Learning To Rank:

— From RankNet to LambdaRank to LambdaMart

Fuka at 2016.09.23

Outline

- Why Learning to Rank(LTR)
- Simple Introduce to LTR
- RankNet
- LambdaRank
- LambdaMart
- Summary

Why Learning to Rank

1. A large number of linear models In Search Engine

    1) Speed is king

    2) Intuitively

    3) Easily achievable

2. Manual parameter tuning is usually difficult and lead to overfitting

    1) Machine Learning

Evaluation Measure

1. WTA(Winner Takes All)

   Zero:Top document for a query is relevant,otherwise one

2. Pair-wise Error

   Counts the number of pairs that are in the errer order

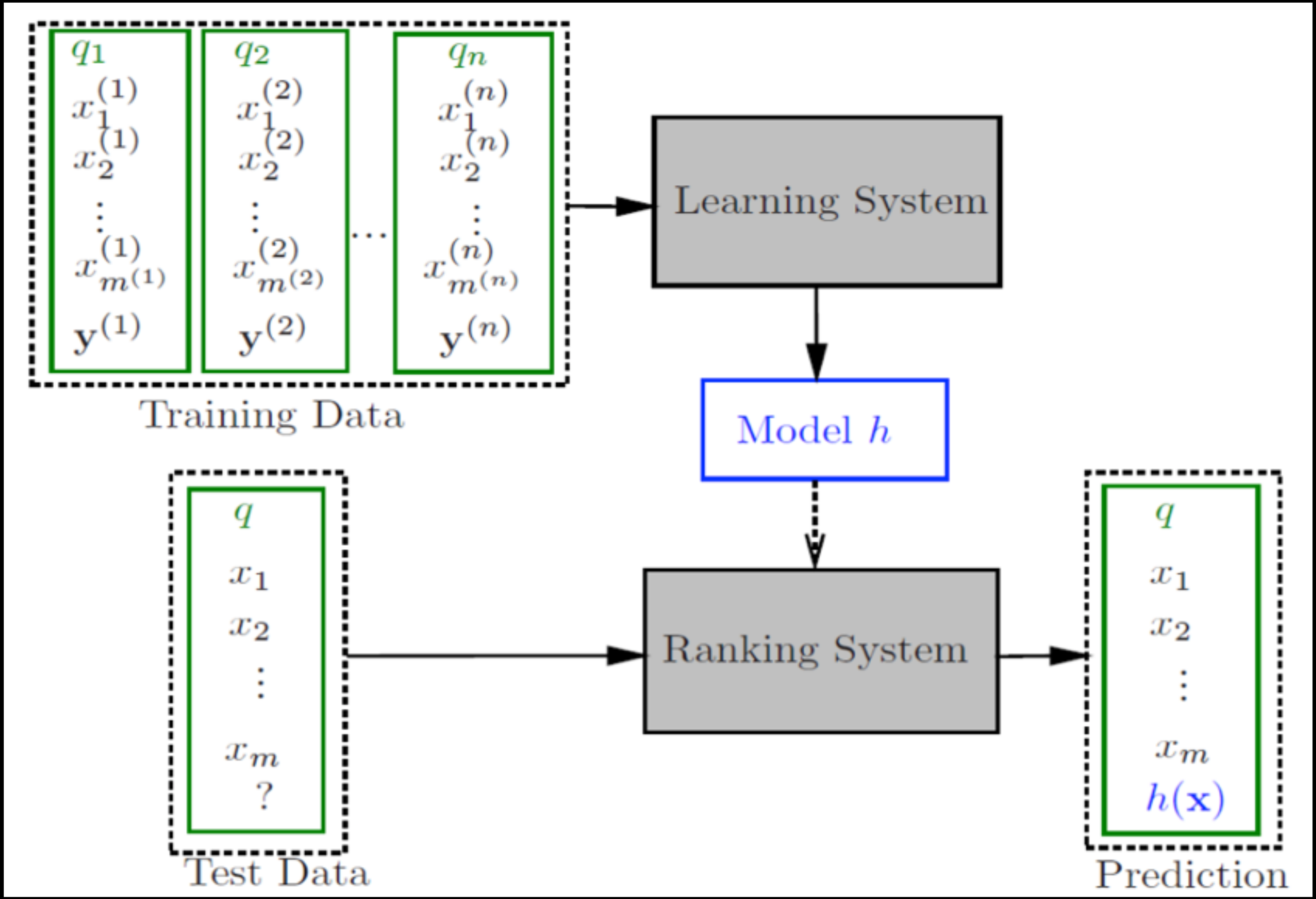
3. MAP(Mean Average Precision)

$$AP = \frac{\sum_{i:relvance}^{n} P_i / R_i}{n}$$

4. NDCG(Normalized Discounted Cumulative Gain)

$$NDCG = \frac{1}{IDCG} \cdot \sum_{i=1}^{n} \frac{2^{y_i} - 1}{log(1+i)}$$

Learning to Rank Framework



Pointwise Method

Table 2.7: Characteristics of Pointwise Approach

	Learning	Ranking
Pointwise Approach (Classification)		
Input	feature vector x	feature vectors $\mathbf{x} = \{x_i\}_{i=1}^n$
Output	category $y = \text{classifier}(f(x))$	ranking list $\text{sort}(\{f(x_i)\}_{i=1}^n)$
Model	$\text{classifier}(f(x))$	ranking model $f(x)$
Loss	classification loss	ranking loss
Pointwise Approach (Regression)		
Input	feature vector x	feature vectors $\mathbf{x} = \{x_i\}_{i=1}^n$
Output	real number $y = f(x)$	ranking list $\text{sort}(\{f(x_i)\}_{i=1}^n)$
Model	regression model $f(x)$	ranking model $f(x)$
Loss	regression loss	ranking loss
Pointwise Approach (Ordinal Classification)		
Input	feature vector x	feature vectors $\mathbf{x} = \{x_i\}_{i=1}^n$
Output	ordered category $y = \text{threshold}(f(x))$	ranking list $\text{sort}(\{f(x_i)\}_{i=1}^n)$
Model	$\text{threshold}(f(x))$	ranking model $f(x)$
Loss	ordinal classification loss	ranking loss

Advantage

1) Speedly

2) Low complexity

Shortcoming

1) General performance

2) Ignore document pairs

3) Ignore the relevance of document and query.

Pairwise Method

Table 2.8: Characteristics of Pairwise Approach

Pairwise Approach (Classification)		
	Learning	**Ranking**
Input	feature vectors $x^{(1)}, x^{(2)}$	feature vectors $\mathbf{x} = \{x_i\}_{i=1}^n$
Output	pairwise classification $\text{classifier}(f(x^{(1)}) - f(x^{(2)}))$	ranking list $\text{sort}(\{f(x_i)\}_{i=1}^n)$
Model	$\text{classifier}(f(x))$	ranking model $f(x)$
Loss	pairwise classification loss	ranking loss

Pairwise Approach (Regression)		
	Learning	**Ranking**
Input	feature vectors $x^{(1)}, x^{(2)}$	feature vectors $\mathbf{x} = \{x_i\}_{i=1}^n$
Output	pairwise regression $f(x^{(1)}) - f(x^{(2)})$	ranking list $\text{sort}(\{f(x_i)\}_{i=1}^n)$
Model	regression model $f(x)$	ranking model $f(x)$
Loss	pairwise regression loss	ranking loss

Advantage

1) care document pairs
2) easily to got the pairs

Shortcoming

1) Only order but not position
2) Vastly different to pairs number between querys

Listwise Method

Training as Query

1.Measure specific:usch as $L(F(x), y)) = exp(-NDCG)$

2.Non-measure specific:The loss function not relate to evaluate measure

Advantage

1)Generally outperformance then other method

Shortcoming

2)Two complex

RankNet:ICML2005

Target Probability

$$\overline{P}_{i,j} = P(D_i \triangleright D_j)$$

Modeled Probability

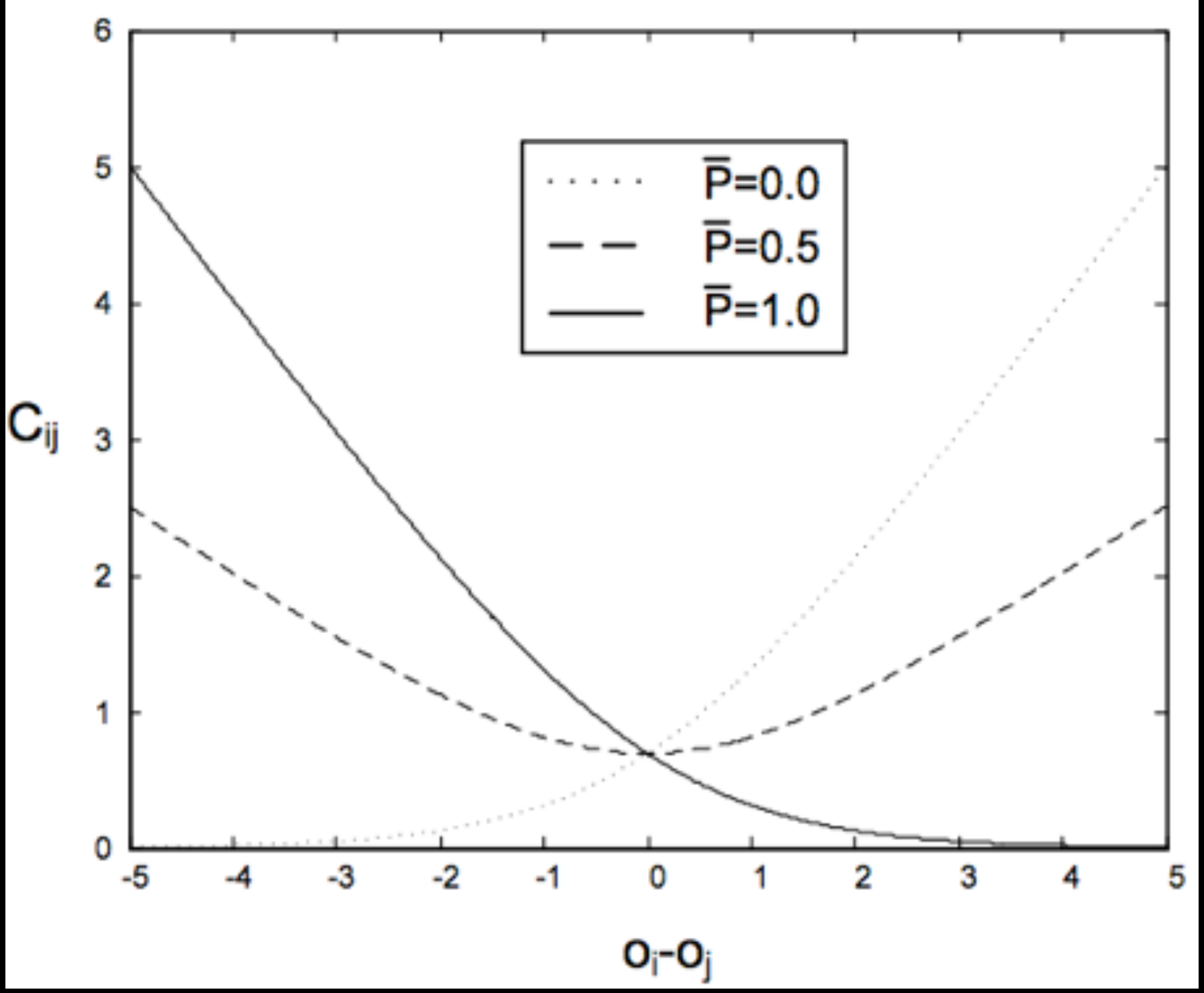
$$P_{i,j} \equiv \frac{e^{o_{i,j}}}{1+e^{o_{i,j}}} \equiv \frac{1}{1+e^{-o_{i,j}}} \qquad o_{i,j} = f(x_i) - f(x_j)$$

Cross entropy as the loss function

$$C_{i,j} = -\bar{P}_{i,j} log P_{i,j} - (1 - \bar{P}_{i,j}) log(1 - P_{i,j})$$

$C_{i,j}$ can write as:

$$C_{i,j} = -\bar{P}_{i,j} o_{i,j} + log(1 + e^{o_{i,j}})$$



So when $\overline{P}_{i,j} = 1$ Cross Entropy will change to:

$$C_{i,j} = log(1 + e^{-o_{i,j}})$$

Partial derivation of the loss function with O

$$\frac{\partial C}{\partial o_i} = \left(-\bar{P}_{i,j} + \frac{e^{o_i - o_j}}{1 + e^{o_i - o_j}}\right) = \left(-\bar{P}_{i,j} + P_{i,j}\right) = -\frac{\partial C}{\partial o_j}$$

Use gradient descent to optimize the Cross Entropy loss

$$w_k \rightarrow w_k - \eta \frac{\partial C}{\partial w_k} = w_k - \eta \left(\frac{\partial C}{\partial o_i}\frac{\partial o_i}{\partial w_k} + \frac{\partial C}{\partial o_j}\frac{\partial o_j}{\partial w_k}\right)$$

η is learning rate,general is 1e-3,1e-5

Now,we can get the change of C

$$\Delta C = \sum_k \frac{\partial C}{\partial w_k}\Delta w_k = \sum_k \frac{\partial C}{\partial w_k}\left(-\eta \frac{\partial C}{\partial w_k}\right) = -\eta \sum_k \left(\frac{\partial C}{\partial w_k}\right)^2 < 0$$

Neural Network is the cal score model for RankNet

$$o = f(x : w, b) = f^{(2)}\left(\sum_l w_l^{(2)} \cdot f^{(1)}\left(\sum_k w_{lk}^{(1)} x_k + b^{(1)}\right) + b^{(2)}\right)$$

Forword progagation for new parameters(document), and backward propagation re-calculation of the parameters(pairs)

Speed up RankNet Training

Given a training pair $\{(x_i, x_j), \bar{P}_{i,j}\}$,will have:

$$\begin{aligned}\frac{\partial C}{\partial w_k} &= \frac{\partial C}{\partial o_i}\frac{\partial o_i}{\partial w_k} + \frac{\partial C}{\partial o_j}\frac{\partial o_j}{\partial w_k} \\ &= \left(-\bar{P}_{i,j} + \frac{e^{o_i - o_j}}{1 + e^{o_i - o_j}}\right)\left(\frac{\partial o_i}{\partial w_k} - \frac{\partial o_j}{\partial w_k}\right) \\ &= \lambda_{i,j}\left(\frac{\partial o_i}{\partial w_k} - \frac{\partial o_j}{\partial w_k}\right)\end{aligned}$$

Now note $\lambda_{i,j} = \frac{\partial C}{\partial o_i} = -\frac{\partial C}{\partial o_j} = \left(-\bar{P}_{i,j} + \frac{e^{o_i - o_j}}{1 + e^{o_i - o_j}}\right)$

We can update the parameters by all document in query instead of pairs:

$$\Delta w_k = -\eta \sum_{i,j \in I} \left(\lambda_{i,j} \frac{\partial o_i}{\partial w_k} - \lambda_{i,j} \frac{\partial o_j}{\partial w_k} \right) = -\eta \sum_i \lambda_i \frac{\partial o_i}{\partial w_k}$$

$\{i, j\}$ satisfy for $D_i \triangleright D_j$

λ_i is:

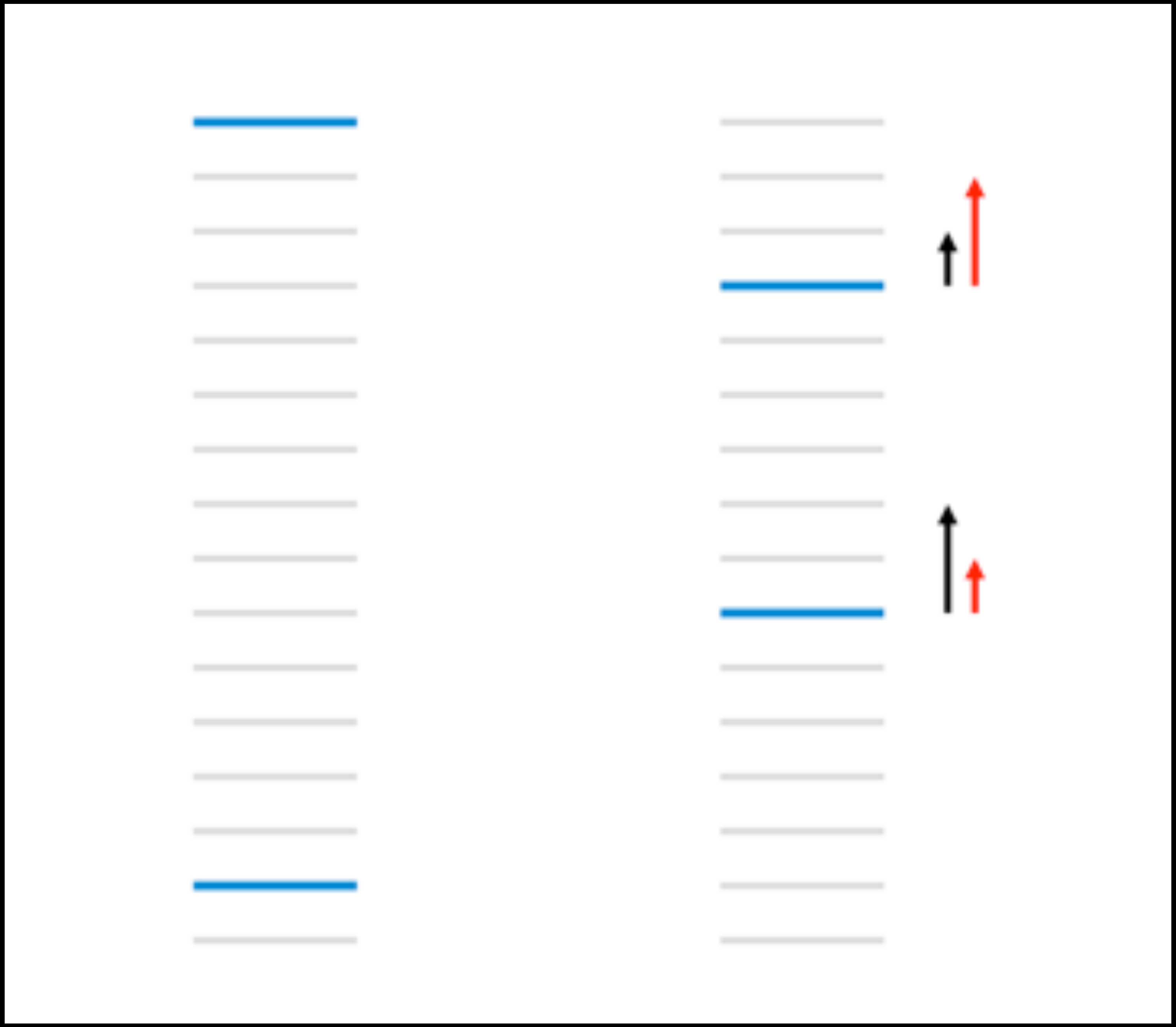
find all: j for which $\{i, j\} \in I$ and k for which $\{k, i\} \in I$

increment λ_i by $\lambda_{i,j}$ and decrement λ_i by $\lambda_{k,i}$

So

$$\lambda_i = \sum_{j:i,j \in I} \lambda_{i,j} - \sum_{j:j,i \in I} \lambda_{i,j}$$

LambdaRank-2006



How could we optmize the evaluate measure (such NDCG)

If $i << j$,then instead of pairwise error,we would prefer and optimization cost C that has property that

$$|\frac{\partial C}{\partial o_i}| >> |\frac{\partial C}{\partial o_j}|$$

The gradient function is defined as:

$$\frac{\partial C}{\partial o_i} = -\lambda_i(o_1, l_1 ... o_n, l_n)$$

The gradient is dependent on query,and the LambdaRank is a ListWise method.

Back for RankNet:

$$\lambda_{i,j} = \frac{\partial C}{\partial o_i} = -\frac{\partial C}{\partial o_j} = -\frac{1}{1 + e^{o_i - o_j}}$$

LambdaRank add evaluate measure to \lambda

$$\lambda_{i,j} = -\frac{1}{1 + e^{o_i - o_j}} |\Delta Z|$$

And

$$\lambda_i = \sum_{j:i,j \in I} \lambda_{i,j} - \sum_{j:j,i \in I} \lambda_{i,j}$$

LambdaMart

Using Mart instead of Neural Networks

$$\gamma_{km} = \frac{\sum_{x_i \in R_{km}} \frac{\partial C}{\partial s_i}}{\sum_{x_i \in R_{km}} \frac{\partial^2 C}{\partial s_i^2}} = \frac{-\sum_{x_i \in R_{km}} \sum_{\{i,j\} \rightleftharpoons I} |\Delta Z_{ij}| \rho_{ij}}{\sum_{x_i \in R_{km}} \sum_{\{i,j\} \rightleftharpoons I} |\Delta Z_{ij}| \sigma \rho_{ij} (1 - \rho_{ij})}$$

Summary

- Start by RankNet
- Change at LambdaRank
- Popular with LambdaMart