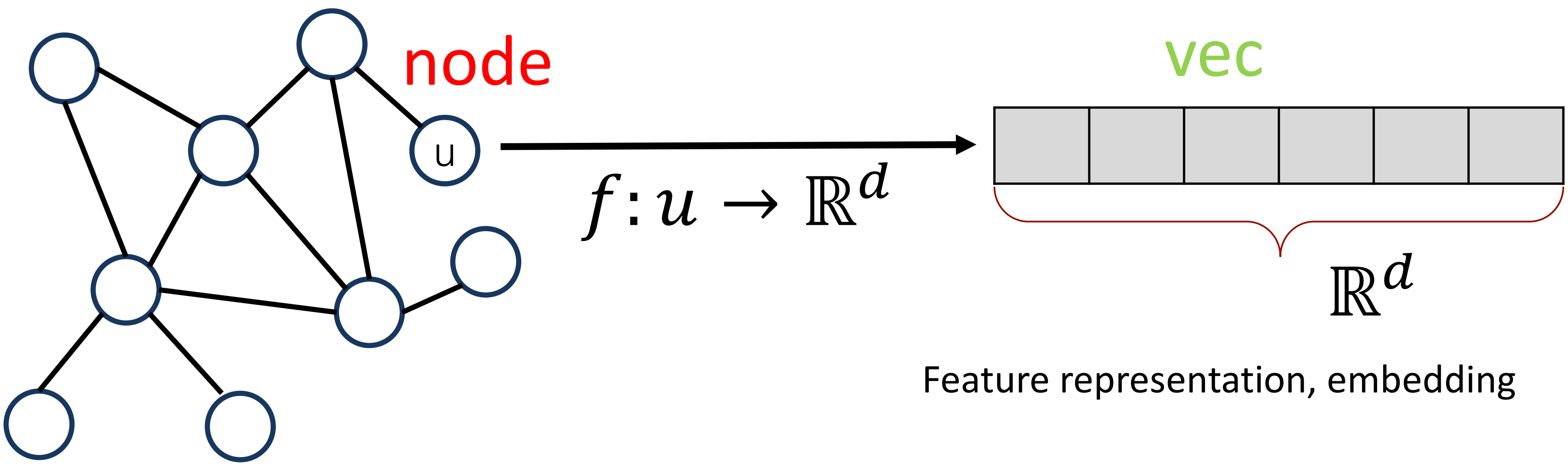
Online Social Networks and Media

Link Prediction, Classification,

Graph Embeddings

Graph embeddings: what are they?

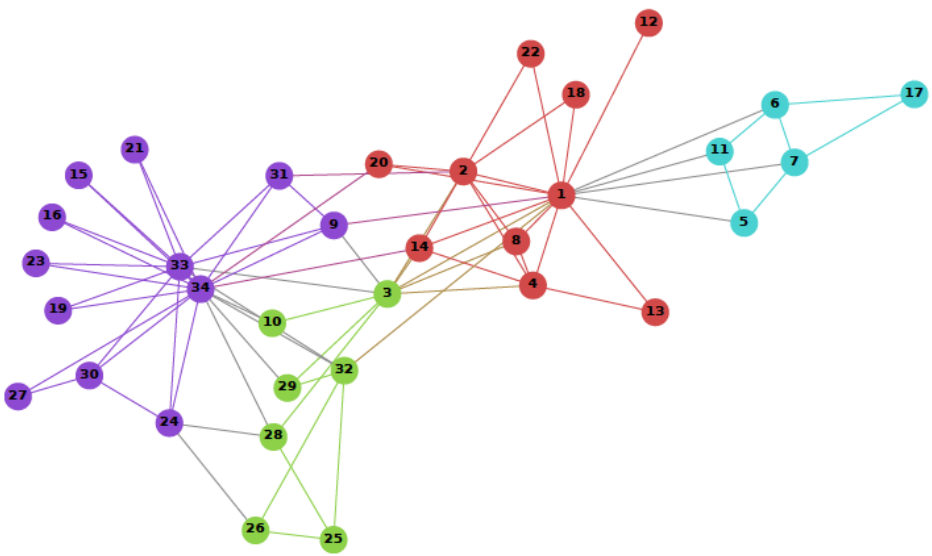


Map nodes to *d*-dimensional vectors so that:
"similar" nodes in the graph have embeddings that *are close together*.

Example

Zachary's Karate Club Network:

Input



Output

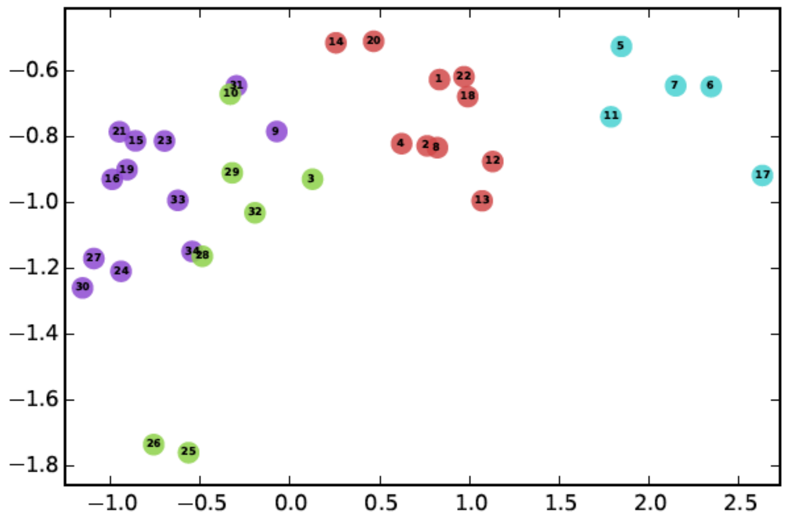
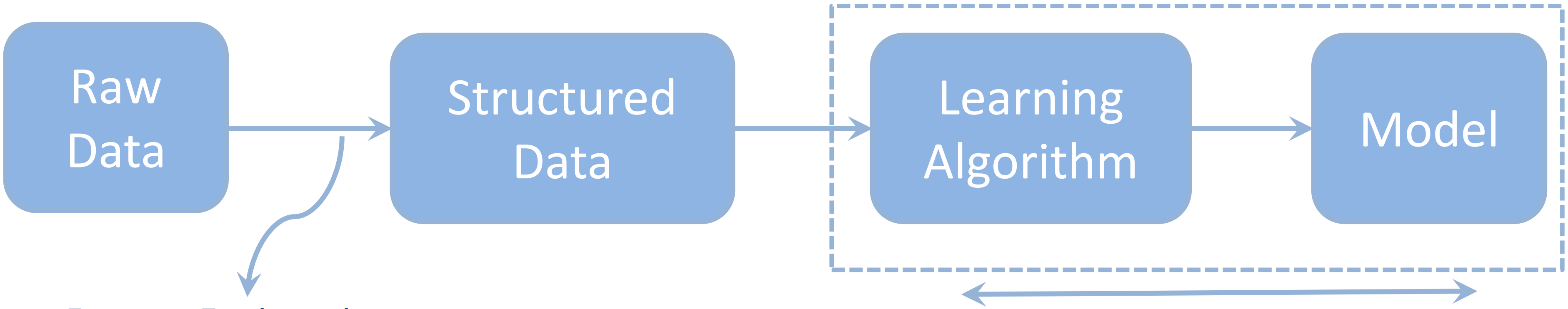


Image from: Perozzi et al.. DeepWalk: Online Learning of Social Representations. KDD 2014.

Graph embeddings: why?

Machine learning lifecycle



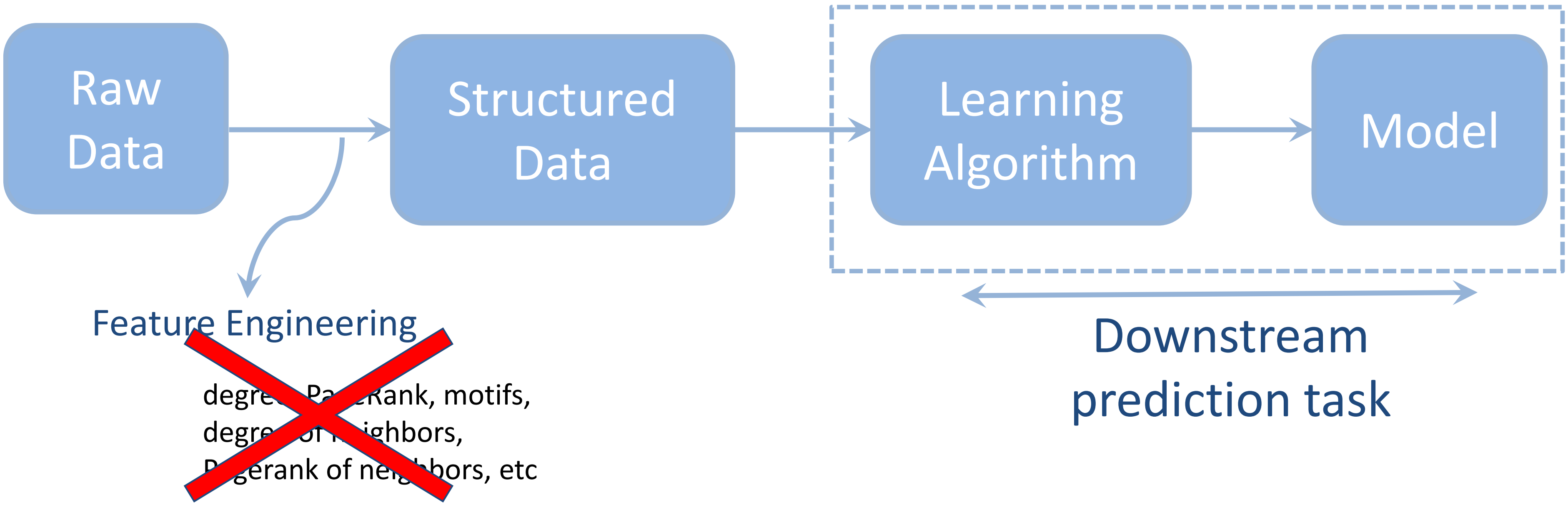
Feature Engineering

degree, PageRank, motifs,
degrees of neighbors,
Pagerank of neighbors, etc

Downstream
prediction task

Graph embeddings: why?

Machine learning lifecycle



Automatically learn the features (embeddings)

Machine Learning tasks in networks

- Link prediction and link recommendations
- Node labeling
- Community detection *(we have already seen an approach)*
- Network similarity

Link Prediction

Motivation

- Recommending *new friends* in online social networks, suggesting *interactions* or *collaborations,* predicting *hidden connections* (e.g., terrorist),

In social networks:

- Increases user engagement
- Controls the growth of the network

Outline

- Estimating a score for each edge (seminal work of Liben-Nowell&Kleinberg)
- Classification approach
- The who to follow service at Twitter (one more application of link analysis)

Problem Definition

Link prediction problem: Given the links in a social network at time *t* (G_{old}), ***predict*** the edges that will be added to the network during the time interval from time *t* to a given future time *t'* (G_{new}).

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

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

    To save experimental effort in the laboratory or in the field

Approach

- Assign a *connection weight score(x, y)* to each pair of nodes <x, y> based on the input graph
- Produce a ranked list of edges in decreasing order of score
- Recommend the ones with the highest score

Note

- We can consider all links incident to *a specific node x*, and recommend to *x* the top ones
- If we focus to a specific *x*, the score can be seen as a centrality measure for x

How to define the score

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

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

Two general methods

- Neighbors
- Paths

Neighborhood-based metrics

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

Common neighbors:

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

A adjacency matrix

$A^2_{x,y}$: number of *different paths of length 2*

Jaccard coefficient:

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

The probability that both x and y have a feature from a randomly selected feature that either x or y has

Neighborhood-based metrics

Adamic Adar:

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

- Weighted version: common neighbors which themselves have few neighbors get larger weights (larger weights to rare features)

- Neighbors who are linked with **2** nodes are assigned weight = 1/log(2)
- Neighbors who are linked with **5** nodes are assigned weight = 1/log(5)

Note: |N(x)| = degree of x, inverse logarithmic centrality

Neighborhood-based metrics

Preferential attachment:

$$score(x, y) = |N(x)||N(y)|$$

- Nodes like to form ties with 'popular' nodes
  - E.g., empirical evidence suggest that co-authorship is correlated with the product of the neighborhood sizes
  - Fall-back strategy: recommending popular users
- Depends ***on the degrees*** of the nodes not on their neighbors per se

Path-based methods

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

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

Katz$_\beta$ measure:

$$\sum_{l=1}^{\infty} \beta^l |path^l_{<x,y>}|$$

- Sum over all paths of length *l*
- *0 < β < 1* parameter of the predictor, exponentially damped to count short paths more heavily

Path-based methods

Katz$_\beta$ measure:

$$\sum_{l=1}^{\infty} \beta^l |path^l_{<x,y>}| = \beta \mathrm{A}_{xy} + \beta^2 A^2_{xy} + \beta^3 A^3_{xy} + \ldots$$

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

- $0 < \beta < 1$
  - Small β much like common neighbors
  - β small: degree, β maximal: eigenvalue
- Weighted version

Path-based methods

Based on ***random walks*** that starts at *x*

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

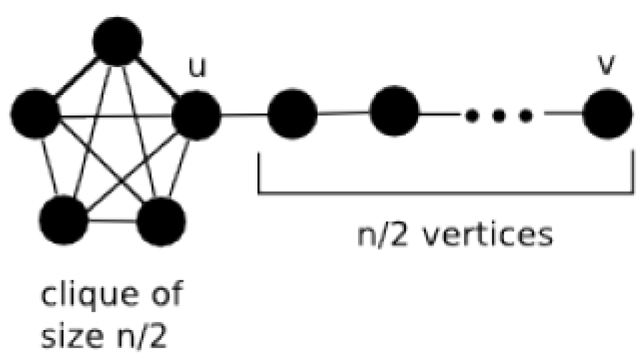
$$\text{score}(x, y) = -H_{x,y}$$

Commute Time $C_{x,y}$ from *x* to *y:* the expected number of steps to travel from *x* to *y* *and* from *y* to *x*

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

Not symmetric, can be shown

$$h_{vu} = \Theta(n^2)$$
$$h_{uv} = \Theta(n^3)$$



Path-based methods

Example: hit time $h_{1,n}$ in a line

Stationary-normed versions:

to counteract the fact that $H_{x,y}$ is rather small when y is a node with a large stationary probability regardless of *x*

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

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

Personalized (or, Rooted) PageRank: with probability $(1 – a)$ moves to a random neighbor and with probability *a* returns to *x*

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

SimRank

Two objects are *similar*, if they are *related to similar objects*

x and *y* are *similar*, if they are related to objects *w* and *z* respectively and *w* and *z* are themselves similar

$$similarity(x, y) = C \; \frac{\sum_{w \in \mathrm{N}(x)} \sum_{z \in N(y)} similarity(w, z)}{|N(x)||N(y)|}$$

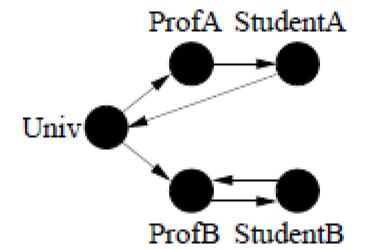
Base case: similarity(x, x) = 1

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

SimRank

Introduced for directed graphs: two objects are similar if referenced by similar objects

$$s(a,b) = \frac{C}{|I(a)||I(b)|} \sum_{i=1}^{|I(a)|} \sum_{j=1}^{|I(b)|} s(I_i(\alpha), \mathrm{I}_j(b))$$



Average similarity between in-neighbors of *a* and in-neighbors of *b*

I(x): in-neighbors of x, *C:* constant between 0 and 1, *decay*

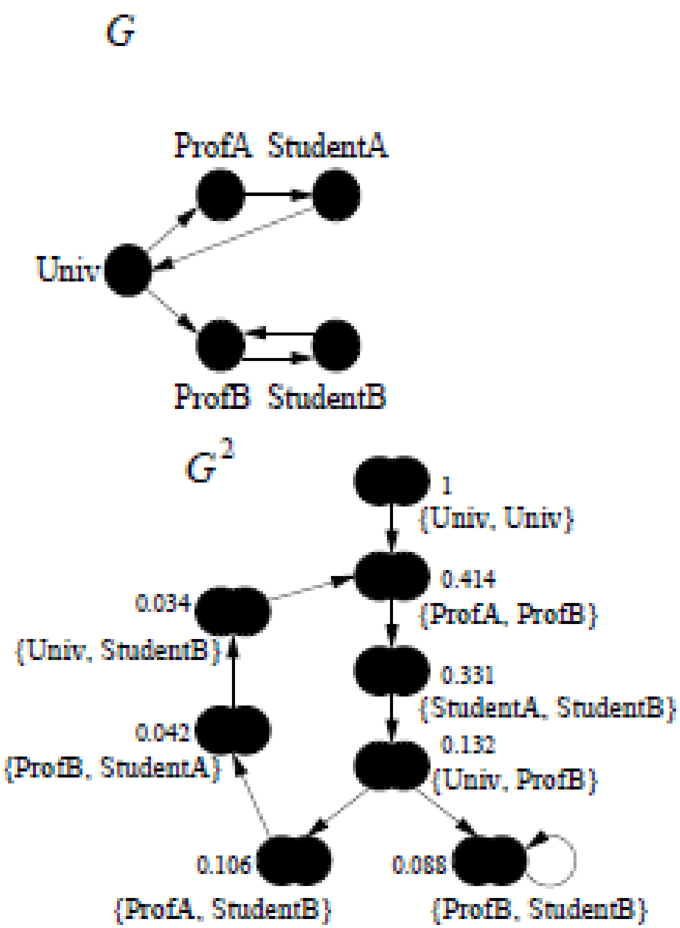
s(a, b) = 1, if a =b

Iterative computation

$s_0(x, y) = 1$ if x = y and 0 otherwise

s_{k+1} based on the s_k values of its (in-neighbors) computed at iteration k

SimRank: the Pair Graph



Pair graph G^2

A node for each pair of nodes

An edge (x, y) → (a, b), if x → a <u>and</u> y → b

a value per node: similarity of the corresponding pairs

Computation starts at singleton nodes (score = 1)

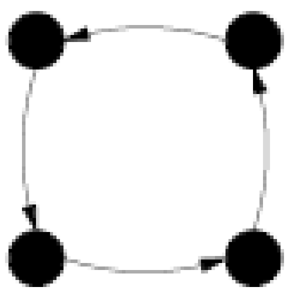
Scores *flow* from a node to its neighbors

C gives the rate of *decay* as similarity flows across edges (*C* = 0.8 in the example)

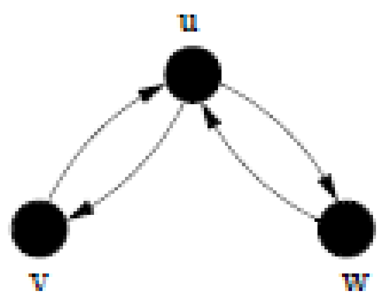
- Symmetric pairs: (a, b) node same as (b, a) node (with the union of associated edges)
- Omit singleton that do not contribute to score (no {ProfA, ProfA} node) and nodes with 0 score {ProfA, StudentA})
- Self-loops and cycles reinforce similarity
- Prune: by considering only nodes within a radius

SimRank and random walks

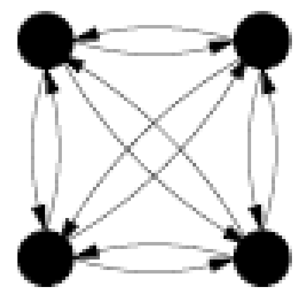
Expected Meeting Distance (EMD) *m (a, b)* between *a* and *b*: the expected number of steps required before two random surfers, one starting at *a* and the other starting at *b*, would meet if they walked the graph randomly at lock-step



= ∞



m(u, v) = m(u,w) = ∞,
m(v, w) = 1
v and w are much more similar than u is to v or w.



= 3,
a lower similarity than between v and w but higher than between u and v (or u and w).

SimRank and random walks

Let us consider G^2
A node (*a*, *b*) as a state of the tour in G:
if *a* moves to *c*, *b* moves to *d* in G,
then (*a*, *b*) moves to (*c*, *d*) in G^2

A tour in G^2 of length n represents a pair of tours in G where each has length n

What are the states in G^2 that correspond to "meeting" points in G?
What is the meeting point of a and b? m(a, b)?

SimRank and random walks

What are the states in G^2 that correspond to “meeting” points in G?

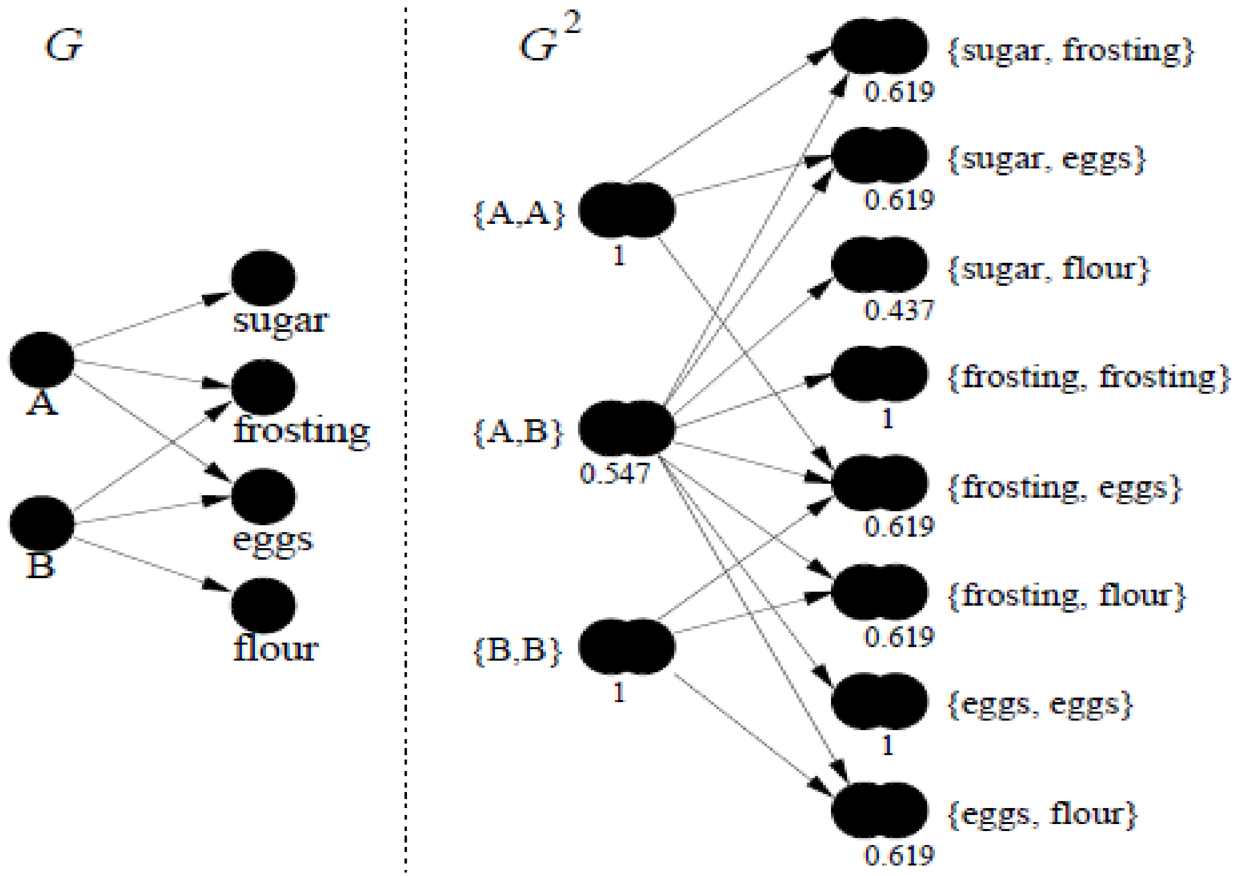
Singleton nodes (common neighbors)

The EMD m(*a*, *b*) is just the expected distance - hitting time in G^2 between (*a*, *b*) and any singleton node

- The sum is taken over all walks that start from (*a*, *b*) and end at a singleton node

This roughly corresponds to the SimRank of (a, b): when two surfers one from a and one from b that randomly walk the graph would meet

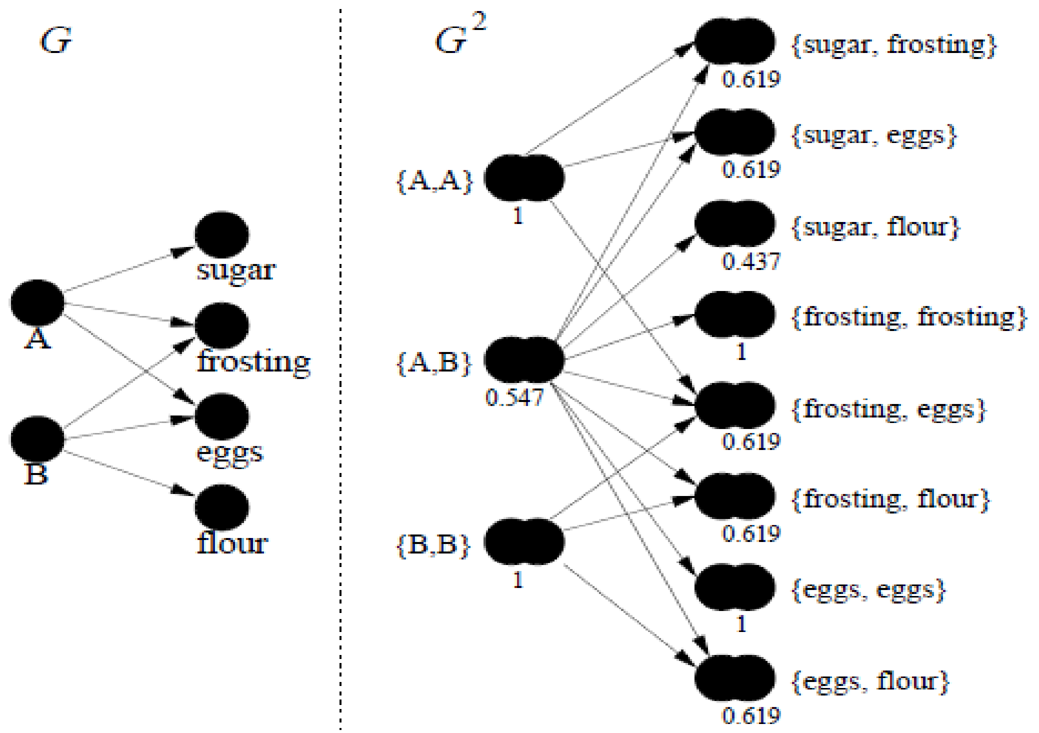
SimRank for bipartite graphs



- People are *similar* if they purchase *similar* items.
- Items are *similar* if they are purchased by *similar* people

Useful for recommendations in general

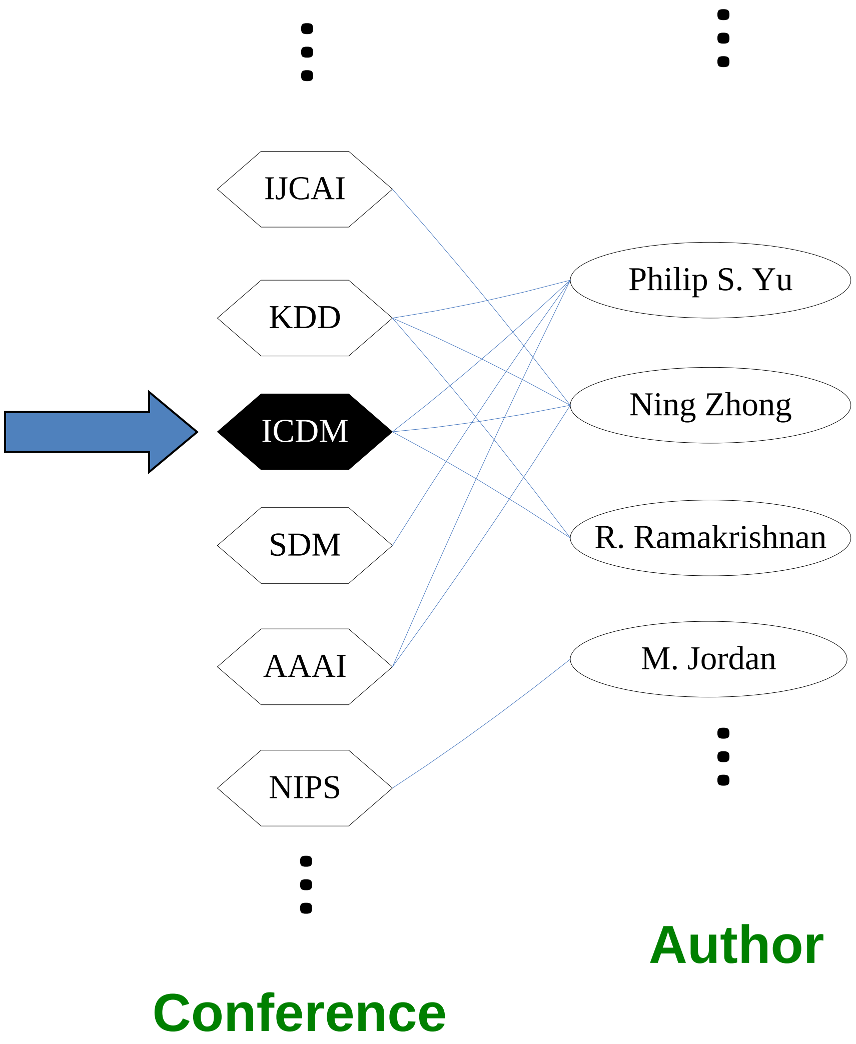
SimRank for bipartite graphs



$$s(A,B) = \frac{C_1}{|O(A)||O(B)|} \sum_{i=1}^{|O(A)|} \sum_{j=1}^{|O(B)|} s(O_i(A), O_j(B))$$

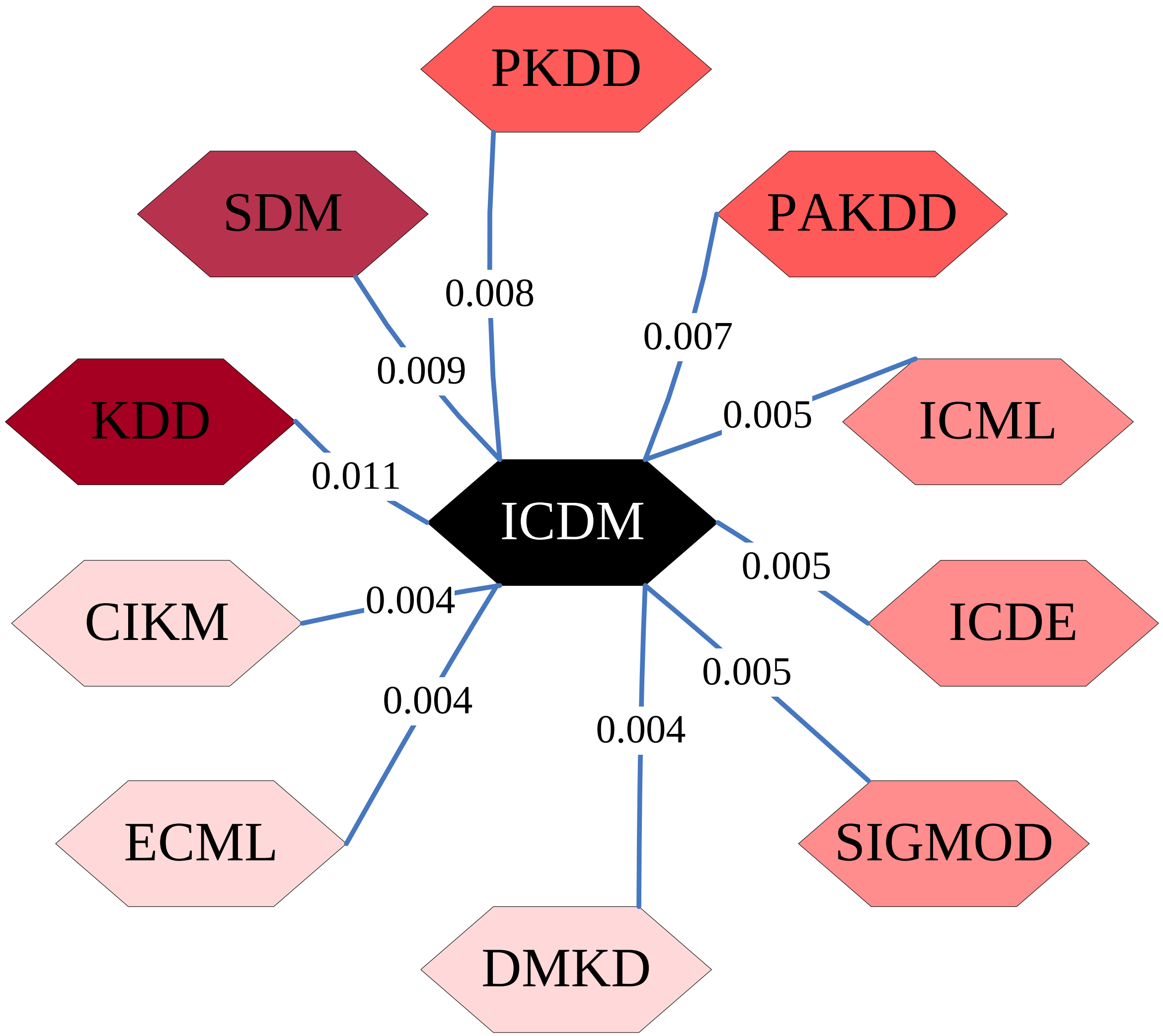
$$s(c,d) = \frac{C_2}{|I(c)||I(d)|} \sum_{i=1}^{|I(c)|} \sum_{j=1}^{|I(d)|} s(I_i(c), I_j(d))$$

SimRank



Q: What is most related conference to ICDM?

SimRank


PKDD
SDM
PAKDD
0.008
0.007
0.009
KDD
ICML
0.005
0.011
ICDM
0.005
CIKM
ICDE
0.004
0.004
0.005
0.004
ECML
SIGMOD
DMKD

Evaluation of link recommendations

Output

a list L_p of pairs in $V \times V - E_{old}$ ranked by score

predicted new links in decreasing order of confidence

Precision at recall

- How many of the top-*n* predictions are correct where *n* = $|E_{new}|$

Improvement over baseline

Baseline: random predictor

Probability that a random prediction is correct: $$\frac{|E_{new}|}{\binom{|V|}{2} - |E_{old}|}$$

Possible correct

Possible predictions

Can we combine the various scores?

How?

Classification (supervised learning)

Classification

Using Supervised Learning

Given a collection of records (*training set*)

Each record contains
a set of *attributes (features) +* the *class attribute*.

Find a *model* for the class attribute as a function of the values of other attributes.

Goal: previously unseen records should be assigned a class as accurately as possible.

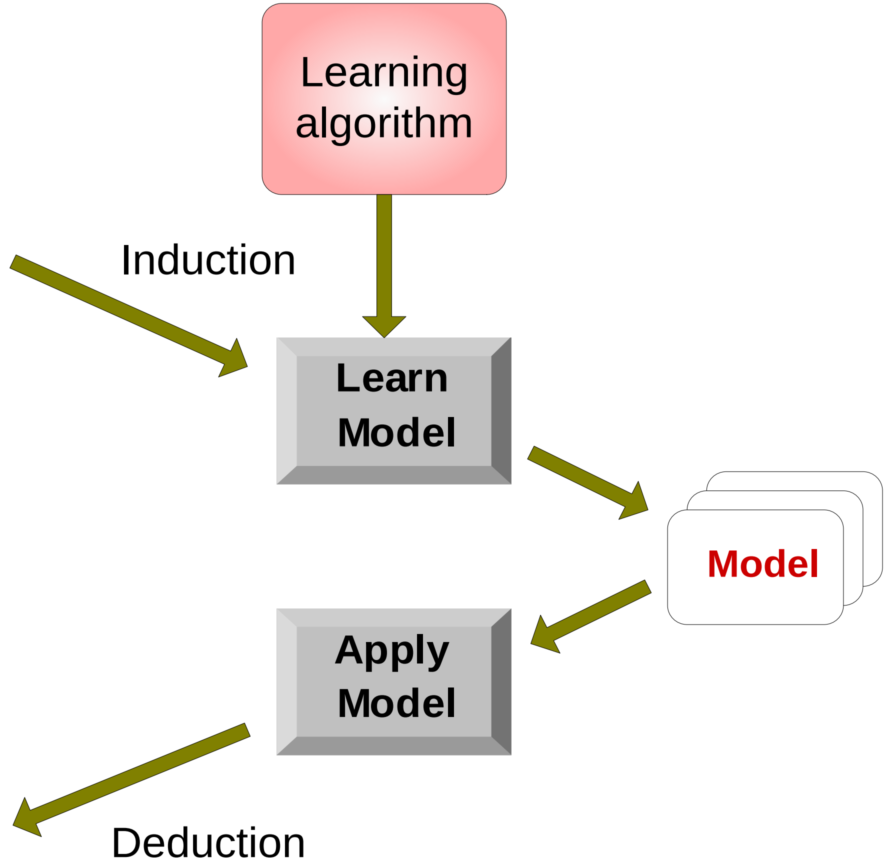
A test set is used to determine the accuracy of the model.

Usually, the given data set is divided into training and test sets, with training set used to build the model and test set used to validate it.

Illustrating the Classification Task

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set



Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set

Classification Techniques

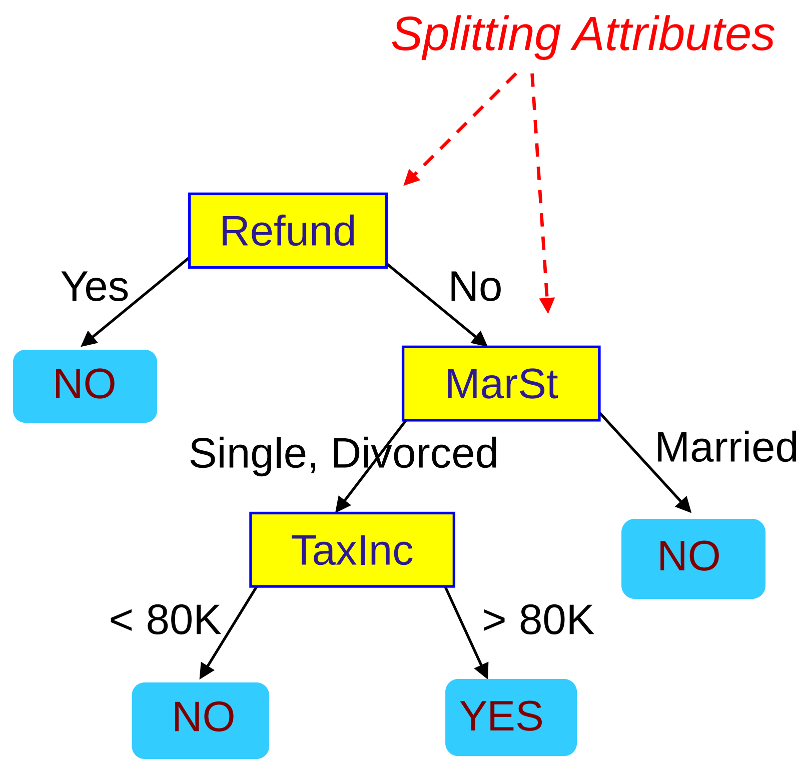
- Decision Tree based methods
- Rule-based methods
- Memory based reasoning
- Neural networks (more soon)
- Naïve Bayes and Bayesian Belief networks
- Support vector machines
- Logistic regression

Example of a Decision Tree

categorical categorical continuous class

Tid	**Refund**	**Marital Status**	**Taxable Income**	**Cheat**
1	Yes	Single	125K	**No**
2	No	Married	100K	**No**
3	No	Single	70K	**No**
4	Yes	Married	120K	**No**
5	No	Divorced	95K	**Yes**
6	No	Married	60K	**No**
7	Yes	Divorced	220K	**No**
8	No	Single	85K	**Yes**
9	No	Married	75K	**No**
10	No	Single	90K	**Yes**

Training Data



Model: Decision Tree

Classification for Link Prediction

Input

Features describing the two nodes

Output

Prediction

Metrics for Performance Evaluation

Confusion Matrix:

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

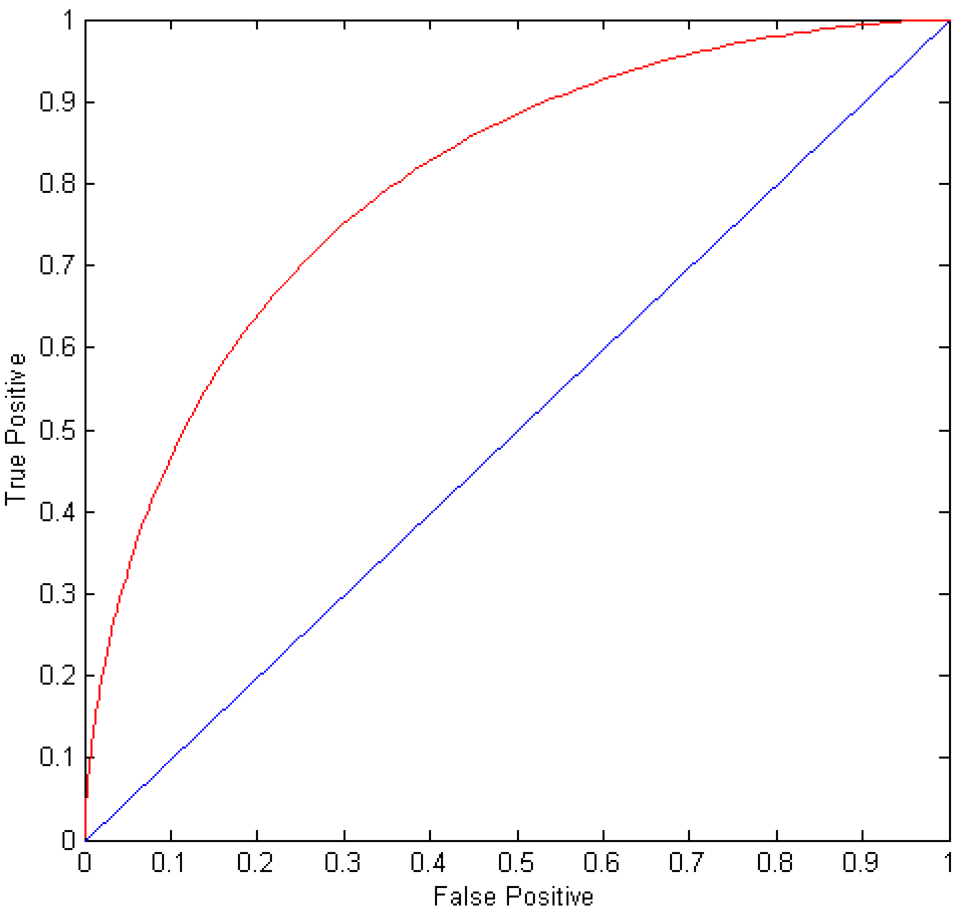
$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

ROC Curve

TPR (sensitivity)=TP/(TP+FN) (percentage of positive classified as 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 opposite of the true class

AUC: area under the ROC

Classification for Link Prediction: features

For each edge (i, j)

Name	Parameters	HPLP	HPLP+
In-Degree(i)	-	✓	✓
In-Volume(i)	-	✓	✓
In-Degree(j)	-	✓	✓
In-Volume(j)	-	✓	✓
Out-Degree(i)	-	✓	✓
Out-Volume(i)	-	✓	✓
Out-Degree(j)	-	✓	✓
Out-Volume(j)	-	✓	✓
Common Nbrs(i,j)	-	✓	✓
Max. Flow(i,j)	$l = 5$	✓	✓
Shortest Paths(i,j)	$l = 5$	✓	✓
PropFlow(i,j)	$l = 5$	✓	✓
Adamic/Adar(i,j)	-		✓
Jaccard's Coef(i,j)	-		✓
Katz(i,j)	$l = 5$, $\beta = 0.005$		✓
Pref Attach(i,j)	-		✓

PropFlow: corresponds to the probability that a restricted random walk starting at *x* ends at *y* in *l* steps or fewer using link weights as transition probabilities (stops in *l* steps or if revisits a node)

How to construct the training set

When to extract features and when to determine class?

Two time instances τ_x and τ_y

- From t_0 to τ_x construct graph and extract features (G_{old})
- From $\tau_x + 1$ to τ_y examine if a link appears (determine class value)

What are good values

- Large τ_x better topological features (as the network reaches saturation)
- Large τ_y larger number of positives (size of positive class)
- Should also match the real-world prediction interval

How to construct the training set

Unsupervised (single feature)

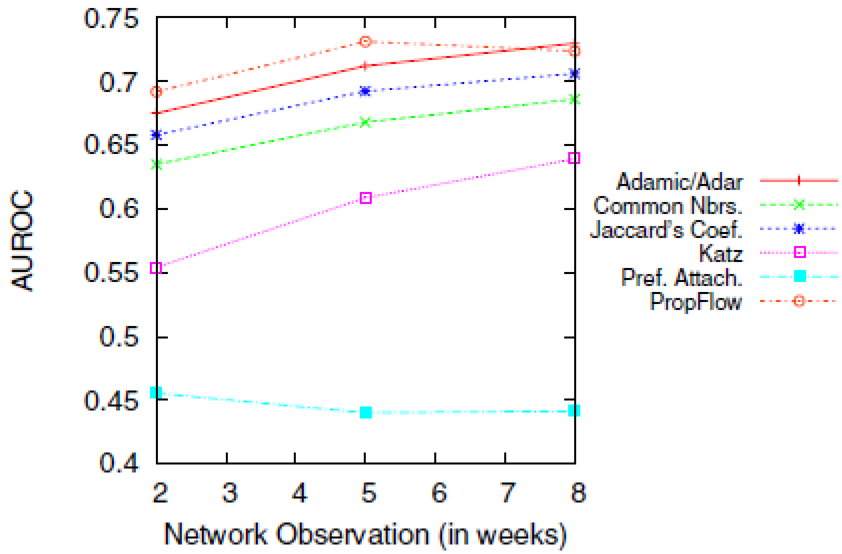


Figure 1: Performance in the second-degree neighborhood as a function of τ_x.

Datasets

712 million cellular phone calls

- weighted, directed networks, weights correspond to the *number of calls*
- use the first 5 weeks of data (5.5M nodes, 19.7M links) for extracting

features and the 6th week (4.4M nodes, 8.5M links) for obtaining ground truth.

19,464 condensed matter physics collaborations from 1995 to 2000.

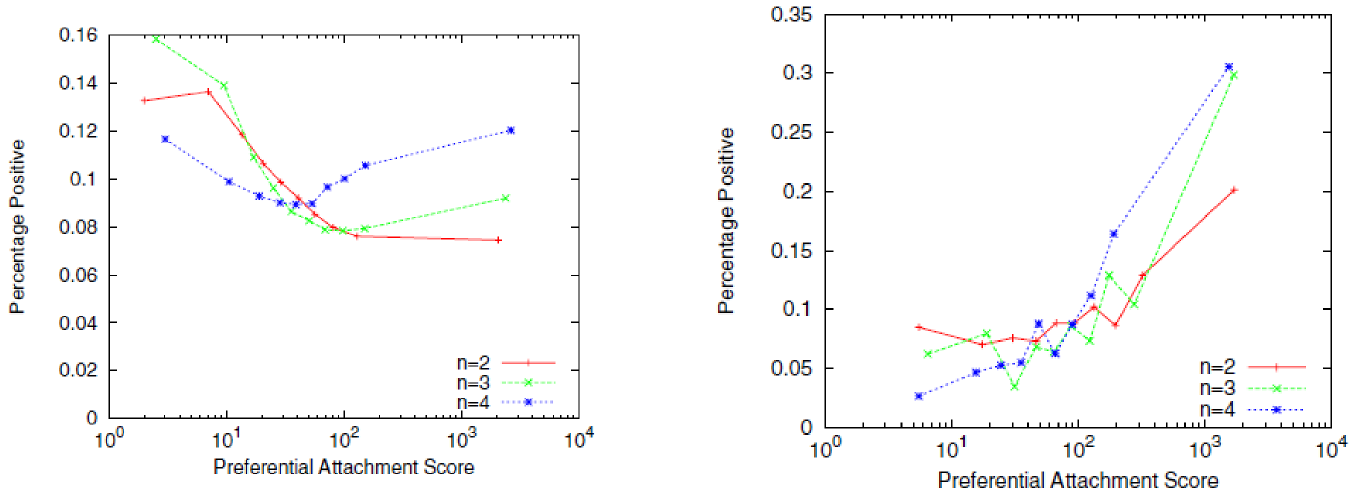
- weighted, undirected networks, weights correspond to the number of collaborations two authors share.
- use the years 1995 to 1999 (13.9K nodes, 80.6K links) for extracting features and the year 2000 (8.5K nodes, 41.0K links) for obtaining ground truth.

Table 1: Network Characteristics

	phone	condmat
Assortativity Coef.	0.293	0.177
Average Clustering Coef.	0.187	0.642
Mean Degree	3.88	6.42
Median Degree	3	4
Number of SCCs	1,023,044	652
Largest SCC	4,293,751	15,081
Largest SCC Diameter	25	19

The **assortativity coefficient** is the *Pearson correlation coefficient* of degree between pairs of linked nodes. Positive values indicate a correlation between nodes of similar degree, while negative values indicate relationships between nodes of different degree.

Using Supervised Learning: why?

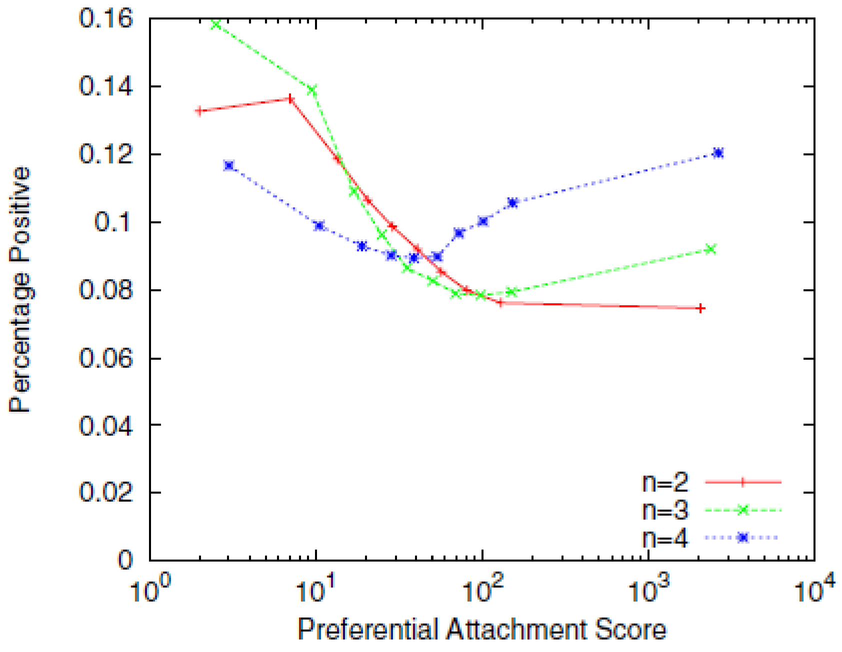


(a) phone (b) condmat

A different prediction model for *each distance*

- Predictors that work well in one network not in another
- Should increase with the score (not in phone)
- Preferential attachment increase with distance (when other may fail)

Using Supervised Learning: why?



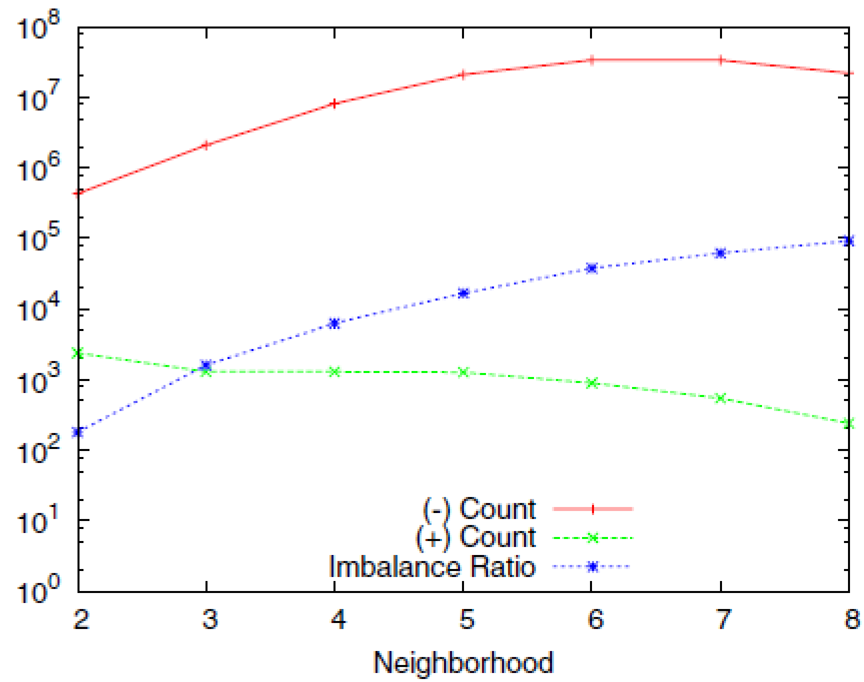
- Even training on a single feature may outperform ranking (if no clear bound on score)
- Dependencies between features – use an ensemble of features

Imbalance

- Sparse networks: |E| = k |V| for constant k << |V|

The class imbalance ratio for link prediction in a sparse network is Ω(|V|/1) when at most |V| nodes are added

Missing links is $|V|^2$
Positives V



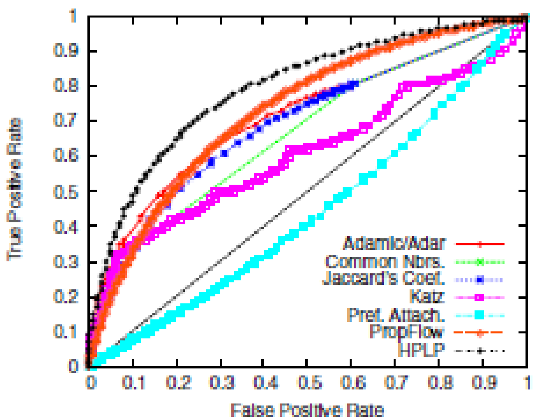
n-neighborhood exactly n hops way
Treat each neighborhood as a separate problem

Results

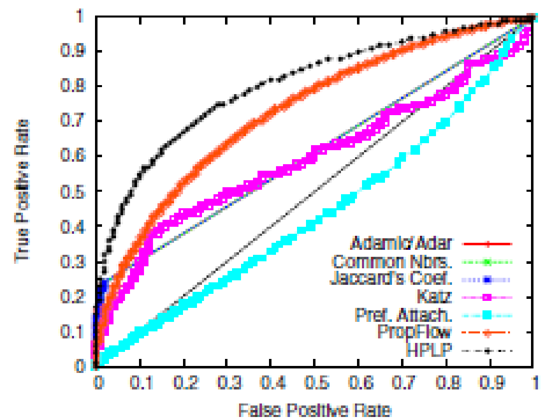
Ensemble of classifiers: Random Forest

Random forest: Ensemble classifier
constructs a multitude of decision trees at training time
output the class that is the mode (most frequent) of the classes (classification) or mean prediction (regression) of the individual trees.

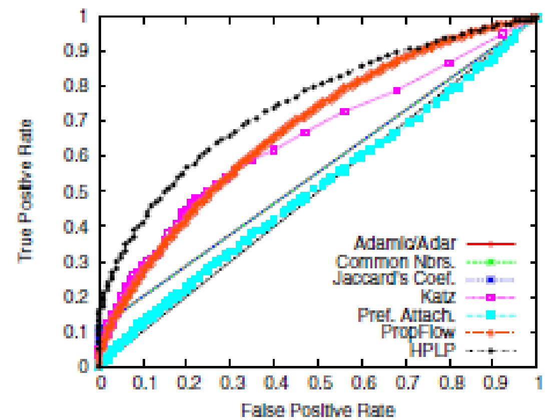
Results



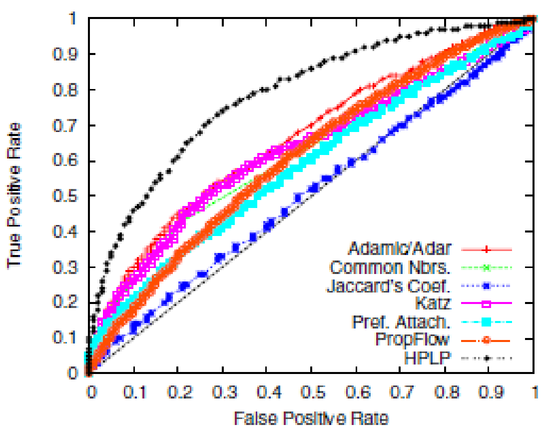
(a) phone $n = 2$



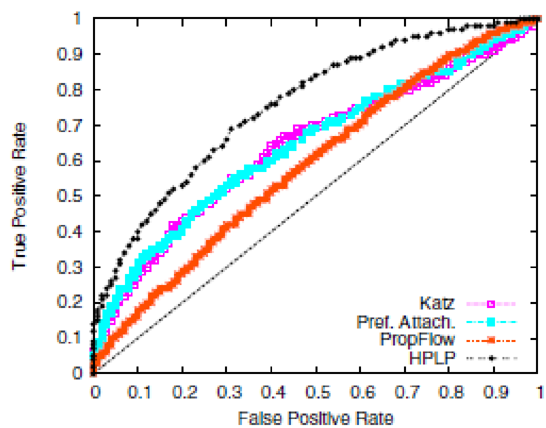
(b) phone $n = 3$



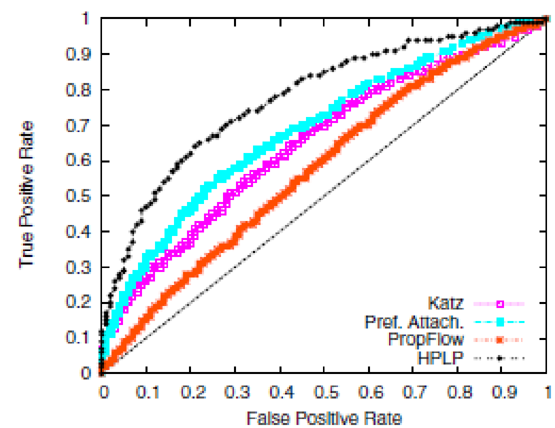
(c) phone $n = 4$



(d) condmat $n = 2$



(e) condmat $n = 3$



(f) condmat $n = 4$

Results

- Mechanism by which links arise different both across networks and geodesic distances.
- Local vs Global (preferential attachment)
  - Better in condmat network,
  - Improves with distance
- HPLP achieves performance levels as much as 30% higher than the best unsupervised methods

Salsa

An application: Wtf

Wtf ("Who to Follow"): *the Twitter user recommendation service*

Twitter graph statistics (August 2012)

- over *20 billion edges* (only active users)
- power law distributions of in-degrees and out-degrees.
  - over 1000 with more than 1 million followers,
  - 25 users with more than 10 million followers.

Is it a "social" network as Facebook?

Difference between:

- *Interested in*
- Similar to

Example (follow @espn but not similar to it)

- do not follow users similar to you, but follow users that the users that are similar to you follow

Algorithms

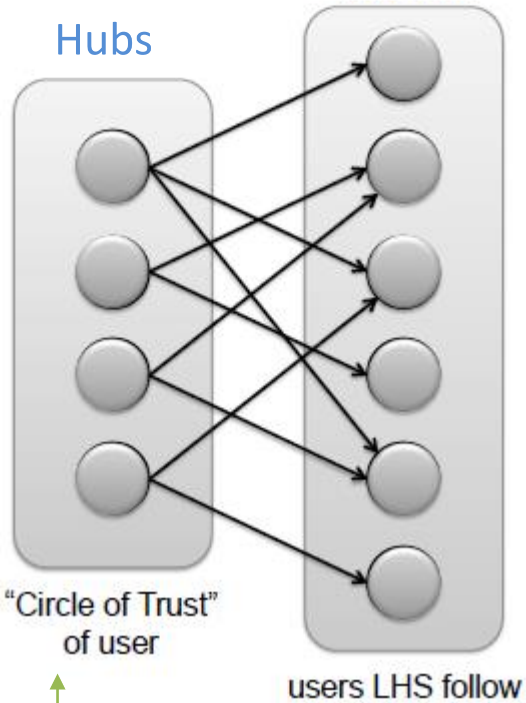
- Asymmetric nature of the follow relationship (other social networks e.g., Facebook or LinkedIn require the consent of both participating members)

- Directed edge case is similar to the user-item recommendations problem where the “item” is also a user.

Bipartite graph

Authorities

Hubs



Users similar to u

Users followed by similar to u users

Hubs: 500 top-ranked nodes from the user's circle of trust

Circle of trust: the result of an egocentric random walk (similar to personalized PageRank)

Authorities: users that the hubs follow.

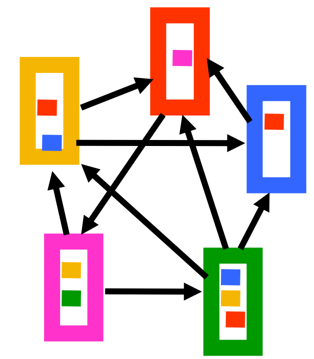
Algorithms: SALSA

SALSA (Stochastic Approach for Link-Structure Analysis)

a variation of HITS

As in HITS

- hubs
- authorities



HITS

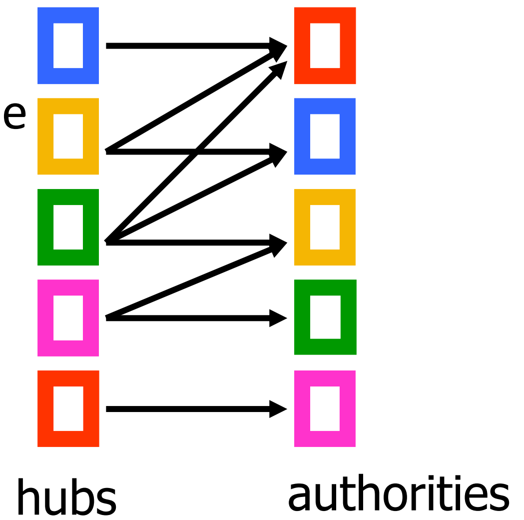
- Good hubs point to good authorities
- Good authorities are pointed by good hubs

hub weight = sum of the authority weights of the authorities pointed to by the hub

$$h_i = \sum_{j: i \to j} a_j$$

authority weight = sum of the hub weights that point to this authority.

$$a_i = \sum_{j: j \to i} h_j$$



Algorithms: SALSA

Random walks to rank hubs and authorities

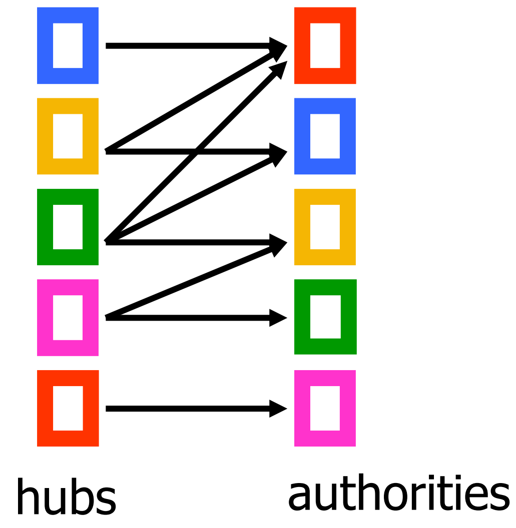
- Two different random walks (Markov chains): a chain of hubs and a chain of authorities
- Each walk traverses nodes only in one side by traversing two links in each step h->a->h, a->h->a

Transition matrices of each Markov chain: H and A

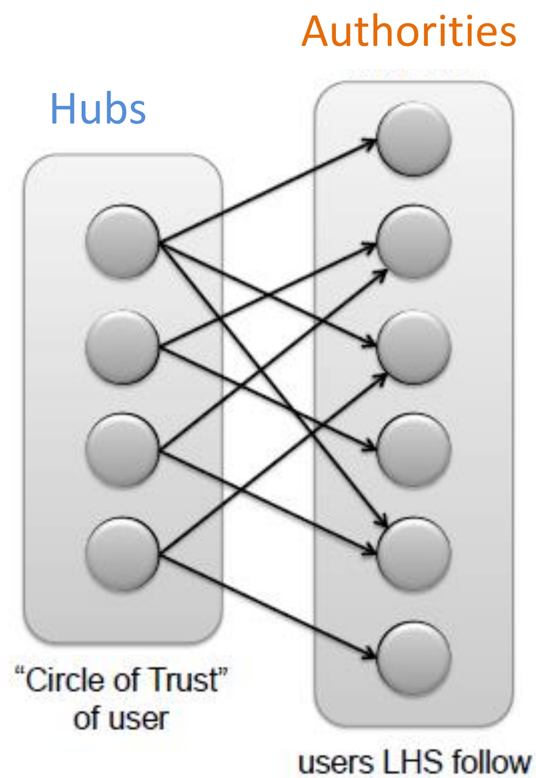
W: the adjacency of the directed graph
W_r: divide each entry by the sum of its row
W_c: divide each entry by the sum of its column

$H = W_r W_c^T$
$A = W_c^T W_r$

Proportional to the degree



Algorithms: SALSA



Use SALSA to assign scores to both sides

Hub scores: user similarity (based on homophily, also useful)

Authority scores : "interested in" user recommendations.

Recommend best in the RHS

SALSA: summary

How it works

SALSA mimics the recursive nature of the problem:

- A user u is likely to follow those who are followed by users that are similar to u.
- A user is similar to u if the user follows the same (or similar) users.

I. SALSA provides *similar users* to u on the LHS and *similar followings* of those on the RHS.
II. The random walk ensures equitable distribution of scores in both directions
III. Similar users are selected from the circle of trust of the user through personalized PageRank.

Real evaluation

- Offline experiments on retrospective data
- Online A/B testing on live traffic

Various parameters may interfere:

- How the results are rendered (e.g., explanations)
- Platform (mobile, etc.)
- New vs old users

Evaluation: metrics

Follow-through rate (FTR) (precision)

- Does not capture recall
- Does not capture lifecycle effects (newer users more receptive, etc.)
- Does not measure the quality of the recommendations: all follow edges are not equal

Engagement per impression (EPI):
After a recommendation is accepted, the *amount of engagement* by the user on that recommendation in a specified time interval called the observation interval.

Extensions

- Add *metadata to vertices* (e.g., user profile information) *and edges* (e.g., edge weights, timestamp, etc.)
- Consider *interaction graphs* (e.g., graphs defined in terms of retweets, favorites, replies, etc.)

References

D. Liben-Nowell, 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)

R. Lichtenwalter, J. T. Lussier, N. V. Chawla: *New perspectives and methods in link prediction*. KDD 2010: 243-252

G. Jeh, J. Widom: *SimRank: a measure of structural-context similarity*. KDD 2002: 538-543

P-N Tan, . Steinbach, V. Kumar. Introduction to Data Mining (Chapter 4)

P. Gupta, A. Goel, J. Lin, A. Sharma, D. Wang, R.Zadeh. *WTF: The Who to Follow Service at Twitter,* WWW 2013

R. Lempel, S. Moran: *SALSA: the stochastic approach for link-structure analysis*. ACM Trans. Inf. Syst. 19(2): 131-160 (2001)