Learning to Rank in Vector Spaces and Social Networks
(WWW 2007 Tutorial)

Soumen Chakrabarti
IIT Bombay
http://www.cse.iitb.ac.in/~soumen
Motivation: Web search

- User query $q$, Web pages $\{v\}$
- $(q, v)$ can be represented with a rich feature vector
- Text match score with title, anchor text, headings, bold text, body text, . . . , of $v$ as a hypertext document
- Pagerank, topic-specific Pageranks, personalized Pageranks of $v$ as a node in the Web graph
- Estimated location of user, commercial intent, . . .
- Must we guess the relative importance of these features?
- How to combine these into a single scoring function on $(q, v)$ so as to induce a ranking on $\{v\}$?
Motivation: Ad and link placement

- Here, the “query” is the surfer’s contextual information
- More noisy than queries, which are noisy enough!
- Plus page and site contents
- A response is an ad to place, or a link to insert
- Must rank and select from a large pool of available ads or links
- (In this tutorial we will ignore issues of bidding and visibility pricing)
Motivation: Desktop search

- The Web has only a few kinds of hyperlinks: same-host subdirectory, same-host superdirectory, same-host across-path, different-host same-domain, different-domain etc.
- Often differentiated by hardwired policy, e.g., HITS completely ignores same-host links
- Entity-relationship (ER) graphs are richer
- E.g. A personal information management (PIM) system has many node/entity types (person, organization, email, paper, conference, phone number) and edge/relation types (works-for, sent, received, authored, published-in)
- Ranking needs to exploit the richer type system
- Don’t want to guess the relative importance of edge types (may be dependent on queries)
Desktop search example
Relevance feedback

- Relevance feedback is well-explored in traditional IR
- User-assisted local modification of ranking function for vector-space models
- How to extend these to richer data representations that incorporate entities, relationship links, entity and relation types?
- Surprisingly unexplored area
Tutorial outline: Preliminaries

- Training and evaluation scenarios
- Measurements to evaluate quality of ranking
  - Label mismatch loss functions for ordinal regression
  - Preference pair violations
  - Area under (true positive, false positive) curve
  - Average precision
  - Rank-discounted reward for relevance
  - Rank correlations
- What’s useful vs. what’s easy to learn
Tutorial outline: Ranking in vector spaces

Instance $v$ is represented by a feature vector $x_v \in \mathbb{R}^d$

- Discriminative max-margin ranking (RankSVM)
- Linear-time max-margin approximation
- Probabilistic ranking in vector spaces (RankNet)
- Sensitivity to absolute rank and cost of poor rankings
Tutorial outline: Ranking in graphs

Instance $v$ is a node in a graph $G = (V, E)$

- The graph-Laplacian approach
  - Assign scores to nodes to induce ranking
  - $G$ imposes a smoothness constraint on node scores
  - Large difference between neighboring node scores penalized

- The Markov walk approach
  - Random surfer, Pagerank and variants; by far most popular way to use graphs for scoring nodes
  - Walks constrained by preferences
  - How to incorporate node, edge types and query words

- Surprising connections between the two approaches
Tutorial outline: Stability and generalization

- Some notes on score- vs. rank-stability
- Stability and generalization of max-margin ranking in vector spaces
- Stability and generalization of graph-Laplacian ranking
- Stability and generalization of Markov walk based ranking
Preliminaries

- Motivation
- Training and evaluation setup
- Performance measures

Ranking in vector spaces

- Discriminative, max-margin algorithms
- Probabilistic models, gradient-descent algorithms

Ranking nodes in graphs

- Roughness penalty using graph Laplacian
- Constrained network flows

Stability and generalization

- Admissibility and stability
- Ranking loss and generalization bounds
Preliminaries

- Motivation
- Training and evaluation setup
- Performance measures

Ranking in vector spaces

- Discriminative, max-margin algorithms
- Probabilistic models, gradient-descent algorithms

Ranking nodes in graphs

- Roughness penalty using graph Laplacian
- Constrained network flows

Stability and generalization

- Admissibility and stability
- Ranking loss and generalization bounds
Preliminaries

▶ Motivation
▶ Training and evaluation setup
▶ Performance measures

Ranking in vector spaces

▶ Discriminative, max-margin algorithms
▶ Probabilistic models, gradient-descent algorithms

Ranking nodes in graphs

▶ Roughness penalty using graph Laplacian
▶ Constrained network flows

Stability and generalization

▶ Admissibility and stability
▶ Ranking loss and generalization bounds
Preliminaries

▶ Motivation
▶ Training and evaluation setup
▶ Performance measures

Ranking in vector spaces

▶ Discriminative, max-margin algorithms
▶ Probabilistic models, gradient-descent algorithms

Ranking nodes in graphs

▶ Roughness penalty using graph Laplacian
▶ Constrained network flows

Stability and generalization

▶ Admissibility and stability
▶ Ranking loss and generalization bounds
Forms of training input

Regression: For each entity \( x \), an absolute real score \( y \) (unrealistic to expect users to assign absolute scores)

Ordinal regression: For each entity \( x \), a score \( y \) from a discrete, ordered domain, such as a \( r \)-point scale (implemented in many sites like Amazon.COM)

Bipartite ranking: Ordinal regression with \( r = 2 \)

Pairwise preferences: A (possibly inconsistent) partial order between entities, expressed as a collection of “\( u \prec v \)” meaning “\( u \) is less preferred than \( v \)” (low cognitive load on users, can be captured from click-logs and eye-tracking data)

Complete rank order: A total order on the entities but no scores (highly impractical for large entity sets)

Prefix of rank order: A total order on the top-\( k \) entities, meaning that all the other entities are worse (iffy)
Evaluation of ranking algorithms

**Error on score vectors:** In case of standard regression, if $\hat{f}$ is the score assigned by the algorithm and $f$ is the “true score”, measure $\|\hat{f} - f\|_1$ or $\|\hat{f} - f\|_2$.

**Ordinal reversals:** If $y_u > y_v$ and $\hat{f}(u) < \hat{f}(v)$ then $u$ and $v$ have been reversed. Count the number of reversed pairs.

**Precision at $k$:** For a specific query $q$, let $T_k^q$ and $\hat{T}_k^q$ be the top-$k$ sets as per $f$ and $\hat{f}$ scores. The precision at $k$ for query $q$ is defined as $|T_k^q \cap \hat{T}_k^q|/k \in [0, 1]$. Average over $q$. 
Evaluation of ranking algorithms II

Relative aggregated goodness (RAG): For a specific query $q$,

$$RAG(k, q) = \frac{\sum_{v \in \hat{T}_k^q} f(v)}{\sum_{v \in T_k^q} f(v)} \in [0, 1]$$

Note that $\hat{f}$ is not used! Average over $q$.

Mean reciprocal rank (MRR): For each query there is one or more correct responses. Suppose for specified query $q$, the first rank at which a correct response occurs is $R(q)$. Then MRR is

$$\frac{1}{|Q|} \sum_{q \in Q} \frac{1}{R(q)}$$
Evaluation of ranking algorithms III

Normalized discounted cumulative gain (NDCG): For a specific query \( q \),

\[
N_q \sum_{i=1}^{k} \frac{2^{\text{rating}(i)} - 1}{\log(1 + i)}
\]

Here \( N_q \) is a normalization factor so that a perfect ordering gets NDCG score of 1 for each query, \( k \) is the number of top responses considered, and \( \text{rating}(i) \) is the evaluator rating for the item returned at position \( i \).

Pair preference violation: If \( u \prec v \) and \( \hat{f}(u) > \hat{f}(v) \) a pair has been violated. Count the number of pair violations.
Evaluation of ranking algorithms IV

Rank correlation: Order entities by decreasing $f(u)$ and compute a rank correlation with the ground truth ranking. Impractical if a full ground truth ranking is expected.

Prefix rank correlation: Let exact and approximate scores be denoted by $S^k_q(v)$ and $\hat{S}^k_q(v)$ respectively for items $v$, where the scores are forced to zero if $v \notin T^q_k$ and $v \notin \hat{T}^q_k$. A node pair $v, w \in T^q_k \cup \hat{T}^q_k$ is concordant if $(S^k_q(v) - S^k_q(w))(\hat{S}^k_q(v) - \hat{S}^k_q(w))$ is strictly positive, and discordant if it is strictly negative. It is an exact-tie if $S^k_q(v) = S^k_q(w)$, and is an approximate tie if $\hat{S}^k_q(v) = \hat{S}^k_q(w)$. If there are $c,$
Evaluation of ranking algorithms V

d, e and a such pairs respectively, and m pairs overall in $T^q_k \cup \hat{T}^q_k$, then Kendall’s $\tau$ is defined as

$$\tau(k, q) = \frac{c - d}{\sqrt{(m - e)(m - a)}} \in [-1, 1].$$

Average over $q$.

- Theoretically sound and scalable rank learning techniques typically address simpler evaluation objectives
- Designing learning algorithms for the more complicated, non-additive evaluation objectives is very challenging
- Sometimes, we are lucky enough to establish a connection between the two classes of objectives
Bipartite ranking and area under curve (AUC)

- In bipartite ranking labeled data is of the form \((x, y)\) where \(y \in \{-1, 1\}\)
- Algorithm orders instances by decreasing \(f(x)\)
- For \(i = 0, 1, \ldots, n\)
  - Assign label +1 to the first \(i\) instances
  - Assign label −1 to the rest
  - True positive rate at \(i\)
    \[
    \frac{\text{number of positive instances labeled positive}}{\text{number of positive instances}}
    \]
  - False positive rate at \(i\)
    \[
    \frac{\text{number of negative instances labeled positive}}{\text{number of negative instances}}
    \]
- Plot \(x = \text{TruePosRate}, y = \text{FalsePosRate}\)
- Measure area under curve
AUC and pair preference violations

- $m$ positive and $n$ negative examples
- Area under curve (AUC) using $f$ for ranking can also be written as

$$\hat{A}(f, T) = \frac{1}{mn} \sum_{i:y_i=+1} \sum_{j:y_j=-1} \left( \mathbb{I}[f(i) > f(j)] + \frac{1}{2} \mathbb{I}[f(i) = f(j)] \right)$$

where $T$ is the training set

- The important part is the fraction of satisfied pair preferences between positive and negative instances
- Optimizing AUC is different from optimizing 0/1 error

<table>
<thead>
<tr>
<th>$y_i$</th>
<th>$-1$</th>
<th>$-1$</th>
<th>$-1$</th>
<th>$-1$</th>
<th>$+1$</th>
<th>$+1$</th>
<th>$+1$</th>
<th>$+1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_1(x_i)$</td>
<td>$-2$</td>
<td>$-1$</td>
<td>$3$</td>
<td>$4$</td>
<td>$1$</td>
<td>$2$</td>
<td>$5$</td>
<td>$6$</td>
</tr>
<tr>
<td>$f_2(x_i)$</td>
<td>$-2$</td>
<td>$-1$</td>
<td>$5$</td>
<td>$6$</td>
<td>$1$</td>
<td>$2$</td>
<td>$3$</td>
<td>$4$</td>
</tr>
</tbody>
</table>
Concordant and discordant instance pairs

- Suppose there are $R$ relevant documents in response to a query.
- The search engine creates a ranking $r_{\text{engine}}$ which lists them at ranks $p_1 < p_2 < \cdots < p_R$.
- An ideal system creates a ranking $r_{\text{ideal}}$ that lists all relevant documents before any irrelevant document.
- But keeps the relative ordering within the relevant and irrelevant subsets the same.

\[
\begin{align*}
    r_{\text{engine}} &= d_1^+, d_2^-, d_3^+, d_4^+, d_5^-, d_6^-, d_7^+, d_8^- \\
    r_{\text{ideal}} &= d_1^+, d_3^+, d_4^+, d_7^+; d_2^-, d_5^-, d_6^-, d_8^-
\end{align*}
\]

- Let there be $Q$ discordant pairs in $r_{\text{engine}}$ compared to $r_{\text{ideal}}$. 
Relating ranks and discordant pairs

- Account for $Q$ as follows: First consider the relevant document at position $p_1$ in $r_{\text{engine}}$. Because it has been pushed out from position 1 to position $p_1$, the number of inversions introduced is $p_1 - 1$.

- For the document at position $p_2$ in $r_{\text{engine}}$, the number of inversions introduced is $p_2 - 1 - 1$, the last “−1” thanks to having the first relevant document ahead of it.

- Summing up, we get

\[
\sum_{i=1}^{R} p_i - 1 - (i - 1) = Q, \quad \text{or}
\]

\[
\sum_{i=1}^{R} p_i = Q + \sum_{i=1}^{R} i = Q + \frac{R(R + 1)}{2} = Q + \binom{R+1}{2}.
\]
Average precision

- The **average precision** of \( r_{\text{engine}} \) wrt \( r_{\text{ideal}} \) is defined as

\[
\text{AvgPrec}(r_{\text{engine}}, r_{\text{ideal}}) = \frac{1}{R} \sum_{i=1}^{R} \frac{i}{p_i}
\]

- Like NDCG, average precision rewards the search engine if all \( p_i \) are as small as possible

- Intuitively, if \( Q \) is small, \( \text{AvgPrec}(r_{\text{engine}}, r_{\text{ideal}}) \) should be large.

- This can be formalized by framing an optimization problem that gives a lower bound to \( \text{AvgPrec}(r_{\text{engine}}, r_{\text{ideal}}) \) given a fixed \( Q \) (and \( R \))
Bounding average precision given $Q$

- To lower bound average precision, optimize:

$$\min_{p_1, \ldots, p_R} \frac{1}{R} \sum_{i=1}^{R} \frac{i}{p_i} \quad \text{such that}$$

$$p_1 + \cdots + p_R = Q + \binom{R+1}{2}$$

$$1 \leq p_1 < p_2 < \cdots < p_R$$

$p_1, \ldots, p_R$ are positive integers

- Relaxing the last two constraints can only decrease the optimal objective, so we still get a lower bound.

- The relaxed optimization is also convex because $1/p_i$ is convex in $p_i$, as far as $p_i$ is concerned the numerator $i$ is a “constant”, and sum of convex functions is convex.
Solving the relaxed optimization

Using the Lagrangian method, we get

\[
\mathcal{L}(p_1, \ldots, p_R; \lambda) = \frac{1}{R} \sum_{i=1}^{R} i + \lambda \left( \sum_{i=1}^{R} p_i - Q - \left( \frac{R + 1}{2} \right) \right)
\]

\[
\therefore \frac{\partial \mathcal{L}}{\partial p_i} = -\frac{i}{R p_i^2} + \lambda \overset{\text{set}}{=} 0 \quad \text{to get} \quad p_i^* = \sqrt{\frac{i}{R \lambda}}.
\]

Replace back in the Lagrangian, set the derivative wrt \( \lambda \) to zero, and again substitute in the Lagrangian to get the optimal objective (in the relaxed optimization) as

\[
\text{AvgPrec}(r_{\text{engine}}, r_{\text{ideal}}) \geq \frac{\left( \sum_{i=1}^{R} \sqrt{i} \right)^2}{R \left( Q + \left( \frac{R + 1}{2} \right) \right)}
\]

\( Q \) and the lower bound on average precision are inversely related, which makes sense.
Preliminaries

- Motivation
- Training and evaluation setup
- Performance measures

Ranking in vector spaces

- Discriminative, max-margin algorithms
- Probabilistic models, gradient-descent algorithms

Ranking nodes in graphs

- Roughness penalty using graph Laplacian
- Constrained network flows

Stability and generalization

- Admissibility and stability
- Ranking loss and generalization bounds
Ordinal regression

- Items assigned *ratings* on a discrete $r$-point scale, e.g., items for sale at Amazon.COM
- The task is to regress instance $x \in \mathcal{X}$ to label $y \in \mathcal{Y}$ where $\mathcal{Y}$ is typically small
- Bipartite ranking is a special case with $|\mathcal{Y}| = 2$ so we can write $\mathcal{Y} = \{-1, +1\}$

Ordinal regression is different from plain classification because

- Unlike in classification, where labels in $\mathcal{Y}$ are *incomparable*, here they have a total order imposed on them. (In standard regression, $\mathcal{Y} = \mathbb{R}$.)
- The accuracy measures of practical interest here are different from those (0/1 error, recall, precision, $F_1$) used in classification.
Max-margin ordinal regression

- Apart from $\beta$, we will optimize over $r - 1$ thresholds

$$-\infty = b_0 \leq b_1 \leq b_2 \leq \cdots \leq b_{r-2} \leq b_{r-1} \leq b_r = +\infty$$

- Let $j \in \{1, \ldots, r\}$ index score levels, and the $i$th instance in the $j$ level be denoted $x^j_i$

- We wish to pick $\beta$ such that, for any $x^j_i$,

$$b_{j-1} < \beta^\top x^j_i < b_j$$

- Using the max-margin principle, we will insist that

$$b_{j-1} + 1 < \beta^\top x^j_i < b_j - 1$$
Max-margin ordinal regression II

- To avoid infeasibility, introduce lower slacks $s^i_j \geq 0$ and upper slacks $\bar{s}^i_j \geq 0$, and relax the above inequalities to

$$b_{j-1} + 1 - s^i_j \leq \beta^\top x^i_j \leq b_j - 1 + \bar{s}^i_j$$
Max-margin ordinal regression III

- The objective to minimize is modified to

\[
\min_{\beta, b, s \geq 0, \bar{s} \geq 0} \frac{1}{2} \beta^T \beta + B \sum_{j, i} (s_i^j + \bar{s}_i^j),
\]

- Yet another quadratic program with linear inequalities

- Training time scales roughly as \(n^{2.18-2.33}\) where \(n\) is the number of training instances

- More accurate than replacing ordinal regression with plain regression
Ranking to satisfy preference pairs

- Suppose $x \in X$ are instances and $\phi : X \rightarrow \mathbb{R}^d$ a feature vector generator
- E.g., $x$ may be a document and $\phi$ maps $x$ to the “vector space model” with one axis for each word
- The score of instance $x$ is $\beta^T \phi(x)$ where $\beta \in \mathbb{R}^d$ is a weight vector
- For simplicity of notation assume $x$ is already a feature vector and drop $\phi$
- We wish to learn $\beta$ from training data $\prec$: “$i \prec j$” means the score of $x_i$ should be less than the score of $x_j$, i.e.,

$$\beta^T x_i \leq \beta^T x_j$$
Soft constraints

- In practice, there may be no feasible $\beta$ satisfying all preferences $\prec$.
- For constraint $i \prec j$, introduce slack variable $s_{ij} \geq 0$

$$\beta^T x_i \leq \beta^T x_j + s_{ij}$$

- Charge a penalty for using $s_{ij} > 0$

$$\min_{s_{ij} \geq 0; \beta \prec} \frac{1}{\beta} \left| \sum_{i \prec j} s_{ij} \right| \text{ subject to } \beta^T x_i \leq \beta^T x_j + s_{ij} \text{ for all } i \prec j$$
A max-margin formulation

▶ Achieve “confident” separation of loser and winner:

\[ \beta^\top x_i + 1 \leq \beta^\top x_j + s_{ij} \]

▶ Problem: Can achieve this by scaling \( \beta \) arbitrarily; must be prevented by penalizing \( \|\beta\| \)

\[
\begin{align*}
\min_{s_{ij} \geq 0; \beta} & \quad \frac{1}{2} \beta^\top \beta + \frac{B}{|\prec|} \sum_{i \prec j} s_{ij} \\
\text{subject to} & \quad \beta^\top x_i + 1 \leq \beta^\top x_j + s_{ij} \quad \text{for all } i \prec j
\end{align*}
\]

▶ \( B \) is a magic parameter that balances violations against model strength
Solving the optimization

\[ \beta^T x_i + 1 \leq \beta^T x_j + s_{ij} \quad \text{and} \quad s_{ij} \geq 0 \quad \text{together mean} \]

\[ s_{ij} = \max\{0, \beta^T x_i - \beta^T x_j + 1\} \quad \text{("hinge loss")} \]

\[ \text{The optimization can be rewritten without using } s_{ij} \]

\[ \min_{\beta} \frac{1}{2} \beta^T \beta + \frac{B}{|\prec|} \sum_{i \prec j} \max\{0, \beta^T x_i - \beta^T x_j + 1\} \]

\[ \max\{0, t\} \quad \text{can be approximated by a number of smooth functions} \]

\[ e^t \quad \text{growth at } t > 0 \, \text{too severe} \]

\[ \log(1 + e^t) \quad \text{much better, asymptotes to } y = 0 \, \text{as} \]

\[ t \to -\infty \quad \text{and to } y = t \, \text{as} \, t \to \infty \]
Approximating with a smooth objective

- Simple unconstrained optimization, can be solved by Newton method

$$\min_{\beta \in \mathbb{R}^d} \frac{1}{2} \beta^T \beta + \frac{B}{|\prec|} \sum_{i \prec j} \log(1 + \exp(\beta^T x_i - \beta^T x_j + 1))$$

- If $\beta^T x_i - \beta^T x_j + 1 \ll 0$, i.e., $\beta^T x_i \ll \beta^T x_j$, then pay little penalty
- If $\beta^T x_i - \beta^T x_j + 1 \gg 0$, i.e., $\beta^T x_i \gg \beta^T x_j$, then pay large penalty
Performance issues

- Common SVM implementations will take time almost quadratic in the number of training pairs.
- Consider a TREC-style relevance judgment: for each query, we are given, say, 10 relevant and (implicitly) $1M - 10$ irrelevant documents.
- Don’t really need to train RankSVM with $10M x_i \prec x_j$ pairs.
- E.g., if $\beta^T x_0 \leq \beta^T x_1$ and $\beta^T x_0 \leq \beta^T x_2$, then $\beta^T x_0 \leq \lambda \beta^T x_1 + (1 - \lambda) \beta^T x_2$ for $\lambda \in [0, 1]$.
- Cannot, in general, say ahead of time which preferences will be redundant.
A linear-time RankSVM approximation

- The primal optimization can be reformulated as

\[
\min_{\beta, s \geq 0} \frac{1}{2} \beta^\top \beta + Bs \quad \text{such that} \quad \forall \vec{c} \in \{0, 1\}^{|\mathcal{X}|} : \frac{1}{|\mathcal{X}|} \beta^\top \sum_{u \prec v} c_{uv} (x_v - x_u) \geq \frac{1}{|\mathcal{X}|} \sum_{u \prec v} c_{uv} - s
\]

- Only one slack variable \( s \), but \( 2^{|\mathcal{X}|} \) primal constraints and corresponding \( 2^{|\mathcal{X}|} \) dual variables

- (But if most primal constraints are redundant, most dual variables will be inactive, i.e., 0)

- Compare with

\[
\min_{\beta, \{s_{uv} \geq 0 : u \prec v\}} \frac{1}{2} \beta^\top \beta + \frac{B}{|\mathcal{X}|} \sum_{u \prec v} s_{uv} \quad \text{(RankSVM1)}
\]

such that \( \forall u \prec v : \beta^\top x_u + 1 \leq \beta^\top x_v + s_{uv} \)
Correctness

Any solution to (RankSVM2) corresponds to a solution to (RankSVM1), and vice versa

- Fix a $\beta_0$ in (RankSVM1)
- For optimality, must pick $s_{uv}^* = \max\{0, 1 + \beta_0^T x_u - \beta_0^T x_v\}$
- Fix the same $\beta_0$ for (RankSVM2)
- For optimality, must pick

$$s^* = \min_{\tilde{c} \in \{0, 1\}} \left\{ \frac{1}{|\prec|} \sum_{u \prec v} c_{uv} \left(1 + \beta_0^T x_u - \beta_0^T x_v\right) \right\}$$

- Pick $\tilde{c}$ element-wise: $c_{uv}^* = \left[1 + \beta_0^T x_u - \beta_0^T x_v \leq 0\right]

- Can verify that objectives of (RankSVM1) and (RankSVM2) will be equal using $\beta_0, \{s_{uv}^*\}, s^*, \{c_{uv}^*\}$
Cutting plane method: General recipe

- Primal: \( \min_x f(x) \) subject to \( g(x) \leq 0 \) (\( g \) is a vector-valued function)

- Dual:

\[
\max_{z,u} \quad z \\
\text{subject to} \quad z \leq f(x) + u^\top g(x) \quad \forall x \\
u \geq 0
\]

- “\( \forall x \)” is generally infinite

- Let \( z_k, u_k \) be a solution

- Find \( \min_x f(x) + u_k^\top g(x) \), let solution be \( x_k \)

- If \( z_k \leq f(x_k) + u_k^\top g(x_k) \), terminate

- Otherwise add \( k \)th constraint \( z \leq f(x_k) + u_k^\top g(x_k) \)

- To approximate and terminate faster, continue only if

\[ z_k > f(x_k) + u_k^\top g(x_k) + \epsilon \]
Gradual dual variable inclusion

- Instead of all \( \{0, 1\}^{|\mathcal{H}|} \), start with \( \mathcal{W} \subset \{0, 1\}^{|\mathcal{H}|} \), typically \( \mathcal{W} = \emptyset \)
- Solve (RankSVM2) with \( \mathcal{W} \) instead of \( \{0, 1\}^{|\mathcal{H}|} \) to get the current \( \beta_0, s^* \)
- Look for a violator \( c^* \) such that
  
  \[
  \frac{1}{|\prec|} \sum_{u \prec v} c^*_{uv} (x_v - x_u) < \frac{1}{|\prec|} \sum_{u \prec v} c^*_{uv} - s^* - \epsilon
  \]

- If no such \( c^* \) found, exit with an objective that is at most the optimal objective plus \( \epsilon \)
- Otherwise add \( c^* \) to \( \mathcal{W} \) and repeat
- For fixed (constant) \( \epsilon, B \) and \( \max \|x_v\|_2 \), the number of inclusions into \( \mathcal{W} \) before no further \( c^* \) is found is constant
- Each loop above can be implemented in \( O(n \log n) \) vector operations in \( \mathbb{R}^d \) where all \( x_v \in \mathbb{R}^d \)
Linear-time (RankSVM2) performance

- Almost linear scaling in practice too
- Dramatic improvement over (RankSVM1)
- (RankSVM1) scales roughly as $n^{3.4}$ (not shown)
A probabilistic interpretation of “ranking loss”

- Apart from $x_i \prec x_j$, trainer gives target probability $\bar{p}_{ij}$ with which trained system should rank $i$ worse than $j$
- The score of $x_i$ is $f(x_i) \in \mathbb{R}$; $f(x_i)$ induces a ranking on \{x_i\}
- The modeled posterior $p_{ij}$ is assumed to have a familiar log-linear form

$$p_{ij} = \frac{\exp(f(x_j) - f(x_i))}{1 + \exp(f(x_j) - f(x_i))}$$

- If $f(x_j) \gg f(x_i)$, $p_{ij} \to 1$; if $f(x_j) \ll f(x_i)$, $p_{ij} \to 0$
- Goal is to design $f$ to minimize divergence between trainer-specified $\bar{p}$ and modeled $p$, e.g.,

$$\ell(\bar{p}_{ij}, p_{ij}) = -\bar{p}_{ij} \log p_{ij} - (1 - \bar{p}_{ij}) \log(1 - p_{ij})$$
Consistency requirements on $\bar{p}_{ij}$

- Trainer cannot assign $\bar{p}_{ij}$ arbitrarily
- $\bar{p}_{ij}$ must be consistent with some ideal node-scoring function $\bar{f}$ such that

$$
\bar{p}_{ij} = \frac{\exp(\bar{f}(x_j) - \bar{f}(x_i))}{1 + \exp(\bar{f}(x_j) - \bar{f}(x_i))}
$$

- Using above, can show that

$$
\bar{p}_{ik} = \frac{\bar{p}_{ij} \bar{p}_{jk}}{1 + 2\bar{p}_{ij} \bar{p}_{jk} - \bar{p}_{ij} - \bar{p}_{jk}}
$$

- Consider $\bar{p}_{ik}$ if $\bar{p}_{ij} = \bar{p}_{kj} = p$, in particular $p = 0, .5, 1$
- Perfect uncertainty and perfect certainty propagate
Fitting $f$ using gradient descent

- Model $f(x_i) = \beta^T x_i$ for simplicity
- During training we are given ($i < j$ with) a target $\bar{p}_{ij}$
- We want to fit $\beta$ so that

$$\bar{p}_{ij} = \frac{\exp(\beta^T x_i - \beta^T x_j)}{1 + \exp(\beta^T x_i - \beta^T x_j)}$$

- We can cast this as, say,

$$\min_{\beta} \sum_{i < j} \left( \bar{p}_{ij} - \frac{\exp(\beta^T x_i - \beta^T x_j)}{1 + \exp(\beta^T x_i - \beta^T x_j)} \right)^2$$

and use gradient descent

- Or we can use more complex forms of $f(x)$, like a neural network
RankBoost

- Given partial orders with preference strengths $\phi(i, j) \geq 0$: if positive, $i \succ j$, otherwise impartial
- Input pair distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{X}$
- **Weak learner** indexed by $t$ gets input pairs as per a distribution $\mathcal{D}_t$ and outputs a weak ranking $h_t : \mathcal{X} \rightarrow \mathbb{R}$
- Initialize $\mathcal{D}_1 = \mathcal{D}$
- For $t = 1, \ldots, T$
  - Train $t$th weak learner using $\mathcal{D}_t$
  - Get weak ranking $h_t : \mathcal{X} \rightarrow \mathbb{R}$
  - Choose $\alpha_t \in \mathbb{R}$
  - Update
    
    $$\mathcal{D}_{t+1}(x_i, x_j) \propto \mathcal{D}_t(x_i, x_j) \exp(\alpha_t(h_t(x_i) - h_t(x_j)))$$
- **Final scoring function** $H(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$
Some properties of RankBoost

- The ranking loss $R_D(H)$ is defined as

$$
\sum_{x_i, x_j} D(x_i, x_j) \mathbb{I}[H(x_i) \leq H(x_j)] = \Pr_{(x_i, x_j) \sim D} [H(x_i) \leq H(x_j)]
$$

- $R_D(H) \leq \prod_{t=1}^{T} Z_t$

- By suitably choosing $\alpha_t$ we can ensure $Z_t \leq 1$

- E.g., if $h: \mathcal{X} \to \{0, 1\}$, we can minimize $Z_t$ analytically:
  - For $b \in \{-1, 0, +1\}$, define
    $$
    W_b = \sum_{x_i, x_j} D(x_i, x_j) \mathbb{I}[h(x_i) - h(x_j)]
    $$
  - $Z_t$ is minimized when $\alpha = \frac{1}{2} \ln(W_{-1}/W_{+1})$

""
Preliminaries

▶ Motivation
▶ Training and evaluation setup
▶ Performance measures

Ranking in vector spaces

▶ Discriminative, max-margin algorithms
▶ Probabilistic models, gradient-descent algorithms

Ranking nodes in graphs

▶ Roughness penalty using graph Laplacian
▶ Constrained network flows

Stability and generalization

▶ Admissibility and stability
▶ Ranking loss and generalization bounds
Undirected graph Laplacian

- Simple unweighted undirected graph $G = (V, E)$ with $|V| = n$, $|E| = m$, no self-loops or parallel edges
- Node-node adjacency matrix $A \in \{0, 1\}^{n \times n}$ with $A(u, v) = 1$ if $(u, v) \in E$ and 0 otherwise
- Node-edge incidence matrix $N \in \{-1, 0, 1\}^{n \times m}$ with

\[
N(v, e) = \begin{cases} 
-1 & \text{if } e = (v, \cdot) \\
1 & \text{if } e = (\cdot, v) \\
0 & \text{if } v \text{ is not either endpoint of } e
\end{cases}
\]

- Consider the graph Laplacian matrix $L_G = NN^\top \in \mathbb{R}^{n \times n}$
- $(NN^\top)(u, u)$ is the degree of node $u$
- $(NN^\top)(u, v)$ is $-1$ if $(u, v) \in E$, 0 otherwise
- Let $D$ be a diagonal matrix with $D(u, u) = \text{degree of } u$
- $NN^\top = D - A$ is a symmetric positive semidefinite matrix
Extending to weighted undirected graphs

- $A$ is not boolean; $A(u, v)$ is the weight of edge $(u, v)$ if any, 0 otherwise

- Modify $N$ to

  $N(v, e) = \begin{cases} 
  -\sqrt{A(e)} & \text{if } e = (v, \cdot) \\
  \sqrt{A(e)} & \text{if } e = (\cdot, v) \\
  0 & \text{if } v \text{ is not either endpoint of } e
  \end{cases}$

- Modify $L_G$ to

  $L_G(u, v) = \begin{cases} 
  \sum_w A(u, w), & \text{if } u = v \\
  -A(u, v), & \text{if } u \neq v, (u, v) \in E \\
  0 & \text{otherwise}
  \end{cases}$

- Modify “degree” matrix $D$ to $D(u, u) = \sum_v A(u, v)$

- Still have $L_G = NN^T = D - A$
Laplacian and node score smoothness

- For any vector \( x \in \mathbb{R}^n \),

\[
x^\top L x = \sum_{(u,v) \in E} A(u, v) (x_u - x_v)^2
\]

- \( x^\top L x \) penalizes node scores that are very different across “heavy” edges
- If \( u \prec v \), we want \( x_u + 1 \leq x_v \)
- Therefore define the ranking loss of score vector \( x \) as

\[
\max \{0, 1 + x_u - x_v\}
\]

- The complete optimization problem is to

\[
\min_x x^\top L x + B \sum_{u \prec v} \max \{0, 1 + x_u - x_v\}
\]

- \( B \) balances between roughness and data fit
- Because \( L \) is positive semidefinite, this is a convex quadratic program with linear constraints
Directed graph Laplacian

- Assume each row of $A$ has at least one nonzero element
- Let $D(u, u)$ be the sum of the $u$th row of $A$
- Define Markovian transition probability matrix $Q \in [0, 1]^{n \times n}$ with $Q(u, v) = \Pr(v|u) = A(u, v)/D(u, u)$
- Assume the Markov random walk is irreducible and aperiodic
- Let $\pi \in \mathbb{R}^n$ be the steady-state probability vector for the random walk, and $\Pi = \text{diag}(\pi)$
- The directed graph Laplacian is defined as

$$L = I - \frac{\Pi^{1/2} Q \Pi^{-1/2} + \Pi^{-1/2} Q \Pi^{1/2}}{2}$$

- Use in optimization in place of undirected graph Laplacian
Smoothing properties

- We can show that

\[ x^\top L x = \sum_{(u,v) \in E} \pi(u) Q(u,v) \left( \frac{x_u}{\sqrt{\pi(u)}} - \frac{x_v}{\sqrt{\pi(v)}} \right)^2 \]

- In \( \min_x x^\top L x + B \sum_{u \preceq v} \max\{0, 1 + x_u - x_v\} \), suppose we set \( B = 0 \) (i.e., only smoothness matters)

- Clearly, \( x_u \propto \sqrt{\pi(u)} \) will minimize \( x^\top L x \)

- I.e., in the absence of training preferences, a directed Laplacian smoother will lead to ordering nodes by decreasing Pagerank
Laplacian smoothing results

In our experiments, we used a subset of the SCOP data consisting of 3314 proteins from two classes. We tokenized the documents using the Rainbow software package written by An-...
Limitations of the graph Laplacian approach

- The “link as hint of score smoothness” view is not universally applicable: millions of obscure pages $u$ link to $v = \text{http://yahoo.com}$, with $x_u \ll x_v$
- While $\pi(u)$ is a probability, $x_u \in \mathbb{R}$ is an arbitrary score that need not satisfy Markov balance constraints (coming soon) and may even be negative
- Dual optimization involves computing the pseudoinverse $L^+$ of the Laplacian matrix
- Unlike $L$, $L^+$ is usually not sparse, and most packages need to hold it in RAM
- The generalization power of the learner (defined later) depends on $\kappa = \max_{u \in V} L^+(u, u)$, a quantity hard to interpret
Pagerank as network circulation

- Can use $Q$ and $\pi$ to define a reference circulation 
  $\{q_{uv} : (u, v) \in E\}$ as follows:

  $$q_{uv} = \pi(u)Q(u, v)$$

- Idea: directly search for a circulation 
  $\{p_{uv} : (u, v) \in E\}$

- Pagerank of node $v$ will fall out naturally as $\sum_{(u,v)\in E} p_{uv}$
What properties must \( \{p_{uv}\} \) satisfy?

- \( p_{uv} \geq 0 \) for all \((u, v) \in E\)
- \( \sum_{(u,v) \in E} p_{uv} = 1 \)
- Flow balance at each node \( v \):
  \[
  \sum_{u \in V} p_{uv} = \sum_{w \in V} p_{vw}
  \]

What roughness penalty should we assess?

- May want to maximize the entropy of \( \{p_{uv} : (u, v) \in E\} \), i.e., \(-\sum_{u,v} p_{uv} \log p_{uv}\)
- May want to propose flow \( \{q_{uv} : (u, v) \in E\} \) as a parsimonious belief and minimize
  \[
  \text{KL}(p||q) = \sum_{u,v} p_{uv} \log \frac{p_{uv}}{q_{uv}}
  \]
- Can show that staying close to \( q \) is good for learning
Unconstrained maximum entropy flows

- Associate dual variable $\beta_v$ for every flow balance constraint

\[ \sum_{u \in V} p_{uv} = \sum_{w \in V} p_{vw} \]

- By dualizing the optimization, we see that HW flows have the form

\[ p_{uv} \propto q_{uv} \exp(\beta_v - \beta_u) \]

- Dual objective is $\min_{\beta} Z$ where

\[ Z = \sum_{(u,v) \in E} q_{uv} \exp(\beta_v - \beta_u) \]
Optimizing $\{p_{uv}\}$ with teleports

- The Markov walk specified by $Q$ need not be irreducible and aperiodic
- As in Pagerank, we can make it so using teleports
- Walk probability $\alpha \in (0, 1)$, teleport probability $1 - \alpha$
- Implement teleport using transition from every $v$ to dummy node $d$ and back
- This leads to additional primal constraints

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

- And dual variables $\tau_v$, leading to the solution

$$p_{dv} \propto q_{dv} \exp(\beta_v - \beta_d)$$
$$p_{vd} \propto q_{dv} \exp(\beta_d - \beta_v + \alpha \tau_v)$$
$$p_{uv} \propto q_{uv} \exp(\beta_v - \beta_u - (1 - \alpha) \tau_u)$$
Preference constraints

- Preference $u \prec v$ leads to constraint

\[ \sum_{(w,u) \in \hat{E}} p_{wu} \leq \sum_{(w,v) \in \hat{E}} p_{wv}, \]

where $\hat{E} = E \cup \{(v, d) : v \in V\} \cup \{(d, v) : v \in V\}$

- Note, no margin (yet)

- Corresponding dual variables $\{\pi_{uv} : u \prec v\}$

- Define $\text{bias}(v) = \sum_{r \prec v} \pi_{rv} - \sum_{v \prec s} \pi_{vs}$

- Modified solution has form

\[
\begin{align*}
p_{dv} & \propto q_{dv} \exp(\beta_v - \beta_d + \text{bias}(v)) \\
p_{vd} & \propto q_{dv} \exp(\beta_d - \beta_v + \alpha \tau_v) \\
p_{uv} & \propto q_{uv} \exp(\beta_v - \beta_u - (1 - \alpha) \tau_u + \text{bias}(v))
\end{align*}
\]
Performance of constrained circulation approach

- Must check primal constraints before terminating dual
- Scales linearly with $|V|$, $|E|$ and $|≺|$
Incorporating an additive margin

- Preference constraints were expressed as
  \[ \sum_{(w,u) \in \hat{E}} p_{wu} \leq \sum_{(w,v) \in \hat{E}} p_{wv}, \text{ not} \]
  \[ 1 + \sum_{(w,u) \in \hat{E}} p_{wu} \leq s_{uv} + \sum_{(w,v) \in \hat{E}} p_{wv} \]

- \( s_{uv} \geq 0 \) is a primal slack variable

- Because \( \sum_{u,v} p_{uv} = 1 \), 1 is “too aggressive” as a margin

- . . . unless we scale up \( \{p_{uv}\} \)

- Let \( q \) be a probability distribution and \( p \) an unnormalized distribution such that \( \sum_x p(x) = F \)
  
  - \( \text{KL}(p\|q) \geq 0 \) if \( F \geq 1 \)
  
  - For a fixed \( F \geq 1 \), \( \text{arg min}_p \text{KL}(p\|q) = Fq \)

- New objective

  \[ \min_{\{p_{uv}\}, \{s_{uv} \geq 0\}, F \geq 1} \text{KL}(p\|q) + C \sum_{u \prec v} s_{uv} + C_1 F^2 \]

- New constraint \( \sum_{u,v} p_{uv} = F \) replaces \( \sum_{u,v} p_{uv} = 1 \)
In Laplace score smoothing, node scores can induce all possible permutations.

In case of network circulation, many node permutations may not be achievable for a given graph.

Smaller hypothesis space, more bias, more stable.

Seems to actually help; even better with additive margin.
Typed edge conductance

- In the constrained circulation formulation, training input has very local effect owing to teleport.
- Beyond a distance of about $1/(1 - \alpha)$, training preferences cannot generalize.
- A different, very common setting associates a type $t(u, v) \in \{1, \ldots, T\}$ with each edge $(u, v)$.
- The weight of edge $(u, v)$ is $\beta(t(u, v))$.
- Given $\prec$ we want to estimate $\beta_1, \ldots, \beta_T$.
- Assuming no dead-end nodes,

$$C(j, i) = \begin{cases} 
\alpha \frac{\beta(t(i, j))}{\sum_{(i, k) \in E} \beta(t(i, k))}, & i \neq d, j \neq d \\
1 - \alpha, & i \neq d, j = d \\
r_j, & i = d, j \neq d \\
0, & i = j = d
\end{cases}$$

- Here $r_j$ is the teleport into node $j$, implemented using dummy node $d$. 
Constrained design of conductance

- Scaling all $\beta$ by any positive factor keeps all $C(\cdot, \cdot)$ unchanged.
- So we can arbitrarily scale $\beta_t \geq 1$.
- $C$ is a function of $\beta$, therefore sometimes written as $C(\beta)$.
- Goal is to find $\beta \geq \vec{1}$ such that:
  - $p = C(\beta)p$
  - $p_i \leq p_j$ for all $i < j$.
- As before, we can change the constraint $p_i \leq p_j$ into a loss function $\text{loss}(p_i - p_j)$.
- Two problems to solve:
  - Break recursion $p = C(\beta)p$ and express $p$ directly in terms of $\beta$, so we can use a numerical optimizer.
  - If there are many solutions $\beta$, which one should we prefer?
Choice of loss function

- Standard hinge hinge\( (y) = \max\{0, 1 + y\} \)
- As before, enforcing additive margin 1 is tricky
- Scaling \( \beta \) has no effect on satisfying margin
- In practice, no margin or very small arbitrary margin makes no difference, both work well
- To make loss smooth and differentiable, could have picked loss\( (y) = \ln(1 + e^y) \)
- But this does not work, experiments suggest that loss\( (0) = 0 \) is essential
- Approximation of hinge with zero margin (hinge\( (y) = \max\{0, y\} \)) with Huber loss:

\[
\text{huber}(y) = \begin{cases} 
0, & y \leq 0 \\
\frac{y^2}{2W}, & y \in (0, W] \\
y - \frac{W}{2}, & W < y 
\end{cases}
\]
Parsimonious choice of $\beta$

- If $\beta = \vec{1}$, we get unweighted Pagerank
- Therefore the model cost can be taken as $\sum_t (\beta(t) - 1)^2$
- In fact, we get unweighted Pagerank if all $\beta(t)$ are equal, not necessarily all equal to one
- Model cost $\sum_{t,t'} (\beta(t) - \beta(t'))^2$ is another possibility
- Discourages large multiplicative factors . . .
  $\text{ModelCost}(K\beta) = K^2\text{ModelCost}($\beta$)$
- . . . but not additive terms:
  $\text{ModelCost}(\beta + K\vec{1}) = \text{ModelCost}($\beta$)$$
- In practice these work about equally well
Breaking the $p = C(\beta)p$ recursion \footnote{\textit{PageRank} usually approximated using the Power Method $p \approx C^H p^0$ where}

- $p^0$ is an initial distribution over nodes, usually uniform
- $H$ is a suitably large horizon for convergence

- Overall optimization problem:

$$
\min_{\beta \geq 1} \sum_t (\beta(t) - 1)^2 + B \sum_{i < j} \text{huber}((C^H p^0)_i - (C^H p^0)_j)
$$

- Unfortunately, not a convex optimization; need some grid plus local gradient search

- Next: computing gradient
Breaking the $p = C(\beta)p$ recursion II

- Compute alongside PageRank (using Chain Rule):

$$\forall i \forall t : \frac{\partial}{\partial \beta(t)}(C^0 p^0)_i = 0$$

$$(C^h p^0)_i = \sum_j C(i, j)(C^{h-1} p^0)_j$$

$$\frac{\partial(C^h p^0)_i}{\partial \beta(t)} = \sum_j \left[ \frac{\partial C(i, j)}{\partial \beta(t)}(C^{h-1} p^0)_j + C(i, j)\frac{\partial}{\partial \beta(t)}(C^{h-1} p^0)_j \right]$$

- Finally,

$$\frac{\partial C(i, j)}{\partial \beta(\tau)} = \begin{cases} -\alpha^{\beta(t(i,j)) - \beta(t(i,w))} \sum_w [\tau = t(i,w)] & \tau \neq t(i,j) \\ \alpha \frac{C(t(i,w)) - C(i, j)}{(\sum_w \beta(t(i,w)))^2} \sum_w [\tau = t(i,w)] & \tau = t(i,j) \end{cases}$$
Exact loss and the approximations

Theoretically, the optimization surface has local minima

Wrt $\beta$, the surface is very benign in practice

If one also wanted to search for $\alpha$, a little more care is needed.
$\beta$ estimation and learning performance

- Fast training rate
- Robust to training noise
- Reconstructs $\beta$ reasonably
Preliminaries

- Motivation
- Training and evaluation setup
- Performance measures

Ranking in vector spaces

- Discriminative, max-margin algorithms
- Probabilistic models, gradient-descent algorithms

Ranking nodes in graphs

- Roughness penalty using graph Laplacian
- Constrained network flows

Stability and generalization

- Admissibility and stability
- Ranking loss and generalization bounds
Some sample results

- Pagerank is score-stable but not rank-stable
- (HITS is not score-stable and not rank-stable)
- More notions of stability, connections with generalization
- Max-margin vector-space ranking is stable
- Ranking based on Laplace smoothing is stable
- Ranking based on constrained circulation is stable
Pagerank is score-stable when $G$ is perturbed

- $V$ kept fixed
- Nodes in $P \subset V$ get incident links changed in any way (additions and deletions)
- Thus $G$ perturbed to $\tilde{G}$
- Let the random surfer visit (random) node sequence $X_0, X_1, \ldots$ in $G$, and $Y_0, Y_1, \ldots$ in $\tilde{G}$
- Coupling argument: instead of two random walks, we will design one joint walk on $(X_i, Y_i)$ such that the marginals apply to $G$ and $\tilde{G}$
Coupled random walks on $G$ and $\tilde{G}$

- Pick $X_0 = Y_0 \sim \text{Multi}(r)$
- At any step $t$, with probability $1 - \alpha$, reset both chains to a common node using teleport $r$: $X_t = Y_t \in_r V$
- With the remaining probability of $\alpha$
  - If $x_{t-1} = y_{t-1} = u$, say, and $u$ remained unperturbed from $G$ to $\tilde{G}$, then pick one out-neighbor $v$ of $u$ uniformly at random from all out-neighbors of $u$, and set $X_t = Y_t = v$.
  - Otherwise, i.e., if $x_{t-1} \neq y_{t-1}$ or $x_{t-1}$ was perturbed from $G$ to $\tilde{G}$, pick out-neighbors $X_t$ and $Y_t$ independently for the two walks.
Analysis of coupled walks

Let $\delta_t = \Pr(X_t \neq Y_t)$; by design, $\delta_0 = 0$.

$$
\delta_{t+1} = \Pr(\text{reset at } t + 1) \Pr(X_{t+1} \neq Y_{t+1}|\text{reset at } t + 1) +
\Pr(\text{no reset at } t + 1) \Pr(X_{t+1} \neq Y_{t+1}|\text{no reset at } t + 1)
= \Pr(\text{reset at } t + 1) \cdot 0 + \alpha \Pr(X_{t+1} \neq Y_{t+1}|\text{no reset at } t + 1)
= \alpha \left( \Pr(X_{t+1} \neq Y_{t+1}, X_t \neq Y_t|\text{no reset at } t + 1) + \Pr(X_{t+1} \neq Y_{t+1}, X_t = Y_t|\text{no reset at } t + 1) \right)
$$
The event $X_{t+1} \neq Y_{t+1}, X_t = Y_t$ can happen only if $X_t \in P$. Therefore we can continue the above derivation as follows:

$$
\delta_{t+1} = \ldots \\
\leq \alpha \left( \Pr(X_t \neq Y_t \mid \text{no reset at } t + 1) + \Pr(X_{t+1} \neq Y_{t+1}, X_t = Y_t, X_t \in P \mid \text{no reset at } t + 1) \right) \\
= \alpha \left( \Pr(X_t \neq Y_t) + \Pr(X_{t+1} \neq Y_{t+1}, X_t = Y_t, X_t \in P \mid \text{no reset at } t + 1) \right) \\
\leq \alpha \left( \Pr(X_t \neq Y_t) + \Pr(X_t \in P) \right) \\
= \alpha \left( \delta_t + \sum_{u \in P} p_u \right),
$$

(using $\Pr(H, J \mid K) \leq \Pr(H \mid K)$, and that events at time $t$ are independent of a potential reset at time $t + 1$)
Analysis of coupled walks III

Unrolling the recursion,
\[ \delta_\infty = \lim_{t \to \infty} \delta_t \leq \left( \sum_{u \in P} p_u \right) / (1 - \alpha) \]

- Standard result: If the probability of a state disagreement between the two walks is bounded, then their Pagerank vectors must also have small $L_1$ distance to each other. In particular,

\[ \|p - \tilde{p}\|_1 \leq \frac{2}{1 - \alpha} \sum_{u \in P} p_u \]

- Lower the value of $\alpha$, the more the random surfer teleports and more stable is the system
- Gives no direct guidance why $\alpha$ should not be set to exactly zero!
Pagerank is not rank-stable when $G$ is perturbed

- Adversarial setting
- $G$ formed by connecting $y$ to $x_a$, $\tilde{G}$ by connecting $y$ to $x_b$
- $\Omega(n^2)$ node pairs flip Pagerank order
- I.e., $L_1$ score stability does not guarantee rank stability
- Can “natural” social networks lead often to such tie-breaking?
Generalization of bipartite ranking

- \( f : \mathcal{X} \rightarrow \mathbb{R} \) is a fixed ranking function
- The ("true") ranking accuracy of \( f \) is
  \[
  A(f) = \mathbb{E}_{X \in \mathcal{D}_1, X' \in \mathcal{D}_{-1}} \left( \left\lfloor f(X) > f(X') \right\rfloor + \frac{1}{2} \left\lfloor f(X) = f(X') \right\rfloor \right)
  \]
- Recall that the empirical ranking accuracy of \( f \) over training set \( T \) is denoted \( \hat{A}(f, T) \)
- We are interested in upper-bounding
  \[
  \Pr(\left| \hat{A}(f, T) - A(f) \right| > \epsilon)
  \]
- Recall that \( T = \{(x_i, y_i \in \{-1, 1\})\} \) in bipartite ranking; projections on \( \mathcal{X} \) and \( \mathcal{Y} \) are called \( T_X \) and \( T_Y \)
- Let there be \( m \) positive and \( n \) negative instances, and \( y \) the sequence of labels
  \[
  \Pr_{T_X|T_Y=y} (\hat{A}(f, T) - A(f) \geq \epsilon) \leq 2e^{-2mn\epsilon^2/(m+n)}
  \]
Generalization of circulation-based ranking

- Given graph \( G = (V, E) \)
- Rewrite regularized optimization objective in the form

\[
R_{\text{reg}}(p) = \frac{1}{m} \sum_{j=1}^{m} \max \left\{ 0, \sum_{(w,u) \in \hat{E}} p_{wu} - \sum_{(w,v) \in \hat{E}} p_{wv} \right\}
\]

\[
+ \lambda \text{KL}(p || q)
\]

- \( \prec \) is sampled randomly from \( V \times V \) according to some unknown fixed distribution
- Over random draws of \( \prec \) with \( |\prec| = m \), with probability at least \( 1 - \delta \),

\[
R \leq R_{\text{emp}} + \frac{4 \ln 2}{\lambda m} + \left( \frac{8 \ln 2}{\lambda} + 1 \right) \sqrt{\frac{\ln(1/\delta)}{2m}}
\]

- Here \( R \) is the true ranking loss and \( R_{\text{emp}} \) is the empirical ranking loss over training data
Preliminaries

- Motivation
- Training and evaluation setup
- Performance measures

Ranking in vector spaces

- Discriminative, max-margin algorithms
- Probabilistic models, gradient-descent algorithms

Ranking nodes in graphs

- Roughness penalty using graph Laplacian
- Constrained network flows

Stability and generalization

- Admissibility and stability
- Ranking loss and generalization bounds
Preliminaries

- Motivation
- Training and evaluation setup
- Performance measures

Ranking in vector spaces

- Discriminative, max-margin algorithms
- Probabilistic models, gradient-descent algorithms

Ranking nodes in graphs

- Roughness penalty using graph Laplacian
- Constrained network flows

Stability and generalization

- Admissibility and stability
- Ranking loss and generalization bounds
Preliminaries

- Motivation
- Training and evaluation setup
- Performance measures

Ranking in vector spaces

- Discriminative, max-margin algorithms
- Probabilistic models, gradient-descent algorithms

Ranking nodes in graphs

- Roughness penalty using graph Laplacian
- Constrained network flows

Stability and generalization

- Admissibility and stability
- Ranking loss and generalization bounds
Preliminaries

- Motivation
- Training and evaluation setup
- Performance measures

Ranking in vector spaces

- Discriminative, max-margin algorithms
- Probabilistic models, gradient-descent algorithms

Ranking nodes in graphs

- Roughness penalty using graph Laplacian
- Constrained network flows

Stability and generalization

- Admissibility and stability
- Ranking loss and generalization bounds
References


References II


References IV


References V


References VI

