Mining Social-Network Graphs

Hung Le

University of Victoria

March 15, 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*.

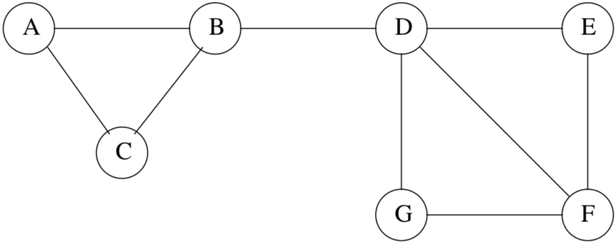


Figure: An example of a small social network.

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

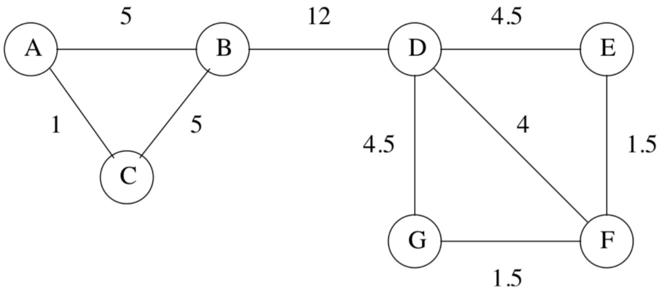
Betweenness of an edge e, denoted by $B(e)$, intuitively is the number of pairs of nodes (x, y) such that $e \in P(x, y)$, where $P(x, y)$ is the shortest path between x, y.

- 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

$\textsc{GirvanNewman}(G(V,E))$
    foreach node $v \in V$
        Find a BFS tree T_v rooted at v.
        $NL_v[1,\ldots,n] \leftarrow \textsc{NodeLabeling}(T_v, G)$
        $EL_v[1,\ldots,n] \leftarrow \textsc{EdgeLabeling}(T_v, G, NL_v)$
    foreach edge $e \in E$
        $B[e] \leftarrow 0$
        foreach node $v \in V$
            $B[e] \leftarrow B[e] + EL_v[e]$
        $B[e] \leftarrow B[e]/2$
    return $B[1,\ldots,m]$

- $NL_v[u]$ 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.)

```
NodeLabeling(T_v, G(V, E))
    v ← the root of T
    {0, 1 ... L} levels of nodes in T
    NL_v[v] ← 1
    for ℓ ← 1 to L
        foreach node u at level ℓ
            P_u = {w : uw ∈ E and level(w) = ℓ − 1}
            NL_v[u] ← Σ_{w∈P(u)} NL_v[w]
    return NL_v[1, ..., 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 betweenness.

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.

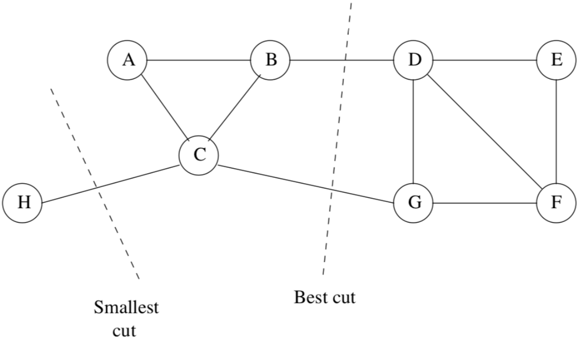


Figure: An example of a good cut.

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:

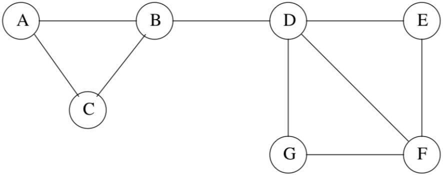
$$\mathrm{NC}(S,T) = \frac{\mathrm{Cut}(S,T)}{\mathrm{Vol}(S)} + \frac{\mathrm{Cut}(S,T)}{\mathrm{Vol}(T)} \tag{3}$$

We want to find cut with minimum $\Phi(S,T)$.

Graphs as Matrices

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

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

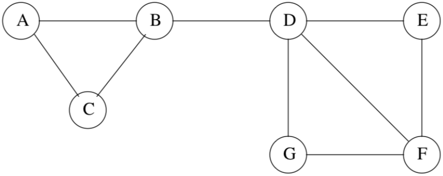


$$\begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

Graphs as Matrices (Cont.)

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

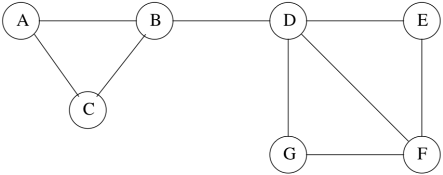
$$D[i,j] = \begin{cases} \deg_G[i] & \text{if edge } i = j \\ 0 & \text{otherwise} \end{cases}$$



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

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 $\mathbf{x}_2$

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

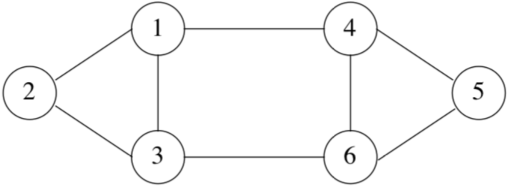
Recall: The second eigenvector, denoted by $\mathbf{x}_2$, associated with λ_2 of L satisfies:

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

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