NYU DATA SCIENCE

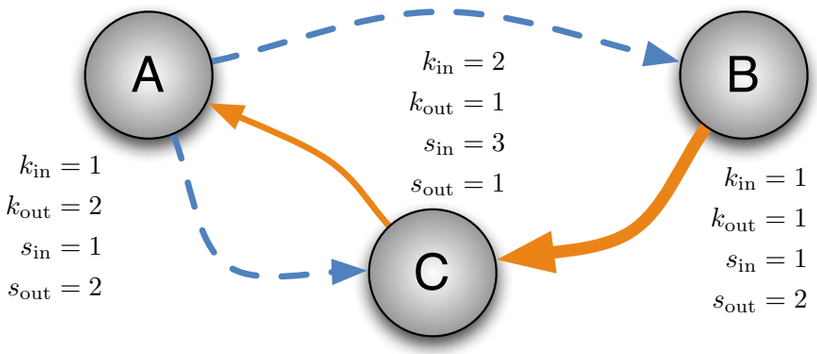
Network Science

Bruno Gonçalves

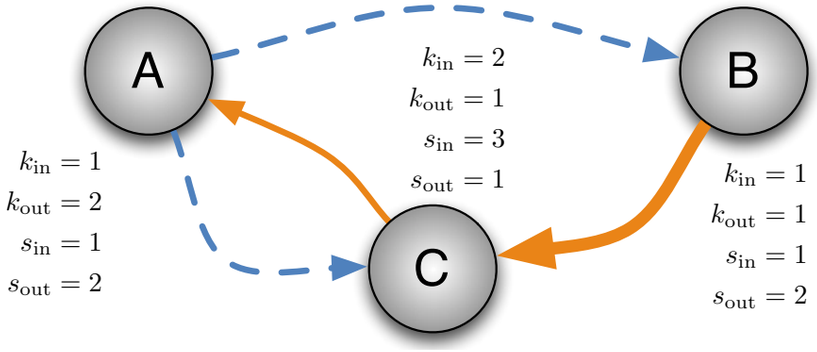
www.bgoncalves.com

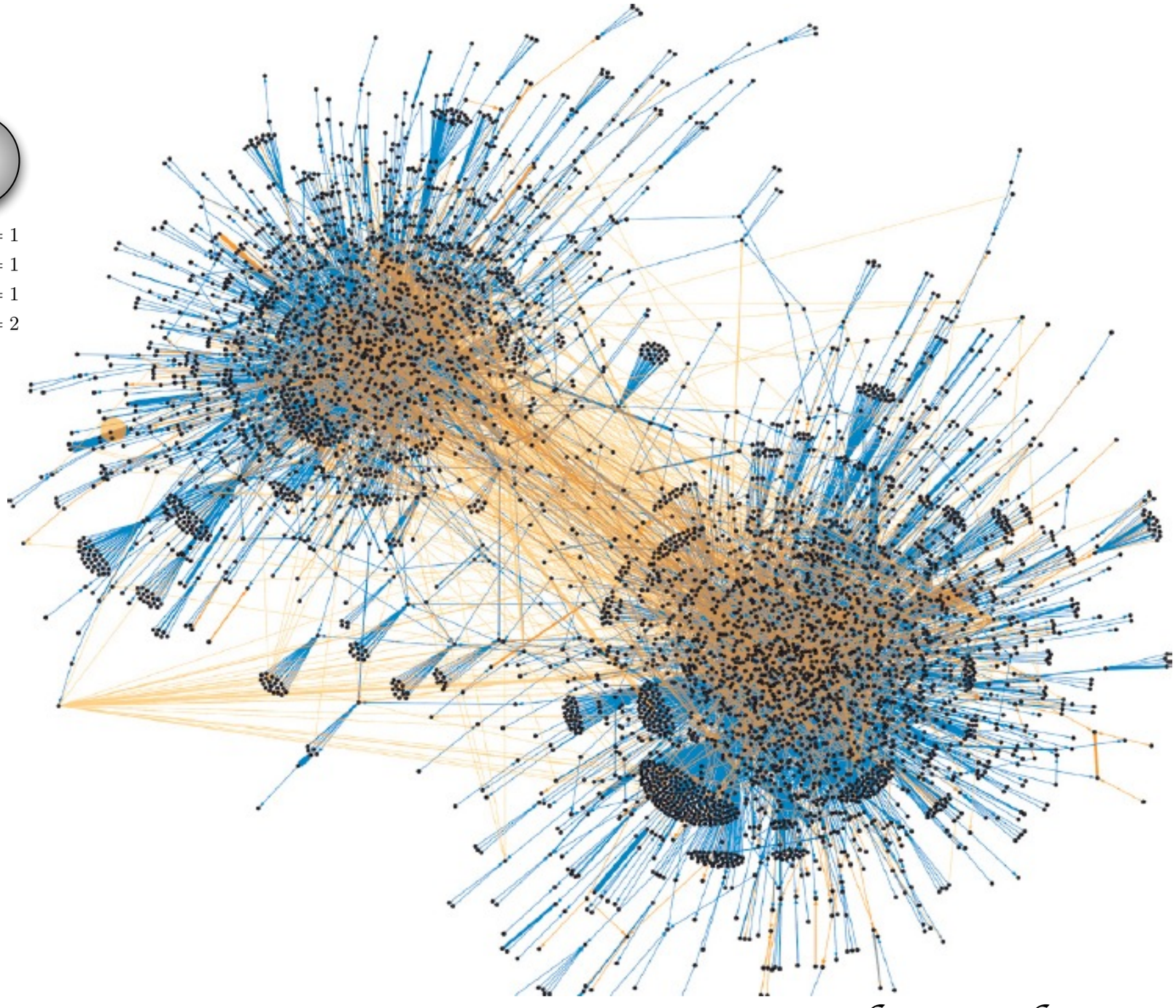
https://github.com/bmtgoncalves/Binghamton

Networks

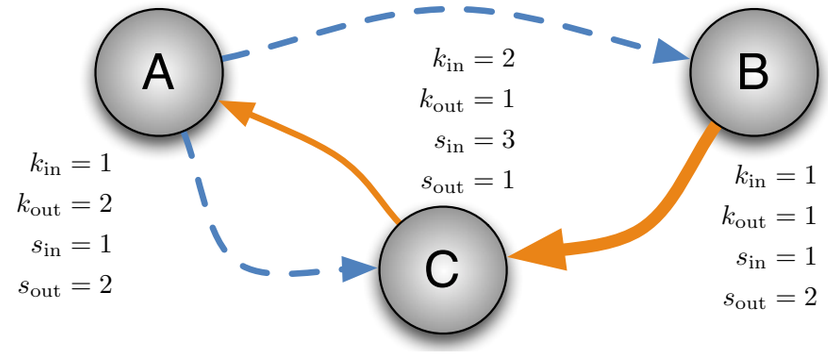

A
B
C
$k_{\text{in}} = 1$
$k_{\text{out}} = 2$
$s_{\text{in}} = 1$
$s_{\text{out}} = 2$
$k_{\text{in}} = 2$
$k_{\text{out}} = 1$
$s_{\text{in}} = 3$
$s_{\text{out}} = 1$
$k_{\text{in}} = 1$
$k_{\text{out}} = 1$
$s_{\text{in}} = 1$
$s_{\text{out}} = 2$

Networks


A
B
C
$k_{\text{in}} = 1$
$k_{\text{out}} = 2$
$s_{\text{in}} = 1$
$s_{\text{out}} = 2$
$k_{\text{in}} = 2$
$k_{\text{out}} = 1$
$s_{\text{in}} = 3$
$s_{\text{out}} = 1$
$k_{\text{in}} = 1$
$k_{\text{out}} = 1$
$s_{\text{in}} = 1$
$s_{\text{out}} = 2$

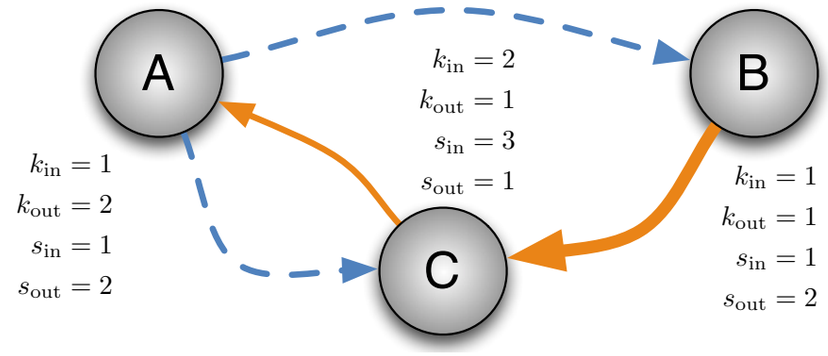


Networks - Basic Definitions



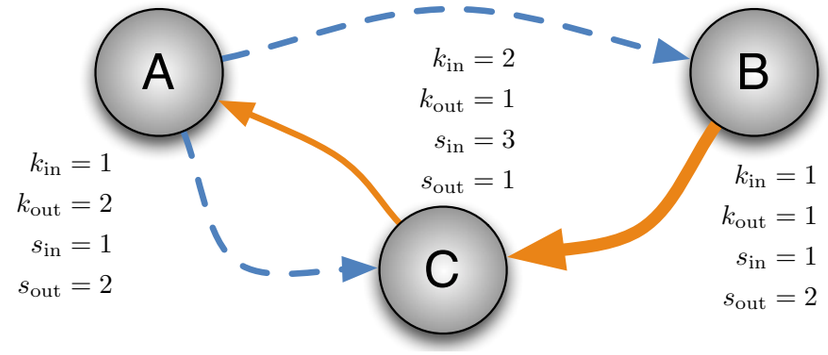
Networks - Basic Definitions

- Mathematical object, with set of nodes and edges



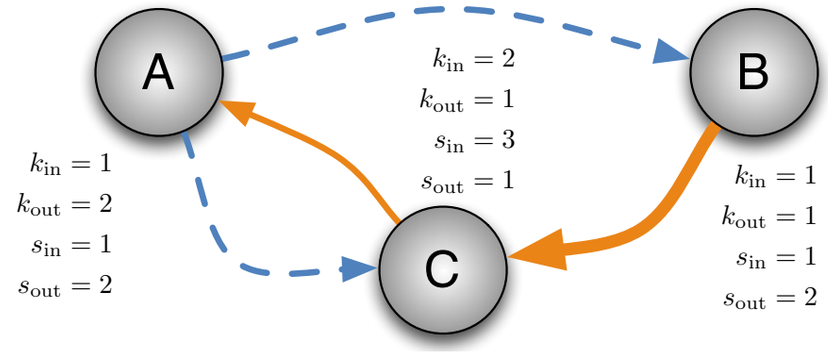
Networks - Basic Definitions

- Mathematical object, with set of nodes and edges
- Node - Individual Element



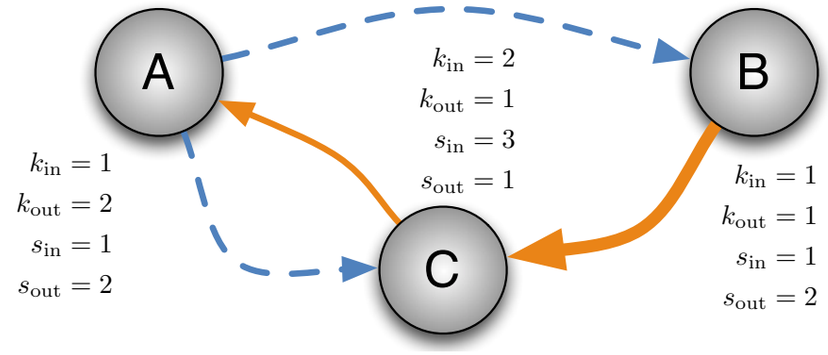
Networks - Basic Definitions

- Mathematical object, with set of nodes and edges
- Node - Individual Element
- Edge - Connection between element



Networks - Basic Definitions

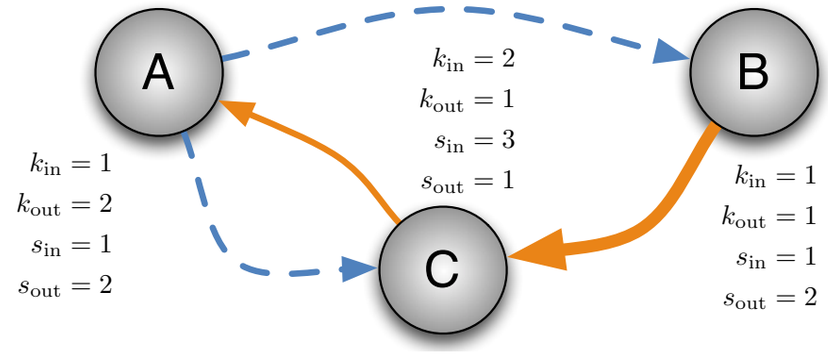
- Mathematical object, with set of nodes and edges
- Node - Individual Element
- Edge - Connection between element



- Degree - Number of edges connected to a node

Networks - Basic Definitions

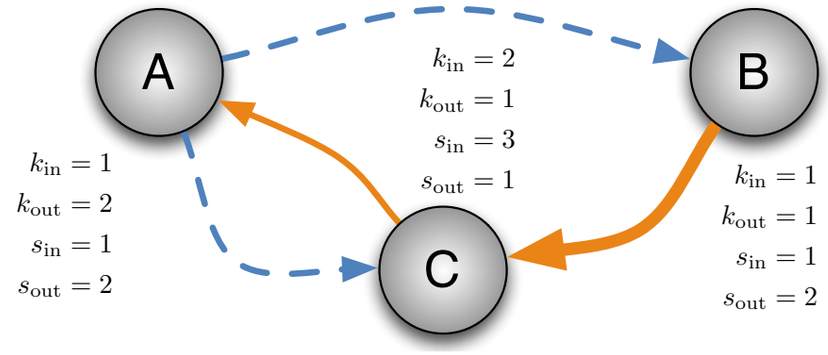
- Mathematical object, with set of nodes and edges
- Node - Individual Element
- Edge - Connection between element



- Degree - Number of edges connected to a node
- Weighted Edge - Edge with a weight associated

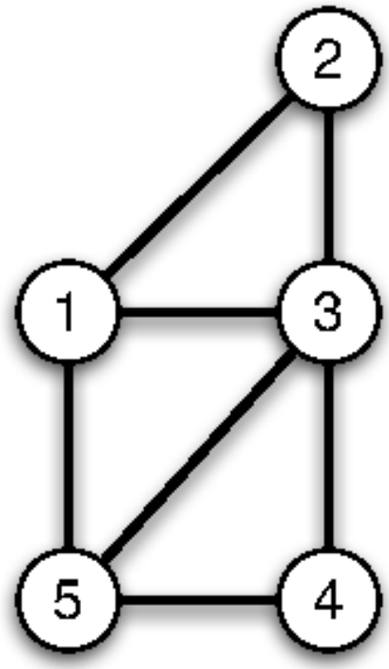
Networks - Basic Definitions

- Mathematical object, with set of nodes and edges
- Node - Individual Element
- Edge - Connection between element



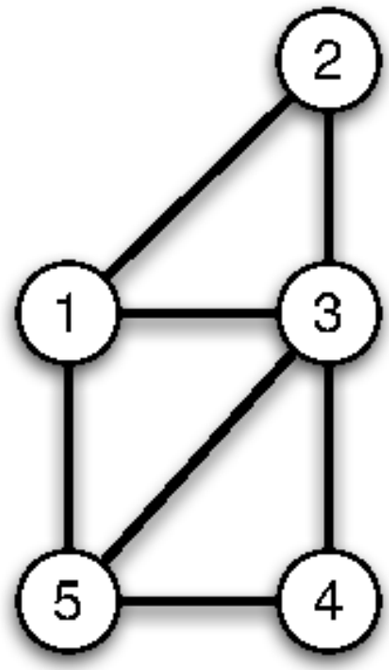
- Degree - Number of edges connected to a node
- Weighted Edge - Edge with a weight associated
- Direct Edge - "One way street"

Adjacency Matrix



Adjacency Matrix

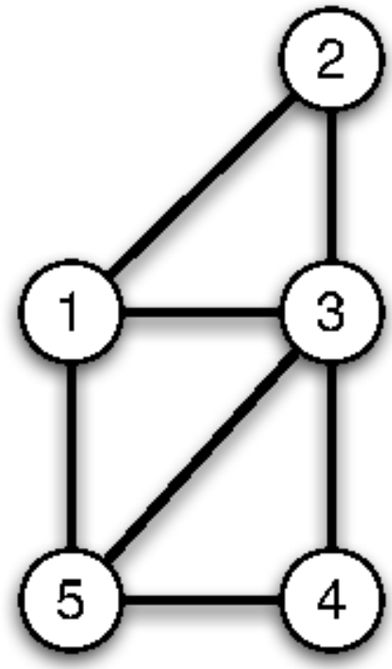
$$a_{ij} = \begin{cases} 1 & \text{i connected to j} \\ 0 & \text{otherwise} \end{cases}$$



Adjacency Matrix

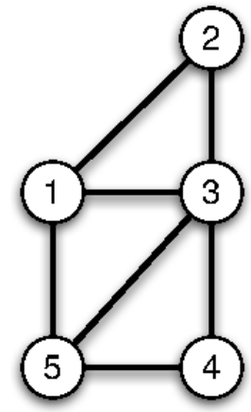
$$a_{ij} = \begin{cases} 1 & \text{i connected to j} \\ 0 & \text{otherwise} \end{cases}$$

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



Degree Distribution

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

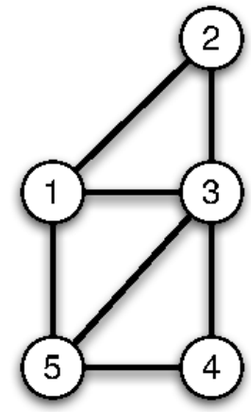


Degree Distribution

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

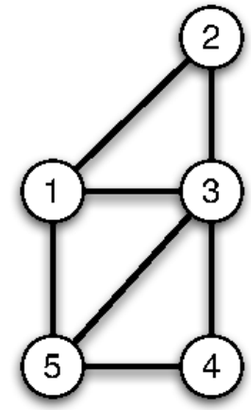
- The degree of a node is the number of edges it is connected to

$$\vec{k} = A \cdot \vec{1}$$



Degree Distribution

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



- The degree of a node is the number of edges it is connected to

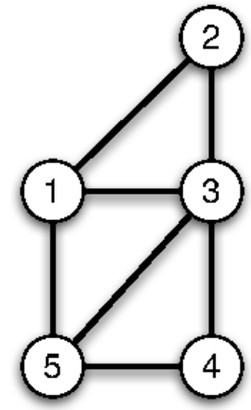
$$\vec{k} = A \cdot \vec{1}$$

- The degree distribution is then:

$$P\left(k\right) = \frac{1}{N} \sum_i \delta\left(k_i - k\right)$$

Degree Distribution

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



- The degree of a node is the number of edges it is connected to

$$\vec{k} = A \cdot \vec{1}$$

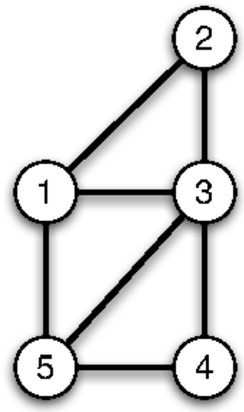
- The degree distribution is then:

$$P\left(k\right) = \frac{1}{N} \sum_i \delta\left(k_i - k\right)$$

- "The probability that a randomly chosen node has degree exactly k"

Paths and Cycles

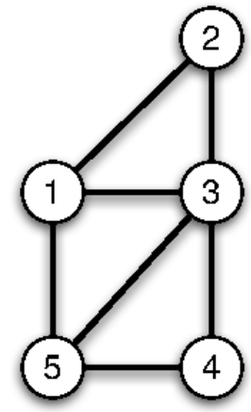
$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \end{bmatrix}$$



Paths and Cycles

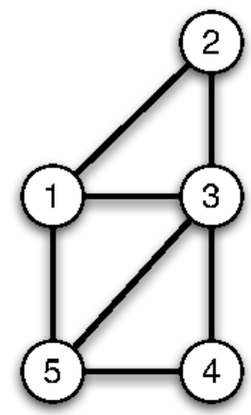
$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \end{bmatrix}$$

- Each non-zero entry of the adjacent matrix represents one edge, or a length one path connecting nodes i and j



Paths and Cycles

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



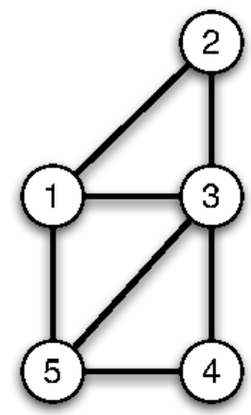
- Each non-zero entry of the adjacent matrix represents one edge, or a length one path connecting nodes i and j

- If we want paths of length 2, we must take two steps, so the total number of ways of going from node i to node k in two steps is:

$$p_{ik} = \sum_j a_{ij} a_{jk}$$

Paths and Cycles

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



- Each non-zero entry of the adjacent matrix represents one edge, or a length one path connecting nodes i and j

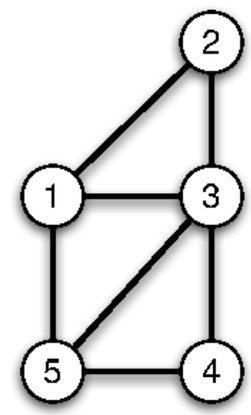
- If we want paths of length 2, we must take two steps, so the total number of ways of going from node i to node k in two steps is:

$$p_{ik} = \sum_j a_{ij} a_{jk}$$

$$A^2 = \begin{bmatrix} 3 & 1 & 2 & 2 & 1 \\ 1 & 2 & 1 & 1 & 2 \\ 2 & 1 & 4 & 1 & 2 \\ 2 & 1 & 1 & 2 & 1 \\ 1 & 2 & 2 & 1 & 3 \end{bmatrix}$$

Paths and Cycles

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



- Each non-zero entry of the adjacent matrix represents one edge, or a length one path connecting nodes i and j

- If we want paths of length 2, we must take two steps, so the total number of ways of going from node i to node k in two steps is:

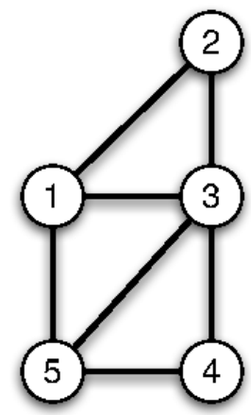
$$p_{ik} = \sum_j a_{ij} a_{jk}$$

- And in general:

$$P^{(n)} = A^n \qquad A^2 = \begin{bmatrix} 3 & 1 & 2 & 2 & 1 \\ 1 & 2 & 1 & 1 & 2 \\ 2 & 1 & 4 & 1 & 2 \\ 2 & 1 & 1 & 2 & 1 \\ 1 & 2 & 2 & 1 & 3 \end{bmatrix}$$

Paths and Cycles

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



- Each non-zero entry of the adjacent matrix represents one edge, or a length one path connecting nodes i and j

- If we want paths of length 2, we must take two steps, so the total number of ways of going from node i to node k in two steps is:

$$p_{ik} = \sum_j a_{ij} a_{jk}$$

- And in general:

$$P^{(n)} = A^n \qquad A^2 = \begin{bmatrix} 3 & 1 & 2 & 2 & 1 \\ 1 & 2 & 1 & 1 & 2 \\ 2 & 1 & 4 & 1 & 2 \\ 2 & 1 & 1 & 2 & 1 \\ 1 & 2 & 2 & 1 & 3 \end{bmatrix}$$

- The diagonal elements of P indicate cycles, paths that start and end in the same node

Clustering Coefficient

Clustering Coefficient

- "The fraction of possible links among my friends that are actually present"

Clustering Coefficient

- "The fraction of possible links among my friends that are actually present"

- The possible number of pairwise combinations of k nodes is:

$$\binom{k_i}{2} \equiv \frac{k_i\left(k_i-1\right)}{2}$$

Clustering Coefficient

- "The fraction of possible links among my friends that are actually present"

- The possible number of pairwise combinations of k nodes is:

$$\binom{k_i}{2} \equiv \frac{k_i\left(k_i-1\right)}{2}$$

- So the clustering coefficient of node i is:

$$C_i = \frac{p_{ii}^{(3)}}{k_i\left(k_i-1\right)}$$

Clustering Coefficient

- "The fraction of possible links among my friends that are actually present"

- The possible number of pairwise combinations of k nodes is:

$$\binom{k_i}{2} \equiv \frac{k_i\left(k_i-1\right)}{2}$$

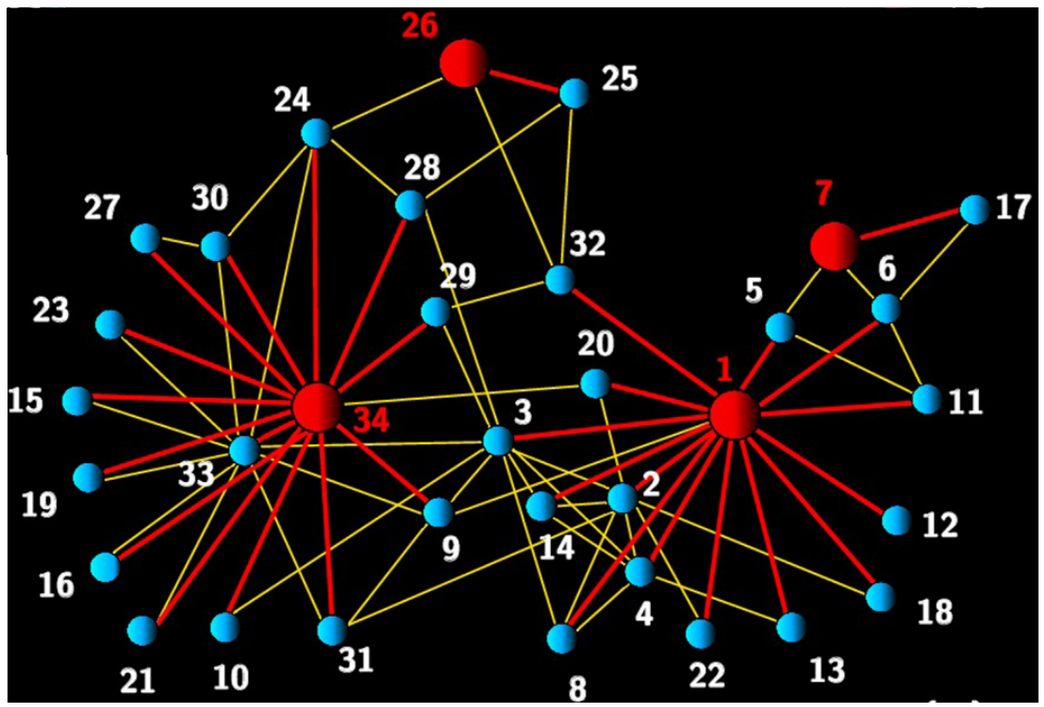
- So the clustering coefficient of node i is:

$$C_i = \frac{p_{ii}^{(3)}}{k_i\left(k_i-1\right)}$$

- And on average:

$$\langle C \rangle = \frac{1}{N}\sum_i C_i$$

Clustering Coefficient

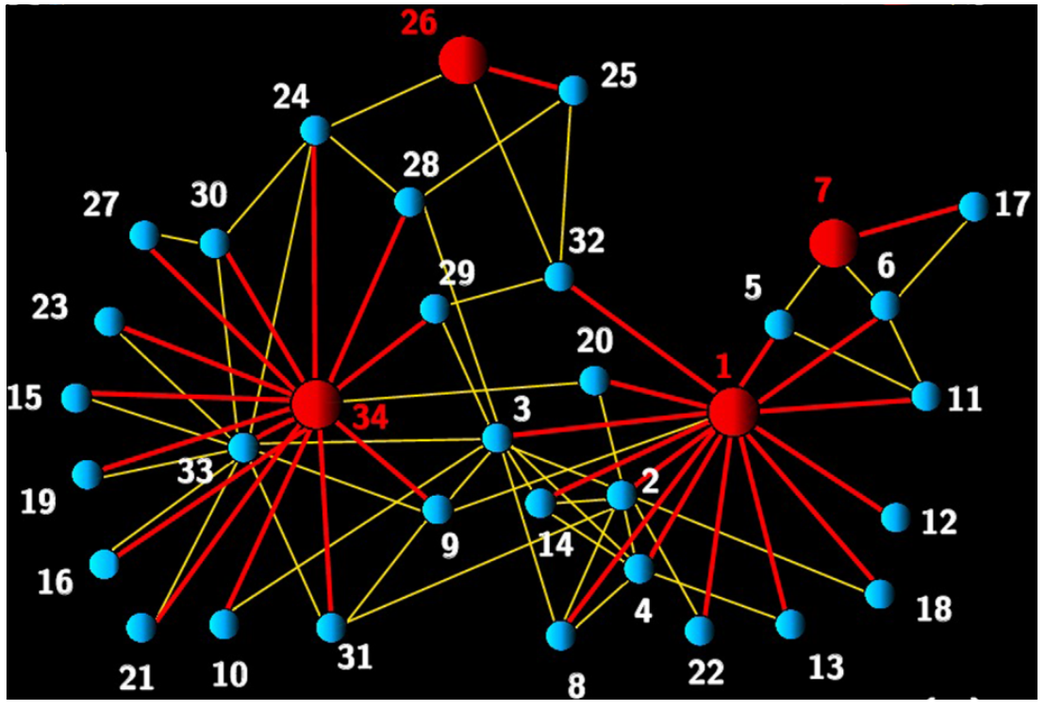


Clustering Coefficient

- Tendencially high in Social Networks

	WWW	Citations	Co-author	Ham Radio	Prison	High School Romance
Number of Nodes	325729	396	81217	44	67	572
Randomness: r	0.57	0.63	4.7	5.0	∞	∞
Avg. In-Degree: m	4.6	5.0	.84	3.5	2.7	.83
Avg. Clustering	.11	.07	.16	.47	.31	-

typically one or more orders of magnitude higher than expected in a random graph



Degree Correlations

Degree Correlations

- Let us define a matrix $E_{kk'}$ that simply counts how many edges there are between nodes of degree k and nodes of degree k'. This matrix has several useful properties:

$$\sum_k E_{kk'} = kN_k$$

$$\sum_{kk'} E_{kk'} = 2E = \langle k \rangle N$$

Degree Correlations

- Let us define a matrix $E_{kk'}$ that simply counts how many edges there are between nodes of degree k and nodes of degree k'. This matrix has several useful properties:

$$\sum_k E_{kk'} = kN_k$$

$$\sum_{kk'} E_{kk'} = 2E = \langle k \rangle N$$

- We can use the last relation to find the joint probability:

$$P\left(k,k'\right) = \frac{E_{kk'}}{\langle k \rangle N}$$

Degree Correlations

- Let us define a matrix $E_{kk'}$ that simply counts how many edges there are between nodes of degree k and nodes of degree k'. This matrix has several useful properties:

$$\sum_k E_{kk'} = kN_k$$

$$\sum_{kk'} E_{kk'} = 2E = \langle k \rangle N$$

- We can use the last relation to find the joint probability:

$$P\left(k,k'\right) = \frac{E_{kk'}}{\langle k \rangle N}$$

- From which we can obtain the degree distribution (since $E_{kk'}$ contains all the connectivity information of the network).

$$P\left(k\right) = \frac{\langle k \rangle}{k} \sum_{k'} P\left(k,k'\right) \propto k^{-\gamma}$$

Degree Correlations

Degree Correlations

- The conditional probability that a node with degree k is connected to a node of degree k' is then:

$$P\left(k'|k\right) = \frac{\langle k \rangle}{k} \frac{P\left(k,k'\right)}{P\left(k\right)}$$

Degree Correlations

- The conditional probability that a node with degree k is connected to a node of degree k' is then:

$$P\left(k'|k\right) = \frac{\langle k \rangle}{k} \frac{P\left(k,k'\right)}{P\left(k\right)}$$

- Using this conditional probability we can easily calculate $\langle k_{nn}\left(k\right)\rangle$ the average degree of the nearest neighbors of a node of degree k

$$\langle k_{nn}\left(k\right)\rangle = \sum_{k'} k' P\left(k'|k\right)$$

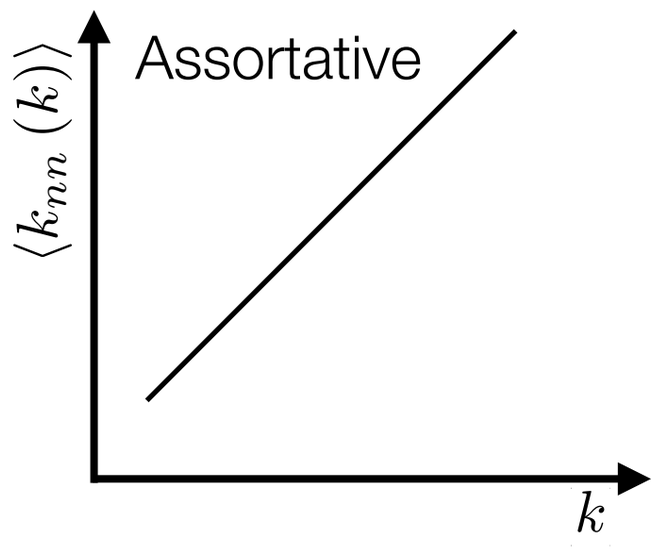
Degree Correlations

- The conditional probability that a node with degree k is connected to a node of degree k' is then:

$$P\left(k'|k\right) = \frac{\langle k \rangle}{k} \frac{P\left(k,k'\right)}{P\left(k\right)}$$

- Using this conditional probability we can easily calculate $\langle k_{nn}\left(k\right)\rangle$ the average degree of the nearest neighbors of a node of degree k

$$\langle k_{nn}\left(k\right)\rangle = \sum_{k'} k' P\left(k'|k\right)$$



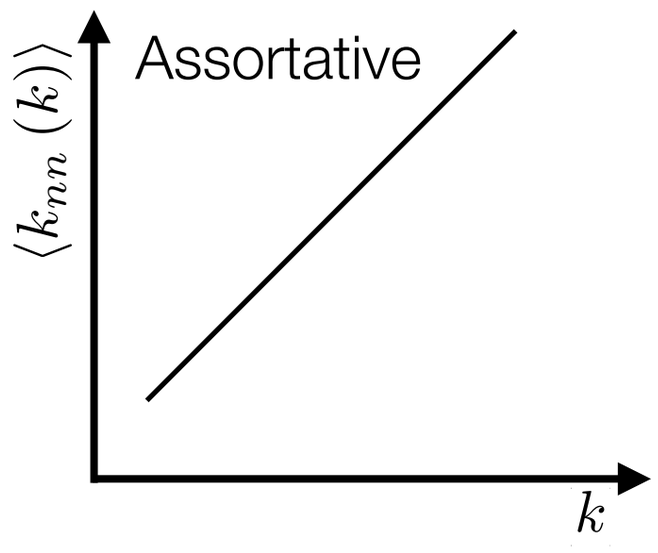
Degree Correlations

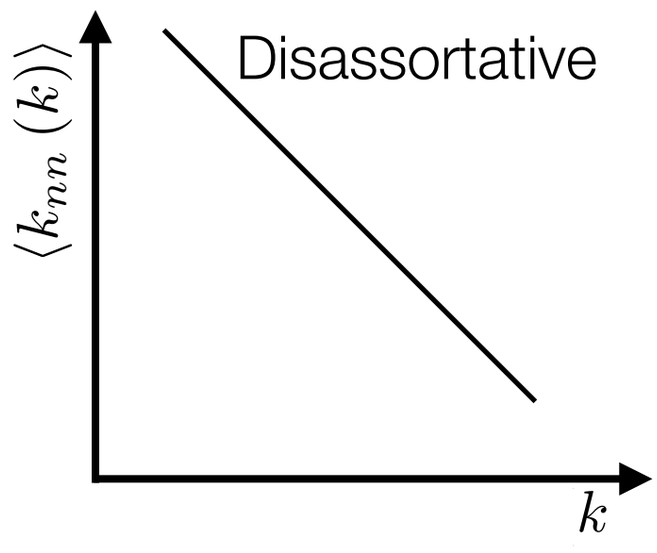
- The conditional probability that a node with degree k is connected to a node of degree k' is then:

$$P\left(k'|k\right) = \frac{\langle k \rangle}{k} \frac{P\left(k,k'\right)}{P\left(k\right)}$$

- Using this conditional probability we can easily calculate $\langle k_{nn}\left(k\right)\rangle$ the average degree of the nearest neighbors of a node of degree k

$$\langle k_{nn}\left(k\right)\rangle = \sum_{k'} k' P\left(k'|k\right)$$





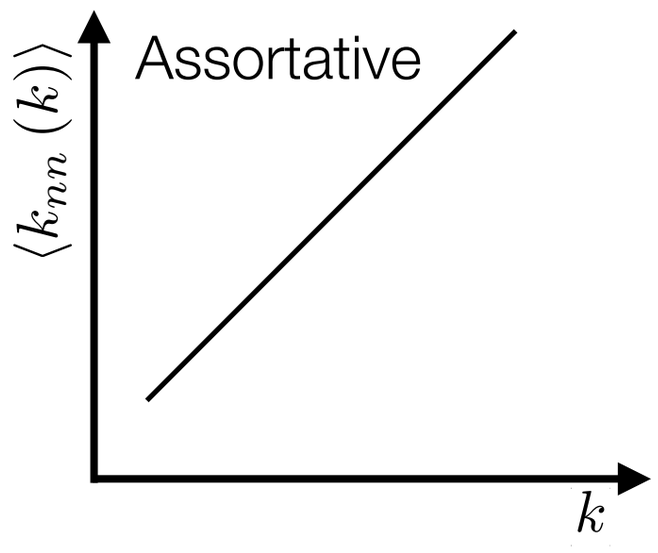
Degree Correlations

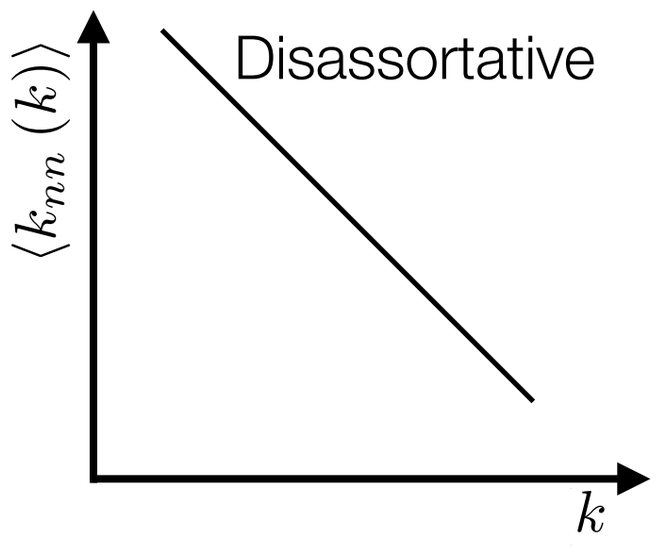
- The conditional probability that a node with degree k is connected to a node of degree k' is then:

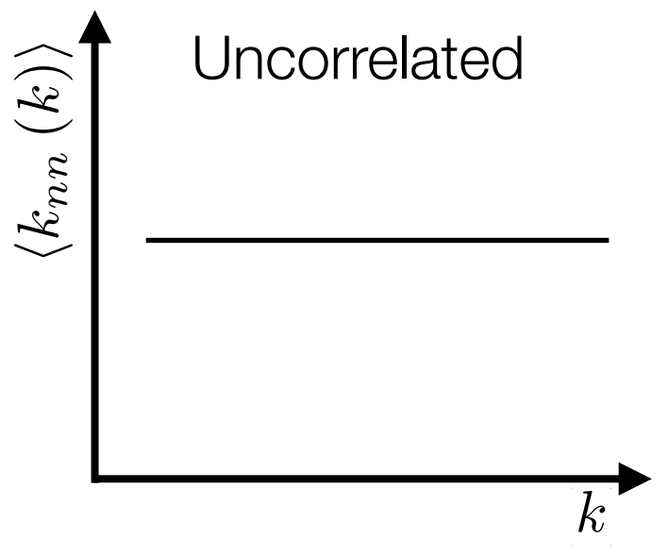
$$P\left(k'|k\right) = \frac{\langle k \rangle}{k} \frac{P\left(k,k'\right)}{P\left(k\right)}$$

- Using this conditional probability we can easily calculate $\langle k_{nn}\left(k\right)\rangle$ the average degree of the nearest neighbors of a node of degree k

$$\langle k_{nn}\left(k\right)\rangle = \sum_{k'} k' P\left(k'|k\right)$$







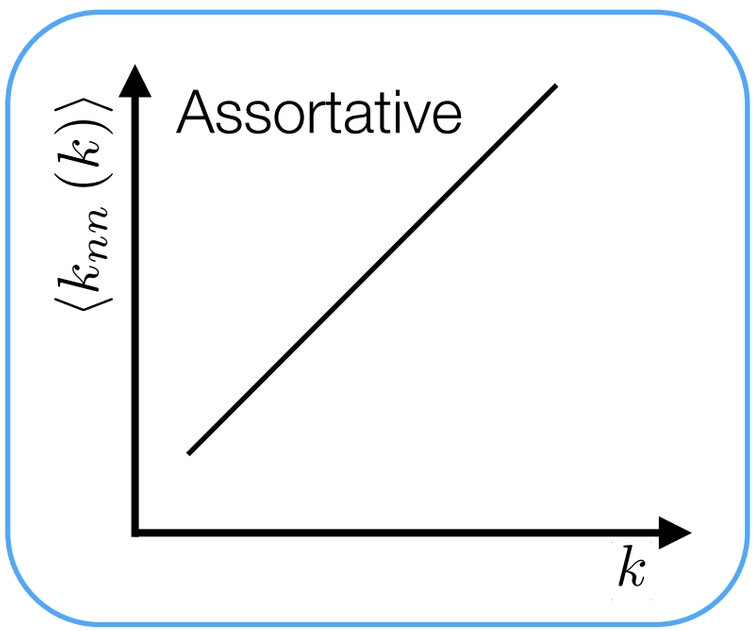
Degree Correlations

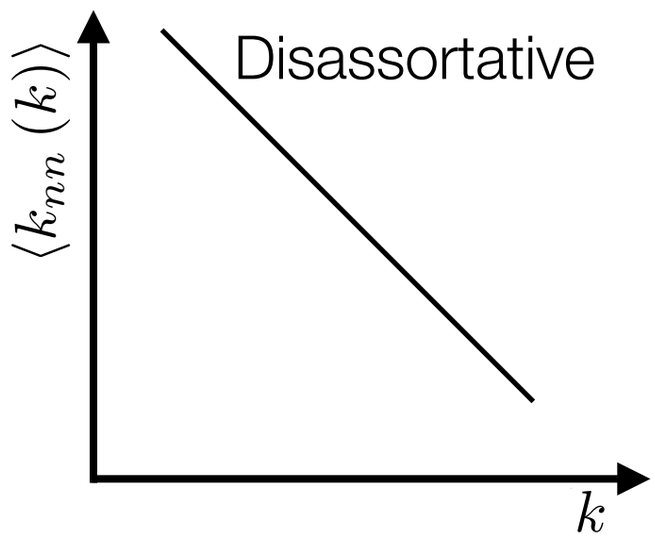
- The conditional probability that a node with degree k is connected to a node of degree k' is then:

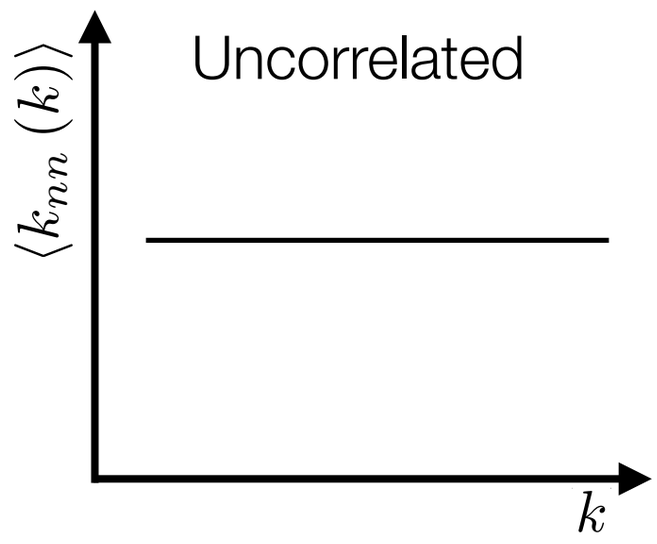
$$P\left(k'|k\right) = \frac{\langle k \rangle}{k} \frac{P\left(k,k'\right)}{P\left(k\right)}$$

- Using this conditional probability we can easily calculate $\langle k_{nn}\left(k\right)\rangle$ the average degree of the nearest neighbors of a node of degree k

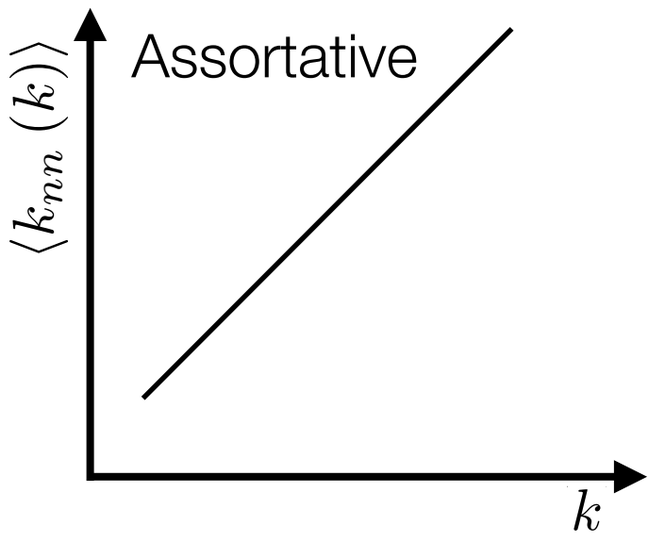
$$\langle k_{nn}\left(k\right)\rangle = \sum_{k'} k' P\left(k'|k\right)$$



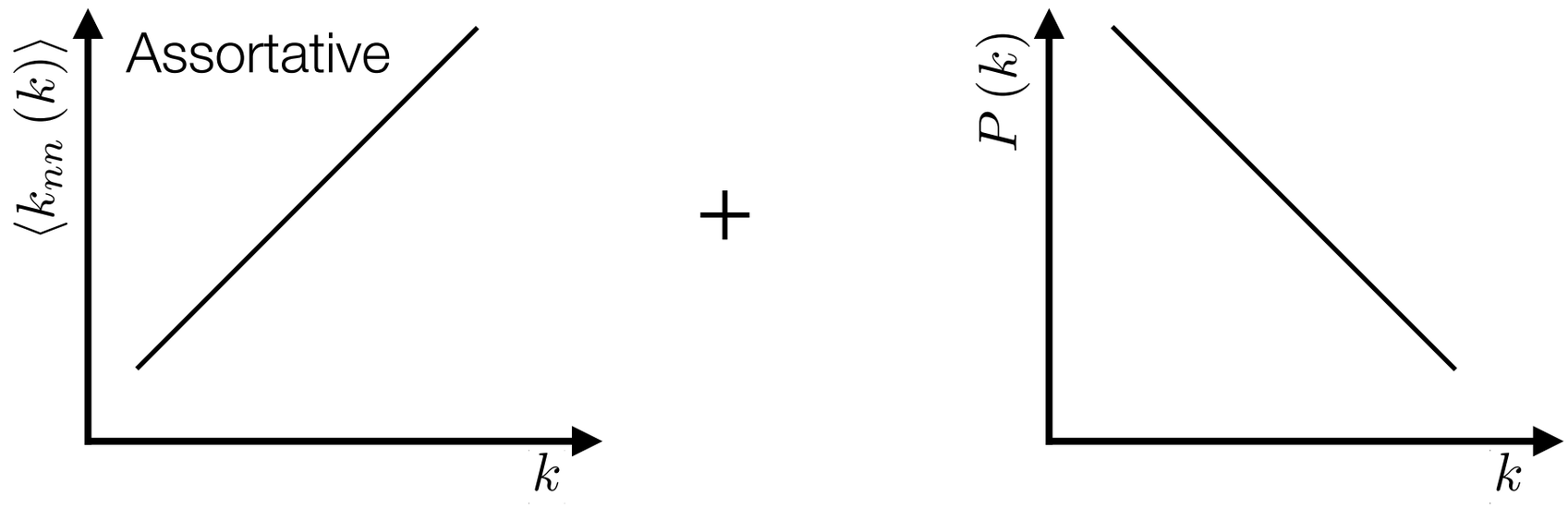




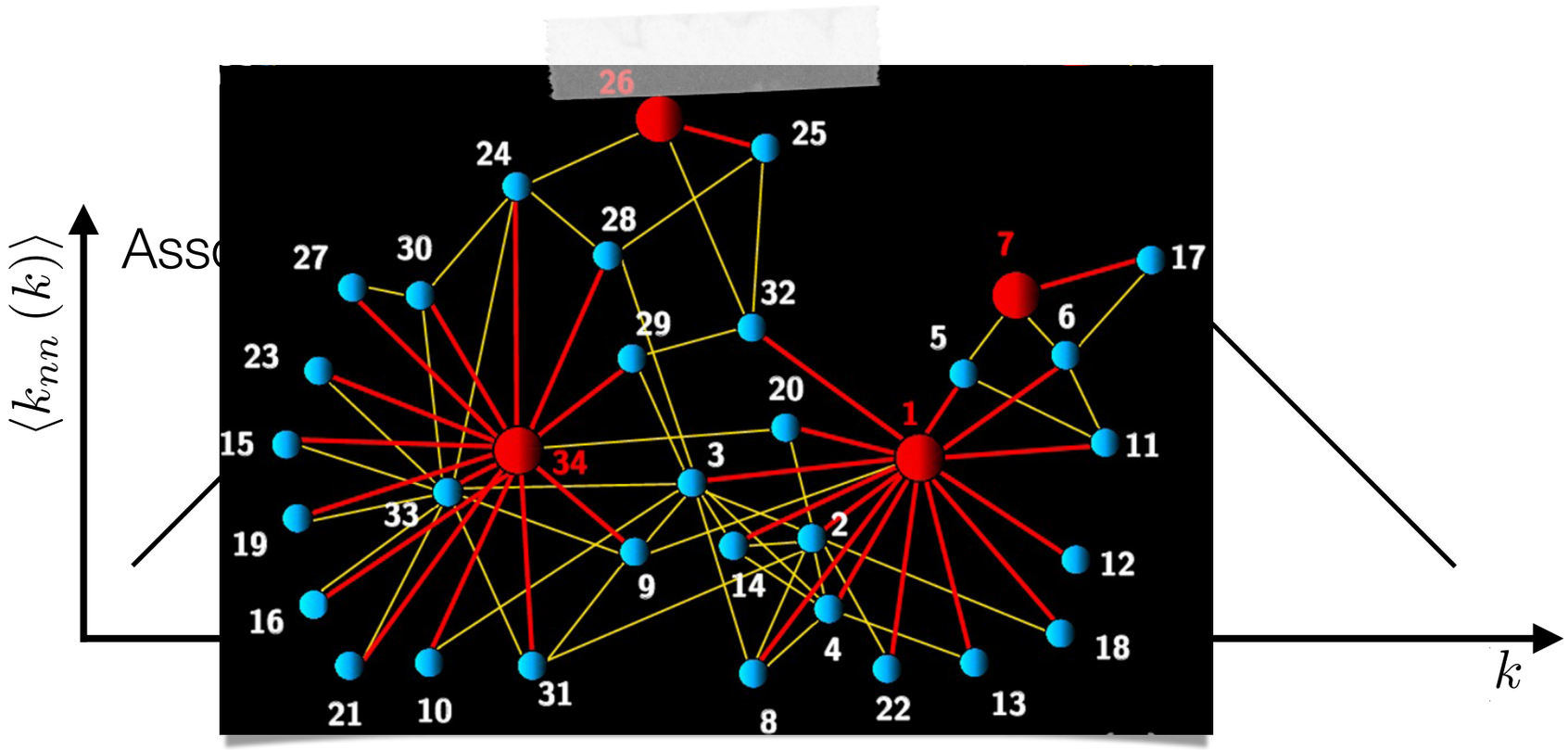
Friendship Paradox



Friendship Paradox



Friendship Paradox



Your Friends Have More Friends Than You

Weight Disparity Measure

Weight Disparity Measure

- Quantifies the local heterogeneity of the weights of a given node. Defined by:

$$\mathcal{Y}_i\left(k\right) = k\sum_j p_{ij}^2$$

Weight Disparity Measure

- Quantifies the local heterogeneity of the weights of a given node. Defined by:

$$\mathcal{Y}_i\left(k\right) = k\sum_j p_{ij}^2$$

- where:

$$p_{ij} = \frac{w_{ij}}{\sum_i w_{ij}}$$

Weight Disparity Measure

- Quantifies the local heterogeneity of the weights of a given node. Defined by:

$$\mathcal{Y}_i\left(k\right) = k\sum_j p_{ij}^2$$

- where:

$$p_{ij} = \frac{w_{ij}}{\sum_i w_{ij}}$$

- are the normalized weights.

Weight Disparity Measure

- Quantifies the local heterogeneity of the weights of a given node. Defined by:

$$\mathcal{Y}_i\left(k\right) = k\sum_j p_{ij}^2$$

- where:

$$p_{ij} = \frac{w_{ij}}{\sum_i w_{ij}}$$

- are the normalized weights.

- If all edges have same weight: $\mathcal{Y}_i\left(k\right) = 1$

Weight Disparity Measure

- Quantifies the local heterogeneity of the weights of a given node. Defined by:

$$\mathcal{Y}_i\left(k\right) = k\sum_j p_{ij}^2$$

- where:

$$p_{ij} = \frac{w_{ij}}{\sum_i w_{ij}}$$

- are the normalized weights.

- If all edges have same weight: $\mathcal{Y}_i\left(k\right) = 1$

- If one edge is dominant: $\mathcal{Y}_i\left(k\right) \sim k$

Weight Disparity Measure

- Quantifies the local heterogeneity of the weights of a given node. Defined by:

$$\mathcal{Y}_i\left(k\right) = k\sum_j p_{ij}^2$$

- where:

$$p_{ij} = \frac{w_{ij}}{\sum_i w_{ij}}$$

- are the normalized weights.

- If all edges have same weight: $\mathcal{Y}_i\left(k\right) = 1$

- If one edge is dominant: $\mathcal{Y}_i\left(k\right) \sim k$

- And typically: $\mathcal{Y}_i\left(k\right) \propto k^{\alpha}$

How to get Social Network Data

Social Network Data - Interviews and Questionnaires

Social Network Data - Interviews and Questionnaires

- - "If you want to know, ask"

Social Network Data - Interviews and Questionnaires

- - "If you want to know, ask"
    - Telephone surveys

Social Network Data - Interviews and Questionnaires

- - "If you want to know, ask"
    - Telephone surveys
    - Polling

Social Network Data - Interviews and Questionnaires

- - "If you want to know, ask"
    - Telephone surveys
    - Polling
    - Mechanical Turk

Social Network Data - Interviews and Questionnaires

- - "If you want to know, ask"
    - Telephone surveys
    - Polling
    - Mechanical Turk
    - etc

Social Network Data - Interviews and Questionnaires

Social Network Data - Interviews and Questionnaires

- Cons:

Social Network Data - Interviews and Questionnaires

- Cons:
    - Misunderstanding and inconsistent interpretation of questions

Social Network Data - Interviews and Questionnaires

- Cons:
    - Misunderstanding and inconsistent interpretation of questions
    - Incomplete sampling

Social Network Data - Interviews and Questionnaires

- Cons:
    - Misunderstanding and inconsistent interpretation of questions
    - Incomplete sampling
    - Perturbative

Social Network Data - Interviews and Questionnaires

- Cons:
    - Misunderstanding and inconsistent interpretation of questions
    - Incomplete sampling
    - Perturbative
    - Biased

Social Network Data - Interviews and Questionnaires

- Cons:
    - Misunderstanding and inconsistent interpretation of questions
    - Incomplete sampling
    - Perturbative
    - Biased
    - Costly

Social Network Data - Interviews and Questionnaires

- Cons:
    - Misunderstanding and inconsistent interpretation of questions
    - Incomplete sampling
    - Perturbative
    - Biased
    - Costly
    - Potentially asymmetric responses (A says B is friend. B says A is not friend)

Social Network Data - Interviews and Questionnaires

Social Network Data - Interviews and Questionnaires

- Pros:

Social Network Data - Interviews and Questionnaires

- Pros:
    - Long tradition in Social Science

Social Network Data - Interviews and Questionnaires

- Pros:
    - Long tradition in Social Science
    - Sophisticated techniques to obtain good statistics with limited sampling

Social Network Data - Interviews and Questionnaires

- Pros:
    - Long tradition in Social Science
    - Sophisticated techniques to obtain good statistics with limited sampling
        - In the case of interviews, the interviewer can help clarify the question or the answer

Social Network Data - Interviews and Questionnaires

- Pros:
    - Long tradition in Social Science
    - Sophisticated techniques to obtain good statistics with limited sampling
    - In the case of interviews, the interviewer can help clarify the question or the answer
    - Very detailed information on each respondent

Social Network Data - Archival Records/Proxy Data

Social Network Data - Archival Records/Proxy Data

- Phone call records

Social Network Data - Archival Records/Proxy Data

- Phone call records
- Declared OSN friendships (Facebook, Twitter, etc…)

Social Network Data - Archival Records/Proxy Data

- Phone call records
- Declared OSN friendships (Facebook, Twitter, etc…)
- Bibliographic Records

Social Network Data - Archival Records/Proxy Data

- Phone call records
- Declared OSN friendships (Facebook, Twitter, etc…)
- Bibliographic Records
- Pros:

Social Network Data - Archival Records/Proxy Data

- Phone call records
- Declared OSN friendships (Facebook, Twitter, etc…)
- Bibliographic Records
- Pros:
    - Potentially much larger sizes

Social Network Data - Archival Records/Proxy Data

- Phone call records
- Declared OSN friendships (Facebook, Twitter, etc…)
- Bibliographic Records
- Pros:
    - Potentially much larger sizes
    - Less prone to misunderstanding and inconsistent interpretations

Social Network Data - Archival Records/Proxy Data

- Phone call records
- Declared OSN friendships (Facebook, Twitter, etc…)
- Bibliographic Records
- Pros:
    - Potentially much larger sizes
    - Less prone to misunderstanding and inconsistent interpretations
    - Much less costly

Social Network Data - Archival Records/Proxy Data

- Phone call records
- Declared OSN friendships (Facebook, Twitter, etc…)
- Bibliographic Records
- Pros:
    - Potentially much larger sizes
    - Less prone to misunderstanding and inconsistent interpretations
    - Much less costly
    - Applicable to historical records (Florentine weddings)

Social Network Data - Archival Records/Proxy Data

Social Network Data - Archival Records/Proxy Data

- Cons:

Social Network Data - Archival Records/Proxy Data

- Cons:
    - Significantly less information available for each individual

Social Network Data - Archival Records/Proxy Data

- Cons:
    - Significantly less information available for each individual
    - Limited by record availability

Social Network Data - Archival Records/Proxy Data

- Cons:
    - Significantly less information available for each individual
    - Limited by record availability
    - Potential self-selection bias (Are people who use Facebook the same as people who don't?)

Social Network Data - Affiliation Networks

Social Network Data - Affiliation Networks

- Edges represent co-membership into some group:

Social Network Data - Affiliation Networks

- Edges represent co-membership into some group:
    - the group of authors of a paper - co-authorship networks

Social Network Data - Affiliation Networks

- Edges represent co-membership into some group:
    - the group of authors of a paper - co-authorship networks
    - the group of actors in a movie - co-staring networks

Social Network Data - Affiliation Networks

- Edges represent co-membership into some group:
    - the group of authors of a paper - co-authorship networks
    - the group of actors in a movie - co-staring networks
        - Usually projections of bi-partite networks

Social Network Data - Ego Networks

Social Network Data - Ego Networks

- Use Interviews or questionnaires

Social Network Data - Ego Networks

- Use Interviews or questionnaires

- focus on specific individual (ego) instead of (social) network structure

Social Network Data - Ego Networks

- Use Interviews or questionnaires
- focus on specific individual (ego) instead of (social) network structure
- Elimination of network structure results in higher respect for privacy and anonymity

Social Network Data - Ego Networks

- Use Interviews or questionnaires
- focus on specific individual (ego) instead of (social) network structure
- Elimination of network structure results in higher respect for privacy and anonymity
- Can't use knowledge of network structure to de-anonymize dataset

Social Network Data - Ego Networks

- Use Interviews or questionnaires
- focus on specific individual (ego) instead of (social) network structure
- Elimination of network structure results in higher respect for privacy and anonymity
  - Can't use knowledge of network structure to de-anonymize dataset
  - Common in public data releases

NetworkX

https://networkx.github.io/

NetworkX

https://networkx.github.io/

- High productivity software for complex networks

NetworkX

https://networkx.github.io/

- High productivity software for complex networks
- Simple Python interface

NetworkX

https://networkx.github.io/

- High productivity software for complex networks
- Simple Python interface
- Four types of graphs supported:

NetworkX

https://networkx.github.io/

- High productivity software for complex networks
- Simple Python interface
- Four types of graphs supported:
    - Graph - UnDirected

NetworkX

https://networkx.github.io/

- High productivity software for complex networks
- Simple Python interface
- Four types of graphs supported:
    - Graph - UnDirected
    - DiGraph - Directed

NetworkX

https://networkx.github.io/

- High productivity software for complex networks
- Simple Python interface
- Four types of graphs supported:
    - Graph - UnDirected
    - DiGraph - Directed
    - MultiGraph - Multi-edged Graph

NetworkX

https://networkx.github.io/

- High productivity software for complex networks
- Simple Python interface
- Four types of graphs supported:
    - Graph - UnDirected
    - DiGraph - Directed
    - MultiGraph - Multi-edged Graph
    - MultiDiGraph - Directed Multigraph

NetworkX

https://networkx.github.io/

- High productivity software for complex networks
- Simple Python interface
- Four types of graphs supported:
    - Graph - UnDirected
    - DiGraph - Directed
    - MultiGraph - Multi-edged Graph
    - MultiDiGraph - Directed Multigraph
- Similar interface for all types of graphs

NetworkX

https://networkx.github.io/

- High productivity software for complex networks
- Simple Python interface
- Four types of graphs supported:
    - Graph - UnDirected
    - DiGraph - Directed
    - MultiGraph - Multi-edged Graph
    - MultiDiGraph - Directed Multigraph
- Similar interface for all types of graphs
- Nodes can be any type of Python object - Practical way to manage relationships

Growing Graphs

Growing Graphs

- .add_node(node_id) Add a single node with ID node_id

Growing Graphs

- .add_node(node_id) Add a single node with ID node_id

- .add_nodes_from() Add a list of node ids

Growing Graphs

- .add_node(node_id) Add a single node with ID node_id
- .add_nodes_from() Add a list of node ids
- .add_edge(node_i, node_j) Adds an edge between node_i and node_j

Growing Graphs

- .add_node(node_id) Add a single node with ID node_id
- .add_nodes_from() Add a list of node ids
- .add_edge(node_i, node_j) Adds an edge between node_i and node_j
- .add_edges_from() Adds a list of edges. Individual edges are represented by tuples

Growing Graphs

- .add_node(node_id) Add a single node with ID node_id
- .add_nodes_from() Add a list of node ids
- .add_edge(node_i, node_j) Adds an edge between node_i and node_j
- .add_edges_from() Adds a list of edges. Individual edges are represented by tuples
- .remove_node(node_id)/.remove_nodes_from() Removing a node removes all associated edges

Growing Graphs

- .add_node(node_id) Add a single node with ID node_id
- .add_nodes_from() Add a list of node ids
- .add_edge(node_i, node_j) Adds an edge between node_i and node_j
- .add_edges_from() Adds a list of edges. Individual edges are represented by tuples
- .remove_node(node_id)/.remove_nodes_from() Removing a node removes all associated edges
- .remove_edge(node_i, node_j)/.remove_edges_from()

Graph Properties

Graph Properties

- **.nodes()** Returns the list of nodes

Graph Properties

- .nodes() Returns the list of nodes

- .edges() Returns the list of edges

Graph Properties

- .nodes() Returns the list of nodes

- .edges() Returns the list of edges

- .degree() Returns a dict with each nodes degree .in_degree()/.out_degree() returns dicts with in/out degree for DiGraphs

Graph Properties

- .nodes() Returns the list of nodes

- .edges() Returns the list of edges

- .degree() Returns a dict with each nodes degree .in_degree()/.out_degree() returns dicts with in/out degree for DiGraphs

- .is_connected() Returns true if the node is connected

Graph Properties

- .nodes() Returns the list of nodes

- .edges() Returns the list of edges

- .degree() Returns a dict with each nodes degree .in_degree()/.out_degree() returns dicts with in/out degree for DiGraphs

- .is_connected() Returns true if the node is connected

- .is_weakly_connected()/.is_strongly_connected() for DiGraph

Graph Properties

- .nodes() Returns the list of nodes

- .edges() Returns the list of edges

- .degree() Returns a dict with each nodes degree .in_degree()/.out_degree() returns dicts with in/out degree for DiGraphs

- .is_connected() Returns true if the node is connected

- .is_weakly_connected()/.is_strongly_connected() for DiGraph

- .connected_components() A list of nodes for each connected component

Social Network Models

Erdos-Renyi Model

Erdos-Renyi Model

- Each possible edge is present with probability p. So the average number of edges is:

$$E = \frac{p}{2} N \left(N - 1 \right)$$

Erdos-Renyi Model

- Each possible edge is present with probability p. So the average number of edges is:

$$E = \frac{p}{2} N \left(N - 1 \right)$$

- And the average degree is:

$$\langle k \rangle = \frac{2E}{N} = p \left(N - 1 \right)$$

Erdos-Renyi Model

- Each possible edge is present with probability p. So the average number of edges is:

$$E = \frac{p}{2} N \left(N - 1 \right)$$

- And the average degree is:

$$\langle k \rangle = \frac{2E}{N} = p \left(N - 1 \right)$$

- So to build a network with a given average degree, we need:

$$p = \frac{\langle k \rangle}{N}$$

Erdos-Renyi Model

Erdos-Renyi Model

- For a node to have degree k, it must be chosen as the endpoint of an edge k times and not be chosen exactly N – k times.

$$P(k) = \frac{(N-1)!}{k!\,(N-k-1)!} p^k (1-p)^{N-k-1}$$

Erdos-Renyi Model

- For a node to have degree k, it must be chosen as the endpoint of an edge k times and not be chosen exactly N – k times.

$$P\left(k\right) = \frac{\left(N-1\right)!}{k!\left(N-k-1\right)!}p^{k}\left(1-p\right)^{N-k-1}$$

- Using Sterlings approximation:

$$n! \approx \sqrt{2\pi n}n^{n}e^{-n}$$

Erdos-Renyi Model

- For a node to have degree k, it must be chosen as the endpoint of an edge k times and not be chosen exactly N – k times.

$$P\left(k\right) = \frac{\left(N-1\right)!}{k!\left(N-k-1\right)!}p^{k}\left(1-p\right)^{N-k-1}$$

- Using Sterlings approximation:

$$n! \approx \sqrt{2\pi n}n^{n}e^{-n}$$

- We obtain:

$$P\left(k\right) = \frac{\left\langle k\right\rangle^{k}}{k!}e^{-\left\langle k\right\rangle}$$

Erdos-Renyi Model

- For a node to have degree k, it must be chosen as the endpoint of an edge k times and not be chosen exactly N – k times.

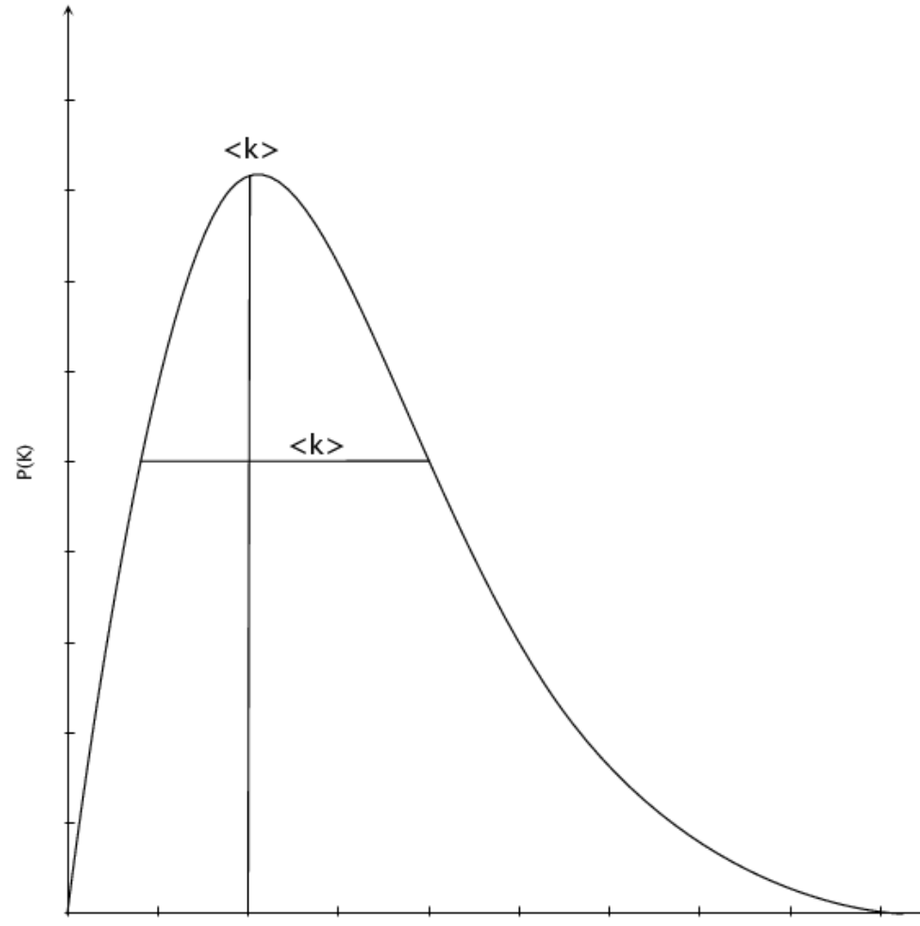
$$P(k) = \frac{(N-1)!}{k!\,(N-k-1)!} p^k (1-p)^{N-k-1}$$

- Using Sterlings approximation:

$$n! \approx \sqrt{2\pi n} n^n e^{-n}$$

- We obtain:

$$P(k) = \frac{\langle k \rangle^k}{k!} e^{-\langle k \rangle}$$



Random Geometric Graphs

Random Geometric Graphs

- Nodes places at random in space (typically 2D).

Random Geometric Graphs

- Nodes places at random in space (typically 2D).
    - uniformly and independently

Random Geometric Graphs

- Nodes places at random in space (typically 2D).
    - uniformly and independently
    - following some distribution

Random Geometric Graphs

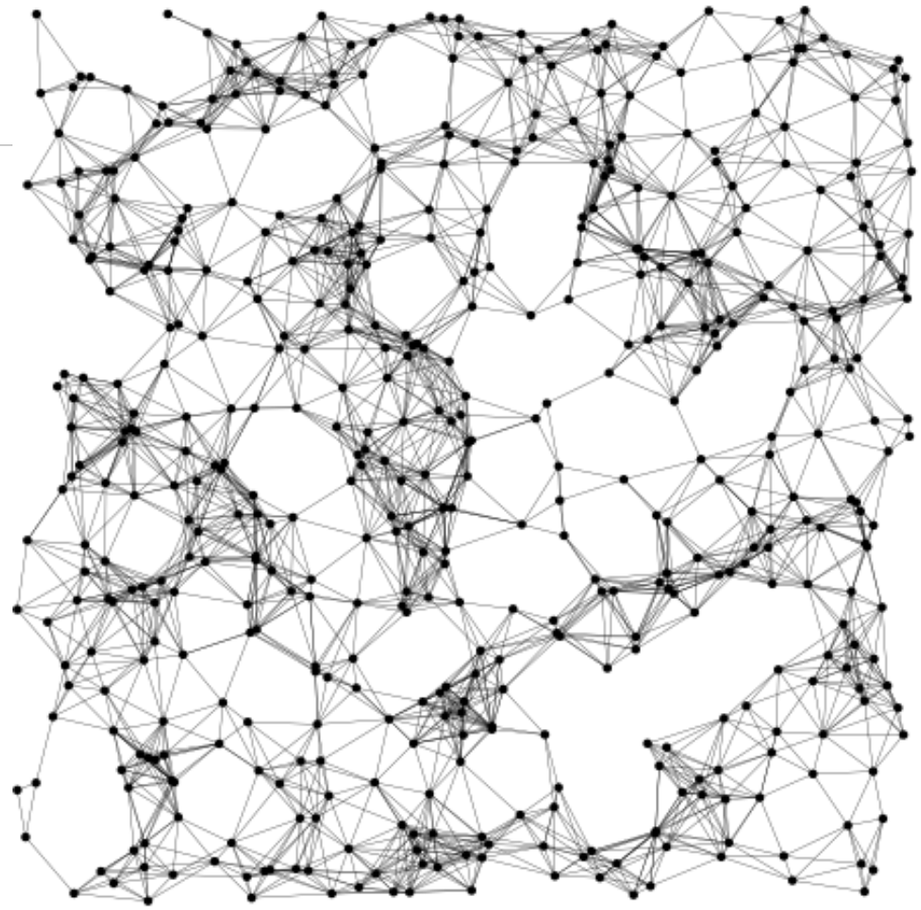
- Nodes places at random in space (typically 2D).
    - uniformly and independently
    - following some distribution
    - by convenience (ad hoc networks)

Random Geometric Graphs

- Nodes places at random in space (typically 2D).
    - uniformly and independently
    - following some distribution
        - by convenience (ad hoc networks)
- Edges are added between any two nodes within a distance d of each other

Random Geometric Graphs

- Nodes places at random in space (typically 2D).
    - uniformly and independently
    - following some distribution
    - by convenience (ad hoc networks)



- Edges are added between any two nodes within a distance d of each other

Random Geometric Graphs

https://networkx.github.io/documentation/networkx-1.10/examples/drawing/random_geometric_graph.html

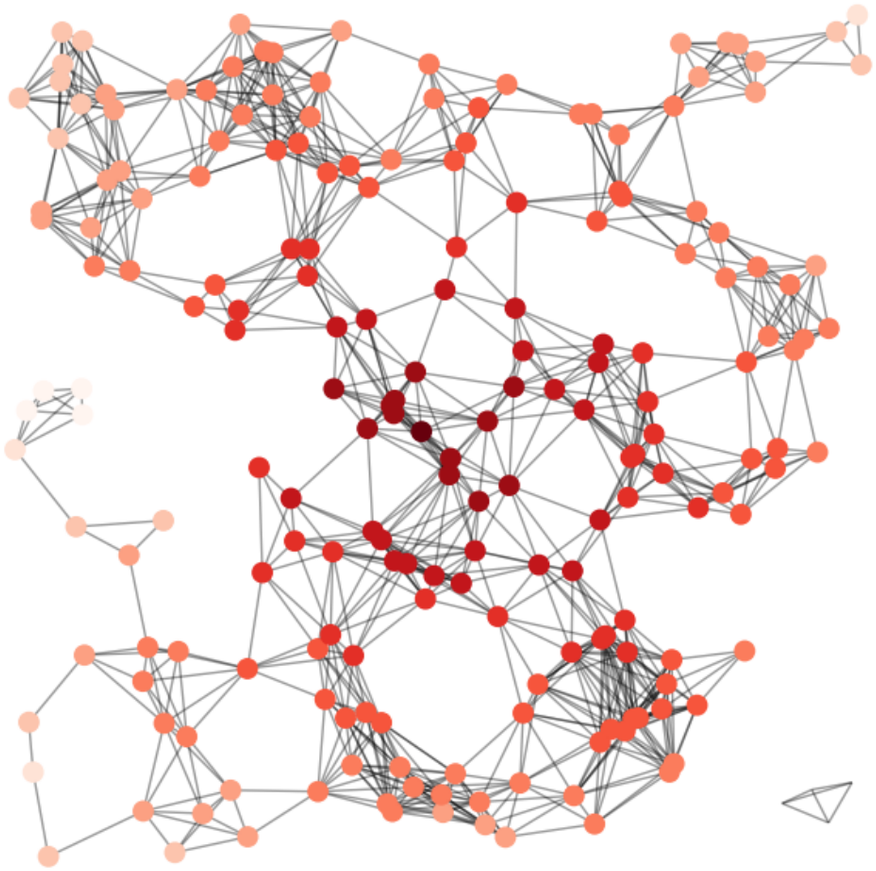
```python
import networkx as nx
import matplotlib.pyplot as plt

G=nx.random_geometric_graph(200,0.125)
# position is stored as node attribute data for
random_geometric_graph
pos=nx.get_node_attributes(G,'pos')

# find node near center (0.5,0.5)
dmin=1
ncenter=0
for n in pos:
    x,y=pos[n]
    d=(x-0.5)**2+(y-0.5)**2
    if d<dmin:
        ncenter=n
        dmin=d

# color by path length from node near center
p=nx.single_source_shortest_path_length(G,ncenter)

plt.figure(figsize=(8,8))
nx.draw_networkx_edges(G,pos,nodelist=[ncenter],alpha=0.4)
nx.draw_networkx_nodes(G,pos,nodelist=list(p.keys()),
                       node_size=80,
                       node_color=list(p.values()),
                       cmap=plt.cm.Reds_r)

plt.xlim(-0.05,1.05)
plt.ylim(-0.05,1.05)
plt.axis('off')
plt.savefig('random_geometric_graph.png')
plt.show()
```

rgg.py

Random Geometric Graphs

https://networkx.github.io/documentation/networkx-1.10/examples/drawing/random_geometric_graph.html

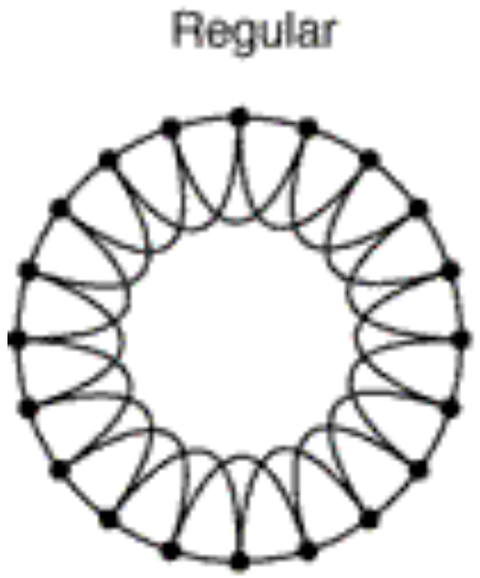
```
import networkx as nx
import matplotlib.pyplot as plt

G=n
# p
ran
pos

# f
dmi
nce
for

# c
p=n

plt
nx.
nx.

plt
plt.ylim(-0.05,1.05)
plt.axis('off')
plt.savefig('random_geometric_graph.png')
plt.show()
```



rgg.py

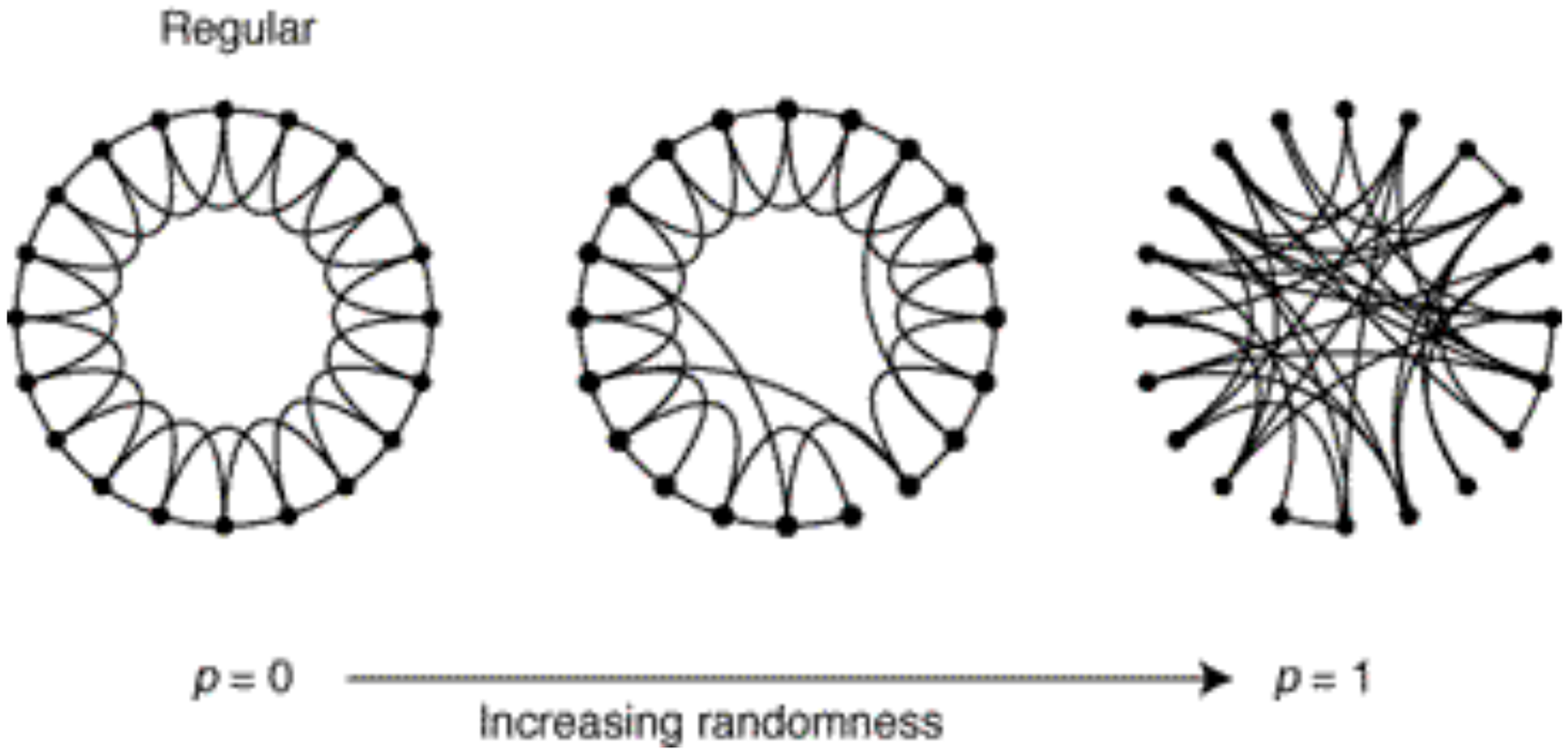
@bgoncalves

Small-World Effect: Watts-Strogatz Model

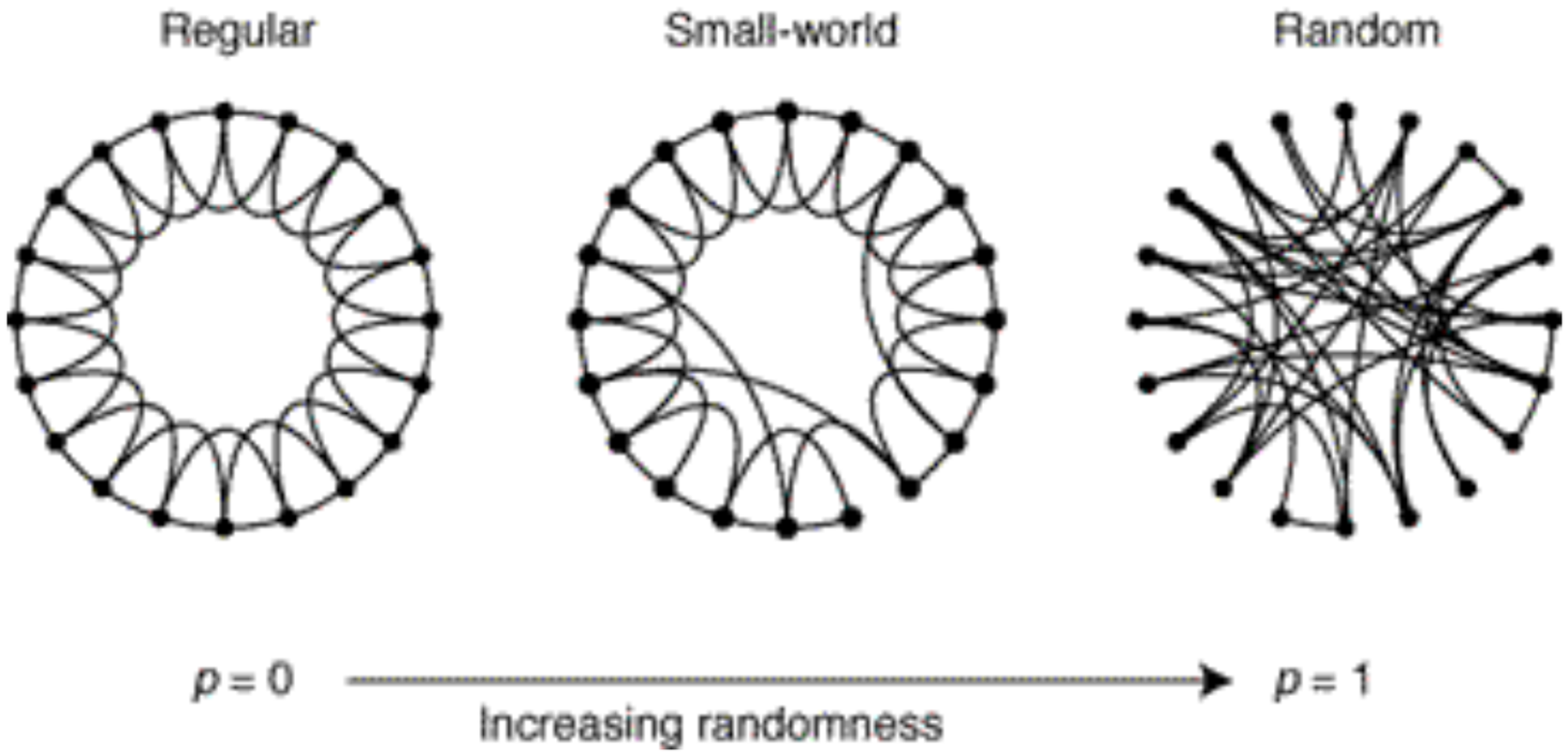


Small-World Effect: Watts-Strogatz Model

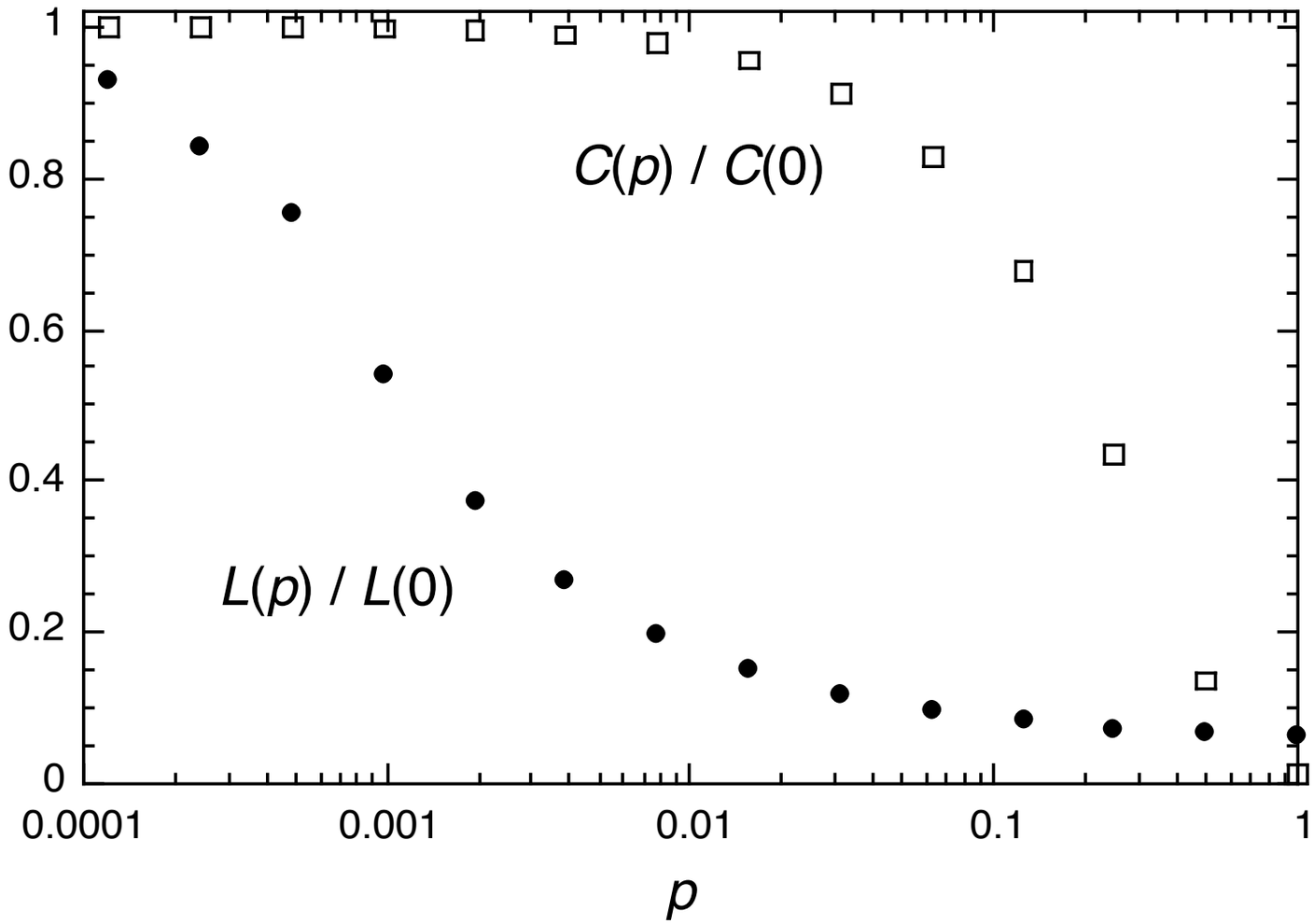
Small-World Effect: Watts-Strogatz Model



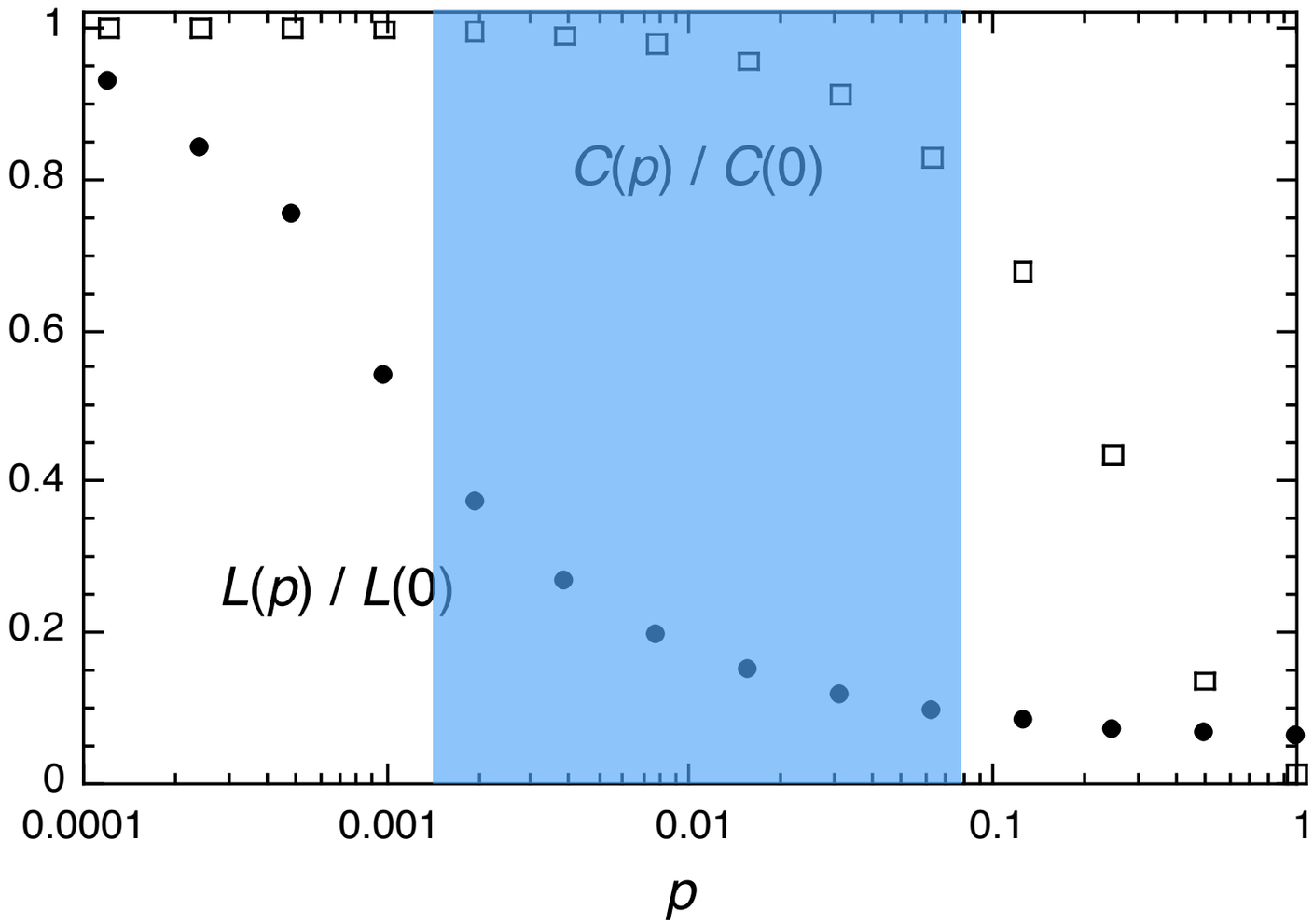
Small-World Effect: Watts-Strogatz Model



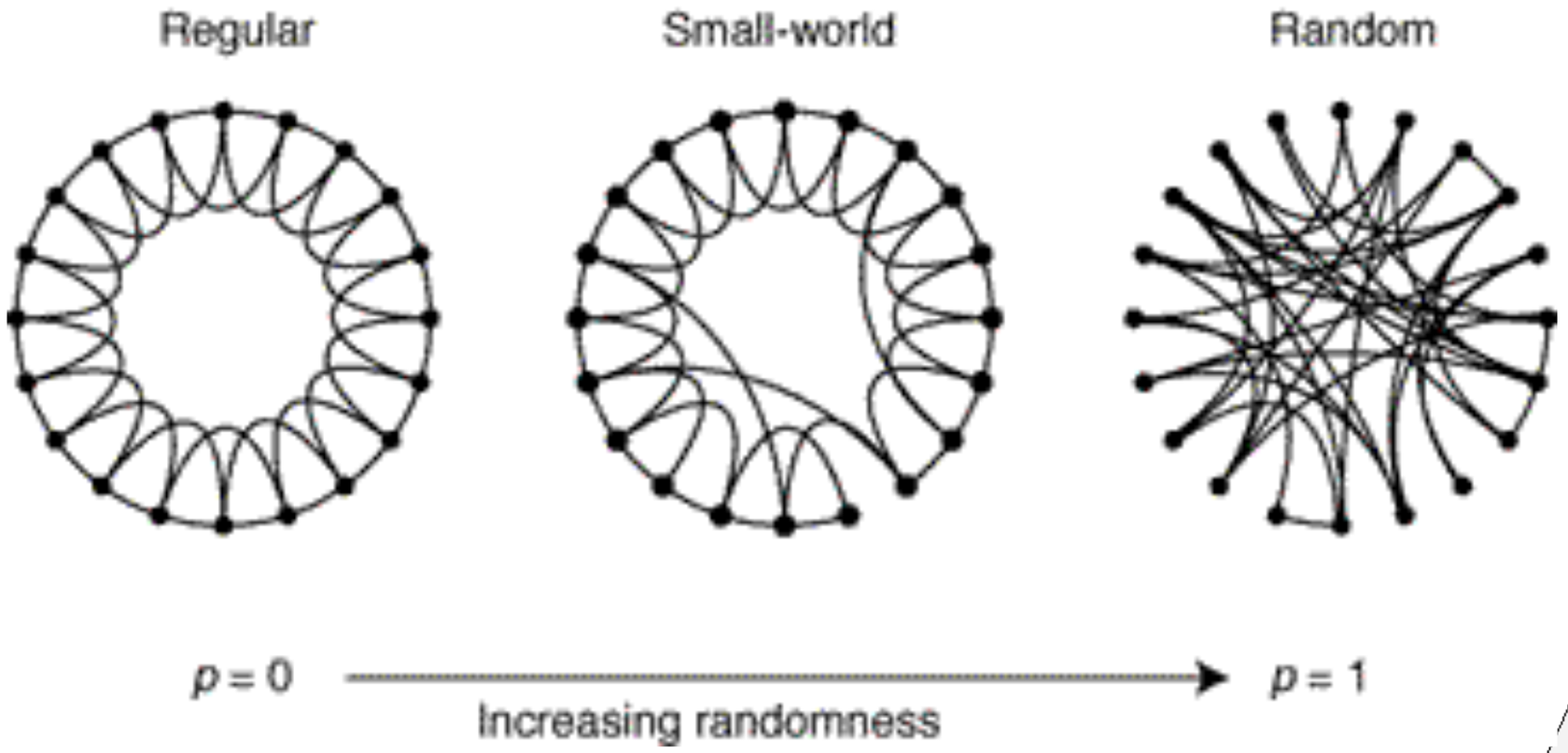
Small-World Effect: Watts-Strogatz Model

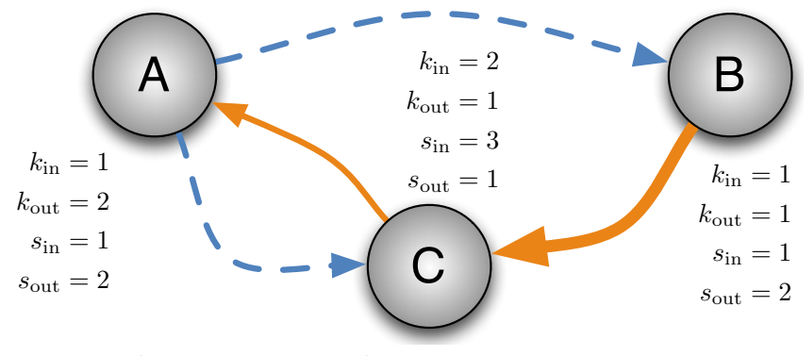


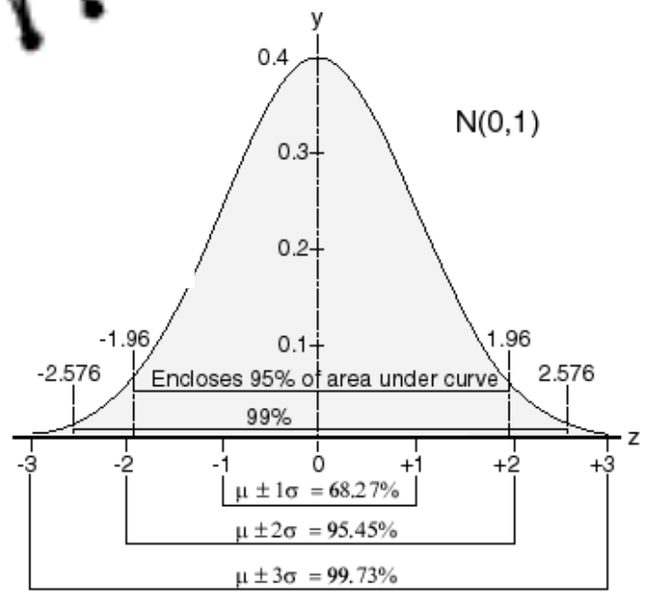
Small-World Effect: Watts-Strogatz Model



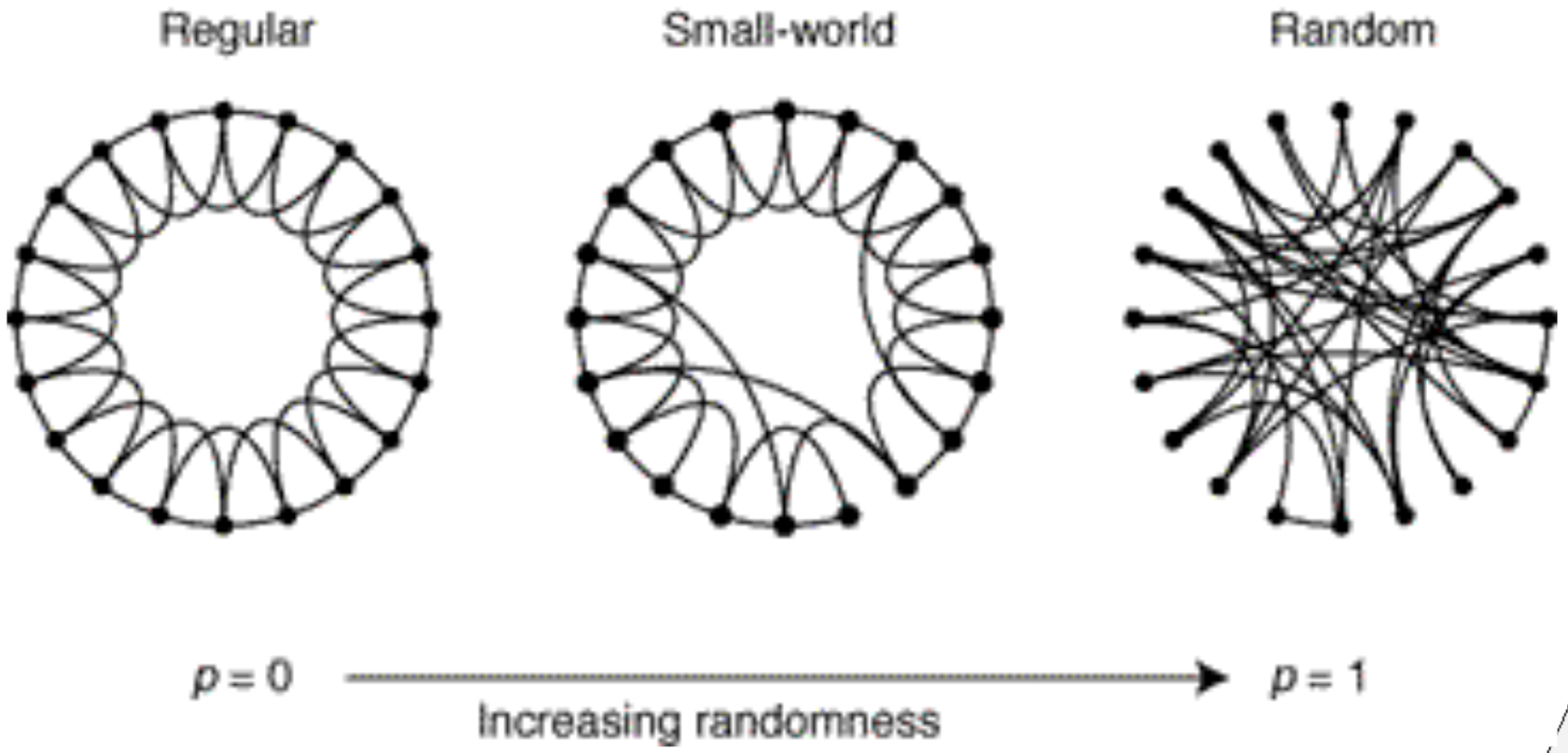
Small-World Effect: Watts-Strogatz Model

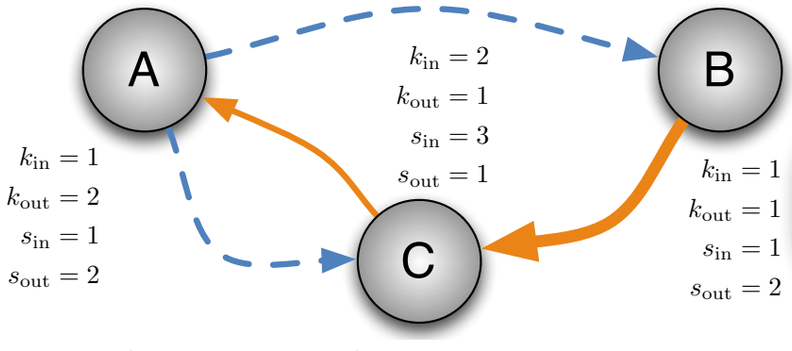




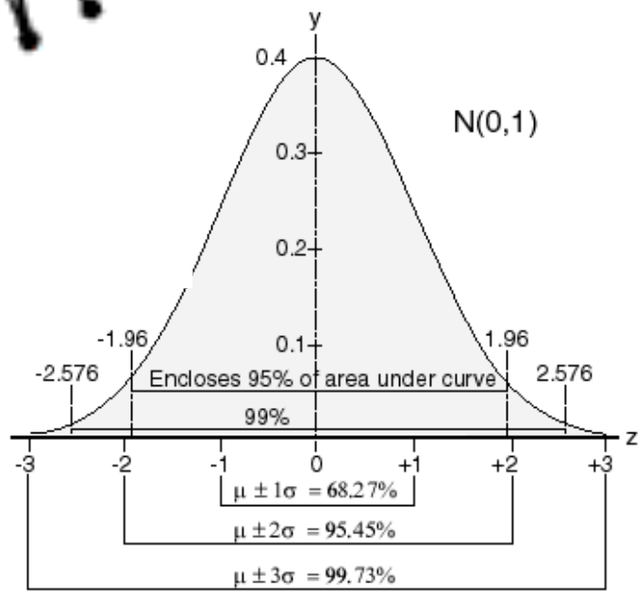


Small-World Effect: Watts-Strogatz Model





No Celebrities!

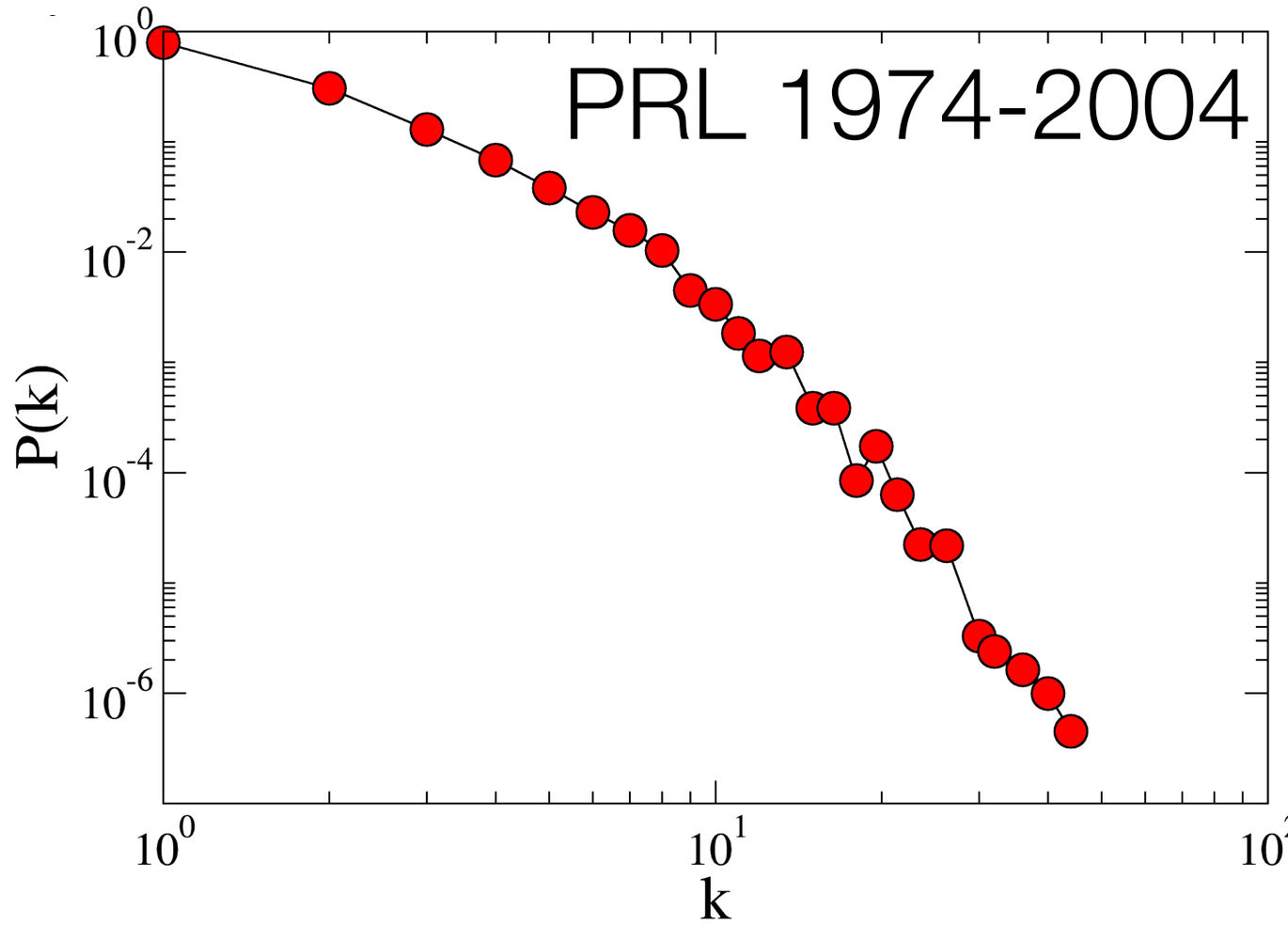


Fat-tailed Degree Distribution

Network or subgraph	Number of vertices	Number of edges	γ	References
Collaboration network of movie actors	212,250	61,085,555	2.3	[55]
'——' (another fitting of the same data)			3.1	[102]
Collaboration network of *Medline*	1,388,989	$1.028\ 10^7$	2.5	[13]
Collaboration net collected from mathematical journals	70,975	0.132×10^6	2.1	[15]
Collaboration net collected from neuro-science journals	209,293	1.214×10^6	2.4	[15]
Web of human sexual contacts[6]	2810	—	3.4	[132]

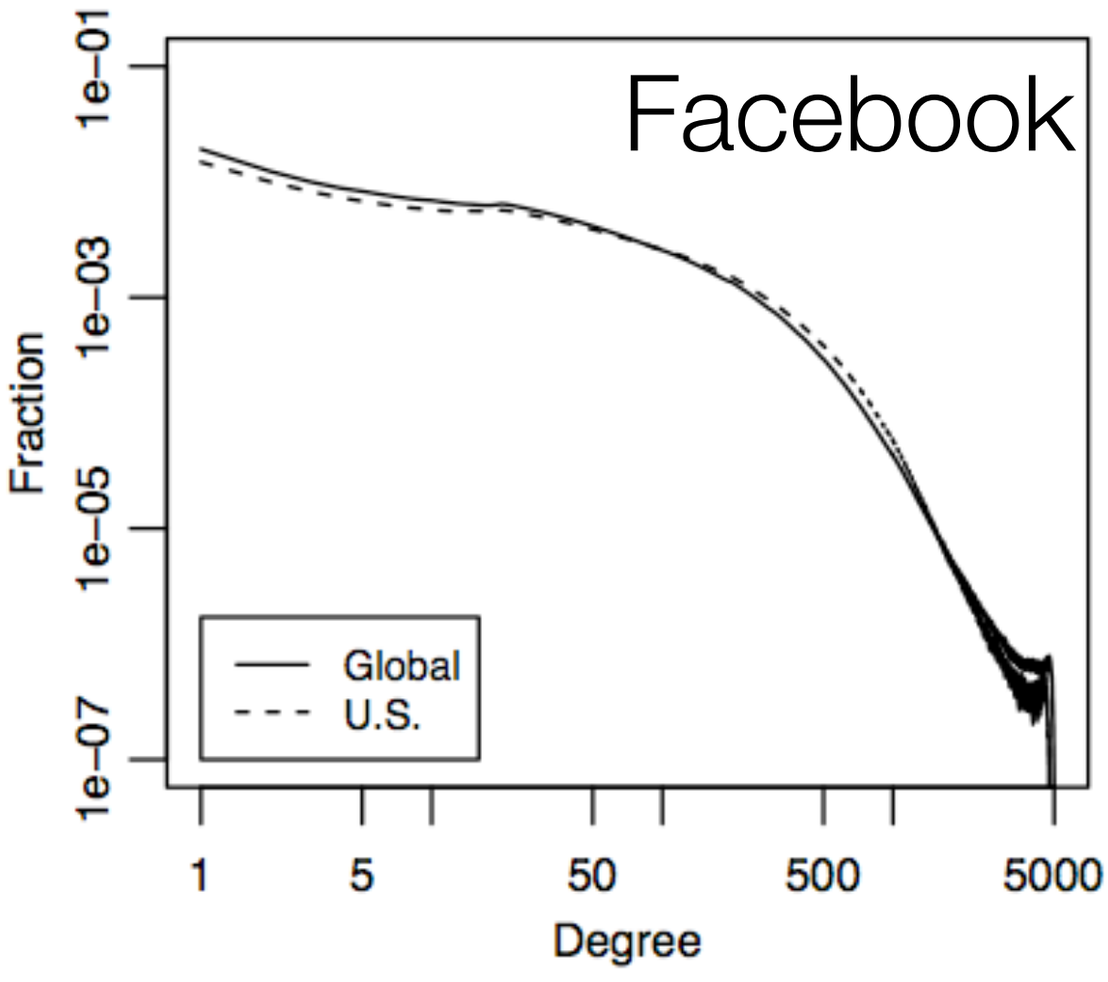
Fat-tailed Degree Distribution

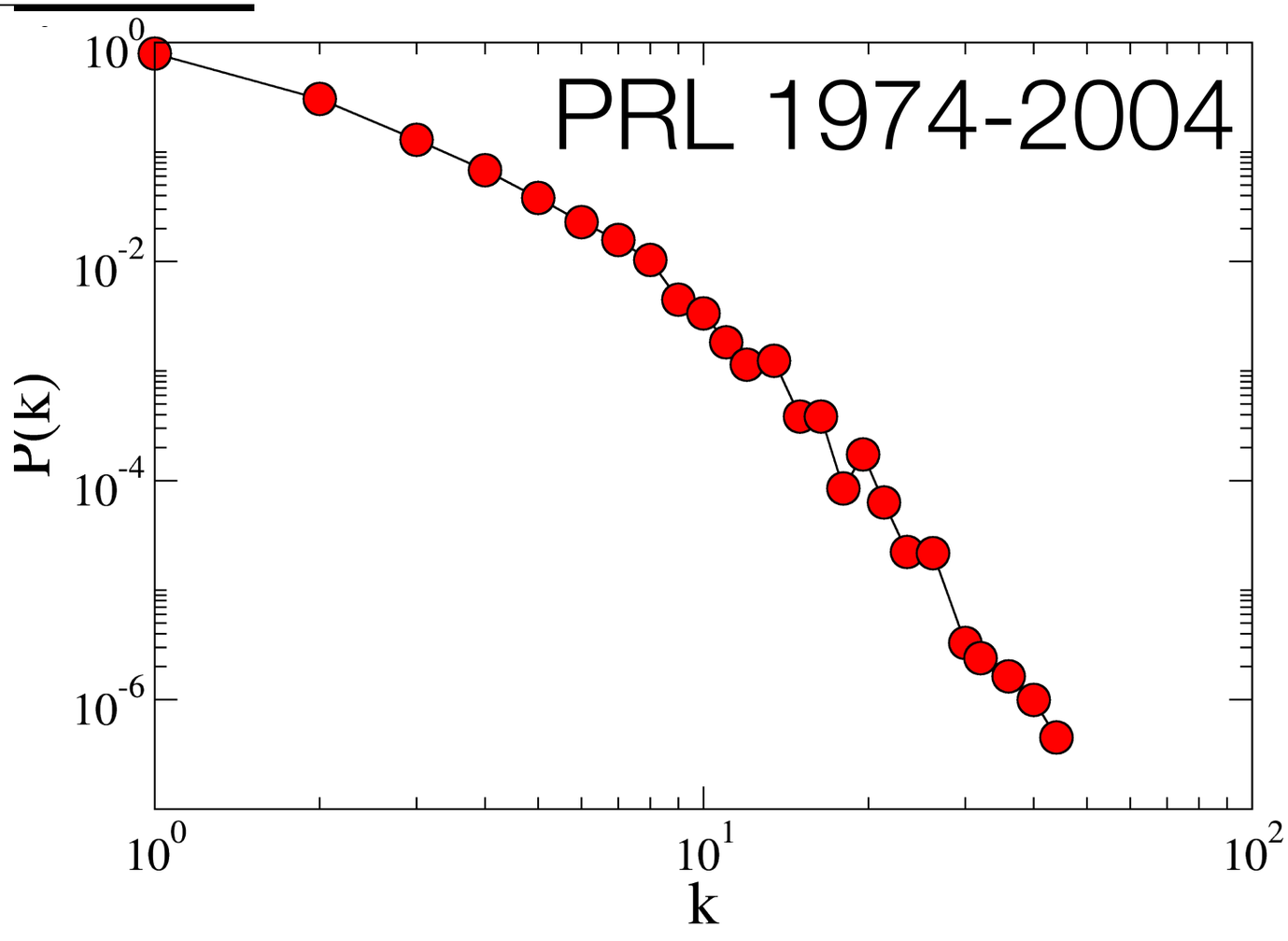
Network or subgraph	Number of vertices	Number of edges	γ	References
Collaboration network of movie actors	212,250	61,085,555	2.3	[55]
'——' (another fitting of the same data)			3.1	[102]
Collaboration network of *Medline*	1,388,989	$1.028\ 10^7$	2.5	[13]
Collaboration net collected from mathematical journals	70,975	0.132×10^6	2.1	[15]
Collaboration net collected from neuro-science journals	209,293	1.214×10^6	2.4	[15]
Web of human sexual contacts[6]	2810	—	3.4	[132]



Fat-tailed Degree Distribution

Network or subgraph	Number of vertices	Number of edges	γ	References
Collaboration network of movie actors	212,250	61,085,555	2.3	[55]
'——' (another fitting of the same data)			3.1	[102]
Collaboration network of *Medline*	1,388,989	$1.028\ 10^7$	2.5	[13]
Collaboration net collected from mathematical journals	70,975	0.132×10^6	2.1	[15]
Collaboration net collected from neuro-science journals	209,293	1.214×10^6	2.4	[15]
Web of human sexual contacts[6]	2810	—	3.4	[132]





Preferential Attachment

Preferential Attachment

- Nodes attract new edges proportionally to their degree

Preferential Attachment

- Nodes attract new edges proportionally to their degree

- Unfair - “oldest” nodes have higher degrees

Preferential Attachment

- Nodes attract new edges proportionally to their degree
- Unfair - "oldest" nodes have higher degrees
- In simplest case, power-law degree distribution with exponent 3

Preferential Attachment

- Nodes attract new edges proportionally to their degree
- Unfair - "oldest" nodes have higher degrees
- In simplest case, power-law degree distribution with exponent 3
- "Younger" nodes connect to "older" nodes

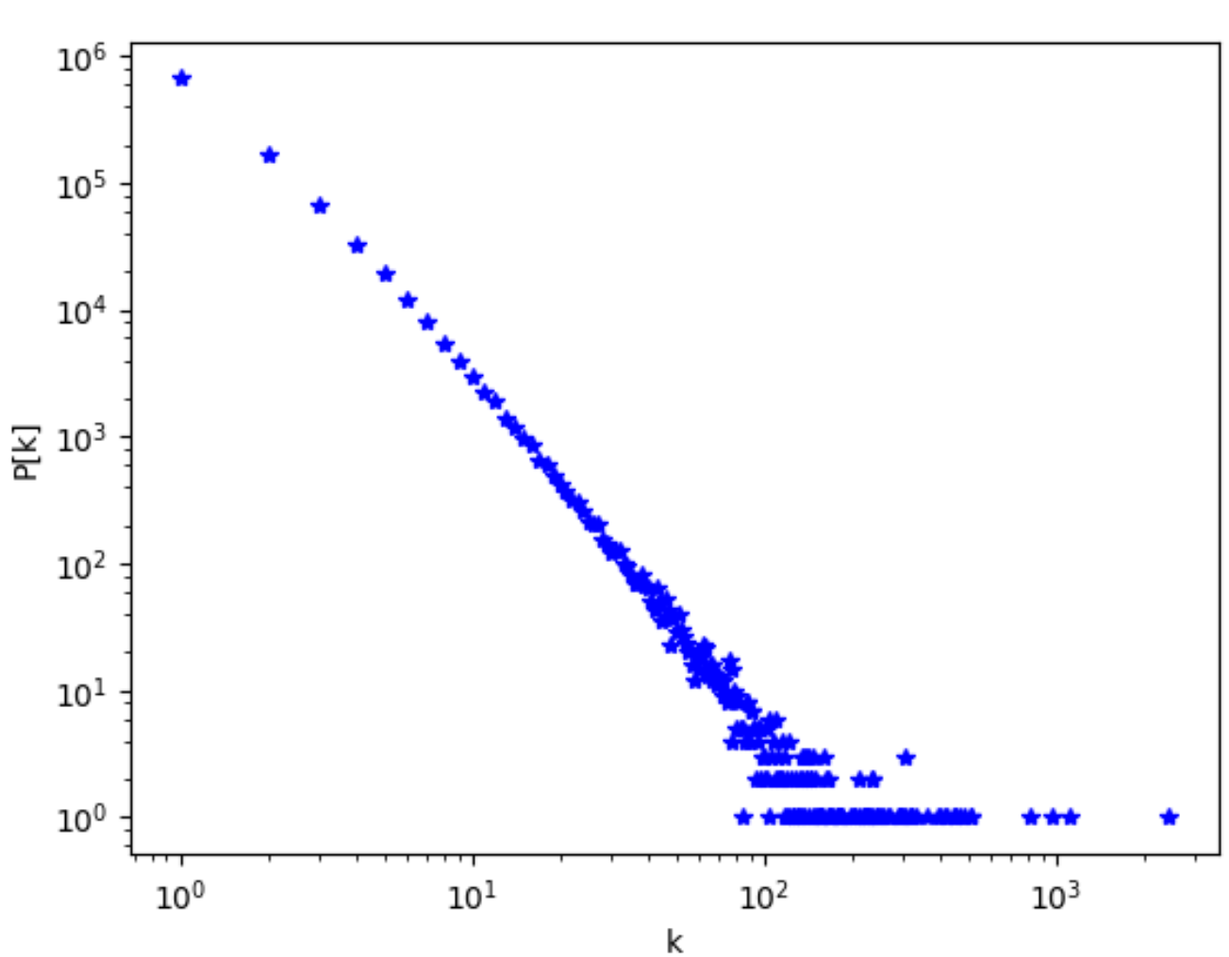
NetworkX - Example

```
import networkx as NX
import numpy as np
from collections import Counter
import matplotlib.pyplot as plt

(…)

net = BarabasiAlbert()

degrees = net.degree()
Pk = np.array(list(Counter(degrees.values()).items()))

plt.loglog(Pk.T[0], Pk.T[1], 'b*')
plt.xlabel('k')
plt.ylabel('P[k]')
plt.savefig('Pk.png')
plt.close()

print("Number of nodes:", net.number_of_nodes())
print("Number of edges:", net.number_of_edges())
```


networkx_demo.py

NetworkX - Example

```
import net
import nur
from colle
import mat

(...)

net = Bara

degrees =
Pk = np.ar

plt.logloc
plt.xlabel
plt.ylabel
plt.savefi
plt.close

print("Nur
print("Nur
```



networkx_demo.py

MLE - Fitting a theoretical function to experimental data

MLE - Fitting a theoretical function to experimental data

- In an experimental measurement, we expect (CLT) the experimental values to be normally distributed around the theoretical value with a certain variance. Mathematically, this means:

$$y - f\left(x\right) \approx \frac{1}{\sqrt{2\sigma^2}} \exp\left[-\frac{\left(y - f\left(x\right)\right)^2}{2\sigma^2}\right]$$

MLE - Fitting a theoretical function to experimental data

- In an experimental measurement, we expect (CLT) the experimental values to be normally distributed around the theoretical value with a certain variance. Mathematically, this means:

$$y - f\left(x\right) \approx \frac{1}{\sqrt{2\sigma^2}} \exp\left[-\frac{\left(y - f\left(x\right)\right)^2}{2\sigma^2}\right]$$

- where y are the experimental values and $f\left(x\right)$ the theoretical ones. The likelihood is then:

MLE - Fitting a theoretical function to experimental data

- In an experimental measurement, we expect (CLT) the experimental values to be normally distributed around the theoretical value with a certain variance. Mathematically, this means:

$$y - f\left(x\right) \approx \frac{1}{\sqrt{2\sigma^2}} \exp\left[-\frac{\left(y - f\left(x\right)\right)^2}{2\sigma^2}\right]$$

- where y are the experimental values and $f\left(x\right)$ the theoretical ones. The likelihood is then:

$$\mathcal{L} = -\frac{N}{2}\log\left[2\sigma^2\right] - \sum_i \left[\frac{\left(y_i - f\left(x_i\right)\right)^2}{2\sigma^2}\right]$$

- Where we see that to maximize the likelihood we must minimize the sum of squares

MLE - Fitting a theoretical function to experimental data

- In an experimental measurement, we expect (CLT) the experimental values to be normally distributed around the theoretical value with a certain variance. Mathematically, this means:

$$y - f\left(x\right) \approx \frac{1}{\sqrt{2\sigma^2}} \exp\left[-\frac{\left(y - f\left(x\right)\right)^2}{2\sigma^2}\right]$$

- where y are the experimental values and $f\left(x\right)$ the theoretical ones. The likelihood is then:

$$\mathcal{L} = -\frac{N}{2}\log\left[2\sigma^2\right] - \sum_i \left[\frac{\left(y_i - f\left(x_i\right)\right)^2}{2\sigma^2}\right]$$

- Where we see that to maximize the likelihood we must minimize the sum of squares

Least Squares Fitting

MLE - Fitting a power-law to experimental data

SIAM Rev. 51, 661 (2009)

MLE - Fitting a power-law to experimental data

SIAM Rev. 51, 661 (2009)

- We often find what look like power-law distributions in empirical data:

$$P(k) = \frac{\gamma - 1}{k_{min}} \left(\frac{k}{k_{min}} \right)^{-\gamma}$$

and we would like to find the right parameter values.

MLE - Fitting a power-law to experimental data

SIAM Rev. 51, 661 (2009)

- We often find what look like power-law distributions in empirical data:

$$P(k) = \frac{\gamma - 1}{k_{min}} \left(\frac{k}{k_{min}} \right)^{-\gamma}$$

and we would like to find the right parameter values.

- The likelihood of any set of points is:

$$\mathcal{L} = \sum_i \log \left[\frac{\gamma - 1}{k_{min}} \left(\frac{k_i}{k_{min}} \right)^{-\gamma} \right]$$

MLE - Fitting a power-law to experimental data

SIAM Rev. 51, 661 (2009)

- We often find what look like power-law distributions in empirical data:

$$P(k) = \frac{\gamma - 1}{k_{min}} \left(\frac{k}{k_{min}} \right)^{-\gamma}$$

and we would like to find the right parameter values.

- The likelihood of any set of points is:

$$\mathcal{L} = \sum_i \log \left[\frac{\gamma - 1}{k_{min}} \left(\frac{k_i}{k_{min}} \right)^{-\gamma} \right]$$

- And maximizing, we find:

$$\gamma = 1 + n \left[\sum_i \log \left(\frac{k_i}{k_{min}} \right) \right]^{-1}$$

MLE - Fitting a power-law to experimental data

SIAM Rev. 51, 661 (2009)

- We often find what look like power-law distributions in empirical data:

$$P(k) = \frac{\gamma - 1}{k_{min}} \left(\frac{k}{k_{min}} \right)^{-\gamma}$$

and we would like to find the right parameter values.

- The likelihood of any set of points is:

$$\mathcal{L} = \sum_i \log \left[\frac{\gamma - 1}{k_{min}} \left(\frac{k_i}{k_{min}} \right)^{-\gamma} \right]$$

- And maximizing, we find:

$$\gamma = 1 + n \left[\sum_i \log \left(\frac{k_i}{k_{min}} \right) \right]^{-1}$$

- with a standard error of:

$$SE = \frac{\gamma - 1}{\sqrt{n}}$$

MLE - Fitting a power-law to experimental data

SIAM Rev. 51, 661 (2009)

- We often find what look like power-law distributions in empirical data:

$$P(k) = \frac{\gamma - 1}{k_{min}} \left(\frac{k}{k_{min}} \right)^{-\gamma}$$

and we would like to find the right parameter values.

- The likelihood of any set of points is:

$$\mathcal{L} = \sum_i \log \left[\frac{\gamma - 1}{k_{min}} \left(\frac{k_i}{k_{min}} \right)^{-\gamma} \right]$$

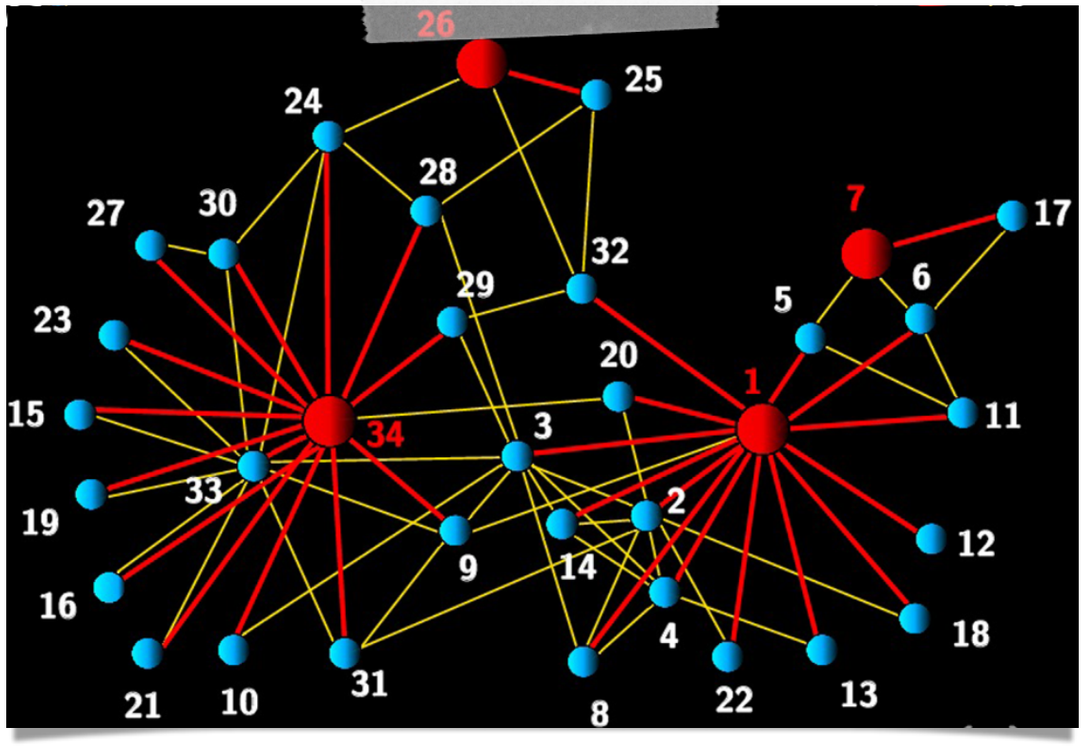
- And maximizing, we find:

$$\gamma = 1 + n \left[\sum_i \log \left(\frac{k_i}{k_{min}} \right) \right]^{-1}$$

- with a standard error of:

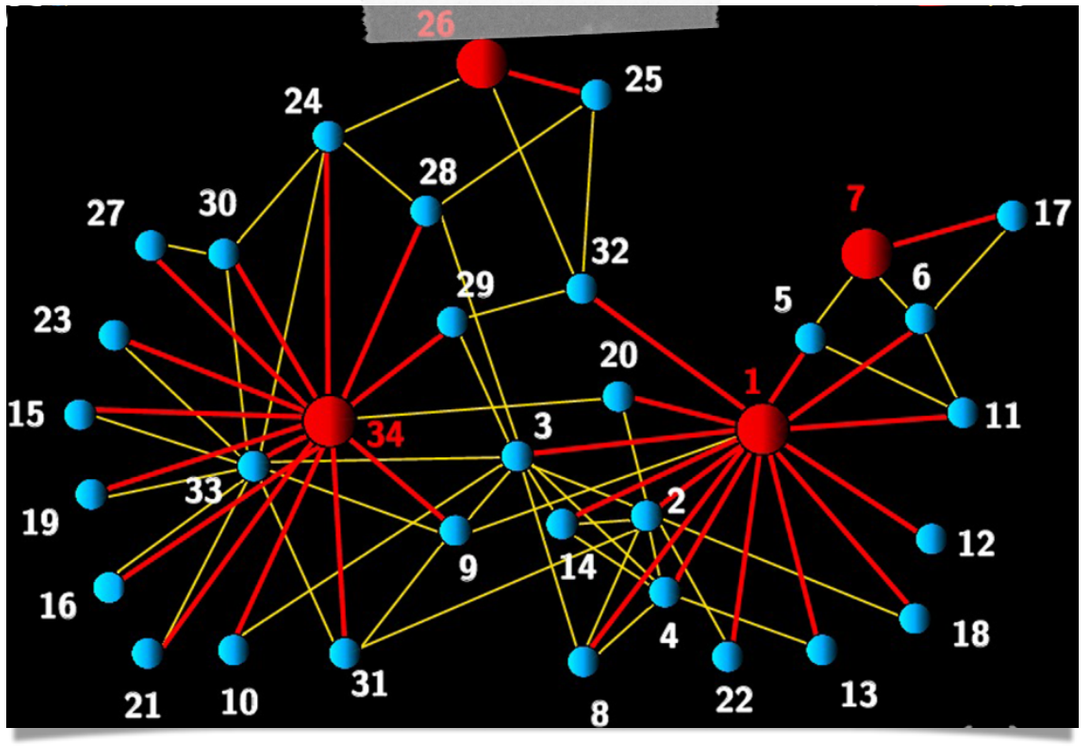
$$SE = \frac{\gamma - 1}{\sqrt{n}}$$

Communities



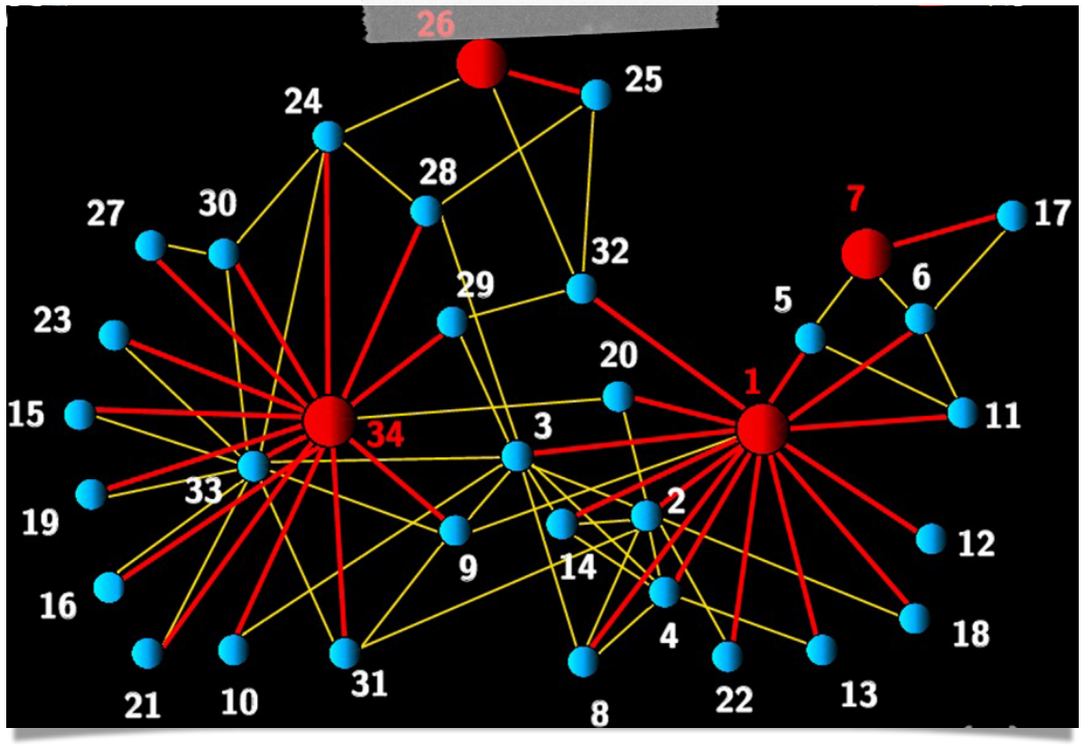
Communities

- Some network regions are denser than others



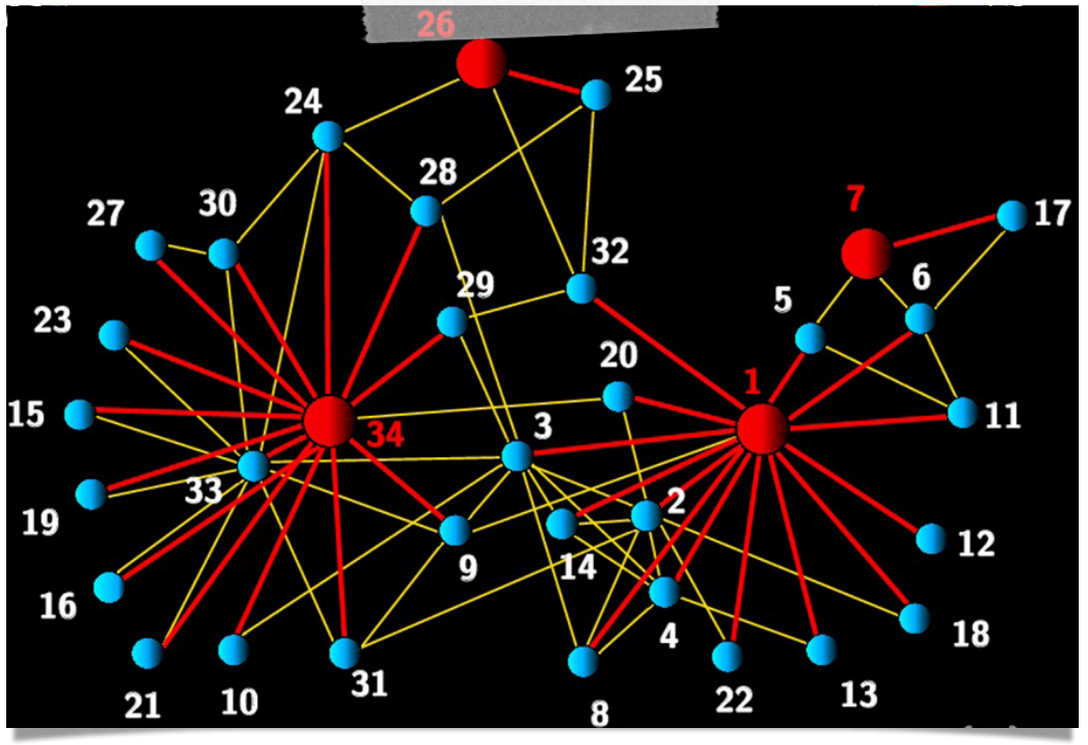
Communities

- Some network regions are denser than others
- Intuition:



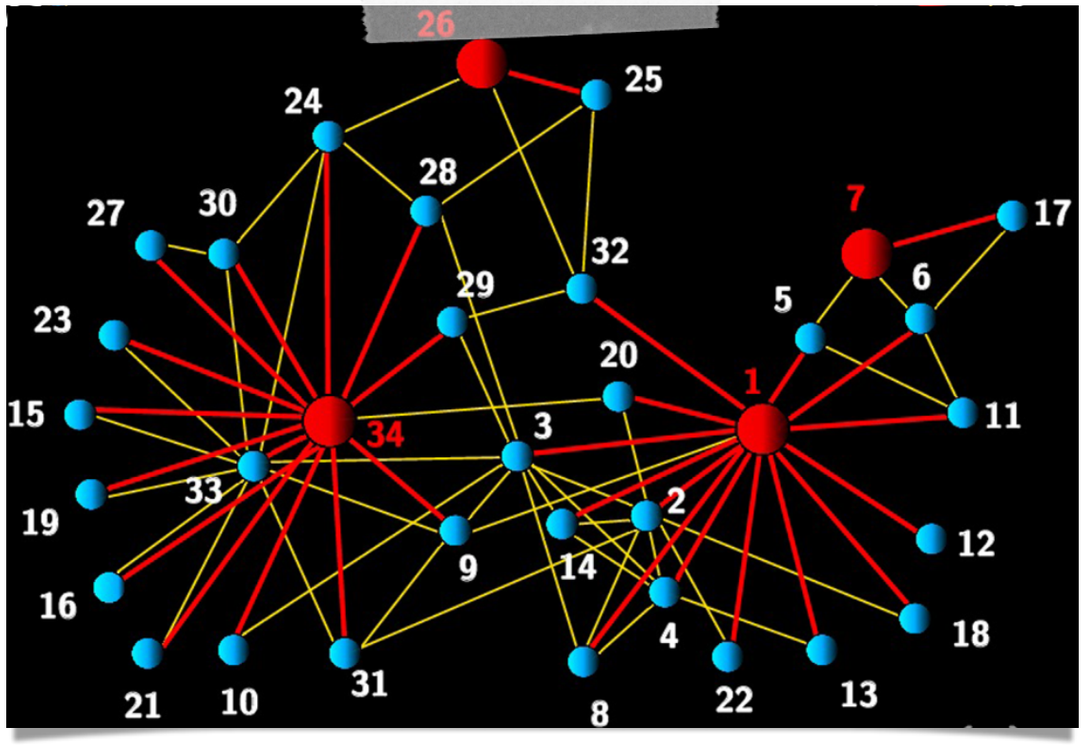
Communities

- Some network regions are denser than others
- Intuition:
  - "Communities" should have more links within the community than without



Communities

- Some network regions are denser than others
- Intuition:
  - "Communities" should have more links within the community than without
- Modularity:



$$Q = \sum_{i,j} \left[\frac{A_{ij}}{2m} - \frac{k_i k_j}{(2m)^2} \right]$$

Snowball Sampling

- Commonly used in Social Science and Computer Science

    1. Start with a single node (or small number of nodes)

    2. Get "friends" list

    3. For each friend get the "friend" list

    4. Repeat for a fixed number of layers or until enough users have been connected

- Generates a connected component from each seed

- Quickly generates a *lot* of data/API calls

Snowball Sampling

```python
import networkx as NX

def snowball(net, seed, max_depth = 3, maxnodes=1000):
    seen = set()
    queue = set()

    queue.add(seed)
    queue2 = set()

    for _ in range(max_depth+1):
        while queue:
            user_id = queue.pop()
            seen.add(user_id)

            NN = net.neighbors(user_id)

            for node in NN:
                if node not in seen:
                    queue2.add(node)

        queue.update(queue2)
        queue2 = set()

    return seen

net = NX.connected_watts_strogatz_graph(10000, 4, 0.01)
neve = snowball(net, 0)

print(neve)
```

snowball.py

Random Walk Sampling

Random Walk Sampling

- Similar to Snowball Sampling

Random Walk Sampling

- Similar to Snowball Sampling

- Follow just one link instead of all links

Random Walk Sampling

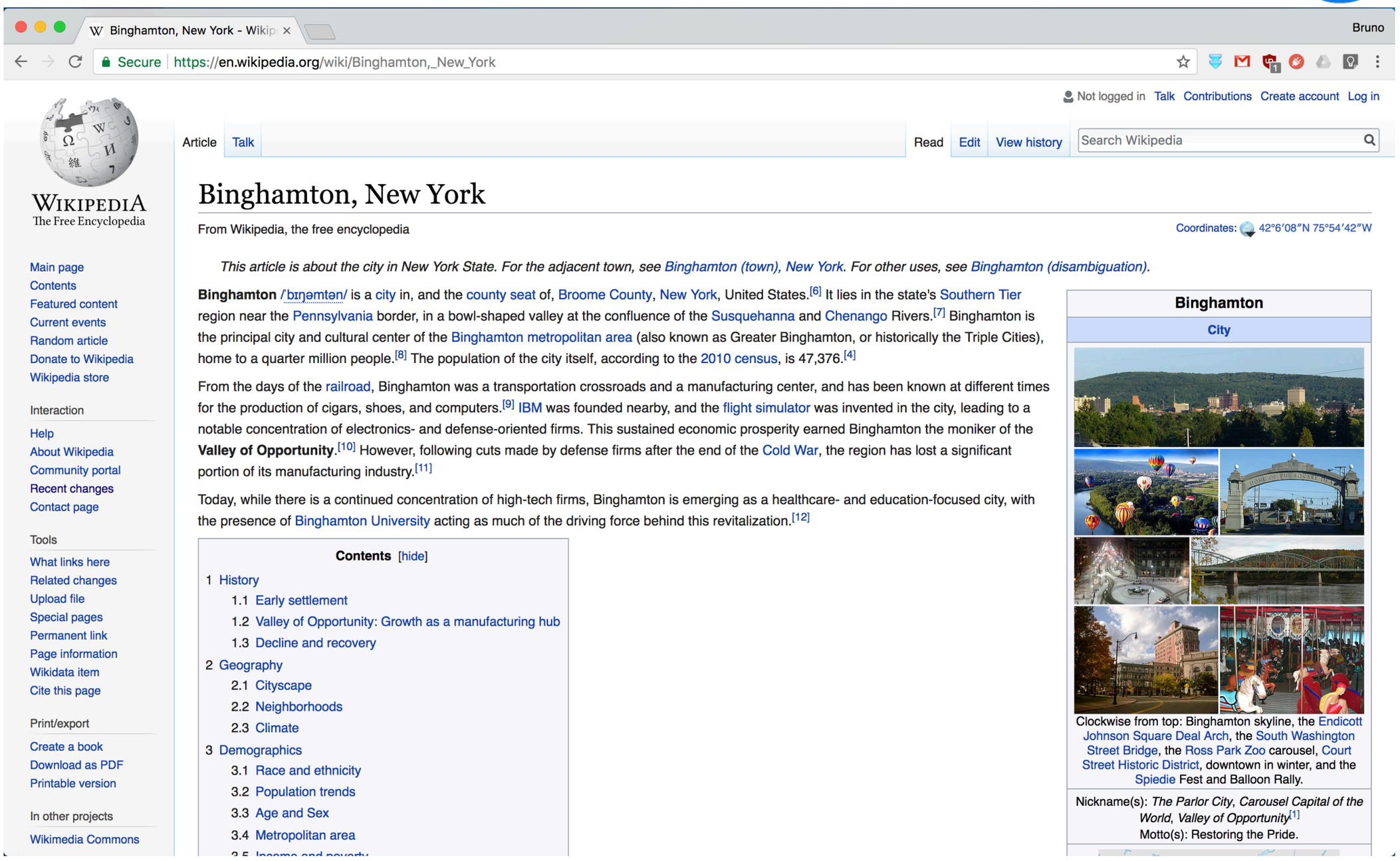
- Similar to Snowball Sampling
- Follow just one link instead of all links
- Asymptotic sampling probability of vertices in a Random Walk is proportional to vertex degree

Random Walk Sampling

- Similar to Snowball Sampling
- Follow just one link instead of all links
- Asymptotic sampling probability of vertices in a Random Walk is proportional to vertex degree
- Asymptotic regime reached quickly even for relatively small sample sizes

PageRank


Google


Binghamton, New York - Wikip
Bruno
Secure
https://en.wikipedia.org/wiki/Binghamton,_New_York
Not logged in
Talk
Contributions
Create account
Log in
Article
Talk
Read
Edit
View history
Search Wikipedia
WIKIPEDIA
The Free Encyclopedia
Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store
Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page
Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page
Print/export
Create a book
Download as PDF
Printable version
In other projects
Wikimedia Commons
Binghamton, New York
From Wikipedia, the free encyclopedia
Coordinates: 42°6′08″N 75°54′42″W
This article is about the city in New York State. For the adjacent town, see Binghamton (town), New York. For other uses, see Binghamton (disambiguation).
Binghamton /ˈbɪŋəmtən/ is a city in, and the county seat of, Broome County, New York, United States.[6] It lies in the state's Southern Tier region near the Pennsylvania border, in a bowl-shaped valley at the confluence of the Susquehanna and Chenango Rivers.[7] Binghamton is the principal city and cultural center of the Binghamton metropolitan area (also known as Greater Binghamton, or historically the Triple Cities), home to a quarter million people.[8] The population of the city itself, according to the 2010 census, is 47,376.[4]
From the days of the railroad, Binghamton was a transportation crossroads and a manufacturing center, and has been known at different times for the production of cigars, shoes, and computers.[9] IBM was founded nearby, and the flight simulator was invented in the city, leading to a notable concentration of electronics- and defense-oriented firms. This sustained economic prosperity earned Binghamton the moniker of the Valley of Opportunity.[10] However, following cuts made by defense firms after the end of the Cold War, the region has lost a significant portion of its manufacturing industry.[11]
Today, while there is a continued concentration of high-tech firms, Binghamton is emerging as a healthcare- and education-focused city, with the presence of Binghamton University acting as much of the driving force behind this revitalization.[12]
Contents [hide]
1 History
1.1 Early settlement
1.2 Valley of Opportunity: Growth as a manufacturing hub
1.3 Decline and recovery
2 Geography
2.1 Cityscape
2.2 Neighborhoods
2.3 Climate
3 Demographics
3.1 Race and ethnicity
3.2 Population trends
3.3 Age and Sex
3.4 Metropolitan area
Binghamton
City
Clockwise from top: Binghamton skyline, the Endicott Johnson Square Deal Arch, the South Washington Street Bridge, the Ross Park Zoo carousel, Court Street Historic District, downtown in winter, and the Spiedie Fest and Balloon Rally.
Nickname(s): The Parlor City, Carousel Capital of the World, Valley of Opportunity[1]
Motto(s): Restoring the Pride.

PageRank

"A page is important if it is pointed to by other important pages"

PageRank

"A page is important if it is pointed to by other important pages"

$$\pi_i = \sum_j a_{ji} \frac{\pi_j}{k_j}$$

PageRank

"A page is important if it is pointed to by other important pages"

$$\pi_i = \sum_j a_{ji} \frac{\pi_j}{k_j}$$

Stationary state of a discrete time random walk

PageRank

"A page is important if it is pointed to by other important pages"

$$\pi_i = \sum_j a_{ji} \frac{\pi_j}{k_j}$$

Stationary state of a discrete time random walk

$$\vec{\pi}\left(t+1\right) = \mathcal{T}\vec{\pi}\left(t\right)$$

PageRank

"A page is important if it is pointed to by other important pages"

$$\pi_i = \sum_j a_{ji} \frac{\pi_j}{k_j}$$

Stationary state of a discrete time random walk

$$\vec{\pi}\left(t+1\right) = \mathcal{T}\vec{\pi}\left(t\right)$$

$$\mathcal{T} = K_o^{-1} A^T$$

PageRank

"A page is important if it is pointed to by other important pages"

$$\pi_i = \sum_j a_{ji} \frac{\pi_j}{k_j}$$

Stationary state of a discrete time random walk

$$\vec{\pi}\left(t+1\right) = \mathcal{T}\vec{\pi}\left(t\right)$$

$$\mathcal{T} = K_o^{-1} A^T$$

Problems:

Solution:

PageRank

"A page is important if it is pointed to by other important pages"

$$\pi_i = \sum_j a_{ji} \frac{\pi_j}{k_j}$$

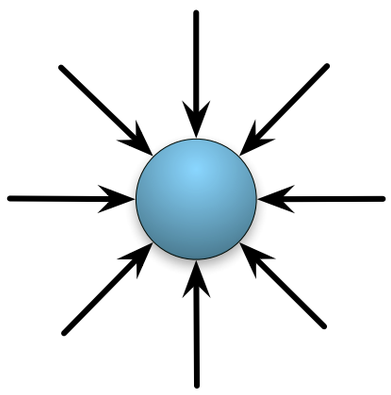
Stationary state of a discrete time random walk

$$\vec{\pi}\left(t+1\right) = \mathcal{T}\vec{\pi}\left(t\right)$$

$$\mathcal{T} = K_o^{-1} A^T$$

Problems:

"Dangling nodes"



Solution:

PageRank

"A page is important if it is pointed to by other important pages"

$$\pi_i = \sum_j a_{ji} \frac{\pi_j}{k_j}$$

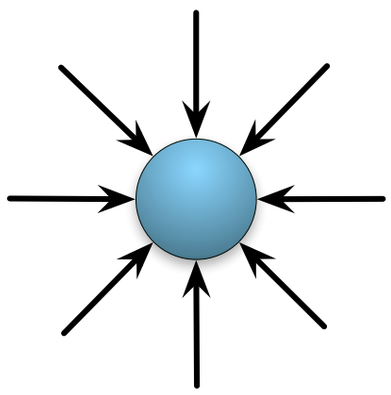
Stationary state of a discrete time random walk

$$\vec{\pi}\left(t+1\right) = \mathcal{T}\vec{\pi}\left(t\right)$$

$$\mathcal{T} = K_o^{-1} A^T$$

Problems:

"Dangling nodes"



Solution:

Connect dangling nodes to every other node

PageRank

"A page is important if it is pointed to by other important pages"

$$\pi_i = \sum_j a_{ji} \frac{\pi_j}{k_j}$$

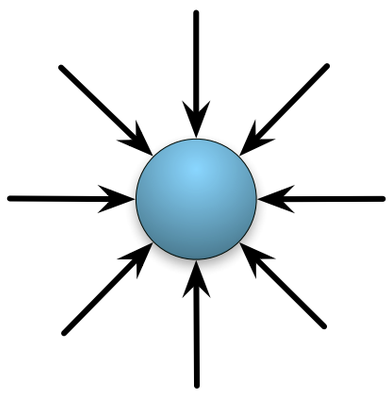
Stationary state of a discrete time random walk

$$\vec{\pi}\left(t+1\right) = \mathcal{T}\vec{\pi}\left(t\right)$$

$$\mathcal{T} = K_o^{-1} A^T$$

Problems:

"Dangling nodes"



Slow convergence

Solution:

Connect dangling nodes to every other node

PageRank

"A page is important if it is pointed to by other important pages"

$$\pi_i = \sum_j a_{ji} \frac{\pi_j}{k_j}$$

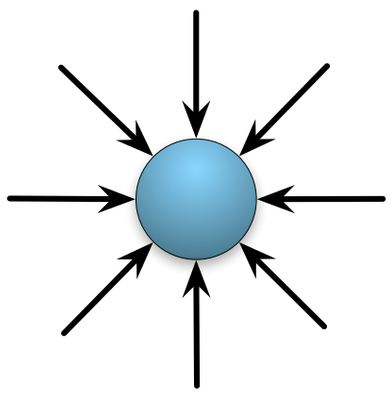
Stationary state of a discrete time random walk

$$\vec{\pi}\left(t+1\right) = \mathcal{T}\vec{\pi}\left(t\right)$$

$$\mathcal{T} = K_o^{-1} A^T$$

Problems:

"Dangling nodes"



Slow convergence

Solution:

Connect dangling nodes to every other node

Add damping factor

Markov Chain

Markov Chain

- Memoryless - "Future depends only on present state and not on past history"

$$\vec{\pi}\left(t+1\right) = \left[\left(1-\alpha\right)\mathcal{S} + \alpha\mathcal{E}\right]\vec{\pi}\left(t\right)$$

Markov Chain

- Memoryless - "Future depends only on present state and not on past history"

$$\vec{\pi}\left(t+1\right) = \left[\left(1-\alpha\right)\mathcal{S} + \alpha\mathcal{E}\right]\vec{\pi}\left(t\right)$$

Damping Factor

Markov Chain

- Memoryless - "Future depends only on present state and not on past history"

$$\vec{\pi}\left(t+1\right) = \left[\left(1-\alpha\right)\mathcal{S} + \alpha\mathcal{E}\right]\vec{\pi}\left(t\right)$$

Fully connected matrix

Damping Factor

Markov Chain

- Memoryless - "Future depends only on present state and not on past history"

$$\vec{\pi}\left(t+1\right) = \left[\left(1-\alpha\right)\mathcal{S} + \alpha\mathcal{E}\right]\vec{\pi}\left(t\right)$$

- Has a stationary state if matrix is:

Markov Chain

- Memoryless - "Future depends only on present state and not on past history"

$$\vec{\pi}\left(t+1\right) = \left[\left(1-\alpha\right)\mathcal{S} + \alpha\mathcal{E}\right]\vec{\pi}\left(t\right)$$

- Has a stationary state if matrix is:

  - Stochastic - Column vectors normalized to 1.

Fully connected matrix

Damping Factor

Markov Chain

- Memoryless - "Future depends only on present state and not on past history"

$$\vec{\pi}\left(t+1\right) = \left[\left(1-\alpha\right)\mathcal{S} + \alpha\mathcal{E}\right]\vec{\pi}\left(t\right)$$

- Has a stationary state if matrix is:

  - Stochastic - Column vectors normalized to 1.

  - Irreducible - All nodes are accessible (graph is connected)

Fully connected matrix

Damping Factor

Markov Chain

- Memoryless - "Future depends only on present state and not on past history"

$$\vec{\pi}\left(t+1\right) = \left[\left(1-\alpha\right)\mathcal{S} + \alpha\mathcal{E}\right]\vec{\pi}\left(t\right)$$

- Has a stationary state if matrix is:

  - Stochastic - Column vectors normalized to 1.

  - Irreducible - All nodes are accessible (graph is connected)

  - Aperiodic - Return to node i does not occur periodically

Fully connected matrix

Damping Factor

Markov Chain

- Memoryless - "Future depends only on present state and not on past history"

$$\vec{\pi}\left(t+1\right) = \left[\left(1-\alpha\right)\mathcal{S} + \alpha\mathcal{E}\right]\vec{\pi}\left(t\right)$$

- Has a stationary state if matrix is:

$$\vec{\pi} = \mathcal{G}\vec{\pi}$$

Fully connected matrix

Damping Factor

  - Stochastic - Column vectors normalized to 1.
  - Irreducible - All nodes are accessible (graph is connected)
  - Aperiodic - Return to node i does not occur periodically

Power Method

$$\vec{\pi} = \mathcal{G}\vec{\pi}$$

Power Method

$$\vec{\pi} = \mathcal{G}\vec{\pi}$$

Eigenvector corresponding to Eigenvalue 1.

Power Method

$$\vec{\pi} = \mathcal{G}\vec{\pi}$$

Eigenvector corresponding to Eigenvalue 1.

Iterate...

Power Method

$$\vec{\pi} = \mathcal{G}\vec{\pi}$$

Eigenvector corresponding to Eigenvalue 1.

Iterate...

$$\vec{\pi}^{(n+1)} = \mathcal{G}\vec{\pi}^{(n)}$$

Power Method

$$\vec{\pi} = \mathcal{G}\vec{\pi}$$

Eigenvector corresponding to Eigenvalue 1.

Iterate...

$$\vec{\pi}^{(n+1)} = \mathcal{G}\vec{\pi}^{(n)}$$

Stop when:

Power Method

$$\vec{\pi} = \mathcal{G}\vec{\pi}$$

Eigenvector corresponding to Eigenvalue 1.

Iterate...

$$\vec{\pi}^{(n+1)} = \mathcal{G}\vec{\pi}^{(n)}$$

Stop when:

$$\vec{\pi}^{(n+1)} \approx \vec{\pi}^{(n)}$$

Power Method

$$\vec{\pi} = \mathcal{G}\vec{\pi}$$

Eigenvector corresponding to Eigenvalue 1.

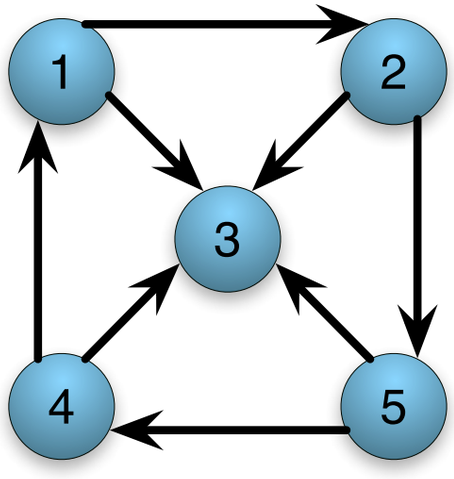
Iterate...

$$\vec{\pi}^{(n+1)} = \mathcal{G}\vec{\pi}^{(n)}$$

Stop when:

$$\vec{\pi}^{(n+1)} \approx \vec{\pi}^{(n)}$$

Example:

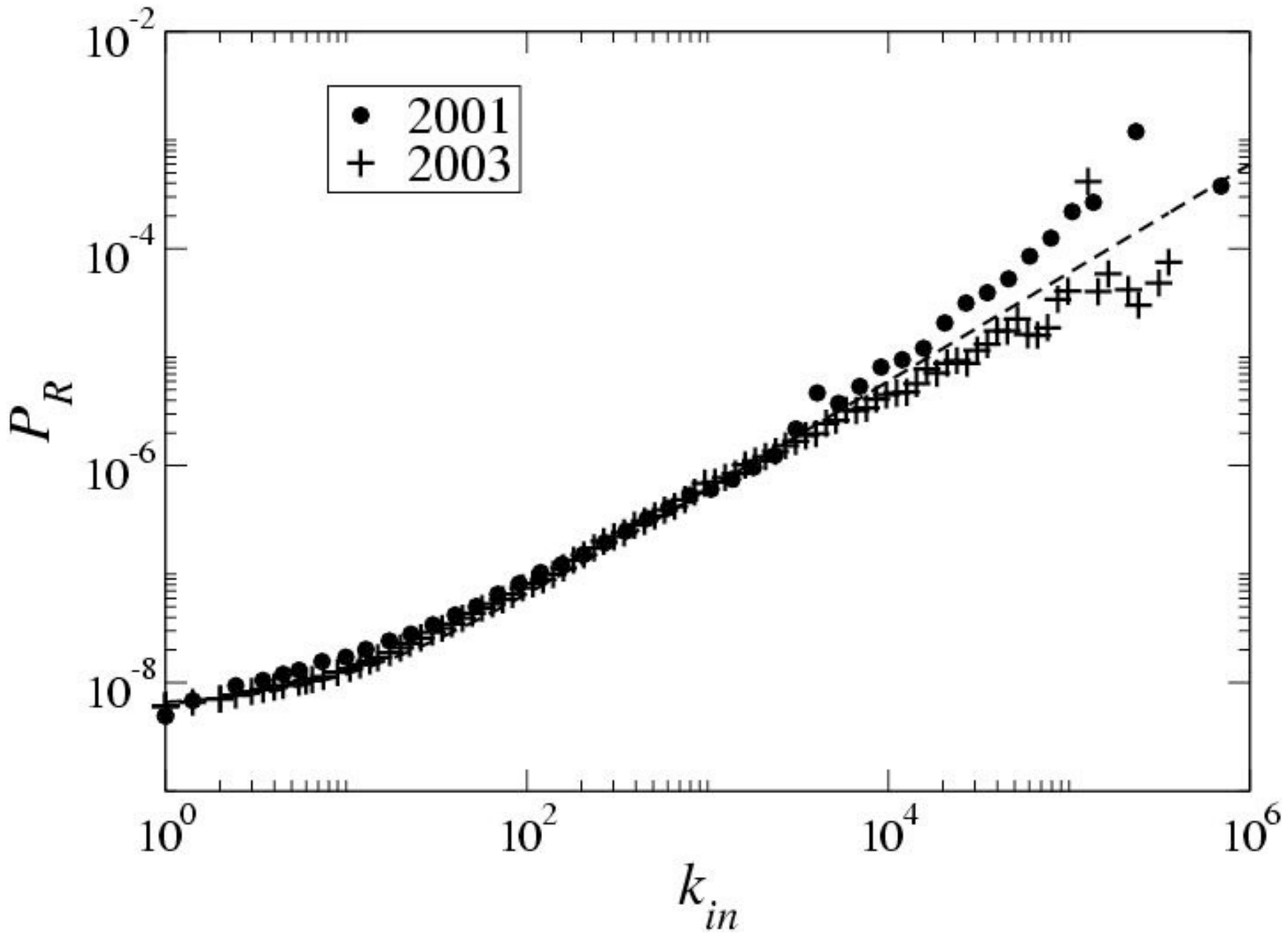


PageRank

```python
def Google_Matrix(A, m):
    N = A.shape[0]
    v = np.ones(N)

    KT = np.dot(A, v)
    A = A.T

    for i in range(N):
        if KT[i] != 0:
            A.T[i] = A.T[i]/KT[i]
        else:
            A.T[i] = np.ones(N)/float(N)

    E = np.ones((N,N))/N
    G = (1-m)*A+m*E

    return G

def Power_Method(G, iter):
    N = G.shape[0]
    x0 = np.ones(N)/N

    print(0, x0, np.sum(x0))

    for i in range(iter):
        x1 = np.copy(x0)
        x0 = np.dot(G,x0)
        print(i+1, x0, np.sum(x0))
        if np.sum(np.abs(x1-x0)) < 1e-10:
            break
```

pagerank.py

Web Graph



Network Backbone

PNAS 106, 6487 (2009)

- Algorithm to identify the most relevant weights in a network.

- Applies a "disparity filter" to identify which edges should be preserved

- For each edge, calculate:

$$\alpha_{ij} = 1 - (k-1)\int_0^{p_{ij}} (1-x)^{k-2}\,\mathrm{d}x$$

- Keep edge if and only if:

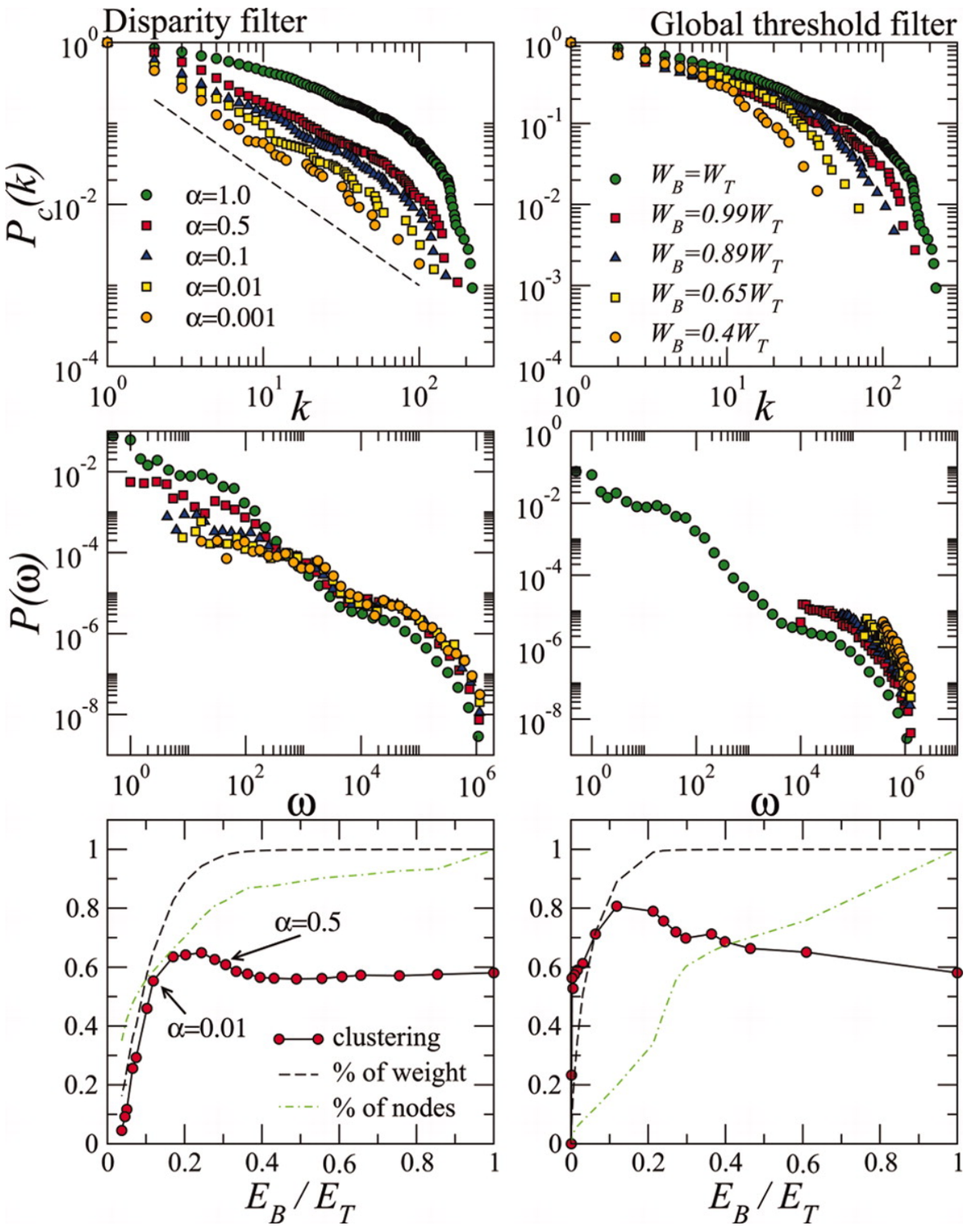
$$\alpha_{ij} < \alpha$$

- For some specified significance value, α

- Similar to p value in statistics.

Network Backbone

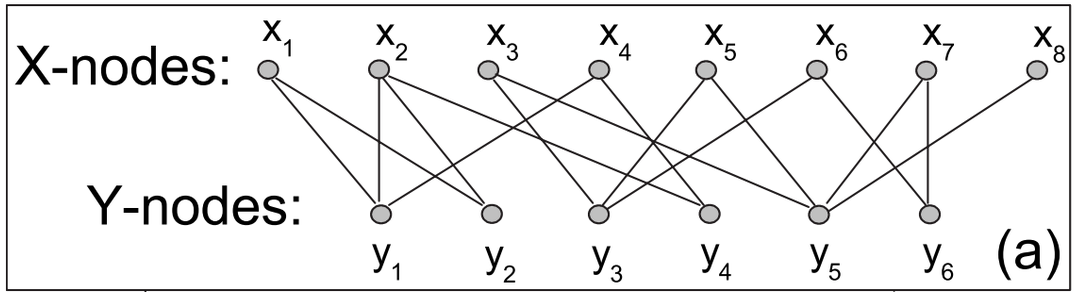
PNAS 106, 6487 (2009)



Bipartite Networks

Bipartite Networks

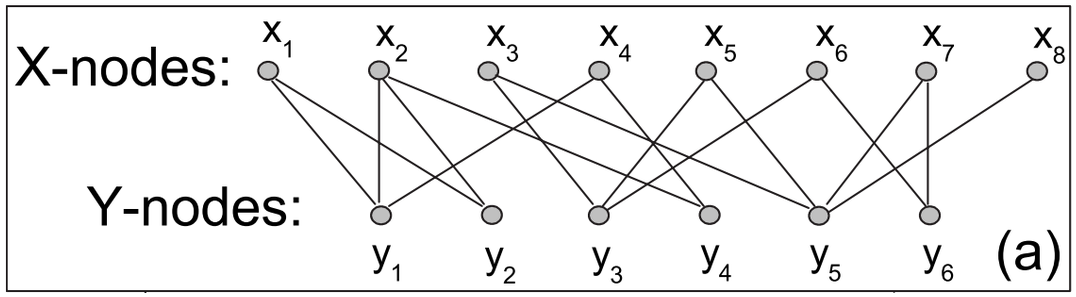
- Two kinds of nodes:



Bipartite Networks

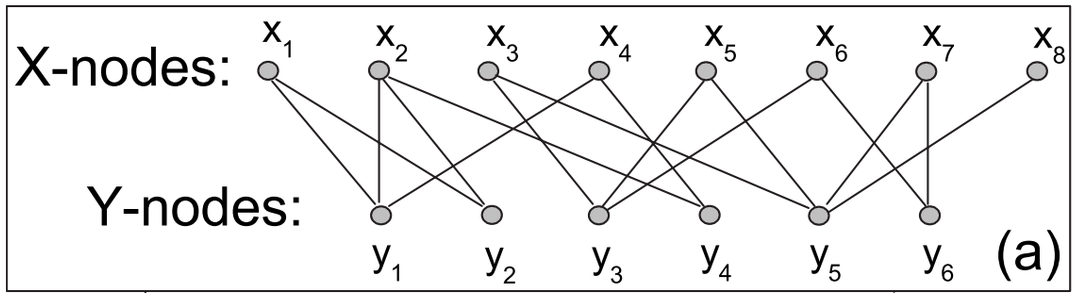
- Two kinds of nodes:

- Edges only allowed between nodes of different kinds



Bipartite Networks

- Two kinds of nodes:

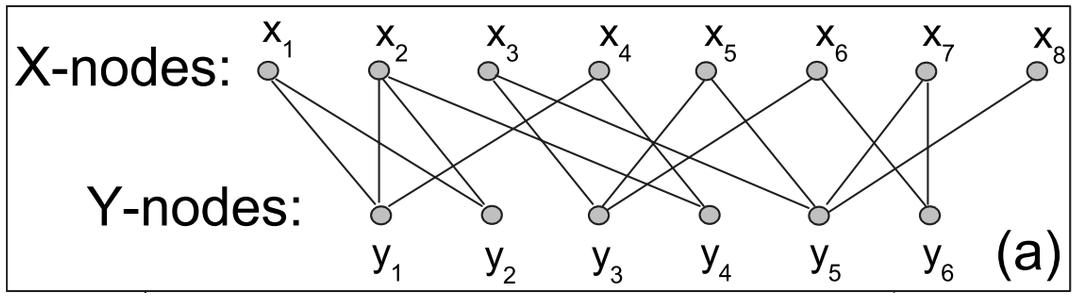


- Edges only allowed between nodes of different kinds

- Typically represent affiliation or collaboration:

Bipartite Networks

- Two kinds of nodes:

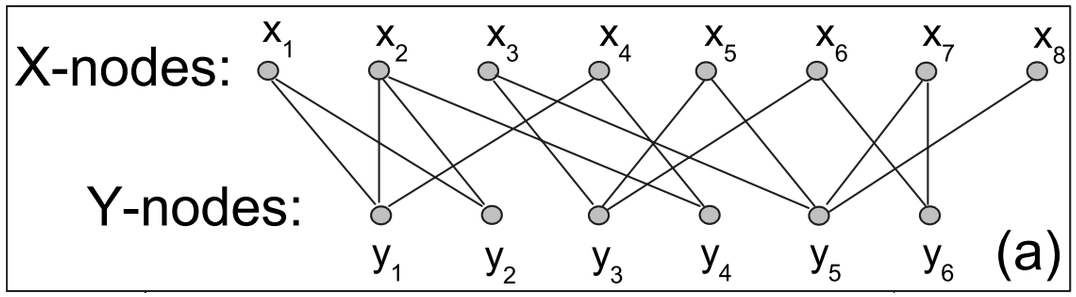


- Edges only allowed between nodes of different kinds

- Typically represent affiliation or collaboration:
  - actors in movies

Bipartite Networks

- Two kinds of nodes:



- Edges only allowed between nodes of different kinds

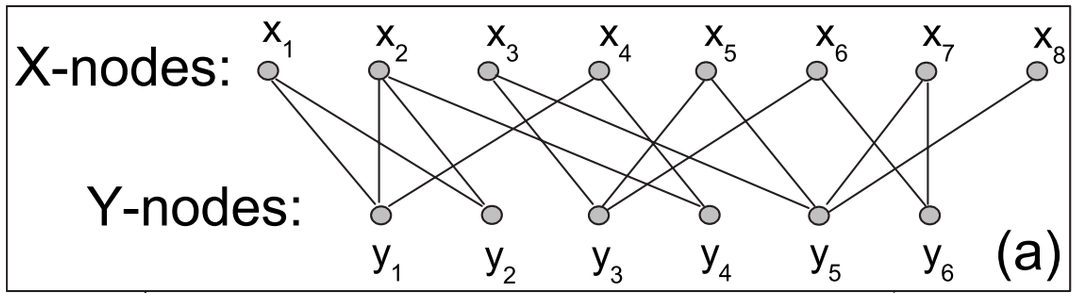
- Typically represent affiliation or collaboration:

  - actors in movies

  - authors in papers

Bipartite Networks

- Two kinds of nodes:

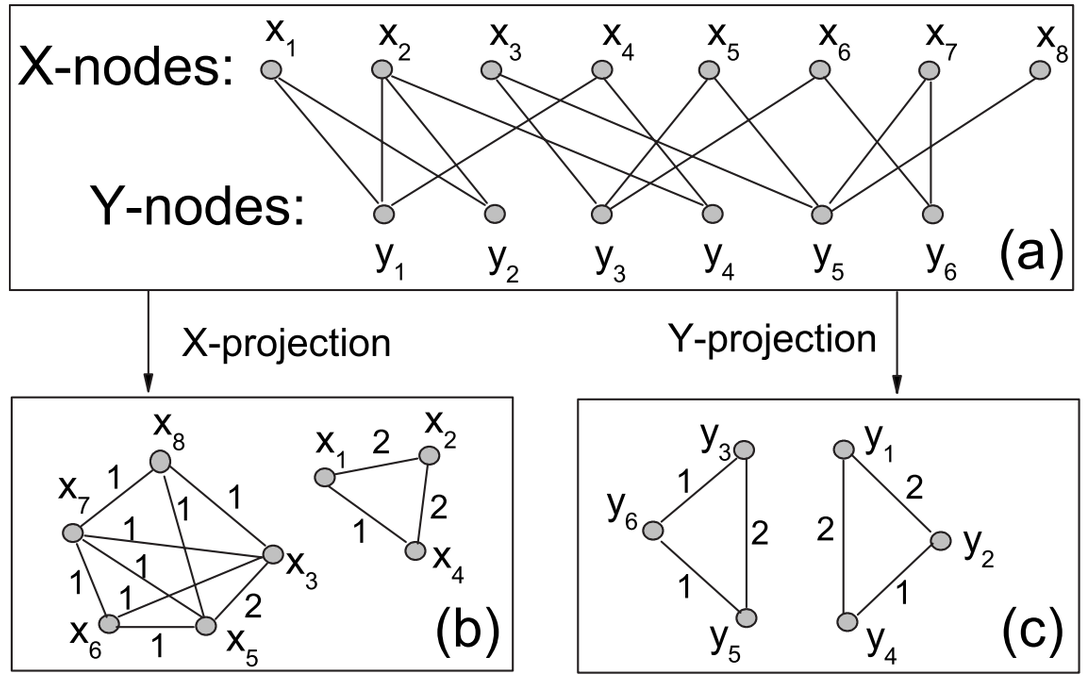


- Edges only allowed between nodes of different kinds

- Typically represent affiliation or collaboration:
  - actors in movies
  - authors in papers
  - etc

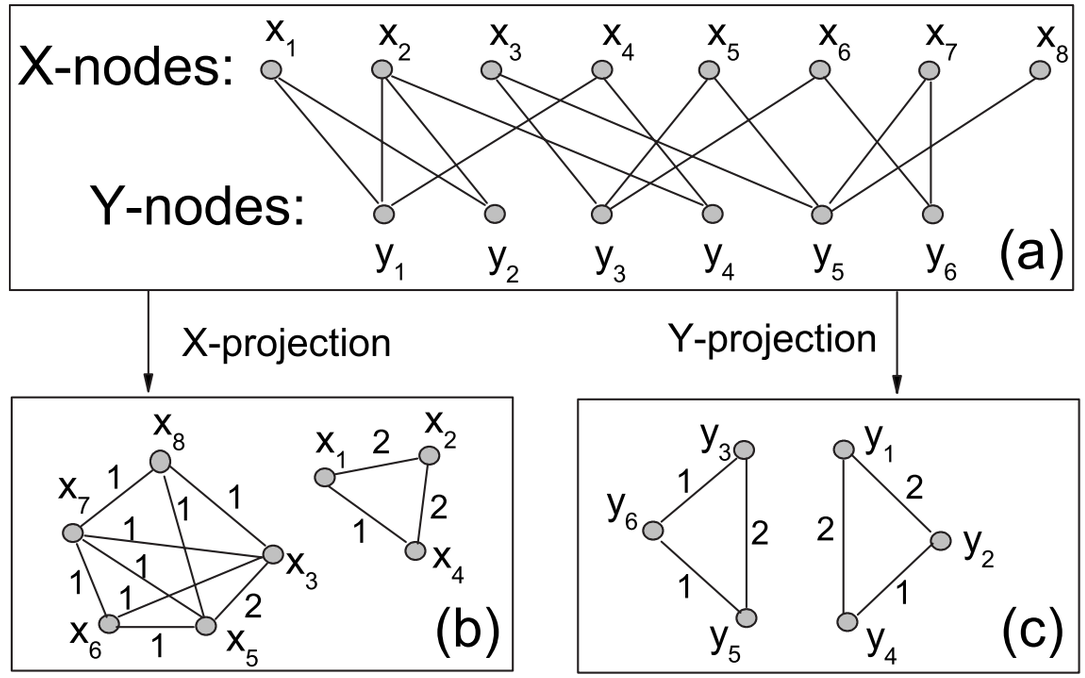
Bipartite Networks

- Two kinds of nodes:
- Edges only allowed between nodes of different kinds
- Typically represent affiliation or collaboration:
  - actors in movies
  - authors in papers
  - etc
- Usually the first step is to project them to a single node type



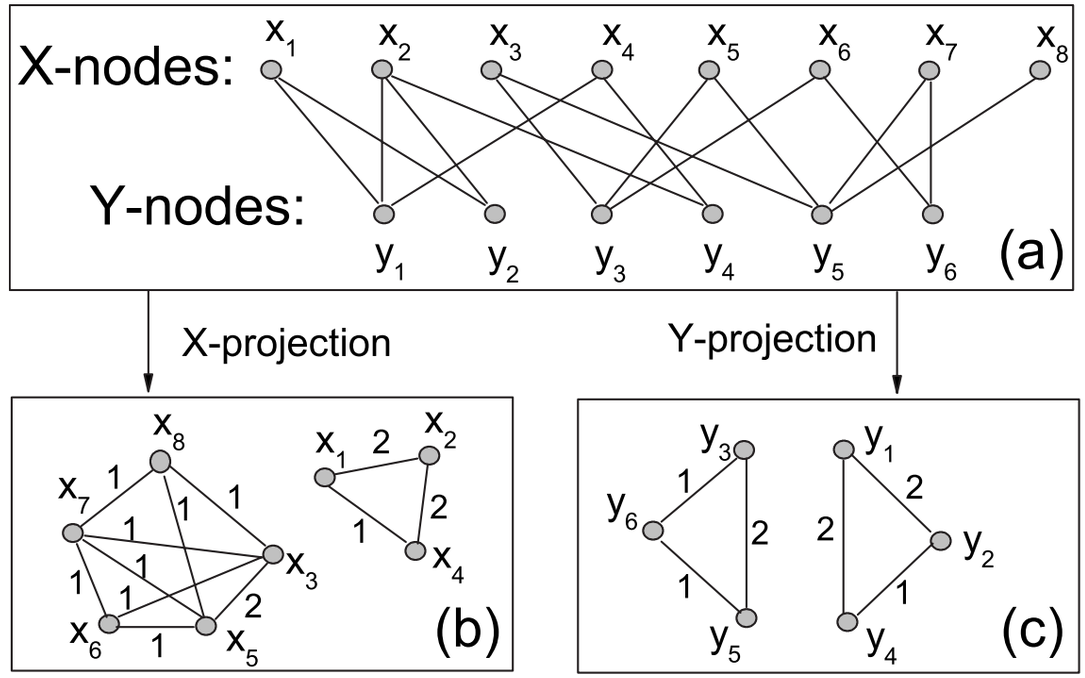
Bipartite Networks - Adjacency Matrix

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



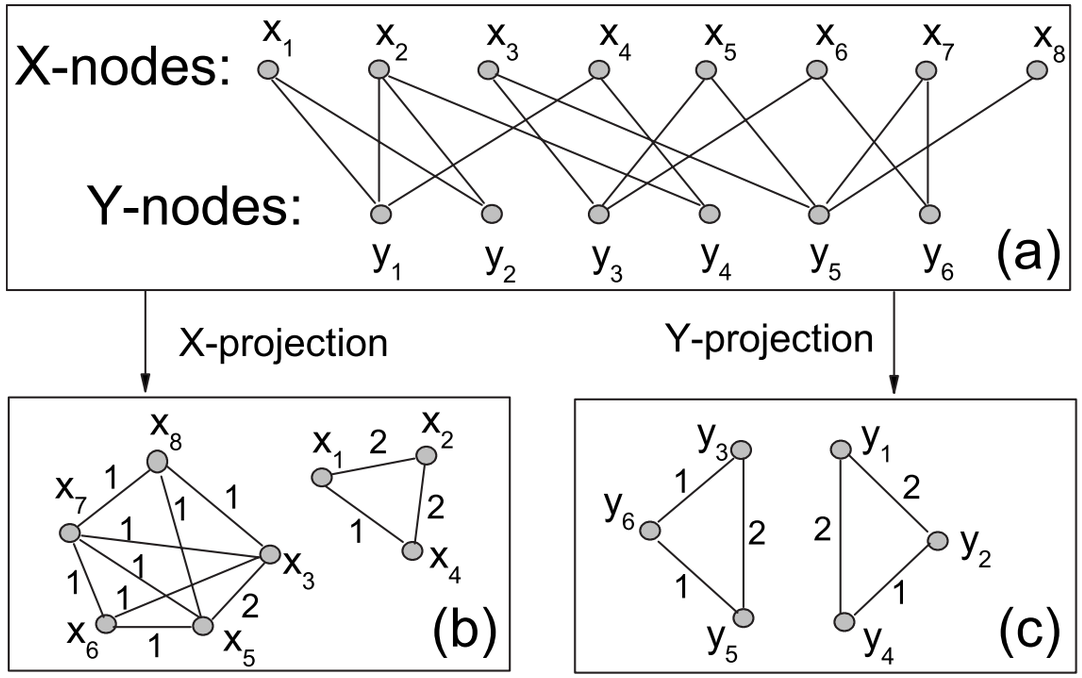
Bipartite Networks - Adjacency Matrix

$$A = \begin{pmatrix} & & & y & & \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$



Bipartite Networks - Adjacency Matrix

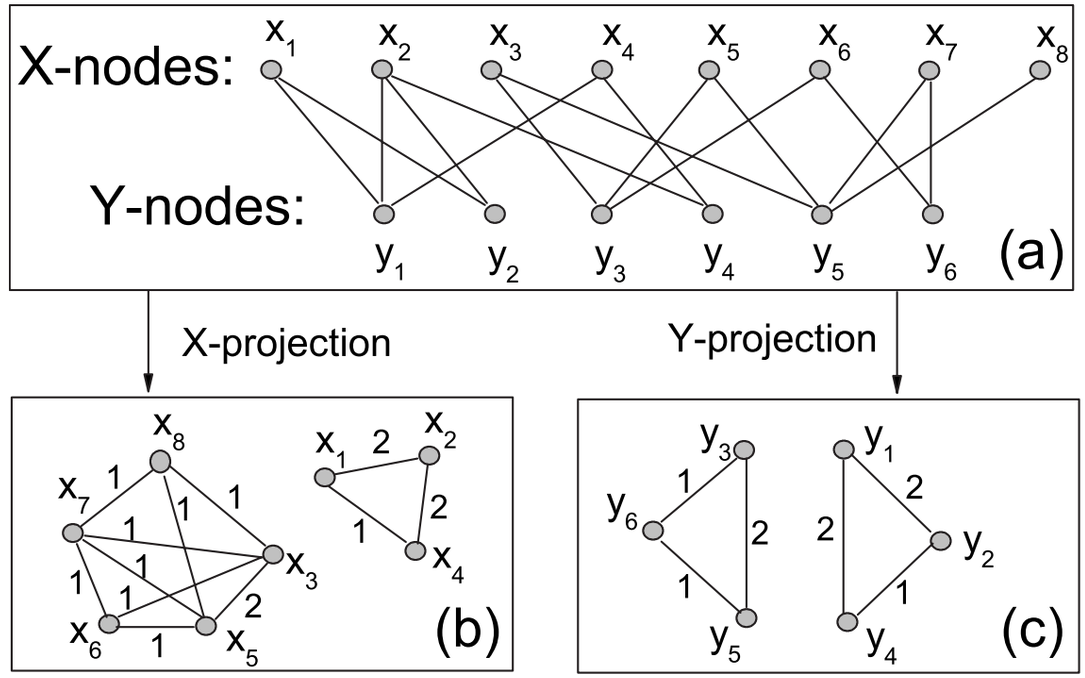
$$A = \begin{matrix} \\ x \end{matrix} \overset{\overset{y}{\begin{matrix} \end{matrix}}}{\begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}}$$



Bipartite Networks - Adjacency Matrix

$$A = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \begin{matrix} \\ \\ \\ \\ \\ \\ \\ \\ \end{matrix} x$$

$$\begin{matrix} \\ y \end{matrix}$$

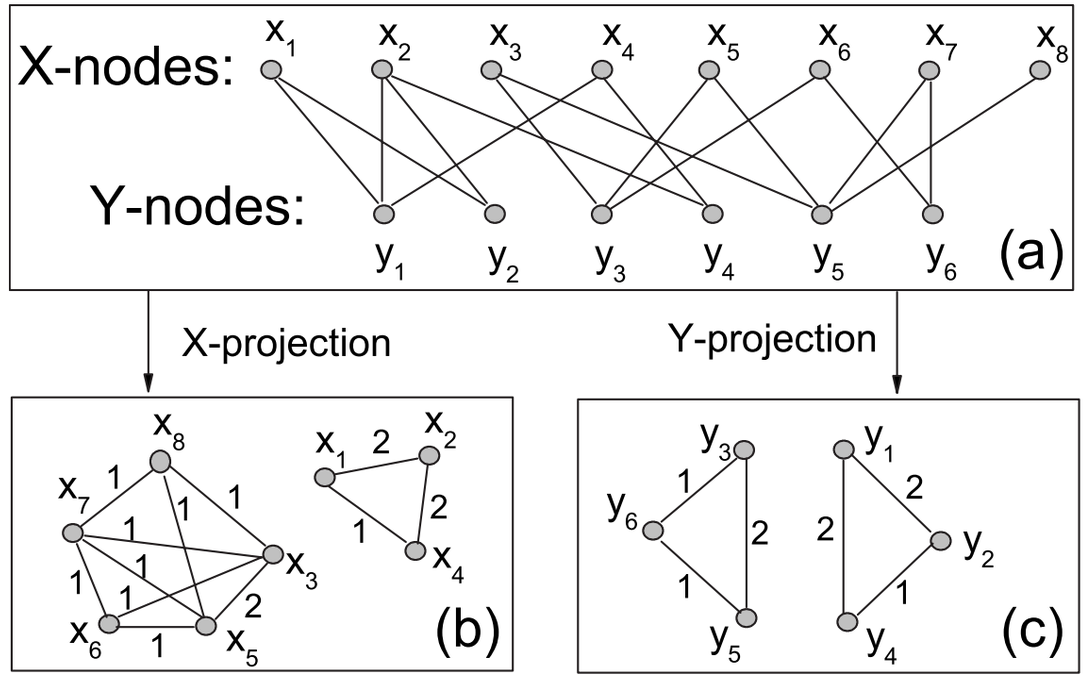


$$X = AA^T$$

Bipartite Networks - Adjacency Matrix

$$A = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \begin{matrix} \\ \\ \\ \\ \\ \\ \\ \\ \end{matrix} x$$

$$\begin{matrix} \\ y \end{matrix}$$



$$X = AA^T$$

$$Y = A^T A$$

Bipartite Networks - Similarities

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

Bipartite Networks - Similarities

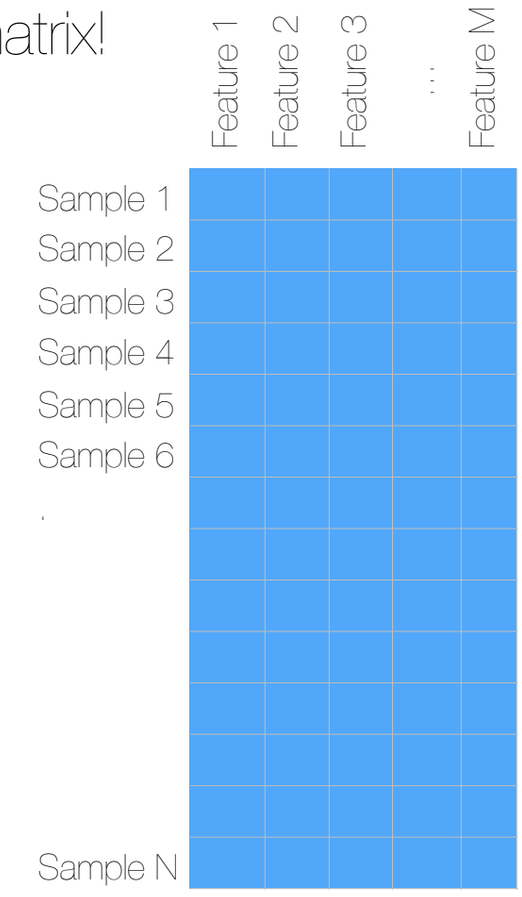
We can treat the adjacency matrix as a feature matrix!

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

Bipartite Networks - Similarities

We can treat the adjacency matrix as a feature matrix!

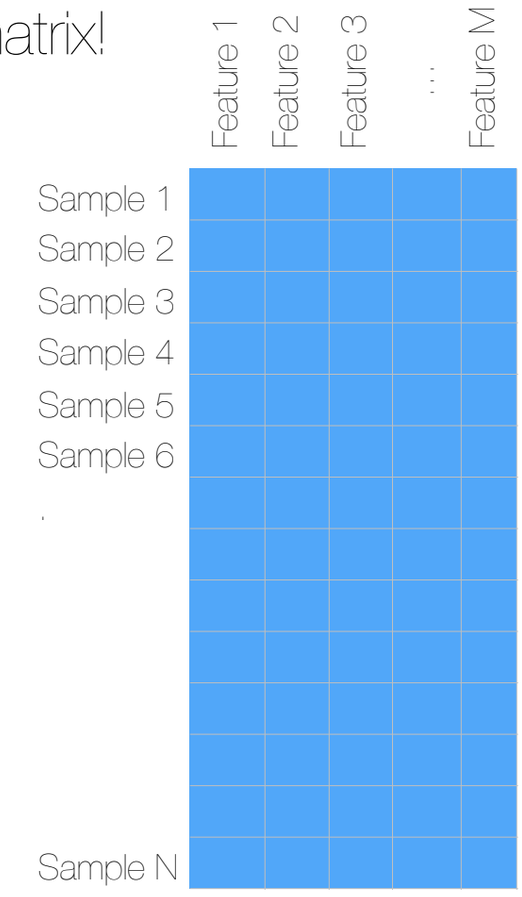
$$A = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$



Bipartite Networks - Similarities

We can treat the adjacency matrix as a feature matrix!

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

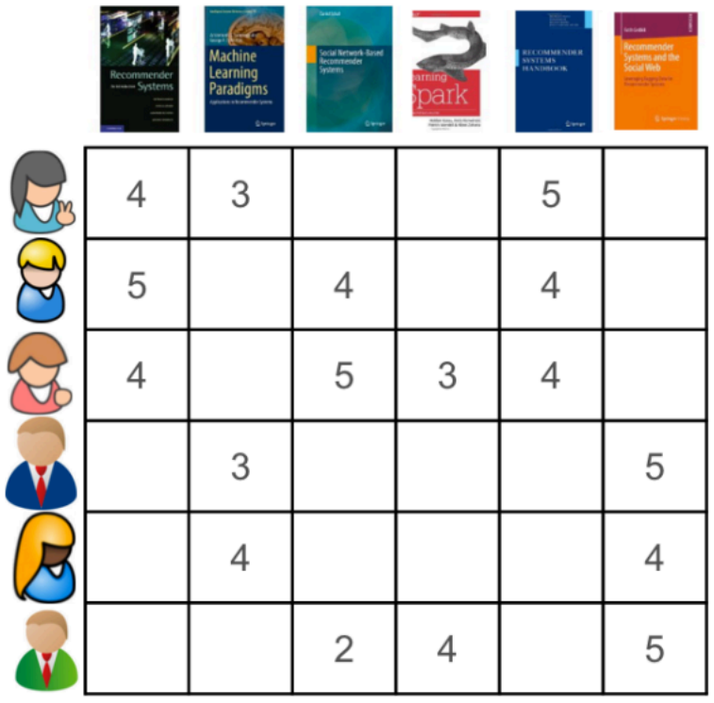


And generate a new matrix based on the similarities between items

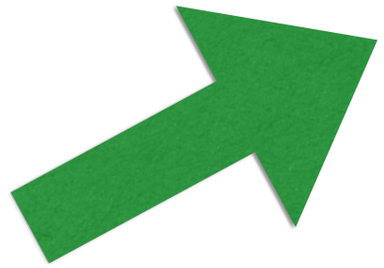
Bipartite Networks - Similarities

	4	3			5	
	5		4		4	
	4		5	3	4	
		3				5
		4				4
			2	4		5

Bipartite Networks - Similarities

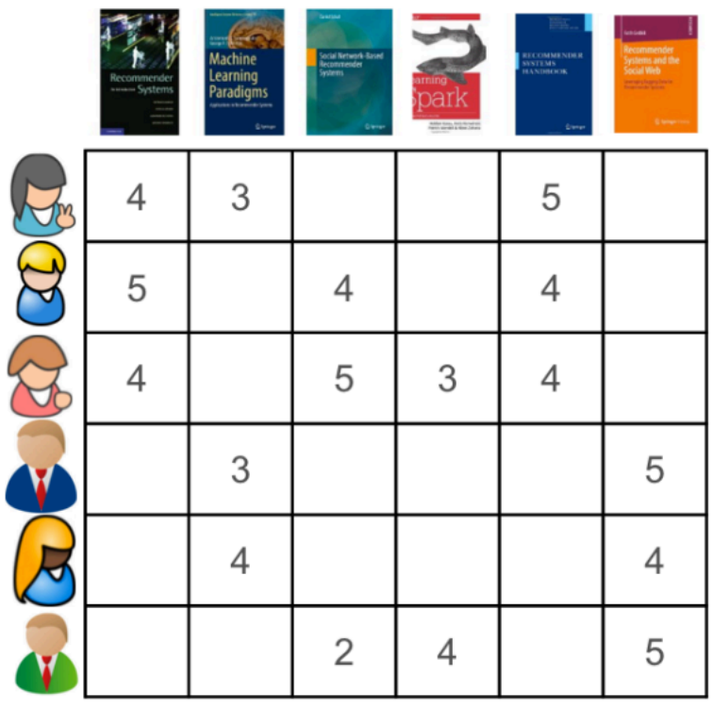


4	3			5	
5		4		4	
4		5	3	4	
	3				5
	4				4
		2	4		5



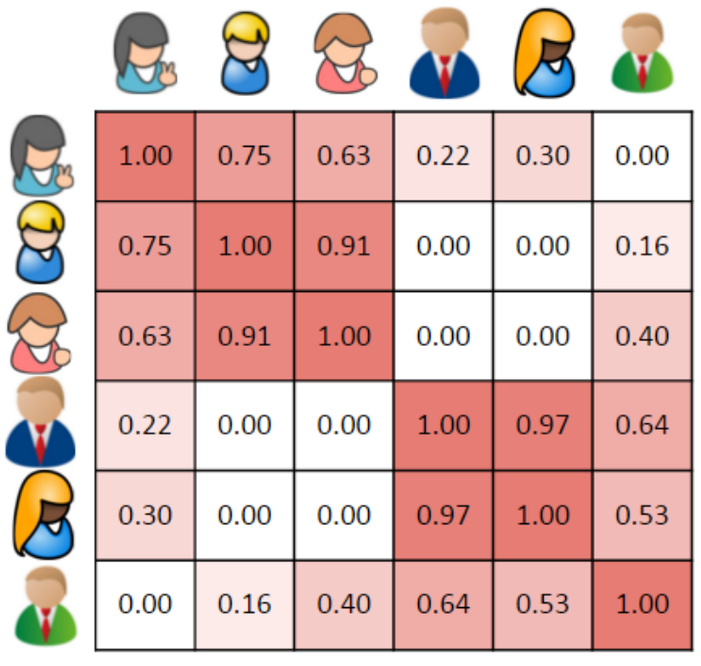
1.00	0.00	0.58	0.00	0.67	0.58
0.00	1.00	0.00	0.41	0.00	0.00
0.58	0.00	1.00	0.00	0.58	0.75
0.00	0.41	0.00	1.00	0.00	0.00
0.67	0.00	0.58	0.00	1.00	0.58
0.58	0.00	0.75	0.00	0.58	1.00

Bipartite Networks - Similarities



4	3			5	
5		4		4	
4		5	3	4	
	3				5
	4				4
		2	4		5

1.00	0.00	0.58	0.00	0.67	0.58
0.00	1.00	0.00	0.41	0.00	0.00
0.58	0.00	1.00	0.00	0.58	0.75
0.00	0.41	0.00	1.00	0.00	0.00
0.67	0.00	0.58	0.00	1.00	0.58
0.58	0.00	0.75	0.00	0.58	1.00



1.00	0.75	0.63	0.22	0.30	0.00
0.75	1.00	0.91	0.00	0.00	0.16
0.63	0.91	1.00	0.00	0.00	0.40
0.22	0.00	0.00	1.00	0.97	0.64
0.30	0.00	0.00	0.97	1.00	0.53
0.00	0.16	0.40	0.64	0.53	1.00

Bipartite Networks - Similarities

Sparse

4	3			5	
5		4		4	
4		5	3	4	
	3				5
	4				4
		2	4		5

Dense

1.00	0.00	0.58	0.00	0.67	0.58
0.00	1.00	0.00	0.41	0.00	0.00
0.58	0.00	1.00	0.00	0.58	0.75
0.00	0.41	0.00	1.00	0.00	0.00
0.67	0.00	0.58	0.00	1.00	0.58
0.58	0.00	0.75	0.00	0.58	1.00

1.00	0.75	0.63	0.22	0.30	0.00
0.75	1.00	0.91	0.00	0.00	0.16
0.63	0.91	1.00	0.00	0.00	0.40
0.22	0.00	0.00	1.00	0.97	0.64
0.30	0.00	0.00	0.97	1.00	0.53
0.00	0.16	0.40	0.64	0.53	1.00

RECOMMENDER SYSTEM ALGORITHMS

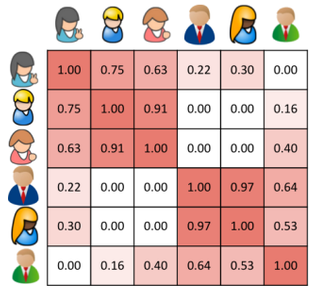
INPUT

4	3			5	
5		4		4	
4		5	3	4	
	3				5
	4				4
		2	4		5

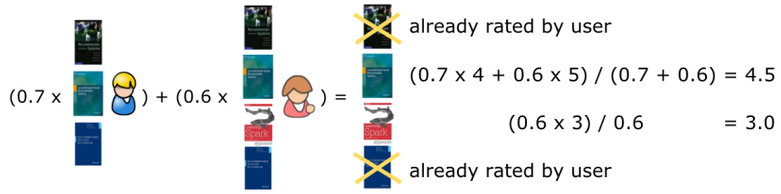
Introduction to Recommender Systems
Machine learning Paradigms
Social Network-based Recommender Systems
Learning Spark
Recommender Systems Handbook
Recommender Systems and the Social Web

FOUR RECOMMENDER ALGORITHMS ARE FED THE SAME INPUT AND PRODUCE DIFFERENT OUTPUTS.
THEIR DEFINITIONS OF USER AND/OR ITEM SIMILARITY ACCOUNT FOR MOST OF THE DIFFERENCES.

USER-BASED COLLABORATIVE FILTERING



1.00	0.75	0.63	0.22	0.30	0.00
0.75	1.00	0.91	0.00	0.00	0.16
0.63	0.91	1.00	0.00	0.00	0.40
0.22	0.00	0.00	1.00	0.97	0.64
0.30	0.00	0.00	0.97	1.00	0.53
0.00	0.16	0.40	0.64	0.53	1.00



ITEM-BASED COLLABORATIVE FILTERING

1.00	0.27	0.79	0.32	0.98	0.00
0.27	1.00	0.00	0.00	0.34	0.65
0.79	0.00	1.00	0.69	0.71	0.18
0.32	0.00	0.69	1.00	0.32	0.49
0.98	0.34	0.71	0.32	1.00	0.00
0.00	0.65	0.18	0.49	0.00	1.00

CONTENT-BASED FILTERING

1.00	0.00	0.58	0.00	0.67	0.58
0.00	1.00	0.00	0.41	0.00	0.00
0.58	0.00	1.00	0.00	0.58	0.75
0.00	0.41	0.00	1.00	0.00	0.00
0.67	0.00	0.58	0.00	1.00	0.58
0.58	0.00	0.75	0.00	0.58	1.00

HYBRID

OUTPUTS