Machine Learning with Graphs (MLG)

Link Prediction

Recommend Links between Nodes on Graphs

Cheng-Te Li (李政德)

Institute of Data Science

National Cheng Kung University

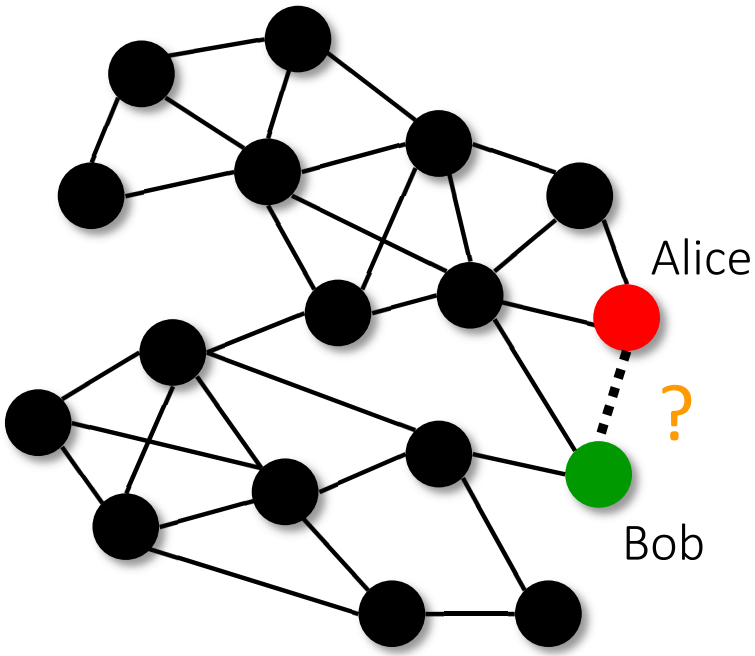
chengte@mail.ncku.edu.tw

Link Prediction

- Who is most likely to be interacted with a given individual?

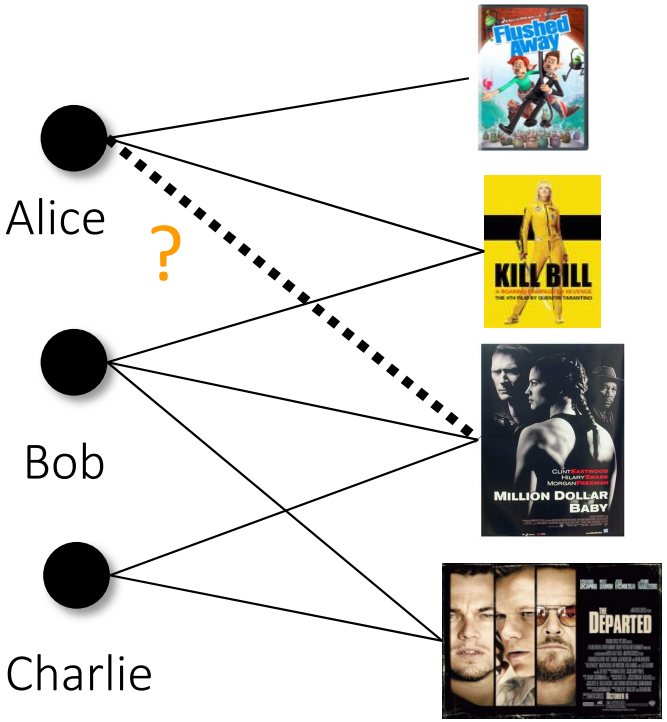
Friend suggestion in Facebook

Should Facebook suggest **Alice** as a friend for **Bob**?



Movie recommendation in Netflix

Should Netflix suggest the **movie** to **Alice**?



Link Prediction in Industry (Recommender Systems)

Proposing friendships

Proposing matches

Proposing items to purchase

Link Prediction: Who to Follow

Link Prediction against Criminals

PLOS ONE

RESEARCH ARTICLE

Link Prediction in Criminal Networks: A Tool for Criminal Intelligence Analysis

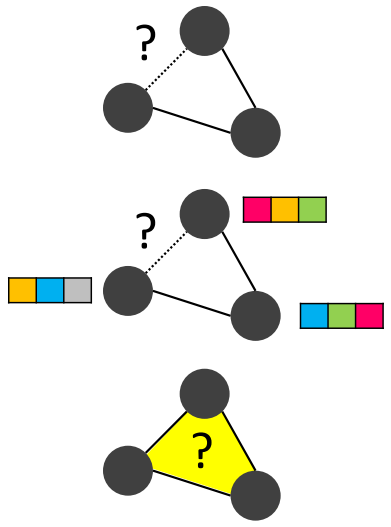
Possible missing links predicted between suspects in organized crime

predicted link (*x*, *y*)	node *x*	node *y*
(n49, n27)	boss' son and important drug dealer	important drug dealer
(n49, n48)	boss' son and important drug dealer	drug wholesaler
(n50, n160)	n49's brother and drug dealer	fugitive and broker
(n118, n36)	n45's wife	drug dealer
(n13, n43)	drug dealer	drug dealer
(n40, n53)	drug retailer	drug retailer
(n40, n147)	drug retailer	drug retailer
(n53, n147)	drug retailer	drug retailer
(n19, n40)	n48's boss and important drug dealer	drug retailer
(n19, n53)	n48's boss and important drug dealer	drug retailer
(n19, n147)	n48's boss and important drug dealer	drug retailer
(n24, n48)	n19's assistant	drug wholesaler
(n24, n147)	n19's assistant	drug retailer
(n28, n26)	n27's younger brother	n27's assistant and drug wholesaler
(n28, n140)	n27's younger brother	n27's assistant and drug wholesaler
(n26, n140)	n27's assistant and drug wholesaler	n27's assistant and drug wholesaler
(n5, n39)	'recruiter' and drug dealer	drug wholesaler

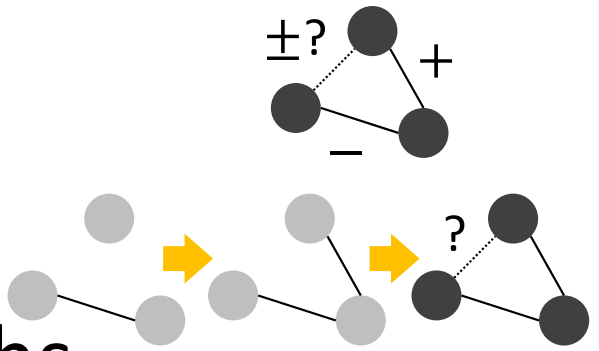
Let's Dive into Link Prediction

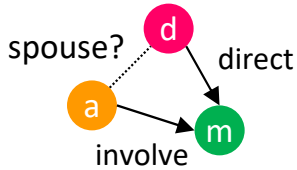
Building Recommender Systems from Link Prediction

1) Link prediction on **simple** graphs
2) Link prediction on **attributed** graphs
3) **High-order** link prediction



4) Link prediction on **signed** graphs
5) Link prediction on **dynamic** graphs
6) Link prediction on **knowledge** graphs





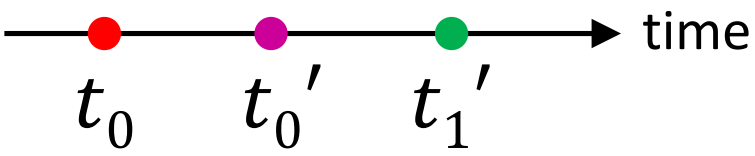
References

- David Liben-Nowell, **Jon Kleinberg**.
  "The Link Prediction Problem for Social Networks."
  ACM CIKM 2003.
  4900 cites
  - https://www.cs.cornell.edu/home/kleinber/link-pred.pdf
- L. Lu and T. Zhou.
  "Link Prediction in Complex Networks: A Survey."
  Physica A: Statistical Mechanics and its Applications, 390(6), 1150-1170, 2011.
  1933 cites
  - http://arxiv.org/pdf/1010.0725v1.pdf

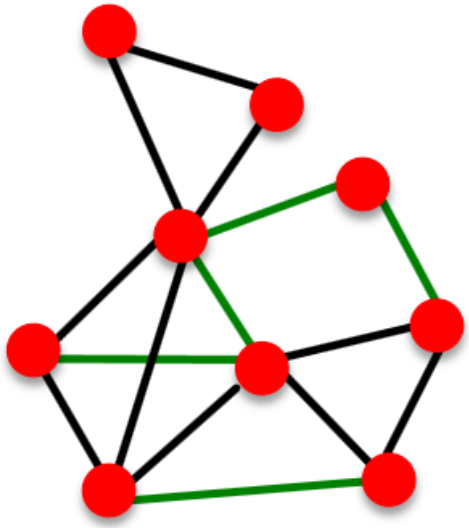
How to predict whether two persons will become friends in the future?

- Strategy 1: Rule-based Knowledge Decision
  - E.g. if common friends > 100,
    consider such two users as friends
  - Cons-1: Need domain experts
  - Cons-2: Could miss patterns that were unknown
- Strategy 2: Data-driven Approach
  - Data mining & machine learning techniques
  - Given a social network, extract features and
    learn the patterns that affect how edges are created
  - The goal is to predict the existence of **future** edges
    - Binary classification (0/1)

Link Prediction in Networks



- Problem Statement
  - **Given:** G[t_0,t_0'] a graph on edges up to t_0'
  - **Output:** A ranked list L of links (not in G[t_0,t_0']) that are predicted to appear in future G[t_0',t_1'] (t_1'>t_0'>t_0) for $(V \times V) \setminus E$ node pairs
- Evaluation
  - $n = |E_{new}|$: the number of edges that appear during the test period [t_0',t_1']
  - Take top-n elements of L and count the correctly-predicted edges



G[t_0, t'_0]

G[t_0',t'_1]

How To: The Setup

- For a given graph
- Split the data into a training set, and a test set
- Choose a link prediction algorithm
- Run the algorithm on the training set, and test it on the test set
- Check the accuracy
- Compare other link prediction algorithms

How To: The Setup

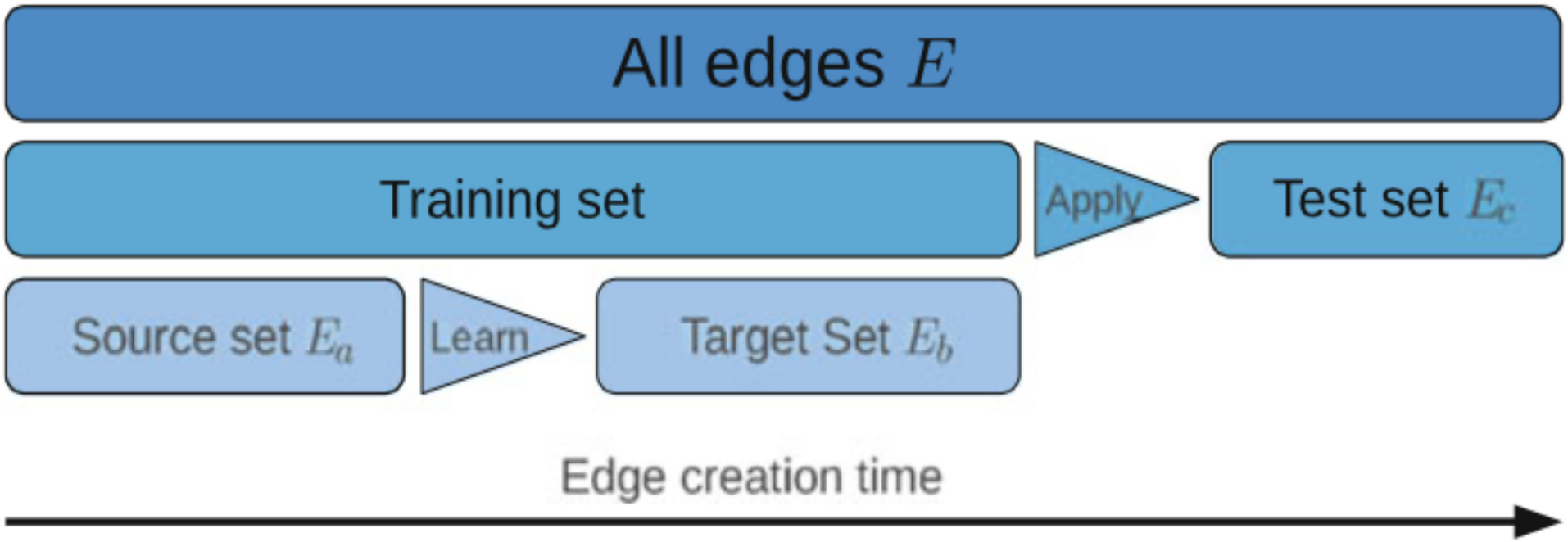
Split edges into a training set and a test set

$$E = E_{train} \cup E_{test}$$

Set of all possible edges in the graph

$$|U| = \frac{|V|(|V|-1)}{2}$$

How To: The Setup



How To: Output of Algorithm

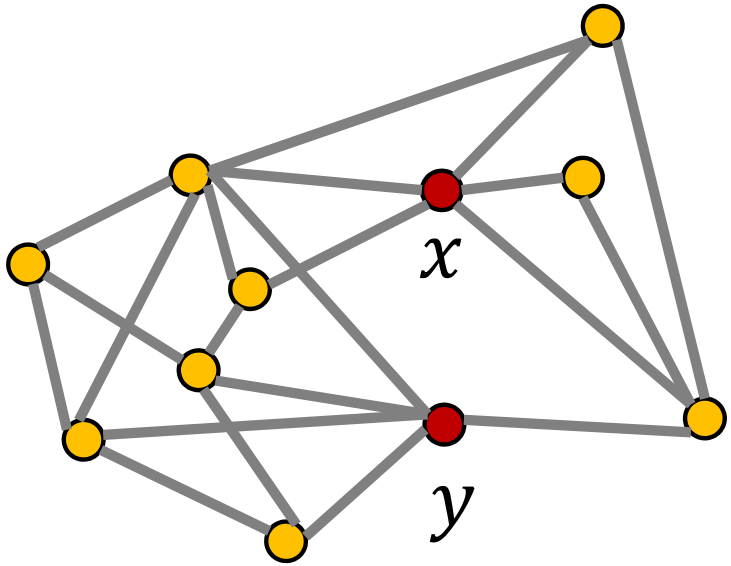
The Link Prediction algorithm will spit out a list, ranked with edges which are most likely to appear at the top, descending.

$$L : e_L \in U - E_{train}$$

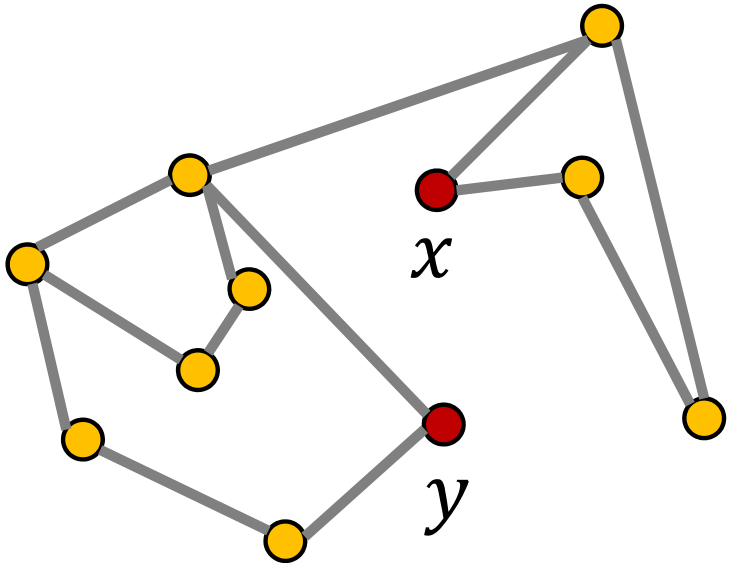
Taking the first n links from the list, and calculating the intersection with the probe set of length n, gives a simple measure of accuracy

Link Prediction: Intuition

- In many networks, people who are "close" belong to the same social circles and will inevitably encounter one another and become linked themselves
- Link prediction heuristics measure how "close" people are



Red nodes are close to each other



Red nodes are more distant

Approach Overview

- Step 1: Determine what is considered to be the **instance** for classification (what is x? what is y?)
  - Node or link?
  - Multi-class or single class?
- Step 2: Obtain features for each instance
  - Topological features
    - e.g. degree, centrality scores, clustering coefficient
  - Attribute features
    - e.g. common interests, similar background, co-locations
  - Social features
    - e.g. common friends, number of LIKE, SHARE, and RESPONSE

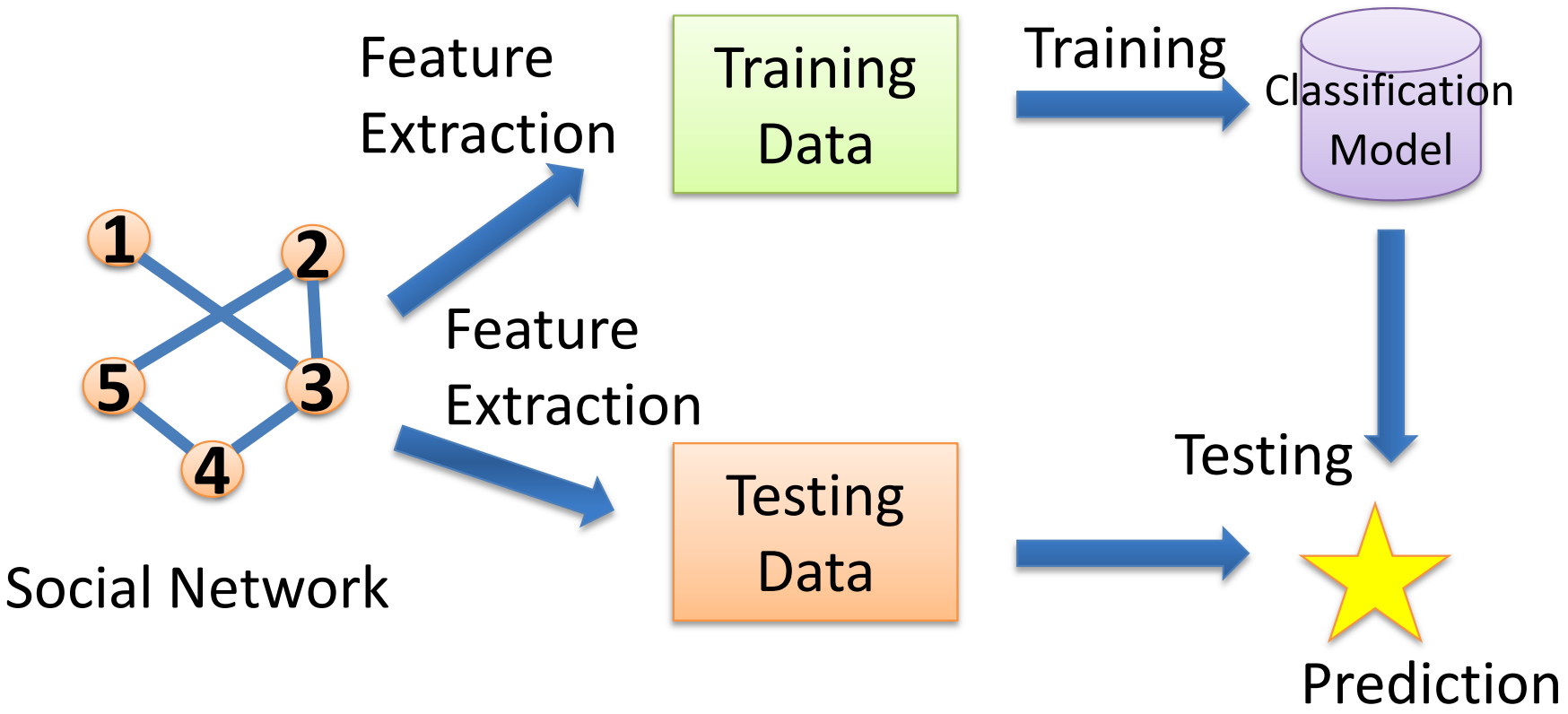
→ [**Unsupervised**] Use feature values to rank node pairs, those ranked at top positions are considered as predicted links

Approach Overview

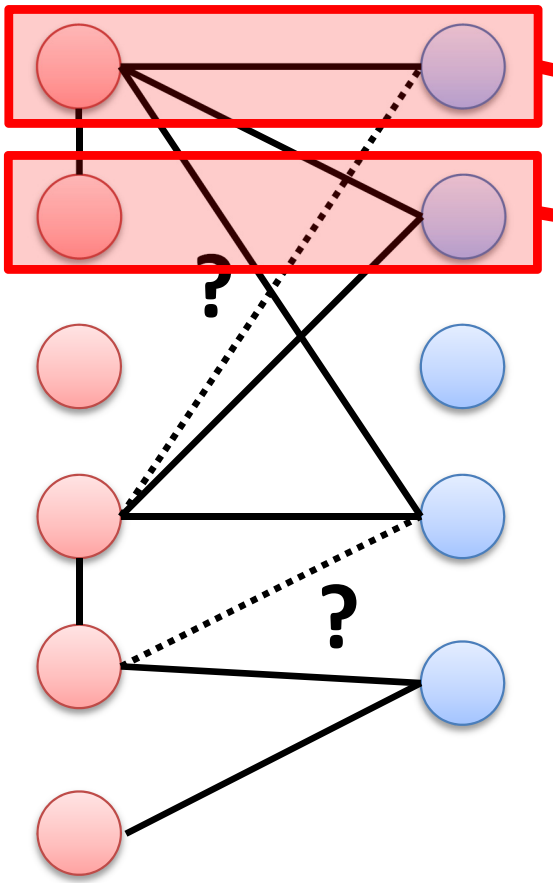
Supervised Approach:

- Step 3: Determine a classifier / predictor to use
  - E.g. SVM, Regression, Decision Tree, Neural Network
- Step 4: Train a classifier and evaluate the results using held-out / testing data
  - Choose a reasonable evaluation metric
  - If the performance is not satisfying, go back to Step 2 and Step 3

Approach Overview



Illustration

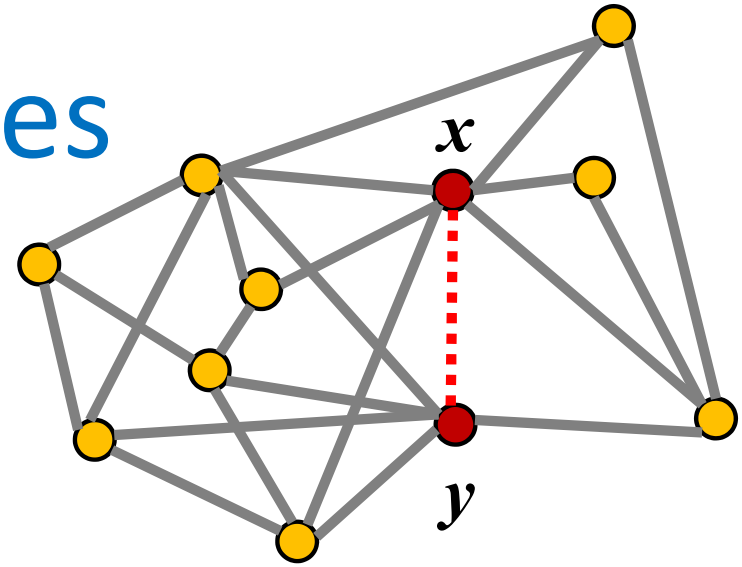


Label	Features
1	0.34, 0.96, -0.50
0	0.90, 0.10, 0.30
1	0.28, 0.77, -0.60
0	1.00, 0.20, 0.40
…	

Feature Extraction: Notation

- G: $graph$
- $\Gamma(x)$: $the\ set\ of\ node\ x's\ neighbors$
- $|\Gamma(x)|$: $the\ degree\ of\ node\ x$
- $Length(p)$: $length\ of\ path\ p$
- A: the adjacency matrix of G
- $score(x, y)$: $the\ feature\ score\ of\ node\ x\ and\ node\ y$

List of Features



- Common Neighbors
- Preferential Attachment
- Jaccard's Coefficient
- Adamic/Adar

Neighborhood-based

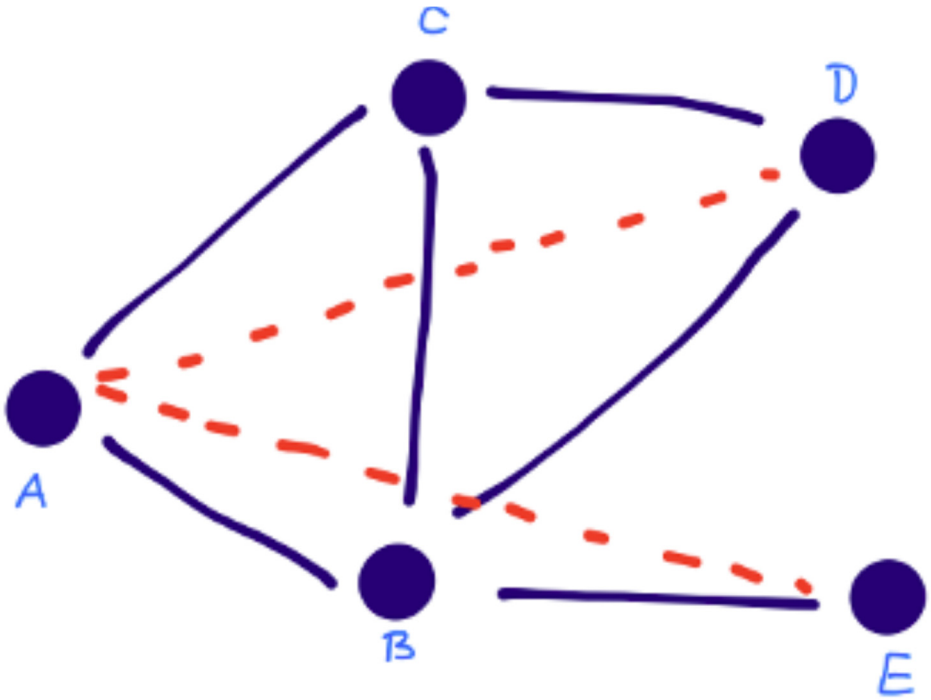
- Graph distance
- Katz_β Score
- Hitting Time / Commute Time
- Rooted Random Walk

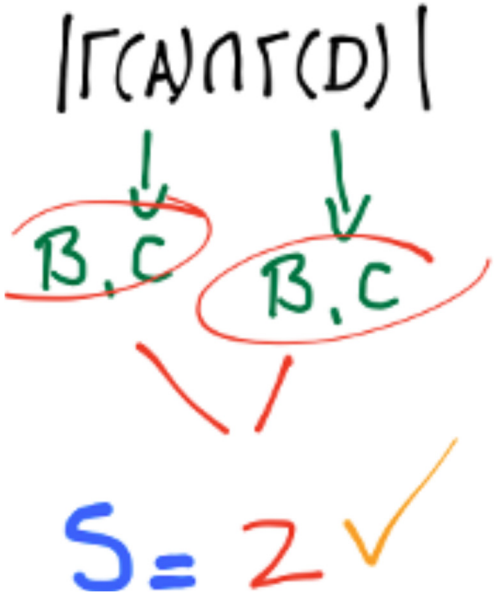
Path-based

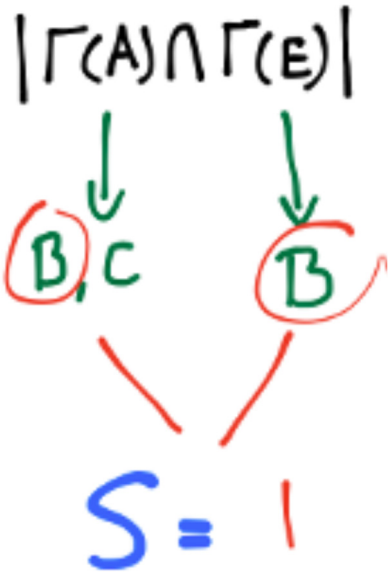
Common Neighbors

- Two persons who more common friends lead to higher potential that they become friends

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



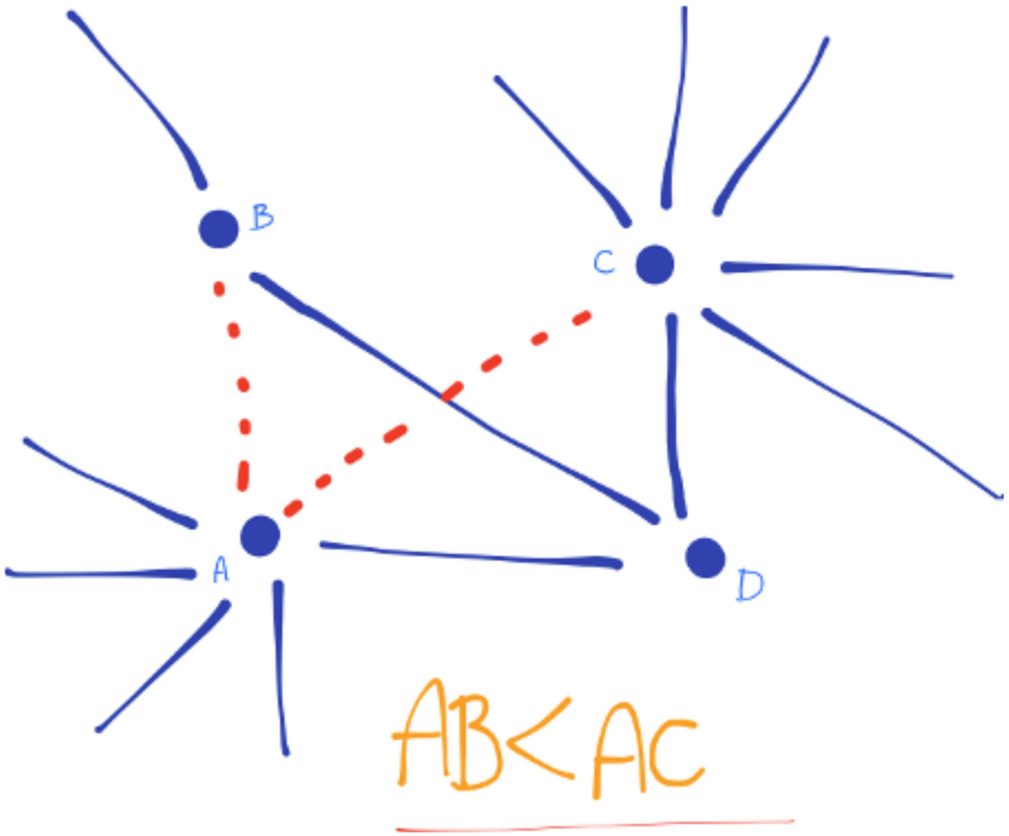




Preferential Attachment

- Those with more friends tend to make friends with people who also have many friends

$$score(x, y) = |\Gamma(x)| \times |\Gamma(y)|$$



Jaccard's Coefficient

- The probability that a common neighbor of a pair of vertices x and y would be selected if the selection is made randomly from the union of the neighbor-sets of x and y
  - i.e., weighting rarer neighbors more heavily

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

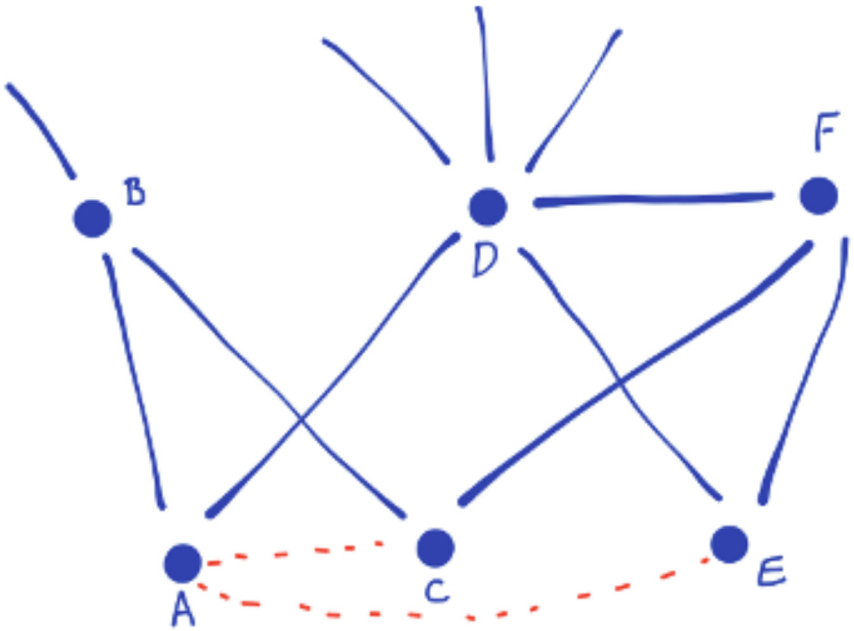
Adamic/Adar (AA)

Lada Adamic

Director of Facebook Social Network Analysis team

- Assign larger score to common neighbors z of x and y which themselves have fewer neighbors $|\Gamma(z)|$

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



$\Gamma(A) \cap \Gamma(C) = B$

$\frac{1}{\log(\Gamma(B))} = \frac{1}{\log 3} = 2.09$

$\Gamma(A) \cap \Gamma(E) = D$

$\frac{1}{\log(\Gamma(D))} = \frac{1}{\lg 6} = 1.2$

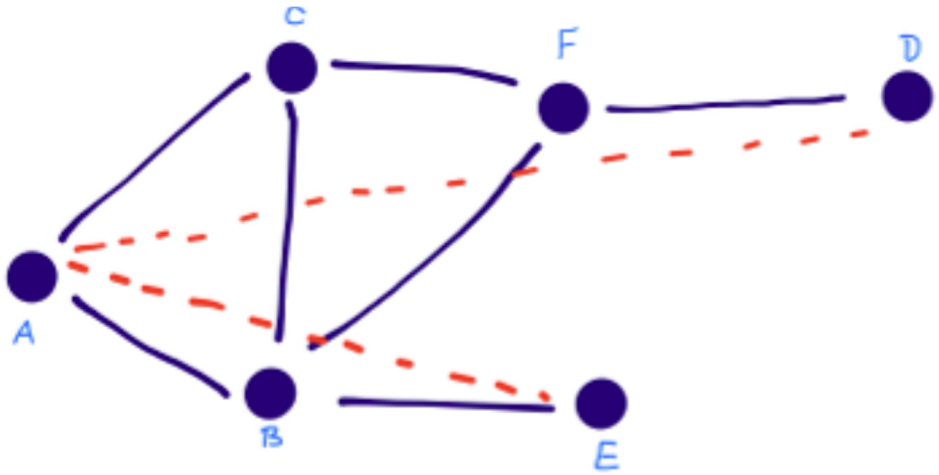
Other Neighborhood-based Methods

- Salton index: $score(x,y) = \frac{|\Gamma(x) \cap \Gamma(y)|}{\sqrt{|\Gamma(x)||\Gamma(y)|}}$
- Sørensen index: $score(x,y) = \frac{2|\Gamma(x) \cap \Gamma(y)|}{|\Gamma(x)|+|\Gamma(y)|}$
- Hub Promoted Index: $score(x,y) = \frac{|\Gamma(x) \cap \Gamma(y)|}{\min\{|\Gamma(x)|,|\Gamma(y)|\}}$
- Hub Depressed Index: $score(x,y) = \frac{|\Gamma(x) \cap \Gamma(y)|}{\max\{|\Gamma(x)|,|\Gamma(y)|\}}$
- Leicht-Holme-Newman Index: $score(x,y) = \frac{|\Gamma(x) \cap \Gamma(y)|}{|\Gamma(x)||\Gamma(y)|}$
- Resource allocation: $score(x,y) = \sum_{z \in |\Gamma(x) \cap \Gamma(y)|} \frac{1}{|\Gamma(z)|}$

Graph Distance

- If two persons close to each other in the social network, they tend to be acquainted

$$score(x, y) = -(Length\ of\ Shortest\ Path\ between\ x\ and\ y)$$



Score(A, E) = -2 ✓

Score(A, D) = -3

desc order

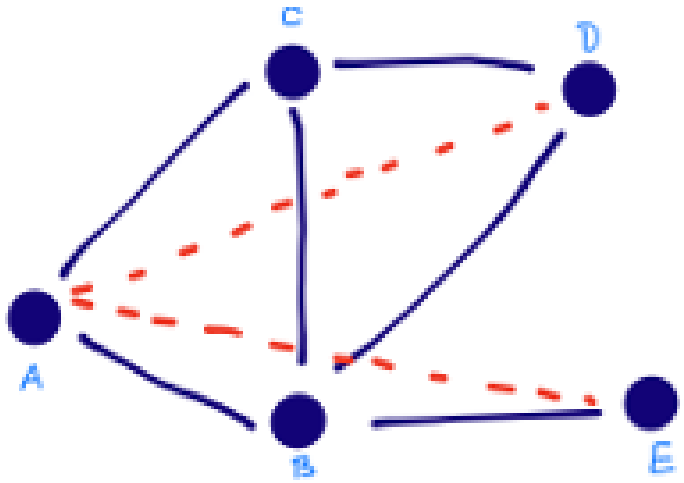
$katz_\beta$ Score

- Sum over all possible paths between x and y, giving higher score to shorter paths

- $katz_\beta(x,y) = \Sigma_{L=1}^{\infty} \beta^L |path(x,y)^L|$
  - $\beta (\leq 1)$: dampling factor

    Exponentially damped to count short paths more heavily
  - $path(x,y)^L$: the set of all length-L paths from x to y
  - Small β: predictions much like common neighbors
- How to calculate $katz_\beta(x,y)$?
  - The number of possible paths between x and y grows exponentially with the length L

$katz_\beta$ Score



# of Hops

$Path^2_{A,D} = 2$ $Path^3_{A,D} = 2$

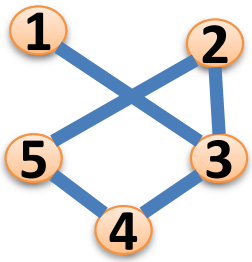
$S = \frac{1}{2} \cdot 2 + \frac{1}{4} \cdot 2 + \cdots$

Damping Factor

$Path^2_{A,E} = 1$ $Path^3_{A,E} = 1$

$S = \frac{1}{2} \cdot 1 + \frac{1}{4} \cdot 1 + \cdots$

Multiplication of adjacency matrix



A^2 is actually the number of all possible length-2 paths from a to b

A

	1	2	3	4	5
1	0	0	1	0	0
2	0	0	1	0	1
3	1	1	0	1	0
4	0	0	1	0	1
5	0	1	0	1	0

$\times$

A

	1	2	3	4	5
1	0	0	1	0	0
2	0	0	1	0	1
3	1	1	0	1	0
4	0	0	1	0	1
5	0	1	0	1	0

$=$

A^2

	1	2	3	4	5
1	1	1	0	1	0
2	1	0	0	2	0
3	0	0	3	0	2
4	1	2	0	2	0
5	0	0	2	0	2

4->3->2, 4->5->2

$katz_\beta$ Score

- $Katz\ score\ matrix$
$$= \Sigma_{L=1}^{\infty} \beta^L A^L$$
$$= I + \beta A + \beta^2 A^2 + \beta^3 A^3 \ldots - I$$
$$= (I - \beta A)^{-1} - I$$
- The katz scores between all pairs of nodes can be computed by finding: $(I - \beta A)^{-1} - I$
  - A is the adjacency matrix
  - I is an identity matrix

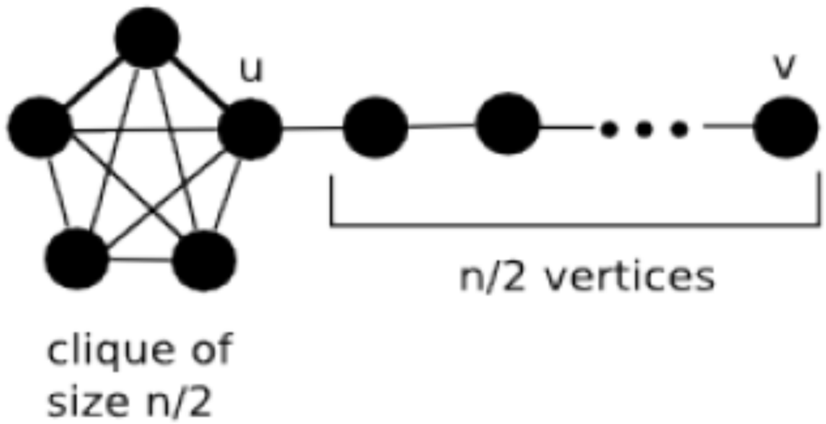
(Closed Form)

Hitting Time / Commute Time

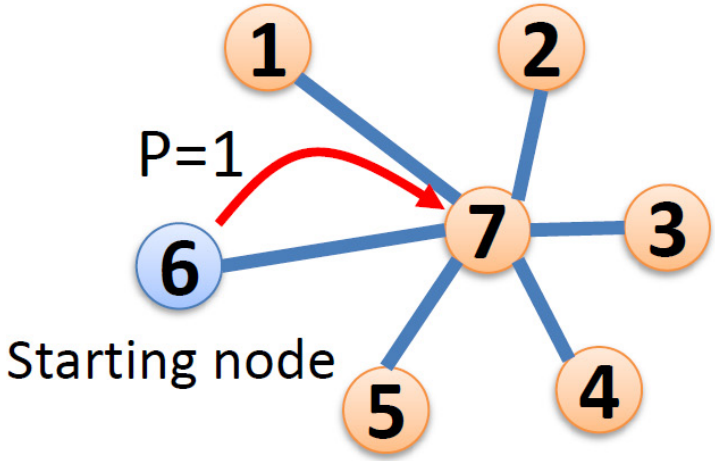
- $H_{x,y}$: the expected number of steps required for a random walk starting from node x to reach y
- For directed graphs, the hitting time from x to y is $H_{x,y}$
  For undirected graphs, we use commute time $H_{x,y} + H_{y,x}$

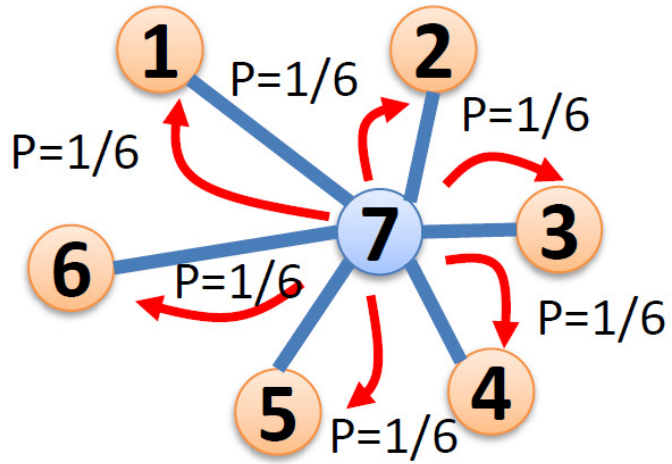
$$score(x,y) = \begin{cases} -\left(H_{x,y} + H_{y,x}\right), \text{for undirected graph} \\ -H_{x,y} \qquad\qquad\quad , \text{for directed graph} \end{cases}$$

- Easy to compute by performing trial random walks



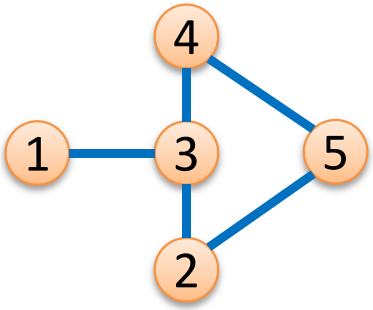
Random Walk





Transition matrix P

	1	2	3	4	5	6	7
1	0	0	0	0	0	0	1
2	0	0	0	0	0	0	1
3	0	0	0	0	0	0	1
4	0	0	0	0	0	0	1
5	0	0	0	0	0	0	1
6	0	0	0	0	0	0	1
7	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	0



Random Walk

- Let D be a diagonal degree matrix with $D[i,j] = \sum_j A[i,j]$. Let $N = D^{-1}A$ denote the adjacency matrix with row sums normalized to 1.

	1	2	3	4	5
1	0	0	1	0	0
2	0	0	1/2	0	1/2
3	1/3	1/3	0	1/3	0
4	0	0	1/2	0	1/2
5	0	1/2	0	1/2	0

N

$\times$

	1	2	3	4	5
1	0	0	1	0	0
2	0	0	1/2	0	1/2
3	1/3	1/3	0	1/3	0
4	0	0	1/2	0	1/2
5	0	1/2	0	1/2	0

N

$\boldsymbol{N^2[x,y]}$ is the probability starting from x to y for a 2-step random walk

Rooted Random Walk

- Due to the power-law degree distribution of social network, some nodes may have very high stationary probability in a random walk
- $score(x, y) =$ the **stationary probability** from x to y under Random Walk with Restart (RWR)
  - If starting from x, each step of random walk has probability $1 - \alpha$ to return to x, and has probability α to move to a random neighbor
  - Stationary probability: the stable probability distribution after running many (infinite) random walk steps

Random Walk with Restart (RWR)

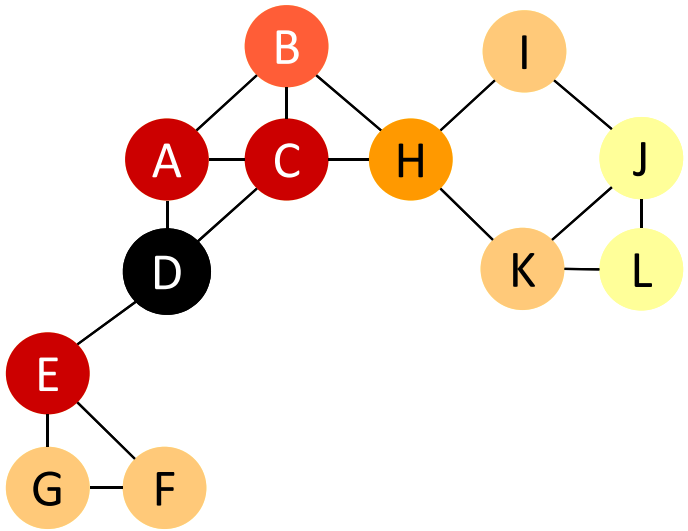
$$\underset{\text{nx1}}{R} = \alpha \underset{\text{nxn}}{\tilde{W}} \underset{\text{nx1}}{R} + (1-\alpha)\underset{\text{nx1}}{E}$$

Score Vector

Adjacency Matrix

Fly-out Probability

Starting Vector



$$\begin{pmatrix} 0.18 \\ 0.18 \\ 0.12 \\ 0.25 \\ 0.13 \\ 0.07 \\ 0.07 \\ 0.08 \\ 0.02 \\ 0.01 \\ 0.02 \\ 0.01 \end{pmatrix} = 0.9 \times \begin{pmatrix} 0 & 1/3 & 1/3 & 1/3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1/3 & 0 & 1/3 & 0 & 0 & 0 & 0 & 1/4 & 0 & 0 & 0 & 0 \\ 1/3 & 1/3 & 0 & 1/3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1/3 & 0 & 1/3 & 0 & 1/4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/3 & 0 & 1/2 & 1/2 & 1/4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/4 & 0 & 1/2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/4 & 1/2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1/3 & 0 & 0 & 1/4 & 0 & 0 & 0 & 1/2 & 0 & 1/3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1/4 & 0 & 1/3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1/2 & 0 & 1/3 & 1/2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1/4 & 0 & 1/3 & 0 & 1/2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1/3 & 1/3 & 0 \end{pmatrix} \begin{pmatrix} 0.13 \\ 0.10 \\ 0.13 \\ 0.22 \\ 0.13 \\ 0.05 \\ 0.05 \\ 0.08 \\ 0.04 \\ 0.03 \\ 0.04 \\ 0.02 \end{pmatrix} + 0.1 \times \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Closed Form of RWR

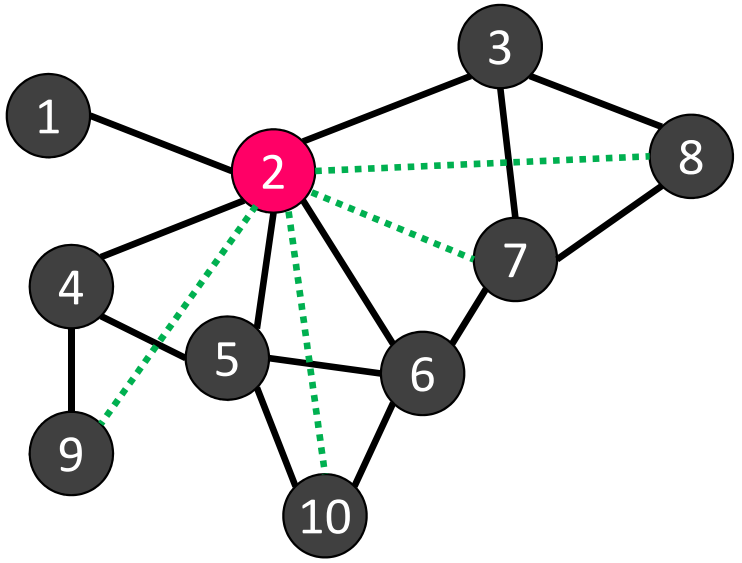
$$s = (1 - \alpha)(I - \alpha W)^{-1} e_x$$

where s is a similarity vector between x and all the other nodes in the graph,

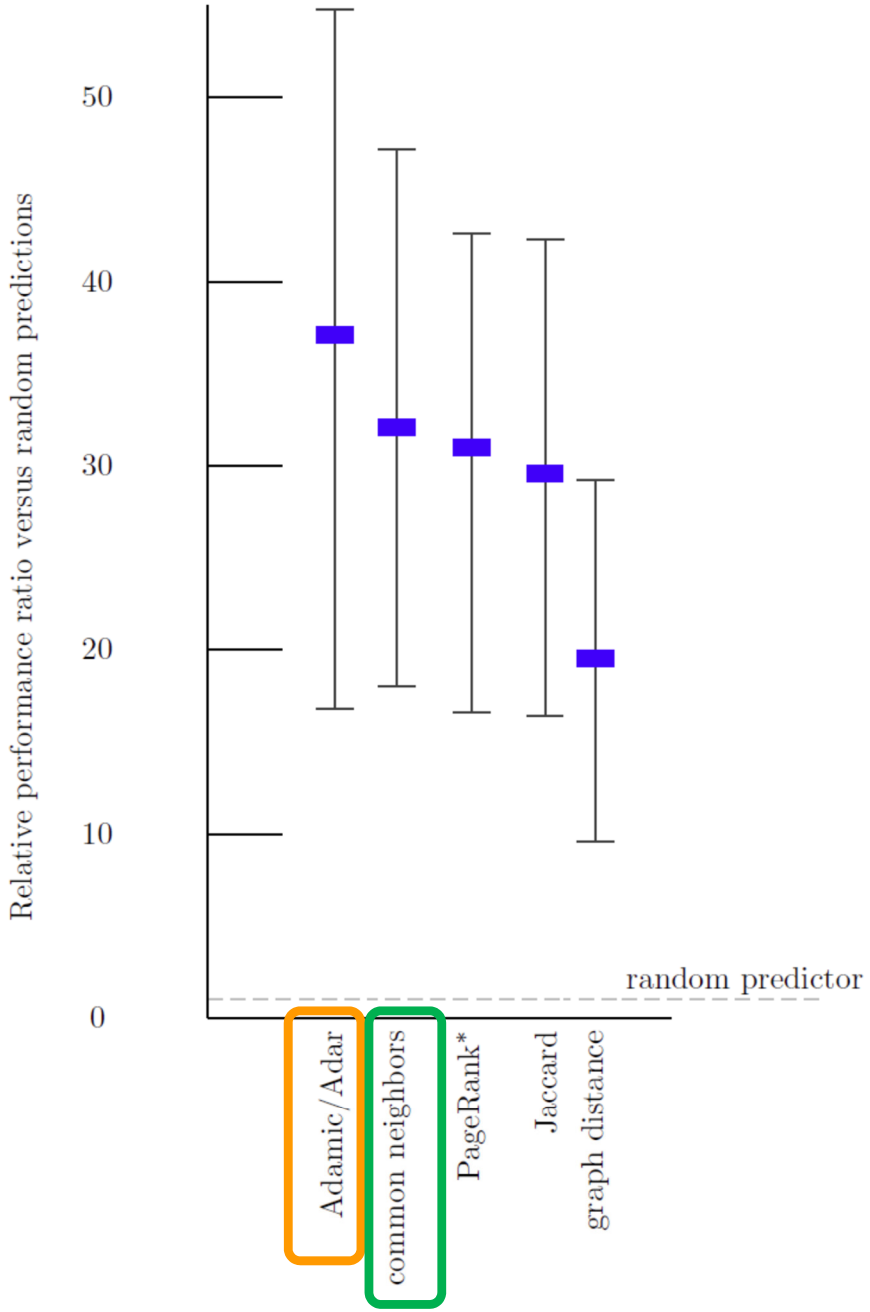
e_x: is the vector that has all 0, but a 1 in position x

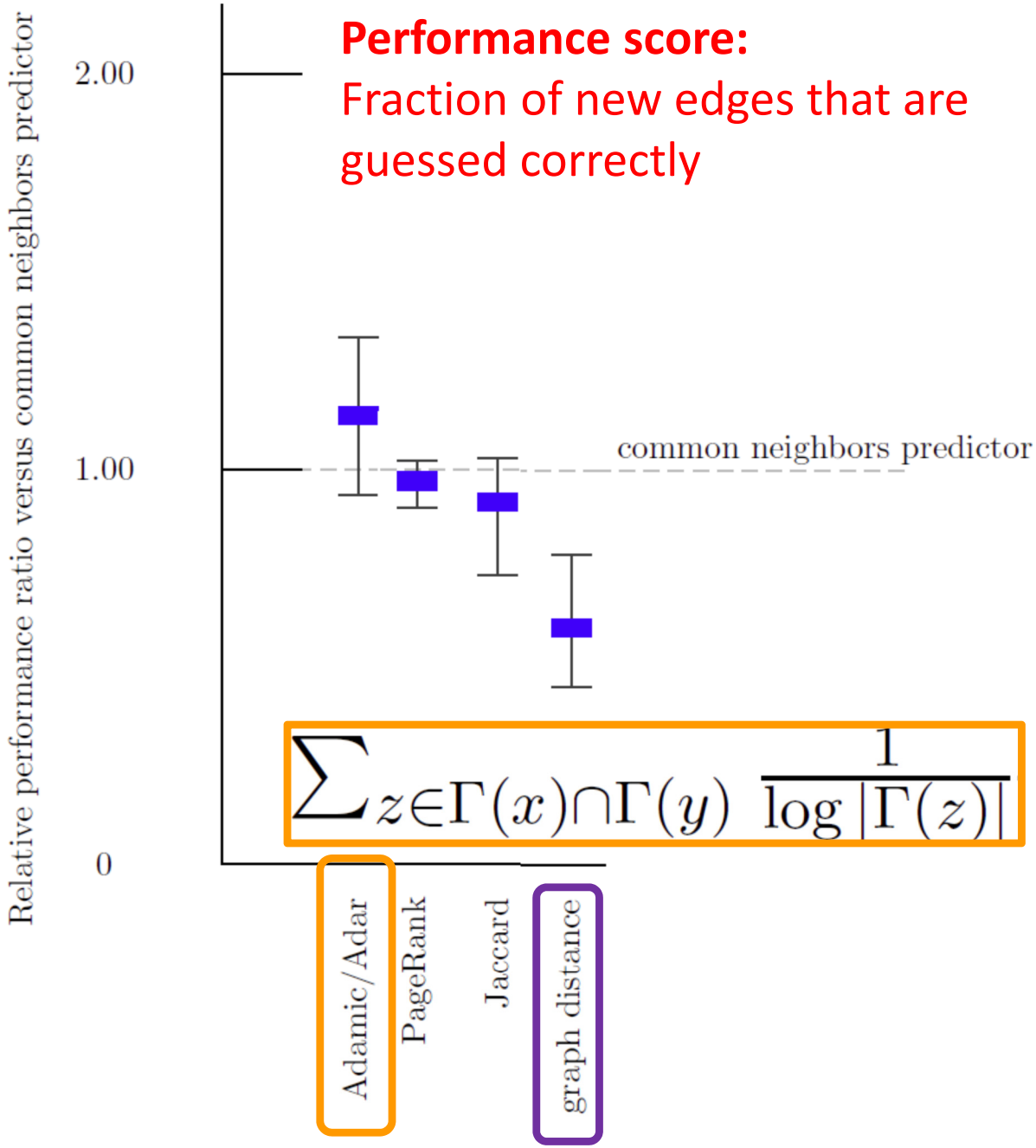
Unsupervised Approach

1) Given the input social network
2) For each pair of nodes (x, y) without links
3) Compute the score(x,y) between x an y
4) Generate a ranking list node pairs in a descending order based on score(x,y)
5) Perform the evaluation



Results: Improvement





Performance score:
Fraction of new edges that are guessed correctly

Supervised Approach

- You can choose any state-of-the-art supervised learning methods to be the classification model for link prediction
  - Naive Bayes
  - Support Vector Machine
  - Decision Tree
  - Random Forest
  - Logistic Regression
  - Neural Network

Metrics for Performance Evaluation

Confusion Matrix: contains the number of

- True positive: correctly predicted links that are actually links
- True negative: number of correctly predicted non-links
- False positive: number of predicted links that are not links
- False negative: number of non-predicted links that are actually links

		PREDICTED CLASS	
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	TP	FN
	Class=No	FP	TN

Precision = TP/(TP+FP)

Metrics for Performance Evaluation

- AUC (Area Under ROC Curve)

Show the performance of a binary classifier

TPR (sensitivity)=TP/(TP+FN) (how many data points are correctly classified among those that are actually positive)

FPR = FP/(TN+FP) (percentage of negative classified as positive)

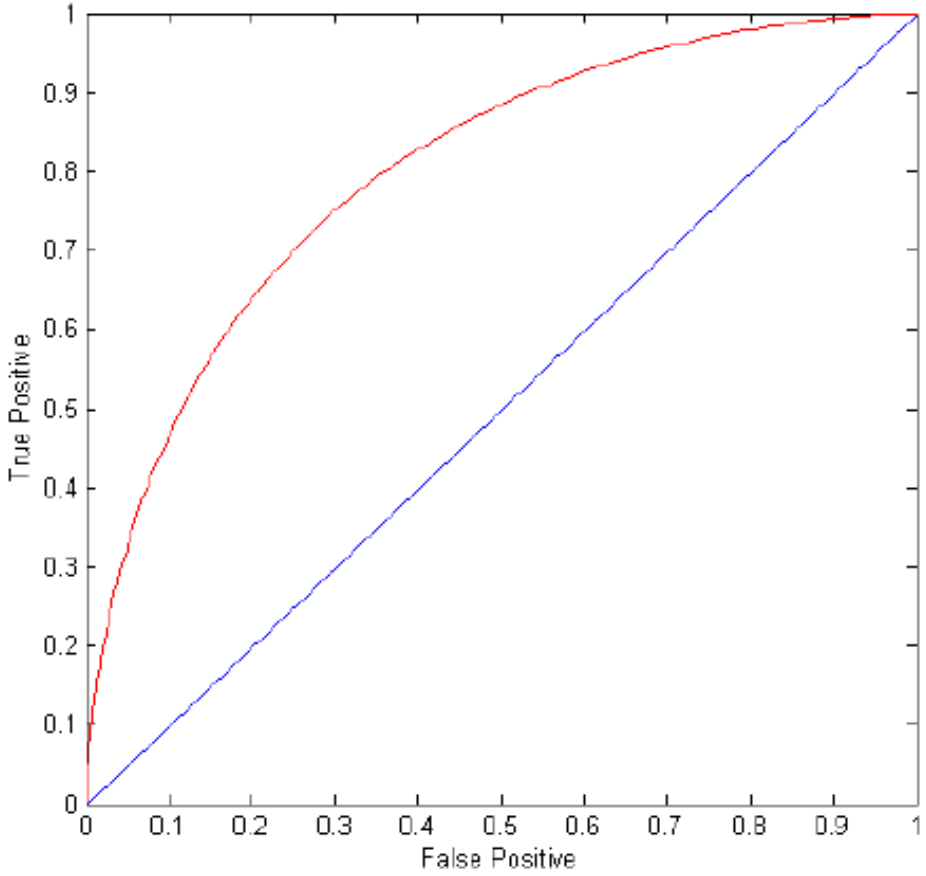
(0,0): declare everything to be negative class

(1,1): declare everything to be positive class

(0,1): ideal

Diagonal line: Random guessing

Below diagonal line: prediction is worse than random



https://en.wikipedia.org/wiki/Receiver_operating_characteristic

Challenges of Link Prediction

- Class Skewness, i.e., Label Imbalance
  - Edges are much less than pairs of nodes
  - Too many negative instances (too few positive instances)
- Solutions
  - Determine the proportion of position/negative instances through sampling
  - Cost-sensitive Classification
  - Use graph distance to filter negative ones
- Model Calibration: determine the decision threshold
- Time Complexity Issue