Learning to Rank Networked Entities

Alekh Agarwal
Soumen Chakrabarti
Sunny Aggarwal

IIT Bombay

www.cse.iitb.ac.in/~soumen/doc/netrank

Learning to rank…

- …feature vectors, studied in detail
  - i th entity represented by feature vector x_i
  - Score of i th entity is dot product $\beta' x_i$
  - Want $\beta' x_i \le \beta' x_j$ if we say "$i \prec j$"
  - Max-margin setup
    $$\min_{\beta \in \mathbb{R}^d} \beta'\beta \text{ subject to } \beta' x_i + 1 \le \beta' x_j \text{ for all } i \prec j$$
  - Other scores, e.g. 2-layer neural net (RankNet)
- …nodes in a graph, less so
  - Strongly motivated by Pagerank and HITS
  - Changing score of one node influences others

Edge conductance and Pagerank

- Conductance of edge $i \to j$ written as $C(j,i)$
  - $C(j,i)$=Pr(j @ this step | i @ previous step)
  - Pagerank vector p satisfies $p = C\,p$
- Unweighted (standard) Pagerank

$$C(j,i) = \begin{cases} \alpha \frac{[(i,j)\in E]}{\text{OutDegree}(i)} + (1-\alpha) r_j & i \in V_o \\ r_j & \text{otherwise} \end{cases}$$

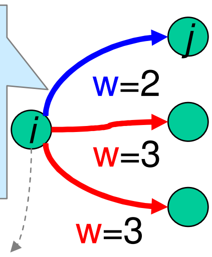
i is a dead-end $\sum_{j \in V} r_j = 1$

- Weighted Pagerank: $i \to j$ edge weight $w(i,j)$

$$C(j,i) = \begin{cases} \alpha \frac{w(i,j)}{\sum_{j'} w(i,j')} + (1-\alpha) r_j & i \in V_o \\ r_j & \text{otherwise} \end{cases}$$

Pr(teleport)

Prob. of following this edge 2/(2+3+3)



Teleport?

Inverse problem

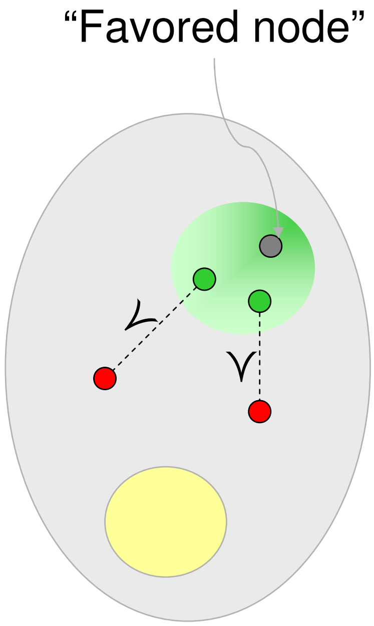
- Traditionally: Given matrix C, find Pagerank
- Clever design of C for various applications
  - Hand-tweak teleport vector r – topic sensitive (Haveliwala), personalized (Jeh+Widom)
  - Hand-tweak $w(i,j)$ (Intelligent Surfer, ObjectRank)
- Our problem: Given partial order $\prec_{\text{train}}$, find C (and p) such that
  - p satisfies $p = Cp$ approximately
  - p satisfies $\prec_{\text{train}}$ and unseen $\prec_{\text{test}}$ well: i.e., $p_i \le p_j$ if $i \prec j$
  - $\prec_{\text{train}}$ and $\prec_{\text{test}}$ comes from same “hypothesis”

Preferred community scenario

- Ranking papers for Data Mining researcher
- Some subgraphs and citations more important than others
- Revealed via pairwise preferences
- Do not estimate $C(j,i)$ directly
- Directly estimate p_{ij}, a constrained "flow" from i to j
- "BTW" $C(j,i) = p_{ij} \big/ \sum_{(k,i)\in E} p_{ki}$

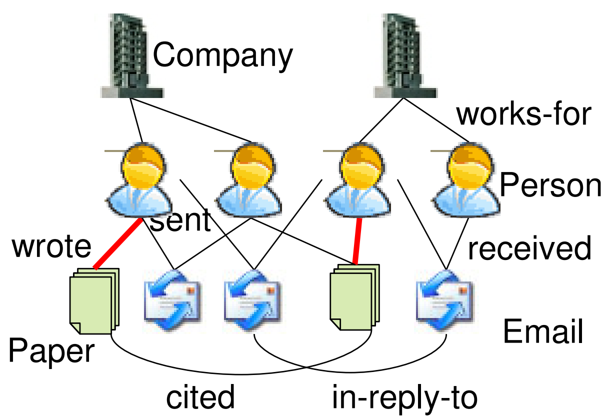
Inflow into i

- Local "transductive" setting
- Lots of parameters



Entity-relationship graph scenario

- Many node and edge types
- Edge e has type $t(e) \in \{1,\ldots T\}$
- Weight $w(i,j) = \beta(t(i,j))$
- Find $\beta(1)$, $\beta(2)$, …, $\beta(T)$ for least violation
- "Global entanglement" but far fewer parameters
- Somewhat "inductive", can augment graph with objects of known types



1: The constrained flow formulation

p_{uv} is interpreted as a flow along (u,v)

$$\min_{\substack{\{0 \le p_{uv} \le 1 : (u,v) \in E'\} \\ \{0 \le s_{uv} : u \prec v\}}} \sum_{(u,v)\in E'} p_{uv} \log \frac{p_{uv}}{q_{uv}} + B \sum_{u \prec v} s_{uv}$$

$\{q_{uv}\}$ is a parsimonious flow (unweighted Pagerank)

$$\sum_{(u,v)\in E'} p_{uv} = 1$$

s_{uv} is the slack or violation on preference $u \prec v$

Balance

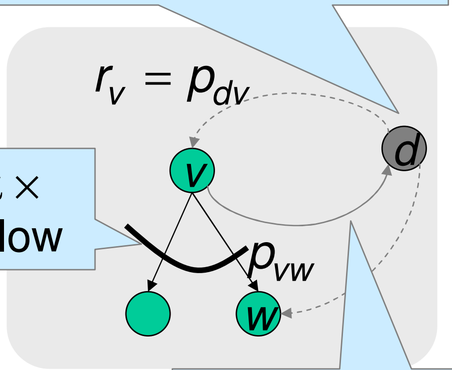
$$\forall v \in V' : \sum_{(u,v)\in E'} p_{uv} = \sum_{(v,w)\in E'} p_{vw}$$

Dummy node implements teleport

Teleport

$$\forall v \in V_o : \frac{p_{vd}}{(1-\alpha)} = \frac{\sum_{(v,w)\in E'} p_{vw}}{\alpha}$$

$\alpha \times$ inflow



Preference

$$\forall u \prec v : (1+\varepsilon) \sum_{(w,u)\in E'} p_{wu} \le s_{uv} + \sum_{(w,v)\in E'} p_{wv}$$

(1-α) × inflow

The dual optimization

- $O(2|V| + |\prec|)$ dual variables
  - Unconstrained β_v for balance, τ_v for teleport
  - $0 \le \pi_{uv} \le B$ for preference
- Primal flows in familiar log-linear form

$$\forall v \in V \quad p_{dv} = (1/Z)\, q_{dv} \exp(\beta_v - \beta_d + \text{bias}(v))$$

$$\forall v \in V_o \quad p_{vd} = (1/Z)\, q_{vd} \exp(\beta_d - \beta_v + \alpha\tau_v)$$

$$\forall v \in V \setminus V_o \quad p_{vd} = (1/Z)\, q_{vd} \exp(\beta_d - \beta_v)$$

$$\forall (u,v) \in E \quad p_{uv} = (1/Z)\, q_{uv} \exp(\beta_v - \beta_u - (1-\alpha)\tau_u + \text{bias}(v))$$

- Where $\text{bias}_\varepsilon(v) = \sum_{r \prec v} \pi_{rv} - (1+\varepsilon) \sum_{v \prec s} \pi_{vs}$
- Dual objective: minimize $\log Z$
- Can include dual vars gradually

Large bias ⇒ large flow into v ⇒ high rank

Competition: Teleport learning via QP

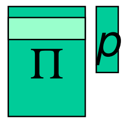
- Let A be node adjacency matrix with no dead ends and rows scaled to sum to 1

$$p = \alpha A' p + (1-\alpha) r, \quad \therefore p = (1-\alpha)(\mathbb{I} - \alpha A')^{-1} r \triangleq M\, r$$

Fit this

- Preference $\prec$ expressed as $\Pi p = \Pi M r \geq 0$

$$\underbrace{(0,\ldots,-1,\ldots,1,\ldots 0)}_{\text{Row of } \Pi}\underbrace{(0,\ldots,p_i,\ldots,p_j,\ldots,0)'}_{\text{Column vector } p} \geq 0$$



- Let r^U be the parsimonious uniform teleport

Deviate from unweighted Pagerank as little as possible…

$$\min_{r \in \mathbb{R}^{|V|}, s \geq 0} (Mr - Mr^U)'(Mr - Mr^U) + B\mathbf{1}' s$$

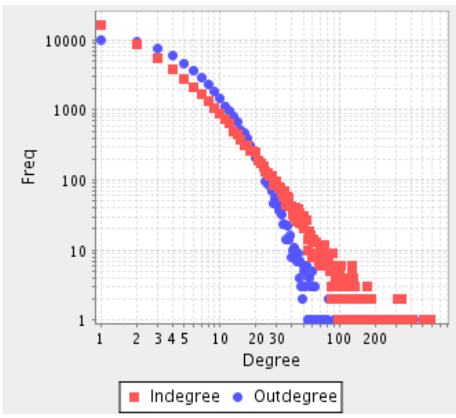
$$\text{s.t.} \quad r \geq \mathbf{0}, \quad \mathbf{1}' r = 1, \quad \Pi M r + s \geq \mathbf{0}$$

Makes QP very expensive

…while satisfying $\prec$

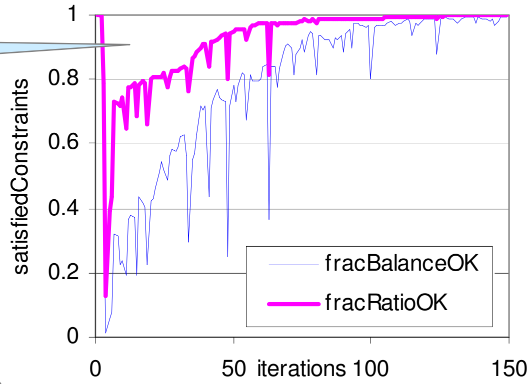
Data set preparation

- No open benchmark for this task
  - ☹No standardized comparison yet
  - ☺We will make code and some data available
  - ☺Synthetic G and $\prec$ can explore space thoroughly
- Generating graph G
  - RMAT (power-law degree, small dia, clustering)
  - Real DBLP+CiteSeer graph
- Generating preference $\prec$
  - Use r_{hidden} to compute p_{hidden}
  - Sample $\prec_{\text{train}}$, $\prec_{\text{test}}$ from p_{hidden}
  - Measure flips on $\prec_{\text{test}}$



Optimization dynamics and performance

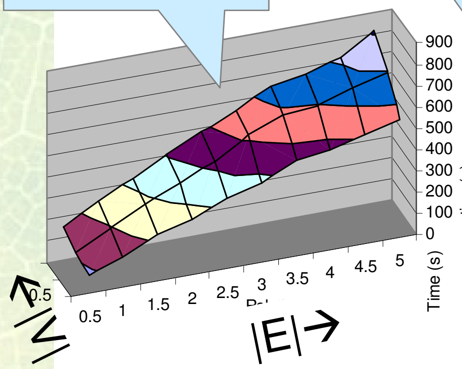
Gradual jittery decrease in primal violations

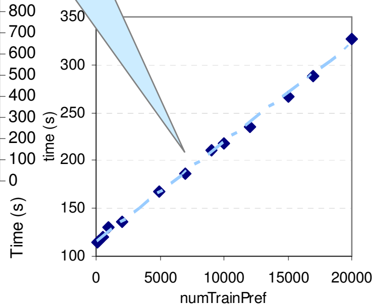


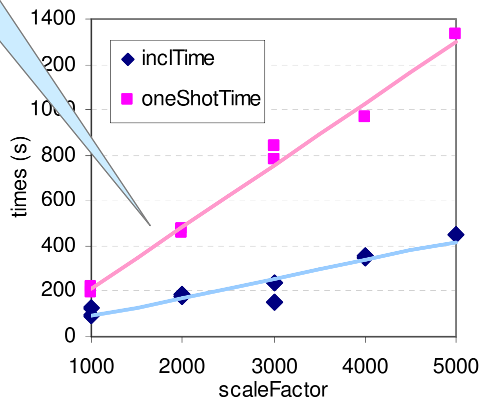
Time scales linearly with $|\prec_{train}|$

Gradual dual var inclusion speedup

Time scales linearly with $|V|$ and $|E|$

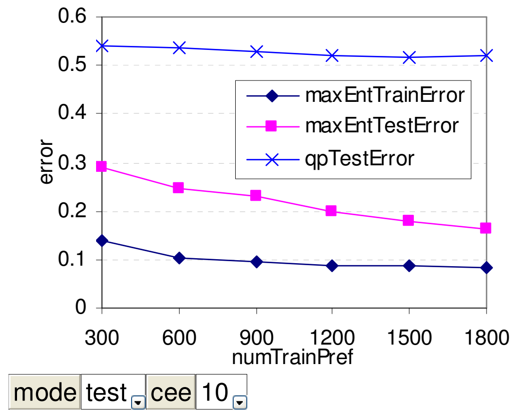


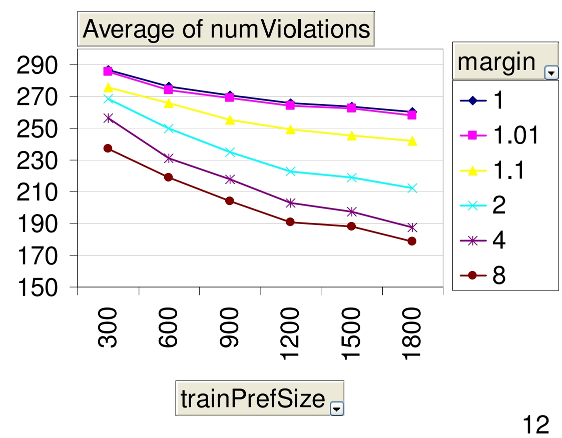




Learning rate and effect of margin

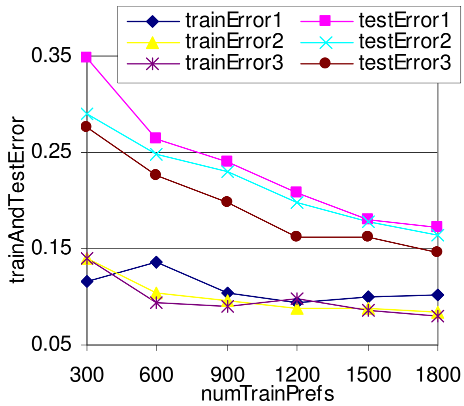
- Without $\|r\|_1 = 1$ constraint QP fails to learn from $\prec$
- Enforcing $\|r\|_1 = 1$ improves learning
- $|V| \times |V|$ inversion impractical, QP slow
  - Days vs. minutes
- Flow formulation with margin is much better
- Margin needs tuning, not scale-insensitive

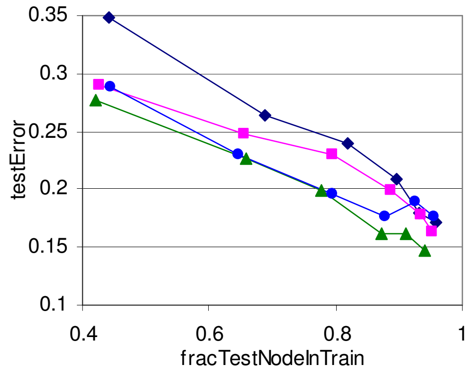




Effect of node overlap

- Nodes involved in $\prec$
  $V(\prec) = \{w : w \prec v \text{ or } u \prec w\}$
- What if $V(\prec_{train})$ and $V(\prec_{test})$ overlap?
- Note, $\prec_{train}$ and $\prec_{test}$ do not overlap!
- Well-motivated in relevance feedback settings
  - Train and test communities overlap
- Test error drops fast





2: The typed conductance formulation

- Edge e has type $t(e) \in \{1, \ldots T\}$
- Weight $w(i,j) = \beta(t(i,j))$, params $\beta(1), \ldots, \beta(T)$
- Matrix C is now a function of β, denoted $C(\beta)$
- Find β so that the p satisfying $p = C(\beta)p$ also satisfies $\prec$

$$\min_{\beta \geq 1} \beta'\beta \text{ subject to:}$$

$$p = C(\beta)p$$

$$p_i \leq p_j \text{ for all } i \prec j$$

Scaling all β preserves p, so we can demand all β(t)≥1

Both β and p are variables, leading to nasty quadratic equality constraints

Two approximations

- Breaking the quadratic constraints
  - Approximate $p \approx C(\beta)^H p^0$ where
    - p^0 is the initial Pagerank vector in power iteration
    - H is a finite horizon (or, stop at convergence)
- Design of a loss function
  - Training loss (not convex or differentiable)

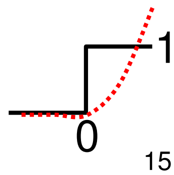
$$\sum_{u \prec v} \text{step}(p_u - p_v) = \sum_{u \prec v} \text{step}\left((C^H p^0)_u - (C^H p^0)_v\right)$$

  - Approximate using a smooth Huber loss

$$\text{huber}(y) = \begin{cases} 0, & y \le 0 \\ y^2 / W, & y \in (0, W] \\ y - W/2, & W < y \end{cases}$$

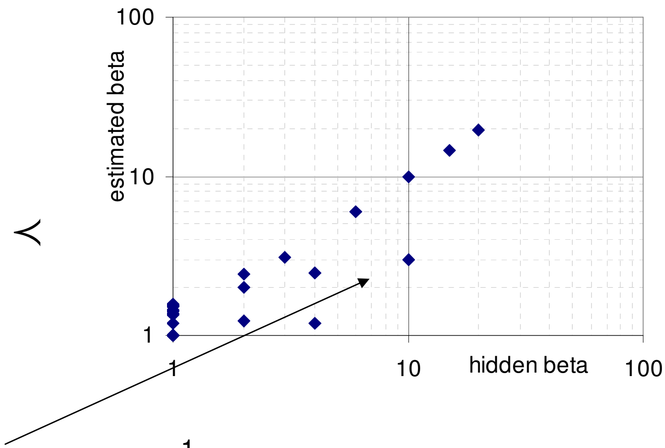
  - Gradient descent search for

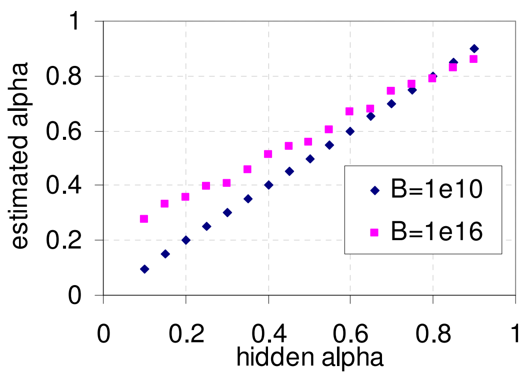
$$\min_{\beta \ge \mathbf{1}} \beta' \beta + B \sum_{u \prec v} \text{huber}\left((C^H p^0)_u - (C^H p^0)_v\right)$$



Discovering hidden edge weights

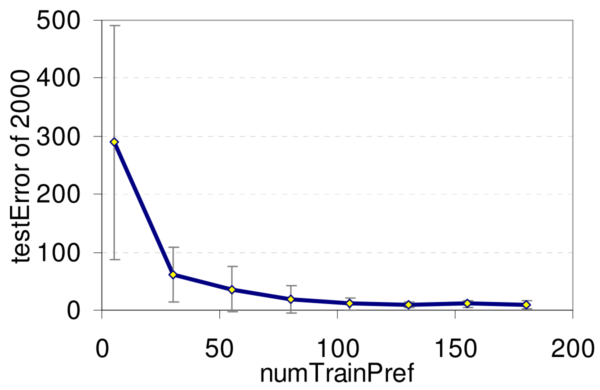
- Assign hidden edge weights to edge types
- Compute weighted Pagerank and sample $\prec$
- Can recover hidden weights fairly well
  - Penalty on β'β shrinks elements toward 0
  - Does not hurt prediction of $\prec_{test}$
- Can also find hidden α
- Time scales as $(|V| + |E|)^{1.34}$

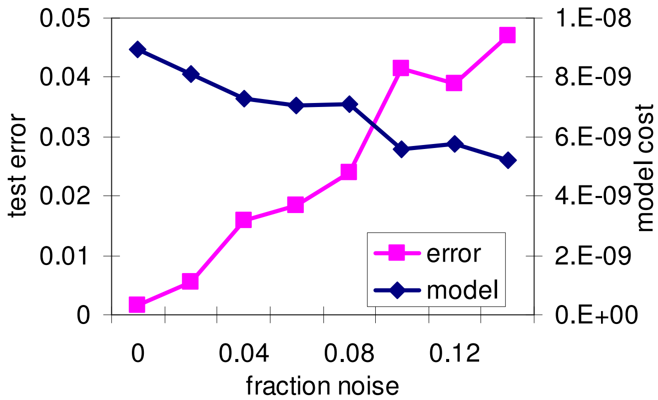




Learning rate and robustness

- 20000-node, 120000-edge graph
  - 100 pairwise training preferences enough to cut down test error to 11 out of 2000
  - Training and test preferences node-disjoint
- 20% random reversal of train pairs → 5% increase in test error
  - Model cost β'β reduces





Summary and ongoing work

- Learning to rank nodes in graph from pairwise preferences: surprisingly unexplored
- I.e., design edge conductance so that dominant eigenvector satisfies preferences
- Two design paradigms: constrained flows and typed edge conductance
- New algorithms to learn design parameters
- Integrating queries and node features into models and algorithms (in PKDD 2006)
- Rank-sensitive score learning