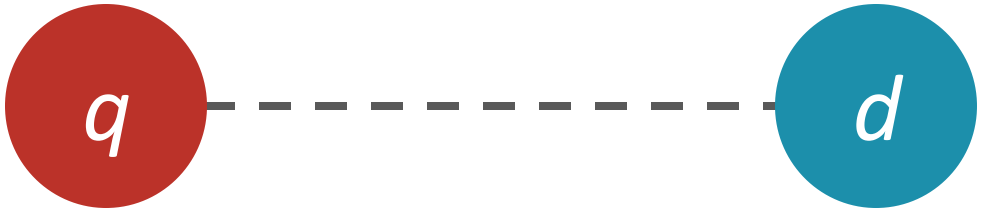
UFMG UNIVERSIDADE FEDERAL DE MINAS GERAIS

Information Retrieval

Learning to Rank: Pairwise and Listwise

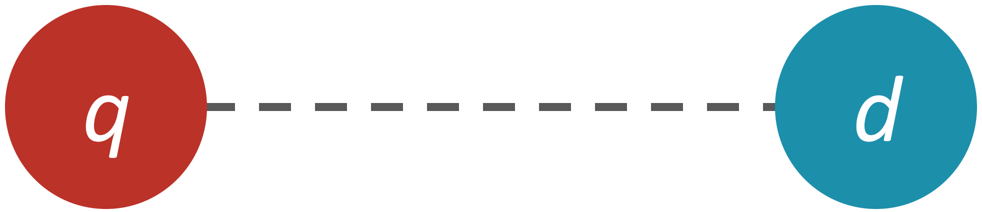
Rodrygo L. T. Santos
rodrygo@dcc.ufmg.br

The ranking problem



$$f(q, d)$$

Learning to rank



$$\boldsymbol{f}(\mathbf{x})$$

Learning to rank

Feature-based representation

- Individual models as ranking ”features”

Discriminative learning

- Effective models learned from data
- Aka machine-learned ranking

Pointwise approach

Several approaches

- Regression-based
- Classification-based
- Ordinal regression-based

$$\{(x_1, y_1), (x_2, y_2), (x_3, y_3), \ldots, (x_n, y_n)\}$$

Limitations of the pointwise approach

Ranking requires getting relative scores right

- Pointwise approaches learn absolute scores

Higher positions should matter more than lower ones

- Pointwise loss functions are agnostic to positions

Queries should be equally important

- Queries with many relevant documents dominate

Pairwise approach

Pairwise classification-based

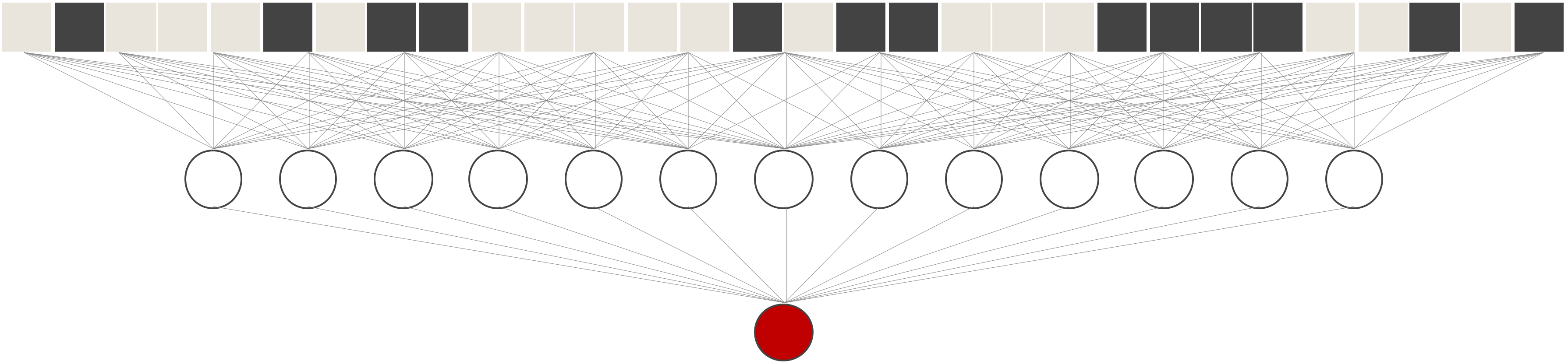
- RankNet
- RankBoost
- Ranking SVM
- IR-SVM

$$\{(x_1, x_2, 1), (x_2, x_1, 0),$$
$$(x_1, x_3, 1), (x_3, x_1, 0),$$
$$\dots$$
$$(x_{n-1}, x_n, 1), (x_n, x_{n-1}, 0)\}$$

RankNet
(Burges et al., ICML 2005)

Shallow (2-layer) neural network model

- Sigmoid activations



Gradient descent optimizer

RankNet
(Burges et al., ICML 2005)

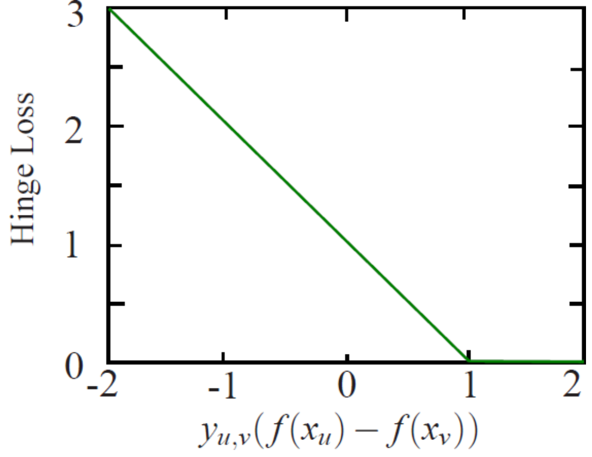
Pairwise prediction converted to probability

- $\hat{y}_{uv} = f(x_u, x_v) = \frac{\exp(f(x_u) - f(x_v))}{1 + \exp(f(x_u) - f(x_v))}$ ***(logistic function)***

Cross entropy loss

- $$\mathcal{L}(f; x_u, x_v, y_{uv}) = -\overbrace{y_{uv}}^{\text{relevant}} \log \overbrace{\hat{y}_{uv}}^{\text{predicted relevant}} - \underbrace{(1 - y_{uv})}_{\text{non-relevant}} \log \underbrace{(1 - \hat{y}_{uv})}_{\text{predicted non-relevant}}$$

Ranking SVM
(Herbrich et al., ALMC 2000; Joachims, KDD 2002)



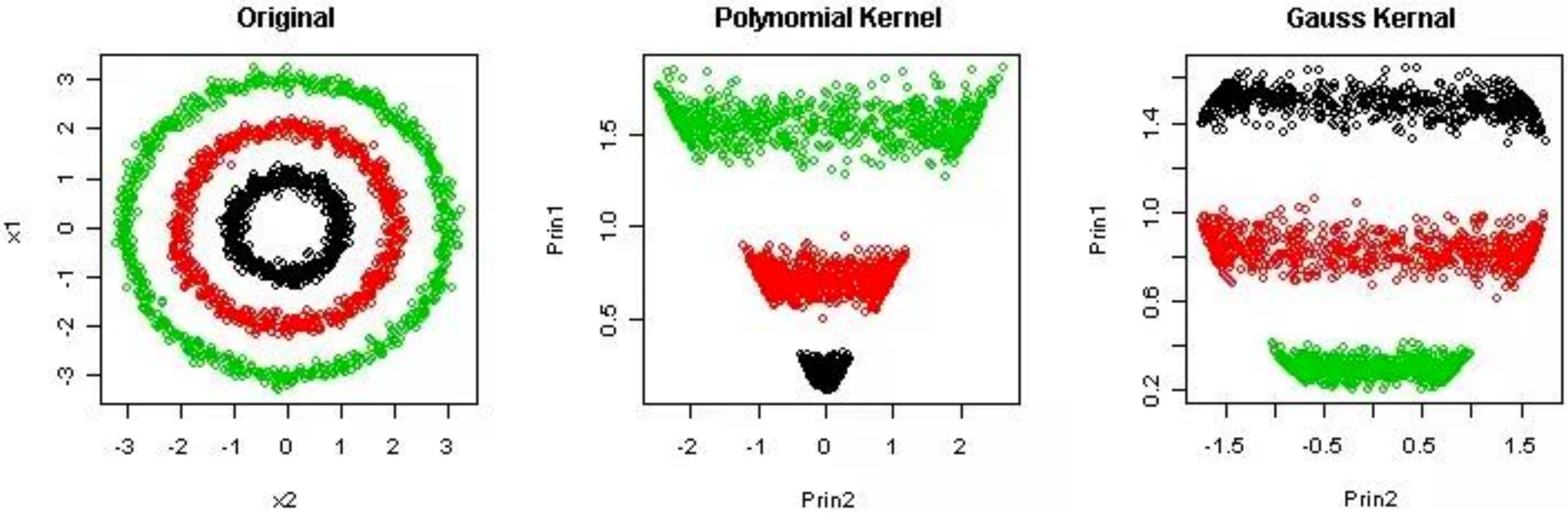
Hinge loss

- $\mathcal{L}(f; x_u, x_v, y_{uv}) = \max(0, 1 - y_{uv}(f(x_u) - f(x_v)))$

Nice properties inherited from standard SVM

- Good generalization via margin maximization
- Non-linear models via the kernel trick

Kernel trick



Limitations of the pairwise approach

Pairwise labels ignore graded relevance

- $(x_1, x_2, 1)$, regardless of the grades of x_1 and x_2

Query dominance exacerbated

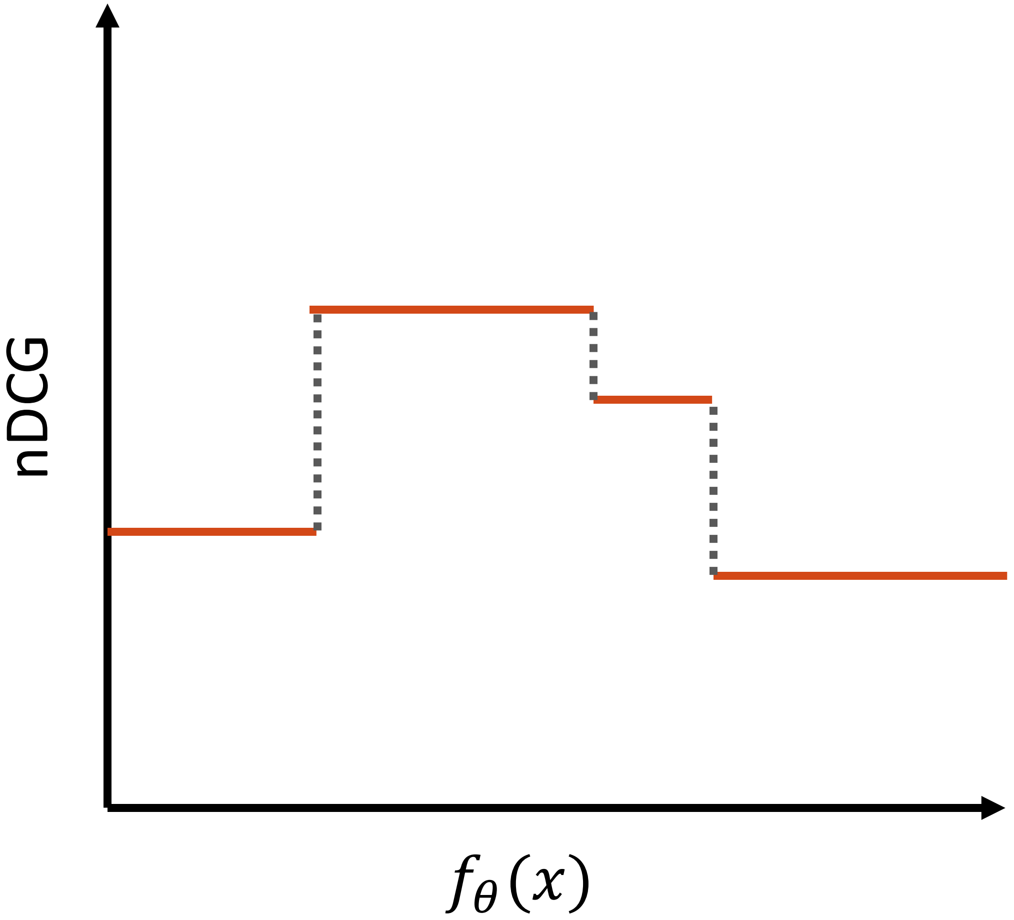
- A query with more docs will have way more pairs

Ranking positions still not taken into account

- Swaps at the top more important than at the bottom

Can we optimize ranking metrics directly?

Ranking metrics generally non-differentiable



Piecewise constant functions

- **Flat:** zero derivative
- **Discontinuous:** undef derivative

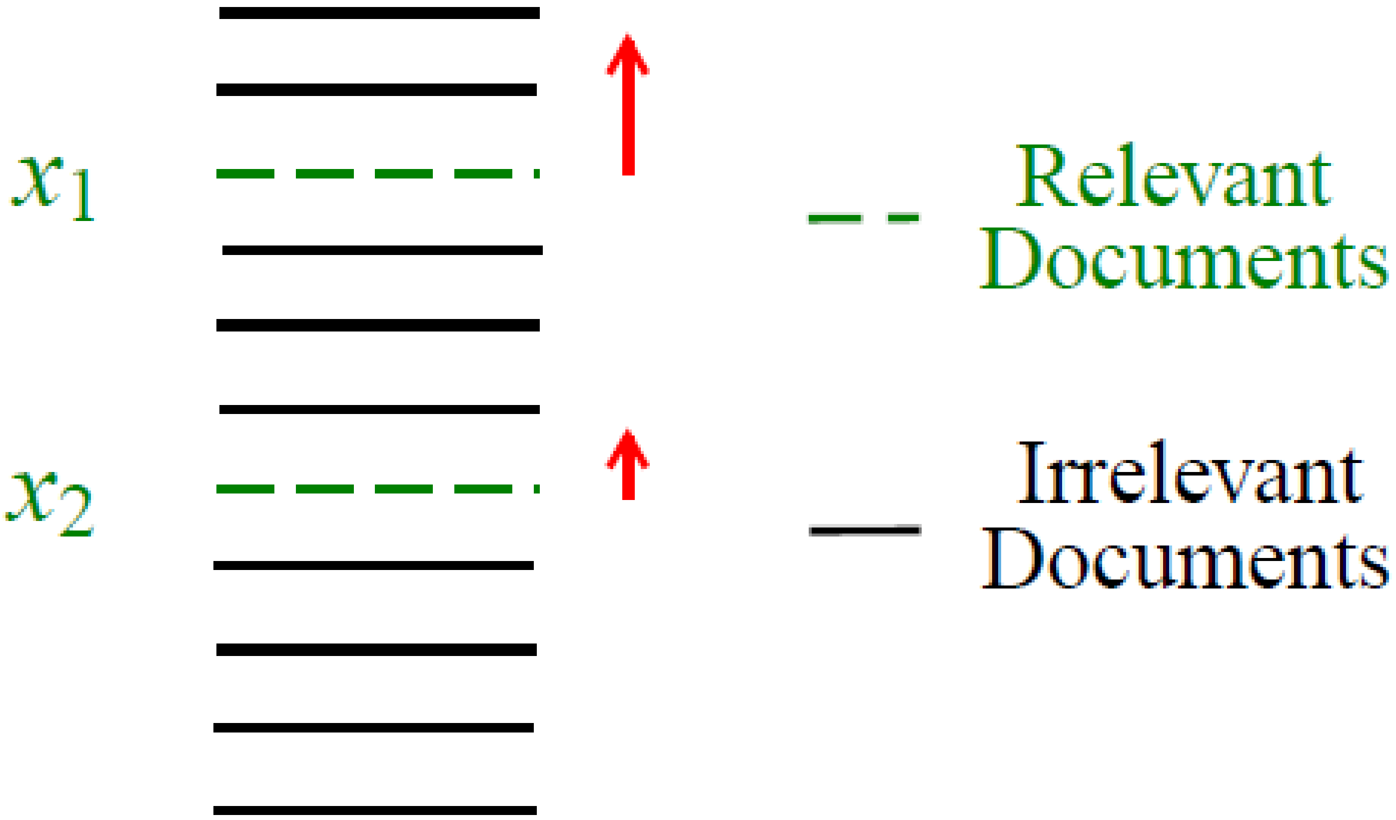
LambdaRank
(Burges, NIPS 2006)

An extension of RankNet

- Ranking evaluation metrics (which are position-based) are directly used to define the gradient with respect to each document pair in the training process

Why is it feasible to directly define the gradient?

LambdaRank
(Burges, NIPS 2006)



$$\frac{\partial L}{\partial s_1} > \frac{\partial L}{\partial s_2}$$

LambdaRank

(Burges, NIPS 2006)

Gradient determines magnitude of updates

- $w = w - \alpha\, \nabla \mathcal{L}(w)$

Vector $\nabla \mathcal{L}(w) = \left(\frac{\partial \mathcal{L}(w)}{\partial w_1}, \frac{\partial \mathcal{L}(w)}{\partial w_2}, \ldots, \frac{\partial \mathcal{L}(w)}{\partial w_d} \right)$

- $\frac{\partial}{\partial w_k} \mathcal{L}(w) = \sum_{\langle u,v \rangle} \underline{\lambda_{uv}}\, \underline{x_k^{(i)}}$

gradient magnitude (prediction error)   *k^{th} feature score*

LambdaRank
(Burges, NIPS 2006)

Lambda function

- An arbitrary surrogate for the gradient magnitude, assuming no particular loss function

$$\lambda_{uv} \equiv \frac{2^{y_u} - 2^{y_v}}{1 + \exp\left(f(x_u) - f(x_v)\right)} \left|\Delta \mathrm{nDCG}(x_u, x_v)\right|$$

LambdaMART
(Wu et al., Tech. Report 2008)

MART = Multiple Additive Regression Trees

- Commercial name for gradient boosted trees

Boosted tree version of LambdaRank

- Lambda functions guide the construction of weak learners (regression trees) via boosting
- $h_t^* = \text{argmin}_{h_t} \sum_{(x,y)} (h_t(x) - (-\alpha \, \nabla \mathcal{L}(f_t))^2$

Listwise approach

Metric-specific loss

- Optimize evaluation metrics

Non-metric-specific loss

- Optimize other listwise functions

$$\{(\mathbf{x}, \mathbf{y})\}$$

Metric-specific listwise ranking

It is natural to directly optimize what is used to evaluate the ranking results, but not trivial

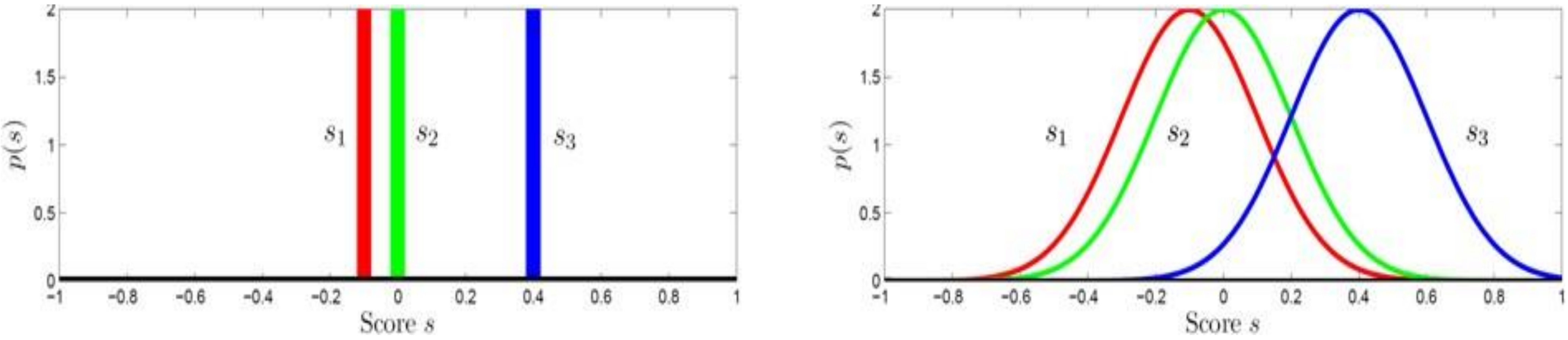
- Evaluation metrics such as nDCG and MAP are non-continuous and non-differentiable
- Most optimization techniques were developed to handle continuous and differentiable cases

SoftRank
(Taylor et al. LR4IR 2007/WSDM 2008)

Key idea: “soften” the evaluation metric

Score s_i as a random variable

- $P(s_i) = N(s_i | f(x_i), \sigma_s^2)$

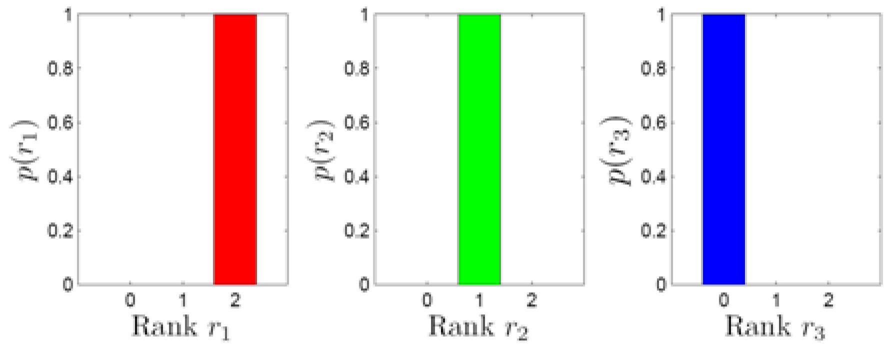


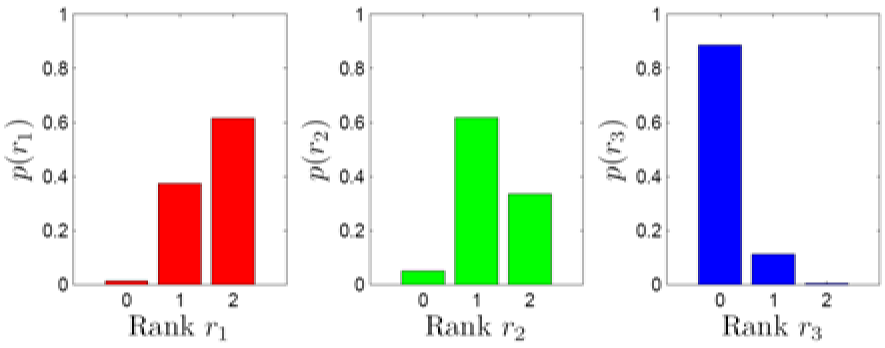
SoftRank
(Taylor et al. LR4IR 2007/WSDM 2008)

We've constructed a score distribution per rank

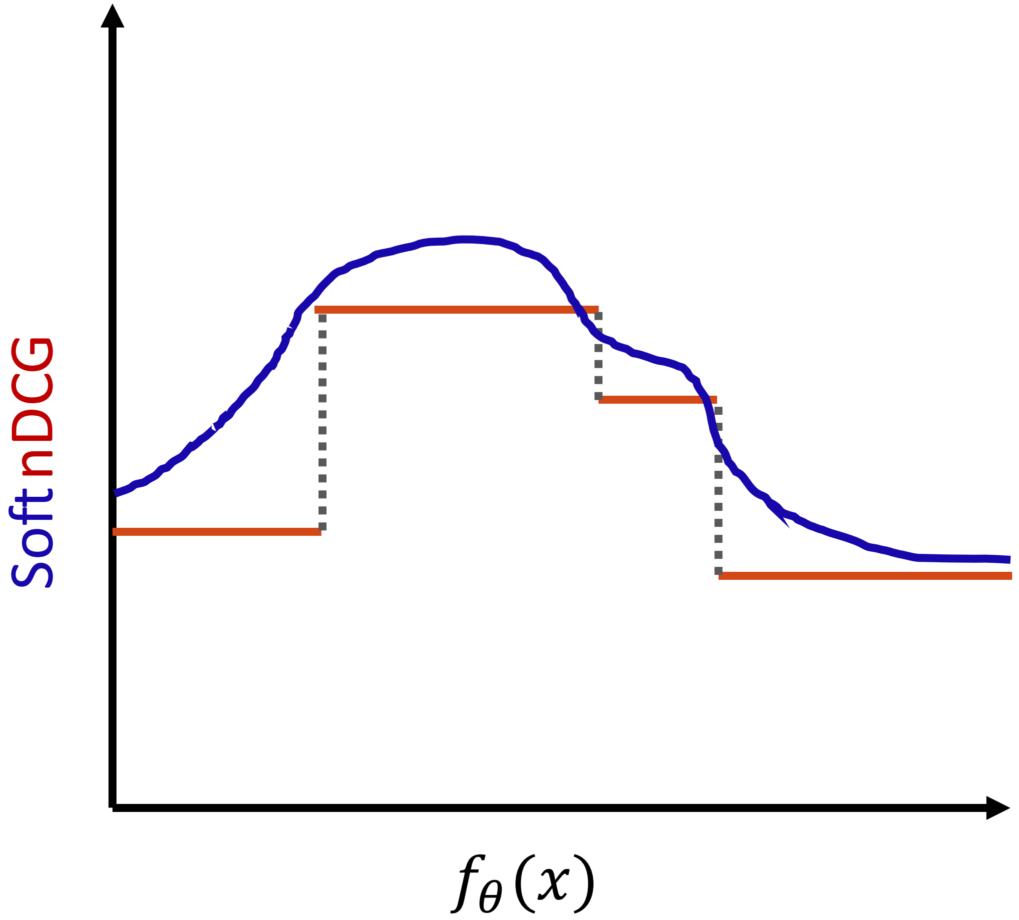
- Non-deterministic scores make it possible to find a document in any ranking position

Map score distribution to rank distribution





SoftRank
(Taylor et al. LR4IR 2007/WSDM 2008)



Now each document has a non-deterministic position

- Each draw results in a permut.
- Each permut. has a given nDCG

Compute SoftNDCG as the expected nDCG over all possible permutations (smooth and differentiable)

Optimize via gradient descent

Metric-specific listwise ranking

Workarounds

- Soften or upper-bound non-smooth objective

Optimize the non-smooth objective directly

- Use genetic programming (RankGP)
- Use ranking evaluation metric to update the training data distribution via boosting (AdaRank)

AdaRank

(Xu and Li, SIGIR 2007)

Algorithm 2 Learning Algorithms for AdaRank

Input: document group for each query

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

training set biased by query difficulty

For $t = 1, \cdots, T$

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

single-feature weak ranker

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

weighted accuracy of f_t

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

update difficulty estimates

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

Non-metric-specific listwise ranking

Defining listwise loss functions based on the understanding of unique properties of ranking

Representative algorithms

- ListNet
- ListMLE
- BoltzRank

Ranking loss is non-trivial!

An example

- Function f: $f(A) = 4, f(B) = 2, f(C) = 3 \quad (ACB)$
- Function h: $h(A) = 4, h(B) = 6, h(C) = 3 \quad (BAC)$
- Ground-truth y: $y(A) = 6, y(B) = 4, y(C) = 3 \quad (ABC)$

Which function (f or h) is closer to the ground-truth?

- According to nDCG, f should be closer to y!

ListNet
(Cao et al., ICML 2007)

Given f: $f(A) = 4, f(B) = 2, f(C) = 3$

- Deterministically: $P(ACB) = 1$

What if other permutations have a non-zero prob?

- For a k-permutation π^k under function f

$$P_f\left(\pi^k\right) = \frac{f(\pi_1)}{\sum_{i=1}^{k} f(\pi_i)} \times \frac{f(\pi_2)}{\sum_{i=2}^{k} f(\pi_i)} \times \cdots \times \frac{f(\pi_{k-1})}{\sum_{i=k-1}^{k} f(\pi_i)}$$

ListNet
(Cao et al., ICML 2007)

Given f: $f(A) = 4, f(B) = 2, f(C) = 3$

- Deterministically: $P(ACB) = 1$

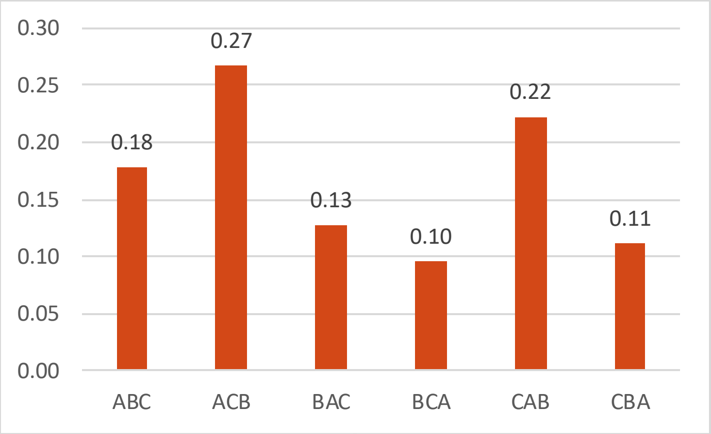
What if other permutations have a non-zero prob?

- Probabilistically: $P(ACB) = \frac{f(A)}{f(A)+f(C)+f(B)} \times \frac{f(C)}{f(C)+f(B)}$

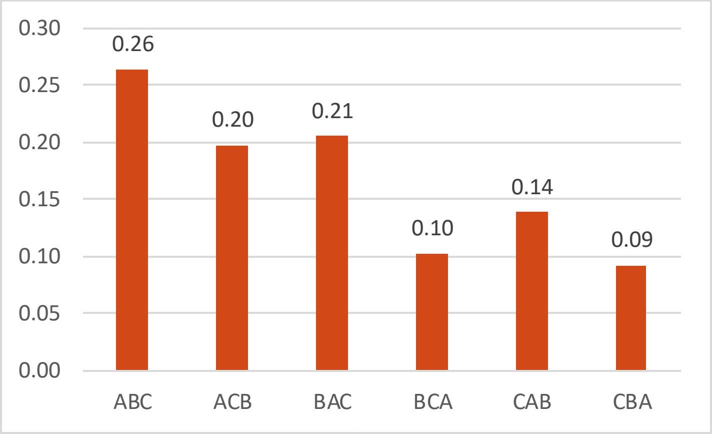
$$= \frac{4}{4+2+3} \times \frac{2}{2+3} = 0.27$$

Distance between ranked lists

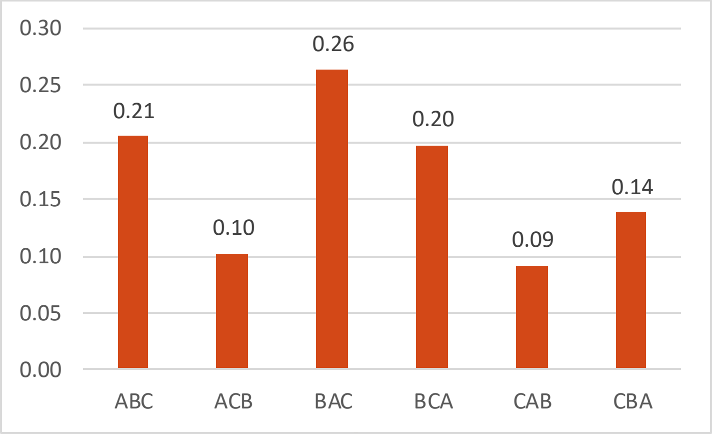
$f\colon f(A) = 4, f(B) = 2, f(C) = 3$



$y\colon y(A) = 6, y(B) = 4, y(C) = 3$



$h\colon h(A) = 4, h(B) = 6, h(C) = 3$



$D(P_y||P_f) = 0.10$

$D(P_y||P_h) = 0.14$

f is closer!

KL divergence loss

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

- $\mathcal{L}(f; \mathbf{x}, y) = D(\underbrace{P_{\varphi(y)}}_{\text{ground-truth probability distro}} || \underbrace{P_{\varphi(f(\mathbf{x}))}}_{\text{predicted probability distro}})$

Neural net model, gradient descent optimizer

Summary

Listwise approaches more closely model ranking

- Take all the documents associated with the same query as one learning instance
- Ranking position is visible to the loss function

State-of-the-art on standard benchmarks (LETOR)

- Particularly effective for top-heavy evaluation

RankLib tutorial

RankLib v2.18

```
0: MART (gradient boosted regression tree)
1: RankNet
2: RankBoost
3: AdaRank
4: Coordinate Ascent
6: LambdaMART
7: ListNet
8: Random Forests
```

RankLib tutorial

https://sourceforge.net/p/lemur/wiki/RankLib%20How%20to%20use/

```
> java -jar bin/RankLib-2.18.jar \
    -train MQ2008/Fold1/train.txt \
    -validate MQ2008/Fold1/vali.txt \
    -test MQ2008/Fold1/test.txt \
    -ranker 6 -metric2t NDCG@10 -metric2T ERR@10 \
    -save mymodel.txt
```

References

Learning to rank for information retrieval
Liu, 2011

Learning to rank for information retrieval and natural language processing
Li, 2014

Learning for Web rankings
Grangier and Paiement, 2011

UFMG UNIVERSIDADE FEDERAL DE MINAS GERAIS

Coming next...

Neural Models

Rodrygo L. T. Santos
rodrygo@dcc.ufmg.br