Statistical Ranking Problem

Tong Zhang

Yahoo! Inc. New York City

Joint work with

David Cossock

Yahoo! Inc. Santa Clara

Agenda

- Some massive data analysis problems on the internet.
- Earlier work on ranking.
- Web-search ranking: some theoretical issues.
  - relation to matrix reconstruction.
  - relating reconstruction error to ranking error.
  - statistical error: derive bounds independent of massive web-size.
  - learning method: importance weighted regression.

Some Massive Data Analysis Problems at Yahoo

- Straight forward applications of basic classification.
- Community, social network and user behavior analysis.
- Advertizing.
- Ranking problems and applications.

Some Basic Classification Problems

- Classification of text-documents.
  - email spam, web-page spam.
  - web-page content classification, document type classification, etc.
  - adversarial scenario; dynamic nature.

- Basic algorithms: linear classification, kernels, boosting, etc.

- Feature engineering very important: text + structured non-text features.

- Some problems need more complicated modeling:
  - methods to use link information (classification with web-graph structure)
  - methods to take advantage of community effect.

Community analysis

- Social network (web 2.0): users help each other.
  - tagging, blogging, reviews, user provided content, etc
  - methods to encourage users to interact and provide contents.
  - methods to help users finding quality information more easily.
  - methods to analyze user behavior/intention.

- Classification: determine content quality, user expertise on topics, etc

- Ranking: rank content based on user intention (question answering, ads).

- Social network connectivity graphs with typed (tagged) edges.
  - link prediction and tag prediction.
  - hidden community discovery.
  - Personalized recommender system (ranking).

Advertizing

- What ads to put on what page:
  - click through rate prediction.
  - user intention analysis.
  - personalization (predict future behavior based on historic behavior).
- Matching:
  - closeness between keywords, queries, contents.
  - suggest better keywords or summaries for advertisers.
- Predict quality of advertisers.
- Predict quality of user clicks.

Ranking Problems

- Rank a set of items and display to users in corresponding order.

- Important in web-search:
  - web-page ranking
    - display ranked pages for a query
  - query-refinement and spelling correction
    - display ranked suggestions and candidate corrections
  - web-page summary
    - display ranked sentence segments
  - related: select advertisements to display for a query.
  - related: crawling/indexing:
    - which page to crawl first
    - pages to keep in the index: priority/quality

Earlier Work on Statistical Ranking

- Statistics: most related is ordinal regression (ordered output)
  - in ranking, we want to order inputs.

- Machine learning: pairwise preference learning (local and global)
  - learn a local scoring function f for items to preserve preference $\prec$.
    * two items x and x': $f(x) < f(x')$ when $x \prec x'$.
    * ordering inputs according to x.
  - learn a pair-wise decision function f
    * $f(x, x') \to \{0, 1\}$: whether $x \prec x'$.
    * need method to order x using $f(x, x')$ (related: sorting with noise).
  - learn a global rank-list decision function f
    * two ordered rank-list $I = \{x_{i_1}, \ldots, x_{i_m}\}$ and $I' = \{x_{i'_1}, \ldots, x_{i'_m}\}$.
    * learn a global scoring function for rank-list: $f(I) < f(I')$ when $I \prec I'$.
    * modeling and search issues (related to structured-output prediction)

Theoretical Results on Ranking

- Global ranking criterion:
  - number of mis-ordered pairs: $\mathbf{E}_x \mathbf{E}_{x'} I(x \prec x' \& f(x) \geq f(x'))$.
  - related to AUC (area under ROC) in binary classification.
  - studied by many authors: Agarwal, Graepel, Herbrich, Har-Peled, Roth, Rudin, Clemencon, Lugosi, Vayatis, Rosset ...
- Practical ranking (e.g. web-search):
  - require subset ranking model
  - focus quality on top (not studied except a related paper [Rudin, COLT 06]).
- Our goal:
  - introduce the sub-set ranking model.
  - theoretically analyze how to solve a large scale ranking problem
    - learnability and error bounds.
    - importance sampling/weighting crucial in the analysis.

Web-Search Problem

- User types a query, search engine returns a result page:
  - selects from billions of pages.
  - assign a score for each page, and return pages ranked by the scores.

- Quality of search engine:
  - relevance (whether returned pages are on topic and authoritative)
  - presentation issues (diversity, perceived relevance, etc)
  - personalization (predict user specific intention)
  - coverage (size and quality of index).
  - freshness (whether contents are timely).
  - responsiveness (how quickly search engine responds to the query).

Relevance Ranking as Matrix Reconstruction

- Massive size matrix
  - rows: all possible queries
  - columns: all web-pages (Yahoo index size disclosed last year: 20 billion)
- Question: can we reconstruct the whole matrix from a few rows?
  - no if treated as matrix reconstruction without additional information
    - why: singular value decays slowly.
  - yes if given additional features characterizing each matrix entry
    - treat as a statistical learning problem.
    - require more complicated learning theory analysis.
    - Frobenius norm (least squares error) not good reconstruction measure.
- Learning theory can give error/concentration bounds for matrix reconstruction.
  - some ideas from matrix reconstruction may be applicable in learning.

Relevance Ranking: Statistical Learning Formulation

- Training:
  - randomly select queries q, and web-pages p for each query.
  - use editorial judgment to assign relevance grade $y(p,q)$.
  - construct a feature $x(p,q)$ for each query/page pair.
  - learn scoring function $\hat{f}(x(p,q))$ to preserve the order of $y(p,q)$ for each q.
- Deployment:
  - query q comes in.
  - return pages $p_1,\ldots,p_m$ in descending order of $\hat{f}(x(p,q))$.

Measuring Ranking Quality

- Given scoring function $\hat{f}$, return ordered page-list $p_1, \ldots, p_m$ for a query q.
  - only the order information is important.
  - should focus on the relevance of returned pages near the top.

- DCG (discounted cumulative gain) with decreasing weight c_i

$$\mathbf{DCG}(\hat{f}, q) = \sum_{j=1}^{m} c_i r(p_i, q).$$

- c_i: reflects effort (or likelihood) of user clicking on the i-th position.

Subset Ranking Model

- $x \in \mathcal{X}$: feature ($x(p,q) \in \mathcal{X}$)
- $S \in \mathcal{S}$: subset of $\mathcal{X}$ ($\{x_1, \ldots, x_m\} = \{x(p,q) : p\} \in \mathcal{S}$)
  - each subset corresponds to a fixed query q.
  - assume each subset of size m for convenience: m is large.
- y: quality grade of each $x \in \mathcal{X}$ ($y(p,q)$).
- scoring function $f : \mathcal{X} \times \mathcal{S} \to R$.
  - ranking function $r_f(S) = \{j_i\}$: ordering of $S \in \mathcal{S}$ based on scoring function f.
- quality: $\mathbf{DCG}(f, S) = \sum_{i=1}^{m} c_i \mathbf{E}_{y_{j_i}|(x_{j_i}, S)}\, y_{j_i}$.

Some Theoretical Questions

- Learnability:
  - subset size m is huge: do we need many samples (rows) to learn.
  - focusing quality on top.
- Learning method:
  - regression.
  - pair-wise learning? other methods?
- Limited goal to address here:
  - can we learn ranking by using regression when m is large.
    - massive data size (more than 20 billion)
    - want to derive: error bounds independent of m.
  - what are some feasible algorithms and statistical implications.

Bayes Optimal Scoring

- Given a set $S \in \mathcal{S}$, for each $x_j \in S$, we define the Bayes-scoring function as

$$f_B(x_j, S) = \mathbf{E}_{y_j|(x_j,S)}\, y_j$$

- The optimal Bayes ranking function r_{f_B} that maximizes DCG
  - induced by f_B
  - returns a rank list $J = [j_1, \ldots, j_m]$ in descending order of $f_B(x_{j_i}, S)$.
  - not necessarily unique (depending on c_j)

- The function is subset dependent: require appropriate result set features.

Simple Regression

- Given subsets $S_i = \{x_{i,1}, \ldots, x_{i,m}\}$ and corresponding relevance score $\{y_{i,1}, \ldots, y_{i,m}\}$.

- We can estimate $f_B(x_j, S)$ using regression in a family $\mathcal{F}$:

$$\hat{f} = \arg\min_{f \in \mathcal{F}} \sum_{i=1}^{n} \sum_{j=1}^{m} (f(x_{i,j}, S_i) - y_{i,j})^2$$

- Problem: m is massive ($>$ 20 billion)
  - computationally inefficient
  - statistically slow convergence
    - ranking error bounded by $O(\sqrt{m})\times$ root-mean-squared-error.

- Solution: should emphasize estimation quality on top.

Importance Weighted Regression

- Some samples are more important than other samples (focus on top).

- A revised formulation: $\hat{f} = \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} L(f, S_i, \{y_{i,j}\}_j)$, with

$$L(f, S, \{y_j\}_j) = \sum_{j=1}^{m} w(x_j, S)(f(x_j, S) - y_j)^2 + u \sup_j w'(x_j, S)(f(x_j, S) - \delta(x_j, S))_+^2$$

- weight w: importance weighting focusing regression error on top
  - zero for irrelevant pages

- weight w': large for irrelevant pages
  - for which $f(x_j, S)$ should be less than threshold δ.

- importance weighting can be implemented through importance sampling.

Relationship of Regression and Ranking

Let $Q(f) = \mathbf{E}_S L(f, S)$, where

$$L(f, S) = \mathbf{E}_{\{y_j\}_j|S} L(f, S, \{y_j\}_j)$$

$$= \sum_{j=1}^{m} w(x_j, S) \mathbf{E}_{y_j|(x_j,S)} (f(x_j, S) - y_j)^2 + u \sup_j w'(x_j, S)(f(x_j, S) - \delta(x_j, S))_+^2.$$

Theorem 1. *Assume that $c_i = 0$ for all $i > k$. Under appropriate parameter choices with some constants u and γ, for all f:*

$$\mathbf{DCG}(r_B) - \mathbf{DCG}(r_f) \leq C(\gamma, u)(Q(f) - \inf_{f'} Q(f'))^{1/2}.$$

Appropriate Parameter Choice (for previous Theorem)

- One possible theoretical choice:
  - Optimal ranking order: $J_B = [j_1^*, \ldots, j_m^*]$, where $f_B(x_{j_i^*})$ is arranged in non-increasing order.
  - Pick δ such that $\exists\gamma \in [0, 1)$ with $\delta(x_j, S) \le \gamma f_B(x_{j_k^*}, S)$.
  - Pick w such that for $f_B(x_j, S) > \delta(x_j, S)$, we have $w(x_j, S) \ge 1$.
  - Pick w' such that $w'(x_j, S) \ge I(w(x_j, S) < 1)$.

- Key in this analysis:
  - focus on relevant documents on top.
  - $\sum_j w(x_j, S)$ is much smaller than m.

Generalization Performance with Square Regularization

Consider scoring $f_{\hat{\beta}}(x,S) = \hat{\beta}^T \psi(x,S)$, with feature vector $\psi(x,S)$:

$$\hat{\beta} = \arg\min_{\beta \in \mathcal{H}} \left[\frac{1}{n} \sum_{i=1}^{n} L(\beta, S_i, \{y_{i,j}\}_j) + \lambda \beta^T \beta \right], \tag{1}$$

$$L(\beta, S, \{y_j\}_j) = \sum_{j=1}^{m} w(x_j, S)(f_\beta(x_j, S) - y_j)^2 + u \sup_j w'(x_j, S)(f_\beta(x_j, S) - \delta(x_j, S))_+^2.$$

Theorem 2. *Let* $M = \sup_{x,S} \|\phi(x,S)\|_2$ *and* $W = \sup_S [\sum_{x_j \in S} w(x_j, S) + u \sup_{x_j \in S} w'(x_j, S)]$. *Let* $f_{\hat{\beta}}$ *be the estimator defined in (1). Then we have*

$$\begin{aligned}
&\mathbf{DCG}(r_B) - \mathbf{E}_{\{S_i, \{y_{i,j}\}_j\}_{i=1}^n} \mathbf{DCG}(r_{f_{\hat{\beta}}}) \\
\leq & C(\gamma, u) \left[\left(1 + \frac{WM}{\sqrt{2}\lambda n}\right)^2 \inf_{\beta \in \mathcal{H}} (Q(f_\beta) + \lambda \beta^T \beta) - \inf_f Q(f) \right]^{1/2}.
\end{aligned}$$

Interpretation of Results

- Result does not depend on m, but the much smaller quantity quantity $W = \sup_S [\sum_{x_j \in S} w(x_j, S) + u \sup_{x_j \in S} w'(x_j, S)]$
  - emphasize relevant samples on top.
  - a refined analysis can replace $\sup$ over S by some p-norm over S.
- Can control generalization for the top portion of the rank-list even with large m.
  - learning complexity does not depend on the majority of items near the bottom of the rank-list.
  - the bottom items are usually easy to estimate.

Some Conclusions

- Web-search ranking problem can be viewed as a more sophisticated matrix reconstruction problem with a different error criterion.

- Ranking quality near the top is most important.

- Solving ranking problem using regression:
  - small least squares error does not imply good ranking error.
  - theoretically solvable using importance weighted regression: can prove error bounds independent of the massive web-size.

- Subset features are important.