Mining Social-Network Graphs

Hung Le

University of Victoria

March 16, 2019

Social-Network Graphs

Social networks become more and more popular now. Most popular social networks (as of January 2019) are:

- Facebook: 2.2 B active users.
- Youtube: 1.9 B active users.
- WhatsApp: 1.5 B active users
- And more[1].

[1] https://www.statista.com/statistics/272014/global-social-networks-ranked-by-number-of-users/

What is a Social Network

Some common characteristics:

- A set of entities in the network.
- At least one relationship between entities, so-called *friend relationship*. It may be:
  - Two-way: typical friend relationship.
  - One-way: following relationship.
  - Weighted: friends, family, acquaintances, etc.
- Locality or nonrandomness such as the formation of communities.

Representing Social Networks

We often represent social networks by graphs, call *social graphs*.

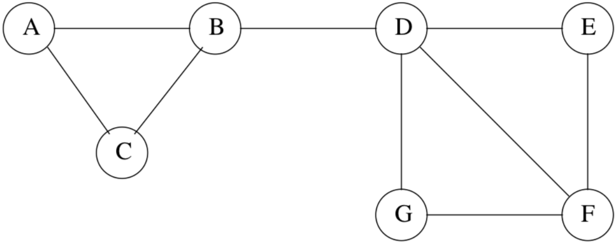


Figure: An example of a small social network.

Examples of Social Networks

Telephone Networks:

- Nodes: phone numbers.
- Edges: Calls placed between phones.
- Communities: groups of people communicate frequently, such as groups of friends, members of a club, or people working at the same company, etc.

Examples of Social Networks (Cont.)

Email Networks:

- Nodes: email addresses.
- Edges: (two-way) email exchanges between addresses.
- Communities: groups of people communicate frequently, such as groups of friends, members of a club, or people working at the same company, etc.

Examples of Social Networks (Cont.)

Collaboration Networks:

- Nodes: people who have published papers.
- Edges: people publishing papers jointly.
- Communities: groups of authors working on particular topics.

Examples of Social Networks (Cont.)

Many other types:

- Information Network (documents, web graphs, patents).
- Infrastructure networks (roads, planes, water pipes, powergrids).
- Biological networks (genes, proteins, food-webs of animals eating each other).
- Many more.

Graphs with more than one Node Types

Facebook has:

- Regular nodes: each node corresponds to a person.
- Group: each node correspond to a group of people sharing a common interest.

Our main goal in this lecture

Identify "communities" which are subset of nodes with unusually strong connections.

Clustering

We can use clustering techniques, such as HC or K-means.

- Distance measure: shortest path distances between nodes in graphs.

This typically produces undesirable or unstable results.

Edge Betweenness

Betweenness of an edge e, denoted by $B(e)$, intuitively is the number of pairs of nodes (x, y) such that $e \in P(x, y)$, where $P(x, y)$ is the shortest path between x, y.

Edge Betweenness

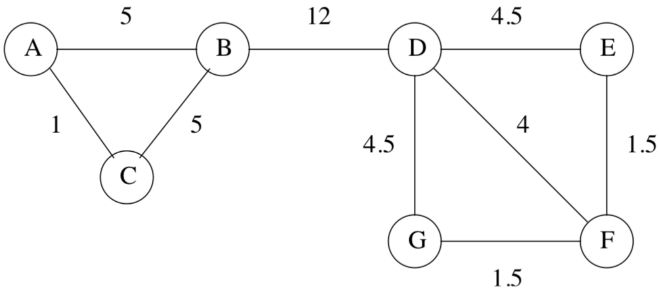
Betweenness of an edge e, denoted by $B(e)$, intuitively is the number of pairs of nodes (x, y) such that $e \in P(x, y)$, where $P(x, y)$ is the shortest path between x, y.

- There maybe more than one shortest path between two nodes x, y.
- Define $B_{xy}(e)$ to be the *fraction* of shortest paths between x, y going through e.

$$B(e) = \sum_{x=1}^{n} \sum_{y=x+1}^{n} B_{x,y}(e) \tag{1}$$

assuming nodes are indexed from 1 to n.

Edge Betweenness - An example



High betweenness means the edge is likely between different communities.

Betweenness to Communities

Remove the edges by *decreasing order* of betweenness until we obtain a desired number of communities.

Computing Edge Betweenness

```
GirvanNewman(G(V, E))
    foreach node v ∈ V
        Find a BFS tree T_v rooted at v.
        NL_v[1, ..., n] ← NodeLabeling(T_v, G)
        EL_v[1, ..., n] ← EdgeLabeling(T_v, G, NL_v)
    foreach edge e ∈ E
        B[e] ← 0
        foreach node v ∈ V
            B[e] ← B[e] + EL_v[e]
        B[e] ← B[e]/2
    return B[1, ..., m]
```

- $NL_v[u]$ is the number of shortest paths from v to u.
- $EL_v[e]$ is the contribution of shortest paths from v to e's betwenness.

Computing Edge Betweenness (Cont.)

$\textsc{NodeLabeling}(T_v, G(V,E))$
$v \leftarrow$ the root of T
$\{0, 1 \ldots L\}$ levels of nodes in T
$NL_v[v] \leftarrow 1$
for $\ell \leftarrow 1$ to L
foreach node u at level ℓ
$P_u = \{w : uw \in E \text{ and } \mathrm{level}(w) = \ell - 1\}$
$NL_v[u] \leftarrow \sum_{w \in P(u)} NL_v[w]$
return $NL_v[1, \ldots, n]$

- $NL_v[u]$ is the number of shortest paths from v to u.

Computing Edge Betweenness (Cont.)

```
EdgeLabeling(T_v, G(V,E), NL_v)
    v ← the root of T
    {0, 1 ... L} levels of nodes in T
    foreach node u at level L
        C[u] ← 1
    for ℓ ← L down to 1
        foreach u at level ℓ
            P_u = {w : uw ∈ E and level(w) = ℓ − 1}
            foreach w ∈ P_u
                EL_v[uw] ← C[u]·NL_v[w] / NL_v[u]
        foreach w at level ℓ − 1
            Pred_w = {u : wu ∈ E and level(u) = ℓ}
            C[w] ← Σ_{u∈Pred_w} EL_v[wu] + 1.0
    return EL_v[1, ..., n]
```

- $EL_v[e]$ is the contribution of shortest paths from v to e's betwenness.

Computing Edge Betweenness (Cont.)

```
GirvanNewman(G(V, E))
    foreach node v ∈ V
        Find a BFS tree T_v rooted at v.
        NL_v[1, ..., n] ← NodeLabeling(T_v, G)
        EL_v[1, ..., n] ← EdgeLabeling(T_v, G, NL_v)
    foreach edge e ∈ E
        B[e] ← 0
        foreach node v ∈ V
            B[e] ← B[e] + EL_v[e]
        B[e] ← B[e]/2
    return B[1, ..., m]
```

Running time: $O(nm)$.

- In practice, we pick a subset of the nodes at random and use these as the roots of breadth-first searches to get an approximation of betweenness.

Graph Partitioning

Divide the graph into two parts so that the *cut*, the set of edges between two parts, is minimized.

- Typically want two parts have roughly equal size.

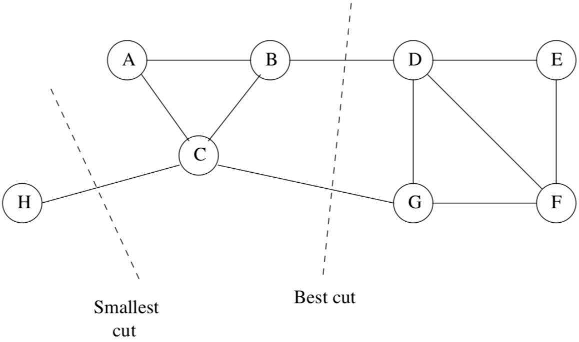


Figure: An example of a good cut.

Normalized Cut

Let $S \subset V$ and $T = V \setminus S$. Let $E(S, T)$ be the set of edges with one endpoint in S and one endpoint in T.

$$\begin{gathered} \mathrm{Cut}(S, T) = |E(S, T)| \\ \mathrm{Vol}(S) = \sum_{u \in S} \deg_G(u) \quad \mathrm{Vol}(T) = \sum_{u \in T} \deg_G(u) \end{gathered} \tag{2}$$

The *normalized cut value* for S, T, denoted by $\mathrm{NC}(S, T)$, is:

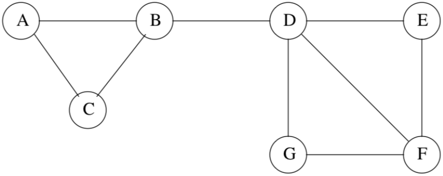
$$\mathrm{NC}(S, T) = \frac{\mathrm{Cut}(S, T)}{\mathrm{Vol}(S)} + \frac{\mathrm{Cut}(S, T)}{\mathrm{Vol}(T)} \tag{3}$$

We want to find cut with minimum $\mathrm{NC}(S, T)$.

Graphs as Matrices

Adjacency matrix $A_{n \times n}$ where:

$$A[i,j] = \begin{cases} 1 & \text{if edge } i - j \in E \\ 0 & \text{otherwise} \end{cases}$$

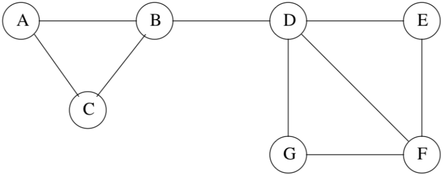


$$\begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

Graphs as Matrices (Cont.)

Degree matrix $D_{n \times n}$ where:

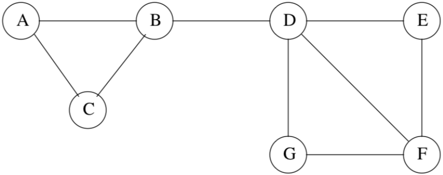
$$D[i,j] = \begin{cases} \deg_G[i] & \text{if edge } i = j \\ 0 & \text{otherwise} \end{cases}$$



$$\begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 \end{bmatrix}$$

Graphs as Matrices (Cont.)

Laplacian Matrix $L = D - A$.



$$\begin{bmatrix} 2 & -1 & -1 & 0 & 0 & 0 & 0 \\ -1 & 3 & -1 & -1 & 0 & 0 & 0 \\ -1 & -1 & 2 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 4 & -1 & -1 & -1 \\ 0 & 0 & 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & -1 & -1 & 3 & -1 \\ 0 & 0 & 0 & -1 & 0 & -1 & 2 \end{bmatrix}$$

Eigenvalues and Eigenvectors of Laplacian Matrices

Laplacian L has an eigenvector $\mathbf{x} \in \mathrm{R}^n$ associated with an eigenvalue $\lambda \in \mathrm{R}$ if:

$$L\mathbf{x} = \lambda\mathbf{x} \tag{4}$$

Fact 1: L has n eigenvalues s.t $0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$.

Eigenvalues and Eigenvectors of Laplacian Matrices

Laplacian L has an eigenvector $\mathbf{x} \in \mathrm{R}^n$ associated with an eigenvalue $\lambda \in \mathrm{R}$ if:

$$L\mathbf{x} = \lambda\mathbf{x} \tag{4}$$

Fact 1: L has n eigenvalues s.t $0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$.
Fact 2: The eigenvector associated with λ_1 $(= 0)$ of L is $\mathbf{1}_n$.

Eigenvalues and Eigenvectors of Laplacian Matrices

Laplacian L has an eigenvector $\mathbf{x} \in \mathrm{R}^n$ associated with an eigenvalue $\lambda \in \mathrm{R}$ if:

$$L\mathbf{x} = \lambda \mathbf{x} \tag{4}$$

Fact 1: L has n eigenvalues s.t $0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$.
Fact 2: The eigenvector associated with λ_1 $(= 0)$ of L is $\mathbf{1}_n$.
Fact 3: The second eigenvector, denoted by $\mathbf{x}_2$, associated with λ_2 of L satisfies:

$$\mathbf{x}_2 = \arg\min \mathbf{x}^T L \mathbf{x} \tag{5}$$

subject to

$$\begin{aligned} \mathbf{x}_2^T \mathbf{1}_n &= 0 \\ \sum_{i=1}^{n} x_2[i]^2 &= 1 \end{aligned} \tag{6}$$

Understanding λ_2 and $\mathbf{x}_2$

$$\mathbf{x}^T L\mathbf{x} = \sum_{(i,j)\in E} (x[i] - x[j])^2 \tag{7}$$

Why? Let $N[i]$ be the set of neighbors of i, including i.

$$\begin{aligned}
\mathbf{x}^T L\mathbf{x} &= \sum_{i=1}^{n} \sum_{j\in N[i]} x[i]L[i,j]x[j] \\
&= \sum_{i=1}^{n} \sum_{j\in N[i]} x[i](D[i,j] - A[i,j])x[j] \\
&= \sum_{i=1}^{n} d[i]x[i]^2 - 2\sum_{(i,j)\in E} x[i]x[j] \\
&= \sum_{(i,j)\in E} (x[i] - x[j])^2
\end{aligned} \tag{8}$$

Understanding λ_2 and $\mathbf{x}_2$

$$\mathbf{x}^T L\mathbf{x} = \sum_{(i,j)\in E} (x[i] - x[j])^2 \tag{9}$$

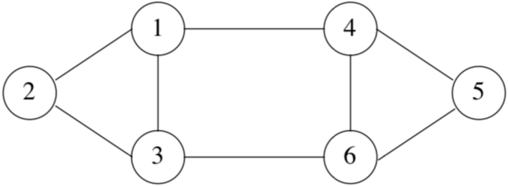
Recall: The second eigenvector, denoted by $\mathbf{x}_2$, associated with λ_2 of L satisfies:

$$\mathbf{x}_2 = \arg\min \mathbf{x}^T L\mathbf{x} \tag{10}$$

subject to

$$\begin{aligned} \mathbf{x}_2^T \mathbf{1}_n &= 0 \\ \sum_{i=1}^{n} x_2[i]^2 &= 1 \end{aligned} \tag{11}$$

Understanding λ_2 and $\mathbf{x}_2$



Eigenvalue	0	1	3	3	4	5
Eigenvector	1	1	-5	-1	-1	-1
	1	2	4	-2	1	0
	1	1	1	3	-1	1
	1	-1	-5	-1	1	1
	1	-2	4	-2	-1	0
	1	-1	1	3	1	-1

Finding Overlapping Community

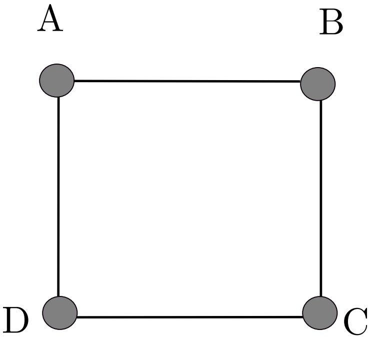
It's is natural to expect that a person belonging to two or more community.

Maximum Likelihood Estimation - MLE

Ideas: Assume that the network is generated by a probabilistic process with a set of parameters $\mathbf{p}$. Find $\mathbf{p}$ so that the probability (or likelihood) of observing the network is maximum.

- The process of finding $\mathbf{p}$ will give us the set of (overlapping) communities.

MLE - An example



Suppose that each edge is generated with probability p.

- What is the probability of observing this graph? Answer: $p^4(1-p)^2$.
- When this probability is maximize? Answer $p = 2/3$ (see the board calculation)

The Affiliation Graph Model

1. There is a given number of communities and nodes.
2. Each community has a set of nodes as members. The memberships are parameters of the model.
3. Each community C has a parameter p_C: two people in the community is connected by an edge with probability p_C. All p_C values are parameters of the model.
4. If two nodes u, v belong to more than one community, then there is an edge uv if any community containing both u, v justifies for it.

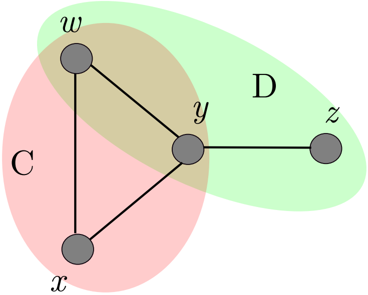
The Affiliation Graph Model

1. There is a given number of communities and nodes.
2. Each community has a set of nodes as members. The memberships are parameters of the model.
3. Each community C has a parameter p_C: two people in the community is connected by an edge with probability p_C. All p_C values are parameters of the model.
4. If two nodes u, v belong to more than one community, then there is an edge uv if any community containing both u, v justifies for it.

Property (4) means:

$$p_{uv} = 1 - \prod_{C:\{u,v\} \subseteq C} (1 - p_C) \tag{12}$$

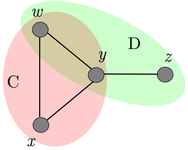
The Affiliation Graph Model - An Example



- Two communities $C = \{x, y, w\}$ and $D = \{y, w, z\}$.
- Unknown parameters: p_C, p_D.

Find p_C, p_D to maximize the MLE of the network.

The Affiliation Graph Model - An Example (Cont.)



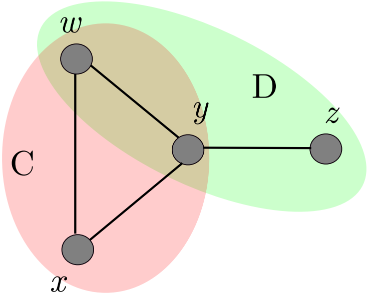
$$p_{xw} = p_{xy} = p_C \quad p_{yz} = p_D \quad p_{wy} = 1 - (1 - p_C)(1 - p_D)$$
$$p_{wz} = 1 - p_D \quad p_{xz} = 1 - \epsilon$$

Then

$$p_{\texttt{network}} = p_C^2 p_D (p_D + p_C - p_C p_D)(1 - p_D)(1 - \epsilon) \tag{13}$$

which is maximized when $p_C = 1$, $p_D = \frac{1}{2}$.

The Affiliation Graph Model - An Example (Cont.)



We found $p_C = 1$, $p_D = \frac{1}{2}$. But what is the point? Our goal is to find overlapping communities, not just the parameters.

The Affiliation Graph Model- Revisited

1. There is a given number of communities and nodes.
2. Each community has a set of nodes as members. The memberships are parameters of the model.
3. Each community C has a parameter p_C: two people in the community is connected by an edge with probability p_C. All p_C values are parameters of the model.
4. If two nodes u, v belong to more than one community, then there is an edge uv if any community containing both u, v justifies for it.

We haven't seen membership parameters. These parameters will give us communities.

The Affiliation Graph Model- Membership Parameters

For each node x and a given community C, there is a *strength of membership* parameter F_{xC}.

- Given a community C and two nodes $u, v \in C$, the probability that there is an edge uv in C is:

$$p_C uv = (1 - e^{-F_{uC}F_{vC}}) \tag{14}$$

(No need to have p_C anymore.)

Key point: each node belongs to every community but with different degree of membership.

The Affiliation Graph Model- Membership Parameters (Cont.)

Key point: each node belongs to every community but with different degree of membership.

$$p_{uv} = 1 - \prod_{C} (1 - p_C(uv)) = 1 - e^{-\sum_C F_{uC} F_{vC}} \tag{15}$$

The likelihood of the graph:

$$p_{\texttt{network}} = \prod_{uv \in E} (1 - e^{-\sum_C F_{uC} F_{vC}}) \prod_{uv \notin E} e^{-\sum_C F_{uC} F_{vC}} \tag{16}$$

How to maximize $p_{\texttt{network}}$? Answer: we maximize $\log(p_{\texttt{network}})$ instead.

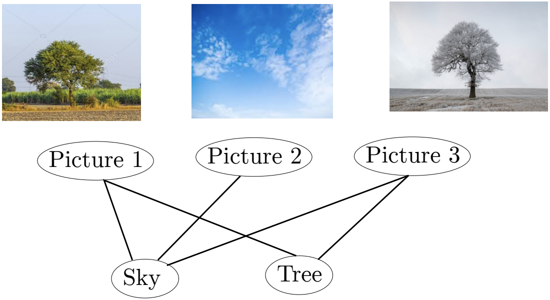
The Affiliation Graph Model- Membership Parameters (Cont.)

$$\log p_{\texttt{network}} = \sum_{uv \in E} \log(1 - e^{-\sum_C F_{uC} F_{vC}}) - \sum_{uv \notin E} \sum_C F_{uC} F_{vC} \tag{17}$$

How to maximize $\log p_{\texttt{network}}$? Answer: find each F_{uC} one at a time, assuming other values are fixed.

SimRank

Measure similarities between nodes in social graphs of many node types.



SimRank (Cont.)

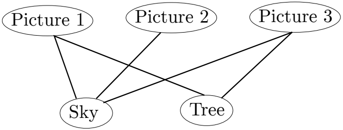
Given a node N, we want to find the similarity between N and other nodes.

Idea: start a random walk from N, with restart. The limiting distribution will give us a similarity measure.

SimRank (Cont.)

Transition matrix $M[i,j]$:

$$M[i,j] = \begin{cases} \frac{1}{\deg_G(i)} & \text{if } (i,j) \in E \\ 0 & \text{otherwise} \end{cases}$$



$$\begin{bmatrix} 0 & 0 & 0 & 1/3 & 1/2 \\ 0 & 0 & 0 & 1/3 & 0 \\ 0 & 0 & 0 & 1/3 & 1/2 \\ 1/2 & 1 & 1/2 & 0 & 0 \\ 1/2 & 0 & 1/2 & 0 & 0 \end{bmatrix}$$

Random Walk with Restart

Random walk with teleportation:

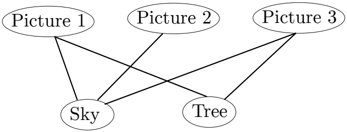
$$\mathbf{v}_t = \beta M \mathbf{v}_{t-1} + (1-\beta)\mathbf{1}_n \tag{18}$$

Random walk with restart:

$$\mathbf{v}_t = \beta M \mathbf{v}_{t-1} + (1-\beta)\mathbf{e}_N \tag{19}$$

where $\mathbf{e}[N] = 1$ and $\mathbf{e}[\mathbf{i}] = 0$ for all $i \neq N$

Random Walk with Restart - An example

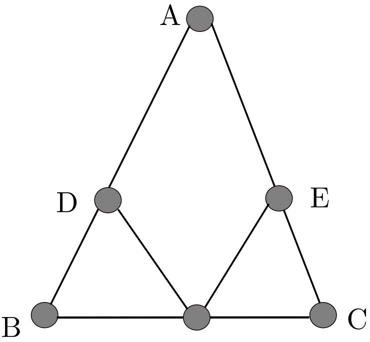


$$\mathbf{v}_t = \begin{bmatrix} 0 & 0 & 0 & 4/15 & 2/5 \\ 0 & 0 & 0 & 4/15 & 0 \\ 0 & 0 & 0 & 4/15 & 2/5 \\ 2/5 & 4/5 & 2/5 & 0 & 0 \\ 2/5 & 0 & 2/5 & 0 & 0 \end{bmatrix} \mathrm{v}_{t\text{-}1} + \begin{bmatrix} 1/5 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1/5 \\ 0 \\ 0 \\ 2/5 \\ 2/5 \end{bmatrix}, \begin{bmatrix} 35/75 \\ 8/75 \\ 20/75 \\ 6/75 \\ 6/75 \end{bmatrix}, \begin{bmatrix} 95/375 \\ 8/375 \\ 20/375 \\ 142/375 \\ 110/375 \end{bmatrix}, \begin{bmatrix} 2353/5625 \\ 568/5625 \\ 1228/5625 \\ 786/5625 \\ 690/5625 \end{bmatrix}, \ldots, \begin{bmatrix} .345 \\ .066 \\ .145 \\ .249 \\ .196 \end{bmatrix}$$

Counting Triangles

A triangle is a triangle.



How many triangle do we have in this figure?

Why Counting Triangles?

Given a graph with n nodes and m edges. How many triangle do you expect to find?

- Assume that each edges is generated with probability $\frac{m}{\binom{n}{2}}$.

Why Counting Triangles?

Given a graph with n nodes and m edges. How many triangle do you expect to find?

- Assume that each edges is generated with probability $\frac{m}{\binom{n}{2}}$.

$$\mathbb{E}[\# \text{ triangles}] = \binom{n}{3}\left(\frac{m}{\binom{n}{2}}\right)^3 \sim \frac{4}{3}(m/n)^3 \tag{20}$$

Why Counting Triangles?

Given a graph with n nodes and m edges. How many triangle do you expect to find?

- Assume that each edges is generated with probability $\frac{m}{\binom{n}{2}}$.

$$\mathbb{E}[\#\text{ triangles}] = \binom{n}{3}\left(\frac{m}{\binom{n}{2}}\right)^3 \sim \frac{4}{3}(m/n)^3 \tag{20}$$

We expect the social network graph has *much larger* # triangles because A is a friend of B, B is a friend of C then A likely is a friend of C.

- We can qualify non-randomness of the social network by counting triangles.

Counting Triangles- A Naive Algorithm

$\textsc{CountingTriangle}(G(V,E))$
$\quad C \leftarrow 0$
$\quad$ **foreach** edge $uv \in E$
$\quad\quad C \leftarrow C + |N(u) \cap N(v)|$
$\quad$ return $C/3$

Running time? $O(m\Delta)$ where Δ is the maximum degree of the graph.

Counting Triangles with High Degree Vertices

- Assume that nodes are from $\{1, 2, \dots, n\}$.
- Call a node v *heavy hitter* if $\deg_G(v) \geq \sqrt{m}$. Call it *light* otherwise.

How many heavy hitters can we have?

Counting Triangles with High Degree Vertices

- Assume that nodes are from $\{1, 2, \ldots, n\}$.
- Call a node v *heavy hitter* if $\deg_G(v) \geq \sqrt{m}$. Call it *light* otherwise.

How many heavy hitters can we have?

$$\sum_{v \in V} \deg_G(v) = 2m \tag{21}$$

implies $\#$ of heavy hitters is at most $2\sqrt{m}$.

Counting Triangles with High Degree Vertices- Step 1

Step 1: Counting all triangles that only contain heavy hitters.

```
COUNTHEAVYTRIANGLES(G(V, E))
    V_heavy ← ∅
    for each node v ∈ V
        if deg_G(v) ≥ √m
            add v to V_heavy
    C_heavy ← 0
    for each triple {u, v, w} ⊂ V_heavy
        if uv ∈ E and uw ∈ E and vw ∈ E
            C_heavy ← C_heavy + 1
    return C_heavy
```

Running time $O(m^{1.5})$ if using a Hash table to index E.

Counting Triangles with High Degree Vertices- Step 2

Step 2: Counting all triangles that contains at least one light vertex.

- Say $v \prec u$ if (i) $\deg_G(v) < deg_v(u)$ or (ii) $\deg_G(v) = \deg_G(u)$ and $v < u$.

$\textsc{CountLightTriangles}(G(V,E))$
 $V_{\mathtt{light}} \leftarrow V \setminus V_{\mathtt{heavy}}$
 $C_{\mathtt{light}} \leftarrow 0$
 for each edge $uv \in V$
  if $\{u, v\} \cap V_{\mathtt{light}} \neq \emptyset$
   suppose $v \prec u$
   for each $w \in N(v)$
    if $v \prec w$
     $C_{\mathtt{light}} \leftarrow C_{\mathtt{light}} + 1$
 return $C_{\mathtt{light}}$

Running time $O(m\sqrt{m})$

Counting Triangles with High Degree Vertices

$\textsc{CountTriangles}(G(V, E))$
$\quad C_{\texttt{heavy}} \leftarrow \textsc{CountHeavyTriangles}(G(V, E))$
$\quad C_{\texttt{light}} \leftarrow \textsc{CountLightTriangles}(G(V, E))$
$\quad$ return $C_{\texttt{heavy}} + C_{\texttt{light}}$

Overall running time $O(m\sqrt{m})$.

- Recall the naive algorithm has running time $O(m\Delta)$ where Δ is the maximum degree.