

Large Scale Ranking Problem: some theoretical and algorithmic issues

Tong Zhang

Yahoo Inc.! New York City

collaborators: David Cossock (Yahoo), Christoph Tillmann (IBM)

Agenda

- Some machine learning problems on the internet.
- Ranking: theoretical issues in the context of web-search.
 - with David Cossock at Yahoo.
- Ranking: algorithmic issues in the context of machine translation
 - with Christoph Tillmann at IBM.

A Subset of Machine Learning Activities 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 used: linear classification, kernels, boosting, etc.
- Feature engineering very important: text + structured non-text features.
- Some problems can benefit from 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).

Advertisizing

- 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.

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)
 - presentation issues (diversity, perceived relevance, etc)
 - personalization (predict user specific intention)
 - coverage (size of the index).
 - freshness (whether contents are timely).
 - responsiveness (how quick search engine responds to the query).

Relevance Ranking

- 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 a scoring function $\hat{f}(x(p, q))$ that preserves the order of $y(p, q)$ for each q .
- Deployment:
 - query q comes in.
 - Return pages p_1, \dots, p_m in descending order of $\hat{f}(x(p, q))$.

Measuring Ranking Quality

- Given a scoring function \hat{f} returning ordered list of pages p_1, \dots, p_m for a query q .
 - only the order information is important.
 - should focus on the relevance of pages near the top.
- DCG (discounted cumulative gain) with decreasing weight c_i

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

- 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} ($\{x_1, \dots, x_m\} = \{x(p, q) : p\} \in \mathcal{S}$)
 - 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 $f : \mathcal{X} \times \mathcal{S} \rightarrow R$.
 - ranking function $r_f(S) = \{j_i\}$: ordering of $S \in \mathcal{S}$ based on scoring function f .
- quality: $\text{DCG}(f, S) = \sum_{i=1}^m c_i \mathbf{E}_{y_{j_i} | (x_{j_i}, S)} y_{j_i}$.

Some Theoretical Questions

- Learnability:
 - subset size m is huge: do we need many samples 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.
 - what are some important considerations.

Bayes Optimal Scoring

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

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

- Then the optimal ranking function maximizing DCG is the rank function r_{f_B} induced by the Bayes-scoring function.
 - returns a rank list $J = [j_1, \dots, j_m]$ in descending order of $f_B(x_{j_i}, S)$.
- The function is subset dependent: require appropriate result set features.

Simple Regression

- Given subsets $S_i = \{x_{i,1}, \dots, x_{i,m}\}$ and corresponding relevance score $\{y_{i,1}, \dots, y_{i,m}\}$.
- 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: the effect of large m .
 - computationally inefficient
 - statistically slow convergence
- Solution: should emphasize estimation quality on top.

Importance Weighted Regression

- Some samples are more important than other samples (focus on top).
- A revised formulation:

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

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

- 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 .

Relationship of Regression and Ranking (general)

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

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

Theorem 1. *Assume that $c_i = 0$ for all $i > k$. Under appropriate parameter choices, 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

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

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

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

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 $W = \sup_S [\sum_{x_j \in S} w(x_j, S) + u \sup_{x_j \in S} w'(x_j, S)]$ emphasizing samples on top.
- Can control generalization for the top 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

- For most applications, ranking quality near the top is most important.
- Can be solved with importance sample weighted regression.
- Subset features are important.

Related Work on Statistical Ranking

- Statistics: ordinal regression with ordered output
 - we want to order inputs.
- Machine learning: pairwise preference learning :
 - learning a scoring function to preserve order
 - * $f(x) < f(x')$ when $x \prec x'$.
 - some recent learning theory results studying global criterion
 - * Many authors: Agarwal, Graepel, Herbrich, Har-Peled, Roth, Rudin, Clemencon, Lugosi, Vayatis, Rosset ...
 - * number of mis-ordered pairs: $\mathbf{E}_x \mathbf{E}_{x'} I(x \prec x' \& f(x) \geq f(x'))$.
 - * web-search: require subset ranking model focusing quality on top.

Can we improve regression based ranking

- Is pair-wise learning more appropriate (e.g. in MS ranking, Burges, et al)?
 - probably not much if regression is used corrected.
 - similar effects (benefits) can be achieved with regression and set dependent features.
- Proper uncertainty control (e.g. importance weights) for each point.
- Proper set-dependent scaling of output y .
- Regression learning with more complicated subset models.

Some Algorithmic Issues in Large-scale Ranking (Statistical Machine Translation)

- Translation problem:
 - given a source sentence in one language.
 - generate a target sentence in another language.
- General approach:
 - scoring function for each input/output pair (ranking): measuring the quality of candidate translation for source sentence
 - * similar to web-search (input: query, output: page).
 - Translation step:
 - * effectively generate target sentence candidates.
 - * search for the optimal score (or possibly top k scores).

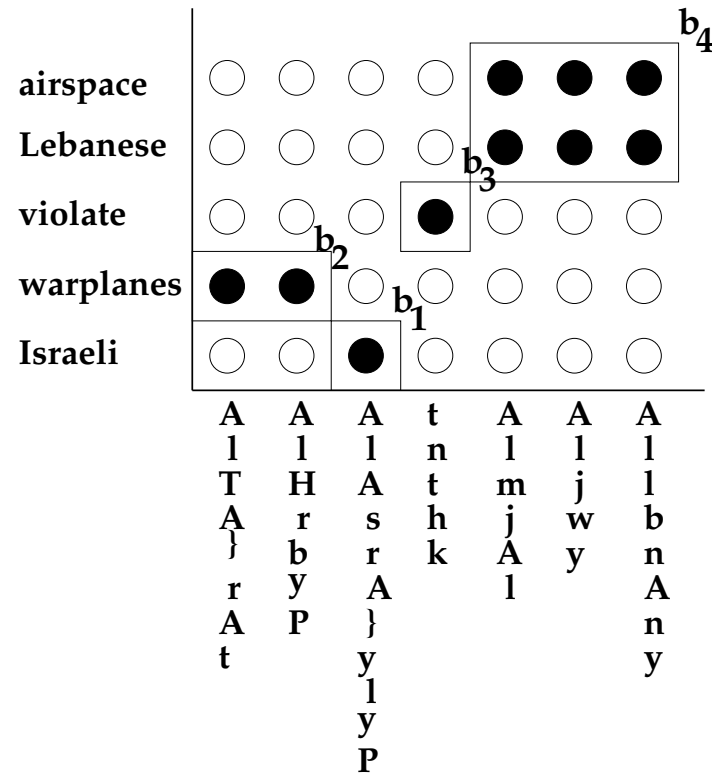
Analogy: Spelling correction in web-search

- Input: a query (mis-spelled)
- Output: ranked list of suggested corrections.
- Machine learning approach:
 - generate candidate corrections.
 - score each correction, and output the top selections.
- Difference in the machine translation problem:
 - search space (candidate sentences) is much larger.
 - efficient strategy to search candidate space is important.

Candidate Generator: block sequence decoder

- Database: a set of possible translation blocks
 - a block of tokens (phrase) in the source language can be translated into a block of tokens (phrase) in the target language
 - e.g. “a b” translates into “z x y”.
- Scoring function:
 - candidate translation: ordered block sequence (b_1, \dots, b_n)
 - map each sequence of blocks into a non-negative score.
- Input: source sentence.
- Translation:
 - generate block sequence consistent with the source sentence.
 - find the sequence with largest score.

Graphical illustration



More Detailed Decoder Components

- A set of candidate translation blocks
 - obtained using alignment model + pattern mining.
 - purpose: to limit the search space and simplify search strategy.
- Search strategies (with limited reordering)
 - monotone decoding.
 - * try to generate all possible blocks that cover the source sentence
 - * left to right decoding with pruning (dynamic programming, beam search)
 - monotone decoding with local reordering (swap).
 - * the order of adjacent blocks can be flipped.
- Scoring function: Markov linear scoring function
$$s_w(b_1^n) = \sum_{i=1}^n w^T F(b_{i-1}, b_i; o_i): o_i \text{ is local ordering information.}$$

Learning Assumptions: an abstract decoder

- Block sequence $z = \{b_1, \dots, b_n\}$.
- Source sentence S .
 - $V(S)$: block sequence consistent with source and searchable by decoder
- Scoring function: $s_w(z)$
 - goal of learning: optimize w on training data.
- Decoder implements: $\hat{z}(w, S) = \arg \max_{z \in V(S)} s_w(z)$.
- Metrics: BLEU score
 - averaged n -gram matches between machine and human translations

Decoder Training

- Given source/target sentence pairs $\{(S_i, T_i)\}_{i=1, \dots, n}$.
- Given a decoding scheme implementing: $\hat{z}(w, S) = \arg \max_{z \in V(S)} s_w(z)$.
- Find parameter w such that on the training data:
 - $\hat{z}(w, S_i)$ achieves high BLEU score on average.
- Key difficulty: large search space (similar issue in web-search)
- Traditional tuning.
- Our method.

Traditional scoring function

- A bunch of local statistics gathered from the training data, and beyond.
 - different language models of the target $P(T_i|T_{i-1}, T_{i-2}\dots)$.
 - * can depend on statistics beyond training data.
 - * best results benefit significantly from huge language models.
 - orientation (swap) frequency.
 - block frequency.
 - block quality score:
 - * e.g. normalized in-block unigram translation probability:
 $(S, T) = (\{s_1, \dots, s_p\}, \{t_1, \dots, t_q\}); P(S|T) = \prod_j n_j \sum_i p(s_j|t_i)$.
- Linear combination of log-frequencies (five or six features):
 - $s_w(\{b_1^n\}) = \sum_i \sum_j w_j \log f_j(b_i; b_{i-1}, \dots)$
- Tuning: hand-tuning; gradient descent adjustment to optimize BLEU score

Large scale training

- Motivation:
 - the ability of incorporating large number of features.
 - direct discriminative learning.
- Require: automated training method to optimize millions of parameters.
- Challenge: search space $V(S)$ is too large.
- Solution:
 - break it down using most important/relevant samples (near top)
 - can treat decoder as a black-box.

Global training of decoder parameter

- Treat decoder as a black box, implementing: $\hat{z}(w, S) = \arg \max_{z \in V(S)} s_w(z)$.
- No need to know $V(S)$.
- Generate truth set as block sequences with K -largest blue scores: $V_K(S)$.
- Generate alternatives as a subset of $V(S)$ that are “most relevant”.
 - if w does well on the relevant alternatives, it does well on the whole $V(S)$.
 - essentially a sampling/importance-weighting method.

Learning method

- Try to minimize the following regularized empirical risk minimization:

$$\hat{w} = \arg \min_w \left[\frac{1}{m} \sum_{i=1}^N \frac{1}{K} \sum_{z \in V_K} \xi_i(w, z) + \lambda w^2 \right]$$
$$\xi_i(w, z) \geq \max_{z' \in V - V_K} \psi(w, z, z'),$$
$$\psi(w, z, z') = \phi(s_w(z), \text{Bl}(z); s_w(z'), \text{Bl}(z')),$$

- Relevant set: let $\max_{z' \in V(S_i) - V_K(S_i)} \psi(w, z, z')$

$$V^{(r)}(\hat{w}, S_i) = \cup_{z \in V_K(S_i)} \{z' \in V(S_i) : \psi(\hat{w}, z, z') = \xi_i(\hat{w}, z) \neq 0\}.$$

- Key observation: can replace V by $V^{(r)}$ without changing solution.

Example loss functions

- Truth $z \in V_K$, alternative $z' \in V - V_K$ (or in $V^{(r)}$).
- $\text{Bl}(z) > \text{Bl}(z')$: want to penalize if score $s_w(z) \leq s_w(z')$.
- Estimate $s_w(z)$ using least squares:

$$\phi(s_w(z), \text{Bl}(z); s_w(z'), \text{Bl}(z')) = \alpha(s_w(z) - \text{Bl}(z))^2 + \alpha'(s_w(z') - \text{Bl}(z'))^2.$$

– not easy to implement correctly (requires scaling of $\text{Bl}(z)$ and proper weighting).

- Estimate $s_w(z)$ using pair-wise loss:

$$\phi(s_w(z), \text{Bl}(z); s_w(z'), \text{Bl}(z')) = (\text{Bl}(z) - \text{Bl}(z')) \max(1 - (s_w(z) - s_w(z')), 0)^2.$$

Approximate Relevant Set Method

- Observation:
 - relevant set depends on w ;
 - w can be calculated based on an approximation of relevant set.
- Iterate to find both.
 - fix $V^{(r)}$, update w .
 - fix w , update $V^{(r)}$.
- Comment: relevant set corresponds to pages on top in web-search.

Table 1: Generic Relevant Set Algorithm

divide training points into m blocks J_1, \dots, J_m

initialize weight vector $w \leftarrow w_0$

for each data point S

 initialize truth $V_K(S)$ and alternative $V^{(r)}(S)$

for $\ell = 1, \dots, L$

for $j = 1, \dots, m$

for each $S \in J_j$

 select relevant points $\{\tilde{z}_k\} \in S$ (*)

 update $V^{(r)}(S) \leftarrow V^{(r)}(S) \cup \{\tilde{z}_k\}$

 update w by solving the following approximately (**)

$$\min_w \frac{1}{m} \sum_{i=1}^N \Phi(w, V_K(S_i), V^{(r)}(S_i)) + \lambda w^2$$

Convergence analysis

- Approximate relevant set size:
 - relevant set update rule (*): $\tilde{z}_k \in V(S) - V_k(S)$ for each $z_k \in V_K(S)$ ($k = 1, \dots, K$) such that

$$\psi(w, z_k, \tilde{z}_k) = \max_{z' \in V(S) - V_K(S)} \psi(w, z_k, z')$$

- convergence bound independent of the size of $V(S)$.
 - generalization bound: independent of size of $V(S)$.
 - real implementation using decoder output: $\tilde{z} = \max_{z' \in V(S)} s_w(z')$
- Weight update rule in (**):
 - can be implemented using stochastic gradient descent.

Experiments (MT03 Arabic-English DARPA evaluation)

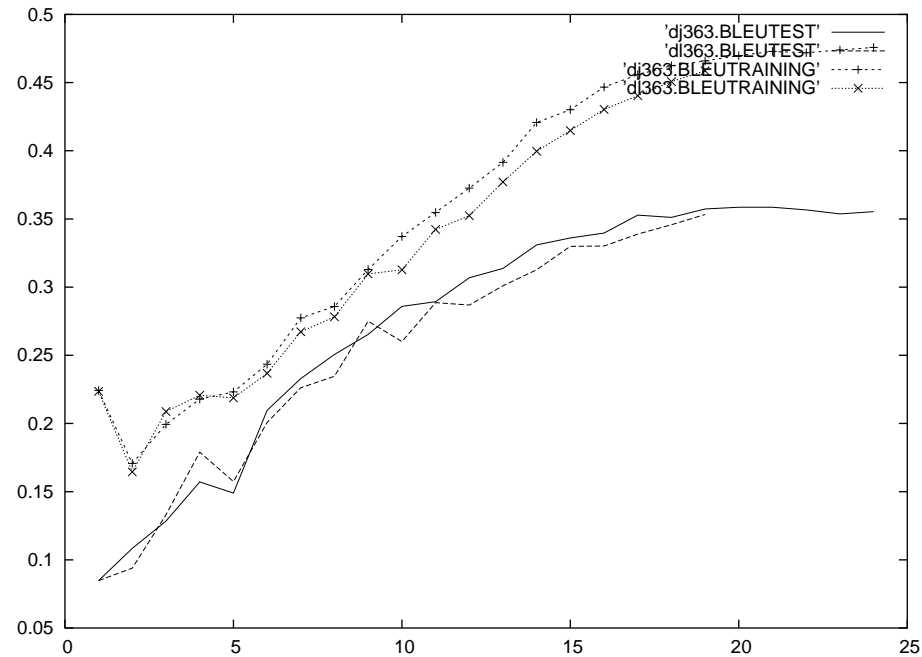


Figure 1: BLEU score on test and training data as a function of the number of training iterations.

Translation results (25 training iterations)

Model	training	test
MON-PHR	0.3661	0.261
MON	0.4773	0.359
SWAP	0.4741	0.362

- Using only simple binary-features without language models, etc (never done before in SMT)
 - MON-PHR: phrase id based features;
 - MON/SWAP: plus internal word features with and without swapping.
- Competitive with results using similar features hand tuned for decoder.
 - many published grammar based methods, in the .20s.
 - best published result around .45 (but with many additional engineering tunings and features).

Conclusion

- Method to handle large search space for arbitrary decoding procedure
 - key: restrict search space dependency
 - * through a loss function of the form $\max_{z' \in V(S)-A} \phi(z', \dots)$.
 - z' achieved at a small relevant set (can ignore the non-relevant points)
 - using decoder output to approximate relevant set.
 - * automatic construction of most relevant alternatives.
- Demonstrate that large scale ranking system can be effectively learned.
- Critical idea:
 - importance sampling (relevant set).
 - analogous to importance weighting in web-search.