LETOR
Learning to Rank
for Information Retrieval

Microsoft Research

A tutorial at WWW 2009

Learning to Rank for Information Retrieval

Tie-Yan Liu
Microsoft Research Asia

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.

Background Knowledge Required

- Information Retrieval.
- Machine Learning.
- Probability Theory.

Outline

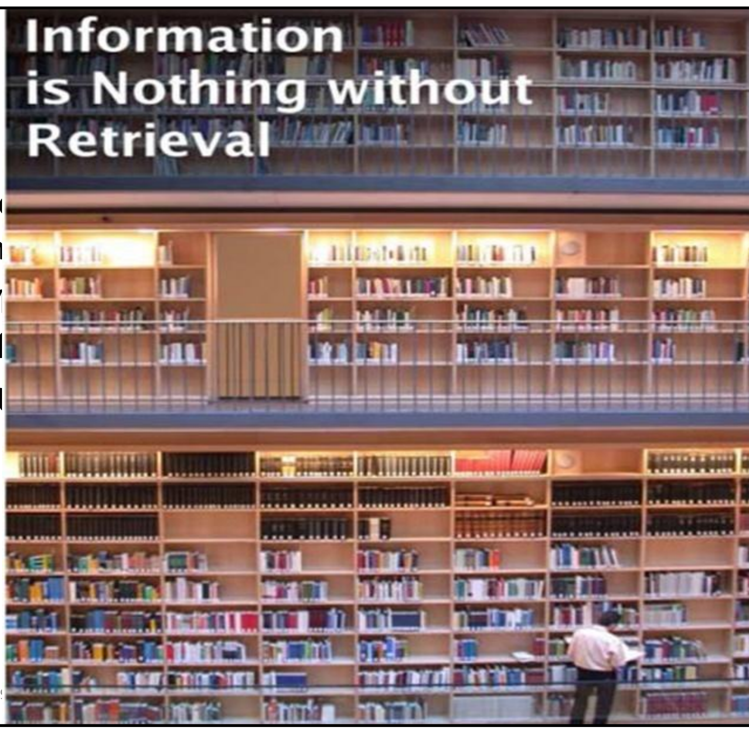
- Introduction
- Learning to Rank Algorithms
  - Pointwise approach
  - Pairwise approach
  - Listwise approach
  - Analysis of the approaches
- Statistical Ranking Theory
  - Query-level ranking framework
  - Generalization analysis for ranking
- Benchmarking Learning to Rank methods
- Summary

Introduction

Overwhelmed by Flood of Information

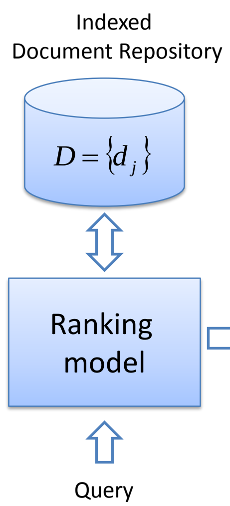
Facts about the Web

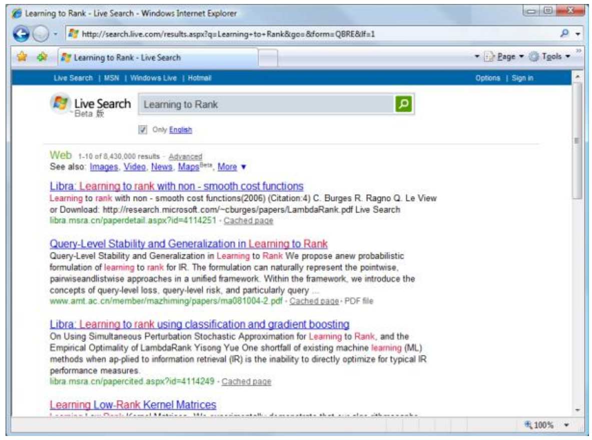
- According to www.worldwidewebsize.com, there are more than 25 billion pages on the Web.
- Major search engines indexed at least tens of billions of web pages.
- CUIL.com indexed more than 120 Billion web pages.



- A… e
m…
- M… ns
of…
- Cl… .

Ranking is Essential





Applications of Ranking

- Document retrieval
- Collaborative filtering
- Key term extraction
- Definition finding
- Important email routing
- Sentiment analysis,
- Product rating
- Anti Web spam
- ……

Scenarios of Ranking
(Document Retrieval as Example)

- Rank the documents purely according to their relevance with regards to the query.
- Consider the relationships of similarity, website structure, and diversity between documents in the ranking process (relational ranking).
- Aggregate several candidate ranked lists to get a better ranked list (meta search).
- Find whether and to what degree a property of a webpage influences the ranking result (reverse engineering).
- …

Evaluation of Ranking Results

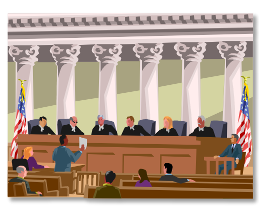
- Construct a test set containing a large number of (randomly sampled) queries, their associated documents, and relevance judgment for each query-document pair.
- Evaluate the ranking result for a particular query in the test set with an evaluation measure.
- Use the average measure over the entire test set to represent the overall ranking performance.

Collecting Documents for A Query

- Pooling strategy used in TREC
  - A pool of possibly relevant documents is created by taking a sample of documents selected by the various participating systems.
  - Top 100 documents retrieved in each submitted run for a given query are selected and merged into the pool for human assessment.
  - On average, an assessor judges the relevance of approximately 1500 documents per query.

Relevance Judgment

- Degree of relevance
  - Binary: relevant vs. irrelevant
  - Multiple ordered categories: Perfect > Excellent > Good > Fair > Bad
- Pairwise preference
  - Document *A* is more relevant than document *B*
- Total order
  - Documents are ranked as {A,B,C,..} according to their relevance



Evaluation Measure - MAP

- Precision at position *k* for query *q:*

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

- Average precision for query *q*:

$$AP = \frac{\sum_k P@k \cdot l_k}{\#\{\text{relevant documents}\}}$$

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

- MAP: averaged over all queries.

Evaluation Measure - NDCG

- NDCG at position *n* for query *q*:

$$NDCG@k = Z_k \sum_{j=1}^{k} G\left(\pi^{-1}(j)\right)\eta(j)$$

Normalization **Cumulating** **Gain** **Position discount**

$$G\left(\pi^{-1}(j)\right) = 2^{l_{\pi^{-1}(j)}} - 1, \quad \eta(j) = 1/\log(j+1)$$

- Averaged over all queries.

Evaluation Measure - Summary

- Query-level: every query contributes equally to the measure.
  - Computed on documents associated with the same query.
  - Bounded for each query.
  - Averaged over all test queries.
- Position-based: rank position is explicitly used.
  - Top-ranked objects are more important.
  - Relative order vs. relevance score of each document.
  - Non-continuous and non-differentiable w.r.t. scores

Conventional Ranking Models

- Query-dependent
  - 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.
- Query-independent
  - PageRank, TrustRank, BrowseRank, etc.

Problems with Conventional Models

- Manual parameter tuning is usually difficult, especially when there are many parameters and the evaluation measures are non-smooth.
- Manual parameter tuning sometimes leads to over-fitting.
- It is non-trivial to combine the large number of models proposed in the literature to obtain an 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”
  - In general, those methods that use machine learning technologies to solve the problem of ranking can be named as “learning to rank” methods.

Learning to Rank

- In most recent works, learning to rank is defined as having the following two properties:
  - Feature based;
  - Discriminative training.

Feature Based

- Documents represented by feature vectors.
  - Even if a feature is the output of an existing retrieval model, one assumes that the parameter in the model is fixed, and only learns the optimal way of combining these features.
- The capability of combining a large number of features is very promising.
  - It can easily incorporate any new progress on retrieval model, by including the output of the model as a feature.

Discriminative Training

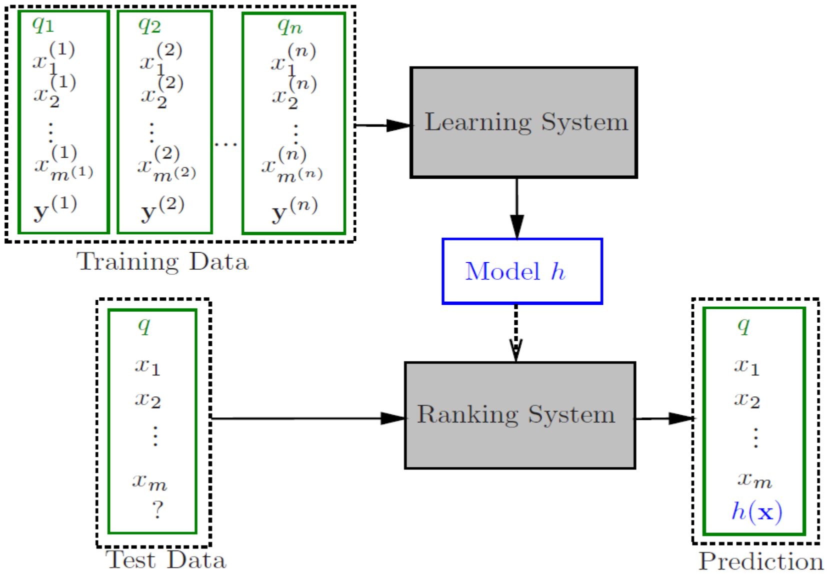
- An automatic learning process based on the training data,
  - With the following four pillars: Input space, output space, hypothesis space, loss function.
  - Not even necessary to have a probabilistic explanation.
- This is highly demanding for real search engines,
  - Everyday these search engines will receive a lot of user feedback and usage logs
  - It is very important to automatically learn from the feedback and constantly improve the ranking mechanism.

Learning to Rank

- In most recent works, learning to rank is defined as having the following two properties:
  - Feature based;
  - Discriminative training.

- Automatic parameter tuning for existing ranking models, and learning-based generative ranking models do not belong to the current definition of "learning to rank."

Learning to Rank Framework



Learning to Rank Algorithms

Least Square Retrieval Function (TOIS 1989)
Query refinement (WWW 2008)
SVM-MAP (SIGIR 2007)
Nested Ranker (SIGIR 2006)
ListNet (ICML 2007)
Pranking (NIPS 2002)
LambdaRank (NIPS 2006)
MPRank (ICML 2007)
Frank (SIGIR 2007)
MHR (SIGIR 2007)
RankBoost (JMLR 2003)
Learning to retrieval info (SCC 1995)
LDM (SIGIR 2005)
Large margin ranker (NIPS 2002)
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)

Key Questions to Answer

- To what respect are these learning to rank algorithms similar and in which aspects do they differ? What are the strengths and weaknesses of each algorithm?
- What are the unique theoretical issues for ranking that should be investigated?
- Empirically speaking, which of those many learning to rank algorithms perform the best?
- Are there many remaining issues regarding learning to rank to study in the future?

Learning to Rank Algorithms

Categorization of the Algorithms

Category	Algorithms
Pointwise Approach	**Regression**: Least Square Retrieval Function (TOIS 1989), Regression Tree for Ordinal Class Prediction (Fundamenta Informaticae, 2000), Subset Ranking using Regression (COLT 2006), … **Classification**: Discriminative model for IR (SIGIR 2004), McRank (NIPS 2007), … **Ordinal regression**: 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), GBRank (SIGIR 2007), QBRank (NIPS 2007), MPRank (ICML 2007), IRSVM (SIGIR 2006), …
Listwise Approach	**Listwise loss minimization**: RankCosine (IP&M 2008), ListNet (ICML 2007), ListMLE (ICML 2008), … **Direct optimization of IR measure**: LambdaRank (NIPS 2006), AdaRank (SIGIR 2007), SVM-MAP (SIGIR 2007), SoftRank (LR4IR 2007), GPRank (LR4IR 2007), CCA (SIGIR 2007), …

Characterizing the Approaches

- Four pillars of discriminative learning
  – Input space
  – Output space
  – Hypothesis space
  – Loss function

The Pointwise Approach

	The Pointwise Approach		
	Regression	Classification	Ordinal Regression
Input Space	Single documents y_j		
Output Space	Real values	Non-ordered Categories	Ordinal categories
Hypothesis Space	Scoring function $f(x)$		
Loss Function	Regression loss	Classification loss	Ordinal regression loss
	$L(f; x_j, y_j)$		

The Pointwise Approach

- Reduce ranking to
  - Regression
    - Subset Ranking
  - Classification
    - Discriminative model for IR
    - MCRank
  - Ordinal regression
    - Pranking
    - Ranking with large margin principle

$$q \leftrightarrow \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{pmatrix}$$

$$\Downarrow$$

$$\{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}$$

Subset Ranking using Regression

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

- Regard relevance degree as real number, and use regression to learn the ranking function.

$$L(f; x_j, y_j) = \left| f(x_j) - y_j \right|^2$$

Discriminative Model for IR

(R. Nallapati, SIGIR 2004)

- Classification is used to learn the ranking function
  - Treat relevant documents as positive examples (y_j=+1), while irrelevant documents as negative examples (y_j=-1).
  - $L(f; x_j, y_j)$ is surrogate loss of $I_{\{f(x_j) \neq y_j\}}$
- Different algorithms optimize different surrogate losses.
  - e.g., Support Vector Machines (hinge loss)

McRank

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

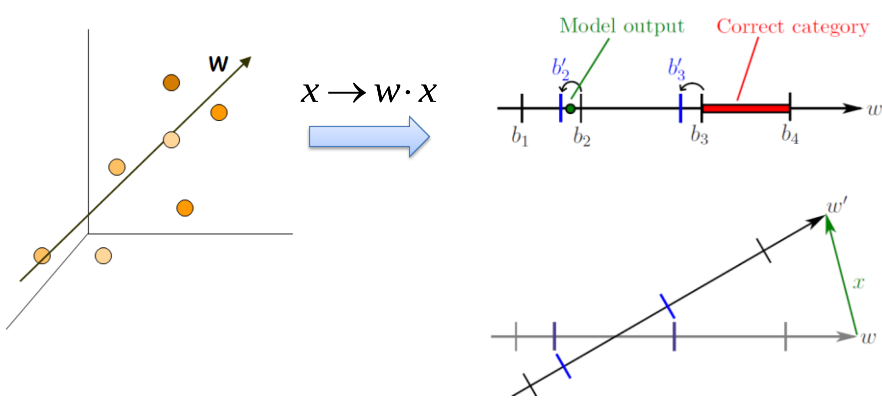
- Multi-class classification is used to learn the ranking function.
- Ranking is produced by combining the outputs of the classifiers.

$$\hat{p}_{j,k} = P(y_j = k), \quad f(x_j) = \sum_{k=0}^{K-1} \hat{p}_{j,k} k$$

Pranking with Ranking

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

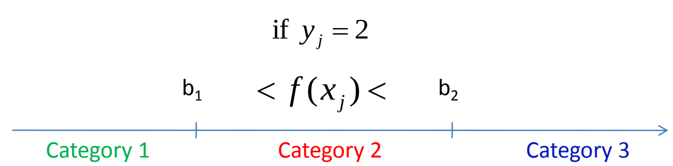
- Ranking is solved by ordinal regression



Ranking with Large Margin Principles

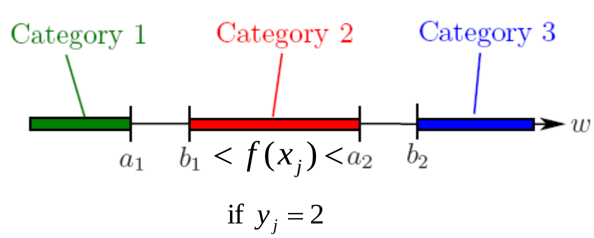
(A. Shashua and A. Levin, NIPS 2002)

- Fixed margin strategy
  - Margin: $2/\|w\|^2$
  - Constraints: every document is correctly placed in its corresponding category.



Ranking with Large Margin Principles

- Sum of margins strategy
  - Margin: $\sum_k (b_k - a_k)$
  - Constraints: every document is correctly placed in its corresponding category



Problem with Pointwise Approach

- Properties of IR evaluation measures have not been well considered.
  - The fact is ignored that some documents are associated with the same query and some are not. When the number of associated documents varies largely for different queries, the overall loss function will be dominated by those queries with a large number of documents.
  - The position of documents in the ranked list is invisible to the loss functions. The pointwise loss function may unconsciously emphasize too much those unimportant documents.

The Pairwise Approach

	The Pairwise Approach
Input Space	Document pairs (x_u, x_v)
Output Space	Preference $y_{u,v} \in \{+1, -1\}$
Hypothesis Space	Preference function $h(x_u, x_v) = 2 \cdot I_{\{f(x_u) > f(x_v)\}} - 1$
Loss Function	Pairwise classification loss $L(h; x_u, x_v, y_{u,v})$

The Pairwise Approach

- Reduce ranking to pairwise classification
  - Learning to order things
  - RankNet and Frank
  - RankBoost
  - Ranking SVM
  - Multi-hyperplane ranker
  - IR-SVM

$$q \leftrightarrow \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{pmatrix}$$

$$\Downarrow$$

$$\begin{Bmatrix} (x_1, x_2, +1), (x_2, x_1, -1), \ldots, \\ (x_2, x_m, +1), (x_m, x_2, -1) \end{Bmatrix}$$

Learning to Order Things

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

- Pairwise loss function

$$L(h; x_u, x_v, y_{u,v}) = \frac{\left| y_{u,v} - h(x_u, x_v) \right|}{2}$$

- Pairwise ranking function

$$h(x_u, x_v) = \sum w_t h_t(x_u, x_v)$$

  - Weighted majority algorithm, e.g., the Hedge algorithm, is used to learn the parameters *w*.
  - From preference to total order: $\max_{\pi} \sum_{u<v} h(x_{\pi(u)}, x_{\pi(v)})$

RankNet

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

- Target probability:

$$\overline{P}_{u,v}=1, \text{if } y_{u,v}=1;\ \overline{P}_{u,v}=0, \text{otherwise}$$

- Modeled probability:

$$P_{u,v}(f)\equiv\frac{\exp(f(x_u)-f(x_v))}{1+\exp(f(x_u)-f(x_v))}$$

- Cross entropy as the loss function

$$l(f;x_u,x_v,y_{u,v})$$
$$=-\overline{P}_{u,v}\log P_{u,v}(f)-(1-\overline{P}_{u,v})\log(1-P_{u,v}(f))$$

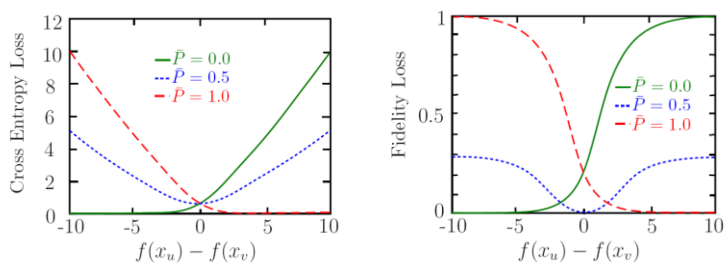
- Use Neural Network as model, and gradient descent as algorithm, to optimize the cross-entropy loss.

FRank

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

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

$$L(f;x_u,x_v,y_{u,v})$$
$$=1-\sqrt{\overline{P}_{u,v}\cdot P_{u,v}(f)}-\sqrt{(1-\overline{P}_{u,v})\cdot(1-P_{u,v}(f))}$$



RankBoost

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

- Exponential loss

$$L(f;x_u,x_v,y_{u,v})=\exp(-y_{u,v}(f(x_u)-f(x_v)))$$

Algorithm 1 Learning Algorithm for RankBoost

Input: document pairs

Given: initial distribution D_1 on input document pairs.

For $t=1,\cdots,T$

Train weak ranker f_t based on distribution D_t.

Choose α_t

Update $D_{t+1}(x_u^{(i)},x_v^{(i)})=\frac{1}{Z_t}D_t(x_u^{(i)},x_v^{(i)})\exp(\alpha_t(f_t(x_u^{(i)})-f_t(x_v^{(i)})))$

where $Z_t=\sum_{i=1}^{n}\sum_{u,v:y_{u,v}^{(i)}=1}D_t(x_u^{(i)},x_v^{(i)})\exp(\alpha_t(f_t(x_u^{(i)})-f_t(x_v^{(i)})))$.

Output: $f(x)=\sum_t \alpha_t f_t(x)$.

Ranking SVM

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

- Hinge loss

$$L(f;x_u,x_v,y_{u,v})=\left(1-y_{u,v}(f(x_u)-f(x_v))\right)_+$$

$$\min \frac{1}{2}\|w\|^2+C\sum_{i=1}^{n}\sum_{u,v:y_{u,v}^{(i)}=1}\xi_{u,v}^{(i)}$$

$$w^T\left(\boxed{x_u^{(i)}-x_v^{(i)}}\right)\geq 1-\xi_{u,v}^{(i)}, \text{if } y_{u,v}^{(i)}=1.$$

$$\xi_{uv}^{(i)}\geq 0, i=1,\ldots,n.$$

x_u-x_v as positive instance of learning

Use SVM to perform binary classification on these instances, to learn model parameter *w*

Extensions of the Pairwise Approach

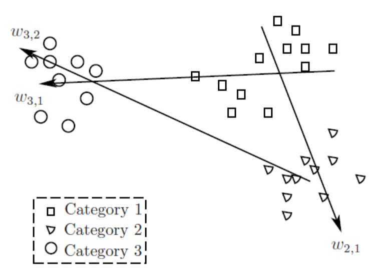
- When constructing document pairs, category information has been lost: different pairs of labels are treated identically.
  - Can we maintain more information about ordered category?
  - Can we use different learner for different kinds of pairs?
- Solutions
  - Multi-hyperplane ranker

Multi-Hyperplane Ranker

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

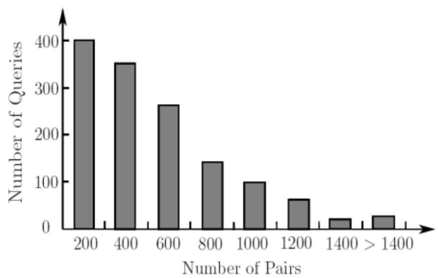
- If we have *K* categories, then will have *K*(*K*-1)/2 pairwise preferences between two labels.
  - Learn a model $f_{k,l}$ for the pair of categories *k* and *l*, using any previous method (for example Ranking SVM).
- Testing
  - Rank Aggregation

$$f(x)=\sum_{k,l}\alpha_{k,l}f_{k,l}(x)$$



Improvement of Pairwise Approach

- Advantage
  - Predicting relative order is closer to the nature of ranking than predicting class label or relevance score.
- Problem
  - The distribution of document pair number is more skewed than the distribution of document number, with respect to different queries.

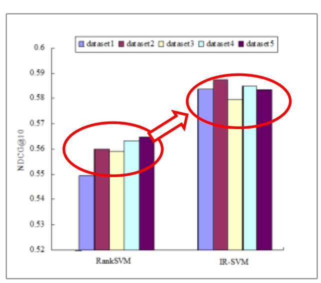


IR-SVM

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

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

Query-level normalizer



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

The Listwise Approach

	The Listwise Approach	
	Listwise Loss Minimization	Direct Optimization of IR Measure
Input Space	Document set $\mathbf{x} = \{x_j\}_{j=1}^m$	
Output Space	Permutation π_y	Ordered categories $\mathbf{y} = \{y_j\}_{j=1}^m$
Hypothesis Space	$h(\mathbf{x}) = \mathrm{sort} \circ f(\mathbf{x})$	$h(\mathbf{x}) = f(\mathbf{x})$
Loss Function	Listwise loss $L(h; \mathbf{x}, \pi_y)$	1-surrogate measure $L(h; \mathbf{x}, \mathbf{y})$

Listwise Ranking: Two Major Branches

- Direct optimization of IR measures
  - Try to optimize IR evaluation measures, or at least something correlated to the measures.
- Listwise loss minimization
  - Minimize a loss function defined on permutations, which is designed by considering the 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-continuous and non-differentiable since they depend on the rank positions. *(S. Robertson and H. Zaragoza, Information Retrieval 2007; J. Xu, T.-Y. Liu, et al. SIGIR 2008).*
  - It is challenging to optimize such objective functions, since most optimization techniques in the literature were developed to handle continuous and differentiable cases.

Tackle the Challenges

- Approximate the objective
  - Soften (approximate) the evaluation measure so as to make it smooth and differentiable (SoftRank)
- Bound the objective
  - Optimize a smooth and differentiable upper bound of the evaluation measure (SVM-MAP)
- Optimize the non-smooth objective directly
  - Use IR measure to update the distribution in Boosting (AdaRank)
  - Use genetic programming (RankGP)

SoftRank

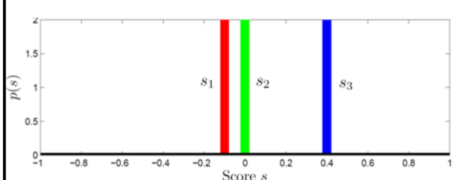
(M. Taylor, J. Guiver, et al. LR4IR 2007/WSDM 2008)

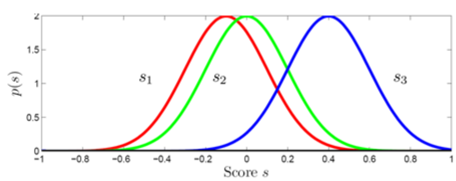
- Key Idea:
  - Avoid sorting by treating scores as random variables
- Steps
  - Construct score distribution
  - Map score distribution to rank distribution
  - Compute expected NDCG (SoftNDCG) which is smooth and differentiable.
  - Optimize SoftNDCG by gradient descent.

Construct Score Distribution

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

$$p(s_j) = N(s_j \mid f(x_j), \sigma_s^2)$$





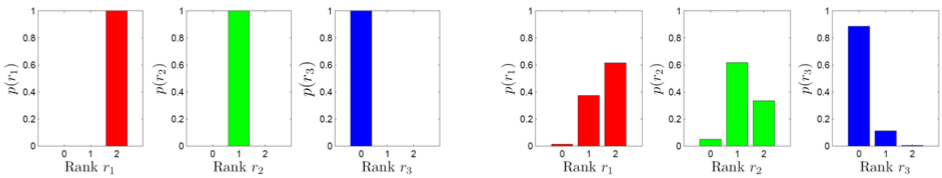
From Scores to Rank Distribution

- Probability of d_u being ranked before d_v

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

- 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

- Expected NDCG as the objective, or in the language of loss function,

$$L(f;\mathbf{x},\mathbf{y}) = 1 - E[NDCG] = 1 - Z_m \sum_{j=1}^{m} (2^{y_j} - 1) \sum_{r=0}^{m-1} \eta_r p_j(r)$$

- A neural network is used as the model, and gradient descent is used as the algorithm to optimize the above loss function.

SVM-MAP

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

- Use the framework of Structured SVM for optimization
  - I. Tsochantaridis, et al. ICML 2004; T. Joachims, ICML 2005.
- *Sum of slacks $\sum\xi$ upper bounds $\sum$ (1-AP).*

$$\min \frac{1}{2}\|w\|^2 + \frac{C}{N}\sum_{i=1}^{N}\xi^{(i)}$$

$$\forall y' \neq \mathbf{y}^{(i)}: \quad w^T\boxed{\Psi(\mathbf{y}^{(i)},\mathbf{x}^{(i)})} \geq w^T\Psi(\mathbf{y}',\mathbf{x}^{(i)}) + \boxed{\Delta(\mathbf{y}^{(i)},\mathbf{y}')} - \xi^{(i)}$$

$$\Psi(\mathbf{y}',\mathbf{x}) = \sum_{u,v:y_u=1,y_v=0}(y'_u - y'_v)(x_u - x_v)$$

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

Challenges in SVM-MAP

- For AP, 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’: $\arg\max_{y'}\left(\Delta(\mathbf{y},\mathbf{y}')+\Psi(\mathbf{y}',\mathbf{x})\right)$
- If the relevance at each position is fixed, Δ(**y**,**y**’) will be the same. But if the documents are sorted by their scores in descending order, the second term will be maximized.
- Strategy (with a complexity of O(*m*log*m*))
  - 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.

AdaRank

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

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

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

Emphasize those hard queries by setting their distributions.

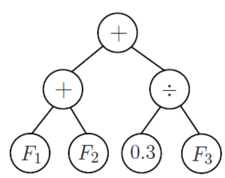
Embed IR evaluation measure in the distribution update of Boosting

4/20/2009 Tie-Yan Liu @ WWW 2009 Tutorial on Learning to Rank 63

RankGP

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

- Define ranking function as a tree



- 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).

4/20/2009 Tie-Yan Liu @ WWW 2009 Tutorial on Learning to Rank 64

Listwise Loss Minimization

- Defining listwise loss functions based on the understanding on the unique properties of ranking for IR.
- Representative Algorithms
  - ListNet
  - ListMLE

Ranking Loss is Non-trivial!

- An example:
  - function *f*: $f(\mathrm{A})=3,\ f(\mathrm{B})=0,\ f(\mathrm{C})=1$ ACB
  - function *h*: $h(\mathrm{A})=4,\ h(\mathrm{B})=6,\ h(\mathrm{C})=3$ BAC
  - ground truth *g*: $g(\mathrm{A})=6,\ g(\mathrm{B})=4,\ g(\mathrm{C})=3$ 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?

f: $\langle 3,0,1\rangle$ g: $\langle 6,4,3\rangle$ Closer! h: $\langle 4,6,3\rangle$

$sim(f,g)=0.85$ $sim(g,h)=0.93$

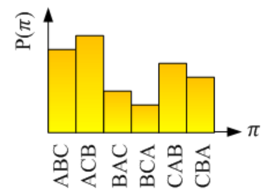
- However, according to NDCG, *f* should be closer to *g*!

Permutation Probability Distribution

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

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

↔



Luce Model: Defining Permutation Probability

- Probability of a permutation π is defined as

$$P(\pi \mid f) = \prod_{j=1}^{m} \frac{\varphi(f(x_{\pi(j)}))}{\sum_{k=j}^{m} \varphi(f(x_{\pi(k)}))}$$

- Example:

$$P(\mathrm{ABC} \mid f) = \boxed{\frac{\varphi(f(\mathrm{A}))}{\varphi(f(\mathrm{A}))+\varphi(f(\mathrm{B}))+\varphi(f(\mathrm{A}))}} \cdot \boxed{\frac{\varphi(f(\mathrm{B}))}{\varphi(f(\mathrm{B}))+\varphi(f(\mathrm{C}))}} \cdot \boxed{\frac{\varphi(f(\mathrm{C}))}{\varphi(f(\mathrm{C}))}}$$

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

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

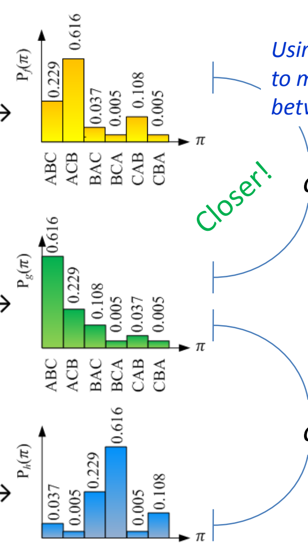
Distance between Ranked Lists

$\varphi = exp$

f: $f(\mathrm{A}) = 3, f(\mathrm{B})=0, f(\mathrm{C})=1$;
Ranking by f: ABC ↔

g: $g(\mathrm{A}) = 6, g(\mathrm{B})=4, g(\mathrm{C})=3$; ↔
Ranking by g: ABC

h: $h(\mathrm{A}) = 4, h(\mathrm{B})=6, h(\mathrm{C})=3$; ↔
Ranking by h: ACB



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

$dis(f,g) = 0.46$

$dis(g,h) = 2.56$

ListNet

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

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

$$L(f;\mathbf{x},\pi_y) = D\left(\boxed{P_y(\pi)} \,\|\, \boxed{P(\pi \mid \varphi(f(\mathbf{x})))}\right)$$

Probability distribution defined by the ground truth

Probability distribution defined by the model output

- Model = Neural Network
- Algorithm = Gradient Descent

Limitations of ListNet

- Too complex to use in practice: O($m \cdot m!$)
- By using the top-k Luce model, the complexity of ListNet can be reduced to polynomial order of *m*.
- However,
  - When k is large, the complexity is still high!
  - When k is small, the information loss in the top-k Luce model will affect the effectiveness of ListNet.

Solution

- 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*·*m*!) to O(*m*).

ListMLE Algorithm

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

- Likelihood Loss

$$L(f; x, \pi_y) = -\log P(\pi_y \mid \varphi(f(\mathbf{x})))$$

- Model = Neural Network
- Algorithm = Stochastic Gradient Descent

Extension of ListMLE

- Dealing with other types of ground truth labels using the concept of "equivalent permutation set".
  - For relevance degree

$$\Omega_y = \{\pi_y \mid u < v, \text{if } l_{\pi_y^{-1}(u)} \succ l_{\pi_y^{-1}(u)}\}$$

$$L(f, \mathbf{x}, \Omega_y) = -\sum_{\pi_y \in \Omega_y} \log P\big(\pi_y \mid \varphi\big(f(w, \mathbf{x})\big)\big)$$

  - For pairwise preference

$$\Omega_y = \{\pi_y \mid u < v, \text{if } l_{\pi_y^{-1}(u), \pi_y^{-1}(u)} = 1\}$$

$$L(f, \mathbf{x}, \Omega_y) = -\max_{\pi_y \in \Omega_y} \log P\big(\pi_y \mid \varphi\big(f(w, \mathbf{x})\big)\big)$$

Discussions

- Advantages
  - Take all the documents associated with the same query as the learning instance
  - Rank position is visible to the loss function
- Problems
  - Complexity issue.
  - The use of the position information is insufficient.

Analysis of the Approaches

- Different loss functions are used in different approaches, while the models learned with all the approaches are evaluated by IR measures.
- Question
  - What are the relationship between the loss functions and IR evaluation measures?

The Pointwise Approach: McRank

Pointwise Approach

- For regression based algorithms (D. Cossock and T. Zhang, COLT 2006)

$$1 - NDCG(f) \le \frac{1}{Z_m}\left(2\sum_{j=1}^{m}\eta_j^{\varepsilon}\right)^{1/\alpha}\left(\sum_{j=1}^{m}\left(f(x_j)-y_j\right)^{\beta}\right)^{1/\beta}$$

- For multi-class classification based algorithms (P. Li, C. Burges, et al. NIPS 2007)

$$1 - NDCG(f) \le \frac{15}{Z_m}\sqrt{2\left(\sum_{j=1}^{m}\eta_j^2 - m\prod_{j=1}^{m}\eta_j^{\frac{2}{m}}\right)\cdot\sum_{j=1}^{m}I_{\{y_j \ne f(x_j)\}}}$$

Pointwise Approach

- Although it seems the loss functions can bound (1-NDCG), the constants before the losses seem too large.

x_i, y_i $\Rightarrow$ $Z_m \approx 21.4$ $\quad$ $x_i, f(x_i)$

$$\begin{pmatrix} x_1,4 \\ x_2,3 \\ x_3,2 \\ x_4,1 \end{pmatrix} \qquad DCG(f) \approx 21.4 \;\Downarrow\; |1-NDCG(f)| = 0 \qquad \Leftarrow \begin{pmatrix} x_1,3 \\ x_2,2 \\ x_3,1 \\ x_4,0 \end{pmatrix}$$

$$\frac{15}{Z_m}\sqrt{2\left(\sum_{j=1}^{m}\left(\frac{1}{\log(j+1)}\right)^2 - m\sum_{j=1}^{m}\left(\frac{1}{\log(j+1)}\right)^{\frac{2}{m}}\right) \cdot \sum_{j=1}^{m} I_{\{y_j \neq f(x_j)\}}} \approx 1.15 > 1$$

Pairwise Approach

- Unified loss
  - Model ranking as a sequence of classifications

Suppose the ground truth permutation is $\{x_1 \succ x_2 \succ x_3 \succ x_4\}$

No	Classification Task
1	$\{x_1\}:\{x_2, x_3, x_4\}$
2	$\{x_2\}:\{x_3, x_4\}$
3	$\{x_3\}:\{x_4\}$

Classifier: $\arg\max_{k \geq t}\{f(x_k)\}$

$$L_t(f) = I_{\left\{\arg\max_{k\geq t}\{f(x_k)\} \neq t\right\}}$$

$$\Downarrow$$

$$L(f) = \sum_t \beta_t I_{\left\{\arg\max_{k\geq t}\{f(x_k)\} \neq t\right\}}$$

$$\Downarrow$$

$$\tilde{L}(f) = \sum_t \beta_t I_{\left\{l_{\arg\max_{k\geq t}\{f(x_k)\}} \neq l_t\right\}}$$

Pairwise Approach

(W. Chen, T.-Y. Liu, et al. 2009)

- Unified loss vs. (1-NDCG)
  - When $\beta_t = \frac{G(t)\eta(t)}{Z_m}$, $\tilde{L}(f)$ is a tight bound of (1-NDCG).
- Surrogate function of Unified loss
  - After introducing weights β_t, loss functions in Ranking SVM, RankBoost, RankNet are *Cost-sensitive Pairwise Comparison* surrogate functions, and thus are *consistent* with and are *upper bounds* of the unified loss.
  - Consequently, they also upper bound (1-NDCG).

Listwise Approach

(W. Chen, T.-Y. Liu, et al. 2009)

- Listwise ranking loss minimization
  - After introducing weights β_t, the loss function in ListMLE is the Unconstrained Background Discriminative surrogate function of the unified loss, and thus are consistent with and are upper bounds of the unified loss.
  - Consequently, it also upper bounds (1-NDCG).

Listwise Approach

- Direct optimization
  - Many direct optimization methods try to optimize smooth surrogates of IR measures, and this is why they are called direct optimization.
  - Question, are they really direct?
  - Directness

$$D(w,\tilde{M},M)=\frac{1}{\sup_{\mathbf{x},\Omega_y}|M(w,\mathbf{x},\Omega_y)-\tilde{M}(w,\mathbf{x},\Omega_y)|}$$

Listwise Approach

(Y. He, T.-Y. Liu, et al. 2008)

- Direct optimization
  - For SoftRank, it has been proved that the surrogate measure can be as direct as possible to NDCG, by setting σ_s as small as possible.
  - For SVM-MAP, there always exists some instances in the input /output space that make the surrogate measure not direct to AP on them.
- Good or bad?

Discussions

- The tools to prove the bounds are not yet "unified".
  - What about the pointwise approach?
- Quantitative analysis between different bounds is missing.
- Upper bounds may not mean too much.
- Directness vs. Consistency.

Statistical Ranking Theory

Query-level Ranking Framework

Query-level Generalization Analysis

Conventional Learning Framework

- Mainly about the pairwise approach.
- Assume (X_u, Y_u) and (X_v, Y_v) are i.i.d. samples, define $y_{u,v} = 2 \cdot I_{\{y_u > y_v\}} - 1$, then the expected and empirical risks are defined as below.

$$R(f) = \int_{(\mathcal{X}\times\mathcal{Y})^2} L(f, x_u, x_v, y_{u,v})\, P_{XY}(dx_u, dy_u)\, P_{XY}(dx_v, dy_v)$$

$$\hat{R}(f) = \frac{1}{C_m^2} \sum_{u=1}^{m} \sum_{v=u+1}^{m} L(f, x_u, x_v, y_{u,v})$$

$$C_m^2 = m(m-1)/2$$

U-statistics

Generalization in Learning to Rank

- The goal of Learning to Rank: to minimize the expected query-level risk $R(f)$.
- However, as the distribution is unknown, one minimizes the empirical query-level risk $\hat{R}(f)$.
- The generalization in ranking is concerned with the bound of the difference between the expected and empirical risks.
- Intuitively, the generalization ability in ranking can be understood as whether the ranking model learned from the training set can have a good performance on new test queries.

Conventional Generalization Analysis

- VC dimension
- Rank shatter coefficient
- Stability theory

Complexity of the function class

Robustness of the algorithm

Limitations of Previous Works

- 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.

Query-level Ranking Framework

- Assume queries as i.i.d. random variables
- Query-level risks

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

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

Query-level Ranking Framework

- The pointwise approach
  - Further assume documents as i.i.d. random variables.

$$L(f;q) = \int_{\mathcal{X}\times\mathcal{Y}} L(f;x,y)\, \mathcal{D}_q^{(1)}(dx,dy)$$

$$\hat{L}(f;q) = \frac{1}{m}\sum_{j=1}^{m} L(f;x_j,y_j)$$

Query-level Ranking Framework

- The pairwise approach
  - The U-statistics view: further assume document as i.i.d. random variables.

$$L(f;q) = \int_{(\mathcal{X}\times\mathcal{Y})^2} L(f;x_1,x_2,y_1,y_2)\,\mathcal{D}_q^{(2)}(dx_1,dx_2,dy_1,dy_2)$$

$$\hat{L}(f;q) = \frac{1}{C_m^2}\sum_{u=1}^{m}\sum_{v=u+1}^{m} L(f;x_u,x_v,y_u,y_v)$$

  - The Average view: further assume document pairs as i.i.d. random variables.

$$L(f;q) = \int_{\mathcal{X}\times\mathcal{X}\times\mathcal{Y}} L(f;x_1,x_2,y)\,\mathcal{D}_q^{(3)}(dx_1,dx_2,dy)$$

$$\hat{L}(f;q) = \frac{1}{\tilde{m}}\sum_{j=1}^{\tilde{m}} L(f;x_{j_1},x_{j_2},y_j)$$

Statistical Ranking Theory: Query-level Ranking Framework

Query-level Ranking Framework

- The listwise approach
  - Listwise loss minimization

$$L(f;q) = l(f;\mathbf{x},\pi_y)$$

  - Direct optimization of IR measure

$$\hat{L}(f;q) = l(f;\mathbf{x},\mathbf{y})$$

Case Studies: The Pairwise Approach

- 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.

$$\left| l\left(\boxed{f_{\{(q_i,S^{(i)})\}_{i=1}^n}}, x_1, x_2, y\right) - l\left(\boxed{f_{\{(q_i,S^{(i)})\}_{i=1,i\neq u}^n}}, x_1, x_2, y\right) \right| \le \boxed{\tau(n)}$$

Function learned from the original training data.

Function learned from the training data that eliminate the j-th "samples" (both the query and the associates) from the original training data.

Stability coefficient

Statistical Ranking Theory: Generalization for the Pairwise Approach

Case Studies: The Pairwise Approach

- Generalization Bound based on Query-Level Stability

$$R_L\left(f_{\{(q_i,S^{(i)})\}_{i=1}^n}\right) \le \widehat{R_L}\left(f_{\{(q_i,S^{(i)})\}_{i=1}^n}\right) + \boxed{2\tau(n) + (4n\tau(n)+B)\sqrt{\frac{\ln\frac{1}{\delta}}{2n}}}$$

  - The bound is related to the number of training queries *n*, and the stability coefficient $\tau(n)$.
  - If $\tau(n)$ →0 very fast as n→∞, then the bound will tend to zero, i.e. the generalization ability is good.

Stability of Ranking SVM and IRSVM

- Ranking SVM

$$\boxed{\tau(n) \le \frac{4\kappa^2}{\lambda n} \boxed{\frac{1+\frac{\sigma}{\mu\sqrt{\frac{\delta}{n}}}}{1-\frac{\varepsilon}{\mu}}}} \quad O\left(\frac{1}{\sqrt{n}}\right)$$

≥ 1

$\geq$

- IRSVM

$$\boxed{\tau(n) = \frac{4\kappa^2}{\lambda n}} \quad O\left(\frac{1}{n}\right)$$

The generalization bound for Ranking SVM is much looser than that for IRSVM

Case Studies: The Listwise Approach

- Rademacher Average based generalization theory.

Theorem Let $\mathcal{A}$ denote a listwise ranking algorithm, and let $L_{\mathcal{A}}(f;\mathbf{x},\pi_y)$ be its listwise loss, given the training data $S = \{(\mathbf{x}^{(i)},\pi_y^{(i)}), i=1,\cdots,n\}$, if $\forall f \in \mathcal{F}, (\mathbf{x},\pi_y) \in \mathcal{X}^m \times \mathcal{Y}, L_{\mathcal{A}}(f;\mathbf{x},\pi_y) \in [0,1]$, then with probability at least $1-\delta$, the following inequality holds:

$$\sup_{f\in\mathcal{F}}(R_{L_{\mathcal{A}}}(f) - \widehat{R}_{L_{\mathcal{A}}}(f;S)) \le 2C_{\mathcal{A}}(\varphi)N(\varphi)\widehat{\mathcal{R}}(\mathcal{F}) + \sqrt{\frac{2\ln\frac{2}{\delta}}{n}},$$

where $\widehat{\mathcal{R}}(\mathcal{F})$ is the Rademacher average of the scoring function class (for the linear scoring function, we have $\widehat{\mathcal{R}}(\mathcal{F}) \le \frac{2BM}{\sqrt{n}}$); $N(\varphi) = \sup_{x\in[-BM,BM]}\varphi'(x)$ measures the smoothness of the transformation function φ; $C_{\mathcal{A}}(\varphi)$ is an algorithm-dependent factor.

Case Studies: The Listwise Approach

- Detailed analysis for ListNet and ListMLE.

φ	$N(\varphi)$	$C_{ListMLE}(\varphi)$	$C_{ListNet}(\varphi)$
φ_L			$\frac{b+aBM}{b-aBM})$
φ_E			
φ_S			$)$

Convergence rate:
- Both algorithms: $O(n^{-1/2})$

Generalization ability:
- ListMLE > ListNet, especially when m is large
- Linear transformation function has the best generalization ability.

Discussion

- Generalization is not the entire story.
  - Generalization is defined with respect to a surrogate loss function *L*.
  - Consistency further discusses whether the risk defined by *L* can converge to the risk defined by the true loss for ranking, at the large sample limit.
- What is the true loss for ranking?
  - Permutation level 0-1?
  - IR measures?

Benchmarking Learning to Rank Methods

Role of a Benchmark Dataset

- Enable researchers to focus on technologies instead of experimental preparation
- Enable fair comparison among algorithms

- Reuters and RCV-1 for text classification
- UCI for general machine learning

The LETOR Collection

- A benchmark dataset for learning to rank
  - Document Corpora
  - Document Sampling
  - Feature Extraction
  - Meta Information
  - Dataset Finalization

Document Corpora

- The ".gov" corpus
  - TD2003, TD2004
  - HP 2003, HP 2004
  - NP 2003, NP 2004
- The OHSUMED corpus

Document Sampling

- The ".gov" corpus
  - Top 1000 documents per query returned by BM25
- The OHSUMED corpus
  - All judged documents

Feature Extraction

- The ".gov" corpus
  - 64 features
  - TF, IDF, BM25, LMIR, PageRank, HITS, Relevance Propagation, URL length, etc.
- The OHSUMED corpus
  - 40 features
  - TF, IDF, BM25, LMIR, etc.

Meta Information

- Statistical information about the corpus, such as the number of documents, the number of streams, and the number of (unique) terms in each stream.
- Raw information of the documents associated with each query, such as the term frequency, and the document length.
- Relational information, such as the hyperlink graph, the sitemap information, and the similarity relationship matrix of the corpora.

Finalizing Datasets

- Five-fold cross validation

Folds	Training set	Validation set	Test set
Fold1	{S1,S2,S3}	S4	S5
Fold2	{S2,S3,S4}	S5	S1
Fold3	{S3,S4,S5}	S1	S2
Fold4	{S4,S5,S1}	S2	S3
Fold5	{S5,S1,S2}	S3	S4

http://research.microsoft.com/~LETOR/

Algorithms under Investigation

- The pointwise approach
  - Linear regression
- The pairwise approach
  - Ranking SVM, RankBoost, Frank
- The listwise approach
  - ListNet
  - SVM-MAP, AdaRank

Linear ranking model (except RankBoost)

Experimental Results

Table 7.5 Results on the TD2003 dataset

Algorithm	N@1	N@3	N@10	P@1	P@3	P@10	MAP
Regression	0.320	0.307	0.326	0.320	0.260	0.178	0.241
RankSVM	0.320	0.344	0.346	0.320	0.293	0.188	0.263
RankBoost	0.280	0.325	0.312	0.280	0.280	0.170	0.227
FRank	0.300	0.267	0.269	0.300	0.233	0.152	0.203
ListNet	0.400	0.337	0.348	0.400	0.293	0.200	0.275
AdaRank	0.260	0.307	0.306	0.260	0.260	0.158	0.228
SVM^{map}	0.320	0.320	0.328	0.320	0.253	0.170	0.245

Experimental Results

Table 7.6 Results on the TD2004 dataset

Algorithm	N@1	N@3	N@10	P@1	P@3	P@10	MAP
Regression	0.360	0.335	0.303	0.360	0.333	0.249	0.208
RankSVM	0.413	0.347	0.307	0.413	0.347	0.252	0.224
RankBoost	0.507	0.430	0.350	0.507	0.427	0.275	0.261
FRank	0.493	0.388	0.333	0.493	0.378	0.262	0.239
ListNet	0.360	0.357	0.317	0.360	0.360	0.256	0.223
AdaRank	0.413	0.376	0.328	0.413	0.369	0.249	0.219
SVM^{map}	0.293	0.304	0.291	0.293	0.302	0.247	0.205

Experimental Results

Table 7.7 Results on the NP2003 dataset

Algorithm	N@1	N@3	N@10	P@1	P@3	P@10	MAP
Regression	0.447	0.614	0.665	0.447	0.220	0.081	0.564
RankSVM	0.580	0.765	0.800	0.580	0.271	0.092	0.696
RankBoost	0.600	0.764	0.807	0.600	0.269	0.094	0.707
FRank	0.540	0.726	0.776	0.540	0.253	0.090	0.664
ListNet	0.567	0.758	0.801	0.567	0.267	0.092	0.690
AdaRank	0.580	0.729	0.764	0.580	0.251	0.086	0.678
SVM^{map}	0.560	0.767	0.798	0.560	0.269	0.089	0.687

Experimental Results

Table 7.8 Results on the NP2004 dataset

Algorithm	N@1	N@3	N@10	P@1	P@3	P@10	MAP
Regression	0.373	0.555	0.653	0.373	0.200	0.082	0.514
RankSVM	0.507	0.750	0.806	0.507	0.262	0.093	0.659
RankBoost	0.427	0.627	0.691	0.427	0.231	0.088	0.564
FRank	0.480	0.643	0.729	0.480	0.236	0.093	0.601
ListNet	0.533	0.759	0.812	0.533	0.267	0.094	0.672
AdaRank	0.480	0.698	0.749	0.480	0.244	0.088	0.622
SVM^{map}	0.520	0.749	0.808	0.520	0.267	0.096	0.662

Experimental Results

Table 7.9 Results on the HP2003 dataset

Algorithm	N@1	N@3	N@10	P@1	P@3	P@10	MAP
Regression	0.420	0.510	0.594	0.420	0.211	0.088	0.497
RankSVM	0.693	0.775	0.807	0.693	0.309	0.104	0.741
RankBoost	0.667	0.792	0.817	0.667	0.311	0.105	0.733
FRank	0.653	0.743	0.797	0.653	0.289	0.106	0.710
ListNet	0.720	0.813	0.837	0.720	0.320	0.106	0.766
AdaRank	0.733	0.805	0.838	0.733	0.309	0.106	0.771
SVM^{map}	0.713	0.779	0.799	0.713	0.309	0.100	0.742

Experimental Results

Table 7.10 Results on the HP2004 dataset

Algorithm	N@1	N@3	N@10	P@1	P@3	P@10	MAP
Regression	0.387	0.575	0.646	0.387	0.213	0.08	0.526
RankSVM	0.573	0.715	0.768	0.573	0.267	0.096	0.668
RankBoost	0.507	0.699	0.743	0.507	0.253	0.092	0.625
FRank	0.600	0.729	0.761	0.600	0.262	0.089	0.682
ListNet	0.600	0.721	0.784	0.600	0.271	0.098	0.690
AdaRank	0.613	0.816	0.832	0.613	0.298	0.094	0.722
SVM^{map}	0.627	0.754	0.806	0.627	0.280	0.096	0.718

Experimental Results

Table 7.11 Results on the OHSUMED dataset

Algorithm	N@1	N@3	N@10	P@1	P@3	P@10	MAP
Regression	0.446	0.443	0.411	0.597	0.577	0.466	0.422
RankSVM	0.496	0.421	0.414	0.597	0.543	0.486	0.433
RankBoost	0.463	0.456	0.430	0.558	0.561	0.497	0.441
FRank	0.530	0.481	0.443	0.643	0.593	0.501	0.444
ListNet	0.533	0.473	0.441	0.652	0.602	0.497	0.446
AdaRank	0.539	0.468	0.442	0.634	0.590	0.497	0.449
SVM^{map}	0.523	0.466	0.432	0.643	0.580	0.491	0.445

Winner Numbers

$$S_i(M) = \sum_{j=1}^{7} \sum_{k=1}^{7} I_{\{M_i(j) > M_k(j)\}}$$

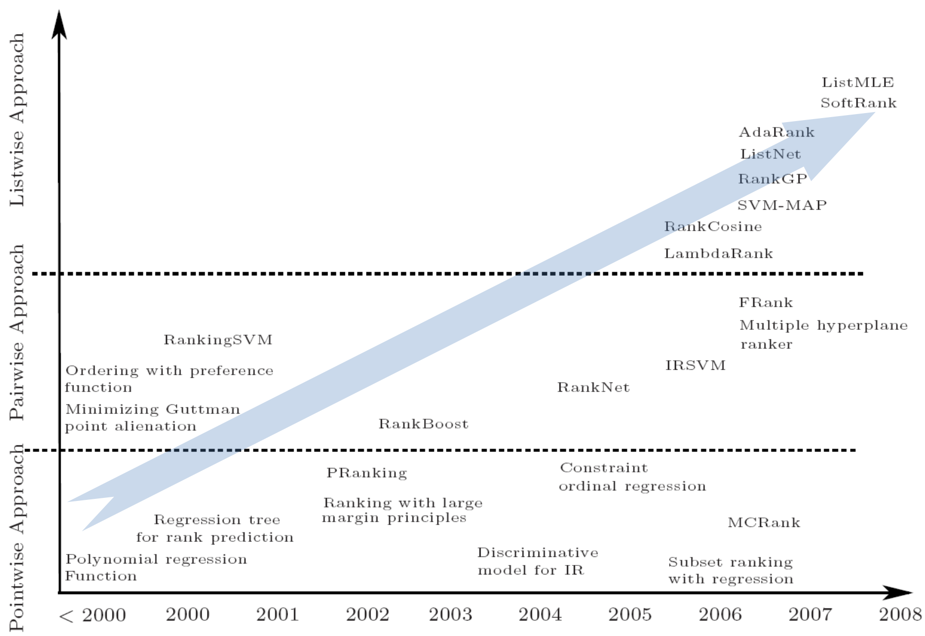
Table 7.12 Winner Number of Each Algorithm

Algorithm	N@1	N@3	N@10	P@1	P@3	P@10	MAP
Regression	4	4	4	5	5	5	4
RankSVM	21	22	22	21	22	22	24
RankBoost	18	22	22	17	22	23	19
FRank	18	19	18	18	17	23	15
ListNet	29	31	33	30	32	35	33
AdaRank	26	25	26	23	22	16	27
SVM^{map}	23	24	22	25	20	17	25

Discussions

- NDCG@1
  - {ListNet, AdaRank} > {SVM-MAP, RankSVM} > {RankBoost, FRank} > {Regression}
- NDCG@3, @10
  - {ListNet} > {AdaRank, SVM-MAP, RankSVM, RankBoost} > {FRank} > {Regression}
- MAP
  - {ListNet} > {AdaRank, SVM-MAP, RankSVM} > {RankBoost, FRank} > {Regression}

Summary



Listwise Approach
Pairwise Approach
Pointwise Approach
ListMLE
SoftRank
AdaRank
ListNet
RankGP
SVM-MAP
RankCosine
LambdaRank
FRank
Multiple hyperplane ranker
RankingSVM
Ordering with preference function
IRSVM
RankNet
Minimizing Guttman point alienation
RankBoost
PRanking
Constraint ordinal regression
Ranking with large margin principles
Regression tree for rank prediction
MCRank
Discriminative model for IR
Polynomial regression Function
Subset ranking with regression
< 2000
2000
2001
2002
2003
2004
2005
2006
2007
2008

Other Works

- Ground truth mining
  - targets automatically mining ground truth labels for learning to rank, mainly from click-through logs of search engines.
- Feature engineering
  - includes feature selection, dimensionality reduction, and effective feature learning.
- Query-dependent ranking
  - adopts different ranking models for different types of queries, based on either hard query type classification or soft nearest neighbor approach.

Other Works

- Supervised rank aggregation
  - learns the ranking model not to combine features, but to aggregate candidate ranked lists.
- Semi-supervised / active ranking
  - leverages the large number of unlabeled queries and documents to improve the performance of ranking model learning.
- Relational / global ranking
  - does not only make use of the scoring function for ranking, but considers the inter-relationship between documents to define more complex ranking models.

Reference papers http://research.microsoft.com/~letor/paper.html

Answers to the Question 1

- To what respect are these learning to rank algorithms similar and in which aspects do they differ? What are the strengths and weaknesses of each algorithm?

Approach	Modeling	Pro	Con
Pointwise	Regression/ classification/ ordinal regression	Easy to leverage existing theories and algorithms	• Accurate ranking ≠ Accurate score or category • Position info is invisible to the loss
Pairwise	Pairwise classification		
Listwise	Ranking	query-level and position based	• More complex • New theory needed

Answers to the Question 2

- What are the unique theoretical issues for ranking that should be investigated?
  - Unique properties of ranking for IR: evaluation is performed at query level and is position based.
  - The risks should also be defined at the query level, and a query-level theoretical framework is needed for conducting analyses on the learning to rank methods.
  - The “true loss” for ranking should consider the position information in the ranking result, but not as simple as the 0−1 loss in classification.

Answers to the Question 3

- Empirically speaking, which of those many learning to rank algorithms perform the best?
  - LETOR makes it possible to perform fair comparisons among different learning to rank methods.
  - Empirical studies on LETOR have shown that the listwise ranking algorithms seem to have certain advantages over other algorithms, especially for top positions of the ranking result, and the pairwise ranking algorithms seem to outperform the pointwise algorithms.

Answers to the Question 4

- Are there many remaining issues regarding learning to rank to study in the future?
  - Learning from logs
  - Feature engineering
  - Advanced ranking model
  - Ranking theory

Learning from Logs

- Data scale matters a lot!
  - Ground truth mining from logs is one of the possible ways to construct large-scale training data.
- Ground truth mining may lose information .
  - User sessions, frequency of clicking a certain document, frequency of a certain click pattern, and diversity in the intentions of different users.
- Directly learn from logs.
  - E.g., Regard click-through logs as the "ground truth", and define its likelihood as the loss function.

Feature Engineering

- Ranking is deeply rooted in IR
  - Not just new algorithms and theories.
- Feature engineering is very importance
  - How to encode the knowledge on IR accumulated in the past half a century in the features?
  - Currently, these kinds of works have not been given enough attention to.

Advanced Ranking Model

- Listwise ranking function
  - Natural but challenging
  - Complexity is a big issue
- Generative ranking model
  - Why not generative?
- Semi-supervised ranking
  - The difference from semi-supervised classification

Ranking Theory

- A more unified view on loss functions
- True loss for ranking
- Statistical consistency for ranking
- Complexity of ranking function class
- ……

Opportunities and Challenges

- Learning to rank is HOT, but…

Learning to rank

☹

Hey, too much math!
Too theoretical!
We want intuition and insights!

Application is too narrow!
Ranking is not our central problem!

Information Retrieval

Machine Learning

Opportunities and Challenges

- Learning to rank is HOT, but…

Learning to rank

☺

Hey, machine learning can help us build the best ranking model!

Good example of applying machine learning to solve real problems!
Ranking should be our next big thing!

Information Retrieval

Machine Learning

Thanks!

tyliu@microsoft.com
http://research.microsoft.com/people/tyliu/