



Learning to Rank for Information Retrieval

Tie-Yan Liu

Lead Researcher

Microsoft Research Asia

The Speaker

- Tie-Yan Liu
 - Lead Researcher, Microsoft Research Asia.
 - Research Interests: Learning to rank for information retrieval, and large scale machine learning.
 - ~70 Papers: *SIGIR*(9), *WWW*(3), *ICML* (3), *KDD*(2), etc.
 - Senior PC Member of *SIGIR* 2008.
 - Editorial Board Member of *Information Retrieval*.
 - Co-chair, *SIGIR* Workshop on Learning to Rank, 2007 & 2008.
 - Tutorial Speaker, *WWW 2008*, *AIRS* 2008, etc.

This Tutorial

- Learning to rank for information retrieval
 - But not ranking problems in other fields.
- Supervised learning
 - But not unsupervised or semi-supervised learning.
- Learning in vector space
 - But not on graphs or other structured data.
- Mainly based on papers at SIGIR, WWW, ICML, and NIPS.
 - Papers at other conferences and journals might not be covered comprehensively.
- Downloadable at
 - <http://research.microsoft.com/users/tyliu/>

Background Knowledge Required

- Information Retrieval.
 - Machine Learning.
 - Probability Theory.
 - Linear Algebra.
 - Optimization.
-
- *You are assumed to be familiar with these fields, and no comprehensive introduction to them will be given in this tutorial ☺.*

Outline

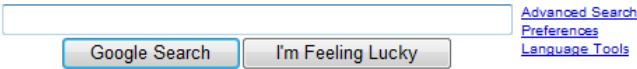
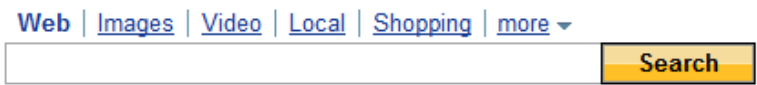
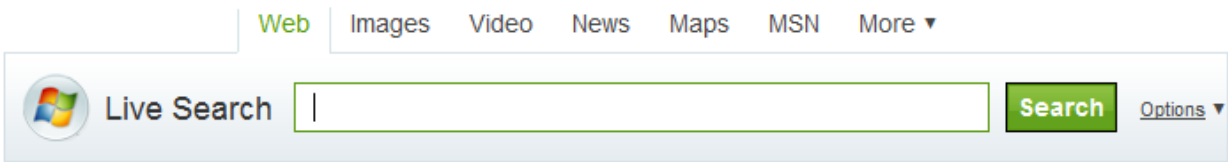
- Ranking in IR
- Learning to Rank for IR
 - Conventional machine learning approach
 - Ordinal regression: a pointwise approach
 - Preference learning: a pairwise approach
 - Listwise ranking: a listwise approach
- Advanced Topics
 - Ranking functions
- Appendix

Ranking in IR

Information is Nothing without Retrieval

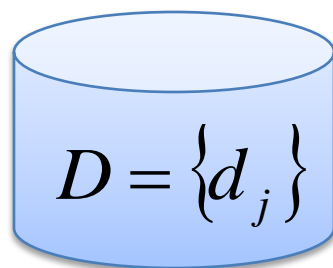


Search Engine as A Tool



Ranking inside Search Engine

Indexed Document Repository



Ranked List of Documents

A blue box with a scroll effect representing the Ranked List of Documents. It contains the following text:
 $d_1 = \text{www.sigir2008.org}$
 $d_2 = \text{research.microsoft.com/users/lr4ir - 2008/}$
 \vdots
 \vdots
 $d_M = \text{www.yr-bcn.es/sigir08/}$

Query

$q = \{SIGIR2008\}$

A blue rectangular box labeled "Ranking model".

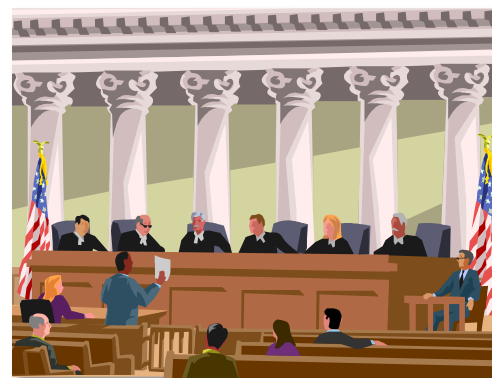
Evaluation of Ranking

- A standard test set
 - Containing a large number of (randomly sampled) queries, their associated documents, and relevance judgment for each query-document pair.
 - How to sample this set? 😊
- A metric
 - Evaluating the effectiveness of a ranking model for a particular query.
 - Averaged over the entire test set to represent the expected effectiveness of the ranking model.

Widely-used Judgment

- Pointwise
 - Binary judgment
 - Relevant vs. Irrelevant
 - Multi-valued disjunctive judgment
 - Perfect > Excellent > Good > Fair > Poor
- Pairwise
 - Pairwise preference
 - Document A is more relevant than document B w.r.t. query q
- Listwise
 - Partial or total orders
 - Could be mined from click-through logs

There are always unjudged documents. We usually regard them as irrelevant; otherwise hole filling is needed.



Rank-Based Evaluation Measures

- Winners Take All (WTA)
- Mean Reciprocal Rank (MRR)
- Mean Average Precision (MAP)
- Normalized Discounted Cumulative Gain (NDCG)
- Rank Correlation (RC)
-

WTA

- For query q , if top ranked document is relevant:
 $WTA(q) = 1$; otherwise $WTA(q) = 0$.
- Do not care about other documents.
- Averaged over all queries.

MRR

- For query q , rank position of the first relevant document is denoted as $R(q)$. Documents ranked below $R(q)$ are not considered.
- $1/R(q)$ is used as the measure for query q .
- Averaged over all queries.

MAP

- Precision at position n for query q :

$$P @ n = \frac{\#\{\text{relevant documents in top } n \text{ results}\}}{n}$$

- Average precision for query q :

$$AP = \frac{\sum_n P @ n \cdot I\{\text{document } n \text{ is relevant}\}}{\#\{\text{relevant documents}\}}$$



$$AP = \frac{1}{3} \cdot \left(\frac{1}{1} + \frac{2}{3} + \frac{3}{5} \right) \approx 0.76$$

- MAP: averaged over all queries.

NDCG

- NDCG at position n for query q :

$$NDCG @ n = \underbrace{Z_n}_{\text{Normalization}} \underbrace{\sum_{j=1}^n}_{\text{Cumulating}} \underbrace{(2^{c(j)} - 1)}_{\text{Gain}} \underbrace{/\log(1+j)}_{\text{Position discount}}$$

- Averaged over all queries.

NDCG

Query={abc}

Gain according to labels of the documents

	URL	Gain
#1	http://abc.go.com/	Perfect: $31=2^5-1$
#2	http://www.abcteach.com/	Fair: $3=2^2-1$
#3	http://abcnews.go.com/sections/	Excellent: $15=2^4-1$
#4	http://www.abc.net.au/	Excellent: 15
#5	http://abcnews.go.com/	Excellent: 15

NDCG

Query={abc}

Cumulative gain if every document is equally important to users

	URL	Gain	CG
#1	http://abc.go.com/	Perfect: $31=2^5-1$	31
#2	http://www.abcteach.com/	Fair: $3=2^2-1$	$34 = 31 + 3$
#3	http://abcnews.go.com/sections/	Excellent: $15=2^4-1$	$49 = 31 + 3 + 15$
#4	http://www.abc.net.au/	Excellent: 15	$64 = 31 + 3 + 15 + 15$
#5	http://abcnews.go.com/	Excellent: 15	$79 = 31 + 3 + 15 + 15 + 15$

NDCG

Query={abc}

Discounting factor: $\text{Log}(2) / (\text{Log}(1+\text{rank}))$

	URL	Gain	DCG
#1	http://abc.go.com/	Perfect: $31=2^5-1$	$31 = 31 \times 1$
#2	http://www.abcteach.com/	Fair: $3=2^2-1$	$32.9 = 31 + 3 \times 0.63$
#3	http://abcnews.go.com/sections/	Excellent: $15=2^4-1$	$40.4 = 32.9 + 15 \times 0.50$
#4	http://www.abc.net.au/	Excellent: 15	$46.9 = 40.4 + 15 \times 0.43$
#5	http://abcnews.go.com/	Excellent: 15	$52.7 = 46.9 + 15 \times 0.39$

NDCG

Query={abc}

Sort documents according to their labels, get max DCG

	URL	Gain	DCG	Max DCG
#1	http://abc.go.com/	Perfect: $31=2^5-1$	31	$31 = 31 \times 1$
#2	http://www.abcteach.com/	Fair: $3=2^2-1$	32.9	$40.5 = 31 + 15 \times 0.63$
#3	http://abcnews.go.com/sections/	Excellent: $15=2^4-1$	40.4	$48.0 = 40.5 + 15 \times 0.50$
#4	http://www.abc.net.au/	Excellent: 15	46.9	$54.5 = 48.0 + 15 \times 0.43$
#5	http://abcnews.go.com/	Excellent: 15	52.7	$60.4 = 54.5 + 15 \times 0.39$

NDCG

Query={abc}

Normalize DCG with max DCG

	URL	Gain	DCG	Max DCG	NDCG
#1	http://abc.go.com/	Perfect: $31=2^5-1$	31	31	$1 = 31/31$
#2	http://www.abcteach.com/	Fair: $3=2^2-1$	32.9	40.5	$0.81=32.9/40.5$
#3	http://abcnews.go.com/sections/	Excellent: $15=2^4-1$	40.4	48.0	$0.84=40.4/48.0$
#4	http://www.abc.net.au/	Excellent: 15	46.9	54.5	$0.86=46.9/54.5$
#5	http://abcnews.go.com/	Excellent: 15	52.7	60.4	$0.87=52.7/60.4$

Rank Correlation

- For query q , given the predicted ranked list and the ground-truth ranked list, the correlation between the two lists is used as the measure.
 - For example, weighted Kendall's τ can be used.

$$K(q) = \frac{\sum_{u,v} w_{u,v} (1 + \text{sgn}((\pi_u - \pi_v)(\sigma_u - \sigma_v)))}{2 \sum_{u,v} w_{u,v}}$$

Position-based weights

- Averaged over all queries.

More about Evaluation Measures

- Query-level
 - Every query contributes somehow equally to the measure.
 - Averaged over all testing queries.
 - Bounded for each query.
- Rank-based
 - Rank position is explicitly used.
 - Non-smooth and non-differentiable w.r.t. scores
 - With small changes to the scores, the ranks will not change until one document's score passes another.

Conventional Ranking Models

- Content relevance
 - Boolean model, extended Boolean model, etc.
 - Vector space model, latent semantic indexing (LSI), etc.
 - BM25 model, statistical language model, etc.
 - Span based model, distance aggregation model, etc.
- Page importance
 - Link analysis: HITS, PageRank, TrustRank, etc.
 - Log mining: DirectHITS, BrowseRank, etc.

More About Conventional Ranking Models

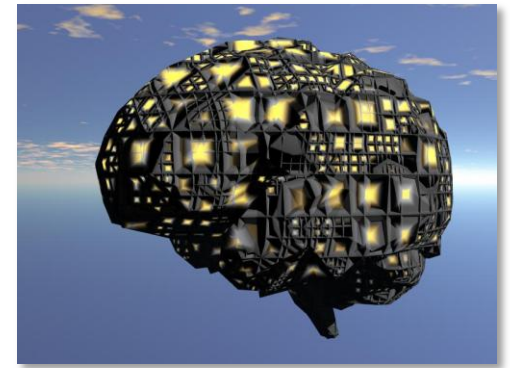
- Please refer to the following tutorials:
 - [The Probabilistic Relevance Model: BM25 and Beyond](#)
(*H. Zaragoza and S. Robertson*)
 - [A Tutorial to Formal Models of Information Retrieval](#)
(*D. Hiemstra*)

Discussions on Conventional Ranking Models

- For a particular model
 - Manual parameter tuning is usually difficult, especially when there are many parameters.
- For comparison between two models
 - Given a test set, it is difficult / unfair to compare two models if one is over-tuned while the other is not.
- For a collection of models
 - There are hundreds of models proposed in the literature.
 - It is non-trivial to combine them to produce a even more effective model.

Machine Learning Can Help

- Machine learning is an effective tool
 - To automatically tune parameters.
 - To combine multiple evidences.
 - To avoid over-fitting (by means of regularization, etc.)
- **“Learning to Rank”**
 - Use machine learning technologies to train the ranking model.
 - A hot research topic these years.



Learning to Rank Algorithms

Overview

Conventional ML Approach

Ordinal Regression: A Pointwise Approach

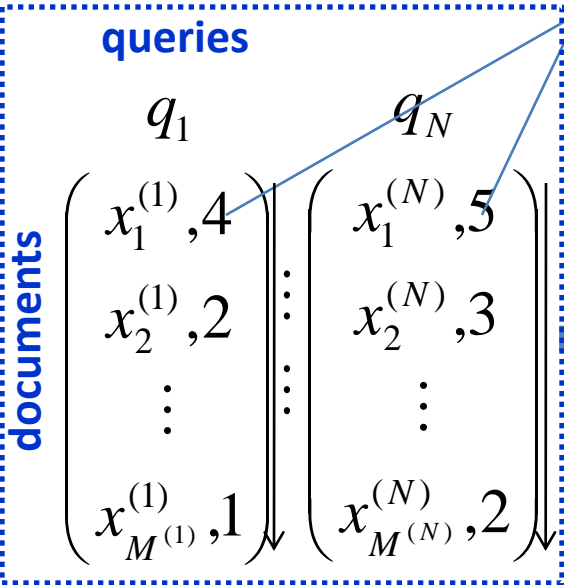
Preference Learning: A Pairwise Approach

Listwise Ranking: A Listwise Approach

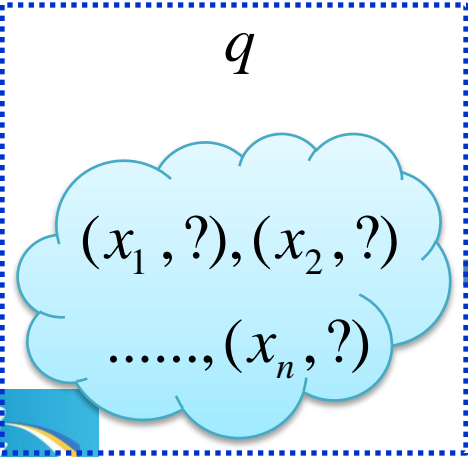
Framework of Learning to Rank

Labels, refer to the judgments in IR evaluation.

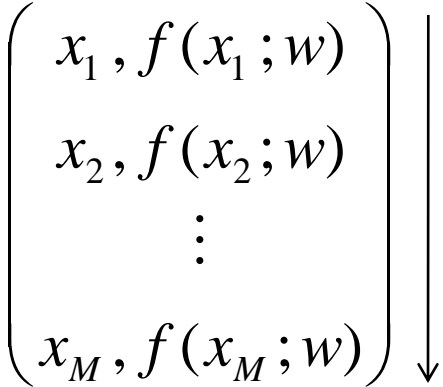
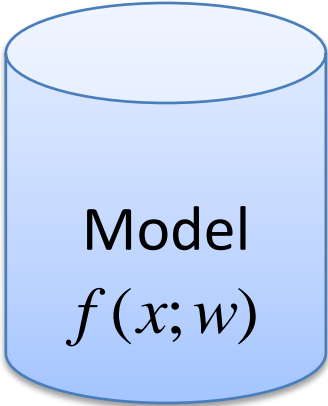
How to sample the most appropriate training set?
How to extract the most useful features?



Training Data



Test data



Learning to Rank Algorithms

- Least Square Retrieval Function (TOIS 1989)
- Query refinement (WWW 2008)
- ListNet (ICML 2007)
- SVM-MAP (SIGIR 2007)
- Nested Ranker (SIGIR 2006)
- LambdaRank (NIPS 2006)
- Pranking (NIPS 2002)
- MPRank (ICML 2007)
- Frank (SIGIR 2007)
- MHR (SIGIR 2007)
- RankBoost (JMLR 2003)
- Learning to retrieval info (SCC 1995)
- Large margin ranker (NIPS 2002)
- LDM (SIGIR 2005)
- RankNet (ICML 2005)
- Ranking SVM (ICANN 1999)
- IRSVM (SIGIR 2006)
- Discriminative model for IR (SIGIR 2004)
- SVM Structure (JMLR 2005)
- OAP-BPM (ICML 2003)
- Subset Ranking (COLT 2006)
- GPRank (LR4IR 2007)
- QBRank (NIPS 2007)
- GBRank (SIGIR 2007)
- Constraint Ordinal Regression (ICML 2005)
- McRank (NIPS 2007)
- SoftRank (LR4IR 2007)
- AdaRank (SIGIR 2007)
- CCA (SIGIR 2007)
- ListMLE (ICML 2008)
- RankCosine (IP&M 2007)
- Supervised Rank Aggregation (WWW 2007)
- Relational ranking (WWW 2008)
- Learning to order things (NIPS 1998)
- Round robin ranking (ECML 2003)



Categorization of the Algorithms

- Conventional machine learning
 - Classification / Regression.
- Beyond conventional machine learning
 - Ordinal regression (*a pointwise approach*).
 - Preference learning (*a pairwise approach*).
 - Listwise ranking (*a listwise approach*).

Categorization of the Algorithms

Category	Algorithms
Conventional ML Approach	Least Square Retrieval Function (TOIS 1989), Regression Tree for Ordinal Class Prediction (Fundamenta Informaticae, 2000), Discriminative model for IR (SIGIR 2004), Subset Ranking using Regression (COLT 2006), McRank (NIPS 2007), ...
Pointwise Approach	Pranking (NIPS 2002), OAP-BPM (EMCL 2003), Ranking with Large Margin Principles (NIPS 2002), Constraint Ordinal Regression (ICML 2005), ...
Pairwise Approach	Learning to Retrieve Information (SCC 1995), Learning to Order Things (NIPS 1998), Ranking SVM (ICANN 1999), RankBoost (JMLR 2003), LDM (SIGIR 2005), RankNet (ICML 2005), Frank (SIGIR 2007), MHR(SIGIR 2007), Round Robin Ranking (ECML 2003), GBRank (SIGIR 2007), QBRank (NIPS 2007), MPRank (ICML 2007), IRSVM (SIGIR 2006), ...
Listwise Approach	LambdaRank (NIPS 2006), AdaRank (SIGIR 2007), SVM-MAP (SIGIR 2007), SoftRank (LR4IR 2007), GPRank (LR4IR 2007), CCA (SIGIR 2007), RankCosine (IP&M 2007), ListNet (ICML 2007), ListMLE (ICML 2008), ...

Notations

Feature vector of document j associated with query i : d_j^i	$x_j^{(i)} = (x_{j,1}^{(i)}, \dots, x_{j,T}^{(i)})$
Relevance score given by the scoring function	$f(x)$
Ground truth label for documents	y
Mapping function	ψ
Transformation function	ϕ
Number of documents w.r.t. a query q	M
Number of training queries	N

Learning to Rank Algorithms

Overview

Conventional ML Approach

Ordinal Regression: A Pointwise Approach

Preference Learning: A Pairwise Approach

Listwise Ranking: A Listwise Approach

Conventional ML Approach

- Directly apply regression or classification methods to solve the problem of ranking
 - Regard binary judgments or multi-valued discrete as “non-ordered” categories, or real values.
 - Although ground truths are neither “non-ordered” categories nor real values 😊.

Least Square Retrieval Function

(N. Fuhr, TOIS 1989)

- Relevance judgment for a query-document pair is represented by a vector:
 - For binary judgment: $y=(1, 0)$ or $(0, 1)$
 - For multi-valued discrete: $y=(1,0,0)$, $(0,1,0)$, or $(0,0,1)$
- Use polynomial function as the ranking function $f(x)$.
- Use least square error (LSE) method to learn the regression function

$$\min \sum_{i=1}^N \sum_{j=1}^{M^{(i)}} \left| y_j^{(i)} - f(x_j^{(i)}) \right|^2$$

Regression Tree for Ordinal Class Prediction

(S. Kramer, Fundamenta Informaticae, 2000)

- Simply run a regression algorithm like S-CART as if the ordinal classes were real values.
- Post-process the regression tree in order to translate real-valued predictions into discrete class labels.
 - Round to the nearest of the ordinal classes.
 - Modify the way S-CART computes the target values during tree construction (e.g., always predict integer values).

Discriminative Model for IR

(R. Nallapati, SIGIR 2004)

- Treat relevant documents as positive examples, while irrelevant documents as negative examples.
- Learning algorithms used:

- Max Entropy

$$P(R | d_j, q) = \frac{1}{Z(d_j, q)} \exp \left(\sum_{t=1}^T \lambda_{t,R} x_{j,t} \right)$$

$$f(x_j) = \log \frac{P(R | d_j, q)}{P(\bar{R} | d_j, q)} = \sum_{t=1}^T (\lambda_{t,R} - \lambda_{t,\bar{R}}) x_{j,t}$$

- Support Vector Machines

$$f(x) = w \cdot x + b$$

Subset Ranking using Regression

(D. Cossock and T. Zhang, COLT 2006)

- Ranked based loss in terms of DCG is upper bounded by regression error.

$$DCG(r_{f_B}) - DCG(r_f) \leq \left(2 \sum_{j=1}^M c_j^p \right)^{1/p} \left(\sum_{j=1}^M (f(x_j) - f_B(x_j))^q \right)^{1/q}$$

- Regression is used to learn the ranking function.

$$\hat{f} = \arg \min \sum_{i=1}^N \sum_{j=1}^{M^{(i)}} (f(x_j^{(i)}) - y_j^{(i)})^2$$

McRank

(P. Li, C. Burges, et al. NIPS 2007)

- Loss in terms of DCG is upper bounded by multi-class classification error.

$$DCG(y) - DCG(r_f) \leq 15\sqrt{2} \left(\sum_{j=1}^M c_j^2 - M \sum_{j=1}^M c_j^{2/M} \right)^{1/2} \left(\sum_{j=1}^M 1_{y_j \neq f(x_j)} \right)^{1/2}$$

- Multi-class classification: $\hat{p}_{j,k} = P(y_j = k)$, $s_j = \sum_{k=0}^{K-1} \hat{p}_{j,k} k$
- Multiple ordinal classification: $P(y_j \leq k)$
- Regression: $\hat{f} = \arg \min \sum_{i=1}^N \sum_{j=1}^{M^{(i)}} \left| f(x_j^{(i)}) - 2^{y_j^{(i)} - 1} \right|$

Discussions

- Cannot handle ground truths of pairwise preference and partial (total) orders.
- Absolute and independent relevance is assumed.
 - Might not be true in ranking.
 - Relevance is relative and defined only among documents associated with the same query: an irrelevant document for a popular query might have higher term frequency than a relevant document for a rare query.
- Unique properties of ranking (and particularly, ranking for IR) have not been well considered.

Unique Properties of Ranking for IR

- Query-level
 - Query determines the logical structure of the ranking problem in IR.
 - Relevance is defined on documents associated with the same query.
 - IR evaluation measures are computed at query level.
- Rank-based
 - Learning objective is non-smooth and non-differentiable.
 - Top-ranked objects are more important.
 - Relative order is important: no need to predict accurate category, or value of $f(x)$.

Bridge the Gap

- Go beyond conventional ML methods
 - Ordinal regression (*a pointwise approach*)
 - Target the ground truth of multi-valued discrete.
 - Preference learning (*a pairwise approach*)
 - Target the ground truth of pairwise preference.
 - Also compatible with that of multi-valued discrete.
 - Listwise ranking (*a listwise approach*)
 - Target the ground truth of partial / total order.
 - Also compatible with other types of ground truths.

Learning to Rank Algorithms

Overview

Conventional ML Approach

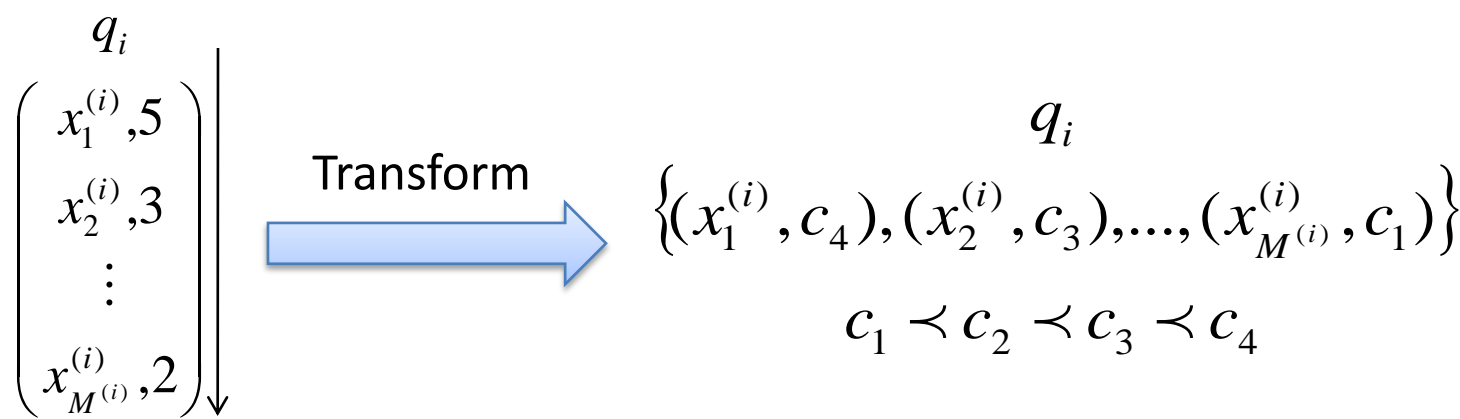
Ordinal Regression: A Pointwise Approach

Preference Learning: A Pairwise Approach

Listwise Ranking: A Listwise Approach

Ordinal Regression: A Pointwise Approach

- Input space
 - Features of a single document (w.r.t. a query): $X \in R^T$
- Output space
 - **Ordered** categories: $Y \in \{c_1 \prec c_2 \prec \dots \prec c_K\}$



Ordinal Regression vs. Regression/Classification

- Regression: Real values
 - Classification: Non-ordered categories
 - Ordinal regression: Discrete values / Ordered categories
-
- Ordinal regression can be regarded as something between regression and classification.

Pranking with Ranking

(K. Krammer and Y. Singer, NIPS 2002)

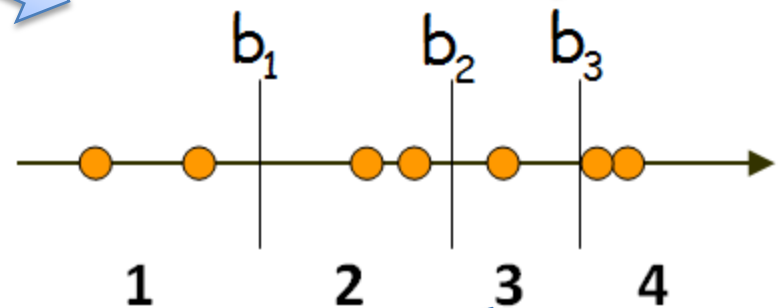
- Project

$$x \rightarrow w \cdot x$$

- Apply Thresholds

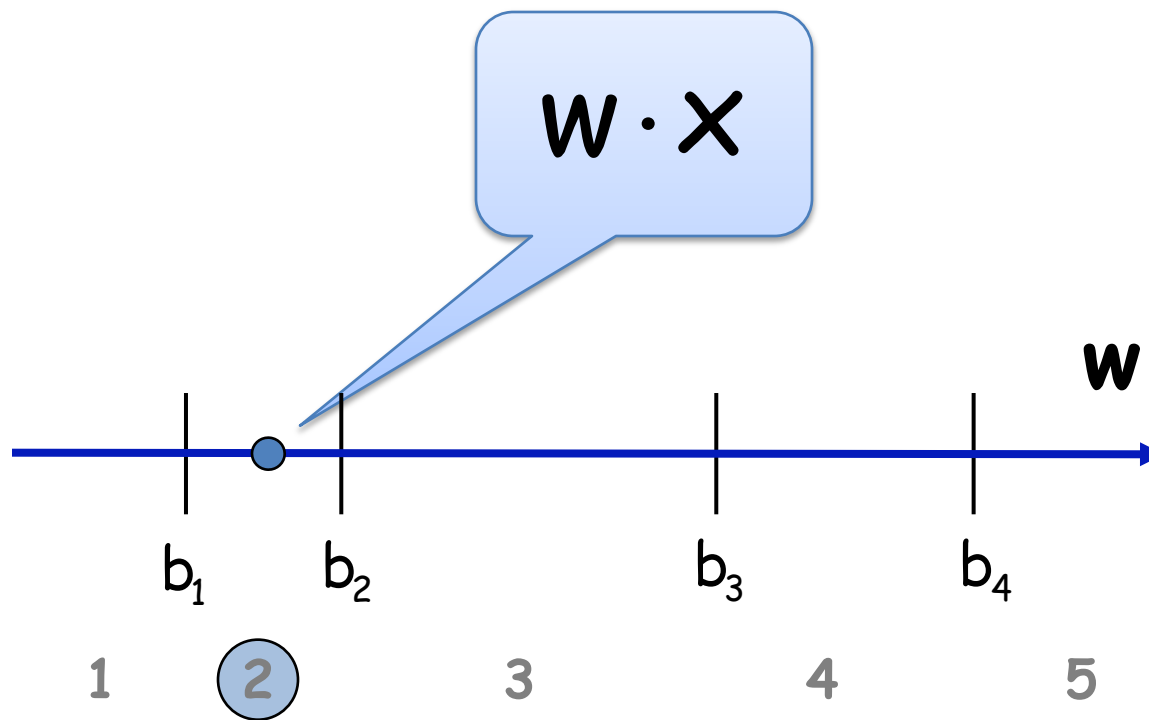
$$w \cdot x \geq 1$$

Thresholds

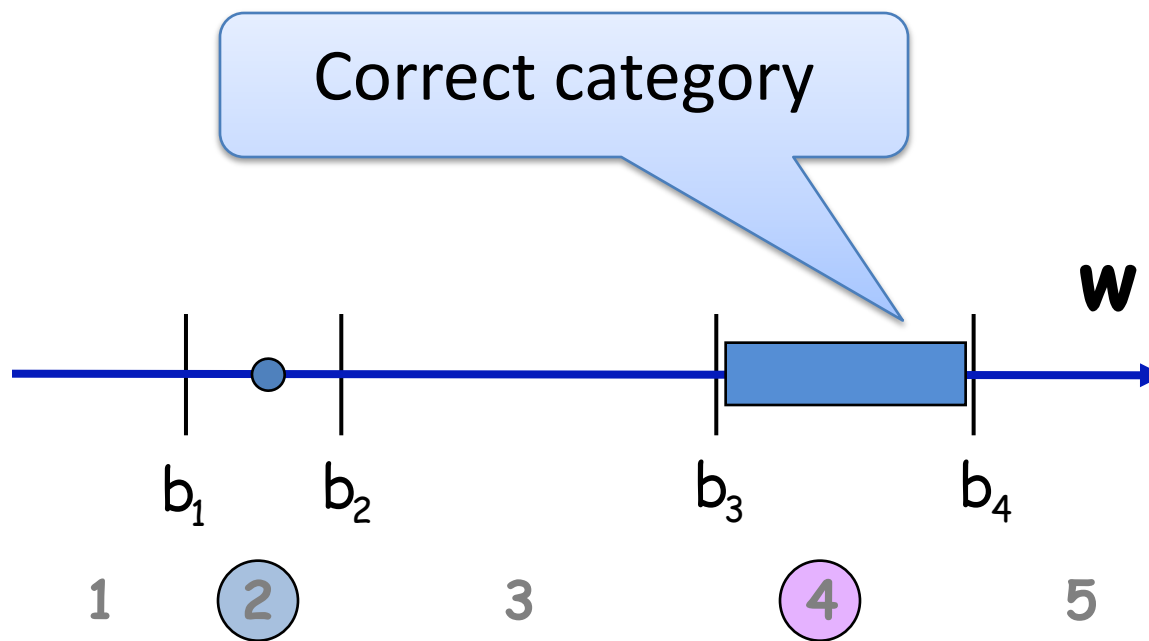


Ordered Categories

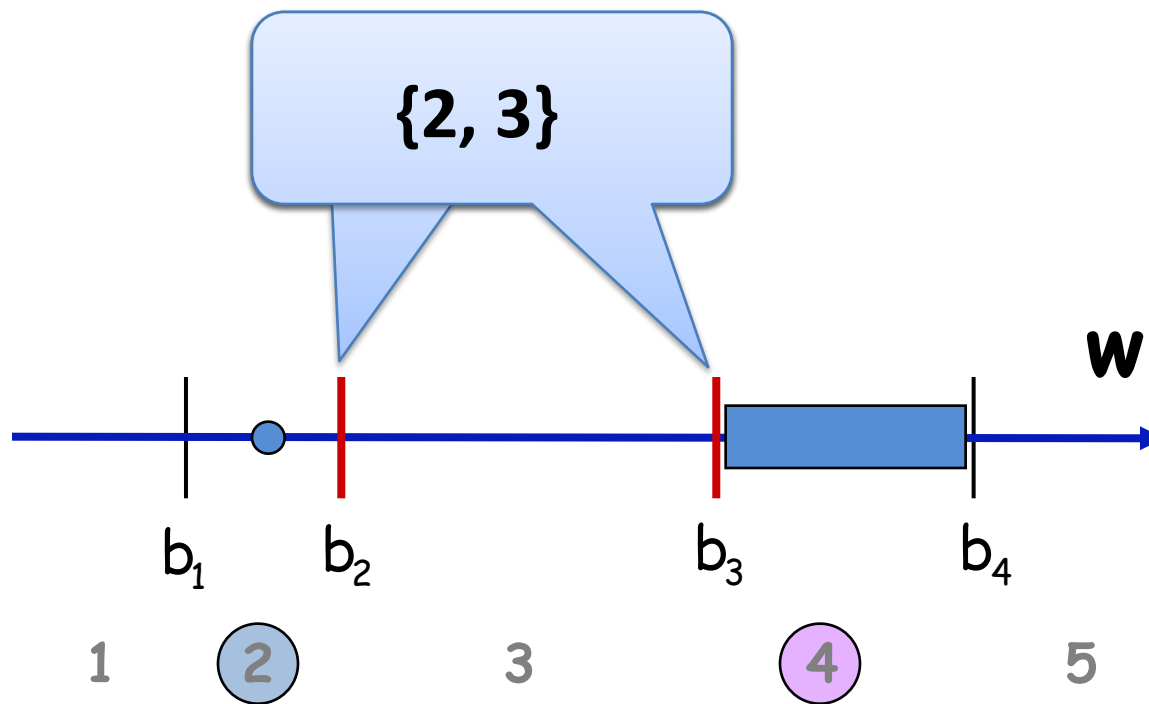
Pranking with Ranking



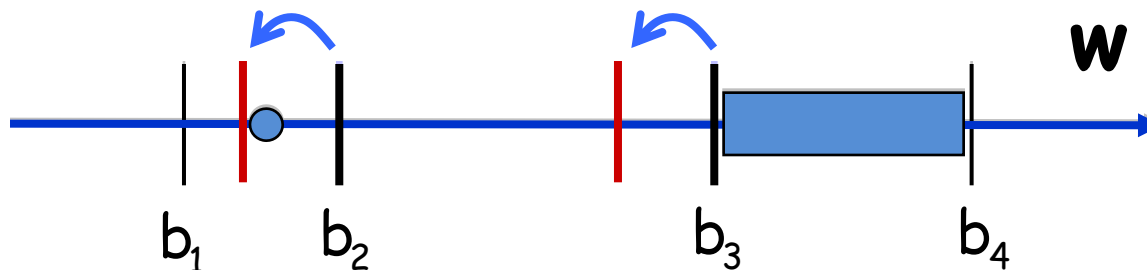
Pranking with Ranking



Pranking with Ranking

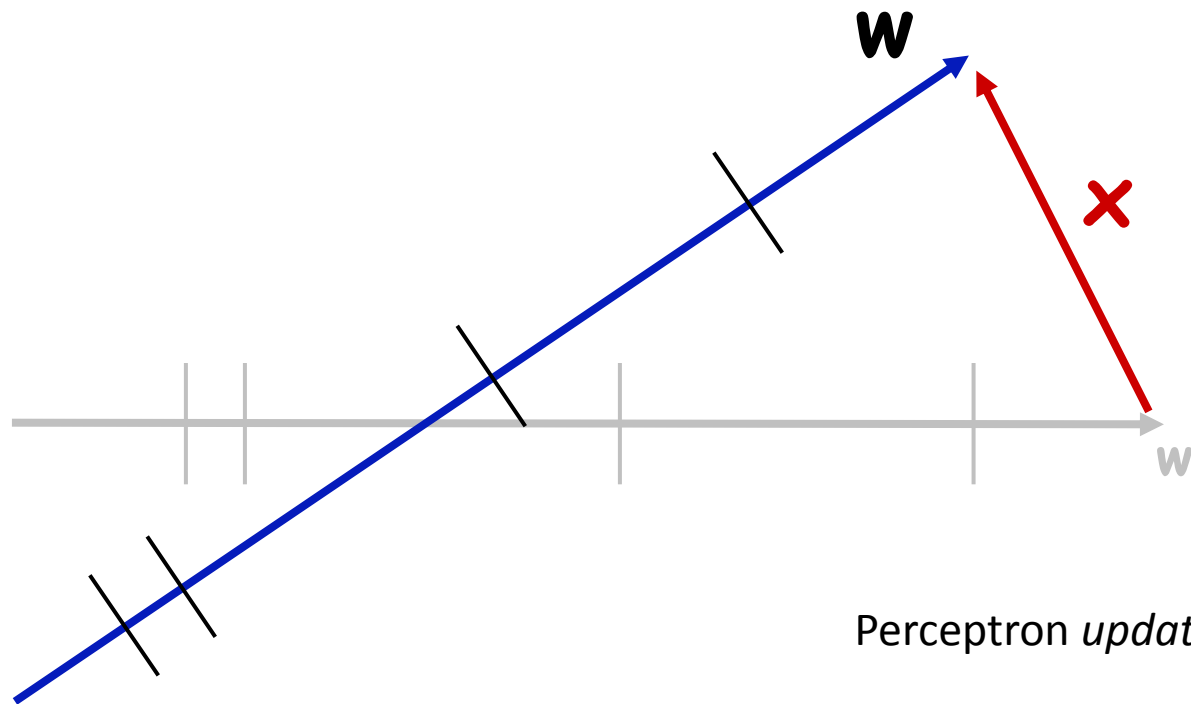


Pranking with Ranking



Threshold adjustment

Pranking with Ranking



Perceptron *update*

$$\mathbf{w}^{t+1} \leftarrow \mathbf{w}^t + \mathbf{x}^t$$

OAP-BPM

(E. Harrington, ICML 2003)

- Train several Prank models
 - Randomly subsample the incoming training examples for training each model.
- Approximate the Bayes point by averaging the weights and thresholds associated with these Prank models.
 - Bayes point is the single hypothesis chosen from a fixed class of hypotheses that achieves the minimum probability of error.
 - The average technique was proven to lead to a good approximation to the Bayes point (Herbrich, Craepel & Campbell. JMLR 2001)
 - Better generalization ability is achieved.

Ranking with Large Margin Principles

(A. Shashua and A. Levin, NIPS 2002)

- Fixed margin strategy
 - Margin defined by the closest (neighboring) pair of classes.

$$\begin{aligned} \min \quad & \frac{1}{2} \|w\|^2 + C \sum_{i=1}^N \sum_{j=1}^{M^{(i)}} \sum_{k=1}^K \left(\varepsilon_{j,k}^{(i)} + \varepsilon_{j,k+1}^{(i)*} \right) \\ \text{s.t.} \quad & w^T x_j^{(i)} - b_k \leq -1 + \varepsilon_{j,k}^{(i)}, \text{ if } y_j^{(i)} = c_k. \\ & w^T x_j^{(i)} - b_k \geq 1 - \varepsilon_{j,k+1}^{(i)*}, \text{ if } y_j^{(i)} = c_{k+1}. \\ & \varepsilon_{j,k}^{(i)} \geq 0, \varepsilon_{j,k+1}^{(i)*} \geq 0. \end{aligned}$$

Ranking with Large Margin Principles

- Sum of margins strategy
 - Margin defined as the sum of margins of all sub problems

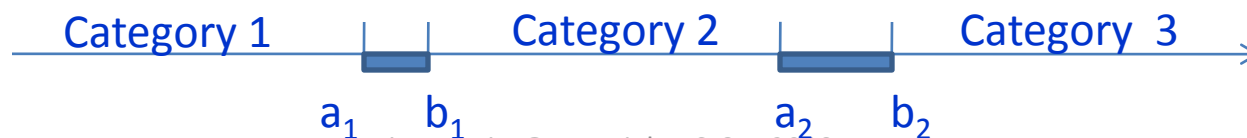
$$\min \sum_k (a_k - b_k) + C \sum_{i=1}^N \sum_{j=1}^{M^{(i)}} \sum_{k=1}^K (\varepsilon_{j,k}^{(i)} + \varepsilon_{j,k+1}^{(i)*})$$

$$\text{s.t. } a_k \leq b_k \leq a_{k+1}$$

$$w^T x_j^{(i)} \leq a_k + \varepsilon_{j,k}^{(i)}, \text{ if } y_j^{(i)} = c_k.$$

$$w^T x_j^{(i)} \geq b_k - \varepsilon_{j,k+1}^{(i)*}, \text{ if } y_j^{(i)} = c_{k+1}.$$

$$w^T w \leq 1, \varepsilon_{j,k}^{(i)} \geq 0, \varepsilon_{j,k+1}^{(i)*} \geq 0.$$



Constraint Ordinal Regression

(W. Chu, S. Keerthi, ICML 2005)

- Fix the problem of disordered threshold
 - Explicit constraints on thresholds

$$b_{k-1} \leq b_k$$

- Implicit constraints on thresholds

$$\min \frac{1}{2} \|w\|^2 + C \sum_{i=1}^N \sum_{j=1}^{M^{(i)}} \sum_{k=1}^K \left(\varepsilon_{j,k}^{(i)} + \varepsilon_{j,k+1}^{(i)*} \right)$$

$$\text{s.t. } w^T x_j^{(i)} - b_k \leq -1 + \varepsilon_{j,k}^{(i)}, \text{ if } y_j^{(i)} \preceq c_k.$$

$$w^T x_j^{(i)} - b_k \geq 1 - \varepsilon_{j,k+1}^{(i)*}, \text{ if } y_j^{(i)} \preceq c_{k+1}.$$

$$\varepsilon_{j,k}^{(i)} \geq 0, \varepsilon_{j,k+1}^{(i)*} \geq 0.$$

Redundant training examples to guarantee implicit constraints on thresholds.

Discussions

- Only slightly different from conventional machine learning methods.
- Also cannot handle ground truths of pairwise preference and partial (total) orders.
- Some of the unique properties of ranking for IR have not been considered either.

Learning to Rank Algorithms

Overview

Conventional ML Approach

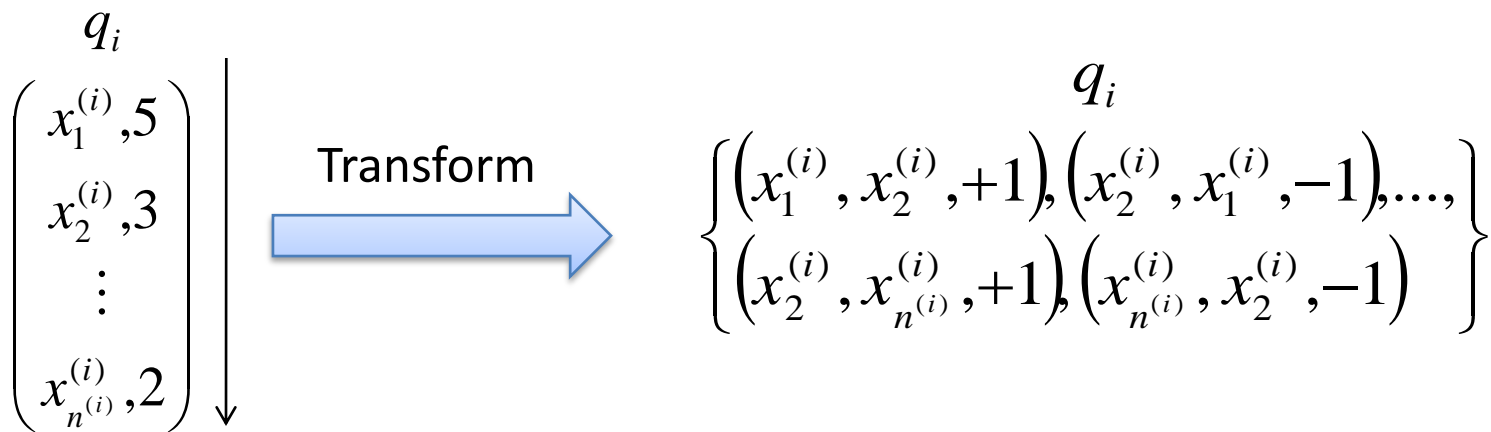
Ordinal Regression: A Pointwise Approach

Preference Learning: A Pairwise Approach

Listwise Ranking: A Listwise Approach

Preference Learning: A Pairwise Approach

- Input space
 - Document pairs: $(X_u, X_v) \in R^T \times R^T$
- Output space
 - Preference: $Y \in \{+1, -1\}$



Learning to Retrieve Information

(B. Bartell, G. W. Cottrell, et al. SCC 1995)

- A pairwise loss function based on Guttman's Point Alienation

$$- L(f) = -\frac{1}{N} \sum_{i=1}^N \frac{\sum_{x_u^{(i)} \succ x_v^{(i)}} (f(x_u^{(i)}) - f(x_v^{(i)}))}{\sum_{x_u^{(i)} \succ x_v^{(i)}} |f(x_u^{(i)}) - f(x_v^{(i)})|}$$

- Proven to be highly correlated with AP
- Gradient Descent is used for the optimization

Learning to Order Things

(W. Cohen, R. Schapire, et al. NIPS 1998)

- Pairwise ranking function

$$- f(x_u, x_v) = \sum w_t f_t(x_u, x_v)$$

- Pairwise loss function

$$- L(f) = \frac{\sum_{i=1}^N \sum_{x_u^{(i)} \succ x_v^{(i)}} (1 - f(x_u^{(i)}, x_v^{(i)}))}{\sum_{i=1}^N \sum_{x_u^{(i)} \succ x_v^{(i)}} 1}$$

- Weighted majority algorithm, e.g., the Hedge algorithm, is used to learn the parameters w from the pairwise ground truth.

Learning to Order Things

- From pairwise preference to total order
 - $\max_{\rho} AGREE(\rho, f) = \sum_{x_u, x_v: \rho(x_u) > \rho(x_v)} f(x_u, x_v)$
 - Proven: the optimal total order construction is NP hard.
- Approximation algorithm
 - A greedy ordering algorithm
 - It has been proven that the agreement for the approximation algorithm is at least half the agreement for the optimal algorithm.

RankNet

(C. Burges, T. Shaked, et al. ICML 2005)

- Target probability:
 - \bar{P}_{uv} (1 means $x_u \succ x_v$; 0 mean $x_u \prec x_v$).

- Modeled probability:
 - $P_{uv} \equiv P(d_u \succ d_v) = \frac{\exp(o_{uv})}{1 + \exp(o_{uv})}$
 - Where $o_{uv} = f(x_u) - f(x_v)$

- Cross entropy as the loss function

$$L_{uv} \equiv -\bar{P}_{uv} \log P_{uv} - (1 - \bar{P}_{uv}) \log(1 - P_{uv})$$

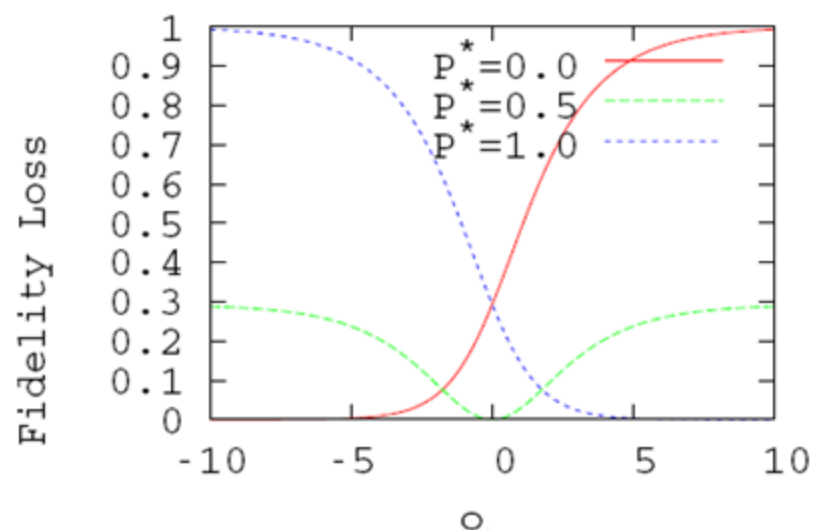
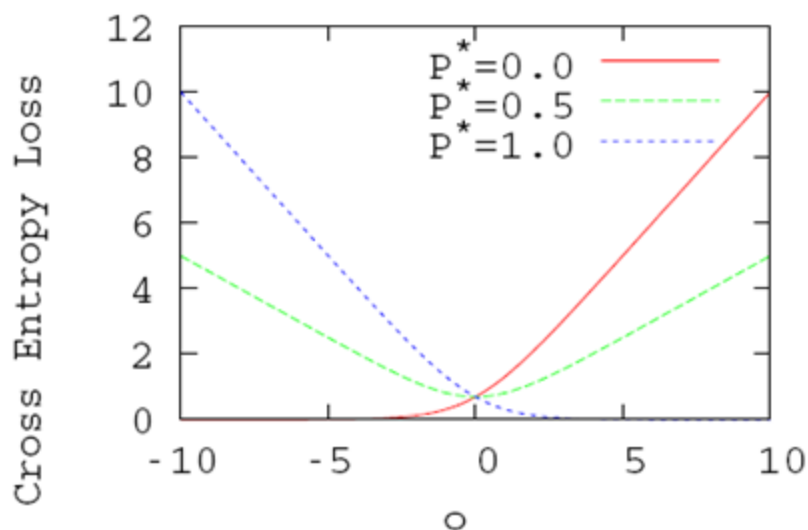
- Use Neural Network as model, and gradient descent as algorithm, to optimize the cross-entropy loss.

FRank

(M. Tsai, T. Liu, et al. SIGIR 2007)

- Similar setting to that of RankNet
- New loss function: fidelity

$$L_{uv} \equiv 1 - (\sqrt{\bar{P}_{uv} \cdot P_{uv}} + \sqrt{(1 - \bar{P}_{uv}) \cdot (1 - P_{uv})})$$



FRank (2)

RankNet (Cross entropy loss)



- Non-zero minimum loss
- Unbounded



- Convex

FRank (Fidelity loss)



- Zero minimum loss
- Bounded within $[0,1]$



- Non-convex

RankBoost

(Y. Freund, R. Iyer, et al. JMLR 2003)

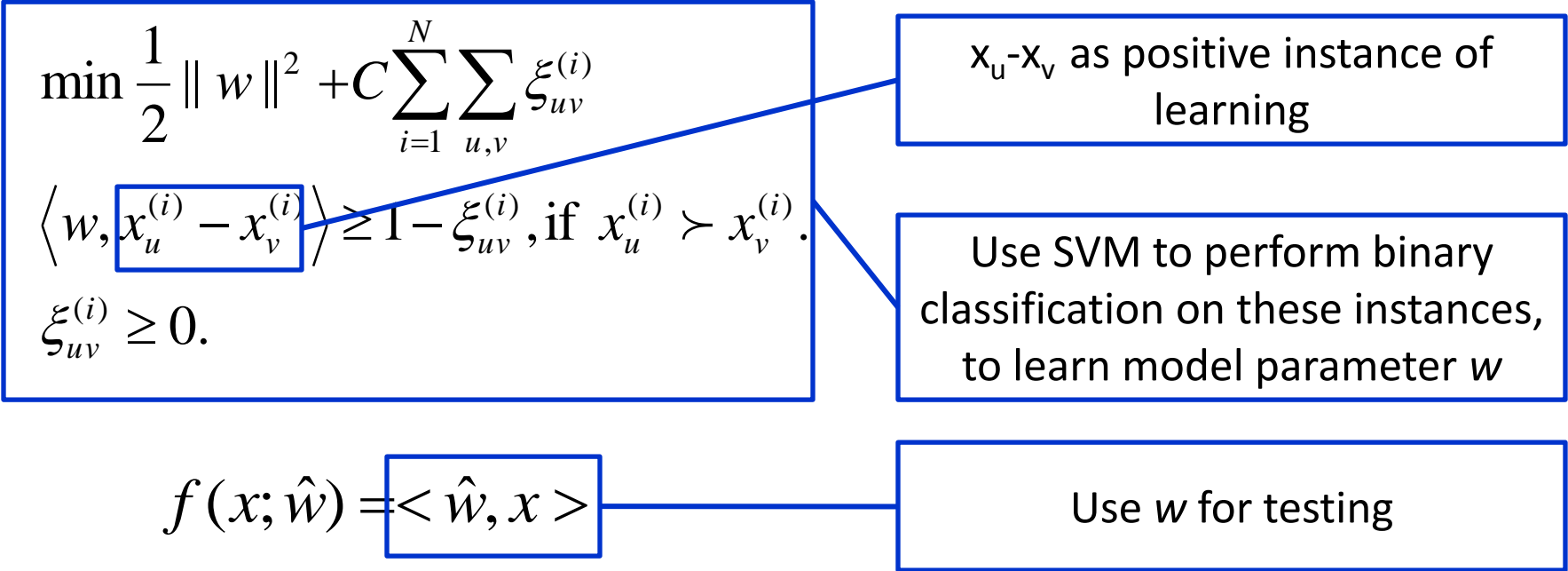
- Given: initial distribution D_1
- For $t=1, \dots, T$:
 - Train weak learner using distribution D_t
 - Get weak ranker $h_t: X \rightarrow R$
 - Choose $\alpha_t \in R$
 - Update: $D_{t+1}(x_u, x_v) = \frac{1}{Z_t} D_t(x_u, x_v) \exp(\alpha_t (h_t(x_u) - h_t(x_v)))$
where $Z_t = \sum_{x_u, x_v} D_t(x_u, x_v) \exp(\alpha_t (h_t(x_u) - h_t(x_v)))$
- Output the final ranking: $f(x) = \sum_t \alpha_t h_t(x)$

Use AdaBoost to perform pairwise classification

Ranking SVM

(R. Herbrich, T. Graepel, et al. , Advances in Large Margin Classifiers, 2000; T. Joachims, KDD 2002)

Formally discussed that ordinal regression can be solved by pairwise preference learning



Use SVM to perform the pairwise classification

From Binary to Multi-valued

- When constructing document pairs, category information has been lost: different pairs of labels are treated identically.
 - Can we maintain more information about ordered category?
 - Can we use different learner for different kinds of pairs?
- Solutions
 - Multi-hyperplane ranker

Multi-Hyperplane Ranker

(T. Qin, T. Liu, et al. SIGIR 2007)

- If we have K categories, then will have $K(K-1)/2$ pairwise preferences between two labels.
 - Learn a model f_{uv} for the pair of label r_u and label r_v , using any previous method (for example Ranking SVM).
- Testing
 - Rank Aggregation (Weighted BordaCount)

$$f(x) = \sum_{u,v} \alpha_{uv} f_{uv}(x)$$

- Weights are learned from a validation set.

Other Algorithms in This Category

- Linear discriminant model for information retrieval (LDM), (J. Gao, H. Qi, et al. SIGIR 2005)
- Preference Learning with Gaussian Process (W. Chu, and Z. Ghahramani , ICML 2005).
- A Regression Framework for Learning Ranking Functions using Relative Relevance Judgments (GBRank), (Z. Zheng, H. Zha, et al. SIGIR 2007)
- A General Boosting Method and its Application to Learning Ranking Functions for Web Search (QBRank), (Z. Zheng, H. Zha, et al. NIPS 2007)
- Magnitude-preserving Ranking Algorithms (MPRank), (C. Cortes, M. Mohri, et al. ICML 2007)
-

Discussions

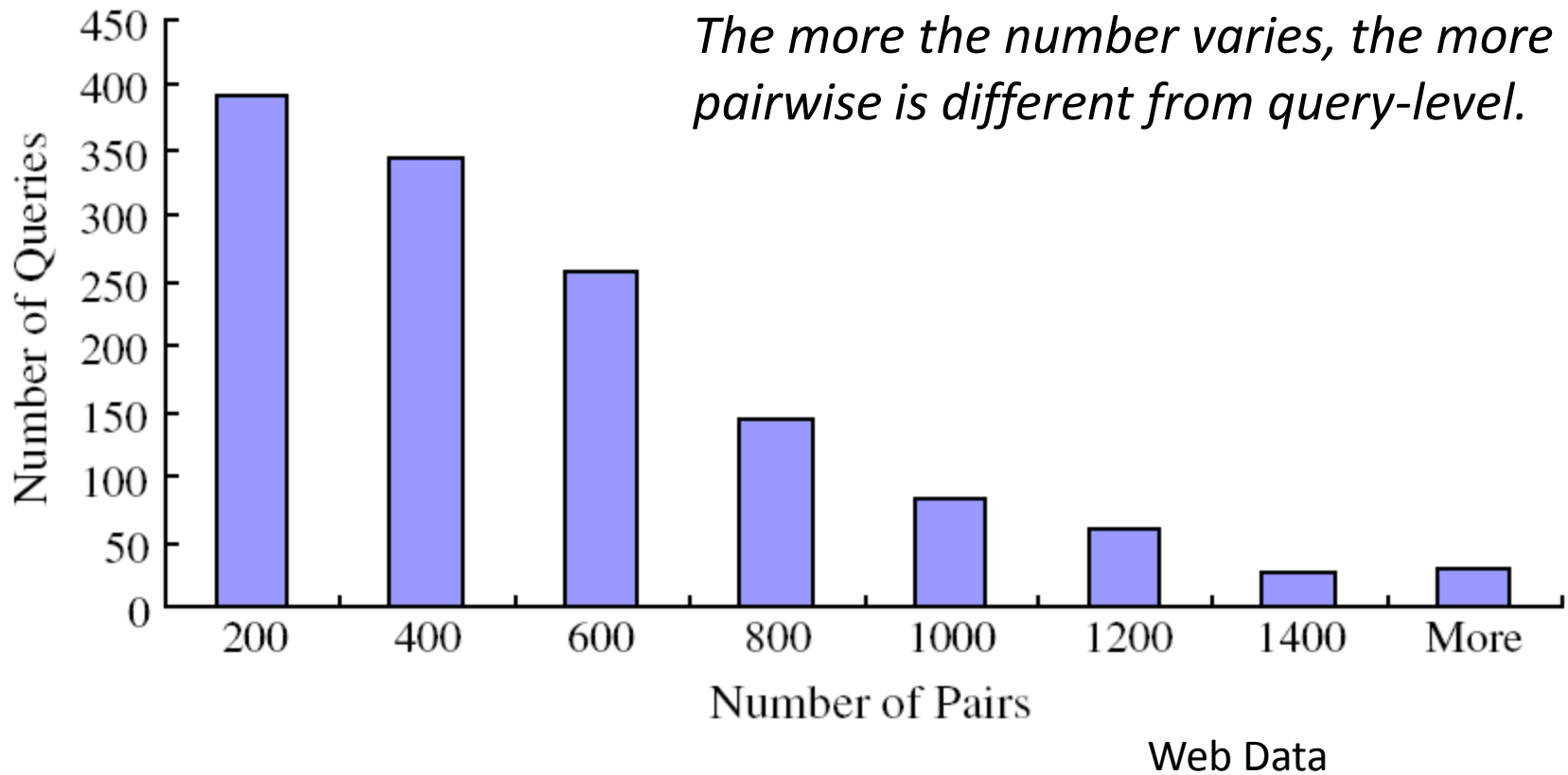
- Progress has been made as compared with the pointwise approach:
 - No longer assuming absolute relevance.
 - Using pairwise relationship to represent relative ranking.
- However,
 - Cannot handle ground truth in terms of partial / total orders.
 - Some unique properties of ranking in IR have not been considered yet.

Case Study: Pairwise vs. Query-level

- Number of document pairs varies according to queries.
 - Two queries in total
 - Same error in terms of pairwise classification $780/790 = 98.73\%$.
 - Different errors in terms of query-level evaluation **99%** vs. **50%**.

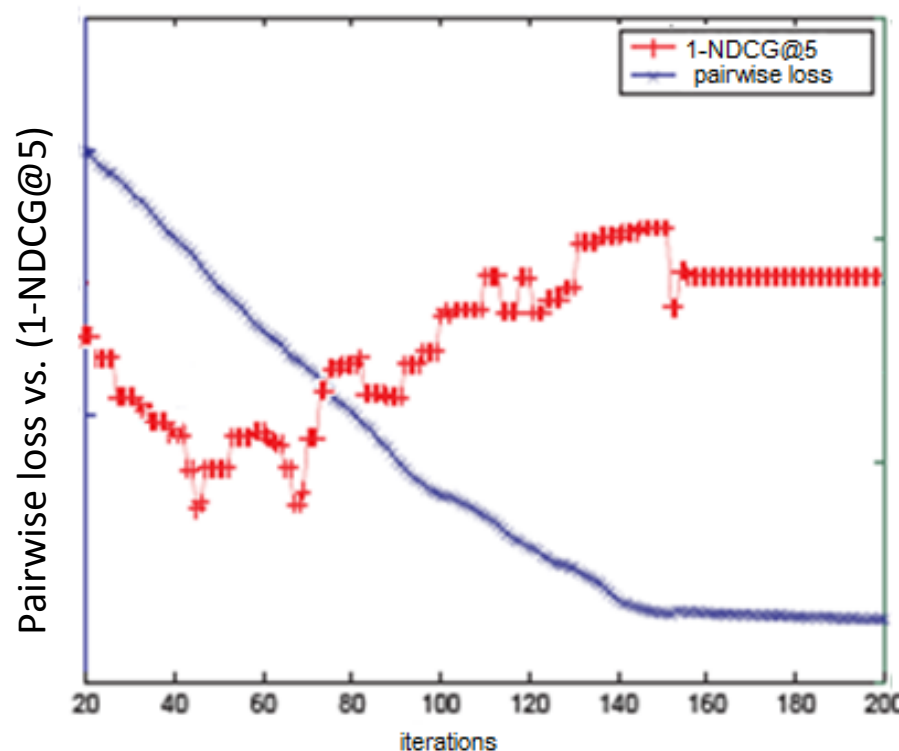
		Model 1	Model 2
Document pairs of q_1	correctly ranked	770	780
	wrongly ranked	10	0
	Accuracy	98.72%	100%
Document pairs of q_2	correctly ranked	10	0
	wrongly ranked	0	10
	Accuracy	100%	0%
overall accuracy	document level	98.73%	98.73%
	query level	99.36%	50%

Distribution of Pair Numbers in Real Dataset



As a Result, ...

- It is not clear how pairwise loss correlates with query-level IR evaluation measures.



TREC Dataset

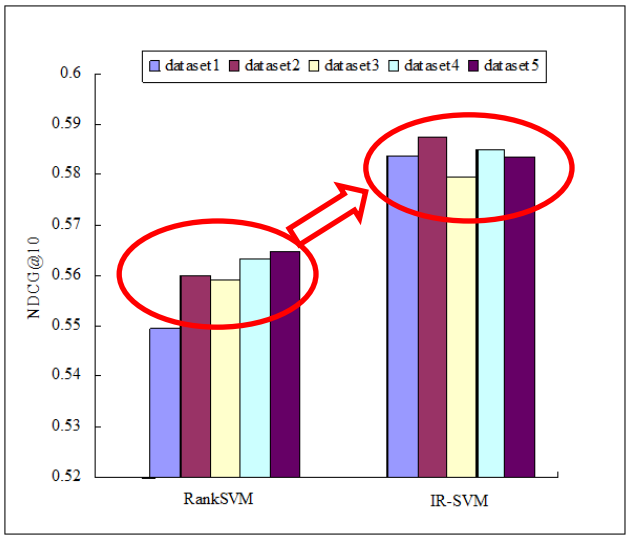
A Possible Solution

- Introduce query-level normalization to the pairwise loss function.
- IRSVM (Y. Cao, J. Xu, et al. SIGIR 2006; T. Qin, T. Liu, et al. IPM 2007)

$$\min \frac{1}{2} \| w \|^2 + C \sum_{i=1}^N \mu^{(i)} \sum_{u,v} \xi_{uv}^{(i)}$$

Query-level normalizer

$$\frac{\max_i \# \{ \text{instance pairs associated with } q_i \}}{\# \{ \text{instance pairs associated with } q_i \}}$$



Significant improvement has been observed by using the query-level normalizer.

Theoretical Explanation

(Y. Lan, T. Liu, et al. ICML 2008)

- Explanation on the empirical improvement of IRSVM over Ranking SVM:

The generalization ability of IRSVM is much better than that of Ranking SVM.

Generalization in Learning to Rank for IR

- Theories on regression and classification have been used to analyze the generalization ability of pointwise or pairwise approaches to learning to rank.
 - Pointwise: classification / regression
 - Pairwise : pairwise classification / U-statistics
- However, are these expected risks really what one cares about in real applications?

Query-level Generalization in Learning to Rank for IR

- Considering that IR evaluations are conducted at the query level, the generalization ability should also be analyzed at the query level.
- Existing work can only give the generalization ability at document level or document pair level, which is not consistent with the evaluation in information retrieval.
- New theory needs to be developed.

Two-Layer Probabilistic Framework for Ranking

- Query – Associate
 - q stands for query, which is viewed as a random variable sampled from query space Q according to an unknown probability distribution P_Q .
 - $(w^{(q)}, g(w^{(q)}))$ stands for the associate of the query and its ground-truth, which is viewed as a random variable sampled from space $\Omega \times \mathcal{G}$ according to an unknown probability distribution D_q .

Associates in Different Approaches

Approach	$w^{(q)}$	$g(w^{(q)})$
Pointwise	Document	Relevance score Class label
Pairwise	Document pair	Partial order
...

Training Data

$$\{(q_1, S_1), \dots, (q_N, S_N)\}$$

$$q_1, \dots, q_N \text{ i.i.d. } \sim P_Q$$

$$S_i = \left(\left(w_1^{(i)}, g(w_1^{(i)}) \right), \dots, \left(w_{n_i}^{(i)}, g(w_{n_i}^{(i)}) \right) \right)$$
$$\left(w_1^{(i)}, g(w_1^{(i)}) \right), \dots, \left(w_{n_i}^{(i)}, g(w_{n_i}^{(i)}) \right) \text{ i.i.d. } \sim D_{q_i}$$

Query-level Loss and Risk

- Expected Query-Level Loss

$$L(f; q) = \int_{\Omega \times \mathcal{G}} l(f; \omega^{(q)}, g(\omega^{(q)})) D_q(d\omega^{(q)}, dg(\omega^{(q)}))$$

- Empirical Query Level Loss

$$\hat{L}(f; q) = \frac{1}{M^{(q)}} \sum_{j=1}^{M^{(q)}} l(f; w_j^{(q)}, g(w_j^{(q)}))$$

- Expected Query-Level Risk

$$R_l(f) = E_{\mathcal{Q}} L(f; q) = \int_{\mathcal{Q}} L(f; q) P_{\mathcal{Q}}(dq)$$

- Empirical Query-Level Risk

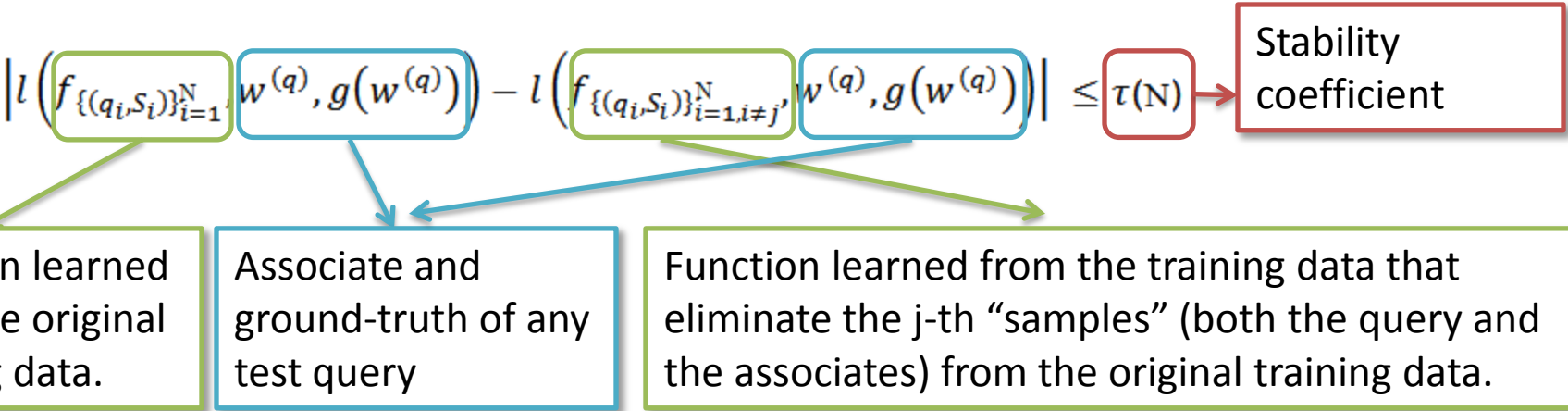
$$\hat{R}_l(f) = \frac{1}{N} \sum_{i=1}^N \hat{L}(f; q_i)$$

Generalization Bound in Learning to Rank

- The goal of Learning to Rank should be to minimize the expected query-level risk $R_l(f)$
- However, as the distribution is unknown, one minimizes the empirical query-level risk $\widehat{R}_l(f)$.
- The generalization in learning to rank for IR is concerned with the bound of the difference between the expected and empirical query-level risks $R_l(f) - \widehat{R}_l(f)$.

Query-level Stability Theory

- Query-Level Stability
 - Query-level stability represents the degree of change in the loss of prediction when randomly removing a query and its associates from the training data.



Query-level Stability Theory

- Generalization Bound based on Query-Level Stability

$$R_l \left(f_{\{(q_i, S_i)\}_{i=1}^N} \right) \leq \widehat{R}_l \left(f_{\{(q_i, S_i)\}_{i=1}^N} \right) + \boxed{2\tau(N) + (4N\tau(N) + B) \sqrt{\frac{\ln \frac{1}{\delta}}{2N}}}$$

- The bound is related to the number of training queries N , and the stability coefficient $\tau(N)$.
- If $\tau(N) \rightarrow 0$ as $N \rightarrow \infty$, then as N tends to infinity, the bound tends to zero, i.e. the generalization ability is good.

Stability of Ranking SVM and IRSVM

- Ranking SVM

$$\tau(N) = \frac{4\kappa^2}{\lambda N} \frac{1 + \frac{\sigma}{\mu \sqrt{\frac{\delta}{N}}}}{1 - \frac{\varepsilon}{\mu}}$$

$O\left(\frac{1}{\sqrt{N}}\right)$

- IRSVM

$$\tau(N) = \frac{4\kappa^2}{\lambda N}$$

$O\left(\frac{1}{N}\right)$

IV

Generalization Bounds of Ranking SVM and IRSVM

- Ranking SVM

$$R_l \left(f_{\{(q_i, S_i)\}_{i=1}^N} \right) \leq \widehat{R}_l \left(f_{\{(q_i, S_i)\}_{i=1}^N} \right) \quad \text{much looser}$$

$$+ \frac{8\kappa^2}{\lambda N} \frac{1 + \frac{\sigma}{\mu \sqrt{\frac{\delta}{N}}}}{1 - \frac{\varepsilon}{\mu}} + \left(\frac{16\kappa^2 \frac{1 + \frac{\sigma}{\mu \sqrt{\frac{\delta}{N}}}}{1 - \frac{\varepsilon}{\mu}} + \lambda(1 + 2C\kappa)}{\lambda} \right) \sqrt{\frac{\ln \frac{1}{\delta}}{2N}}$$

- IRSVM

IV

$$R_l \left(f_{\{(q_i, S_i)\}_{i=1}^N} \right) \leq \widehat{R}_{l_h} \left(f_{\{(q_i, S_i)\}_{i=1}^N} \right)$$

$$+ \frac{8\kappa^2}{\lambda N} + \frac{16\kappa^2 + \lambda(1 + 2C\kappa)}{\lambda} \sqrt{\frac{\ln \frac{1}{\delta}}{2N}} \quad \text{much tighter}$$

Discussions

- When N tends to infinity, the upper bound of IRSVM will tends to zero and the convergent rate is $o\left(\frac{1}{\sqrt{N}}\right)$
- For Ranking SVM, as N tends to infinity, the upper bound may be a constant: $O(1)$.
- In other words, for Ranking SVM, with infinite number of training queries, there is still a gap between the query-level empirical and query-level expected risks.

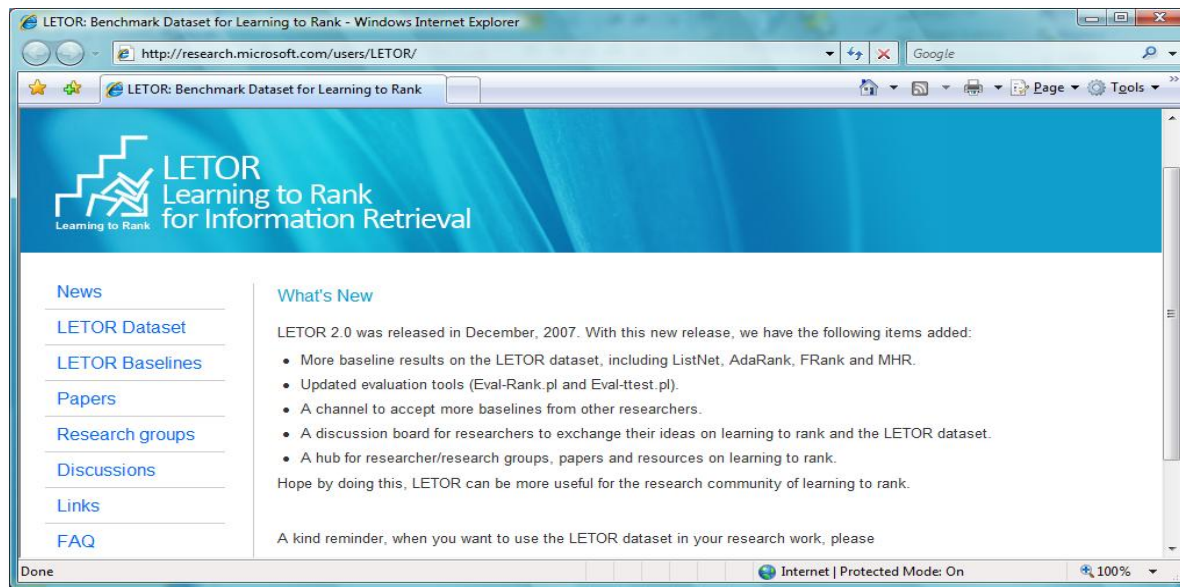
Summary

- We have introduced pointwise and pairwise approaches, and have mentioned the “patches” to improve these approaches.
- A question arises:

Are there more fundamental approaches to the problem of learning to rank?

Contextual Ads ☺

- LETOR: Benchmark Dataset for Learning to Rank
 - <http://research.microsoft.com/users/LETOR/>
 - 500+ downloads till now.
 - Maintained by Tie-Yan Liu, Jun Xu, Tao Qin, et al.
 - A major release by the end of this year.



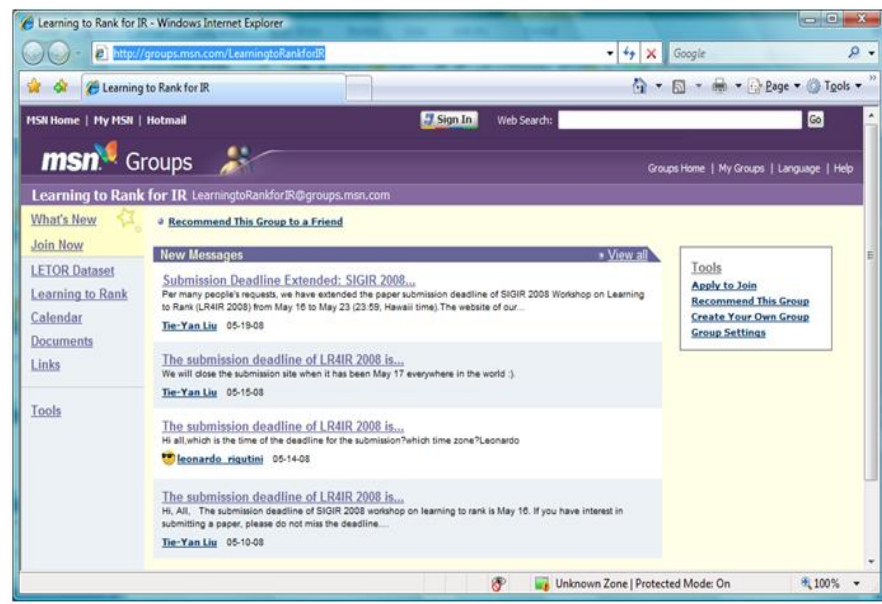
Contextual Ads 😊

- SIGIR 2008 workshop on learning to rank for IR
 - <http://research.microsoft.com/users/LR4IR-2008/>
 - Organized by Hang Li, Tie-Yan Liu, Chengxiang Zhai.
- Agenda
 - Keynotes (given by Steven Robertson and Hongyuan Zha)
 - Technical paper presentation (8 papers)
 - Opinion & commentary session



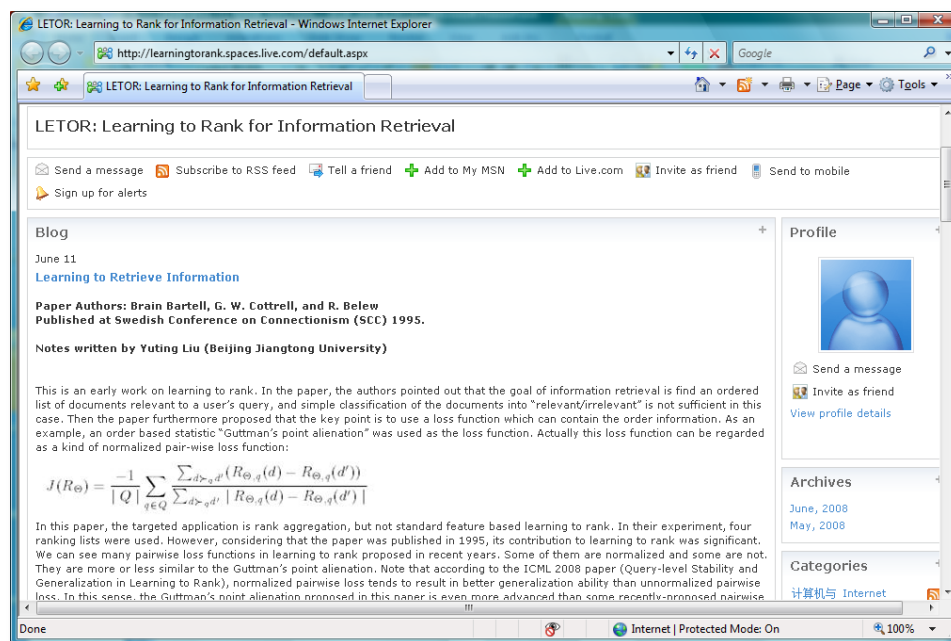
Contextual Ads 😊

- Learning to Rank Discussion Group
 - <http://groups.msn.com/LearningtoRankforIR>
 - News about learning to rank activities
 - Technical discussions on learning to rank
 - Maintained by
 - Tie-Yan Liu
 - Jun Xu



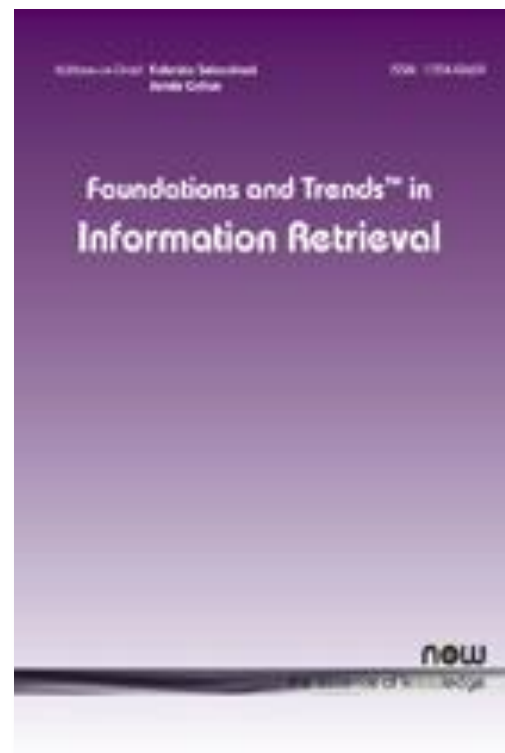
Contextual Ads ☺

- Learning to Rank Technical Blog
 - <http://learningtorank.spaces.live.com/>
 - Paper reading notes
 - Individual opinions
 - Invited articles
 - Maintained by
 - Tie-Yan Liu
 - Tao Qin



Contextual Ads 😊

- “Learning to Rank for Information Retrieval” at Foundations and Trends in Information Retrieval (FnTIR), 2009.
 - A paper version of this tutorial
 - Being longer (~100 pages) and containing more information.
- Authored by:
 - Tie-Yan Liu



- Next -

Listwise Approach to Learning to Rank

Coffee Break



Learning to Rank Algorithms

Overview

Conventional ML Approach

Ordinal Regression: A Pointwise Approach

Preference Learning: A Pairwise Approach

Listwise Ranking: A Listwise Approach

Why Listwise Approach?

- By treating the list of documents associated with the same query as a learning instance, one can naturally obtain
 - The rank (position) information,
 - The query-level information.
- And thus can embed more unique properties of ranking for IR in the learning process.

Listwise Ranking: A Listwise Approach

- Input space
 - Document collection w.r.t. a query

$$(X_1^{(q)}, \dots, X_{M^{(q)}}^{(q)}) \in (R^T)^{M^{(q)}}$$

- Output space
 - Permutation of these documents: $Y \in \Pi_{M^{(q)}}$

Listwise Ranking: Two Major Branches

- Direct optimization of IR evaluation measures
 - Try to optimize rank-based IR evaluation measures, or at least something correlated to the measures.
- Listwise loss minimization
 - Minimize a loss function defined on permutations (ranked document lists).
 - The loss function is designed by considering unique properties of ranking for IR.

Direct Optimization of IR Measures

- It is natural to directly optimize what is used to evaluate the ranking results.
- However, it is non-trivial.
 - Evaluation measures such as NDCG are non-smooth and non-differentiable since they depend on the ranks.
(S. Robertson and H. Zaragoza, Information Retrieval 2007; J. Xu, T. Liu, et al. SIGIR 2008).
 - It is challenging to optimize such objective functions, since most optimization techniques in the literature were developed to handle smooth and differentiable cases.

Tackle the Challenges

- Smoothen the objective
 - Optimize a smooth and differentiable upper bound of the evaluation measure ([SVM-MAP](#))
 - Soften (approximate) the evaluation measure so as to make it smooth and differentiable ([SoftRank](#))
- Optimize the non-smooth objective directly
 - Use IR measure to update the distribution in Boosting ([AdaRank](#))
 - Use genetic programming ([RankGP](#))
 - Directly define gradient ([LambdaRank](#))

SVM-MAP

(Y. Yue, T. Finley, et al. SIGIR 2007)

- Use the framework of SVM-Struct for optimization
 - I. Tsochantaridis, et al. ICML 2004; T. Joachims, ICML 2005.
- Incorporate Average Precision in the constraints:
 - The score of a true ranking should be higher than those of incorrect rankings.
- *Sum of slacks $\sum \xi$ upper bounds $\sum (1-AP(y'))$.*

$$\min \frac{1}{2} \|w\|^2 + \frac{C}{N} \sum_{i=1}^N \xi^{(i)}$$
$$\forall y' \neq y: w^T \Psi(y^{(i)}, x^{(i)}) \geq w^T \Psi(y', x^{(i)}) + \Delta(y^{(i)}, y') - \xi^{(i)}$$

$$\Psi(y', x^{(i)}) = \sum_{u:rel} \sum_{v:rel} y'_{uv} \cdot (x_u^{(i)} - x_v^{(i)})$$

$$\Delta(y^{(i)}, y') = 1 - AP(y')$$

Challenges in SVM-MAP

- For Average Precision, the **true ranking** is a ranking where the relevant documents are all ranked in the front, e.g.,

$y =$ 

- An **incorrect ranking** would be any other ranking, e.g.,

$y' =$ 

- Exponential number of rankings, thus an exponential number of constraints!

Structural SVM Training

- **STEP 1:** Perform optimization on only current **working set of constraints**.
- **STEP 2:** Use the model learned in STEP 1 to find the most violated constraint from the exponential set of constraints.
- **STEP 3:** If the constraint returned in STEP 2 is more violated than the most violated constraint in the working set, add it to the working set.

STEP 1-3 is guaranteed to loop for at most a polynomial number of iterations. [I. Tsochantaridis et al. 2005]

Finding the Most Violated Constraint

- Most violated y :

$$\arg \max_{y' \in Y} \left(\Delta(y^{(i)}, y') + \sum_{u:rel} \sum_{v:!rel} y'_{uv} \cdot (w^T x_u^{(i)} - w^T x_v^{(i)}) \right)$$

- If the relevance at each position is fixed, $\Delta(y^{(i)}, y')$ will be the same. But if the documents are sorted by their scores in descending order, the second term will be maximized.
- Strategy
 - Sort the relevant and irrelevant documents and form a perfect ranking.
 - Start with the perfect ranking and swap two adjacent relevant and irrelevant documents, so as to find the optimal interleaving of the two sorted lists.
 - For real cases, the complexity is $O(M \log M)$.

SoftRank

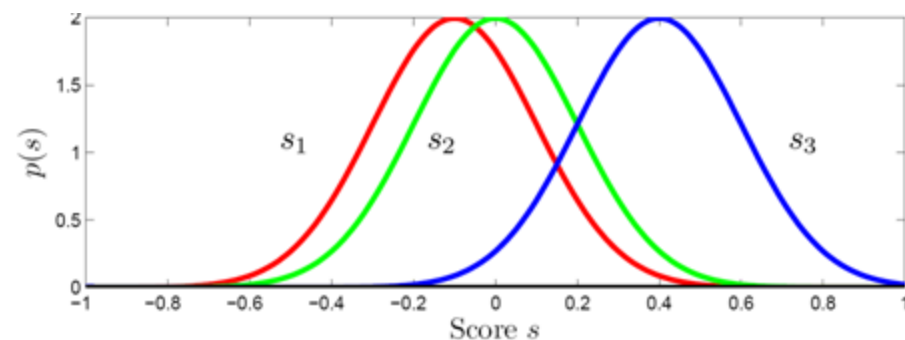
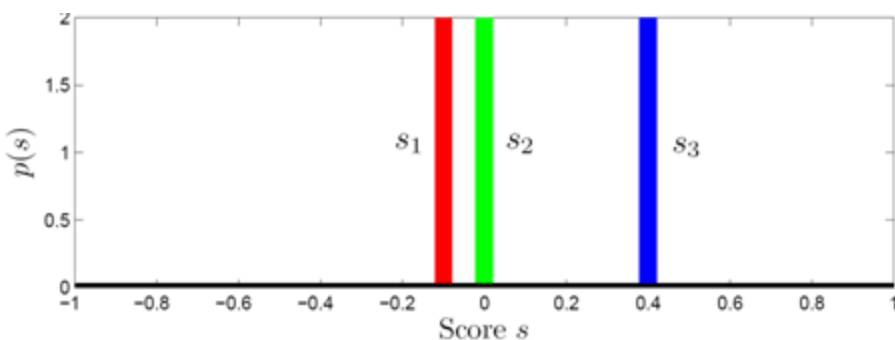
(M. Taylor, J. Guiver, et al. LR4IR 2007/WSDM 2008)

- Key Idea:
 - Avoid sorting by treating scores as random variables
- Steps
 - Construct score distribution
 - Map score distribution to rank distribution
 - Compute expected NDCG (SoftNDCG) which is smooth and differentiable.
 - Optimize SoftNDCG by gradient descent.

Construct Score Distribution

- Considering the score for each document s_j as random variable, by using Gaussian.

$$p(s_j) = N(s_j | f(x_j; w), \sigma_s^2)$$



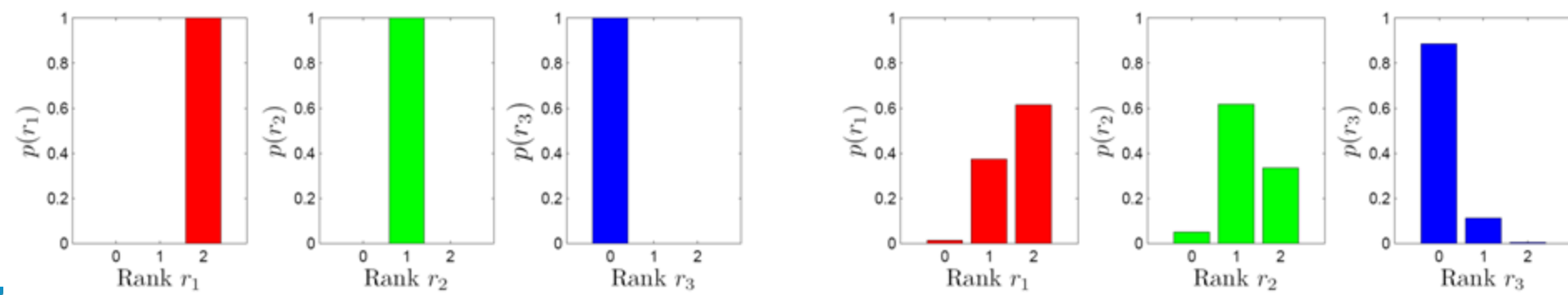
From Scores to Rank Distribution

- Probability of d_u being ranked before d_v

$$\pi_{uv} = \int_0^\infty N(s \mid f(x_u; w) - f(x_v; w), 2\sigma_s^2) ds$$

- Expected rank $E[r_j] = \sum_{u \neq j} \pi_{uj}$
- Rank Distribution

$$p_j^{(u)}(r) = p_j^{(u-1)}(r-1)\pi_{uj} + p_j^{(u-1)}(r)(1-\pi_{uj})$$



Define the probability of a document being ranked at a particular position

Soft NDCG

- From hard to soft

$$G_R = G_{R,max}^{-1} \sum_{r=0}^{R-1} g(r) D(r) \Rightarrow \mathcal{G} = G_{max}^{-1} \sum_{j=1}^N g(j) \sum_{r=0}^{N-1} D(r) p_j(r)$$

- Since $p_j(r)$ depends on π_{uj} , the optimization can be done if we know $\frac{\partial \pi_{uj}}{\partial \bar{s}_m}$.

$$\frac{\partial \pi_{uj}}{\partial \bar{s}_m} = \begin{cases} \mathcal{N}(0 | \bar{s}_m - \bar{s}_j, 2\sigma_s^2) & m = u, m \neq j \\ -\mathcal{N}(0 | \bar{s}_u - \bar{s}_m, 2\sigma_s^2) & m \neq u, m = j \\ 0 & m \neq u, m \neq j \end{cases}$$

AdaRank

(J. Xu, H. Li. SIGIR 2007)

- The optimization process
- Use the exponential loss function to update the distributions D of learning instances and to determine the parameter α

Given: $(x_1, y_1), \dots, (x_m, y_m); \quad x_i \in \mathcal{X}$
Initialize $D_1(i) = 1/m$.
For $t = 1, \dots, T$:

- Train weak learner using distribution D_t .
- Get weak hypothesis $h_t : \mathcal{X} \rightarrow \mathbb{R}$.
- Choose $\alpha_t \in \mathbb{R}$.

$$\alpha = \frac{1}{2} \ln \left(\frac{1+r}{1-r} \right) \quad r = \sum_i D(i) u_i$$

- Update:

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution)

Output the final hypothesis:

$$H(x) = \text{sign} \left(\sum_{t=1}^T \alpha_t h_t(x) \right).$$

AdaRank

Weak learner that can rank important queries more correctly will have larger weight α

Emphasize those hard queries by setting their distributions.

- Given: initial distribution D_1 for each query
- For $t=1, \dots, T$:
 - Train weak learner using distribution D_t
 - Get weak ranker $h_t: X \rightarrow R$
 - Choose $\alpha = \frac{1}{2} \ln \frac{\sum_{i=1}^N D_t(i) \{1 + E(q_i, h_t, y^{(i)})\}}{\sum_{i=1}^N D_t(i) \{1 - E(q_i, h_t, y^{(i)})\}}$
 - Create $f_t(x) = \sum_{s=1}^t \alpha_s h_s(x)$
 - Update: $D_{t+1}(i) = \frac{\exp\{-E(q_i, f_t, y^{(i)})\}}{\sum_{j=1}^N \exp\{-E(q_j, f_t, y^{(j)})\}}$
- Output the final ranker: $f(x) = \sum_t \alpha_t h_t(x)$

Embed IR evaluation measure in the distribution update of Boosting

Theoretical Analysis

- Training error will be continuously reduced during learning phase as long as $e^{-\delta^t_{min}} \sqrt{1 - \dots}$ Assumes the linearity of the IR evaluation measure: only if the measure is linear function, δ can be very small ($\rightarrow 0$).

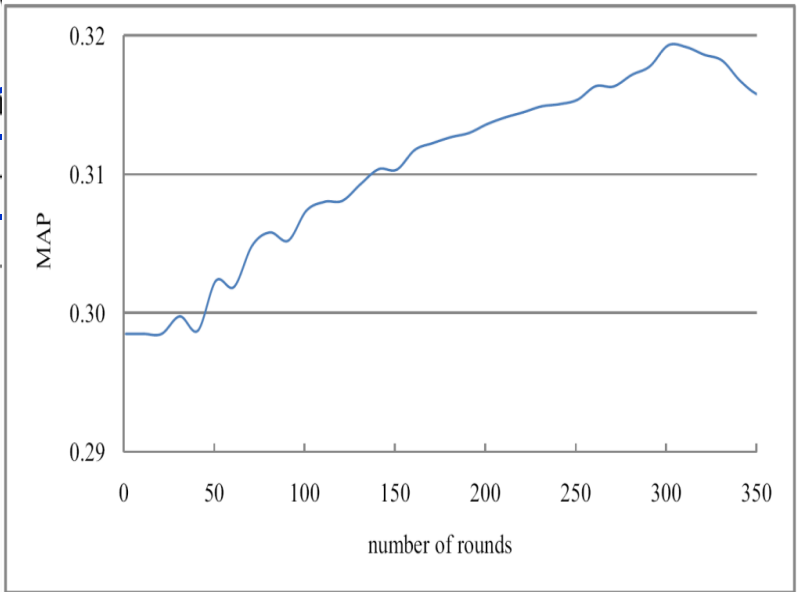
THEOREM 1. The following bound holds for the accuracy of the IR-Boost algorithm on training data.

$$\frac{1}{N} \sum_{i=1}^N E(q_i, f_T, \mathbf{y}^{(i)}) \geq 1 - \prod_{t=1}^T e^{-\delta^t_{min}}$$

where $\varphi(t) = \sum_{i=1}^N P_t(i) E(q_i, h_t, \mathbf{y}^{(i)})$, $\delta^t_{min} = \min_i \delta_i^t$

$$\delta_i^t = E(q_i, f_t, \mathbf{y}^{(i)}) - E(q_i, f_{t-1}, \mathbf{y}^{(i)})$$

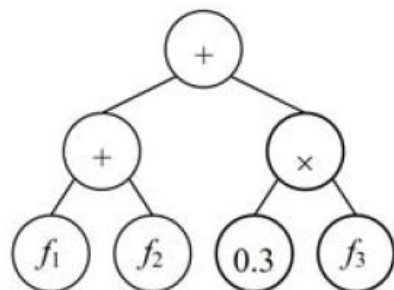
for all $i = 1, 2, \dots, N$ and $t = 1, 2, \dots, T$.



RankGP

(J. Yeh, J. Lin, et al. LR4IR 2007)

- Define ranking function as a tree



$$I = (S_v, S_c, S_{op})$$

$$S_v = \{f_i \mid f_i \in F\},$$

$$S_c = \{0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\},$$

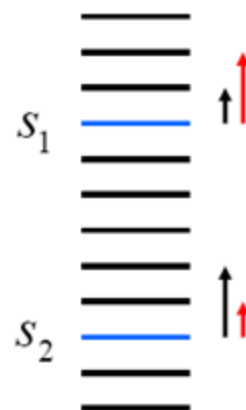
$$S_{op} = \{+, -, \times, /\}.$$

- Learning
 - Single-population GP, which has been widely used to optimize non-smoothing non-differentiable objectives.
- Evolution mechanism
 - Crossover, mutation, reproduction, tournament selection.
- Fitness function
 - IR evaluation measure (MAP).

LambdaRank

(C. Burges, R. Ragno, et al. NIPS 2006)

- Basic idea
 - IR evaluation measures are non-smooth and non-differentiable.
 - It is usually much easier to specify rules determining how to change rank order of documents than to construct a smooth loss function.
- Example
 - Two relevant documents d_1 and d_2 ;
 - WTA as the evaluation measure.



To achieve WTA=1

$$\text{Want: } \left| \frac{\partial L}{\partial s_1} \right| \gg \left| \frac{\partial L}{\partial s_2} \right|$$

LambdaRank

- Instead of explicitly defining the loss function, directly define the gradient:

$$\frac{\partial L}{\partial s_j} = -\lambda_j(s_1, y_1, \dots, s_n, y_n) \quad S: \text{score}; y: \text{label}.$$

- A lambda which was shown to effectively optimize NDCG.

$$\lambda = M\left(\frac{1}{1 + e^{s_i - s_j}}\right)(2^{c(i)} - 2^{c(j)})\left(\log\left(\frac{1}{1 + r_i}\right) - \log\left(\frac{1}{1 + r_j}\right)\right)$$

Other Work in This Category

- Using coordinate ascent for optimization
 - Direct maximization of rank-based metrics for information retrieval, (D. Metzler, W. Croft. et al. CIIR Technical Report IR-429, 2005).
- Using genetic algorithm in different ways
 - Discovery of context-specific ranking functions for effective information retrieval using genetic programming (W. Fan, D. Gordon, et al. IEEE TKDE, 2004)
 - A generic ranking function discovery framework by genetic programming for information retrieval (W. Fan, D. Gordon, et al. IPM, 2004)
 - Ranking function optimization for effective Web search by Genetic Programming: an empirical study (W. Fan, D. Gordon, et al. HICSS, 2004)
 - Learning to Rank (A. Trotman, Information Retrieval, 2005)
 - A combined component approach for finding collection-adapted ranking functions based on genetic programming (CCA), (H. Almeida, M. Goncalves, et al. SIGIR 2007).

Discussions

- Open problems:
 - Is the upper bound in SVM-MAP tight enough?
 - How does SoftNDCG correlate with NDCG?
 - What is the correlation between the implicit loss function used in LambdaRank and NDCG?
 - Multiple measures do not necessarily agree with each other: which to optimize and how to tradeoff?

Listwise Loss Minimization

- Defining listwise loss functions based on the understanding on the unique properties of ranking for IR.
 - Hopefully correlated with all IR evaluation measures 😊.
- Representative Algorithms
 - RankCosine
 - ListNet
 - ListMLE

RankCosine

(T. Qin, T. Liu, et al. IP&M 2007)

- Loss function define as

$$L(f) = \frac{1}{2} \left(1 - \frac{\sum_{j=1}^M \phi(\psi(y_j)) \phi(f(x_j))}{\sqrt{\sum_{j=1}^M \phi^2(f(x_j))} \sqrt{\sum_{j=1}^M \phi^2(\psi(y_j))}} \right)$$

Transformation Mapping from ground truth to score Model output

- Learning
 - Additive ranking model (like Boosting).
 - Gradient descent for optimization of the loss.

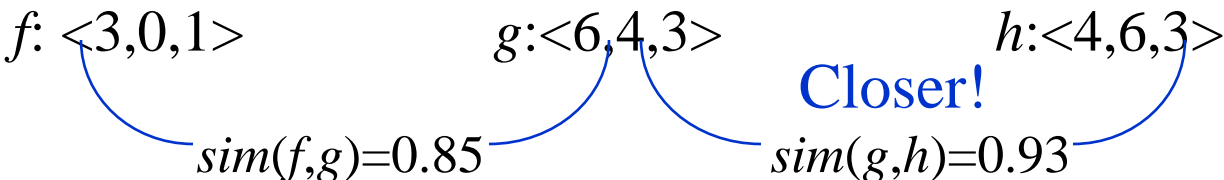
RankCosine

- **Properties of the Cosine loss**
 - Loss is computed by using all documents of a query.
 - Insensitive to number of documents / document pairs per query due to the denominator.
 - Emphasize correct ranking by setting high scores for top in ground truth.
 - Bounded: $0 \leq L(f) \leq 1$

A kind of query-level loss function

More Investigation on Loss Functions

- A Simple Example:
 - function f : $f(A)=3, f(B)=0, f(C)=1$ ACB
 - function h : $h(A)=4, h(B)=6, h(C)=3$ BAC
 - ground truth g : $g(A)=6, g(B)=4, g(C)=3$ ABC
- Question: which function is closer to ground truth?
 - Based on pointwise similarity: $sim(f,g) < sim(g,h)$.
 - Based on pairwise similarity: $sim(f,g) = sim(g,h)$
 - Based on cosine similarity between score vectors?

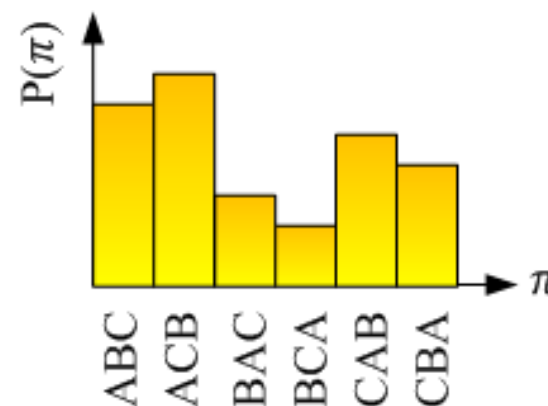
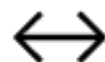


- However, according to NDCG or RC, f should be closer to g !

Permutation Probability Distribution

- Question:
 - How to represent a ranked list?
- Solution
 - Ranked list \leftrightarrow Permutation probability distribution
 - More informative representation for ranked list: permutation and ranked list has 1-1 correspondence.

$f: f(A)=3, f(B)=0, f(C)=1;$
Ranking by f : ABC



Plackett Luce Model: Defining Permutation Probability

- Probability of a permutation π is defined as

$$P_s(\pi) \equiv P(\pi \mid \varphi(s)) = \prod_{j=1}^M \frac{\varphi(s_{\pi(j)})}{\sum_{k=j}^M \varphi(s_{\pi(k)})}, \text{ e.g., } s_{\pi(j)} = f(x_{\pi(j)})$$

- Example:

$$P_f(\text{ABC}) = \frac{\varphi(f(A))}{\varphi(f(A)) + \varphi(f(B)) + \varphi(f(C))} \cdot \frac{\varphi(f(B))}{\varphi(f(B)) + \varphi(f(C))} \cdot \frac{\varphi(f(C))}{\varphi(f(C))}$$

P(A ranked No.1)

P(B ranked No.2 | A ranked No.1)
= P(B ranked No.1)/(1- P(A ranked No.1))

P(C ranked No.3 | A ranked No.1, B ranked No.2)

Properties of Luce Model

- Nice properties
 - Continuous, differentiable, and concave w.r.t. score s .
 - Suppose $f(A) = 3, f(B)=0, f(C)=1$, $P(ACB)$ is largest and $P(BCA)$ is smallest; for the ranking based on f : ACB, swap any two, probability will decrease, e.g., $P(ACB) > P(ABC)$
 - Translation invariant, when $\varphi(s) = \exp(s)$

$$P_s(\pi) = \prod_{j=1}^M \frac{\varphi(s_{\pi(j)})}{\sum_{k=j}^M \varphi(s_{\pi(k)})} = P_{\lambda+s}(\pi) = \prod_{j=1}^M \frac{\varphi(\lambda + s_{\pi(j)})}{\sum_{k=j}^M \varphi(\lambda + s_{\pi(k)})}$$

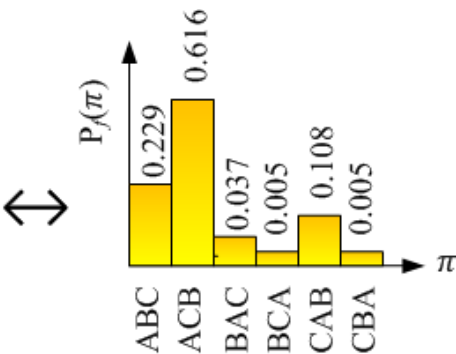
- Scale invariant, when $\varphi(s) = a \cdot s$

$$P_s(\pi) = \prod_{j=1}^M \frac{\varphi(s_{\pi(j)})}{\sum_{k=j}^M \varphi(s_{\pi(k)})} = P_{\lambda s}(\pi) = \prod_{j=1}^M \frac{\varphi(\lambda s_{\pi(j)})}{\sum_{k=j}^M \varphi(\lambda s_{\pi(k)})}$$

Distance between Ranked Lists

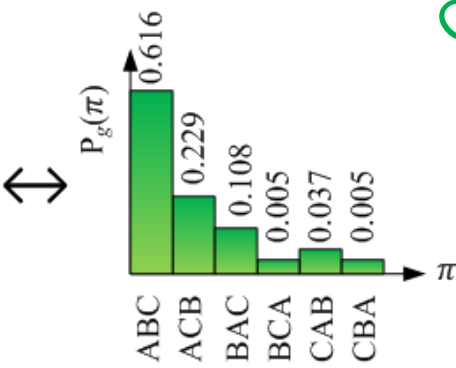
$\varphi = \exp$

f : $f(A) = 3, f(B)=0, f(C)=1$;
Ranking by f : ABC



Using **KL-divergence**
to measure difference
between distributions

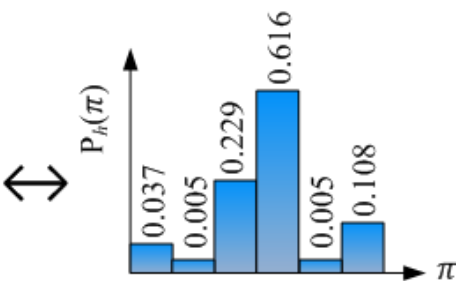
g : $g(A) = 6, g(B)=4, g(C)=3$;
Ranking by g : ABC



Closer!

$dis(f,g) = 0.46$

h : $h(A) = 4, h(B)=6, h(C)=3$;
Ranking by h : ACB



$dis(g,h) = 2.56$

ListNet

(Z. Cao, T. Qin, T. Liu, et al. ICML 2007)

- Loss function = KL-divergence between two permutation probability distributions ($\varphi = \exp$)

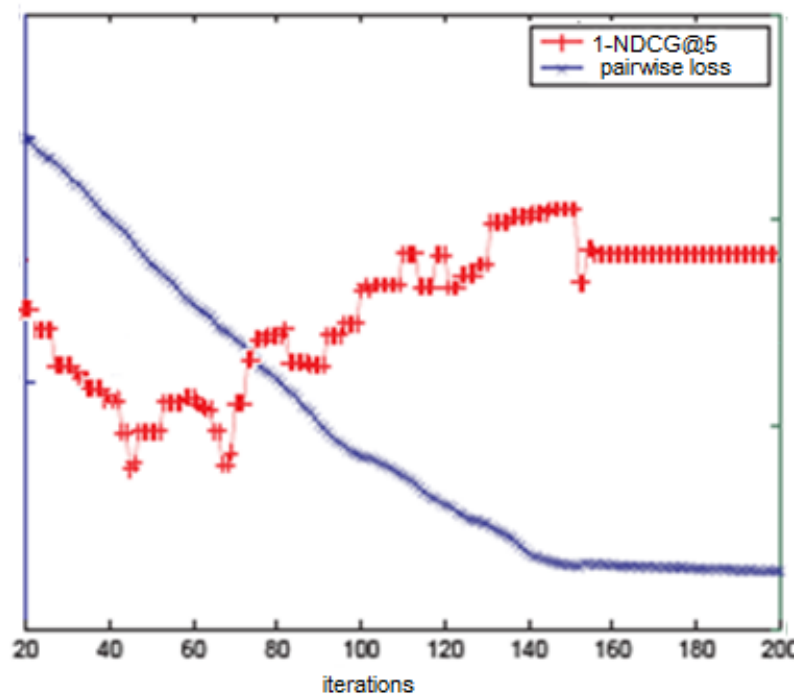
$$L(f) \propto D(\underbrace{P(\pi | \varphi(\psi(y)))}_{\text{Probability distribution defined by the ground truth}} \parallel \underbrace{P(\pi | \varphi(f(x)))}_{\text{Probability distribution defined by the model output}})$$

Probability distribution defined
by the ground truth

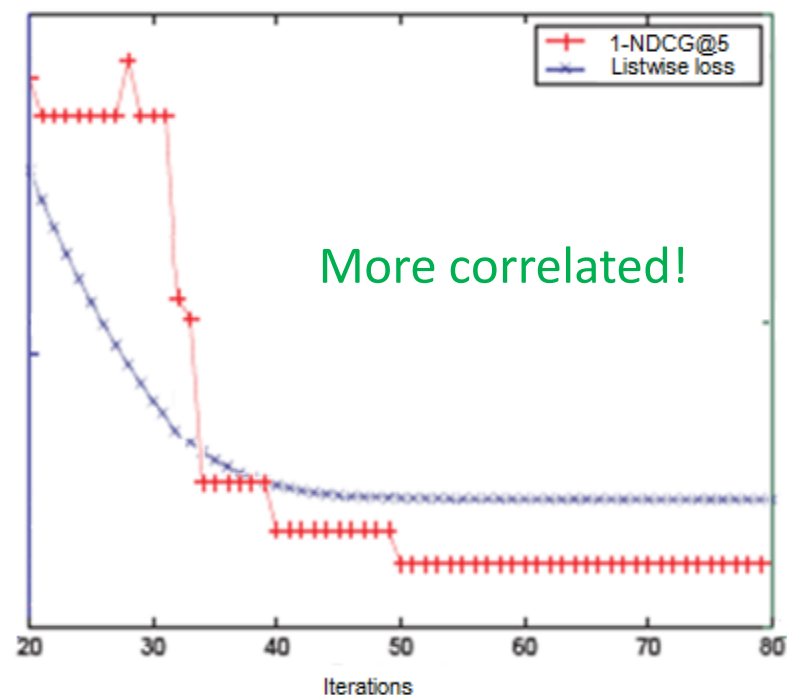
Probability distribution defined
by the model output

- Model = Neural Network
- Algorithm = Gradient Descent

Experimental Results



Pairwise (RankNet)



Listwise (ListNet)

Training Performance on TD2003 Dataset

Limitations of ListNet

- Too complex to use in practice: $O(M \cdot M!)$
- Performance of ListNet depends on the mapping function from ground truth label to scores
 - Ground truths are permutations (ranked list) or group of permutations; but not naturally scores.

Solution

- Never assume ground truth to be scores.
- Given the scores outputted by the ranking model, compute the likelihood of a ground truth permutation, and maximize it to learn the model parameter.
- The complexity is reduced from $O(M \cdot M!)$ to $O(M)$.

ListMLE Algorithm

(F. Xia, T. Liu, et al. ICML 2008)

- Permutation Likelihood

$$P(\pi \mid \varphi(f(x))) = \prod_{j=1}^M \frac{\varphi(f(x_{\pi(j)}))}{\sum_{u=j}^M \varphi(f(x_{\pi(u)}))}$$

- Likelihood Loss

$$L(f) = -\sum_{i=1}^N \log P(y^{(i)} \mid \varphi(f(x^{(i)})))$$

- Model = Neural Network
- Algorithm = Stochastic Gradient Descent

Discussions

- In addition to Luce model, likelihood defined by Mallows model has also been used in listwise ranking.
 - Cranking: Combining rankings using conditional probability models on permutations (G. Lenanon and J. Lafferty, ICML 2002)
- However, we will not further discuss this work due to the following reasons.
 - The Mallow model is more suitable for combining multiple input rankings (rank aggregation), but not for “learning to rank” as mentioned in this tutorial.
 - The testing complexity might be very high since Mallows model does not have the same properties as Luce model.

Theoretical Analysis on Listwise Loss Minimization

(F. Xia, T. Liu, et al. ICML 2008)

- The listwise approach captures the ranking problem in a conceptually more natural way.
- However, the listwise approach lacks of theoretical analysis.
 - Existing work focuses more on algorithm and experiments, than theoretical analysis.
 - While many existing theoretical results on regression and classification can be applied to the pointwise and pairwise approaches, the theoretical study on the listwise approach is not sufficient.

Listwise Ranking

- Input space: X
 - Elements in X are sets of objects to be ranked
- Output space: Y
 - Elements in Y are permutations of objects
- Joint probability distribution: P_{XY}
- Hypothesis space: H
 - $\mathbf{h} \in H : X \rightarrow Y$
- Expected loss

$$R(\mathbf{h}) = \int_{X \times Y} l(\mathbf{h}(\mathbf{x}), \mathbf{y}) dP(\mathbf{x}, \mathbf{y})$$

Empirical loss

$$R_S(\mathbf{h}) = \frac{1}{m} \sum_{i=1}^m l(\mathbf{h}(\mathbf{x}^{(i)}), \mathbf{y}^{(i)}).$$

True Loss in Listwise Ranking

- To analysis theoretical properties of listwise loss functions, the “true” loss of ranking is to be defined.
 - The true loss describes the difference between a given ranked list (permutation) and the ground truth ranked list (permutation).
- Ideally, the “true” loss should be cost-sensitive, but for simplicity, we investigate the “0-1” loss.
 - $$l(\mathbf{h}(\mathbf{x}), \mathbf{y}) = \begin{cases} 1, & \text{if } \mathbf{h}(\mathbf{x}) \neq \mathbf{y} \\ 0, & \text{if } \mathbf{h}(\mathbf{x}) = \mathbf{y}, \end{cases} \quad \mathbf{h} \in H : X \rightarrow Y$$

Surrogate Loss in Listwise Ranking

- Widely-used ranking function
 - $h(x^{(i)}) = \text{sort}(f(x_1^{(i)}), \dots, f(x_{M^{(i)}}^{(i)}))$
- Corresponding empirical loss
 - $R_S(f) = \frac{1}{N} \sum_{i=1}^N l(\text{sort}(f(x_1^{(i)}), \dots, f(x_{M^{(i)}}^{(i)})), y^{(i)})$
- Challenges
 - Due to the sorting function and the 0-1 loss, the empirical loss is non-differentiable.
 - To tackle the problem, a surrogate loss is used.
 - $R_S^\phi(f) = \frac{1}{N} \sum_{i=1}^N \phi(f(x^{(i)}), y^{(i)})$

Surrogate Loss Minimization Framework

- RankCosine, ListNet and ListMLE can all be well fitted into this framework of surrogate loss minimization.

- RankCosine

$$\phi(f(x), y) = \frac{1}{2} \left(1 - \frac{\sum_{j=1}^M \phi(\psi(y_j)) \phi(f(x_j))}{\sqrt{\sum_{j=1}^M \phi^2(f(x_j))} \sqrt{\sum_{j=1}^M \phi^2(\psi(y_j))}} \right)$$

- ListNet

$$\phi(f(x), y) = D(P(\pi | \phi(\psi(y))) \| P(\pi | \phi(f(x))))$$

- ListMLE

$$\phi(f(x), y) = -\log P(y | \phi(f(x)))$$

Evaluation of Surrogate Losses

- Continuity, differentiability, and convexity.
- Computational efficiency
- Statistical consistency
- Soundness
- Generalization ability

These properties have been well studied in classification, but not sufficiently in ranking.

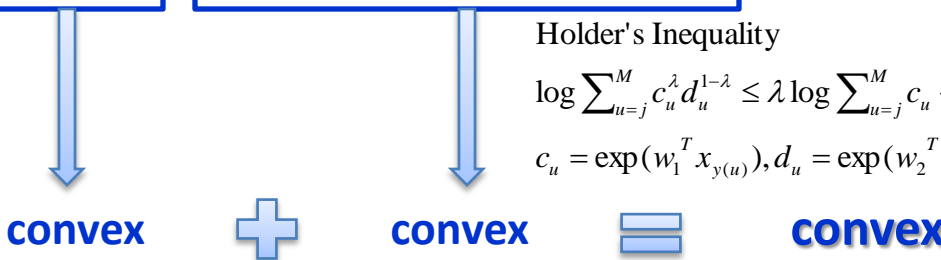
Continuity, Differentiability, Convexity, and Efficiency

Loss	Continuity	Differentiability	Convexity	Efficiency
Cosine Loss (RankCosine)	✓	✓	X	$O(M)$
Cross-entropy loss (ListNet)	✓	✓	✓	$O(M \cdot M!)$
Likelihood loss (ListMLE)	✓	✓	✓	$O(M)$

Proof for Convexity

- For likelihood loss

$$\begin{aligned} \phi(f(x), y) &= -\log P(y | \phi(f(x))) \\ &= \boxed{-w^T x_{y(j)}} - \boxed{\log \sum_{u=j}^M \exp(w^T x_{y(u)})} \end{aligned}$$




convex + convex = convex

Holder's Inequality
 $\log \sum_{u=j}^M c_u^\lambda d_u^{1-\lambda} \leq \lambda \log \sum_{u=j}^M c_u + (1-\lambda) \log \sum_{u=j}^M d_u$
 $c_u = \exp(w_1^T x_{y(u)}), d_u = \exp(w_2^T x_{y(u)})$

- For cross entropy loss

$$\phi(f(x), y) = \sum_{\pi} P_{\psi} \left(\boxed{-\log P(\pi | \phi(f(x)))} \right)$$



Σ convex = convex

Statistical Consistency

- When minimizing the expected surrogate loss $R^\phi(f)$ is equivalent to minimizing the expected 0-1 loss $R(h)$ (which solution is Bayes ranker y^*), we say the surrogate loss function is consistent.

Theorem. Let ϕ be an **order sensitive** loss function on $\Omega \subset R^n$. $\forall n$ objects, if its permutation probability space is **order preserving** with respect to $n - 1$ objective pairs $(j_1, j_2), (j_2, j_3), \dots, (j_{n-1}, j_n)$. Then the loss ϕ is consistent.

The perfect ranking of an object is inherently determined by its own.

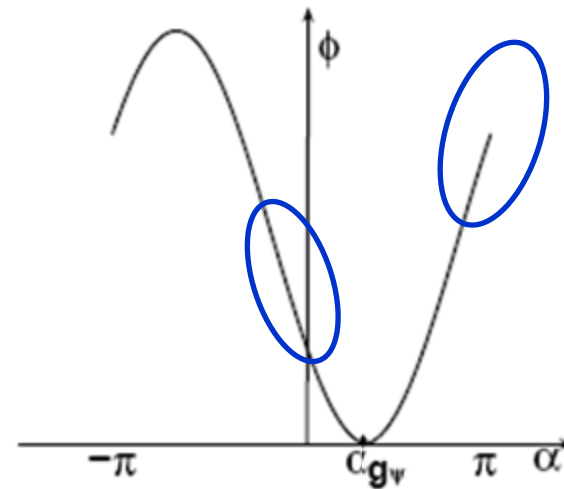
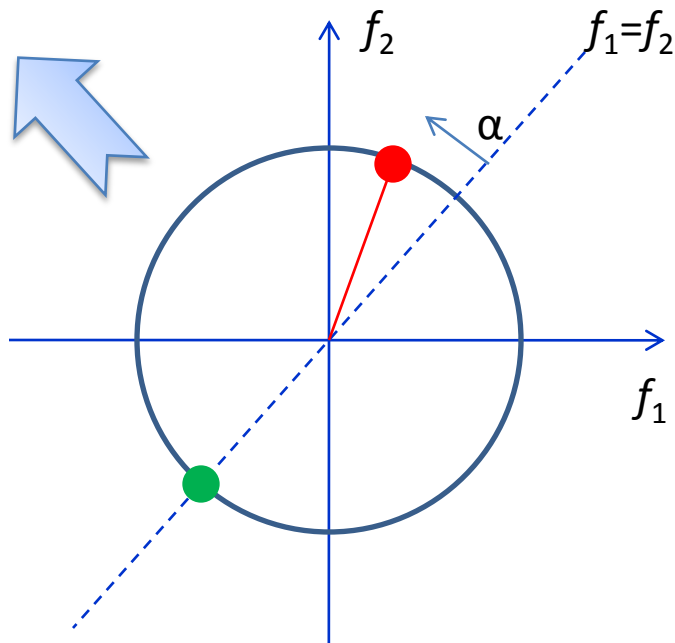
Minimum ϕ is achieved when sorting $f(x)$ results in the same permutation with a given y .

Statistical Consistency (3)

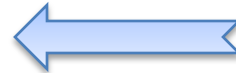
- It can be proven:
 - Cosine Loss is statistically consistent.
 - Cross entropy loss is statistically consistent.
 - Likelihood loss is statistically consistent.

Soundness

- Cosine loss is not very sound
 - Suppose we have two documents $x_2 \succ x_1$.



Incorrect Ranking

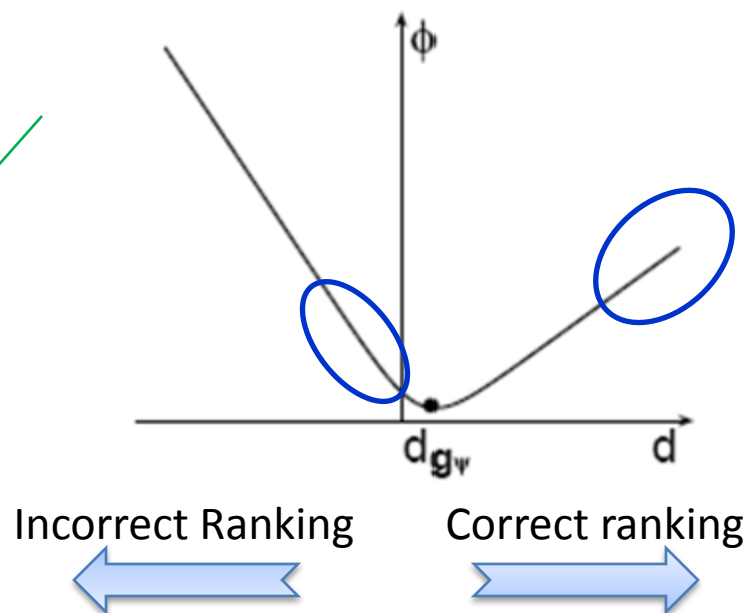
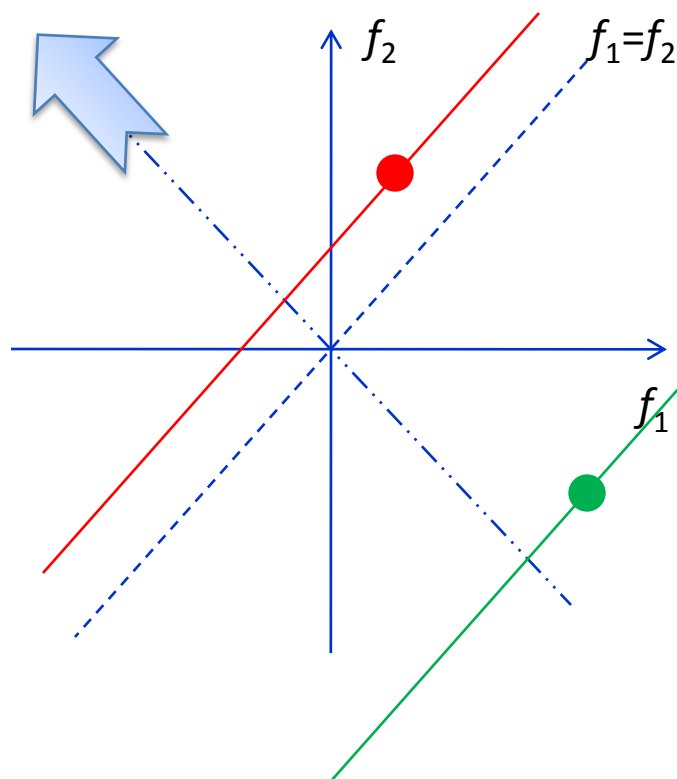


Correct ranking



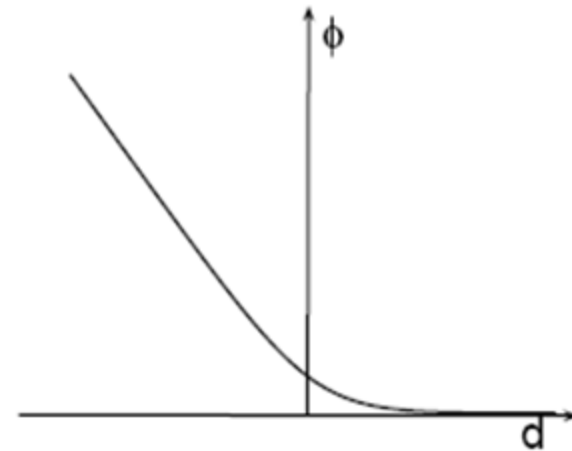
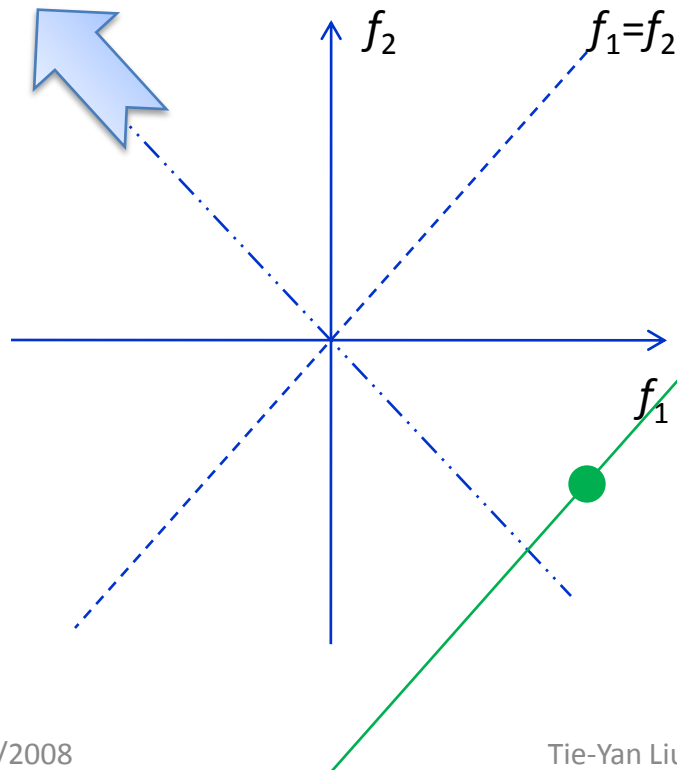
Soundness (2)

- Cross entropy loss is not very sound
 - Suppose we have two documents $x_2 \succ x_1$.



Soundness (3)

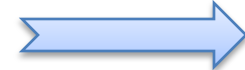
- Likelihood loss is sounder
 - Suppose we have two documents $x_2 \succ x_1$.



Incorrect Ranking



Correct ranking



Generalization Ability

- Rademacher Average based generalization theory.

Theorem 1. Let \mathcal{A} denote a listwise ranking algorithm, and let $\phi_{\mathcal{A}}(f; z, y)$ be its surrogate listwise loss, given the training data $S = \{(z_i, y_i), i = 1, \dots, N\}$, if $\forall f \in \mathcal{F}, (z, y) \in \mathcal{Z} \times \mathcal{Y}, \phi_{\mathcal{A}}(f; z, y) \in [0, 1]$, then with probability at least $1 - \delta$, the following inequality holds:

$$\sup_{f \in \mathcal{F}} (R_{\phi_{\mathcal{A}}}(f) - \hat{R}_{\phi_{\mathcal{A}}}(f; S)) \leq 2\hat{\mathcal{R}}(\phi_{\mathcal{A}} \circ \mathcal{F}) + \sqrt{\frac{2 \ln \frac{2}{\delta}}{N}},$$

where $\hat{\mathcal{R}}(\phi_{\mathcal{A}} \circ \mathcal{F}) = E_{\sigma} \sup_{f \in \mathcal{F}} \frac{1}{N} \sum_{i=1}^N \sigma_i \phi_{\mathcal{A}}(f; z_i, y_i)$.

Generalization ability

Rademacher Average of $\phi_{\mathcal{A}} \circ \mathcal{F}$

Rademacher Averages of Listwise Ranking Algorithms

Theorem 2. The upper bounds of $\widehat{\mathcal{R}}(\phi_{\mathcal{A}} \circ \mathcal{F})$ of ListMLE, ListNet and RankCosine can all be represented in the following form:

$$\widehat{\mathcal{R}}(\phi_{\mathcal{A}} \circ \mathcal{F}) \leq C_{\mathcal{A}}(\varphi) N(\varphi) \widehat{\mathcal{R}}(\mathcal{F}),$$

where \mathcal{A} stands for a listwise ranking algorithm, $N(\varphi) = \sup_{x \in [-BA, BA]} \varphi'(x)$, and $C_{\mathcal{A}}(\varphi)$ is defined as follows:

φ	$N(\varphi)$	$C_{ListMLE}(\varphi)$	$C_{ListNet}(\varphi)$	$C_{RankCosine}(\varphi)$
φ_L	a	$\frac{2}{(b-aBA)(\log M + \log \frac{b+aBA}{b-aBA})}$	$\frac{2M!}{(b-aBA)(\log M + \log \frac{b+aBA}{b-aBA})}$	$\frac{\sqrt{M}}{2(b-aBA)}$
φ_E	ae^{aBA}	$\frac{2e^{aBA}}{\log M + 2aBA}$	$\frac{2M!e^{aBA}}{\log M + 2aBA}$	$\frac{\sqrt{M}e^{aBA}}{2}$
φ_S	$\frac{ae^{aBA}}{(1+e^{-aBA})^2}$	$\frac{2(1+e^{aBA})}{\log M + aBA}$	$\frac{2M!(1+e^{aBA})}{\log M + aBA}$	$\frac{\sqrt{M}(1+e^{aBA})}{2}$

$$\forall x \in \mathcal{X}, \|x\| \leq A \qquad \forall x \in \mathcal{X}, |f(x)| \leq BA$$

Findings on Generalization Ability

- The convergence rate of the three algorithms are all $O\left(\frac{1}{\sqrt{N}}\right)$.
- The bound for ListMLE decreases monotonously, while the bounds for ListNet and RankCosine increase monotonously, w.r.t. the list length M .
- *The bound for ListMLE is always tighter than that for ListNet. When the list length M is larger than or equal to 6, the generalization bound for ListMLE is tighter than that for RankCosine.*
- The exponential transformation function is not the best choice in terms of generalization bound. Linear transformation function is even better in most cases.

Discussions

- Based on the above theoretical analyses, it seems that Luce model based likelihood loss (corresponding to the ListMLE algorithm) is one of the best listwise loss functions.
 - The effectiveness of ListMLE has also been validated by empirical study, see (F. Xia, T. Liu, et al. ICML 2008).
- Unsolved problems
 - In real ranking problems, the true loss should be cost-sensitive but not 0-1.
 - Approximation error is also important in generalization analysis.

Advanced Topics

Ranking Functions

Ranking Functions

- As can be seen, in most previous work, attention was paid to the loss function and the learning process.
- The ranking function is assumed to be $f(x)$ (i.e. independent relevance), and mostly a linear combination of pre-defined features, for simplicity.
- Is it sufficient?
 - Beyond independent relevance.
 - Beyond single ranking function.
 - Beyond linear combination of features.

Beyond Independent Relevance

- Ranking is a task much more complicated than independent relevance.
- Examples
 - Pseudo relevance feedback (PRF)
 - Topic distillation (TD)
 - Diversify search results (DS)
 - Ranking with both content and hyperlinks.
 - ...

Relational Ranking

(T. Qin, T. Liu, et al. WWW 2008)

Content Relation

Ranking function $f(x, R)$

Learning

Ranking function Ground truth

$$\hat{f} = \arg \min \sum_{i=1}^N \underbrace{L\left(f(x^{(i)}, R^{(i)}), y^{(i)}\right)}_{\text{Objective function}}$$

Relational Ranking Function for PRF

$$f(x, R) = \arg \min_z \left\{ \overset{\text{Objective 1}}{l_1(h(x, w), z)} + \beta \overset{\text{Objective 2}}{l_2(R, z)} \right\}$$

$$l_1(h(x, w), z) = \|h(x, w) - z\|^2 \quad l_2(R, z) = \frac{1}{2} \sum_u \sum_v R_{u,v} (z_u - z_v)^2$$

$$f(x, R) = (I + \beta(D - R))^{-1} h(x, w)$$

Pseudo Relevance Feedback

Relational Ranking Function for TD

Objective 1

Objective 2

$$f(x, R) = \arg \min_z \{l_1(h(x, w), z) + \beta l_2(R, z)\}$$



$$l_1(h(x, w), z) = \|h(x, w) - z\|^2 \quad l_2(R, z) = \sum_u \sum_v R_{u,v} \exp(z_u - z_v)$$



$$f(x, R) = (2I + \beta(2D - R - R^T))^{-1} (2h(x, w) - \beta g_1)$$

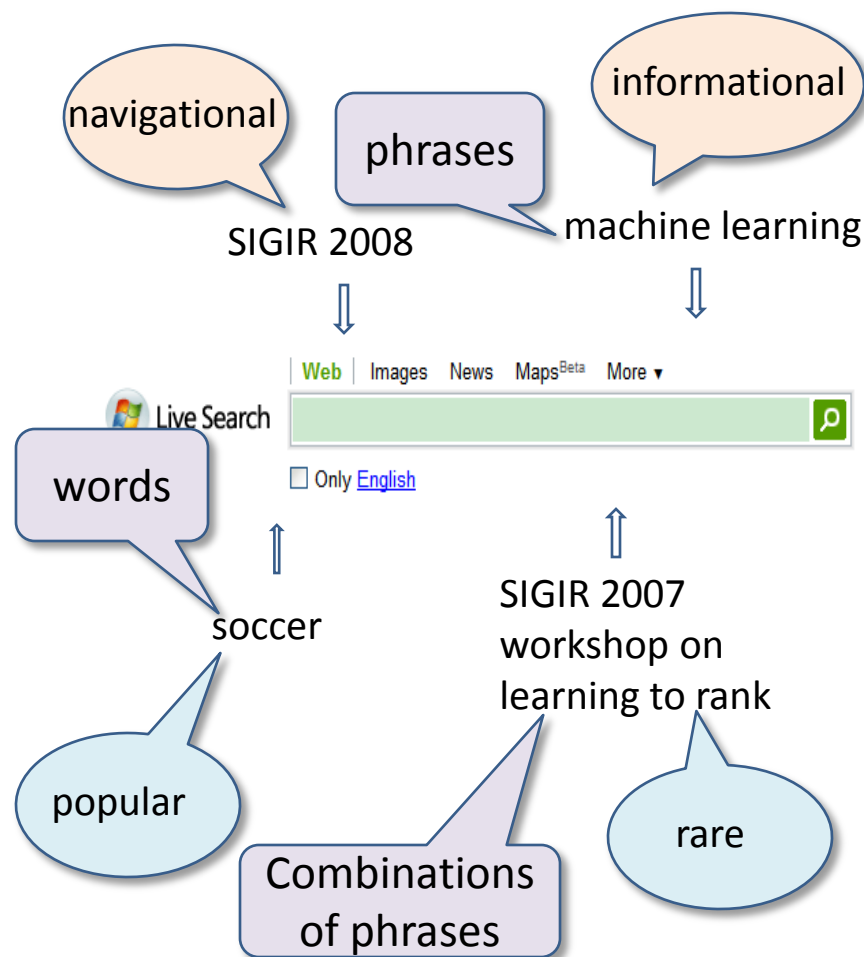
Topic Distillation

Other Work on Relational Ranking

- Learning Diverse Rankings with Multi-Armed Bandits (F. Radlinski, R. Kleinberg, et al. ICML 2008)
- Predicting Diverse Subsets Using Structural SVMs (Y. Yue and T. Joachims, ICML 2008).

Beyond Single Ranking Function

- Queries are quite diverse, different ranking functions might need to be used to rank documents associated with different queries.



Query-dependent ranking using K-Nearest Neighbors

(X. Geng, T. Liu, et al. SIGIR 2008)

- Use the query level features to define the similarity among queries.
- Use the most similar training queries to the test query to train the model online.
- Further move the costly computations to offline.
- Theoretical guarantee of the difference in accuracy between the offline version and the online version.

Beyond Linear Combination of Features

- Learning with IR model
 - Existing learning to rank work assumes features to be pre-computed, with a specific (or default) parameter.
 - Learning process only focus on the combination of the features.
 - Can we also learn the optimal parameters of the features from the training data?
- Other work
 - Supervised Rank Aggregation (Y. Liu, T. Liu, et al. WWW 2007)
 - Ranking Refinement and Its Application to Information Retrieval (R. Jin, H. Valizadegan, et al. WWW 2008)

Ranking Function Discovery using GA

(W. Fan, M. Gordon, et al. HICSS 2004)

- Use *tf*, *idf*, document length, etc. as “low-level” features
- Use +, -, *, /, log as operators.
- Use GA to discover the best ranking function.

$$\log \left(\frac{tf_Doc}{tf_max_Doc} \times \frac{df_max_Doc}{df_Doc} \times \frac{length_avg_Abstract_Col}{tf_avg_Abstract} \right)$$

Normalized *tf*

A new normalized *idf*

Structural part: average *tf* in the abstract of all documents

Optimizing Ranking Function with Multiple Parameters

(M. Taylor, H. Zaragoza, et al. CIKM 2006)

- Use BM25F as the ranking model, and use RankNet as the learning machine to tune its parameter based on the training data.
- The performance was slightly improved over the default parameters in BM25F.
 - Not statistically significant.
 - Possible explanation: the BM25 model has been “perfectly” manually tuned in the literature.

Learning to Rank Networked Entities

(A. Agarwal, S. Chakrabarti, et al. KDD 2006)

- Use MaxEnt to learn optimal PageRank

$$\min_{\{0 \leq p_{uv} \leq 1\}} \sum_{(u,v) \in E'} p_{uv} \log p_{uv} \quad (\text{HardObjective})$$

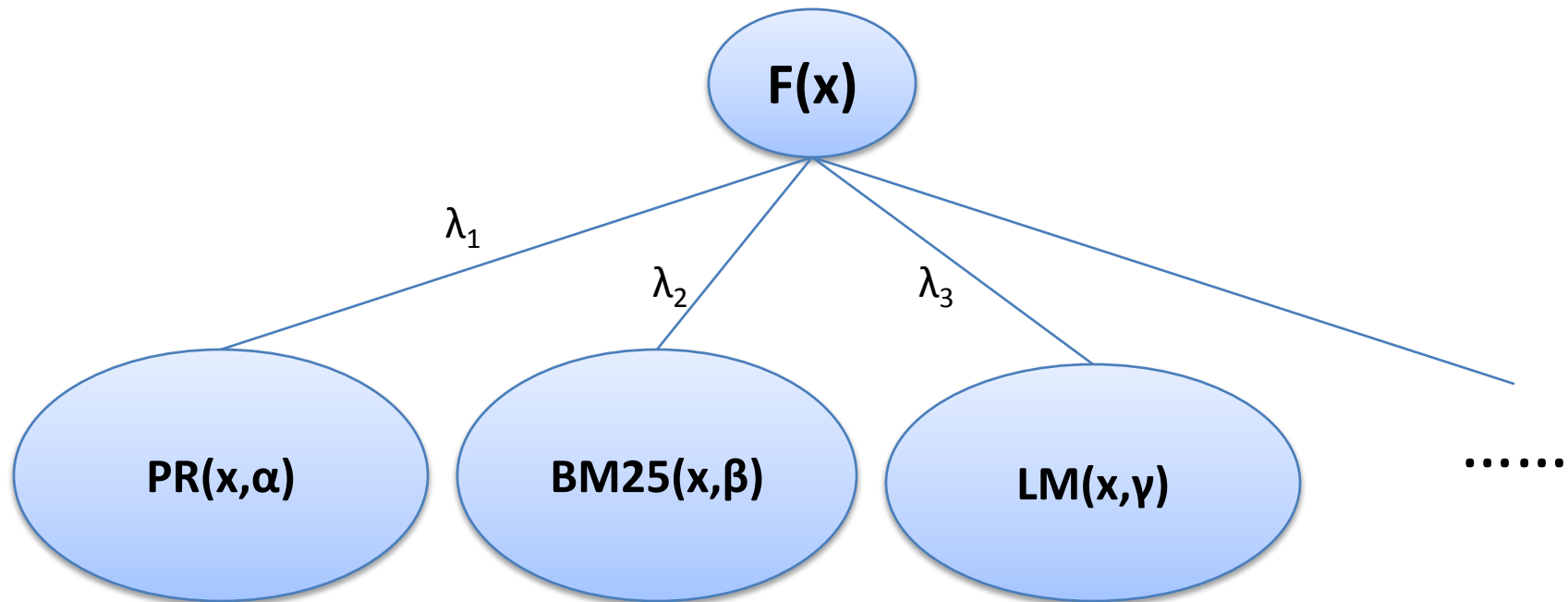
$$\text{such that } \sum_{(u,v) \in E'} p_{uv} - 1 = 0 \quad (\text{Total})$$

$$\forall v \in V' : - \sum_{(u,v) \in E'} p_{uv} + \sum_{(v,w) \in E'} p_{vw} = 0 \quad (\text{Balance})$$

$$\forall v \in V_o : - \alpha p_{vd} + (1 - \alpha) \sum_{(v,w) \in E} p_{vw} = 0 \quad (\text{Teleport})$$

$$\forall u \prec v : \sum_{(w,u) \in E'} p_{wu} - \sum_{(w,v) \in E'} p_{wv} \leq 0 \quad (\text{Preference})$$

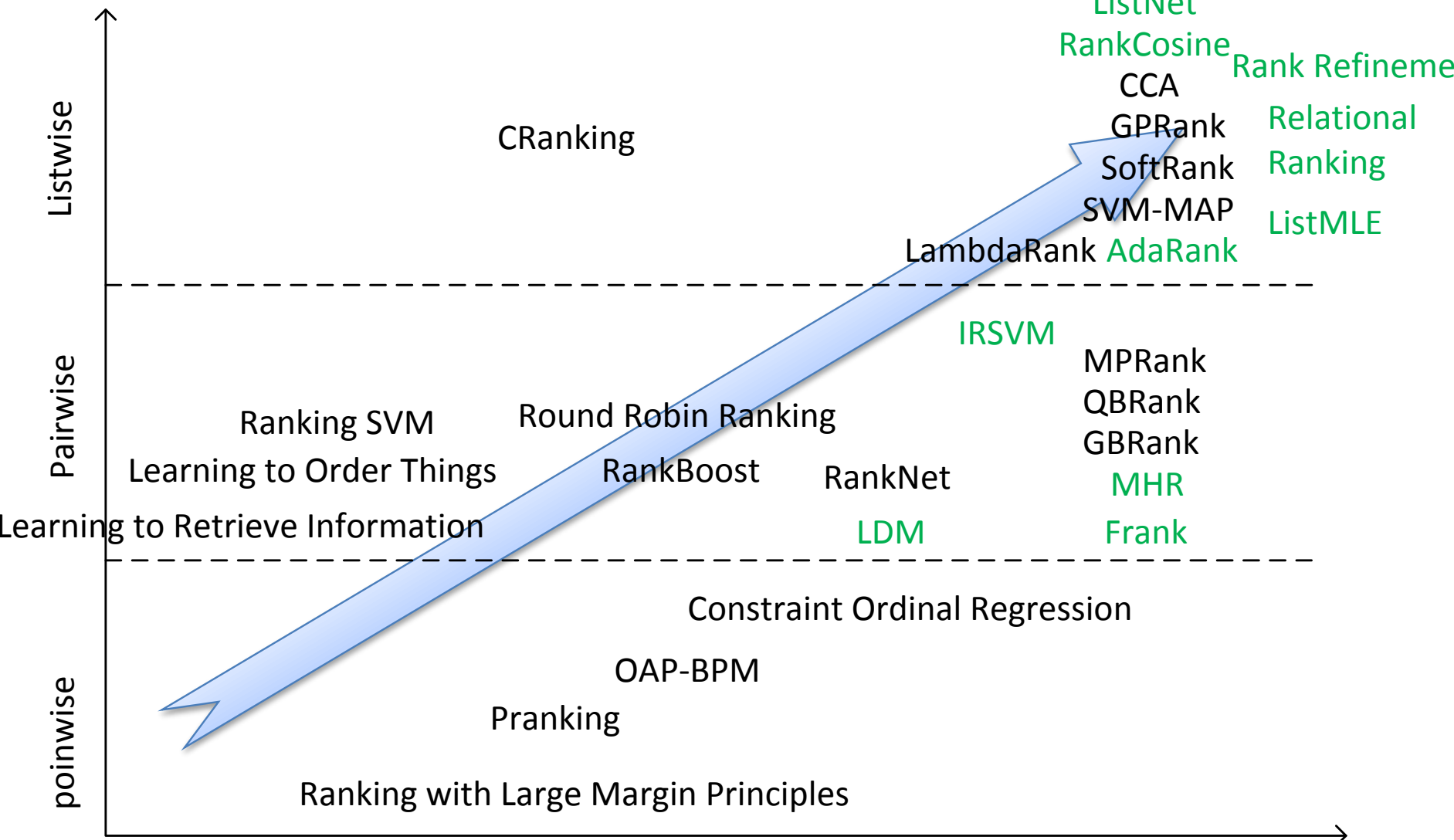
Two Layer Ranking Model



λ , α , β , γ , etc. are all parameters to learn within the 2-layer model

Appendix

* Papers published by our group



Other Topics in Learning to Rank

- Ground Truth Mining / mining rankings directly from logs
 - Active Exploration for Learning Rankings from Clickthrough Data (F. Radlinski, T. Joachims, KDD 2007)
 - Query Chains: Learning to Rank from Implicit Feedback (F. Radlinski, T. Joachims, KDD 2005)
- Feature Selection
 - Feature Selection for Ranking (X. Geng, T. Liu, et al. SIGIR 2007)
- Active Learning
 - SVM Selective Sampling for Ranking with Application to Data Retrieval (H. Yu, KDD 2005)
 - Active Exploration for Learning Rankings from Clickthrough Data (F. Radlinski, T. Joachims, KDD 2007)
-

References (1)

- P. McCullagh, Regression models for ordinal data. Journal of the Royal Statistical Society, 1980
- N. Fuhr. Optimum polynomial retrieval functions based on the probability ranking principle, TOIS, 1989.
- T. Bartell, G. W. Cottrell, et al. Automatic combination of multiple ranked retrieval systems. SIGIR 1994.
- T. Bartell, G. W. Cottrell, learning to retrieval information, SCC, 1995.
- W. W. Cohen, R. E. Shapire, et al. Learning to order things, Journal of Artificial intelligence research, 1999.
- R. Herbrich, T. Graepel, et al. Support Vector Learning for Ordinal Regression, ICANN1999
- R. Herbrich, T. Graepel, et al. Large Margin Rank Boundaries for Ordinal Regression, Advances in large margin classifiers, 2000
- S. Kramer, G. Widmer, et al. Prediction of ordinal classes using regression trees. Fundamenta Informaticae, 2000.
- T. Joachims, Optimizing Search Engines Using Clickthrough Data, KDD 2002.
- A. Shashua, A. Levin, Ranking with large margin principle: Two approaches. NIPS 2002.
- G. Lebanon, and J. Lafferty, Cranking: combining rankings using conditional probability model on permutations, ICML 2002 .
- K. Crammer, Y. Singer, PRanking with ranking, NIPS 2002.

References (2)

- Y. Freund, R. Iyer, et al. An Efficient Boosting Algorithm for Combining Preferences, JMLR 2003.
- F. Harrington. Online ranking/collaborative filtering using the perceptron algorithm. ICML 2003
- S. Rajaram, A. Garg, et al. Classification approach towards ranking and sorting problems, 2003.
- R. Nallapati, Discriminative model for information retrieval, SIGIR 2004.
- I. Tsochantaridis, T. Hofmann, et al. Support vector machine learning for interdependent and structured output space, ICML 2004.
- W. Fan, E. A. Fox, et al. The effects of fitness functions on genetic programming based ranking discovery for web search, Journal of the American Society for Information Science and Technology, 2004.
- W. Fan, M. D. Gordon, et al. Discovery of context-specific ranking functions for effective information retrieval using genetic programming. IEEE TKDE, 2004.
- W. Fan, M. D. Gordon, et al. A generic ranking function discovery framework by genetic programming for information retrieval, IPM, 2004.
- W. Fan, M. D. Gordon, P. Pathak, W. Xi and E. A. Fox, Ranking function optimization for effective Web search by Genetic Programming: an empirical study, in the Proceedings of 37th Hawaii International Conference on System Sciences (HICSS), 2004.

References (3)

- J. Gao, H. Qi, et al. Linear discriminant model for information retrieval, SIGIR 2005.
- C.J.C. Burges, T. Shaked, et al. Learning to Rank using Gradient Descent, ICML 2005.
- I. Tsochantaridis, T. Joachims, et al. Large Margin Methods for Structured and Interdependent Output Variables, JMLR, 2005.
- F. Radlinski, T. Joachims, Query Chains: Learning to Rank from Implicit Feedback, KDD 2005.
- A. Trotman, Learning to rank, Information Retrieval, 2005
- B. Taskar, V. Chatalbashev, et al. Learning structured prediction models: a large margin application. ICML 2005.
- D. Metzler, W. B. Croft, et al. Direct maximization of rank-based metrics for information retrieval, CIIR, 2005
- H. Yu, SVM Selective sampling for ranking with application to data retrieval, KDD 2005.
- I. Tsochantaridis, T. Hofmann, et al. large margin methods for structured and interdependent output variables, JMLR, 2005.
- T. Joachims, A support vector method for multivariate performance measures, ICML 2005.
- W. Chu, and Z. Ghahramani. Gaussian processes for ordinal regression, JMLR, 2005.
- W. Chu, and Z. Ghahramani. Preference learning with Gaussian processes, ICML, 2005.

References (4)

- W. Chu, S. Sathya Keerthi, et al. New approaches to support vector ordinal regression. ICML 2005.
- W. Fan, M. D. Gordon, et al. Genetic programming based discovery of ranking functions for effective web search, journal of management of information system, 2005.
- I. Matveeva, C.J.C. Burges, et al. High accuracy retrieval with multiple nested ranker, SIGIR 2006.
- C.J.C. Burges, R. Ragno, et al. Learning to Rank with Nonsmooth Cost Functions , NIPS 2006
- Y. Cao, J. Xu, et al. Adapting Ranking SVM to Information Retrieval, SIGIR 2006.
- Cartetette, D. Petkova, learning a ranking from pairwise preferences, SIGIR 2006.
- D. Cossock, and T. Zhang, Subset ranking using regression. COLT, 2006.
- J. Xu, Y. Cao, et al. Cost-sensitive learning of SVM for ranking, ECML 2006.
- M. Taylor, H. Zaragoza, et al. Optimisation methods for ranking functions with multiple parameters, CIKM 2006.
- W. Fan, M. D. Gordon, et al. On linear mixture of expert approaches to information retrieval. Decision support systems. 2006.
- Z. Cao, T. Qin, et al. Learning to Rank: From Pairwise to Listwise Approach, ICML 2007.
- T. Qin, T.-Y. Liu, et al, Multiple hyperplane Ranker, SIGIR 2007.

References (5)

- T.-Y. Liu, J. Xu, et al. LETOR: Benchmark dataset for research on learning to rank for information retrieval, LR4IR 2007.
- M.-F. Tsai, T.-Y. Liu, et al. FRank: A Ranking Method with Fidelity Loss, SIGIR 2007.
- Z. Zheng, H. Zha, et al. A General Boosting Method and its Application to Learning Ranking Functions for Web Search, NIPS 2007. Z. Zheng, H. Zha, et al, A Regression Framework for Learning Ranking Functions Using Relative Relevance Judgment, SIGIR 2007.
- M. Taylor, J. Guiver, et al. SoftRank: Optimising Non-Smooth Rank Metrics, LR4IR 2007.
- J.-Y. Yeh, J.-Y. Lin, et al. Learning to Rank for Information Retrieval using Generic Programming, LR4IR 2007.
- T. Qin, T.-Y. Liu, et al, Query-level Loss Function for Information Retrieval, Information Processing and Management, 2007.
- X. Geng, T.-Y. Liu, et al, Feature Selection for Ranking, SIGIR 2007.
- Y. Yue, T. Finley, et al. A Support Vector Method for Optimizing Average Precision, SIGIR 2007.
- Y.-T. Liu, T.-Y. Liu, et al, Supervised Rank Aggregation, WWW 2007.
- J. Xu and H. Li, A Boosting Algorithm for Information Retrieval, SIGIR 2007.
- F. Radlinski, T. Joachims. Active Exploration for Learning Rankings from Clickthrough Data, KDD 2007.

References (6)

- P. Li, C. Burges, et al. McRank: Learning to Rank Using Classification and Gradient Boosting, NIPS 2007.
- C. Cortes, M. Mohri, et al. Magnitude-preserving Ranking Algorithms, ICML 2007.
- H. Almeida, M. Goncalves, et al. A combined component approach for finding collection-adapted ranking functions based on genetic programming, SIGIR 2007.
- S. Robertson, and H. Zaragoza. On rank-based effectiveness measures and optimization. Information Retrieval, 2007.
- T. Qin, T.-Y. Liu, et al, Learning to Rank Relational Objects and Its Application to Web Search, WWW 2008.
- R. Jin, H. Valizadegan, et al. Ranking Refinement and Its Application to Information Retrieval, WWW 2008.
- F. Xia, T.-Y. Liu, et al. Listwise Approach to Learning to Rank – Theory and Algorithm, ICML 2008.
- Y. Lan, T.-Y. Liu, et al. On Generalization Ability of Learning to Rank Algorithms, ICML 2008.

Acknowledgement

- Hang Li (Microsoft Research Asia)
- Wei-Ying Ma (Microsoft Research Asia)
- Jun Xu (Microsoft Research Asia)
- Tao Qin (Microsoft Research Asia)

Thanks!

tyliu@microsoft.com

<http://research.microsoft.com/users/tyliu/>