Announcements

- HW4 is due today
- **Project report due:**
  Monday December 10 at 11:59am Pacific Time
  - Note this is 1 day earlier than originally announced!
  - Hand-in a paper copy **and** submit a PDF at http://snap.stanford.edu/submit/
- **Project poster session:**
  Monday December 10 12:00-3:00pm

Link Prediction and Social Circle Detection

CS224W: Social and Information Network Analysis

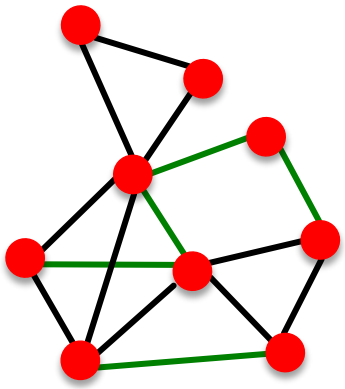
Jure Leskovec, Stanford University

http://cs224w.stanford.edu

[LibenNowell-Kleinberg '03]

Link Prediction in Networks

- **The link prediction task:**
  - Given $G[t_0,t_0']$ a graph on edges up to time t_0' **output a ranked list** L of links (not in $G[t_0,t_0']$) that are predicted to appear in $G[t_1,t_1']$



- **Evaluation:**
  - $n = |E_{new}|$: # new edges that appear during the test period $[t_1,t_1']$
  - Take top n elements of L and count correct edges

[LibenNowell-Kleinberg '03]

Link Prediction via Proximity

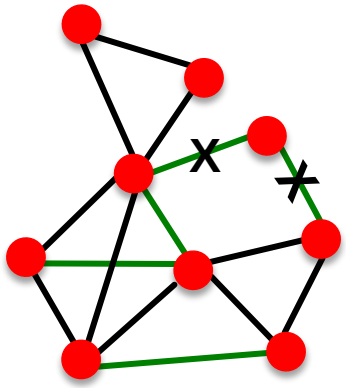
- **Predict links in a evolving collaboration network**

	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

- **Core:** Since network data is very sparse
  - Consider only nodes with in-degree and out-degree of at least 3

Link Prediction via Proximity

- **Methodology:**
  - For each pair of nodes *(x,y)* compute score *c(x,y)*
    - For example: # of common neighbors *c(x,y)* of *x* and *y*
  - Sort pairs *(x,y)* by the decreasing score *c(x,y)*
    - **Note:** Only consider/predict edges where both endpoints are in the core (*deg. > 3*)
  - **Predict top *n* pairs as new links**
  - **See which of these links actually appear in $G[t_1, t'_1]$**



[LibenNowell-Kleinberg '03]

Link Prediction via Proximity

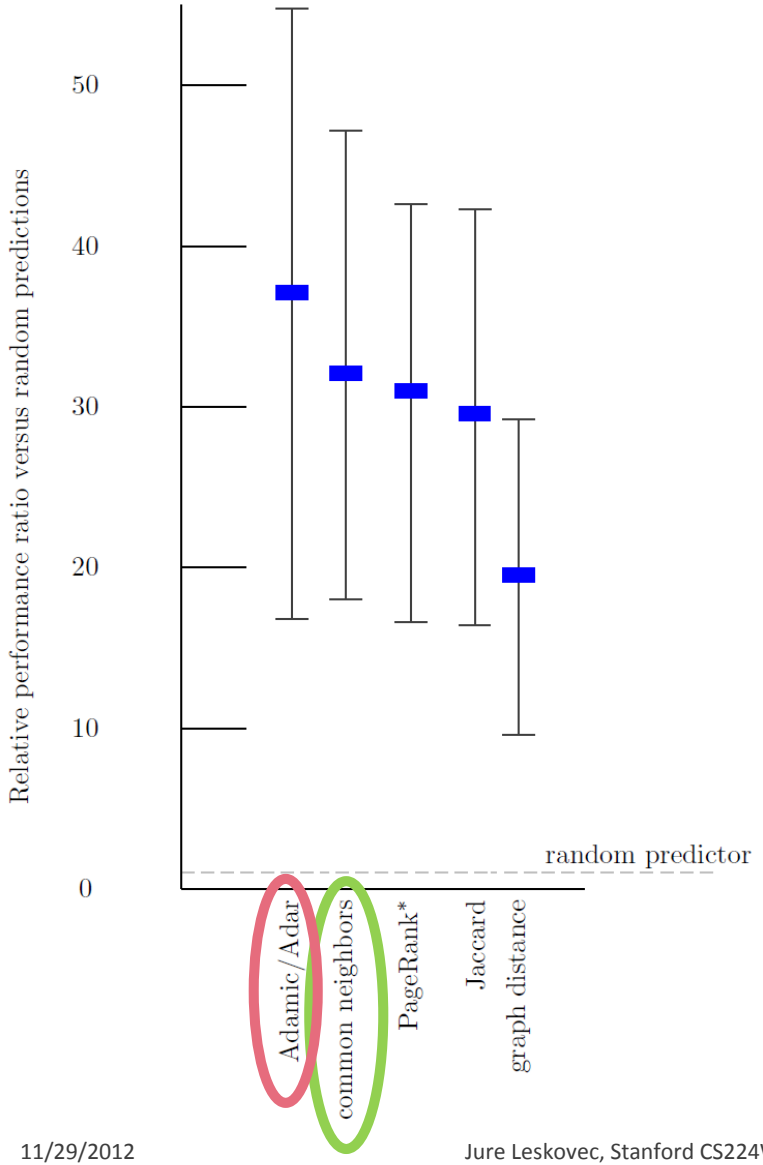
- **Different scoring functions *c(x,y)***
  - **Graph distance:** (negated) Shortest path length
  - **Common neighbors:** $|\Gamma(x) \cap \Gamma(y)|$
  - **Jaccard's coefficient:** $|\Gamma(x) \cap \Gamma(y)|/|\Gamma(x) \cup \Gamma(y)|$
  - **Adamic/Adar:** $\sum_{z \in \Gamma(x) \cap \Gamma(y)} 1/\log |\Gamma(z)|$
  - **Preferential attachment:** $|\Gamma(x)| \cdot |\Gamma(y)|$
  - **PageRank:** $r_x(y) + r_y(x)$
    - $r_x(y)$ … stationary distribution weight of *y* under the random walk:
      - with prob. 0.15, jump to *x*
      - with prob. 0.85, go to random neighbor of current node

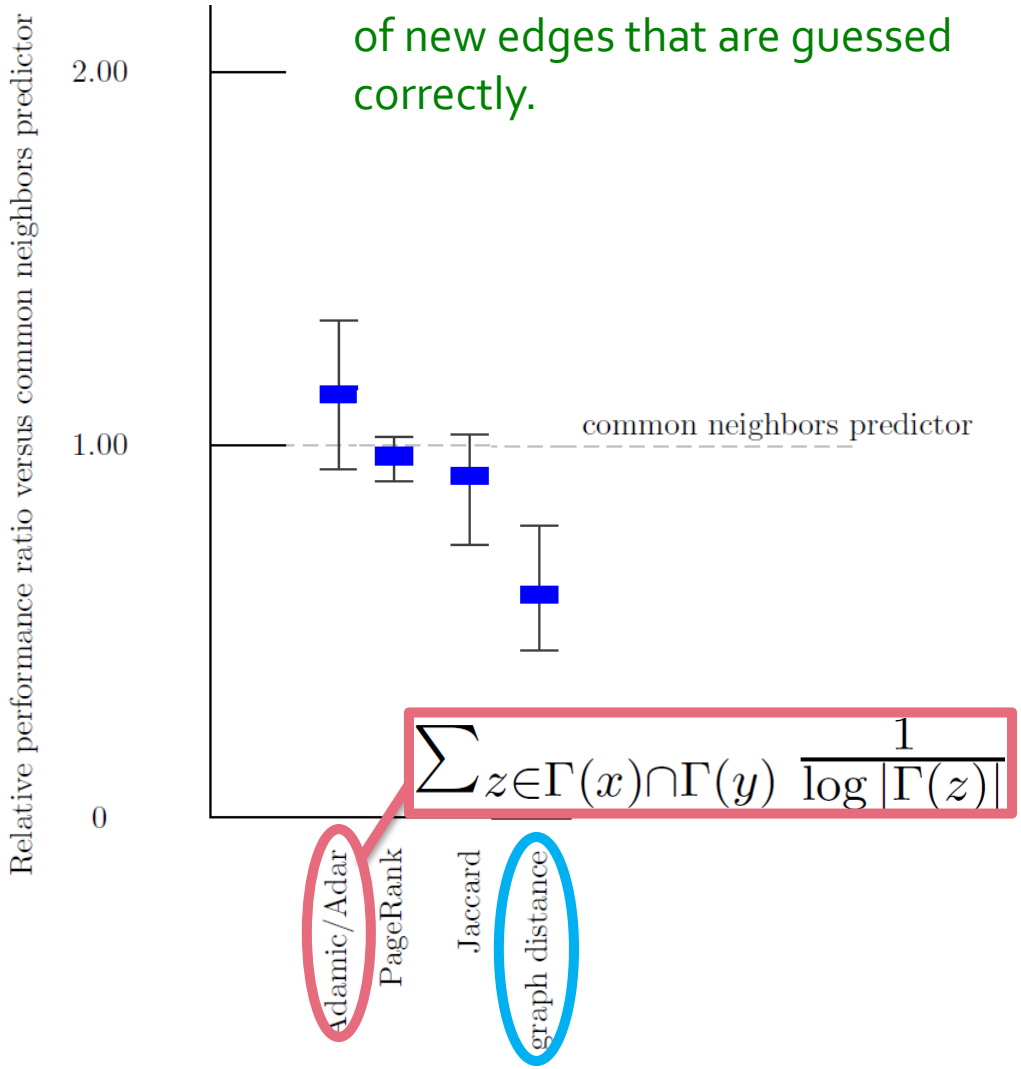
$\Gamma(x)$ … neighbors of node x

- **Then, for a particular choice of *c(·)***
  - For every pair of nodes *(x,y)* compute *c(x,y)*
  - Sort pairs *(x,y)* by the decreasing score *c(x,y)*
  - **Predict top *n* pairs as new links**

[LibenNowell-Kleinberg ' 03]

Results: Improvement





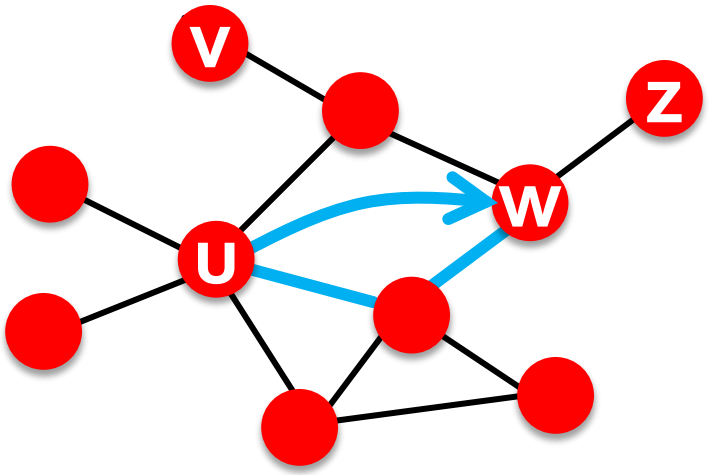
Performance score: Fraction of new edges that are guessed correctly.

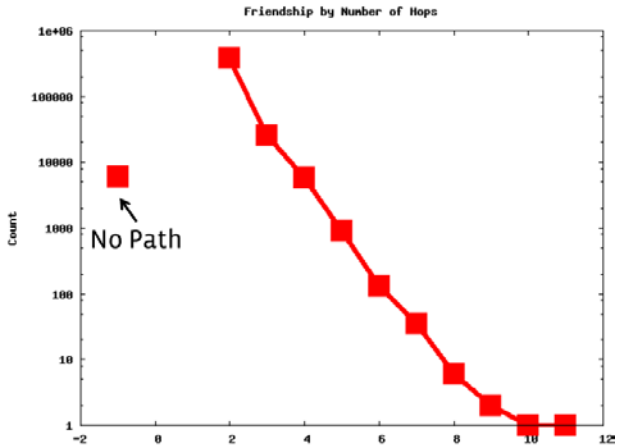
Supervised Random Walks

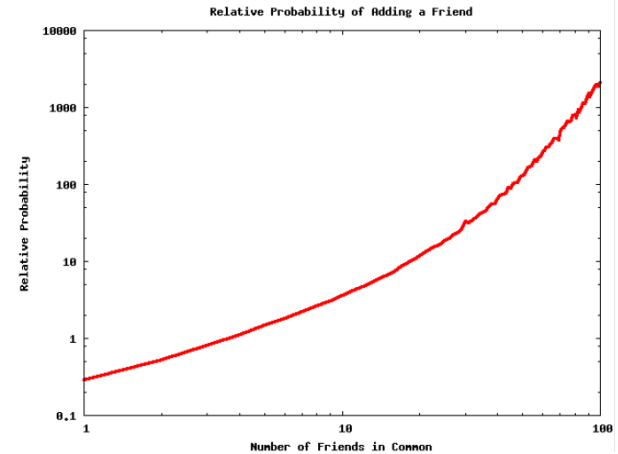
[WSDM '11]

Supervised Link Prediction

- **Can we learn to predict new friends?**
  - Facebook's People You May Know
  - **Let's look at the data:**
    - 92% of new friendships on FB are friend-of-a-friend
    - More common friends helps

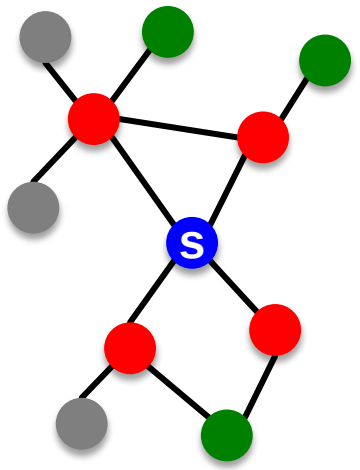






Supervised Link Prediction

- **Goal: Recommend a list of possible friends**
- **Supervised machine learning setting:**
  - **Labeled training examples:**
    - For every user s have a list of others she will create links to $\{v_1 \ \ldots \ v_k\}$ **in the future**
      - *U*se FB network from May 2012 and $\{v_1 \ \ldots \ v_k\}$ are the new friendships you created since then
      - These are the "positive" training examples
    - Use all other users as "negative" example
  - **Task:**
    - For a given node s **score** nodes $\{v_1 \ \ldots \ v_k\}$ **higher** than any other node in the network

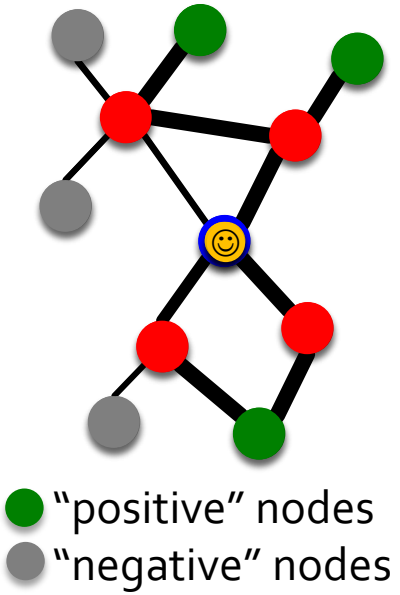


"positive" nodes
"negative" nodes

Green nodes are the nodes to which **s** creates links in the future

Supervised Link Prediction

- **How to combine node/edge features and the network structure?**
  - Estimate **strength** of each friendship (u, v) using:
    - Profile of user u, profile of user v
    - Interaction history of users u and v
  - This creates a **weighted graph**
  - Do **Personalized PageRank from s** and measure the "**proximity**" (the visiting prob.) of any other node w from s
  - Sort nodes w by decreasing **"proximity"**

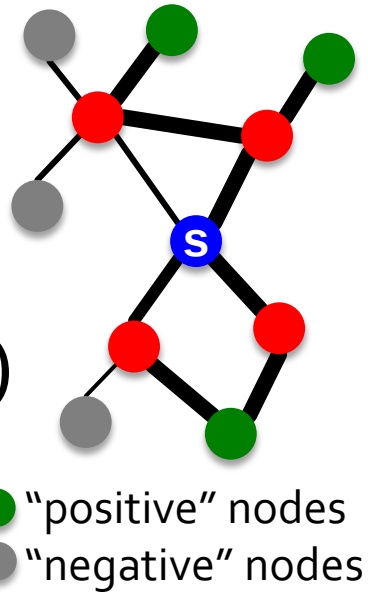


Supervised Random Walks

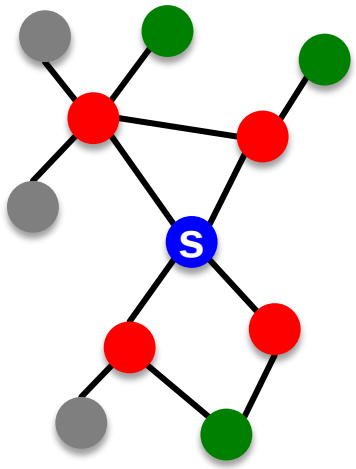
- Let s be the center node
- Let $f_\beta(u, v)$ be a function that assigns **strength a_{uv} to edge (u, v)**

$$a_{uv} = f_\beta(u, v) = \exp(-\sum_i \beta_i \cdot \mathrm{x}_{uv}[i])$$

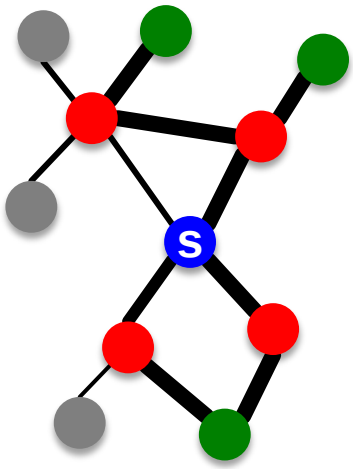
  - **x_{uv} is a feature vector of (u, v)**
    - Features of node u
    - Features of node v
    - Features of edge (u, v)
  - **Note: β is the weight vector we will later estimate!**
- Do **Random Walk with Restarts** from s where transitions are according to edge strengths a_{uv}



SRW: Prediction



Network



Set edge strengths
$a_{uv} = f_\beta(u,v)$

Random Walk with Restarts on the weighted graph. Each node *w* has a PageRank proximity p_w

Sort nodes by the decreasing PageRank score p_w

Recommend top *k* nodes with the highest proximity p_w to node *s*

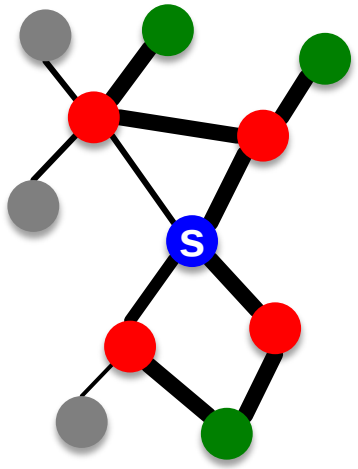
- How to estimate edge strengths?
  - How to set parameters β of $f_\beta(u,v)$?
- **Idea:** Set β such that it (correctly) predicts the known future labels

[WSDM '11]

Personalized PageRank

- a_{uv} …. Strength of edge (u, v)
- **Random walk transition matrix:**

$$Q'_{uv} = \begin{cases} \frac{a_{uv}}{\sum_w a_{uw}} & \text{if } (u,v) \in E, \\ 0 & \text{otherwise} \end{cases}$$



- **PageRank transition matrix:**

$$Q_{ij} = (1-\alpha)Q'_{ij} + \alpha \mathbf{1}(j = s)$$

  - Where with prob. α we jump back to node s

- Compute PageRank vector: $p = p^T Q$
- Rank nodes w by decreasing p_w

The Optimization Problem

- **Positive** examples
  $D = \{d_1, \ldots, d_k\}$
- **Negative** examples
  $L = \{other\ nodes\}$
- **What do we want?**

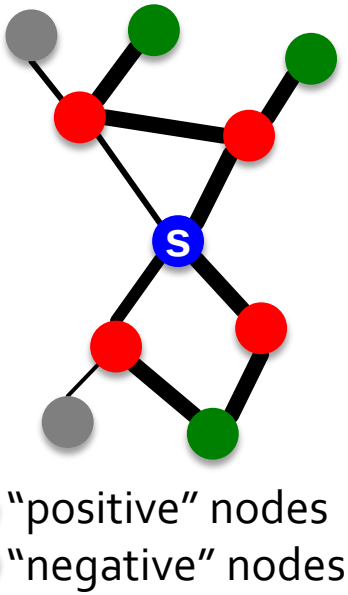
$$\min_{\beta} F(\beta) = ||\beta||^2$$

We prefer small weights β

such that

$$\forall\, d \in D, l \in L: \; p_l < p_d$$

Every positive example has to have higher PageRank score than every negative example



- **Note:**
  - Exact solution to this problem may not exist
  - So we make the constrains "soft" (i.e., optional)

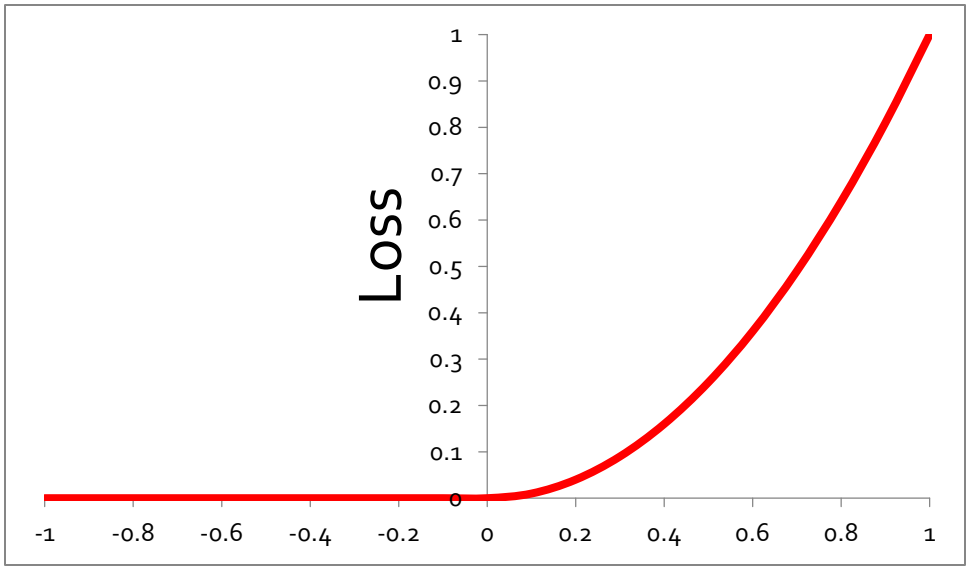
Making Constraints "Soft"

- **Want to minimize:**

$$\min_{\beta} F(\beta) = \sum_{d \in D, l \in L} h(p_l - p_d) + \lambda ||\beta||^2$$

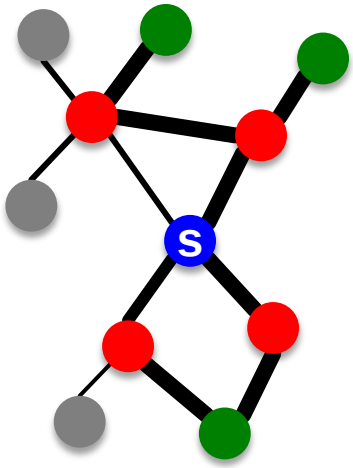
  - **Loss:** $h(x) = 0$ if $x < 0$, or x^2 else

Penalty for violating the constraint that $p_d > p_l$



$p_l < p_d$ $p_l = p_d$ $p_l > p_d$

Solving the Problem: Intuition



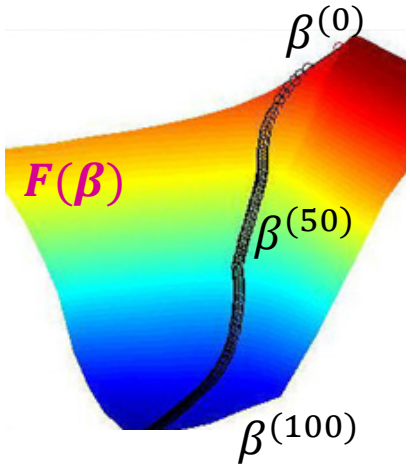
- **How to minimize $F(\beta)$?**

$$\min_{\beta} F(\beta) = \sum_{d \in D, l \in L} h(p_l - p_d) + \lambda ||\beta||^2$$

  - **Both p_l and p_d depend on β**
    - Given β assign edge weights $a_{uv} = f_\beta(u, v)$
    - Using $Q = [a_{uv}]$ compute PageRank scores p_w
    - Rank nodes by the decreasing score
- **Idea:**
  - Start with some random $\beta^{(0)}$
  - Evaluate the derivative of $F(\beta)$ and do a small step in the opposite direction

$$\beta^{(t+1)} = \beta^{(t)} - \eta \frac{\partial F(\beta^{(t)})}{\partial \beta}$$

  - Repeat until convergence



Gradient Descent

- **What's the derivative $\frac{\partial F(\beta^{(t)})}{\partial \beta}$?**

$$\frac{\partial F(\beta)}{\partial \beta} = \sum_{l,d} \frac{\partial h(p_l - p_d)}{\partial \beta} + 2\lambda\beta$$

$$= \sum_{l,d} \frac{\partial h(p_l - p_d)}{\partial (p_l - p_d)} \left(\frac{\partial p_l}{\partial \beta} - \frac{\partial p_d}{\partial \beta}\right) + 2\lambda\beta$$

$h(x) = \max\{x, 0\}^2$

Easy!

- **We know:**

$$p = p^T Q \quad \textit{that is} \quad p_u = \sum_j p_j Q_{ju}$$

- **So:**

$$\frac{\partial p_u}{\partial \beta} = \sum_j Q_{ju} \frac{\partial p_j}{\partial \beta} + p_j \frac{\partial Q_{ju}}{\partial \beta}$$

Gradient Descent

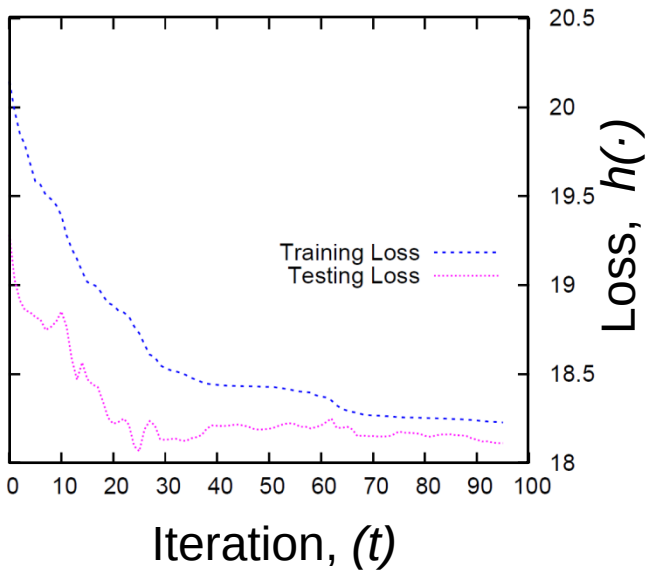
- **We just got:** $\frac{\partial p_u}{\partial \beta} = \sum_j Q_{ju} \frac{\partial p_j}{\partial \beta} + p_j \frac{\partial Q_{ju}}{\partial \beta}$
  - Few details:
    - Computing $\partial Q_{ju}/\partial \beta$ is easy. **Remember:** $Q'_{uv} = \begin{cases} \frac{a_{uv}}{\sum_w a_{uw}} & \text{if } (u,v) \in E, \\ 0 & \text{otherwise} \end{cases}$

      $a_{uv} = f_\beta(u,v)$

      $= \exp\left(-\sum_i \beta_i \cdot \mathrm{x}_{uv}[i]\right)$
    - We want $\frac{\partial p_j}{\partial \beta}$ but it appears on both sides of the equation. Notice the whole thing looks like a PageRank equation: $x = Q \cdot x + z$
- **As with PageRank we can use the power-iteration to solve it:**
  - Start with a random $\frac{\partial p^{(0)}}{\partial \beta}$
  - Then iterate: $\frac{\partial p^{(t+1)}}{\partial \beta} = Q \cdot \frac{\partial p^{(t)}}{\partial \beta} + \frac{\partial Q_{ju}}{\partial \beta} \cdot p$

Optimizing $F(\beta)$

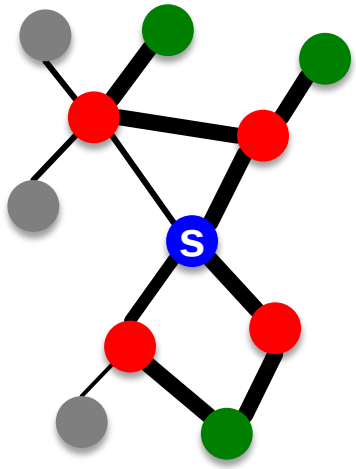
- **To optimize $F(\beta)$, use gradient descent:**
  - Pick a random starting point $\beta^{(0)}$
  - Using current $\beta^{(t)}$ compute edge strenghts and the transition matrix Q
  - Compute PageRank scores p
  - Compute the gradient with respect to weight vector $\beta^{(t)}$
  - Update $\beta^{(t+1)}$



[WSDM '11]

Data: Facebook

- **Facebook Iceland network**
  - 174,000 nodes (55% of population)
  - Avg. degree 168
  - Avg. person added 26 friends/month
- **For every node s:**
  - **Positive examples:**
    - $D = \{$ new friendships s created in Nov '09 $\}$
  - **Negative examples:**
    - $L = \{$ other nodes s did not create new links to $\}$
  - **Limit to friends of friends:**
    - On avg. there are 20,00 FoFs (maximum is 2 million)!



Experimental setting

- **Node and Edge features for learning:**
  - **Node:** Age, Gender, Degree
  - **Edge:** Age of an edge, Communication, Profile visits, Co-tagged photos
- **Evaluation:**
  - **Precision at top 20**
    - We produce a list of 20 candidates
      - By taking top 20 nodes x with highest PageRank score p_x
    - Measure to what fraction of these nodes s actually links to

Results: Facebook Iceland

- **Facebook:** predict future friends
  - Adamic-Adar already works great
  - Supervised Random Walks (SRW) gives slight improvement

Learning Method	Prec@Top20
Random Walk with Restart	6.80
Adamic-Adar	7.35
Common Friends	7.35
Degree	3.25
SRW: one edge type	6.87
SRW: multiple edge types	**7.57**

Results: Co-Authorship

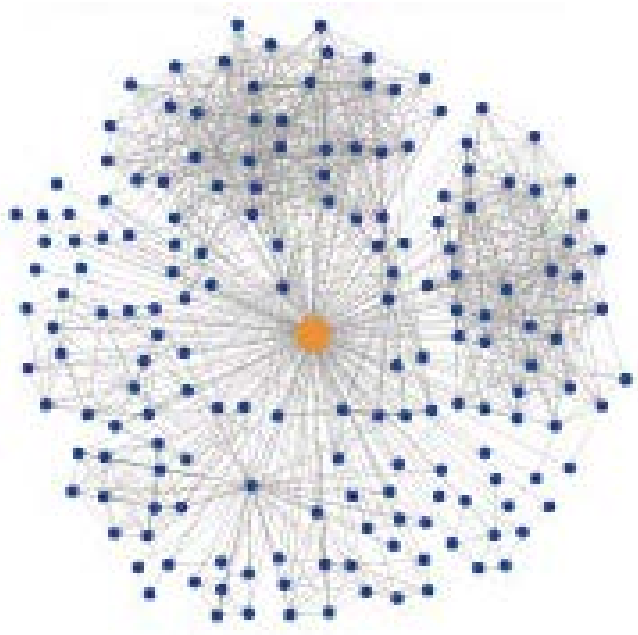
- **Arxiv Hep-Ph collaboration network:**
  - Poor performance of unsupervised methods
  - SRW gives a boost of 25%!

Learning Method	Prec@Top20
Random Walk with Restart	3.41
Adamic-Adar	3.13
Common Friends	3.11
Degree	3.05
SRW: one edge type	4.24
SRW: multiple edge types	**4.25**

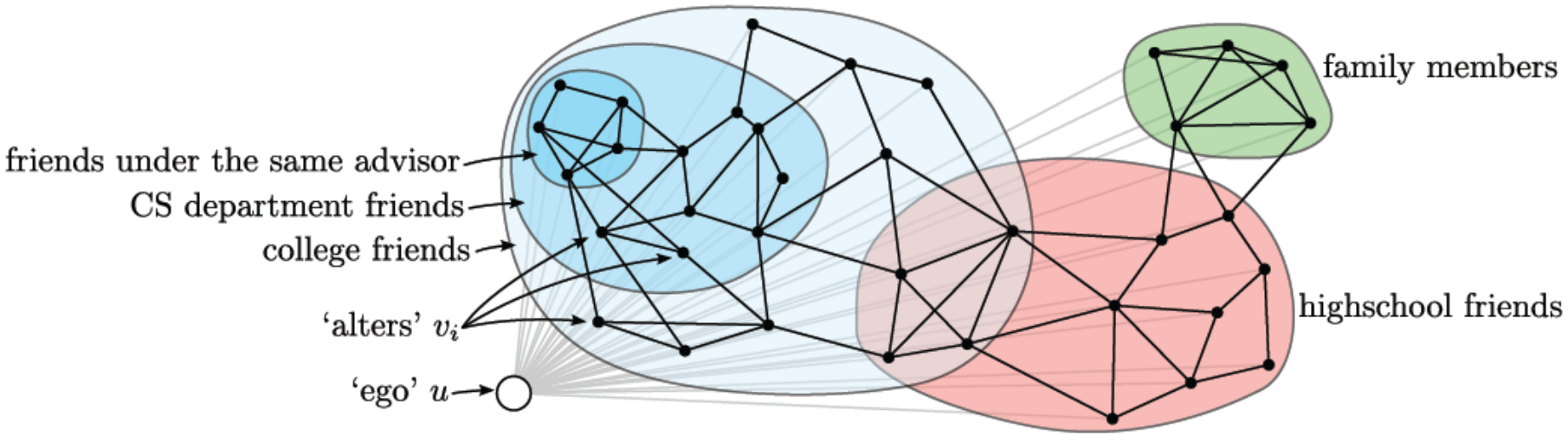
Discovering Social Circles

Social Circles

- **Take a user (yellow circle) and discover her social circles:**
- **Why is it useful?**
  - To organize friend lists
  - Control privacy and access settings
  - Filter content
- **Facebook, Twitter, Google+:**
  - Groups, lists, circles



Social Circles



- Given **ego** node u and a network of her friends
- Find circles!
  - Use network as well as user profile information
  - For each circle we want to know why it is there!

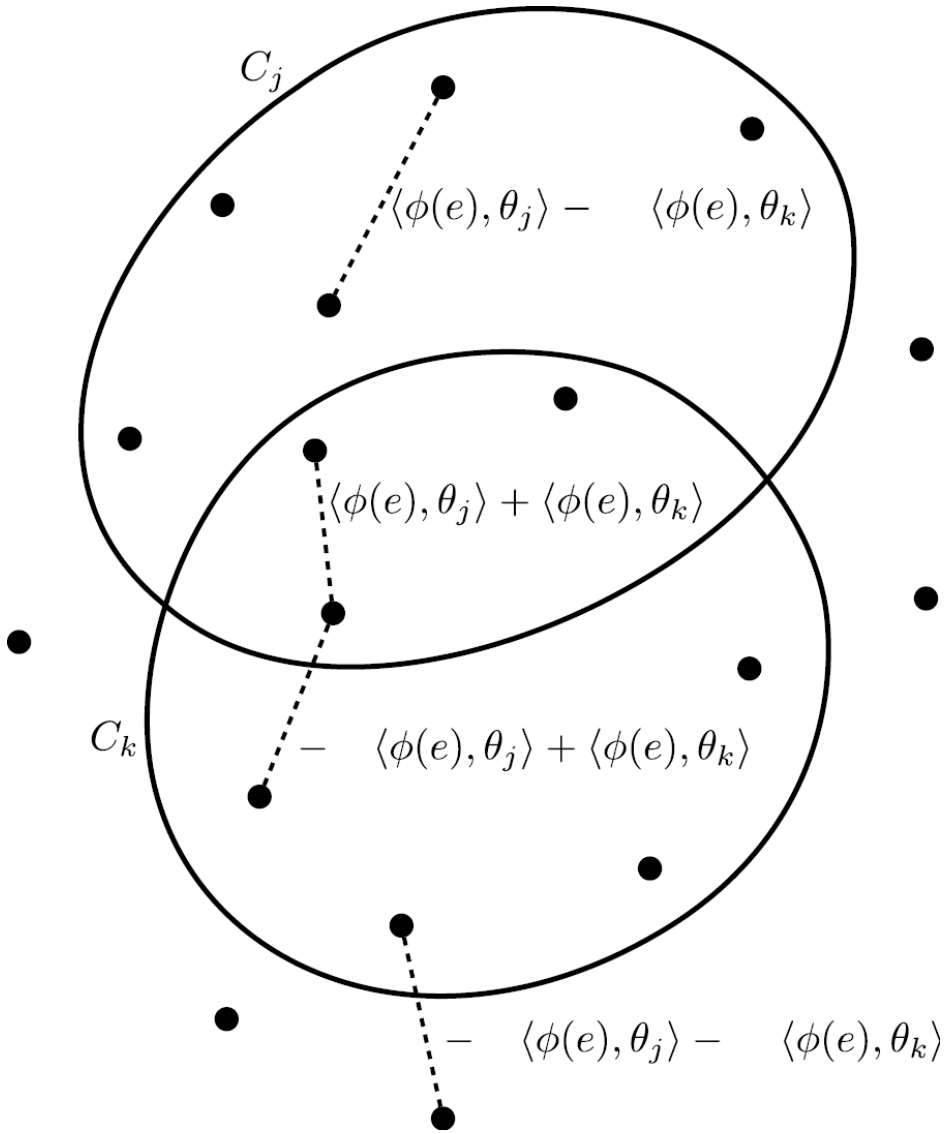
The Model of Circles

- **Suppose we know all the circles C_k in the ego-network**
- **We model the prob. of edge (x, y)**

$$p((x,y) \in E) \propto \exp\left\{ \underbrace{\sum_{C_k \supseteq \{x,y\}} \langle \phi(x,y), \theta_k \rangle}_{\text{circles containing both nodes}} - \underbrace{\sum_{C_k \not\supseteq \{x,y\}} \langle \phi(x,y), \theta_k \rangle}_{\text{all other circles}} \right\}$$

  - where:
    - $\phi(x, y)$… is a feature vector describing (x, y)
      - Are x and y from same school, same town, same age, …
    - θ_k… parameter vector that we aim to estimate
      - High $\theta_k[i]$ means being similar in feature i is important for circle k

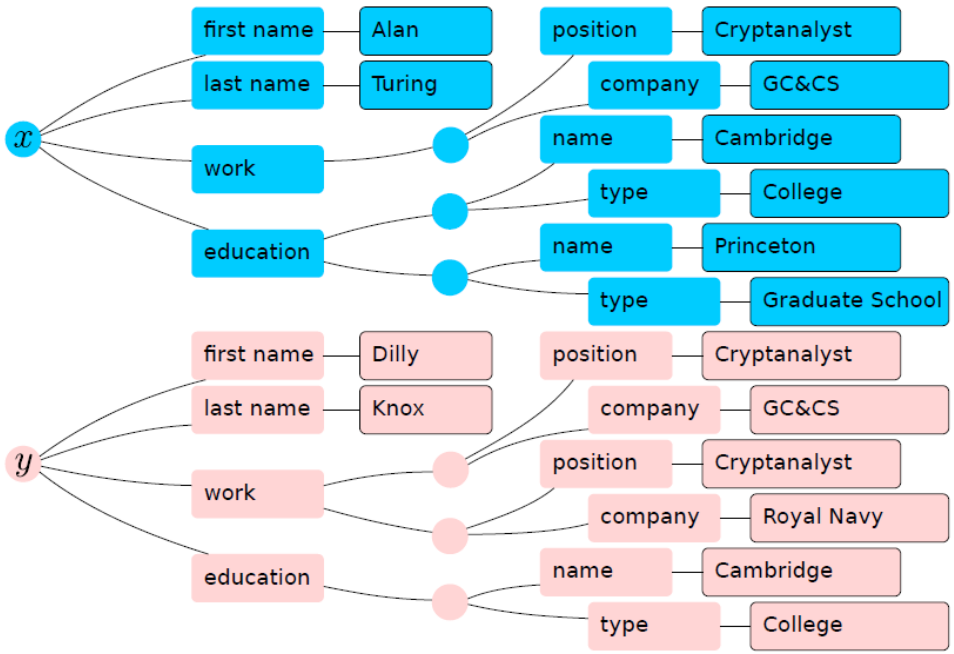
Modeling Edges



- $\phi(e)$… feature vector of edge $e = (x, y)$
  - Are x and y from same school, same town, same age, …
- θ_k… circle node "similarity" parameter
  - High $\theta_k[i]$ means being similar in feature i is important for circle k

Creating the Features $\phi(x, y)$

- **Two ways to create feature vectors $\phi(x, y)$**



$$\phi(x,y) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} \begin{matrix} first\ name : Dilly \\ last\ name : Knox \\ first\ name : Alan \\ last\ name : Turing \\ work : position : Cryptanalyst \\ work : location : GC\&CS \\ work : location : Royal\ Navy \\ education : name : Cambridge \\ education : type : College \\ education : name : Princeton \\ education : type : Graduate\ School \end{matrix}$$

$$\phi(x,y) = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \begin{matrix} first\ name \\ last\ name \\ work : position \\ work : location \\ education : name \\ education : type \end{matrix}$$

Circle Discovery

- **Given an ego-graph G**
- **And edges features $\boldsymbol{\phi(x, y)}$**
- **Want to discover:**
  - Circle node memberships C and
  - Circle parameters θ_k

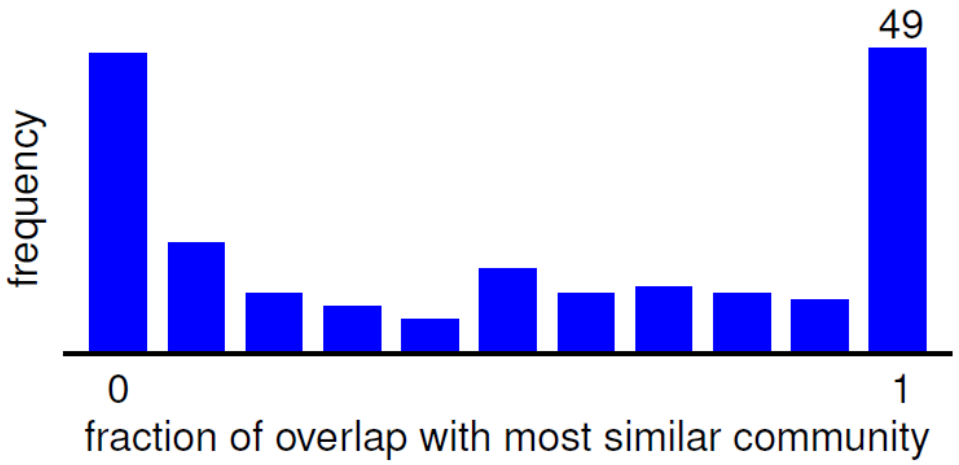
such that we maximize the likelihood:

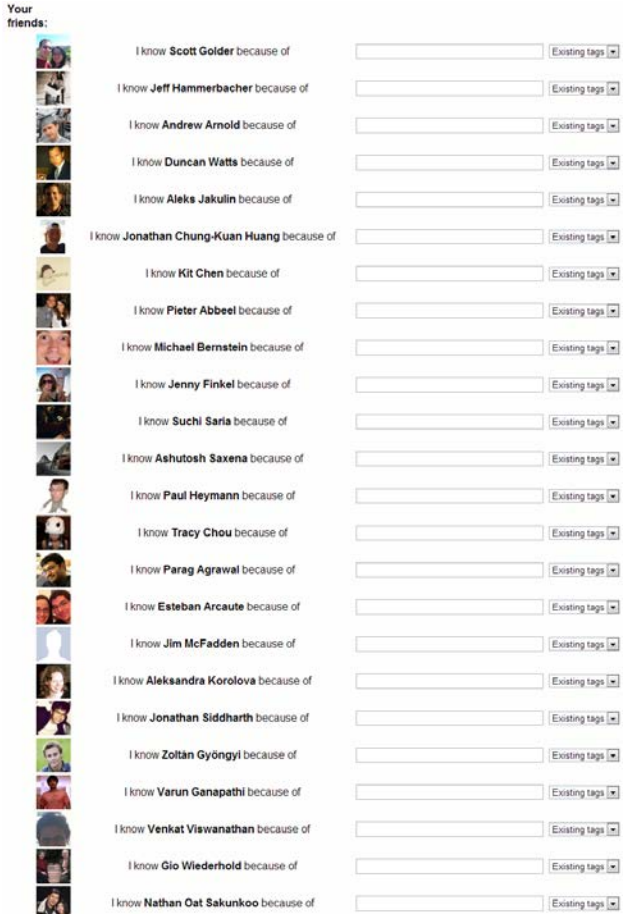
$$P_\Theta(G; \mathcal{C}) = \prod_{e \in E} p(e \in E) \times \prod_{e \notin E} p(e \notin E)$$

To see the details of this is accomplished see: *Discovering Social Circles in Ego Networks* by J. McAuley, J.L. http://arxiv.org/abs/1210.8182

Experiments: Facebook

- **Facebook:**
  - Ask people to go through their friend lists and hand label the circles

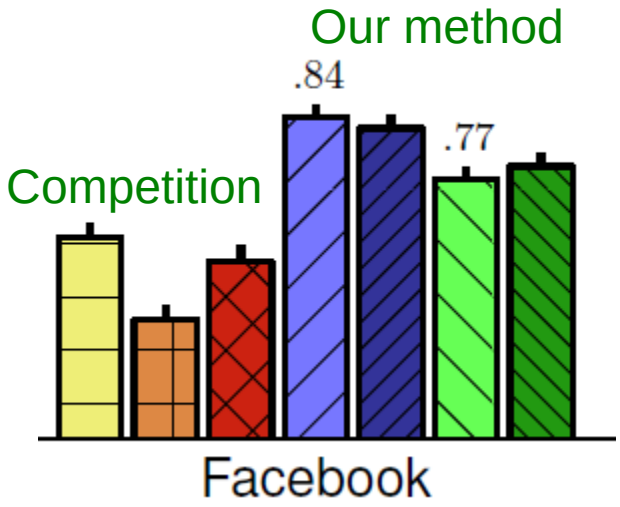




~30% circles don't overlap
~30% overlap
~30% are nested

Experiments: Facebook

- **How well do we recover human identified circles?**



- **Social circles of our very own head TA:**