IntervalRank: Isotonic Regression with Listwise and Pairwise Constraints

Taesup Moon

(Joint work with Alex Smola, Yi Chang, and Zhaohui Zheng)

Yahoo! Labs

WSDM 2010

February 5

1. Backgrounds
2. IntervalRank
   - Loss function via isotonic regression
   - Efficient implementation
3. Experimental results

1. Backgrounds

2. IntervalRank
   - Loss function via isotonic regression
   - Efficient implementation

3. Experimental results

Learning to rank in web search

A supervised machine learning framework for ranking

- relevance labeled data

$$\{(\mathbf{x}_i, y_i)\}_{i=1}^n$$

- a *loss function*

$$L(\{(y_i, f(\mathbf{x}_i))\}_{i=1}^n)$$

- train a ranking function f via *optimizing* the loss
  - e.g., functional gradient descent

Various loss functions have been proposed

pointwise loss functions

- treat each example individually
- e.g., regression, etc.

Various loss functions have been proposed

pointwise loss functions

- treat each example individually
- e.g., regression, etc.

pairwise loss functions

- focus on relative orderings of pairs
- e.g., GBRank, RankNet, RankBoost, etc.

Various loss functions have been proposed

pointwise loss functions

- treat each example individually
- e.g., regression, etc.

pairwise loss functions

- focus on relative orderings of pairs
- e.g., GBRank, RankNet, RankBoost, etc.

listwise loss functions

- treat the whole list jointly
- e.g., LambdaRank, SoftRank, SmoothDCG, etc.

Does one approach dominate others?

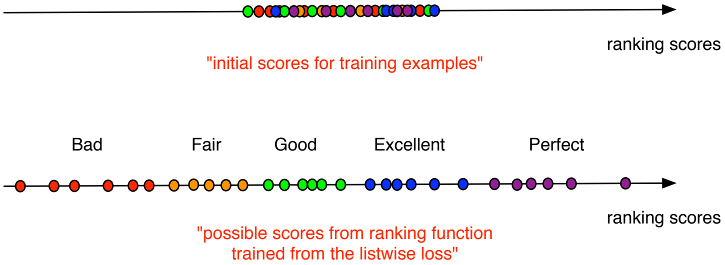
Common wisdom: pointwise $\preceq$ pairwise $\preceq$ listwise

- But, listwise loss functions also have some caveats

Does one approach dominate others?

Common wisdom: pointwise $\preceq$ pairwise $\preceq$ listwise

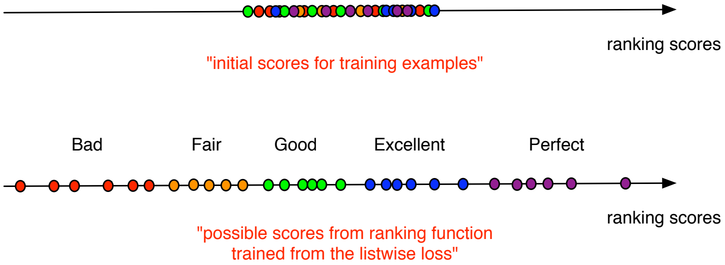
- But, listwise loss functions also have some caveats



Does one approach dominate others?

Common wisdom: pointwise $\preceq$ pairwise $\preceq$ listwise

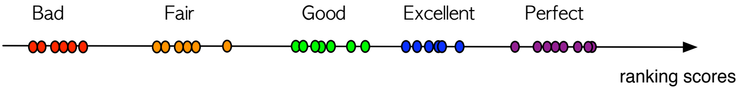
- But, listwise loss functions also have some caveats



"may not assign similar scores to similarly relevant documents"

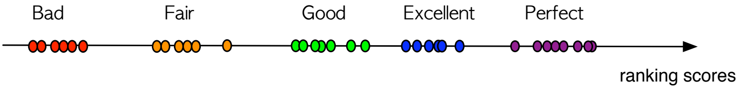
Can we mix the different approaches?

Ideally, we would like to train a function to



Can we mix the different approaches?

Ideally, we would like to train a function to



- *separate* documents with different relevance
- *cluster* documents with similar relevance

1 Backgrounds

2 IntervalRank
- Loss function via isotonic regression
- Efficient implementation

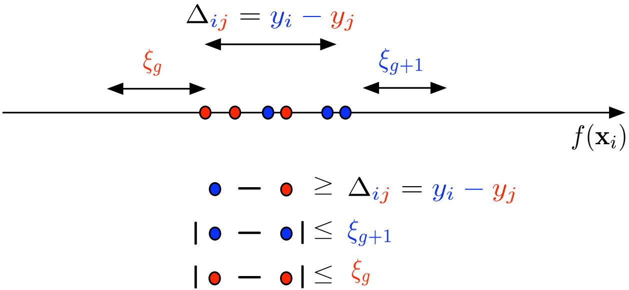
3 Experimental results

Our approach

1. Define a loss function via *isotonic regression*
2. Reformulate the problem to *efficiently* find the gradient

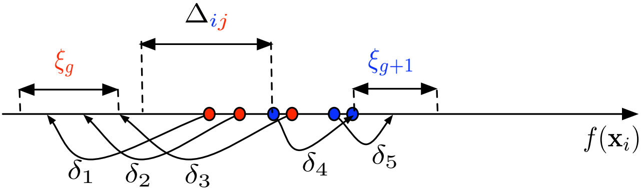
Loss function via isotonic regression

Loss = *minimum total efforts* to make scores satisfy the constraints



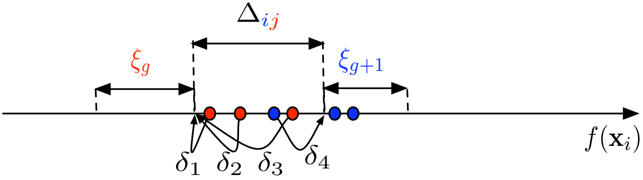
Loss function via isotonic regression

total effort $= \delta_1^2 + \delta_2^2 + \delta_3^2 + \delta_4^2 + \delta_5^2$



Loss function via isotonic regression

total effort $= \delta_1^2 + \delta_2^2 + \delta_3^2 + \delta_4^2$



Loss function via isotonic regression

Loss = *"minimum"* total effort

$$L\Big(\{(y_i, f(\mathbf{x}_i))\}_{i=1}^n\Big) = \min_{\delta\in\mathbb{R}^n} \|\delta\|_2^2, \qquad \text{where } \delta \in \mathbb{R}^n \text{ satisfies}$$

$$f(\mathbf{x}_i) + \delta_i - f(\mathbf{x}_j) - \delta_j \geq \Delta_{ij} \text{ for all } (i,j) \in \{\text{ordered pairs}\}$$

$$|f(\mathbf{x}_i) + \delta_i - f(\mathbf{x}_j) - \delta_j| \leq \xi_{g_i} \text{ for all } (i,j) \in \{\text{tied pairs}\}$$

Loss function via isotonic regression

Loss $=$ *"minimum"* total effort

$$L\big(\{(y_i, f(\mathbf{x}_i))\}_{i=1}^n\big) = \min_{\delta \in \mathbb{R}^n} \|\delta\|_2^2, \qquad \text{where } \delta \in \mathbb{R}^n \text{ satisfies}$$

$$f(\mathbf{x}_i) + \delta_i - f(\mathbf{x}_j) - \delta_j \geq \Delta_{ij} \text{ for all } (i,j) \in \{\text{ordered pairs}\}$$
$$|f(\mathbf{x}_i) + \delta_i - f(\mathbf{x}_j) - \delta_j| \leq \xi_{g_i} \text{ for all } (i,j) \in \{\text{tied pairs}\}$$

- first proposed by [Zheng et.al 2008]

Loss function via isotonic regression

Loss = *"minimum"* total effort

$$L(\{(y_i, f(\mathbf{x}_i))\}_{i=1}^n) = \min_{\delta \in \mathbb{R}^n} \|\delta\|_2^2, \qquad \text{where } \delta \in \mathbb{R}^n \text{ satisfies}$$

$$f(\mathbf{x}_i) + \delta_i - f(\mathbf{x}_j) - \delta_j \geq \Delta_{ij} \text{ for all } (i,j) \in \{\text{ordered pairs}\}$$

$$|f(\mathbf{x}_i) + \delta_i - f(\mathbf{x}_j) - \delta_j| \leq \xi_{g_i} \text{ for all } (i,j) \in \{\text{tied pairs}\}$$

- first proposed by [Zheng et.al 2008]
  - pairwise constraints and listwise objective

Loss function via isotonic regression

Loss = *"minimum"* total effort

$$L(\{(y_i, f(\mathbf{x}_i))\}_{i=1}^n) = \min_{\delta \in \mathbb{R}^n} \|\delta\|_2^2, \qquad \text{where } \delta \in \mathbb{R}^n \text{ satisfies}$$

$$f(\mathbf{x}_i) + \delta_i - f(\mathbf{x}_j) - \delta_j \geq \Delta_{ij} \text{ for all } (i,j) \in \{\text{ordered pairs}\}$$
$$|f(\mathbf{x}_i) + \delta_i - f(\mathbf{x}_j) - \delta_j| \leq \xi_{g_i} \text{ for all } (i,j) \in \{\text{tied pairs}\}$$

- first proposed by [Zheng et.al 2008]
  - pairwise constraints and listwise objective
  - obtain the optimum δ^* and use it as a functional gradient for $f(\mathbf{x}_i)$

Loss function via isotonic regression

Loss = *"minimum"* total effort

$$L\big(\{(y_i, f(\mathbf{x}_i))\}_{i=1}^n\big) = \min_{\delta \in \mathbb{R}^n} \|\delta\|_2^2, \qquad \text{where } \delta \in \mathbb{R}^n \text{ satisfies}$$

$$f(\mathbf{x}_i) + \delta_i - f(\mathbf{x}_j) - \delta_j \geq \Delta_{ij} \text{ for all } (i,j) \in \{\text{ordered pairs}\}$$

$$|f(\mathbf{x}_i) + \delta_i - f(\mathbf{x}_j) - \delta_j| \leq \xi_{g_i} \text{ for all } (i,j) \in \{\text{tied pairs}\}$$

- first proposed by [Zheng et.al 2008]
  - pairwise constraints and listwise objective
  - obtain the optimum δ^* and use it as a functional gradient for $f(\mathbf{x}_i)$
- problems:

Loss function via isotonic regression

Loss = *"minimum"* total effort

$$L\left(\{(y_i, f(\mathbf{x}_i))\}_{i=1}^n\right) = \min_{\delta \in \mathbb{R}^n} \|\delta\|_2^2, \qquad \text{where } \delta \in \mathbb{R}^n \text{ satisfies}$$

$$f(\mathbf{x}_i) + \delta_i - f(\mathbf{x}_j) - \delta_j \geq \Delta_{ij} \text{ for all } (i,j) \in \{\text{ordered pairs}\}$$
$$|f(\mathbf{x}_i) + \delta_i - f(\mathbf{x}_j) - \delta_j| \leq \xi_{g_i} \text{ for all } (i,j) \in \{\text{tied pairs}\}$$

- first proposed by [Zheng et.al 2008]
  - pairwise constraints and listwise objective
  - obtain the optimum δ^* and use it as a functional gradient for $f(\mathbf{x}_i)$
- problems:
  - no formal proof for functional gradient

Loss function via isotonic regression

Loss = *"minimum"* total effort

$$L(\{(y_i, f(\mathbf{x}_i))\}_{i=1}^n) = \min_{\delta \in \mathbb{R}^n} \|\delta\|_2^2, \qquad \text{where } \delta \in \mathbb{R}^n \text{ satisfies}$$

$$f(\mathbf{x}_i) + \delta_i - f(\mathbf{x}_j) - \delta_j \geq \Delta_{ij} \text{ for all } (i,j) \in \{\text{ordered pairs}\}$$

$$|f(\mathbf{x}_i) + \delta_i - f(\mathbf{x}_j) - \delta_j| \leq \xi_{g_i} \text{ for all } (i,j) \in \{\text{tied pairs}\}$$

- first proposed by [Zheng et.al 2008]
  - pairwise constraints and listwise objective
  - obtain the optimum δ^* and use it as a functional gradient for $f(\mathbf{x}_i)$
- problems:
  - no formal proof for functional gradient
  - not practical - quadratic program (QP) with $O(n^3)$ complexity

The optimum δ^* is the functional gradient

We prove that

$$\delta_i^* = \frac{\partial L\big(\{(y_i, f(\mathbf{x}_i))\}_{i=1}^n\big)}{\partial f(\mathbf{x}_i)}$$

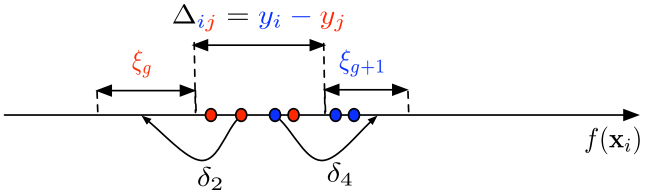
- Lemma 2 in the paper

We can reduce the number of variables

- original QP has n variables and $O(n^2)$ constraints

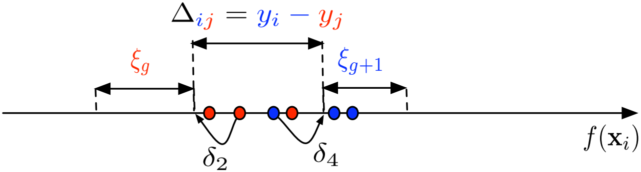
We can reduce the number of variables

- original QP has n variables and $O(n^2)$ constraints
- observation:
  1. δ satisfying constraints with equality is enough



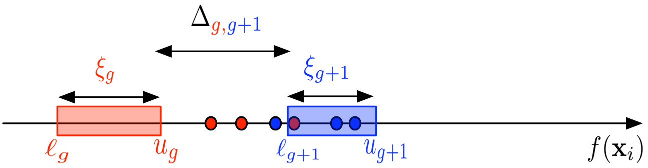
We can reduce the number of variables

- original QP has n variables and $O(n^2)$ constraints
- observation:
  1. δ satisfying constraints with equality is enough



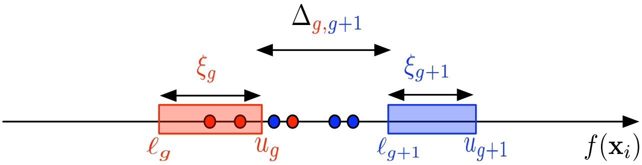
We can reduce the number of variables

- original QP has n variables and $O(n^2)$ constraints
- observation:
  1. δ satisfying constraints with equality is enough
  2. relevance grade interval $\{[\ell_g, u_g]\}$ can be found first, then obtain δ



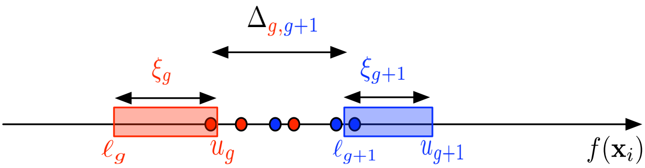
We can reduce the number of variables

- original QP has n variables and $O(n^2)$ constraints
- observation:
  1. δ satisfying constraints with equality is enough
  2. relevance grade interval $\{[\ell_g, u_g]\}$ can be found first, then obtain δ



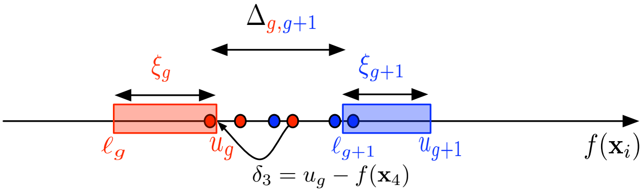
We can reduce the number of variables

- original QP has n variables and $O(n^2)$ constraints
- observation:
  1. δ satisfying constraints with equality is enough
  2. relevance grade interval $\{[\ell_g, u_g]\}$ can be found first, then obtain δ



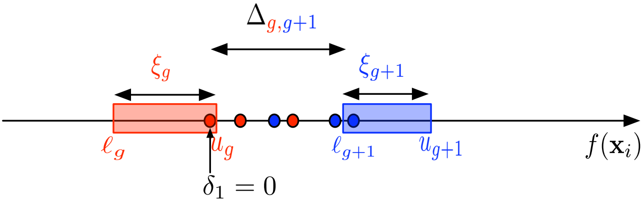
We can reduce the number of variables

- original QP has n variables and $O(n^2)$ constraints
- observation:
  1. δ satisfying constraints with equality is enough
  2. relevance grade interval $\{[\ell_g, u_g]\}$ can be found first, then obtain δ



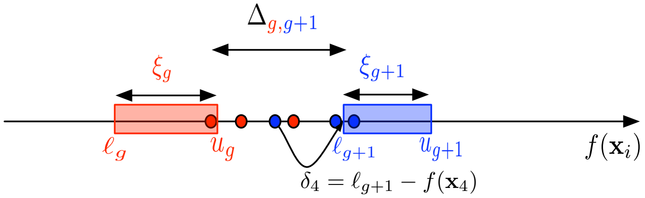
We can reduce the number of variables

- original QP has n variables and $O(n^2)$ constraints
- observation:
  1. δ satisfying constraints with equality is enough
  2. relevance grade interval $\{[\ell_g, u_g]\}$ can be found first, then obtain δ



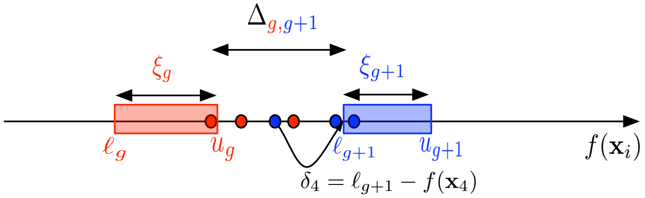
We can reduce the number of variables

- original QP has n variables and $O(n^2)$ constraints
- observation:
  1. δ satisfying constraints with equality is enough
  2. relevance grade interval $\{[\ell_g, u_g]\}$ can be found first, then obtain δ



We can reduce the number of variables

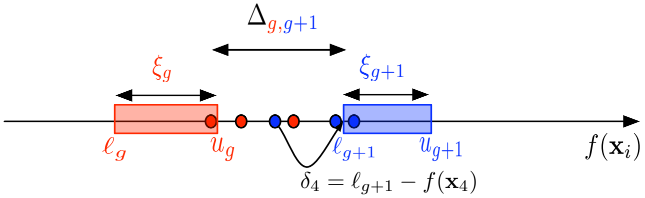
- original QP has n variables and $O(n^2)$ constraints
- observation:
  1. δ satisfying constraints with equality is enough
  2. relevance grade interval $\{[\ell_g, u_g]\}$ can be found first, then obtain δ



- finding *minimum efforts* δ^* can be obtained from the *optimum intervals* $\{[\ell_g^*, u_g^*]\}$ that lead to them

We can reduce the number of variables

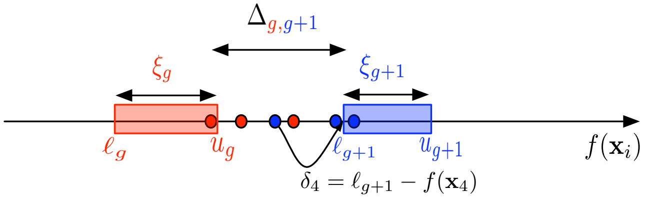
- original QP has n variables and $O(n^2)$ constraints
- observation:
  1. δ satisfying constraints with equality is enough
  2. relevance grade interval $\{[\ell_g, u_g]\}$ can be found first, then obtain δ



- finding *minimum efforts* δ^* can be obtained from the *optimum intervals* $\{[\ell_g^*, u_g^*]\}$ that lead to them
  - $\delta_i^* = \quad \min\{f(\mathbf{x}_i) - u_g^*, 0\}$

We can reduce the number of variables

- original QP has n variables and $O(n^2)$ constraints
- observation:
  1. δ satisfying constraints with equality is enough
  2. relevance grade interval $\{[\ell_g, u_g]\}$ can be found first, then obtain δ



- finding *minimum efforts* δ^* can be obtained from the *optimum intervals* $\{[\ell_g^*, u_g^*]\}$ that lead to them
  - $\delta_i^* = \max\{\ell_g^* - f(\mathbf{x}_i), \min\{f(\mathbf{x}_i) - u_g^*, 0\}\}$

Equivalent problem does not depend on n

Loss function

$$L(\{(y_i, f(\mathbf{x}_i))\}_{i=1}^n) = \min_{\{[\ell_g, u_g]\}} \sum_{g\in\mathcal{G}} \sum_{i\in S_g} \left[(\ell_g - f(\mathbf{x}_i))_+^2 + (f(\mathbf{x}_i) - u_g)_+^2\}\right],$$

where $\{[\ell_g, u_g]\}$ satisfy

$$\ell_g \leq u_g \leq \ell_g + \xi_g, \text{ for all } g \in \{\text{relevance grades}\}$$

$$\ell_{g+1} - u_g \geq \Delta_{g+1,g}, \text{ for all } g \in \{\text{relevance grades}\}$$

- problem reduced to $O(1)$ variables and $O(1)$ constraints
- no longer a QP, but we can still solve this *efficiently*

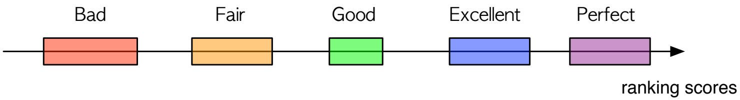
We can solve with $O(n \log n)$ complexity

Apply techniques from convex optimization

- log-barrier method
  - remove inequality constraints via log-barriers
- L-BFGS or conjugate gradient (CG) method
  - need to compute the objective and the gradient for each $\{\ell_g, u_g\}$
  - sorting of sums of $f(\mathbf{x}_i)$ and $f(\mathbf{x}_i)^2$ will do (details in the paper)

Summary of the algorithm

1. Find the intervals that lead to the "minimum efforts"



2. Regress on the intervals to find $\{\delta^*\}_{i=1}^n$ and do functional gradient descent
3. Also add (pointwise) regression loss

$$L(\{(y_i, f(\mathbf{x}_i))\}_{i=1}^n) = \frac{1}{2}\|\delta^*\|_2^2 + \frac{\lambda}{2}\sum_{i=1}^{n}(y_i - f(\mathbf{x}_i))^2$$

- gives absolute score information

1. Backgrounds

2. IntervalRank
   - Loss function via isotonic regression
   - Efficient implementation

3. Experimental results

LETOR 3.0 OHSUMED data

Data

- 106 queries, 16,140 query-document pairs
- 5-fold cross validation with $\frac{3}{5}$ for training, $\frac{1}{5}$ for validation, $\frac{1}{5}$ for test

Functional gradient boosting trees

- *slight variation*: added slack variables for the constraints
- parameters: 125 treees, 20 nodes per tree, shrinkage, $\lambda_1, \lambda_2, \lambda$

LETOR 3.0 OHSUMED data

- NDCG@k results

Algorithms	N@1	N@2	N@3	N@4	N@5
RankBoost	0.4632	0.4504	0.4555	0.4543	0.4494
RankSVM	0.4958	0.4331	0.4207	0.4240	0.4164
FRank	0.5300	0.5008	0.4812	0.4694	0.4588
ListNet	0.5326	0.4810	0.4732	0.4561	0.4432
AdaRank.MAP	0.5388	0.4789	0.4682	0.4721	0.4613
AdaRank.NDCG	0.5330	0.4922	0.4790	0.4688	0.4673
IntervalRank	**0.5628**	**0.5448**	**0.4900**	0.4703	0.4609

LETOR 3.0 OHSUMED data

- Precision@k results

Algorithms	P@1	P@2	P@3	P@4	P@5	MAP
RankBoost	0.5576	0.5481	0.5609	0.5580	0.5447	0.4411
RankSVM	0.5974	0.5494	0.5427	0.5443	0.5319	0.4334
FRank	0.6429	0.6195	0.5925	0.5840	0.5638	0.4439
ListNet	0.6524	0.6093	0.6016	0.5745	0.5502	0.4457
AdaRank.MAP	0.6338	0.5959	0.5895	0.5887	0.5674	0.4487
AdaRank.NDCG	0.6719	0.6236	0.5984	0.5838	0.5767	0.4498
IntervalRank	**0.6892**	**0.6522**	0.5768	0.5556	0.5488	0.4466

Commercial search engine data

Data

- training set : 8,180 queries, 341,300 query-document pairs
- test set: 916 queries, 32,008 query-document pairs
- 5 grade relevance judgments:
  $\mathcal{G} = \{$Perfect, Excellent, Good, Fair, Bad$\}$

Functional gradient boosting trees

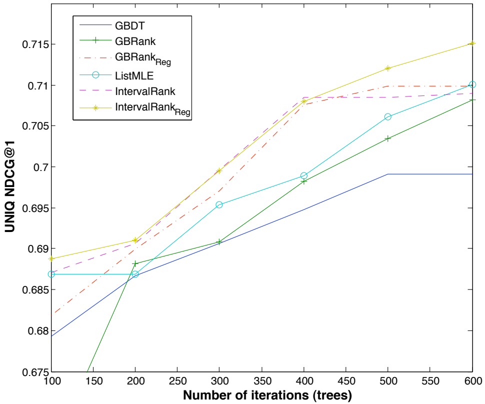
- *slight variation*: added slack variables for the constraints
- parameters: 600 treees, 20 nodes per tree, shrinkage, $\lambda_1, \lambda_2, \lambda$

Comparing schemes

- GBDT (pointwise), GBRank (pairwise), ListMLE (listwise)
- with or without additional regression term

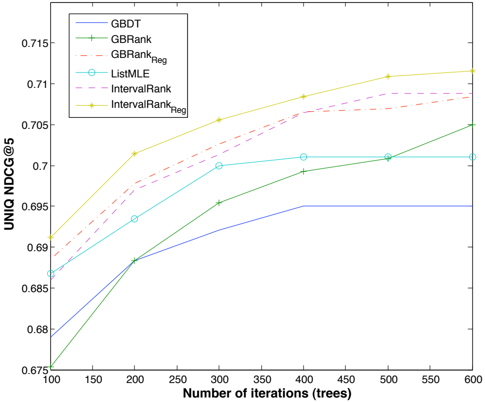
Commercial search engine data

- running time of IntervalRank was in the same range with others
- NDCG@1



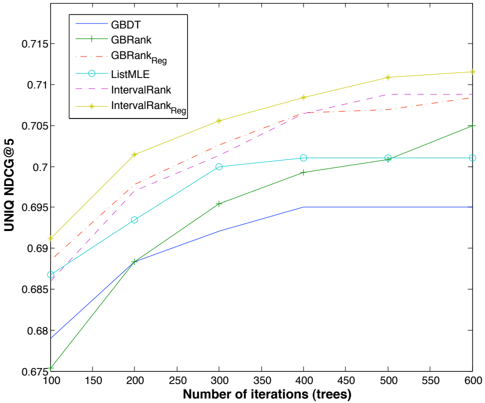
Commercial search engine data

- running time of IntervalRank was in the same range with others
- NDCG@5



Commercial search engine data

- running time of IntervalRank was in the same range with others
- NDCG@5



we gain up to 1% over other methods!