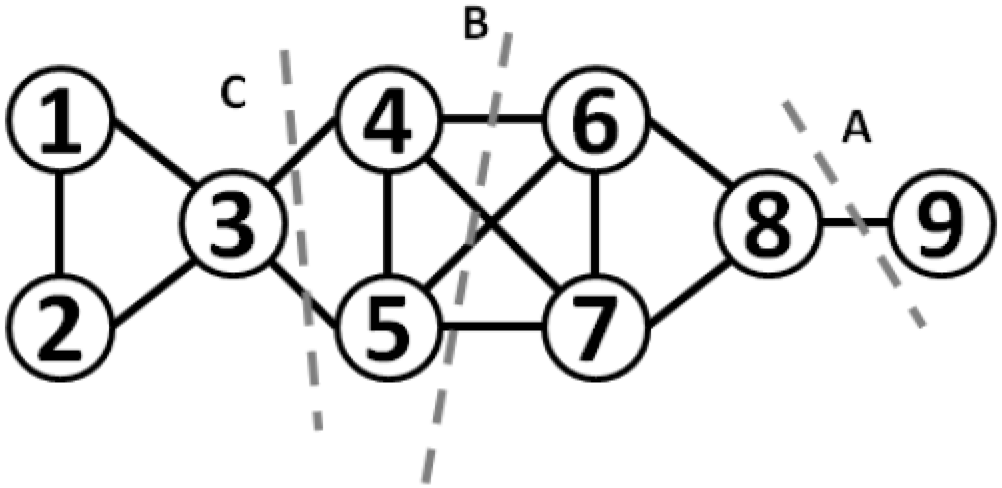
Group-Based Community Detection

- In group-based community detection, the global network information and topology is considered to determine communities

- We search for communities that are:
  - –**Balanced-**> *spectral clustering*
  - –**Modular-**> *modularity maximization*
  - –**Hierarchical-**> *hierarchical clustering / Girvan-Newman algorithm*
  - –**Dense-**> Quasi-cliques
  - –**Robust-**> *k-connected graphs*

Balanced Communities: Spectral Clustering: Cut

- Most interactions are within group whereas interactions between groups are few
- Cut: A partition of vertices of a graph into two disjoint sets
- Minimum cut problem: find a graph partition such that the number of edges between the two sets is minimized
- Community detection ->minimum cut problem



Minimum Cut (A) and Two More Balanced Cuts (B and C) in a Graph.

Ratio Cut & Normalized Cut

- Minimum cut often returns an imbalanced partition, with one set being a singleton
- Change the objective function to consider community size

Consider a graph $G(V, E)$. A partitioning of G into k partitions is a tuple $P = (P_1, P_2, P_3, \ldots, P_k)$, such that $P_i \subseteq V$, $P_i \cap P_j = \emptyset$ and $\bigcup_{i=1}^{k} P_i = V$.

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

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

where $\bar{P}_i = V - P_i$ is the complement cut set, $cut(P_i, \bar{P}_i)$ is the size of the cut, and volume $vol(P_i) = \sum_{v \in P_i} d_v$.

Both objective functions provide a more balanced community size by normalizing the cut size either by the number of vertices in the cutset or the volume (total degree).

Spectral Clustering

Both ratio cut and normalized cut can be reformulated as

$$\min_{X \in \{0,1\}^{|V| \times k}} \operatorname{Tr}(X^R L X)$$

$$L = \begin{cases} D - A & \text{Ratio Cut Laplacian, i.e., Unnormalized Laplacian} \\ I - D^{-1/2} A D^{-1/2} & \text{Normalized Laplacian for Normalized Cut.} \end{cases}$$

$D = diag(d_1, d_2, \cdots, d_n)$ Represents diagonal degree matrix

It has been shown that both ratio cut and normalized cut minimization are NP-hard; therefore, approximation algorithms using relaxations are desired.

Spectral clustering relaxation:

$$\min_X \operatorname{Tr}(X^R L X),$$
$$s.t.\ \ X^T X = I_k$$

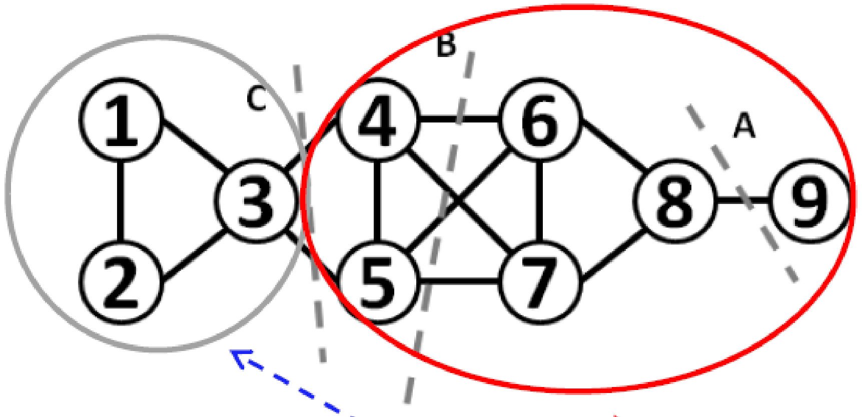
Optimal solution:

Top eigenvectors with the smallest eigenvalues

Given ***L***, the top ***k*** eigenvectors corresponding to the smallest eigen values are computed and used as X, and then k-means is run on X to extract communities memberships (X). and then *k-means* is run on X to extract communities memberships (X).

Spectral Clustering: Example

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



D = diag(2, 2, 4, 4, 4, 4, 4, 3, 1)

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

Eigenvectors

The second is used with k-means;

$$\begin{bmatrix} 0.33 & -0.46 \\ 0.33 & -0.46 \\ 0.33 & -0.26 \\ 0.33 & 1.16 \times 10^{-16} \\ 0.33 & 1.16 \times 10^{-16} \\ 0.33 & 0.13 \\ 0.33 & 0.13 \\ 0.33 & 0.33 \\ 0.33 & 0.59 \end{bmatrix}$$

Modular Communities: Modularity Maxmization

Modularity is a measure that defines how likely the community structure found is created at random.

- Consider a undirected graph *G(V, E)*, |E| = m where the degrees are known beforehand however edges are not
  - – Consider two vertices *vi* and *vj* with degrees *di* and *dj*.
- Now what is an expected number of edges between these two nodes?
- For any edge going out of *vi* randomly the probability of this edge getting connected to vertex *vj* is

$$\frac{d_j}{\sum_i d_i} = \frac{d_j}{2m}$$

Because the degree for v_i is d_i, we have d_i number of such edges

The expected number of edges between v_i and v_j is $\frac{d_i d_j}{2m}$.

Modularity Maximization: Main Idea

- Given a degree distribution, we know the expected number of edges between any pairs of vertices
- We assume that real-world networks should be far from random. Therefore, the more distant they are from this randomly generated network, the more structural they are.
- Modularity defines this distance and modularity maximization tries to maximize this distance

Consider a partitioning of the graph G into *k* partitions, P = (P$_1$, P$_2$, P$_3$, …, P$_k$).

For partition P_x, this distance can be defined as

$$\sum_{v_i, v_j \in P_x} A_{ij} - \frac{d_i d_j}{2m}.$$

This distance can be generalized for a partitioning P with k partitions

$$\sum_{x=1}^{k} \sum_{v_i, v_j \in P_x} A_{ij} - \frac{d_i d_j}{2m}.$$

The normalized version of this distance is defined as Modularity

$$Q = \frac{1}{2m}\left(\sum_{x=1}^{k} \sum_{v_i, v_j \in P_x} A_{ij} - \frac{d_i d_j}{2m} \right).$$

Modularity Maximization

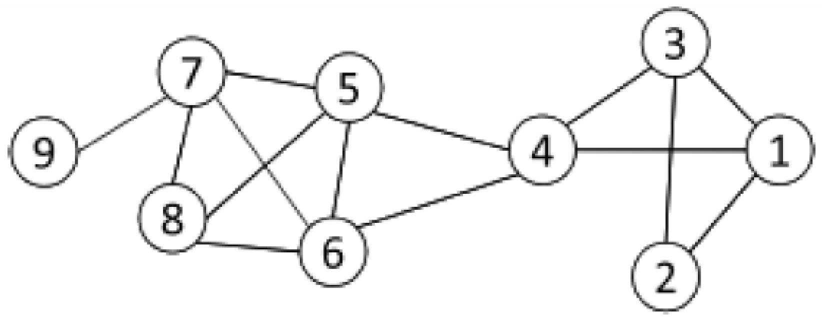
Modularity matrix $B = A - dd^T/2m$

$d \in R^{n*1}$ is the degree vector for all nodes

Reformulation of the modularity $Q = \frac{1}{2m}\mathrm{Tr}(X^T B X)$

where $X \in \mathbb{R}^{n \times k}$ is the indicator (partition membership) function; that is, $X_{ij} = 1$ iff. $v_i \in P_j$. This objective can be maximized such that the best membership function is extracted with respect to modularity. The problem is NP-hard; therefore, we relax X to $\hat{X}$ that has an orthogonal structure ($\hat{X}^T \hat{X} = I_k$). The optimal $\hat{X}$ can be computed using the top k eigenvectors of B corresponding to the largest positive eigenvalues. Similar to spectral clustering, to find X, we can run k-means on $\hat{X}$. Note that this requires that B has at least k positive eigenvalues.

Modularity Maximization: Example



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

$$B = A - \mathbf{d}\mathbf{d}^T/2m \quad (B_{ij} = A_{ij} - d_i d_j/2m)$$

k-means

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

Modularity Matrix

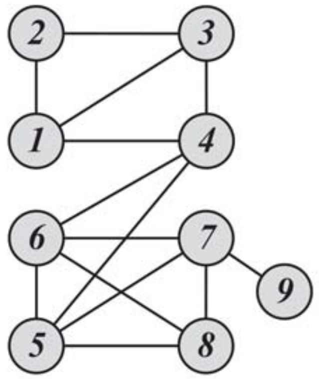
$$S = \begin{bmatrix} 0.44 & -0.00 \\ 0.38 & 0.23 \\ 0.44 & -0.00 \\ 0.17 & -0.48 \\ -0.29 & -0.32 \\ -0.29 & -0.32 \\ -0.38 & 0.34 \\ -0.34 & -0.08 \\ -0.14 & 0.63 \end{bmatrix}$$

Hierarchical Communities

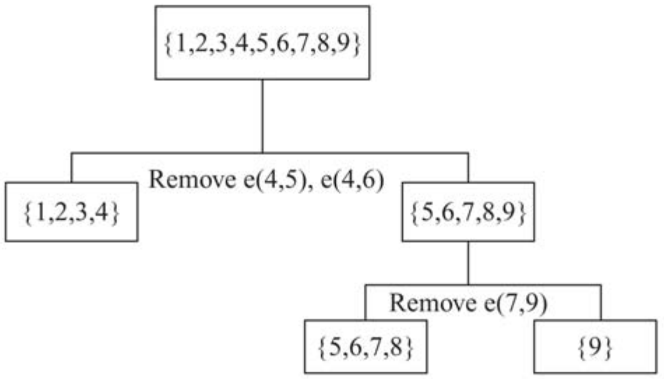
It is common to have hierarchies of communities, in which each community can have sub/super communities. Hierarchical clustering deals with this scenario and generates community hierarchies.

The Girvan-Newman algorithm is specifically designed for finding communities using divisive hierarchical clustering.

1. Calculate ***edge betweenness*** for all edges in the graph.
2. Remove the edge with the highest betweenness.
3. Recalculate betweenness for all edges aected by the edge removal.
4. Repeat until all edges are removed.



(a) Graph



(b) Dendrogram

For an edge e, ***edge betweenness*** is defined as the number of shortest paths between node pairs (v_i, v_j) such that the shortest path between v_i and v_j passes through e.

Community Evolution

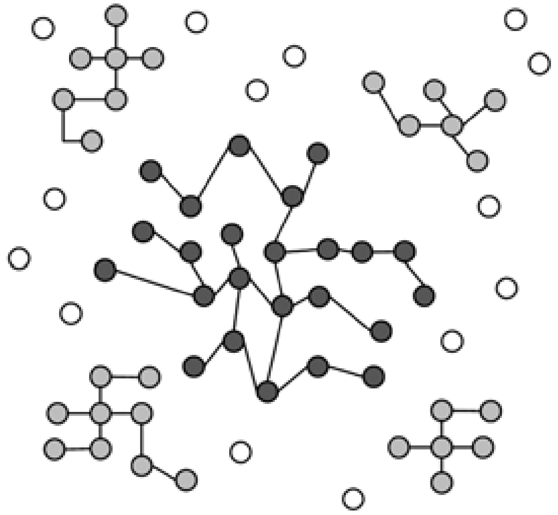
- How networks evolve in general?
- How communities evolve over time?
- How communities can be found in these evolving networks?

How Networks Evolve

We discuss three common patterns that are observed in evolving networks: *segmentation*, *densification*, and *diameter shrinkage*.

Segmentation

1. **Giant Component**: As network connections stabilize, a giant component of nodes is formed, with a large proportion of network nodes and edges falling into this component.
2. **Stars**: These are isolated parts of the network that form star structures. A star is a tree with one internal node and n leaves.
3. **Singletons**: These are orphan nodes disconnected from all nodes in the network.



How Networks Evolve

Graph Densification

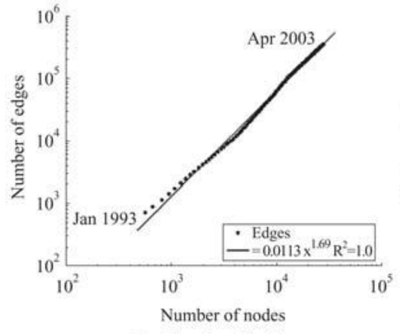
It is observed in evolving graphs that the density of the graph increases as the network grows. In other words, the number of edges increases faster than the number of nodes. This phenomenon is called ***densification***.

Let *V(t)* denote nodes at time *t* and let *E(t)* denote edges at time *t*,

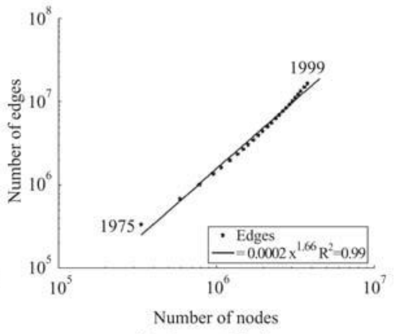
$$|E(t)| \propto |V(t)|^{\alpha}$$

If densification happens, then we have 1 <= α <= 2.

There is linear growth when α = 1, and we get clique structures when α = 2. Networks exhibit values between 1 and 2 when evolving.



(a) Physics Citations



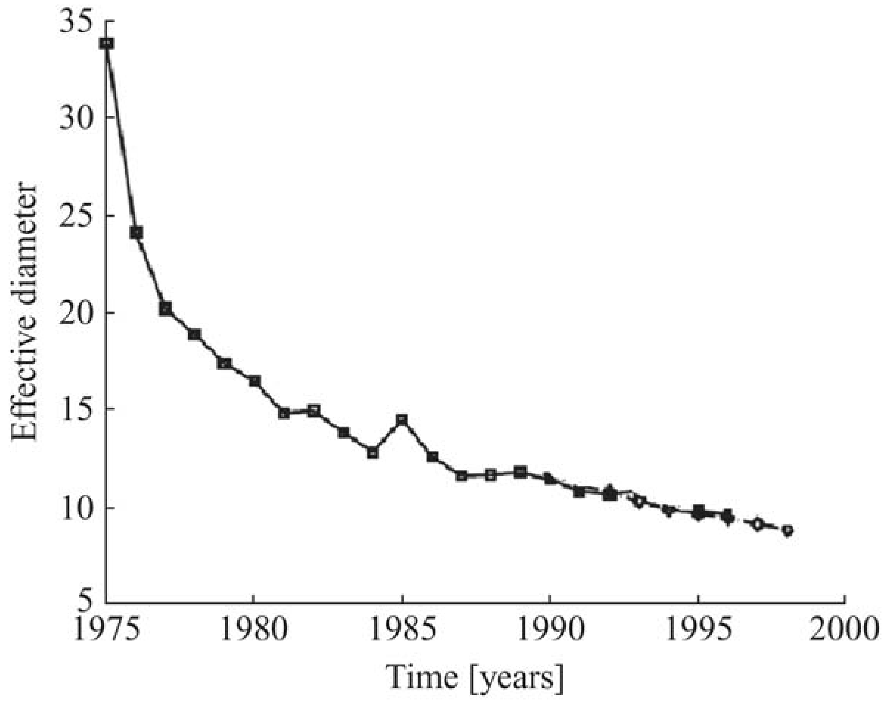
(b) Patent Citations

These recordings show that both networks have α ≈1.6

This value also implies that when V is given, to realistically model a social network, we should generate $O(|V|^{1.6})$ edges.

How Networks

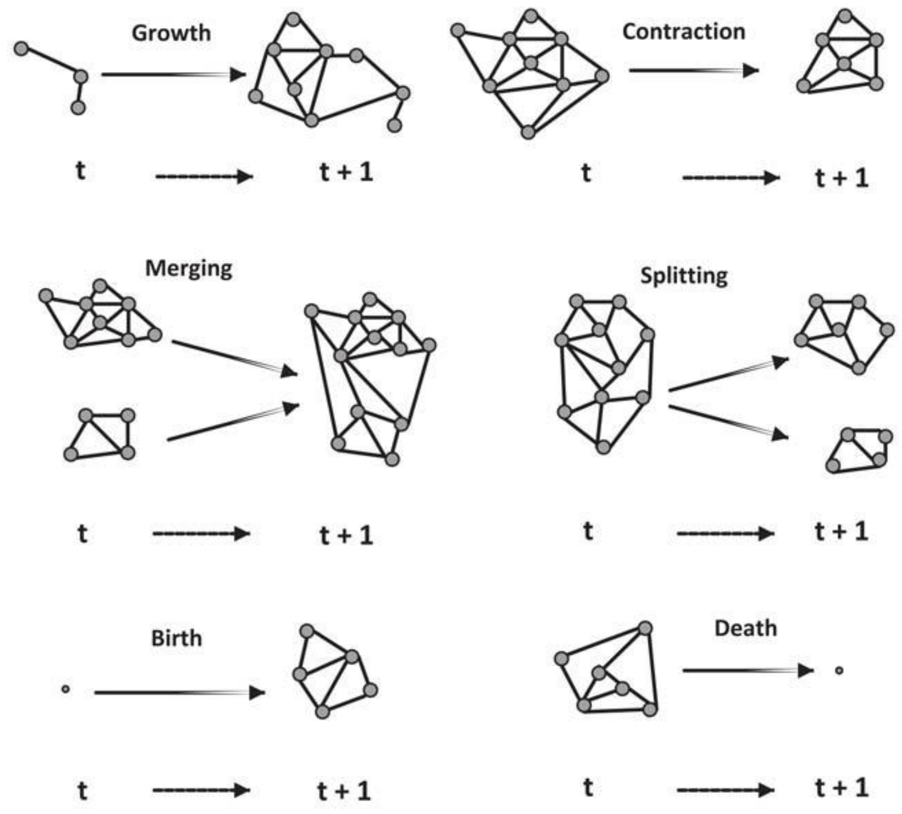
Evolve Diameter Shrinkage



Diameter Shrinkage over Time for a Patent Citation Network

Community Detection in Evolving Networks

We discussed three phenomena that are observed in evolving networks. Communities in evolving networks also evolve. They appear, grow, shrink, split, merge, or even dissolve over time. Both networks and their internal communities evolve over time.



Given evolution information (e.g., when edges or nodes are added),

- How can we study evolving communities?
- Can we adapt static (nontemporal) methods to use this temporal information?

Community Detection in Evolving Networks

In this section, we present community detection algorithms that incorporate temporal information. To incorporate temporal information, we can extend previously discussed static methods as follows:

1. Take *t* snapshots of the network, $G_1, G_2, \ldots, G_t$, where G_i is a snapshot at time *i*.
2. Perform a static community detection algorithm on all snapshots independently.
3. Assign community members based on communities found in all *t* different time stamps.

Unfortunately, this method is unstable in highly dynamic networks because community memberships are always changing. An alternative is to use ***evolutionary clustering****.*

Evolutionary Clustering

In evolutionary clustering, it is assumed that communities do not change most of the time; hence, it tries to minimize an objective function that considers both

- communities at different time stamps (snapshot cost or SC)
- how they evolve throughout time (temporal cost or TC)

$$Cost = \alpha\, SC + (1-\alpha)\, TC, \quad \text{where } 0 \le \alpha \le 1.$$

We know that the objective for spectral clustering is $Tr(X^T L X)$ s.t. $X^T X = I_m$, so we will have the objective function at time *t* as

$$\begin{aligned} Cost_t &= \alpha\, SC + (1-\alpha)\, TC, \\ &= \alpha\, Tr(X_t^T L X_t) + (1-\alpha)\, TC, \end{aligned}$$

where X_t is the community membership matrix at time *t*.

TC is defined as

$$TC = Tr(I - X_t^T X_{t\ 1} X_{t-1}^T X_t),$$

$$\begin{aligned} Cost_t &= \alpha\, Tr(X_t^T L X_t) + (1-\alpha)\, Tr(I - X_t^T X_{t-1} X_{t-1}^T X_t), \\ &= \alpha\, Tr(X_t^T L X_t) + (1-\alpha)\, Tr(X_t^T I X_t - X_t^T X_{t-1} X_{t-1}^T X_t), \\ &= Tr(X_t^T \alpha L X_t) + Tr(X_t^T (1-\alpha) I X_t - X_t^T (1-\alpha) X_{t-1} X_{t-1}^T X_t). \end{aligned}$$

Evolutionary Clustering

Assuming the normalized Laplacian is used in spectral clustering, $L = I - D_t^{-1/2} A_t D_t^{-1/2}$,

$$\begin{aligned} Cost_t &= Tr(X_t^T \alpha (I - D_t^{-1/2} A_t D_t^{-1/2})\, X_t) \\ &\quad + Tr(X_t^T\, (1-\alpha)\, I\, X_t - X_t^T\, (1-\alpha)\, X_{t-1}\, X_{t-1}^T\, X_t), \\ &= Tr(X_t^T\, (I - \alpha D_t^{-1/2} A_t D_t^{-1/2} - (1-\alpha)\, X_{t-1} X_{t-1}^T)\, X_t), \\ &= Tr(X_t \hat{L} X_t), \end{aligned} \tag{6.33}$$

where $\hat{L} = I - \alpha D_t^{-1/2} A_t D_t^{-1/2} - (1-\alpha) X_{t-1} X_{t-1}^T$. Similar to spectral clustering, X_t can be obtained by taking the top eigenvectors of $\hat{L}$.

Note that at time t, we can obtain X_t directly by solving spectral clustering for the laplacian of the graph at time t, but then we are not employing any temporal information. Using evolutionary clustering and the new laplacian $\hat{L}$, we incorporate temporal information into our community detection algorithm and disallow user memberships in communities at time t: X_t to change dramatically from X_{t-1}.

Community Evaluation

In terms of evaluating communities, the task is similar to evaluating clustering methods in data mining.

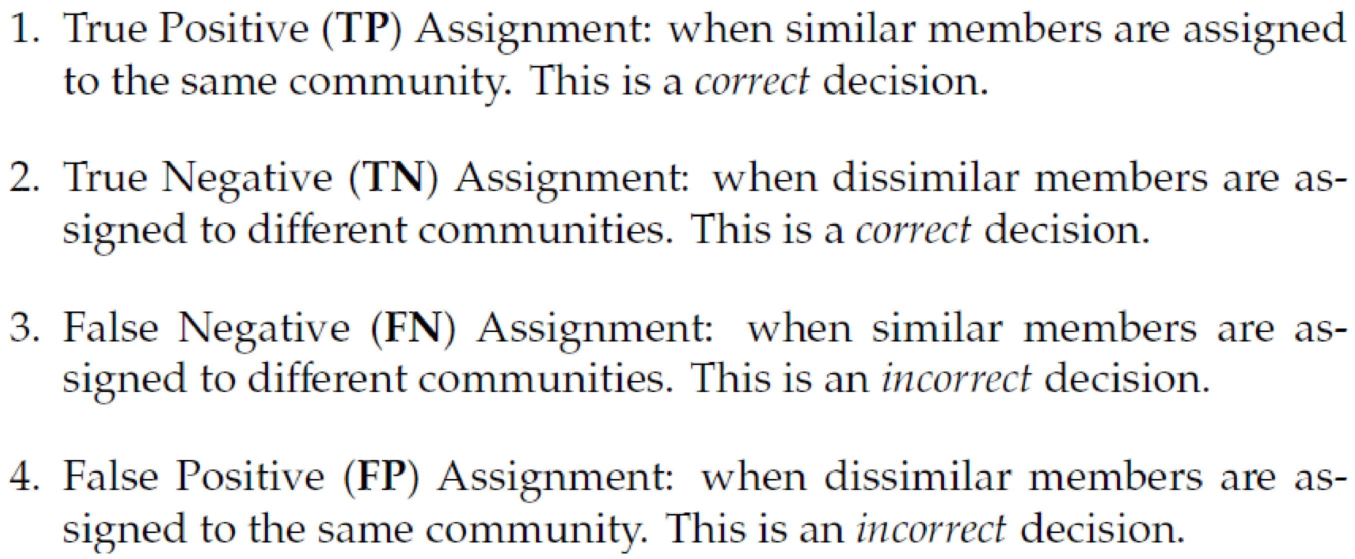
We consider two scenarios: when ground truth is available and when it is not.

Evaluation with Ground Truth

Here, we assume that we are given the correct community (clustering) assignments. We discuss four measures: *precision* and *recall*, *F-measure*, *purity*, and *normalized mutual information* (NMI).

Precision and Recall

Community detection can be considered a problem of assigning all similar nodes to the same community.



1. True Positive (**TP**) Assignment: when similar members are assigned to the same community. This is a *correct* decision.
2. True Negative (**TN**) Assignment: when dissimilar members are assigned to different communities. This is a *correct* decision.
3. False Negative (**FN**) Assignment: when similar members are assigned to different communities. This is an *incorrect* decision.
4. False Positive (**FP**) Assignment: when dissimilar members are assigned to the same community. This is an *incorrect* decision.

Evaluation with Ground Truth

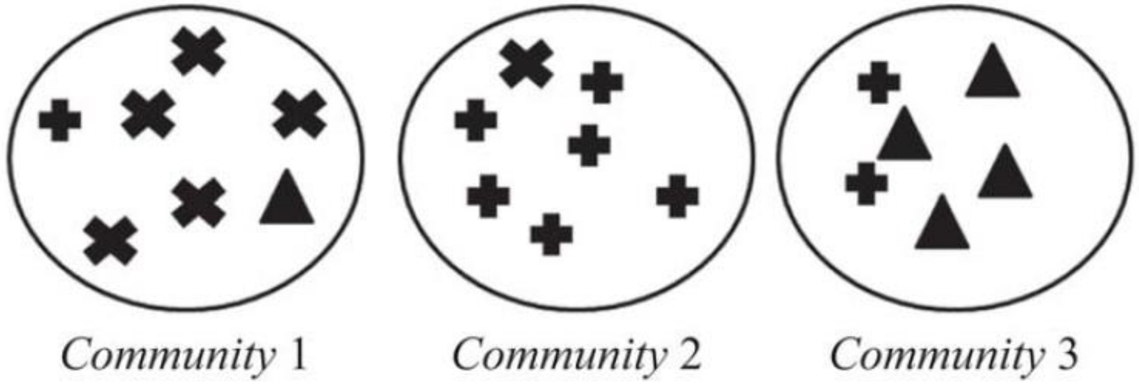
Precision (P) and Recall (R) are defined as follows,

$$\begin{aligned} P &= \frac{TP}{TP+FP}, \\ R &= \frac{TP}{TP+FN}. \end{aligned}$$

Precision defines the fraction of pairs that have been correctly assigned to the same community.

Recall defines the fraction of pairs that the community detection algorithm assigned to the same community of all the pairs that should have been in the same community.

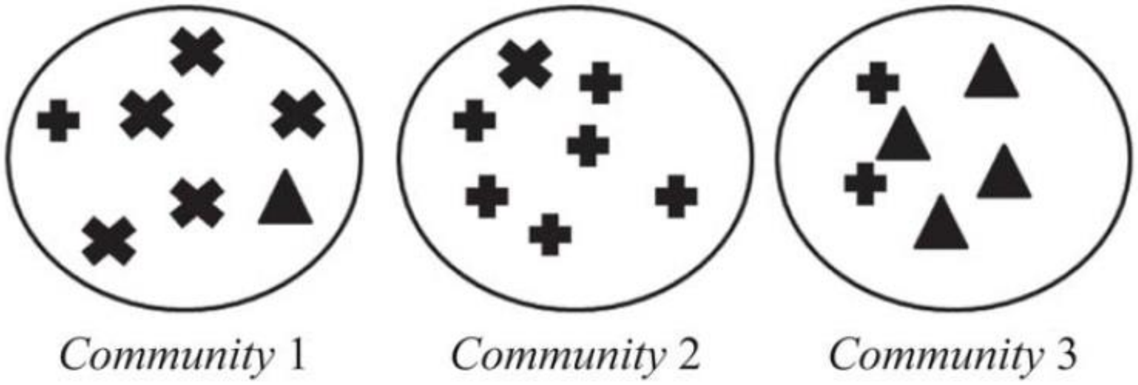
Evaluation with Ground Truth



Example *We compute these values for Figure 6.15. For TP, we need to compute the number of pairs with the same label that are in the same community. For instance, for label × and community 1, we have $\binom{5}{2}$ such pairs. Therefore,*

$$TP = \underbrace{\binom{5}{2}}_{\text{Community 1}} + \underbrace{\binom{6}{2}}_{\text{Community 2}} + \underbrace{\left(\binom{4}{2} + \binom{2}{2}\right)}_{\text{Community 3}} = 32.$$

Evaluation with Ground Truth



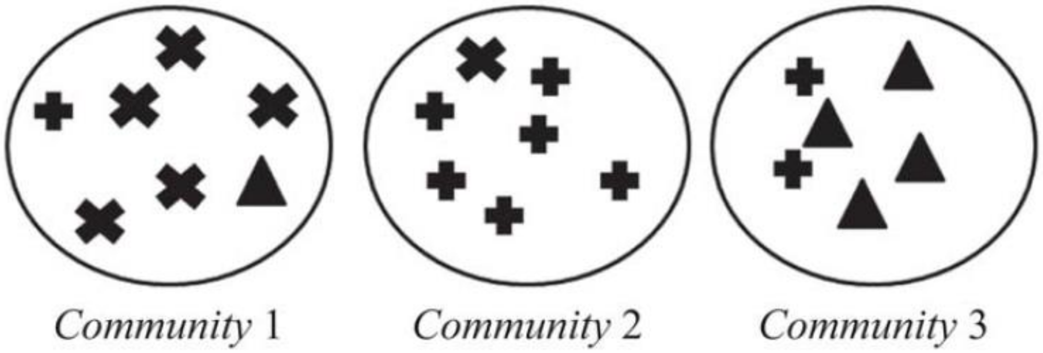
For FP, we need to compute dissimilar pairs that are in the same community. For instance, for community 1, this is $(5 \times 1 + 5 \times 1 + 1 \times 1)$. *Therefore,*

$$FP = \underbrace{(5 \times 1 + 5 \times 1 + 1 \times 1)}_{Community\ 1} + \underbrace{(6 \times 1)}_{Community\ 2} + \underbrace{(4 \times 2)}_{Community\ 3} = 25.$$

FN computes similar members that are in different communities. For instance, for label $+$, *this is* $(6 \times 1 + 6 \times 2 + 2 \times 1)$. *Similarly,*

$$FN = \underbrace{(5 \times 1)}_{\times} + \underbrace{(6 \times 1 + 6 \times 2 + 2 \times 1)}_{+} + \underbrace{(4 \times 1)}_{\triangle} = 29.$$

Evaluation with Ground Truth



Finally, *TN* computes the number of dissimilar pairs in dissimilar communities:

$$TN = \underbrace{(\overbrace{5\times 6}^{\times,+} + \overbrace{1\times 1}^{+,\times} + \overbrace{1\times 6}^{\Delta,+} + \overbrace{1\times 1}^{\Delta,\times})}_{\textit{Communities 1 and 2}}$$
$$+ \underbrace{(\overbrace{5\times 4}^{\times,\Delta} + \overbrace{5\times 2}^{\times,+} + \overbrace{1\times 4}^{+,\Delta} + \overbrace{1\times 2}^{\Delta,+})}_{\textit{Communities 1 and 3}}$$
$$+ \underbrace{(\overbrace{6\times 4}^{+,\Delta} + \overbrace{1\times 2}^{\times,+} + \overbrace{1\times 4}^{\times,\Delta}}_{\textit{Communities 2 and 3}} = 104.$$

Hence,

$$P = \frac{32}{32+25} = 0.56$$
$$R = \frac{32}{32+29} = 0.52.$$

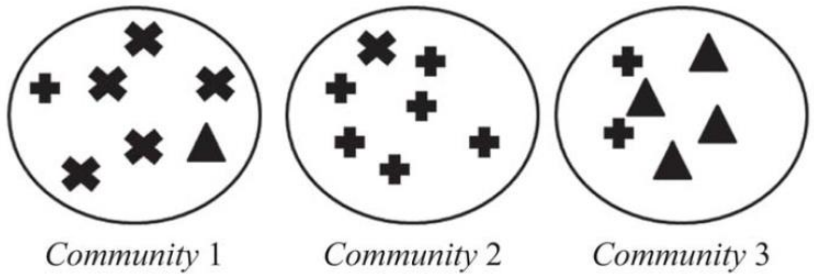
Evaluation with Ground Truth

F-Measure

To consolidate precision and recall into one measure, we can use the harmonic mean of precision and recall:

$$F = 2 \cdot \frac{\mathrm{P} \cdot \mathrm{R}}{\mathrm{P} + \mathrm{R}}.$$

Computed for the same example, we get $F = 0.54$.



Purity

we assume that the majority of a community represents the community. Hence, we use the label of the majority of the community against the label of each member of the community to evaluate the algorithm.

$$Purity = \frac{1}{N} \sum_{i=1}^{k} \max_j |C_i \cap L_j|,$$

where k is the number of communities, N is the total number of nodes, L_j is the set of instances with label j in all communities, and C_i is the set of members in community i.

In the case of our example, purity is $\frac{5+6+4}{20} = 0.75$

Evaluation with Ground Truth

Purity can be easily manipulated to generate high values; consider when nodes represent singleton communities (of size 1) or when we have very large pure communities (*ground truth* = *majority label*). In both cases, purity does not make sense because it generates high values.

Normalized Mutual Information (NMI)

Mutual information (MI) describes the amount of information that two random variables share.

Mutual information of two variables X and Y is denoted as *I(X,Y)*.

We can use mutual information to measure the information one clustering carries regarding the ground truth.

$$MI = I(X, Y) = \sum_{h \in H} \sum_{l \in L} \frac{n_{h,l}}{n} \log \frac{n \cdot n_{h,l}}{n_h n_l}$$

n_h and n_l are the number of data points in community *h* and with label *l*, respectively;
$n_{h,l}$ is the number of nodes in community *h* and with label *l*; and *n* is the number of nodes.

Unfortunately, mutual information is *unbounded*; however, it is common for measures to have values in range [0,1].

Evaluation with Ground Truth

Provide the following equation $MI \leq min(H(L), H(H))$

where $H(\cdot)$ is the entropy function,

$$H(L) = -\sum_{l \in L} \frac{n_l}{n} \log \frac{n_l}{n}$$

$$H(H) = -\sum_{h \in H} \frac{n_h}{n} \log \frac{n_h}{n}.$$

We have MI <= H(L) and MI <= H(H), therefore,

$$(MI)^2 \leq H(H)H(L).$$

Equivalently,

$$MI \leq \sqrt{H(H)}\sqrt{H(L)}.$$

An NMI value close to one indicates high similarity between communities found and labels.

$$NMI = \frac{MI}{\sqrt{H(L)}\sqrt{H(H)}}.$$

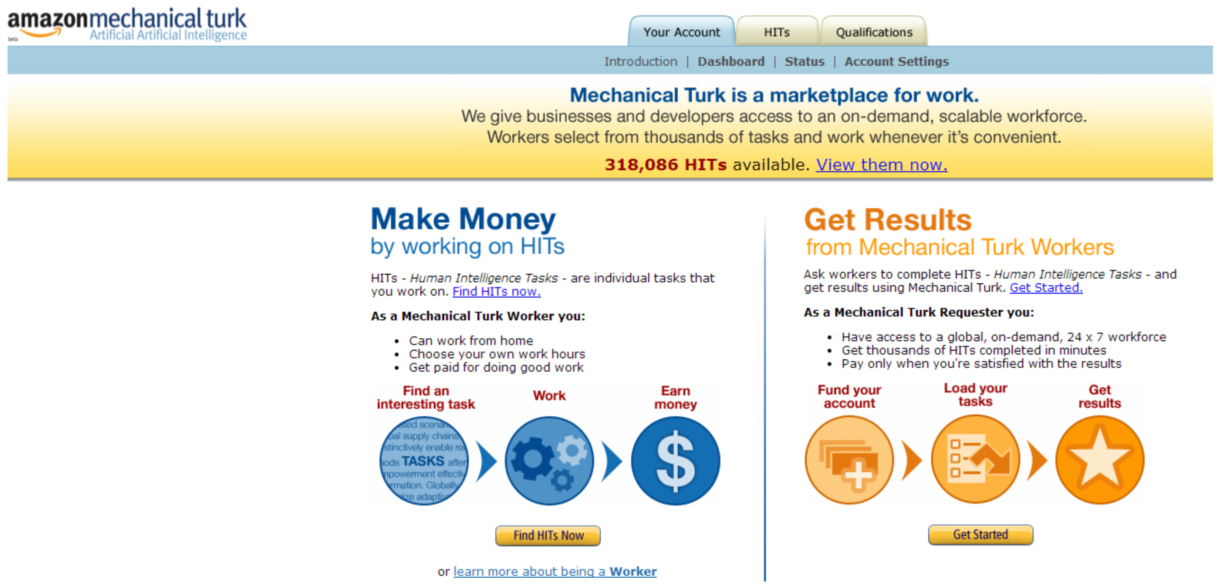
$$NMI = \frac{\sum_{h \in H} \sum_{l \in L} n_{h,l} \log \frac{n \cdot n_{h,l}}{n_h n_l}}{\sqrt{(\sum_{h \in H} n_h \log \frac{n_h}{n})(\sum_{l \in L} n_l \log \frac{n_l}{n})}}.$$

Evaluation without Ground Truth

When no ground truth is available, we can incorporate techniques based on *semantics* or ***clustering quality measures*** to evaluate community detection algorithms.

Evaluation with Semantics

A simple way of analyzing detected communities is to analyze other attributes (posts, profile information, content generated, etc.) of community members to see if there is a coherency among community members. **The coherency is often checked via human subjects.**



amazonmechanical turk
Artificial Artificial Intelligence

Your Account | HITs | Qualifications

Introduction | Dashboard | Status | Account Settings

Mechanical Turk is a marketplace for work.
We give businesses and developers access to an on-demand, scalable workforce.
Workers select from thousands of tasks and work whenever it's convenient.

318,086 HITs available. View them now.

Make Money
by working on HITs

HITs - *Human Intelligence Tasks* - are individual tasks that you work on. Find HITs now.

As a Mechanical Turk Worker you:

- Can work from home
- Choose your own work hours
- Get paid for doing good work

Find an interesting task → Work → Earn money

Find HITs Now

or learn more about being a **Worker**

Get Results
from Mechanical Turk Workers

Ask workers to complete HITs - *Human Intelligence Tasks* - and get results using Mechanical Turk. Get Started.

As a Mechanical Turk Requester you:

- Have access to a global, on-demand, 24 x 7 workforce
- Get thousands of HITs completed in minutes
- Pay only when you're satisfied with the results

Fund your account → Load your tasks → Get results

Get Started

Evaluation without Ground Truth

Evaluation Using Clustering Quality Measures

When experts are not available, an alternative is to use clustering quality measures.

> *Condition*: This approach is commonly used when two or more community detection algorithms are available.

Each algorithm is run on the target network, and ***the quality measure*** is computed for the identified communities. The algorithm that yields a more desirable quality measure value is considered a better algorithm.

NOTE. We must ensure that the clustering quality measure used to evaluate community detection is different from the measure used to find communities.