Listwise Learning to Rank

Research Preparation Exam for

Harsh Pareek (hhp266)

harshp@cs.utexas.edu

The University of Texas at Austin
College of Natural Sciences

UT CS
COMPUTER SCIENCE

Outline

- Learning to Rank
  - Probability Ranking Principle
  - Datasets and Evaluation Metrics
- Loss Functions for Learning to Rank
  - Pointwise, Pairwise and Listwise loss functions
  - LambdaMART
- Listwise Ranking Functions[Pareek and Ravikumar 2014]
  - Need for Listwise Ranking Functions
  - A Representation Theorem

Ranking

- Central problem in Information Retrieval
  - Applies to many problems: document retrieval, recommender systems (collaborative filtering), product rating etc.
- Ranking Problems:
  - Rank the documents purely according to their relevance with regards to the query.
  - Consider relations between documents such as similarity, website structure, and diversity
  - Aggregate several candidate ranked lists to get a better ranked list. Called Rank Aggregation or meta search

Learning to Rank (LETOR)

- Machine Learning-based approach to Ranking
- **Probability Ranking Principle:** For each item to be ranked, predict a score, often a probability or an expected value. e.g.,
  - P(user will click this link)
  - P(user will purchase this item)
  - E[Revenue earned by showing this item]
- Learn weights on features to get a **scoring function**
- Use a loss function to minimize some appropriate **Evaluation Metric**

Yahoo! LETOR dataset

- Queries were randomly sampled from Yahoo! Search query logs
- For each query, documents collected using various base IR systems
- Document relevance scored by expert judges on a scale of 0 (bad) to 4 (perfect)
- Document features:
  - Web Graph features: PageRank, number of inlinks, outlinks
  - Document statistics: Word count features
  - Document classifier: Topic, quality, type of page, spam, adult
  - Query-document compatibility: Text match features
  - Click features: Probability of click/skip, dwell time
  - Page Freshness

Datasets for Learning to Rank

	Queries	Doc.	Rel.	Feat.	Year
LETOR 3.0 – Gov	575	568 k	2	64	2008
LETOR 3.0 – Ohsumed	106	16 k	3	45	2008
LETOR 4.0	2,476	85 k	3	46	2009
Yandex	20,267	213 k	5	245	2009
Yahoo!	36,251	883 k	5	700	2010
Microsoft	31,531	3,771 k	5	136	2010

Chapelle et al (2010)

Evaluation Metrics for LETOR

- Used to the compare the output of a ranking algorithm to ground truth data
- Represents a user model
  - Penalize errors in later positions more
    - E.g. An error in the 101st position should be penalized less than an error in the 3rd position
  - Account for Diversity
    - Penalize algorithms that place too many similar items at the top of the ranking

Evaluation Metrics: MAP

Mean Average Precision

$P@k(q) =$ #{relevant documents in top-k positions} / k

Average Precision

$$AP(q) = \sum_{k=1}^{m} \frac{P@k(q).\, l_k}{\#\{relevant\ docs\}}$$

Evaluation Metrics: NDCG

- Normalized Discounted Cumulative Gain

$$DCG@k = \sum_{i=1}^{k} \frac{2^{l_i} - 1}{\log_2(1+i)}$$

$$NDCG@k = \frac{DCG}{Ideal\ DCG}$$

Evaluation Metrics: ERR

- Inspired by a Cascade User Model

$$ERR = \sum_{i=1}^{n} \frac{1}{i} P(user\ stops\ at\ i)$$

$$= \sum_{\{i=1\}}^{n} \frac{1}{i} R(y_i) \Pi_{j=1}^{i-1} (1 - R(y_j))$$

  - Where $R(y) = \frac{2^y - 1}{16}$

Evaluation Metrics: Diversity

- Extrinsic Diversity:
  - Hedge against uncertainty about the user's information need
  - E.g. A web search query for "Jaguar" could return the car, the animal, the sports team etc.
- Intrinsic Diversity:
  - Avoid redundancy
  - E.g. In the Facebook News Feed, show only one news article about the Syrian Refugee Crisis

Loss functions for Learning to Rank

- Loss functions are be categorized as:
  - Pointwise: Estimate relevance for each individual document. Similar to regression.
  - Pairwise: Ignore individual document relevances. Instead, relative rankings should be correct
  - Listwise: Optimize IR metrics directly. Can take into account position and diversity

Pointwise loss function

- Cast the problem as a regression, classification or Ordinal Regression problem, e.g.
  - Subset ranking using Regression [Cossock and Zhang, 2006] uses the square loss

$$L(f; x_j, y_j) = (y_j - f(x_j))^2.$$

  - McRank[Li et al, 2007]: Adapts multiclass classification techniques for ranking

Pairwise Loss Functions

For each pair of documents, x_u and x_v, minimize the loss,

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

To predict a ranking at test time, perform Rank Aggregation on the partial rankings for each pair.

$$\max_{\pi} \sum_{u<v} h(x_{\pi^{-1}(u)}, x_{\pi^{-1}(v)}).$$

Pairwise Loss Functions

SVMRank: Using $f(x) = w^T x$,

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

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

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

Listwise Losses: AdaRank

For $M(f, x, y)$, an IR measure of interest

Algorithm 2 Learning Algorithms for AdaRank

Input: document group for each query

Given: initial distribution $\mathcal{D}_1$ on input queries

For $t = 1, \ldots, T$

Train weak ranker $f_t(\cdot)$ based on distribution $\mathcal{D}_t$.

Choose $\alpha_t = \frac{1}{2} \log \frac{\sum_{i=1}^{n} \mathcal{D}_t(i)(1+M(f_t, \mathbf{x}^{(i)}, \mathbf{y}^{(i)}))}{\sum_{i=1}^{n} \mathcal{D}_t(i)(1-M(f_t, \mathbf{x}^{(i)}, \mathbf{y}^{(i)}))}$

Update $\mathcal{D}_{t+1}(i) = \frac{\exp(-M(\sum_{s=1}^{t} \alpha_s f_s, \mathbf{x}^{(i)}, \mathbf{y}^{(i)}))}{\sum_{j=1}^{n} \exp(-M(\sum_{s=1}^{t} \alpha_s f_s, \mathbf{x}^{(j)}, \mathbf{y}^{(j)}))}$,

Output: $\sum_t \alpha_t f_t(\cdot)$.

Listwise Losses

- ListNet: Based on the Luce Model for Rankings
  - Minimizes the K-L divergence between the labeling predicted by the algorithm and the permutation implied by relevance labels
- [Ravikumar et al 2011] provide an exhaustive characterization of listwise surrogates for NDCG
  - ListNet is a special case for the Cross-entropy Loss

Listwise Losses

SVM^{MAP} : Optimizes the MAP directly

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

$$\text{s.t.} \quad \forall \mathbf{y}^{c(i)} \neq \mathbf{y}^{(i)},$$

$$w^T\Psi(\mathbf{y}^{(i)},\mathbf{x}^{(i)}) \geq w^T\Psi(\mathbf{y}^{c(i)},\mathbf{x}^{(i)}) + 1 - \mathrm{AP}(\mathbf{y}^{c(i)}) - \xi^{(i)}.$$

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

$$\Psi(\mathbf{y}^c,\mathbf{x}) = \sum_{u,v:y_u=1,y_v=0}(y_u^c - y_v^c)(x_u - x_v).$$

Yahoo! LETOR Results

- The challenge was dominated by methods based on Gradient Boosted Decision Trees
  - LambdaMART won the competition, though many other methods were as competitive
- Pointwise, Pairwise and Listwise methods performed almost equally well
  - With large amounts of data, loss functions are less important than the ranking/scoring functions used

LambdaMART [Burges, 2010]

Listwise loss trained using Gradient Boosted Decision Trees

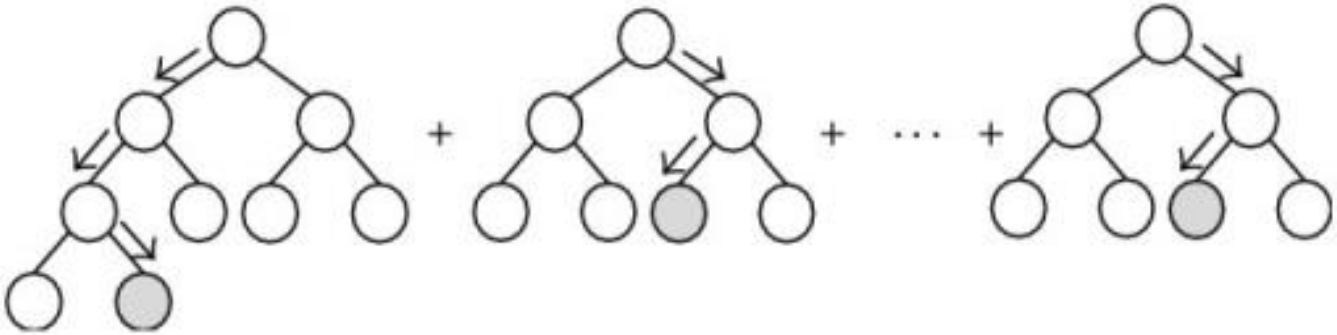

Image credit:
https://openi.nlm.nih.gov/detailedresult.php?img=PMC4538588_CIN2015-149702.002&req=4

LambdaMART

Algorithm: LambdaMART

set number of trees N, number of training samples m, number of leaves per tree L, learning rate η

for $i = 0$ to m **do**

  $F_0(x_i) = \text{BaseModel}(x_i)$ //If BaseModel is empty, set $F_0(x_i) = 0$

end for

for $k = 1$ to N **do**

  for $i = 0$ to m **do**

    $y_i = \lambda_i$

    $w_i = \frac{\partial y_i}{\partial F_{k-1}(x_i)}$

  end for

  $\{R_{lk}\}_{l=1}^{L}$ // Create L leaf tree on $\{x_i, y_i\}_{i=1}^{m}$

  $\gamma_{lk} = \frac{\sum_{x_i \in R_{lk}} y_i}{\sum_{x_i \in R_{lk}} w_i}$ // Assign leaf values based on Newton step.

  $F_k(x_i) = F_{k-1}(x_i) + \eta \sum_l \gamma_{lk} I(x_i \in R_{lk})$ // Take step with learning rate η.

end for

$$\lambda_{ij} \equiv S_{ij} \left| \Delta\text{NDCG} \frac{\partial C_{ij}}{\partial o_{ij}} \right|,$$

$$\lambda_i = \sum_{j \in P} \lambda_{ij}.$$

Our contributions

LISTWISE RANKING FUNCTIONS

A Gap in the Literature

- Loss functions vs Ranking functions
  - Most work in the literature uses pointwise scoring/ranking functions with pointwise, pairwise or listwise loss functions
- This is insufficient for listwise metrics such as diversity
  - And is insufficient for NDCG/ERR as well!

Listwise Ranking Functions

- Pointwise Ranking Function
  - Scores each document individually at test time
  - Sort scores to obtain final ranking
- Pairwise Ranking Function
  - Each pairs of documents is scored. Pairs are aggregated to obtain a ranking
- Listwise Ranking Function
  - The score for a document depends on the other documents in the retrieved set

Rethinking the Probability Ranking Principle

- What if $P(click)$ depends on the order of documents?
  - $P(click_1, click_2) \neq P(click_2, click_1)$
- What if $P(click)$ for a document depends on the documents that have come before it
  - $P(click_1, click_2 \neq P(click_1)P(click_2)$

Listwise Ranking Functions

- Exchangeable LRF: For some g (indep of i),

$$f(\boldsymbol{x})_i = g\left(\boldsymbol{x}_i, \{\boldsymbol{x}_{\backslash \mathrm{i}}\}\right)$$

  – g is symmetric in $\{\boldsymbol{x}_{\backslash \mathrm{i}}\}$
- Every pointwise ranking functions is a listwise ranking function, but the converse is not true

A Representation Theory for Listwise Ranking Functions

- An exchangeable ranking function can be decomposed, for some g_t, as:

$$\left(f(\boldsymbol{x})\right)_i = h(\boldsymbol{x}_i, \{\boldsymbol{x}_{\backslash \mathrm{i}}\}) = \sum_{t=1}^{\infty} \Pi_{j \neq i} \; g_t(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

- Thus, pairwise g_t can be combined to obtain a listwise ranking function

Proofs for Representation Theorem

- Proofs of the representation theorem:
  - Tensor decomposition: Extension of Eigenvector factorization for a matrix
  - Functional Analysis: Similar to Eigenfunction factorization for a linear operator
  - De Finetti's theorem: There exists θ such that

$$p(x_1, x_2, x_3, \dots) = \int \Pi_i p(x_i|\theta) d\theta$$

Listwise Ranking Functions in the Literature

Listwise ranking functions have been previously proposed in the literature by directly minimizing listwise losses.

- Qin et al. [2008] propose the following functions, obtained by optimizing continuous CRFs in closed form:

$$f(X) = (\alpha^T eI + \beta(D - S))^{-1} X\alpha$$

$$f(X) = \frac{1}{\alpha^T e}(2X\alpha + \beta(D_r - D_c)e)$$

- Weston and Blitzer [2012] propose optimizing the following listwise loss directly at inference time (Latent Structured Ranking):

$$f(q) = arg \max_d \sum_{i=1}^{m} w_i q^T U^T V \boldsymbol{d}_i + \sum_{i,j=1}^{m} w_i w_j \boldsymbol{d}_i^T S^T S \boldsymbol{d}_j$$

Using the Representation Theory

- Consider for some base ranker b and similarity kernels S_k:

$$f_i(\boldsymbol{x}) = b(\boldsymbol{x}_i)\Pi_{j\neq i} g\left(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{w}\right)$$

$$= b(\boldsymbol{x}_i)\Pi_{j\neq i} \exp \sum_k w_k S_k(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

  – With preprocessing, computational complexity at training time equivalent to that for pointwise ranking functions.

Experimental results[Pareek and Ravikumar 2014]

Table 1: Results for our reranking procedure across LETOR 3.0 datasets. For each dataset, the first column is the base ranker, second column is the loss function used for reranking.

	OHSUMED		TD2003		NP2003	
	Base RankBoost	Reranked w/ Cross Ent	Base CA	Reranked w/ q-Norm	Base MART	Reranked w/ Square
ndcg@1	0.5104	**0.5421**	**0.3500**	0.3250	0.5467	**0.5600**
ndcg@2	0.4798	**0.4901**	0.2875	**0.3375**	0.6500	**0.6567**
ndcg@5	0.4547	**0.4615**	0.3228	**0.3461**	0.7112	**0.7128**
ndcg@10	0.4356	**0.4445**	0.3210	**0.3385**	0.7326	**0.7344**

	HP2003		HP2004		NP2004	
	Base MART	Reranked w/ Cross Ent	Base RankBoost	Reranked w/ q-Norm	Base MART	Reranked w/ Square
ndcg@1	0.6667	**0.7333**	0.5200	**0.5333**	0.3600	**0.3733**
ndcg@2	0.7667	0.7667	0.6067	**0.6533**	0.4733	**0.4867**
ndcg@5	0.7546	**0.7618**	0.7034	**0.7042**	0.5603	**0.5719**
ndcg@10	0.7740	**0.7747**	0.7387	**0.7420**	0.5951	**0.6102**

Challenges in using Listwise Ranking Functions

- Test time evaluation cost:
  - LRFs may require $O(m^2)$ computations at test time
  - Compared to greedy approaches for diversification, this algorithm may be more efficient and more parallelizable
- Difficulty of collecting Training Data:
  - Accounting for Presentation Bias
  - Off-policy evaluation leads to biased training data

SUMMARY

Learning to Rank

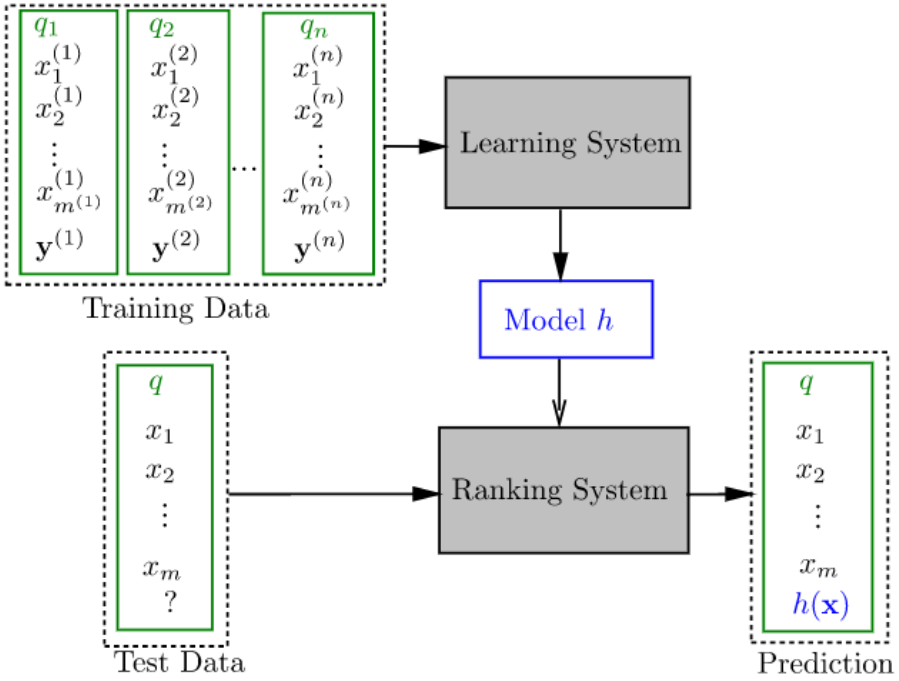

Fig. 1.1 Learning-to-rank framework.

Credit: Liu et al (2009)

Summary

- Considerations in Learning to Rank
  - Evaluation Metric
  - Loss Function
  - Ranking Function
- Listwise Ranking Functions
  - A Representation Theorem

Thank you!

Questions?

References

- The material in this presentation is inspired by
  - Liu et al. 2009
  - Chapelle et al. 2011