Online Social Networks and Media

Absorbing Random Walks

Link Prediction

Why does the Power Method work?

- If a matrix R is real and symmetric, it has real eigenvalues and eigenvectors: $(\lambda_1, w_1), (\lambda_2, w_2), \ldots, (\lambda_r, w_r)$
  - r is the rank of the matrix
  - $|\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_r|$

- The vector space of R is the set of vectors that can be written as a linear combination of its rows (or columns)
- The eigenvectors $w_1, w_2, \ldots, w_r$ of R define a basis of the vector space
  - For any vector x, $Rx = \alpha_1 w_1 + a_2 w_2 + \cdots + a_r w_r$

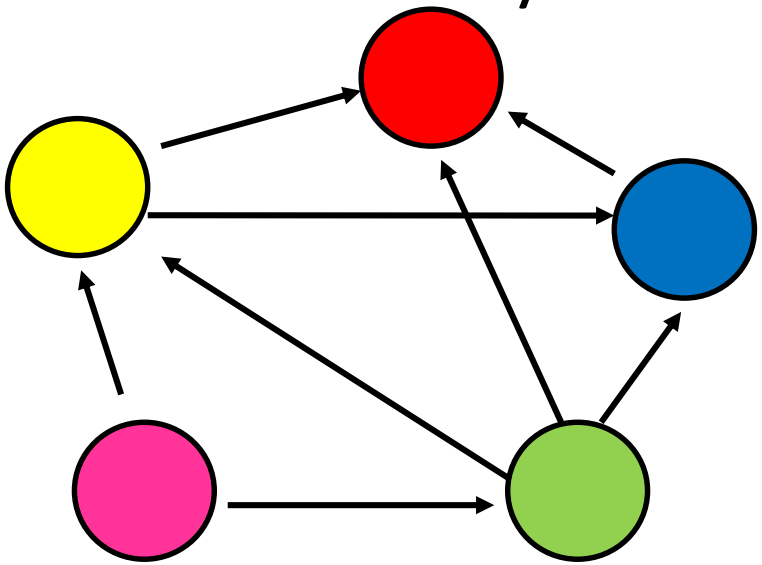
- After t multiplications we have:
  - $R^t x = \lambda_1^{t-1} \alpha_1 w_1 + \lambda_2^{t-1} a_2 w_2 + \cdots + \lambda_2^{t-1} a_r w_r$

- Normalizing leaves only the term w_1.

ABSORBING RANDOM WALKS
LABEL PROPAGATION
OPINION FORMATION ON SOCIAL NETWORKS

Random walk with absorbing nodes

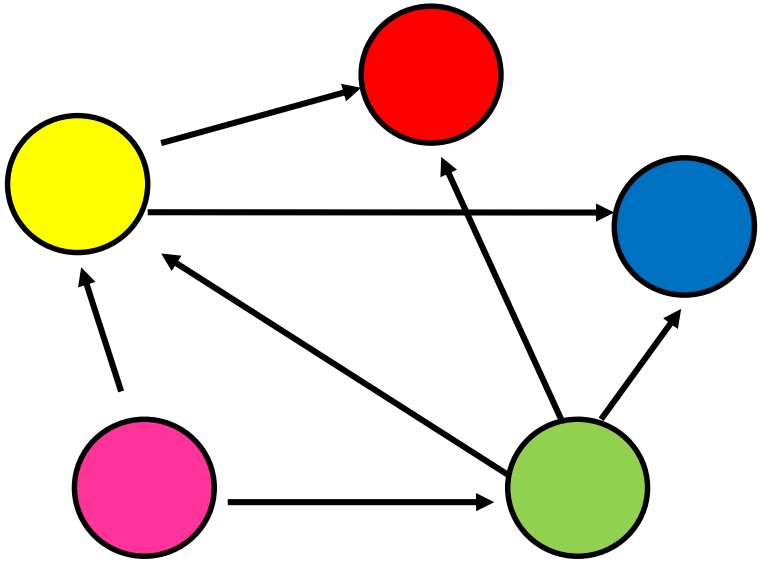
- What happens if we do a random walk on this graph? What is the stationary distribution?



- All the probability mass on the red sink node:
  - The red node is an absorbing node

Random walk with absorbing nodes

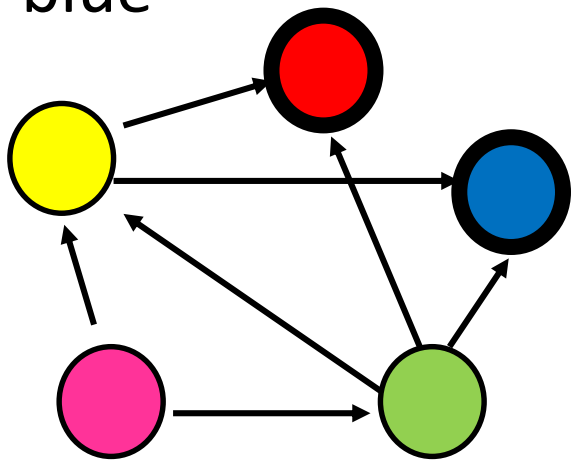
- What happens if we do a random walk on this graph? What is the stationary distribution?



- There are two absorbing nodes: the red and the blue.
- The probability mass will be divided between the two

Absorption probability

- If there are more than one absorbing nodes in the graph a random walk that starts from a non-absorbing node will be absorbed in one of them with some probability
  - The probability of absorption gives an estimate of how close the node is to red or blue



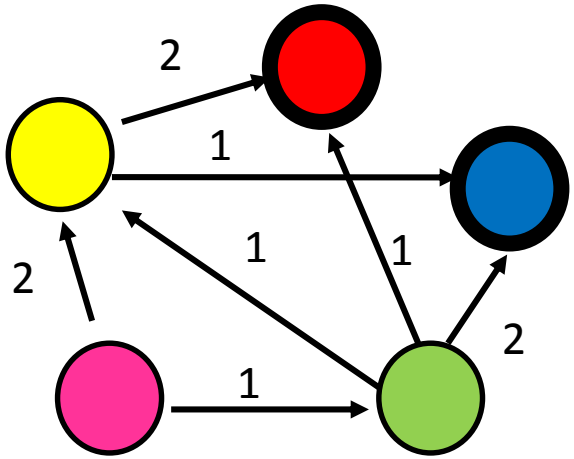
Absorption probability

- Computing the probability of being absorbed:
  - The absorbing nodes have probability 1 of being absorbed in themselves and zero of being absorbed in another node.
  - For the non-absorbing nodes, take the (weighted) average of the absorption probabilities of your neighbors
    - if one of the neighbors is the absorbing node, it has probability 1
  - Repeat until convergence (= very small change in probs)

$$P(Red|Pink) = \frac{2}{3}P(Red|Yellow) + \frac{1}{3}P(Red|Green)$$

$$P(Red|Green) = \frac{1}{4}P(Red|Yellow) + \frac{1}{4}$$

$$P(Red|Yellow) = \frac{2}{3}$$



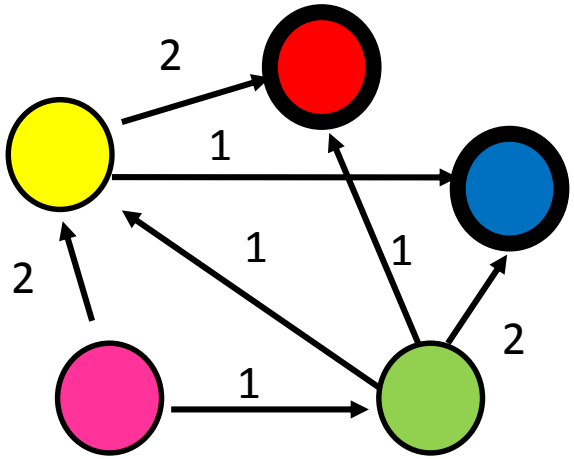
Absorption probability

- Computing the probability of being absorbed:
  - The absorbing nodes have probability 1 of being absorbed in themselves and zero of being absorbed in another node.
  - For the non-absorbing nodes, take the (weighted) average of the absorption probabilities of your neighbors
    - if one of the neighbors is the absorbing node, it has probability 1
  - Repeat until convergence (= very small change in probs)

$$P(Blue|Pink) = \frac{2}{3}P(Blue|Yellow) + \frac{1}{3}P(Blue|Green)$$

$$P(Blue|Green) = \frac{1}{4}P(Blue|Yellow) + \frac{1}{2}$$

$$P(Blue|Yellow) = \frac{1}{3}$$

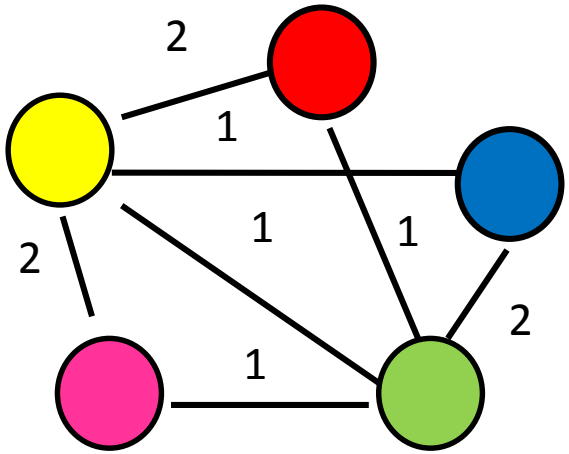


Why do we care?

- Why do we care to compute the absorbtion probability to sink nodes?
- Given a graph (directed or undirected) we can choose to make some nodes absorbing.
  - Simply direct all edges incident on the chosen nodes towards them.
- The absorbing random walk provides a measure of proximity of non-absorbing nodes to the chosen nodes.
  - Useful for understanding proximity in graphs
  - Useful for propagation in the graph
    - E.g, on a social network some nodes have high income, some have low income, to which income class is a non-absorbing node closer?

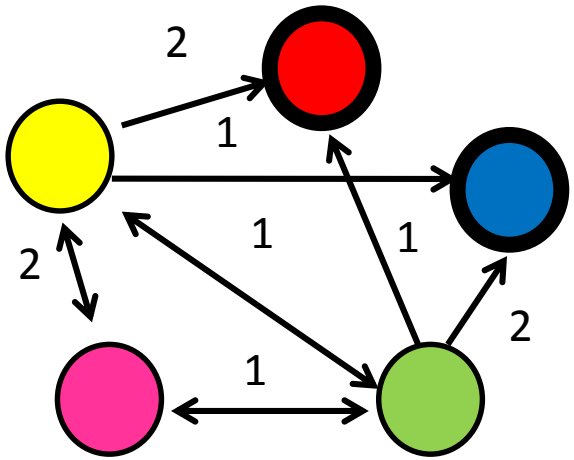
Example

- In this undirected graph we want to learn the proximity of nodes to the red and blue nodes



Example

- Make the nodes absorbing



Absorption probability

- Compute the absorbtion probabilities for red and blue

$$P(Red|Pink) = \frac{2}{3}P(Red|Yellow) + \frac{1}{3}P(Red|Green)$$

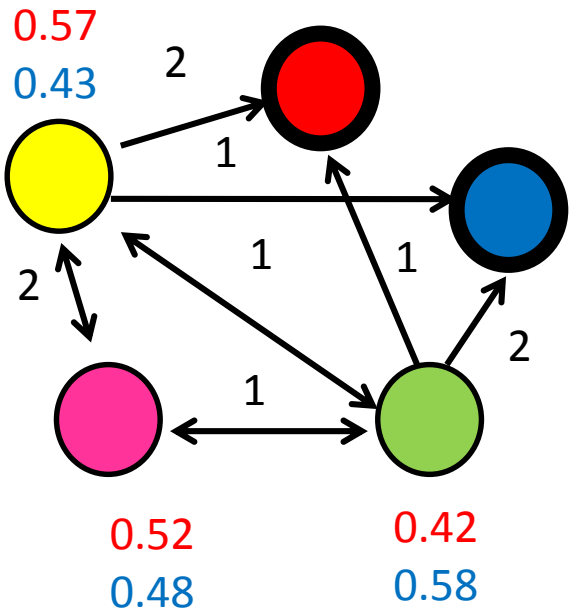
$$P(Red|Green) = \frac{1}{5}P(Red|Yellow) + \frac{1}{5}P(Red|Pink) + \frac{1}{5}$$

$$P(Red|Yellow) = \frac{1}{6}P(Red|Green) + \frac{1}{3}P(Red|Pink) + \frac{1}{3}$$

$$P(Blue|Pink) = 1 - P(Red|Pink)$$

$$P(Blue|Green) = 1 - P(Red|Green)$$

$$P(Blue|Yellow) = 1 - P(Red|Yellow)$$

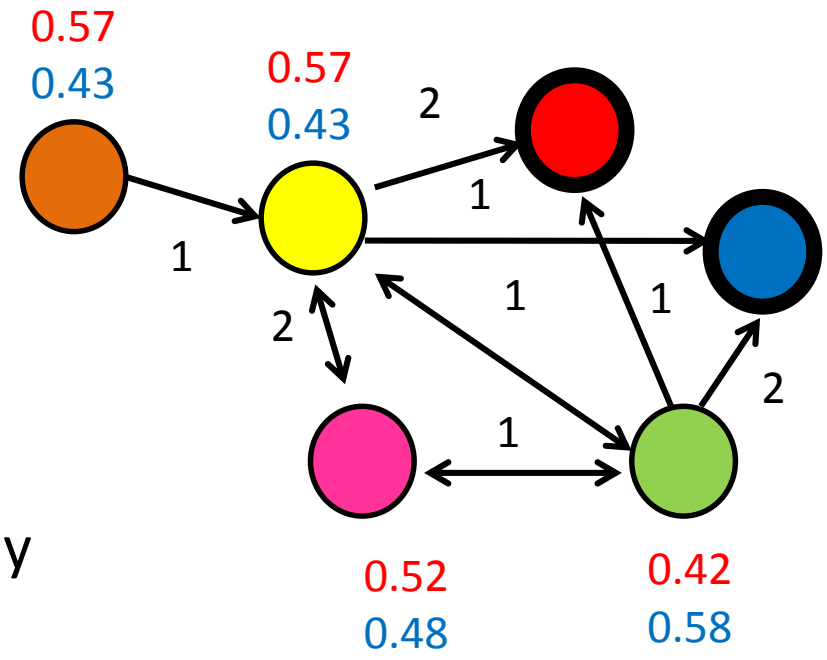


Penalizing long paths

- The orange node has the same probability of reaching red and blue as the yellow one

$$P(Red|Orange) = P(Red|Yellow)$$

$$P(Blue|Orange) = P(Blue|Yellow)$$



- Intuitively though it is further away

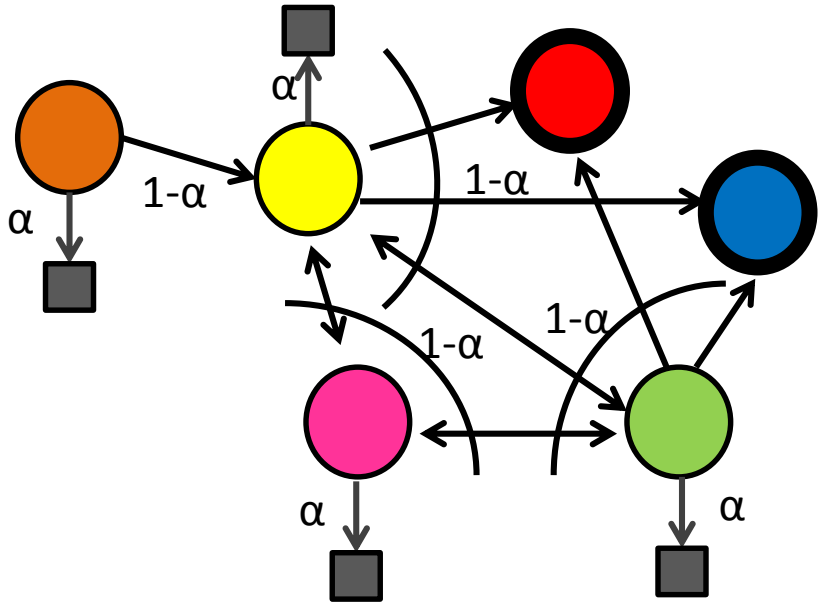
Penalizing long paths

- Add an universal absorbing node to which each node gets absorbed with probability α.

With probability α the random walk dies

With probability (1-α) the random walk continues as before

The longer the path from a node to an absorbing node the more likely the random walk dies along the way, the lower the absorbtion probability



$$P(Red|Green) = (1-\alpha)\left(\frac{1}{5}P(Red|Yellow) + \frac{1}{5}P(Red|Pink) + \frac{1}{5}\right)$$

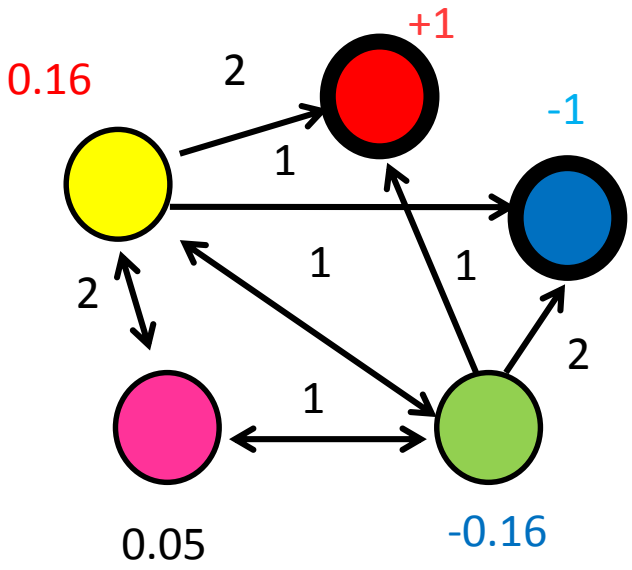
Propagating values

- Assume that Red has a positive value and Blue a negative value
  - Positive/Negative class, Positive/Negative opinion
- We can compute a value for all the other nodes in the same way
  - This is the expected value for the node

$$V(Pink) = \frac{2}{3}V(Yellow) + \frac{1}{3}V(Green)$$

$$V(Green) = \frac{1}{5}V(Yellow) + \frac{1}{5}V(Pink) + \frac{1}{5} - \frac{2}{5}$$

$$V(Yellow) = \frac{1}{6}V(Green) + \frac{1}{3}V(Pink) + \frac{1}{3} - \frac{1}{6}$$



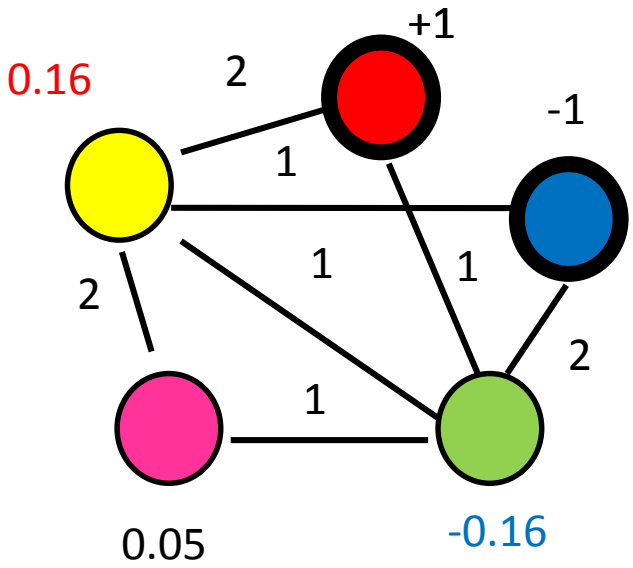
Electrical networks and random walks

- Our graph corresponds to an electrical network
- There is a positive voltage of +1 at the Red node, and a negative voltage -1 at the Blue node
- There are resistances on the edges inversely proportional to the weights (or conductance proportional to the weights)
- The computed values are the voltages at the nodes

$$V(Pink) = \frac{2}{3}V(Yellow) + \frac{1}{3}V(Green)$$

$$V(Green) = \frac{1}{5}V(Yellow) + \frac{1}{5}V(Pink) + \frac{1}{5} - \frac{2}{5}$$

$$V(Yellow) = \frac{1}{6}V(Green) + \frac{1}{3}V(Pink) + \frac{1}{3} - \frac{1}{6}$$



Opinion formation

- The value propagation can be used as a model of opinion formation.
- Model:
  - Opinions are values in [-1,1]
  - Every user u has an internal opinion s_u, and expressed opinion z_u.
  - The expressed opinion minimizes the personal cost of user u:

$$c(z_u) = (s_u - z_u)^2 + \sum_{v:v \text{ is a friend of } u} w_u(z_u - z_v)^2$$

    - Minimize deviation from your beliefs and conflicts with the society

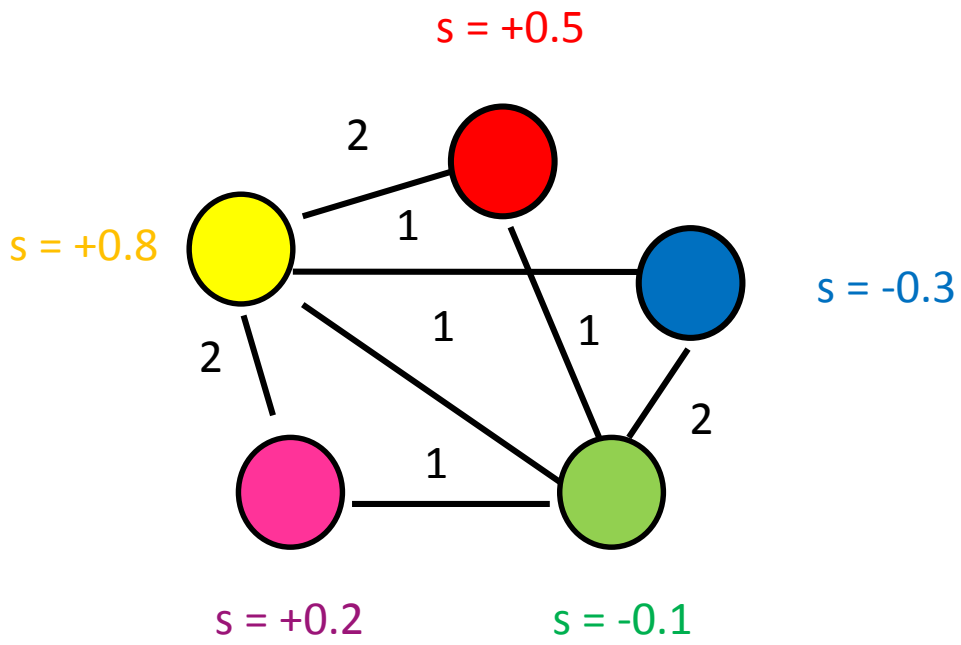
- If every user tries independently (selfishly) to minimize their personal cost then the best thing to do is to set z_u to the average of all opinions:

$$z_u = \frac{s_u + \sum_{v:v \text{ is a friend of } u} w_u z_u}{1 + \sum_{v:v \text{ is a friend of } u} w_u}$$

- This is the same as the value propagation we described before!

Example

- Social network with internal opinions



Example

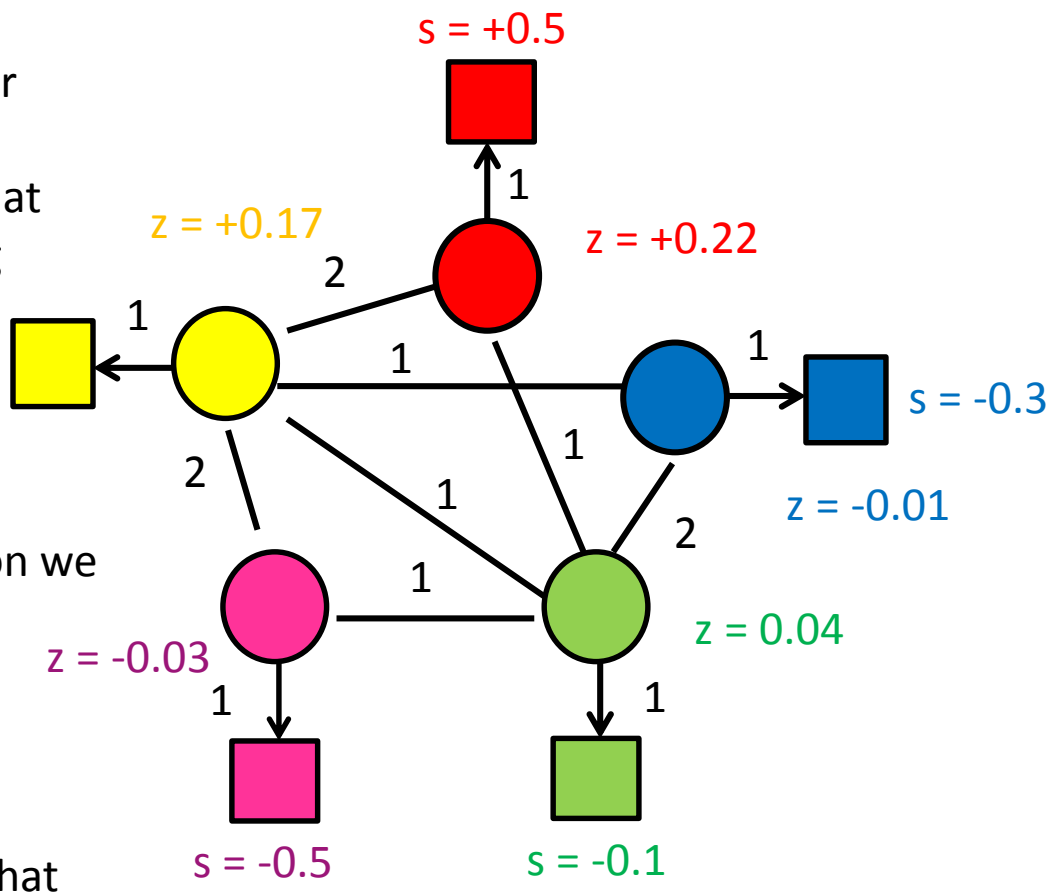
One absorbing node per user with value the internal opinion of the user

One non-absorbing node per user that links to the corresponding absorbing node

The external opinion for each node is computed using the value propagation we described before

- Repeated averaging

Intuitive model: my opinion is a combination of what I believe and what my social network believes.



Transductive learning

- If we have a graph of relationships and some labels on some nodes we can propagate them to the remaining nodes
  - Make the labeled nodes to be absorbing and compute the probability for the rest of the graph
  - E.g., a social network where some people are tagged as spammers
  - E.g., the movie-actor graph where some movies are tagged as action or comedy.

- This is a form of semi-supervised learning
  - We make use of the unlabeled data, and the relationships

- It is also called transductive learning because it does not produce a model, but just labels the unlabeled data that is at hand.
  - Contrast to inductive learning that learns a model and can label any new example

Implementation details

- Implementation is in many ways similar to the PageRank implementation
  - For an edge (u, v)instead of updating the value of v we update the value of u.
    - The value of a node is the average of its neighbors
  - We need to check for the case that a node u is absorbing, in which case the value of the node is not updated.
  - Repeat the updates until the change in values is very small.

LINK PREDICTION

The Problem

Link prediction problem: Given the links in a social network at time *t*, ***predict*** which edges that will be added to the network

- Which features to use?

User characteristics (profile), network interactions, topology

- Different from the problem of *inferring missing* (hidden) links (there is a temporal aspect, uses a static snapshot)

To save experimental effort in the laboratory or in the field

Applications

- *Recommending* new friends on online social networks.
- Predicting the participants or actors in events
- Suggesting interactions between the members of a company/organization
- Predicting connections between members of terrorist organizations who have not been directly observed to work together
- Suggesting collaborations between researchers based on co-authorship.

- Network evolution model

Link Prediction

Unsupervised (usually, assign scores *based on similarity* of endpoints)

Supervised (given some positive (created edges) and negative examples (nonexistent edges)

Classification Problem

Problem: Class imbalance

Instead of 0/1, rank each edge by its probability to appear in the network

D. Liben-Nowell, D. and J. Kleinberg, ***The link-prediction problem for social networks.*** *Journal of the American Society for Information Science and Technology*, 58(7) 1019–1031 (2007)

The Problem

Link prediction problem: Given the links in a social network at time *t*, ***predict*** the edges that will be added to the network during the time interval from time *t* to a given future time *t'*

- Which features to use?

Based solely on the *topology* of the network (social proximity) *(the more general problem also considers attributes of the nodes and links)*

Problem Formulation I

Consider a social network $G = (V, E)$ where each edge $e = \langle y, v \rangle \in E$ represents an interaction between u and v that took place at a particular time $t(e)$

(multiple interactions between two nodes as parallel edges with different timestamps)

$G[t, t']$: subgraph of G consisting of all edges with a timestamp between t and t', $t < t'$,

- For four times, $t_0 < t'_0 < t_1 < t'_1$,

Given $G[t_0, t'_0]$, we wish to output a list of edges not in $G[t_0, t'_0]$ that are predicted to appear in $G[t_1, t'_1]$

✓ $[t_0, t'_0]$ training interval
✓ $[t_1, t'_1]$ test interval

Problem Formulation II

What about new nodes (node not in the training interval)?

Two parameters: $K_{training}$ and K_{test}

Core: all nodes that are incident to at least $K_{training}$ edges in $G[t_0, t'_0]$, and at least K_{test} edges in $G[t_1, t'_1]$

❖*Predict new edges between the nodes in Core*

Example Dataset: co-authorship

	training period			Core		
	authors	papers	collaborations[1]	authors	$\|E_{old}\|$	$\|E_{new}\|$
astro-ph	5343	5816	41852	1561	6178	5751
cond-mat	5469	6700	19881	1253	1899	1150
gr-qc	2122	3287	5724	486	519	400
hep-ph	5414	10254	47806	1790	6654	3294
hep-th	5241	9498	15842	1438	2311	1576

t_0 = 1994, t'_0 = 1996: **training interval -> [1994, 1996]**
t_1 = 1997, t'_1 = 1999: **test interval -> [1997, 1999]**

- G_{collab} = <A, E_{old}> = G[1994, 1996]
- E_{new}: authors in A that co-author a paper during the test interval but not during the training interval

$\kappa_{training}$ = *3,* κ_{test} = 3: **Core** consists of all authors who have written at least 3 papers during the training period and at least 3 papers during the test period

Predict E_{new}

How to Evaluate the Prediction

Each link predictor *p* outputs a ranked list L_p of pairs in $A \times A - E_{old}$: predicted new collaborations in decreasing order of confidence

Actual edges:

$$E*new = E_{new} \cap (Core \times Core),\ n = |E*_{new}|$$

Evaluation method: *Size of the intersection* of

- the first *n* edge predictions from L_p that are in Core × Core *(predicted)* and
- the set $E*_{new}$ *(actual)*

❖How many of the top-*n* predictions are correct (precision?)

Methods for Link Prediction

Assign a connection weight score(x, y) to each pair of nodes <x, y> based on the input graph (G_{collab}) and produce a ranked list of decreasing order of score

How to assign the score between two nodes x and y?

✓Some form of **similarity** or **node proximity**

Most measures focus on the giant component

Methods for Link Prediction: Shortest Path

For $x, y \in A \times A - E_{old}$,

score(x, y) = (negated) length of shortest path between x and y

✓ If there are more than *n* pairs of nodes tied for the shortest path length, order them at random.

Geodesic distance: number of edges in the shortest path

Methods for Link Prediction: Neighborhood-based

The "larger" the overlap of the neighbors of two nodes, the more likely to be linked in the future

Let Γ(x) denote the set of neighbors of x in G_{collab}

Common neighbors:

$$\text{score}(x, y) = |\Gamma(x) \cap \Gamma(y)|$$

A adjacency matrix -> $A_{x,y}^2$
Number of different paths of length 2

Jaccard coefficient:

$$\text{score}(x, y) = \frac{|\Gamma(x) \cap \Gamma(y)|}{|\Gamma(x) \cup \Gamma(y)|}$$

The probability that both x and y have a feature f, for a randomly selected feature that either x or y has

Methods for Link Prediction: Neighborhood-based

Adamic/Adar:

$$\text{score}(x, y) = \sum_{z \in \Gamma(x) \cap \Gamma(y)} \frac{1}{\log |\Gamma(z)|}$$

✓ Assigns large weights to common neighbors *z* of *x* and *y* which themselves have few neighbors (weight rare features more heavily)

connections to “unpopular” nodes are more relevant

Methods for Link Prediction: Neighborhood-based

Preferential attachment:

the probability that a new edge has node x as its endpoint is proportional to |Γ(x)|, i.e., nodes like to form ties with 'popular' nodes

$$\text{score}(x, y) = |\Gamma(x)||\Gamma(y)|$$

✓ Researchers found empirical evidence to suggest that co-authorship is correlated with the product of the neighborhood sizes

Methods for Link Prediction: based on the ensemble of all paths

Not just the shortest, but *all* paths between two nodes

Methods for Link Prediction: based on the ensemble of all paths

Katz$_\beta$ measure:

$$\text{score}(x, y) := \sum_{\ell=1}^{\infty} \beta^{\ell} \cdot |\text{paths}_{x,y}^{\langle \ell \rangle}|$$

$$\sum_{l=1}^{\infty} \beta^{l} \cdot |\text{paths}_{xy}^{\langle l \rangle}| = \beta A_{xy} + \beta^2 (A^2)_{xy} + \beta^3 (A^3)_{xy} + \cdots$$

Sum over all paths of length *l*, β > 0 is a parameter of the predictor, exponentially damped to count short paths more heavily

✓*Small β predictions much like common neighbors*

$$(I - \beta A)^{-1} - I$$

1. *Unweighted* version, in which $\text{path}_{x,y}^{(1)}$ = **1**, if x and y have collaborated, **0** otherwise
2. *Weighted* version, in which $\text{path}_{x,y}^{(1)}$ = **#times** x and y have collaborated

Methods for Link Prediction: based on the ensemble of all paths

Consider a random walk on G_{collab} that starts at x and iteratively moves to a neighbor of x chosen uniformly at random from $\Gamma(x)$.

The Hitting Time $H_{x,y}$ from x to y is the expected number of steps it takes for the random walk starting at x to reach y.

$$score(x, y) = -H_{x,y}$$

(symmetric version) The Commute Time $C_{x,y}$ from x to y is the expected number of steps to travel from x to y and from y to x

$$score(x, y) = -(H_{x,y} + H_{y,x})$$

Can also consider stationary-normed versions:

$$score(x, y) = -H_{x,y}\,\pi_y$$

$$score(x, y) = -(H_{x,y}\,\pi_y + H_{y,x}\,\pi_x)$$

Methods for Link Prediction: based on the ensemble of all paths

The hitting time and commute time measures are sensitive to parts of the graph far away from x and y -> periodically ***reset the walk***

Random walk on G_{collab} that starts at *x* and has a probability of α of returning to x at each step.

Rooted (Personalized) Page Rank: Starts from x, with probability (1 – α) moves to a random neighbor and with probability α returns to x

score(x, y) = stationary probability of y in a rooted PageRank

Methods for Link Prediction: based on the ensemble of all paths

SimRank

$$\text{similarity}(x, y) := \gamma \cdot \frac{\sum_{a \in \Gamma(x)} \sum_{b \in \Gamma(y)} \text{similarity}(a, b)}{|\Gamma(x)| \cdot |\Gamma(y)|}$$

score(x, y) = similarity(x, y)

The expected value of γ^l where l is a random variable giving the time at which random walks started from x and y first meet

Methods for Link Prediction: High-level approaches

Low rank approximations

A adjacency matrix

Apply SVD (singular value decomposition)

The rank-k matrix that best approximates A

Methods for Link Prediction: High-level approaches

Unseen Bigrams

Unseen bigrams: pairs of word that co-occur in a test corpus, but not in the corresponding training corpus

Not just score(x, y) but *score(z, y) for nodes z that are similar to x*

$S_x^{(\delta)}$ the δ nodes most related to x

$$\text{score}^*_{unweighted}(x, y) := \left|\{z : z \in \Gamma(y) \cap S_x^{\langle\delta\rangle}\}\right|$$

$$\text{score}^*_{weighted}(x, y) := \sum_{z \in \Gamma(y) \cap S_x^{\langle\delta\rangle}} \text{score}(x, z)$$

Methods for Link Prediction: High-level approaches

Clustering

- Compute score(x, y) for al edges in E_{old}
- Delete the (1-*p*) fraction of these edges for which the score is the lowest, for some parameter *p*
- Recompute score(x, y) for all pairs in the subgraph

Evaluation: baseline

Baseline: random predictor

Randomly select pairs of authors who did not collaborate in the training interval

Probability that a random prediction is correct,

Number of possible predictions $\binom{|\text{Core}|}{2} - |E_{old}|$

Correct predictions $|E_{new}|$

$$\frac{|E_{new}|}{\binom{|\text{Core}|}{2} - |E_{old}|}$$

In the datasets, from 0.15% (cond-mat) to 0.48% (astro-ph)

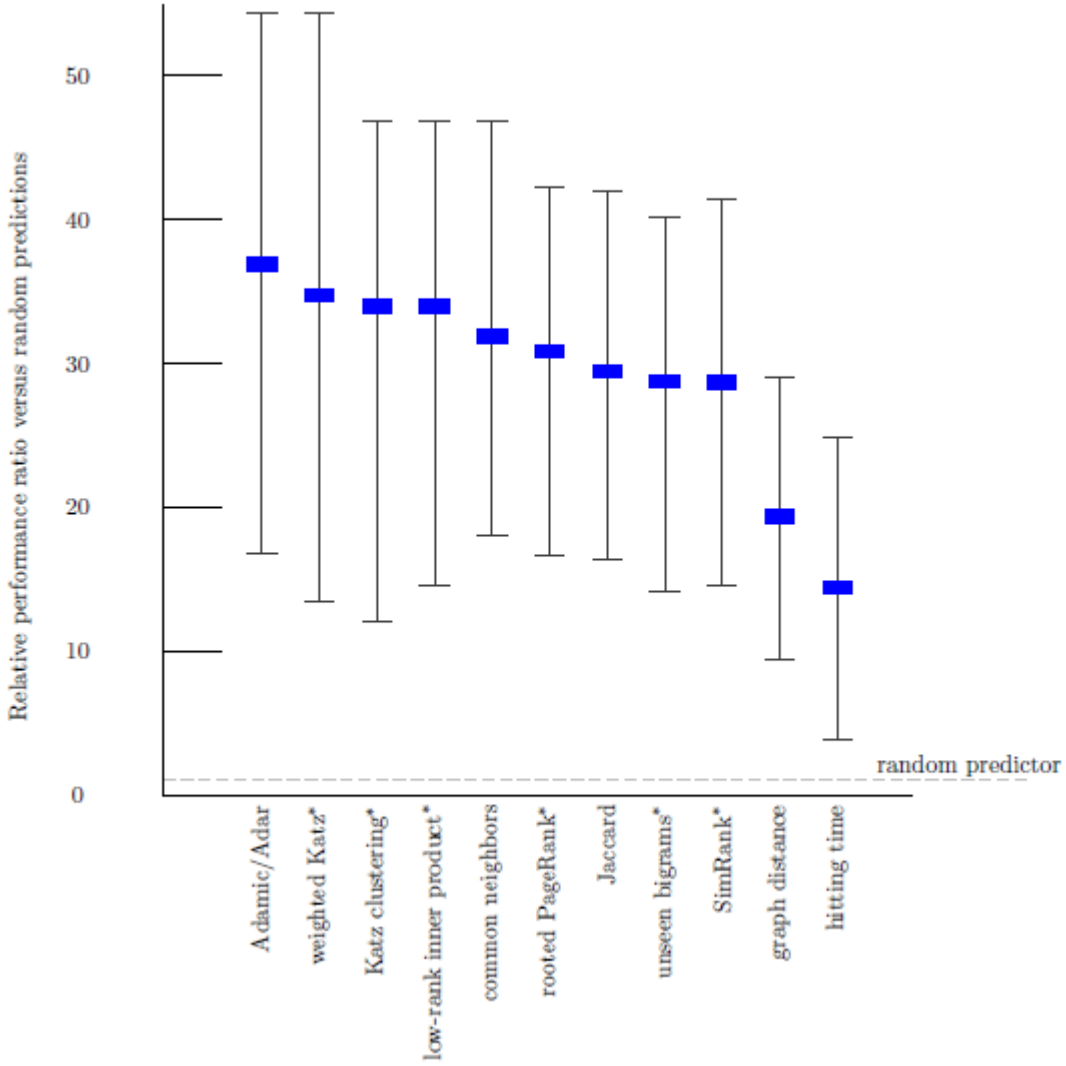
Evaluation: Factor improvement over random

predictor		astro-ph	cond-mat	gr-qc	hep-ph	hep-th
probability that a random prediction is correct		0.475%	0.147%	0.341%	0.207%	0.153%
graph distance (all distance-two pairs)		*9.4*	*25.1*	*21.3*	*12.0*	*29.0*
common neighbors		**18.0**	**40.8**	**27.1**	**26.9**	**46.9**
preferential attachment		4.7	6.0	7.5	*15.2*	7.4
Adamic/Adar		*16.8*	**54.4**	**30.1**	**33.2**	**50.2**
Jaccard		*16.4*	**42.0**	19.8	**27.6**	*41.5*
SimRank	$\gamma = 0.8$	*14.5*	*39.0*	*22.7*	*26.0*	*41.5*
hitting time		6.4	23.7	*24.9*	3.8	13.3
hitting time—normed by stationary distribution		5.3	23.7	11.0	11.3	21.2
commute time		5.2	15.4	**33.0**	*17.0*	23.2
commute time—normed by stationary distribution		5.3	16.0	11.0	11.3	16.2
rooted PageRank	$\alpha = 0.01$	*10.8*	*27.8*	**33.0**	*18.7*	*29.1*
	$\alpha = 0.05$	*13.8*	*39.6*	**35.2**	*24.5*	*41.1*
	$\alpha = 0.15$	*16.6*	**40.8**	**27.1**	**27.5**	*42.3*
	$\alpha = 0.30$	*17.1*	**42.0**	*24.9*	**29.8**	*46.5*
	$\alpha = 0.50$	*16.8*	**40.8**	*24.2*	**30.6**	*46.5*
Katz (weighted)	$\beta = 0.05$	3.0	21.3	19.8	2.4	12.9
	$\beta = 0.005$	*13.4*	**54.4**	**30.1**	*24.0*	**51.9**
	$\beta = 0.0005$	*14.5*	**53.8**	**30.1**	**32.5**	**51.5**
Katz (unweighted)	$\beta = 0.05$	*10.9*	**41.4**	**37.4**	*18.7*	**47.7**
	$\beta = 0.005$	*16.8*	**41.4**	**37.4**	*24.1*	**49.4**
	$\beta = 0.0005$	*16.7*	**41.4**	**37.4**	*24.8*	**49.4**

Evaluation: Factor improvement over random

predictor		astro-ph	cond-mat	gr-qc	hep-ph	hep-th
probability that a random prediction is correct		0.475%	0.147%	0.341%	0.207%	0.153%
graph distance (all distance-two pairs)		*9.4*	*25.1*	*21.3*	*12.0*	*29.0*
common neighbors		**18.0**	**40.8**	**27.1**	**26.9**	**46.9**
Low-rank approximation: Inner product	rank = 1024	*15.2*	**53.8**	**29.3**	**34.8**	**49.8**
	rank = 256	*14.6*	**46.7**	**29.3**	**32.3**	**46.9**
	rank = 64	*13.0*	**44.4**	**27.1**	**30.7**	**47.3**
	rank = 16	*10.0*	21.3	**31.5**	**27.8**	*35.3*
	rank = 4	8.8	15.4	**42.5**	*19.5*	22.8
	rank = 1	6.9	5.9	**44.7**	*17.6*	14.5
Low-rank approximation: Matrix entry	rank = 1024	8.2	16.6	6.6	*18.5*	21.6
	rank = 256	*15.4*	*36.1*	8.1	*26.2*	*37.4*
	rank = 64	*13.7*	**46.1**	16.9	**28.1**	*40.7*
	rank = 16	9.1	21.3	*26.4*	*23.1*	*34.0*
	rank = 4	8.8	15.4	**39.6**	*20.0*	22.4
	rank = 1	6.9	5.9	**44.7**	*17.6*	14.5
Low-rank approximation: Katz ($\beta = 0.005$)	rank = 1024	*11.4*	*27.2*	**30.1**	**27.0**	*32.0*
	rank = 256	*15.4*	**42.0**	11.0	**34.2**	*38.6*
	rank = 64	*13.1*	**45.0**	19.1	**32.2**	*41.1*
	rank = 16	9.2	21.3	**27.1**	*24.8*	*34.9*
	rank = 4	7.0	15.4	**41.1**	*19.7*	22.8
	rank = 1	0.4	5.9	**44.7**	*17.6*	14.5
unseen bigrams (weighted)	common neighbors, $\delta = 8$	*13.5*	*36.7*	**30.1**	*15.6*	**46.9**
	common neighbors, $\delta = 16$	*13.4*	*39.6*	**38.9**	*18.5*	**48.6**
	Katz ($\beta = 0.005$), $\delta = 8$	*16.8*	*37.9*	*24.9*	*24.1*	**51.1**
	Katz ($\beta = 0.005$), $\delta = 16$	*16.5*	*39.6*	**35.2**	*24.7*	**50.6**
unseen bigrams (unweighted)	common neighbors, $\delta = 8$	*14.1*	*40.2*	**27.9**	*22.2*	*39.4*
	common neighbors, $\delta = 16$	*15.3*	*39.0*	**42.5**	*22.0*	*42.3*
	Katz ($\beta = 0.005$), $\delta = 8$	*13.1*	*36.7*	**32.3**	*21.6*	*37.8*
	Katz ($\beta = 0.005$), $\delta = 16$	*10.3*	*29.6*	**41.8**	*12.2*	*37.8*
clustering: Katz ($\beta_1 = 0.001, \beta_2 = 0.1$)	$\rho = 0.10$	7.4	*37.3*	**46.9**	**32.9**	*37.8*
	$\rho = 0.15$	*12.0*	**46.1**	**46.9**	*21.0*	*44.0*
	$\rho = 0.20$	4.6	*34.3*	19.8	*21.2*	*35.7*
	$\rho = 0.25$	3.3	*27.2*	20.5	*19.4*	17.4

Evaluation: Average relevance performance (random)

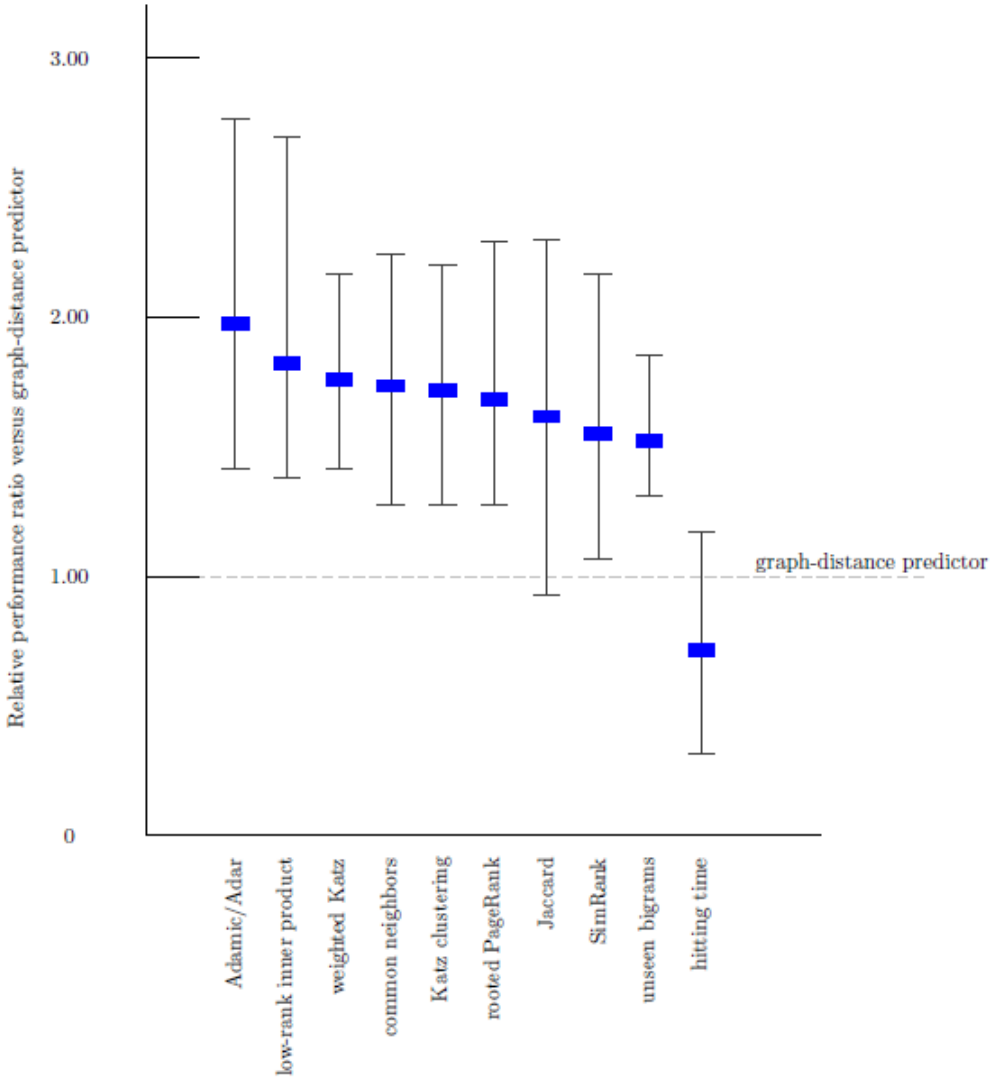


average ratio over the five datasets of the given predictor's performance versus a baseline predictor's performance.

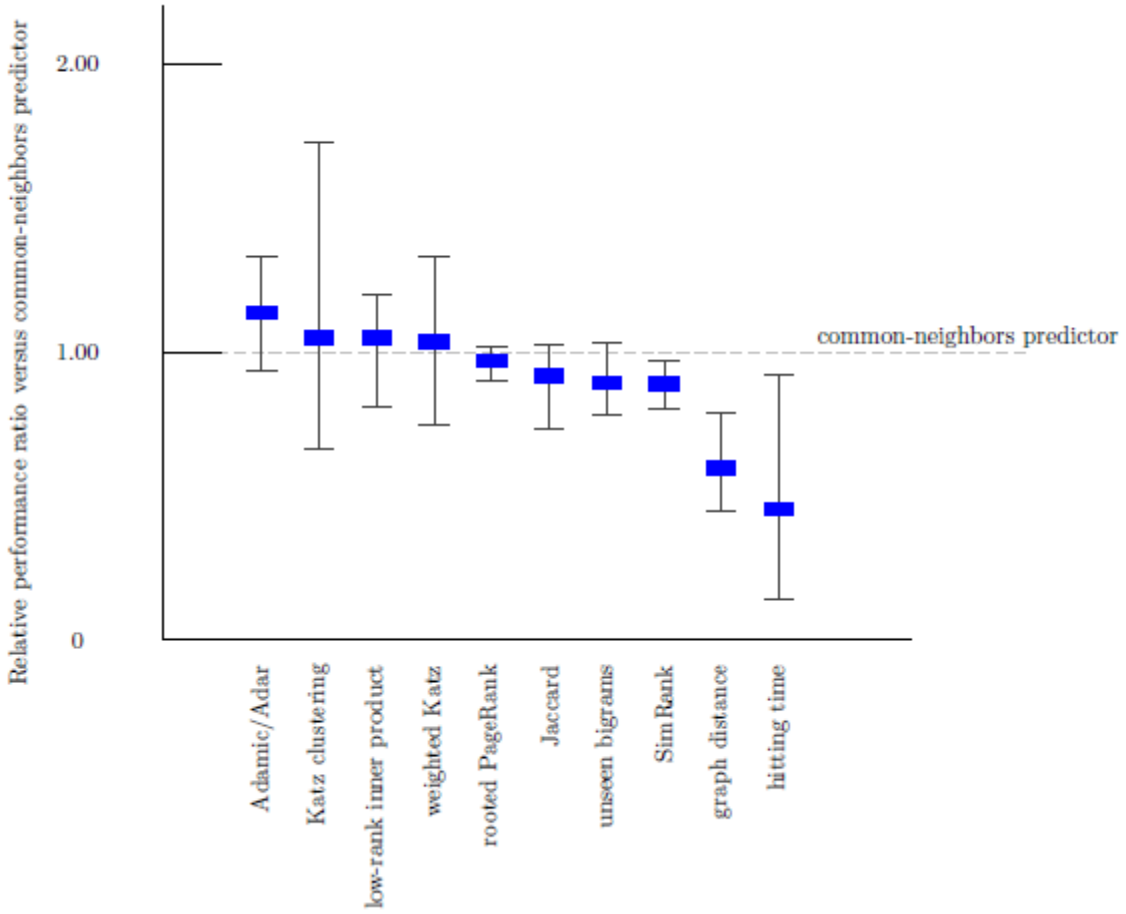
The error bars indicate the minimum and maximum of this ratio over the five datasets.

The parameters for the starred predictors are as follows: (1) for weighted Katz, $\beta = 0.005$; (2) for Katz clustering, $\beta 1 = 0.001$; $\rho = 0.15$; $\beta 2 = 0.1$; (3) for low-rank inner product, rank = 256; (4) for rooted Pagerank, $\alpha = 0.15$; (5) for unseen bigrams, unweighted common neighbors with $\delta = 8$; and (6) for SimRank, $\gamma = 0.8$.

Evaluation: Average relevance performance (distance)



Evaluation: Average relevance performance (neighbors)



Evaluation: prediction overlap

	Adamic/Adar	Katz clustering	common neighbors	hitting time	Jaccard's coefficient	weighted Katz	low-rank inner product	rooted Pagerank	SimRank	unseen bigrams
Adamic/Adar	1150	638	520	193	442	1011	905	528	372	486
Katz clustering		1150	411	182	285	630	623	347	245	389
common neighbors			1150	135	506	494	467	305	332	489
hitting time				1150	87	191	192	247	130	156
Jaccard's coefficient					1150	414	382	504	845	458
weighted Katz						1150	1013	488	344	474
low-rank inner product							1150	453	320	448
rooted Pagerank								1150	678	461
SimRank									1150	423
unseen bigrams										1150

❖ How much similar are the predictions made by the different methods (common predictions)?

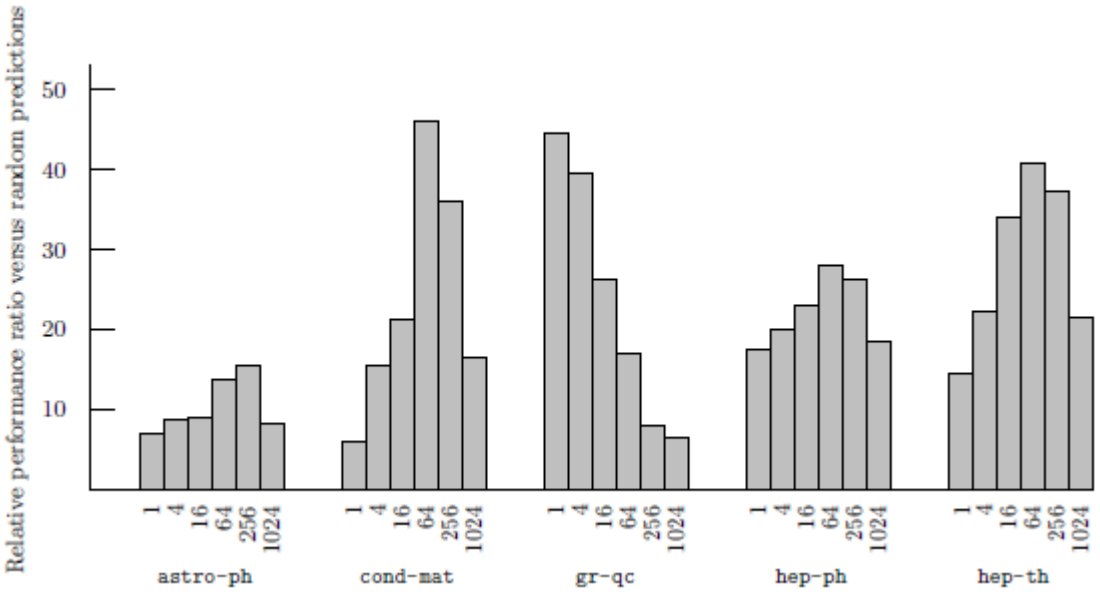
Why?

correct

	Adamic/Adar	Katz clustering	common neighbors	hitting time	Jaccard's coefficient	weighted Katz	low-rank inner product	rooted Pagerank	SimRank	unseen bigrams
Adamic/Adar	92	65	53	22	43	87	72	44	36	49
Katz clustering		78	41	20	29	66	60	31	22	37
common neighbors			69	13	43	52	43	27	26	40
hitting time				40	8	22	19	17	9	15
Jaccard's coefficient					71	41	32	39	51	43
weighted Katz						92	75	44	32	51
low-rank inner product							79	39	26	46
rooted Pagerank								69	48	39
SimRank									66	34
unseen bigrams										68

Evaluation: datasets

❖ How much does the performance of the different methods depends on the dataset?



- (rank) On 4 of the 5 datasets best at an intermediate rank
  On qr-qc, best at rank 1, does it have a “simpler” structure”?
- On hep-ph, preferential attachment the best
- Why is astro-ph “difficult”?

The culture of physicists and physics collaboration

Evaluation: small world

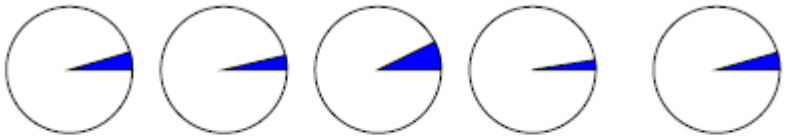
The shortest path even in unrelated disciplines is often very short

Evaluation: restricting to distance three

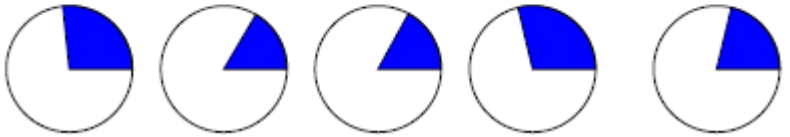
Many pairs of authors separated by a graph distance of 2 who will not collaborate and many pairs who collaborate at distance greater than 2

Disregard all distance 2 pairs

Proportion of distance-two pairs that form an edge:



Proportion of new edges that are between distance-two pairs:



astro-ph cond-mat gr-qc hep-ph hep-th

	astro-ph	cond-mat	gr-qc	hep-ph	hep-th
# pairs at distance two	33862	5145	935	37687	7545
# new collaborations at distance two	1533	190	68	945	335
# new collaborations	5751	1150	400	3294	1576

predictor		astro-ph	cond-mat	gr-qc	hep-ph	hep-th
graph distance (all distance-three pairs)		2.8	5.4	7.7	4.0	8.6
preferential attachment		3.2	2.6	8.6	4.7	1.4
SimRank	$\gamma = 0.8$	5.9	14.3	10.6	7.6	21.9
hitting time		4.4	10.1	13.7	4.5	4.7
hitting time—normed by stationary distribution		2.0	2.5	0.0	2.5	6.6
commute time		3.8	5.9	21.1	5.9	6.6
commute time—normed by stationary distribution		2.6	0.8	1.1	4.8	4.7
rooted PageRank	$\alpha = 0.01$	4.6	12.7	21.1	6.5	12.6
	$\alpha = 0.05$	5.3	13.5	21.1	8.7	16.6
	$\alpha = 0.15$	5.4	11.8	18.0	10.7	19.9
	$\alpha = 0.30$	5.8	13.5	8.4	11.6	19.9
	$\alpha = 0.50$	6.3	15.2	7.4	12.7	19.9
Katz (weighted)	$\beta = 0.05$	1.5	5.9	11.6	2.3	2.7
	$\beta = 0.005$	5.5	14.3	28.5	4.2	12.6
	$\beta = 0.0005$	6.2	13.5	27.5	4.2	12.6
Katz (unweighted)	$\beta = 0.05$	2.3	12.7	30.6	9.0	12.6
	$\beta = 0.005$	9.1	11.8	30.6	5.1	17.9
	$\beta = 0.0005$	9.2	11.8	30.6	5.1	17.9
Low-rank approximation: Inner product	rank = 1024	2.3	2.5	9.5	4.0	6.0
	rank = 256	4.8	5.9	5.3	9.9	10.6
	rank = 64	3.8	12.7	5.3	7.1	11.3
	rank = 16	5.3	6.7	6.3	6.8	15.3
	rank = 4	5.1	6.7	32.7	2.0	4.7
	rank = 1	6.1	2.5	32.7	4.2	8.0
Low-rank approximation: Matrix entry	rank = 1024	4.1	6.7	6.3	5.9	13.3
	rank = 256	3.8	8.4	3.2	8.5	19.9
	rank = 64	2.9	11.8	2.1	4.0	10.0
	rank = 16	4.4	8.4	4.2	5.9	16.6
	rank = 4	4.9	6.7	27.5	2.0	4.7
	rank = 1	6.1	2.5	32.7	4.2	8.0
Low-rank approximation: Katz ($\beta = 0.005$)	rank = 1024	4.3	6.7	28.5	5.9	13.3
	rank = 256	3.6	8.4	3.2	8.5	20.6
	rank = 64	2.8	11.8	2.1	4.2	10.6
	rank = 16	5.0	8.4	5.3	5.9	15.9
	rank = 4	5.2	6.7	28.5	2.0	4.7
	rank = 1	0.3	2.5	32.7	4.2	8.0
unseen bigrams (weighted)	common neighbors, $\delta = 8$	5.8	6.7	14.8	4.2	23.9
	common neighbors, $\delta = 16$	7.9	9.3	28.5	5.1	19.3
	Katz ($\beta = 0.005$), $\delta = 8$	5.2	10.1	22.2	2.8	17.9
	Katz ($\beta = 0.005$), $\delta = 16$	6.6	10.1	29.6	3.7	15.3
unseen bigrams (unweighted)	common neighbors, $\delta = 8$	5.4	5.1	13.7	4.5	21.3
	common neighbors, $\delta = 16$	6.3	8.4	25.3	4.8	21.9
	Katz ($\beta = 0.005$), $\delta = 8$	4.1	7.6	22.2	2.0	17.3
	Katz ($\beta = 0.005$), $\delta = 16$	4.3	4.2	28.5	3.1	16.6
clustering: Katz ($\beta_1 = 0.001, \beta_2 = 0.1$)	$\rho = 0.10$	3.2	4.2	31.7	7.1	8.6
	$\rho = 0.15$	4.6	4.2	32.7	7.6	6.6
	$\rho = 0.20$	2.3	5.9	7.4	4.5	8.0
	$\rho = 0.25$	2.0	11.8	6.3	6.8	5.3

Evaluation: the breadth of data

Three additional datasets

1. Proceedings of STOC and FOCS
2. Papers for Citeseer
3. All five of the arXiv sections

Common neighbors vs Random

STOC/FOCS	arXiv sections	combined arXiv sections	Citeseer
6.1	18.0—46.9	71.2	147.0

Future Directions

❖ Improve performance. Even the best (Katz clustering on gr-qc) correct on only about 16% of its prediction

❖ Improve efficiency on very large networks (approximation of distances)

❖ Treat more recent collaborations as more important

❖ Additional information (paper titles, author institutions, etc)
To some extent latently present in the graph

Future Directions

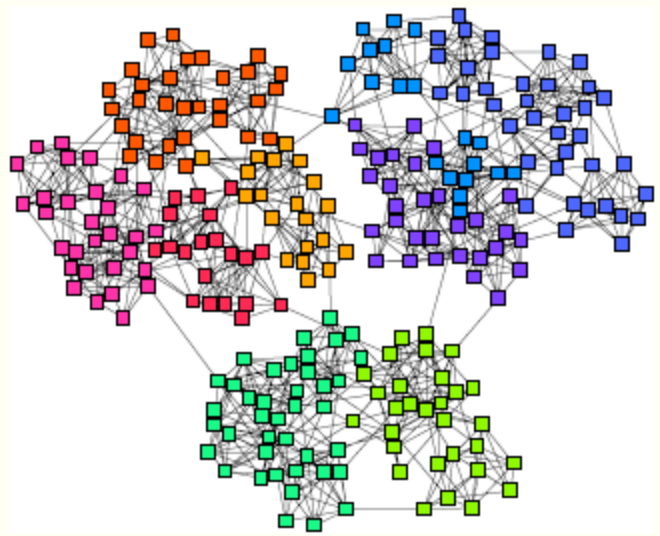
❖ Consider bipartite graph (e.g., some form of an affiliation network)

❖ Apply classification techniques from machine learning

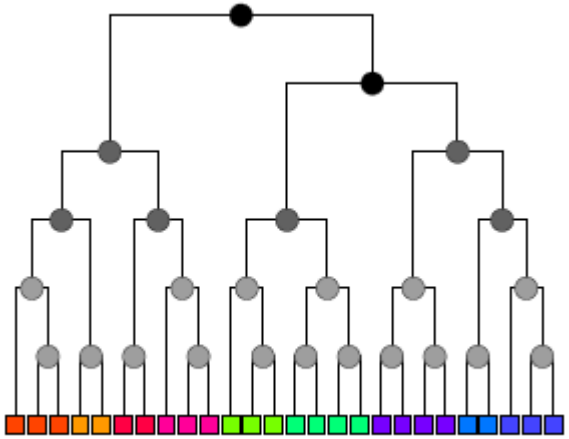
A simple binary classification problem: given two nodes x and y predict whether <x, y> is 1 or 0

Aaron Clauset, Cristopher Moore & M. E. J. Newman. ***Hierarchical structure and the prediction of missing links in network,*** Nature, 453, 98-101 (2008)

Hierarchical Random Graphs



Graph *G* with *n* nodes



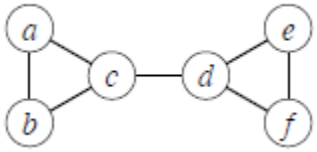
Dendrogram *D* a binary tree with *n* leaves
Each internal node corresponds to the group of nodes that descend from it

Each internal node r of the dendrogram is associated with a probability p_r that a pair of vertices in the left and right subtrees of that node are connected

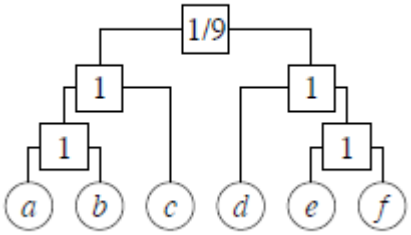
Given two nodes i and j of G the probability p_{ij} that they are connected by an edge is equal to p_r where r is their lowest common ancestor

Hierarchical Random Graphs

Example



graph



Possible dendrogram

Assortativity (dense connections within groups of nodes and sparse between them) -> probabilities p_r decrease as we move up the tree

- Given D and the probabilities p_r, we can generate a graph, called a hierarchical random graph

D: topological structure and parameters $\{p_r\}$

Hierarchical Random Graphs

Use to *predict missing interactions* in the network

- Given an observed but incomplete network, generate a set of hierarchical random graphs (i.e., a dendrogam and the associated probabilities) that **fit** the network (using statistical inference)

- Then look for pair of nodes that have a *high probability* of connection

Is this better than link prediction?

Experiments show that link prediction works well for strongly assortative networks (e.g, collaboration, citation) but not for networks that exhibit more general structure (e.g., food webs)

A rough idea of how to generate the model

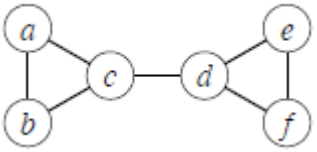
r a node in dendrogram D

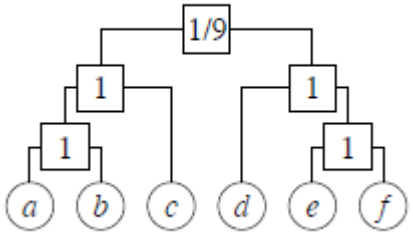
E_r the number of edges in G whose endpoints have r as their lowest common ancestor in D,

L_r and R_r the numbers of leaves in the left and right subtrees rooted at r

Then the likelihood of the hierarchical random graph is

$$\mathcal{L}(D, \{p_r\}) = \prod_{r \in D} p_r^{E_r} (1 - p_r)^{L_r R_r - E_r}$$

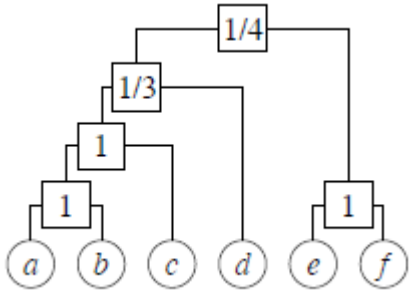




L(D1) = $(1/9)(8/9)^8$ = 0.0433..

If we fix the dendrogram D, it is easy to find the probabilities $\{p_r\}$ that maximize L(D, $\{p_r\}$). For each r, they are given by the fraction of potential edges between the two subtrees of r that actually appear in the graph G.

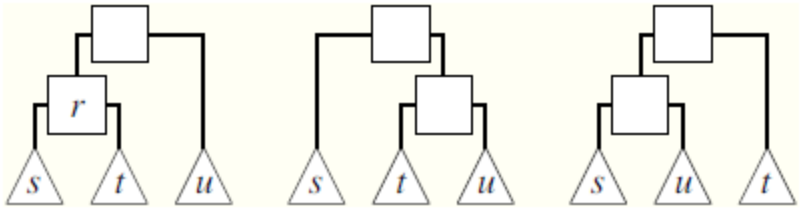
$$\overline{p}_r = \frac{E_r}{L_r R_r}$$



L(D2) = $(1/3)(2/3)^2(1/4)^2(3/4)^6$ = 0.0165 ..

A rough idea of how to generate the model

Sample dendrograms D with probability proportional to their likelihood



✓Choose an internal node uniformly at random and consider one of the two ways to reshuffle

✓Always accept the transition if it increases the likelihood else accept with some probability

How to Evaluate the Prediction (other)

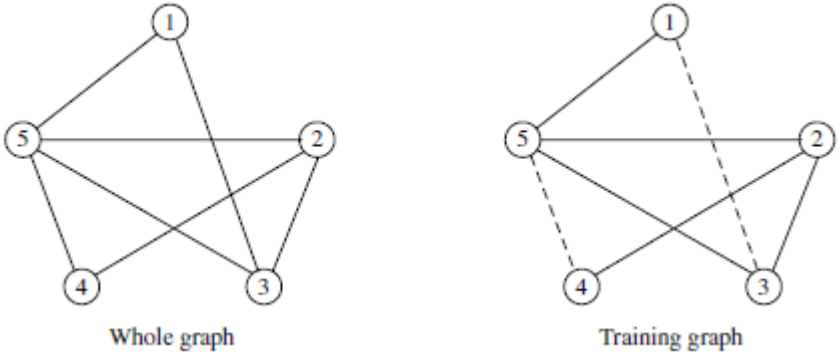
An undirected network G(V, E)
Predict Missing links (links not in E)
To test, randomly divide E into a training set E^T and a probe (test) set E^P
Apply standard techniques (k-fold cross validation)

Each time we randomly pick a missing link and a nonexistent link to compare their scores
If among n independent comparisons, there are n′ times the missing link having a higher score and n′′ times they have the same score, the AUC value is

$$\mathrm{AUC} = \frac{n' + 0.5n''}{n}$$

- the probability that a randomly chosen missing link is given a higher score than a randomly chosen nonexistent link
- If all the scores are generated from an independent and identical distribution, the AUC value should be about 0.5.

How to Evaluate the Prediction (other)



Algorithm assigns scores of all non-observed links as
s12 = 0.4, s13 = 0.5, s14 = 0.6, s34 = 0.5 and s45 = 0.6.

To calculate AUC, compare the scores of a probe (missing) link and a nonexistent link.
(n=) 6 pairs: s13 > s12, s13 < s14, s13 = s34, s45 > s12, s45 = s14, s45 > s34.
AUC = $(3 \times 1 + 2 \times 0.5)/6 \approx 0.67$.