Learning Parameters in Entity Relationship Graphs from Ranking Preferences

Soumen Chakrabarti
Alekh Agarwal

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 $i \rightarrow j$ 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

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

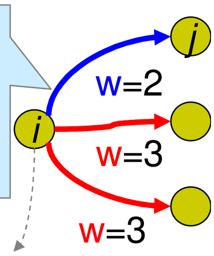
i is a dead-end $\sum_{j \in V} r_j = 1$

- Weighted Pagerank: $i \rightarrow j$ 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?

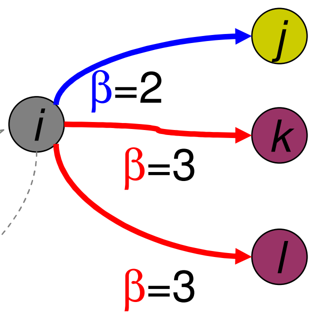
Inverse problem

- Traditionally: Given matrix C, find Pagerank p
- Clever design of C for various applications
  - Tweak or learn r – topic sensitive, personalized
  - Tweak $w(i,j)$ – Intelligent Surfer, ObjectRank
  - Model $p_{ij}=p_i C(j,i)$ as a flow (KDD 2006)
- Our problem: Given partial order $\prec_{\text{train}}$, find C (and thereby p) such that
  - p satisfies $p = Cp$ approximately
  - p satisfies $\prec_{\text{train}}$ and unseen $\prec_{\text{test}}$ well: $p_i \le p_j$ if $i \prec j$
  - Edges in C have weights determined by few types…

Ranking nodes in ER graphs

- Nodes have entity types: Person, Paper, Email, Company
- Edges have relation types: wrote, sent, cited, in-reply-to; edge e has type $t(e) \in \{1, 2, \ldots, T\}$
- Edge $i \rightarrow j$ of type t has *weight* $\beta(t)$ and *conductance* $C(i \rightarrow j)$…

Probability of following blue edge out of i is 2/(2+3+3)



Teleport?

$$C(j,i) = \frac{\beta(t(i,j))}{\sum_{i \to j} \beta(t(i,j))}$$

(For the moment ignore teleport)

Hard constraints

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

$$\min_{\beta \geq \mathbf{1}, p} \ \mathrm{ModelCost}(\beta)$$

subject to:

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

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

It might not be to possible to satisfy constraints exactly. So add slacks.

Remember C is a function of β:

$$C(j,i) = \frac{\beta(t(i,j))}{\sum_{i \to j} \beta(t(i,j))}$$

Model cost

- Parsimonious model is where all β(*t*)s are equal
- Penalize pairwise differences

$$\text{ModelCost}(\beta) = \sum_{t \neq t'} (\beta(t) - \beta(t'))^2$$

- If β is a solution, so is any multiple of β
- Objective should penalize large multiples automatically because, e.g.,

$$\text{ModelCost}(2\beta) > \text{ModelCost}(\beta)$$

Soft constraints with slacks

Like in SVM, balance model cost with data fit

$$\min_{\beta \geq \mathbf{1}, s \geq 0, p} \quad \text{ModelCost}(\beta) + B \sum_{i \prec j} \text{Loss}(s_{ij})$$

Design Loss to approximate training error

subject to:

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

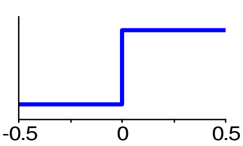
$$p_i \leq p_j + s_{ij} \quad \text{for all } i \prec j$$

Avoid quadratic constraints involving β and *p*

No margin!? –Because an arbitrary margin (say 1) may never be attainable by deviating from the parsimonious model and scaling β (unlike RankSVM)

Smooth loss approximation

$$\mathrm{Loss}(s_{ij}) = \begin{cases} 0 & s_{ij} \le 0 \\ 1 & s_{ij} > 0 \end{cases}$$

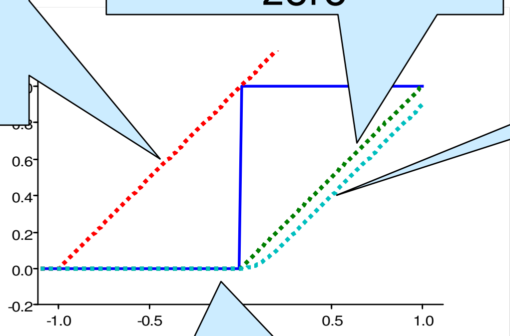


Neither convex nor differentiable

Traditional hinge upper bounds step but won't work for us

Shifted hinge not differentiable at zero

Huber is convex and differentiable



Need gradient of Loss

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

Important to stay at zero for arg ≤ 0

Avoiding quadratic constraints

- $p = Cp$ is usually solved by Power Iteration
  - Start with some p^0
  - Find $C^H p^0$ until increasing **horizon** H makes no difference
- Can get rid of s_{ij} now

$$\min_{\beta \ge \mathbf{1}} \sum_{t_1 \ne t_2} (\beta_{t_1} - \beta_{t_2})^2 + B \sum_{i \prec j} \mathrm{huber}\left((C^H p^0)_i - (C^H p^0)_j\right)$$

Remember C is a function of β

- To use a gradient descent method, need gradient wrt β

Gradient of Huber loss

$$\frac{\partial}{\partial \beta(t)} \text{huber}\left(p_i - p_j\right) =$$

$$\text{huber}'\left(p_i - p_j\right)\left(\frac{\partial p_i}{\partial \beta(t)} - \frac{\partial p_j}{\partial \beta(t)}\right)$$

$$p = Cp$$

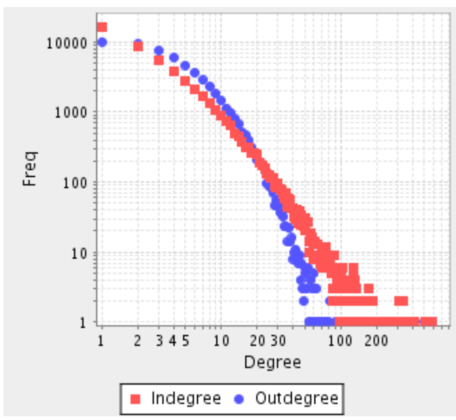
$$\frac{\partial p}{\partial \beta} = C\frac{\partial p}{\partial \beta} + p\frac{\partial C}{\partial \beta}$$

$$\frac{\partial p}{\partial \beta} = (\mathbb{I} - C)^{-1} p \frac{\partial C}{\partial \beta} \approx (\mathbb{I} - C)^{-1} C^H p^0 \frac{\partial C}{\partial \beta}$$

Polynomial ratios and products—surface not monotonic or unimodal, need some grid search

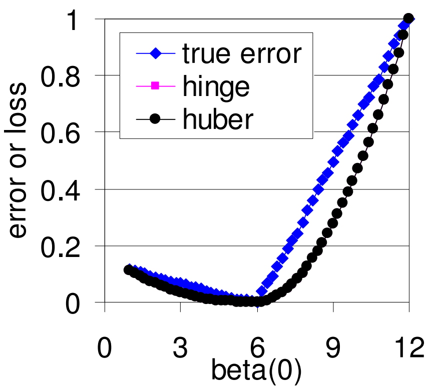
Data set preparation

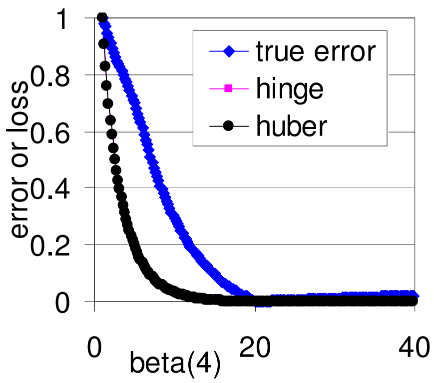
- No open benchmark for this task
  - No standardized comparison yet
  - We will make code and some data available
  - Synthetic G and $\prec$ can explore space thoroughly
- Generating graph G
  - RMAT (power-law degree, small dia, clustering)
  - Real DBLP+CiteSeer graph
- Generating preference $\prec$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

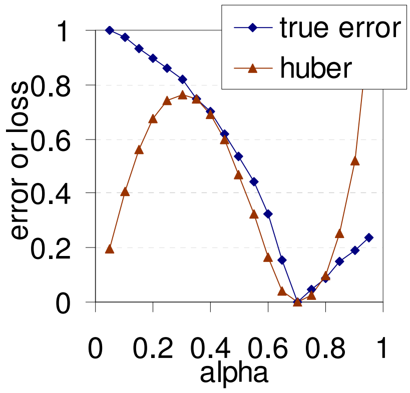


Is loss approximation ok?

- Less reliable than true error, but “in practice”…
- Approx loss wrt β tracks true loss reasonably
  - Min objective same if not always at same β
- α surface more tricky, need grid search
  - Applications use α ≈ 0.85 where tracking is ok

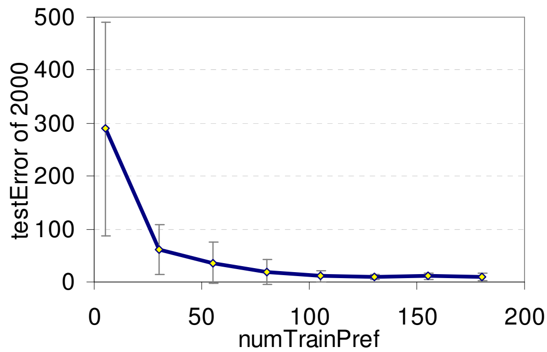


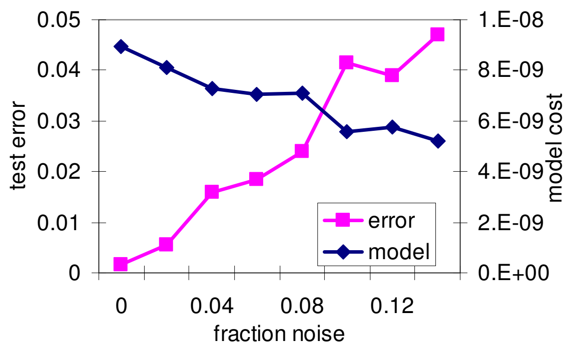




Learning rate and robustness

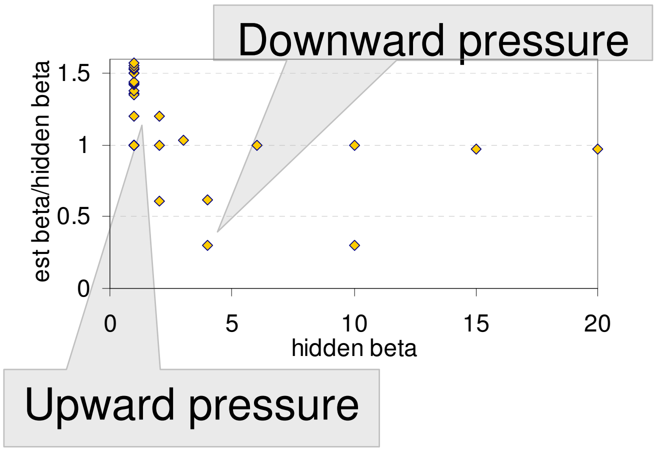
- 20000-node, 120000-edge graph
  - 100 pairwise training preferences enough to cut down test error to 11 out of 2000
  - Training and test preferences were made node-disjoint
- 20% random reversal of train pairs → 5% increase in test error
  - Model cost reduces

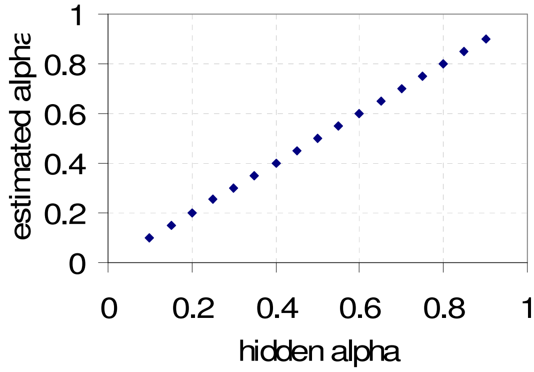




Discovering hidden edge weights

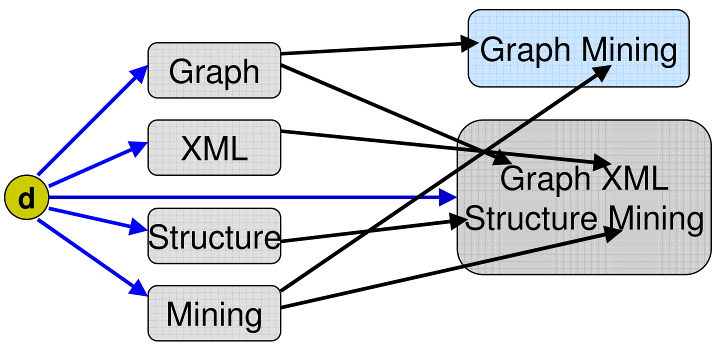
- Assign hidden edge weights to edge types
- Compute weighted Pagerank and sample $\prec$
- Can recover hidden weights fairly well
  - Penalty $\sum_{t,t'}(\beta_t - \beta_{t'})^2$ shrinks β values toward each other
  - Does not hurt error on $\prec_{test}$
- Can also find hidden α
- Time scales as $\left(|V|+|E|\right)^{1.34}$

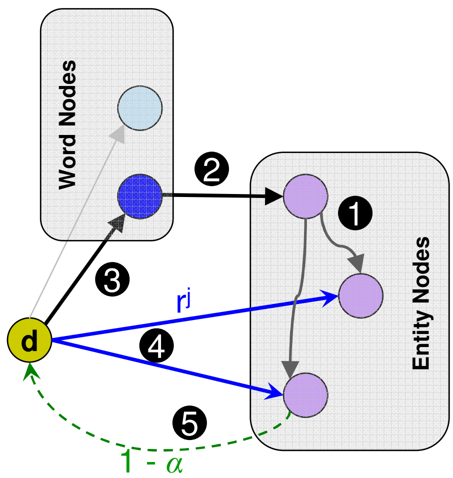




Integrating queries and text match

- Create node for each word, teleport through these
- Dummy connected only to query words
- Word node connected to entity nodes mentioning word
- Edge types 3 and 4 balance text relevance and link prestige





Balancing text match and prestige

- β(dummy→word) balances text match and prestige
- Small: classic papers win; large: relevance matters
- A versatile space of ranking functions

transaction serializability β(dummy→word)=1	Citations
Graph based algorithms for boolean function manipulation	506
Scheduling algorithms for multiprogramming in a hard real time environment	413
A method for obtaining digital signatures and public key cryptosystems	312
Rewrite systems	265
Tcl and the Tk toolkit	242
transaction serializability β(dummy→word)=10^6	Citations
On **serializability** of multidatabase **transactions** through forced local conflicts	38
Autonomous **transaction** execution with epsilon **serializability**	6
The **serializability** of concurrent database updates	104
Serializability a correctness criterion for global concurrency control	41
Using tickets to enforce the **serializability** of multidatabase **transactions**	12

Learning text+link conductance

- DBLP graph, Google Scholar preference pairs
  - Around 25% train and 35-40% test error
  - Two possible reasons:
  - They use larger graph with Web pages and Web links; we use only citations
  - Their proprietary ranking function is not in our class
- DBLP graph, synthetic pref in our pref class
  - Tune β(dummy→word) so ranking looks good to us
  - β(dummy→word) generally overestimated
  - Even so, reliable decrease in train and test errors

Summary and Ongoing Work

- Learning to rank nodes in graph from pairwise preferences: surprisingly unexplored
- Goal: design edge conductance so that dominant eigenvector satisfies preferences
- Integration of queries and node features with link-based learning formulation
- Optimization surface not benign but gradient descent is robust in practice
- Need more study on generalization and text feature integration

Thank You

For code and data please email

soumen@cse.iitb.ac.in