Advanced algorithms for ML/DM (2nd part)

– Learning to rank –

Eric Gaussier

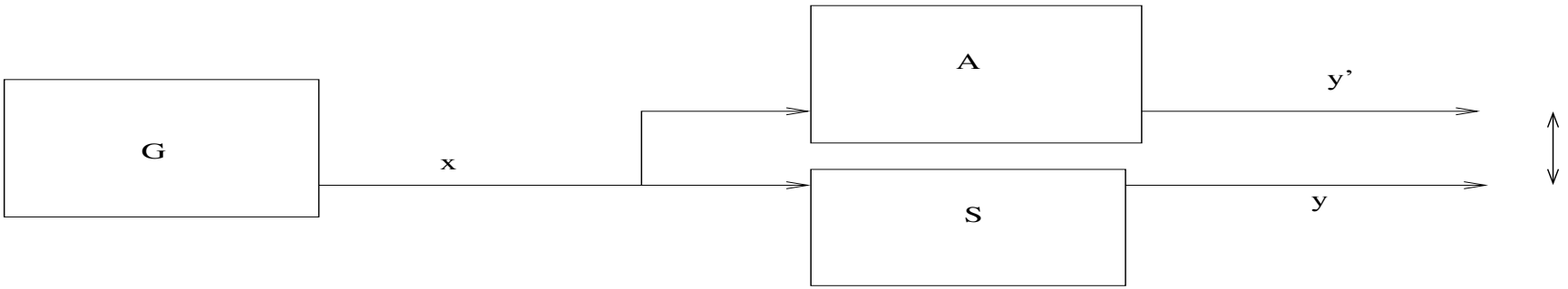
Univ. Grenoble Alpes

UFR-IM^{2}AG, LIG, MIAI@Grenoble Alpes

eric.gaussier@imag.fr

Table of content

Supervised machine learning



- $x \in \mathcal{X} \subseteq \mathbb{R}^p, x = (x_1, x_2, \cdots, x_p)^T$, is an input example
- $y \in \mathcal{Y} = \mathbb{R}$ or $\{1, \cdots, K\}$ is the desired output (produced by unobserved system S)
- From input-output examples (training set), learner A *learns* a function $f \in \mathcal{F}$ that aims at replicating system S ($y' = f(x)$ should be as close as possible to y)

Notations and fundamental concepts

1. *Training set:* $\mathcal{D} = ((x^{(1)}, y^{(1)}), \cdots, (x^{(n)}, y^{(n)}))$
2. *Family of functions:* $\mathcal{F} \subseteq \{\mathcal{X} \to \mathcal{Y}\}$ $(f \in \mathcal{F})$
3. *Loss function:*
$$L : \mathcal{Y} x \mathcal{Y} \to \mathbb{R}, \text{s.t. } L(y, y') \geq 0 \text{ for } y \neq y'$$
4. *Functional risk (true risk) for f:*
$$R(f) = E_{P(x,y)} L(y, f(x)) \; (= \sum_x \sum_y P(x, y) L(y, f(x))$$
5. *Empirical risk (training error) for f:*
$$\text{Remp}(f; \mathcal{D}) = \frac{1}{n} \sum_{i=1}^{n} L(y^{(i)}, f(x^{(i)}))$$

Notations and fundamental concepts

1. *Training set:* $\mathcal{D} = ((x^{(1)}, y^{(1)}), \cdots, (x^{(n)}, y^{(n)}))$
2. *Family of functions:* $\mathcal{F} \subseteq \{\mathcal{X} \to \mathcal{Y}\}$ $(f \in \mathcal{F})$
3. *Loss function:*

$$L : \mathcal{Y}x\mathcal{Y} \to \mathbb{R}, \text{s.t. } L(y, y') \geq 0 \text{ for } y \neq y'$$

4. *Functional risk (true risk) for f:*

$$R(f) = E_{P(x,y)} L(y, f(x)) \; (= \sum_x \sum_y P(x, y) L(y, f(x))$$

5. *Empirical risk (training error) for f:*

$$\text{Remp}(f; \mathcal{D}) = \frac{1}{n} \sum_{i=1}^{n} L(y^{(i)}, f(x^{(i)}))$$

Notations and fundamental concepts

1. *Training set:* $\mathcal{D} = ((x^{(1)}, y^{(1)}), \cdots, (x^{(n)}, y^{(n)}))$
2. *Family of functions:* $\mathcal{F} \subseteq \{\mathcal{X} \to \mathcal{Y}\}$ $(f \in \mathcal{F})$
3. *Loss function:*

$$L : \mathcal{Y}x\mathcal{Y} \to \mathbb{R}, \text{s.t. } L(y, y') \geq 0 \text{ for } y \neq y'$$

4. *Functional risk (true risk) for f :*

$$R(f) = E_{P(x,y)} L(y, f(x)) \; (= \sum_x \sum_y P(x, y) L(y, f(x))$$

5. *Empirical risk (training error) for f :*

$$\text{Remp}(f; \mathcal{D}) = \frac{1}{n} \sum_{i=1}^{n} L(y^{(i)}, f(x^{(i)}))$$

Notations and fundamental concepts

1. *Training set:* $\mathcal{D} = ((x^{(1)}, y^{(1)}), \cdots, (x^{(n)}, y^{(n)}))$
2. *Family of functions:* $\mathcal{F} \subseteq \{\mathcal{X} \to \mathcal{Y}\}$ $(f \in \mathcal{F})$
3. *Loss function:*

$$L : \mathcal{Y}x\mathcal{Y} \to \mathbb{R}, \text{s.t. } L(y, y') \geq 0 \text{ for } y \neq y'$$

4. *Functional risk (true risk) for f:*

$$R(f) = E_{P(x,y)} L(y, f(x)) \; (= \sum_x \sum_y P(x, y) L(y, f(x))$$

5. *Empirical risk (training error) for f:*

$$\text{Remp}(f; \mathcal{D}) = \frac{1}{n} \sum_{i=1}^{n} L(y^{(i)}, f(x^{(i)}))$$

Notations and fundamental concepts

1. *Training set:* $\mathcal{D} = ((x^{(1)}, y^{(1)}), \cdots, (x^{(n)}, y^{(n)}))$
2. *Family of functions:* $\mathcal{F} \subseteq \{\mathcal{X} \to \mathcal{Y}\}$ $(f \in \mathcal{F})$
3. *Loss function:*

$$L : \mathcal{Y}x\mathcal{Y} \to \mathbb{R}, \text{s.t. } L(y, y') \geq 0 \text{ for } y \neq y'$$

4. *Functional risk (true risk) for f :*

$$R(f) = E_{P(x,y)} L(y, f(x)) \; (= \sum_x \sum_y P(x, y) L(y, f(x))$$

5. *Empirical risk (training error) for f :*

$$\text{Remp}(f; \mathcal{D}) = \frac{1}{n} \sum_{i=1}^{n} L(y^{(i)}, f(x^{(i)}))$$

Notations and fundamental concepts (cont'd)

What we would like

$$\text{argmin}_{f \in \mathcal{F}} R(f)$$

What we can have

$$\text{argmin}_{f \in \mathcal{F}} \text{Remp}(f; \mathcal{D}) = \text{argmin}_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} L(y^{(i)}, f(x^{(i)}))$$

Notations and fundamental concepts (cont'd)

What we would like

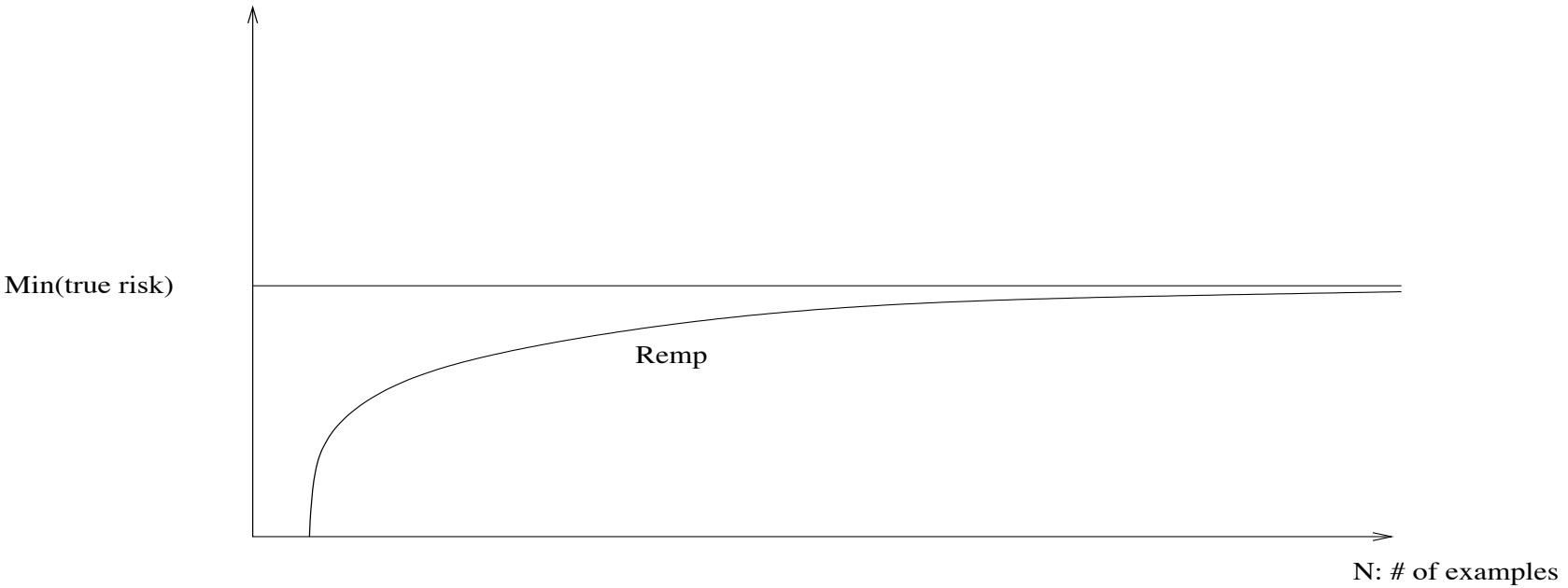
$$\text{argmin}_{f \in \mathcal{F}} R(f)$$

What we can have

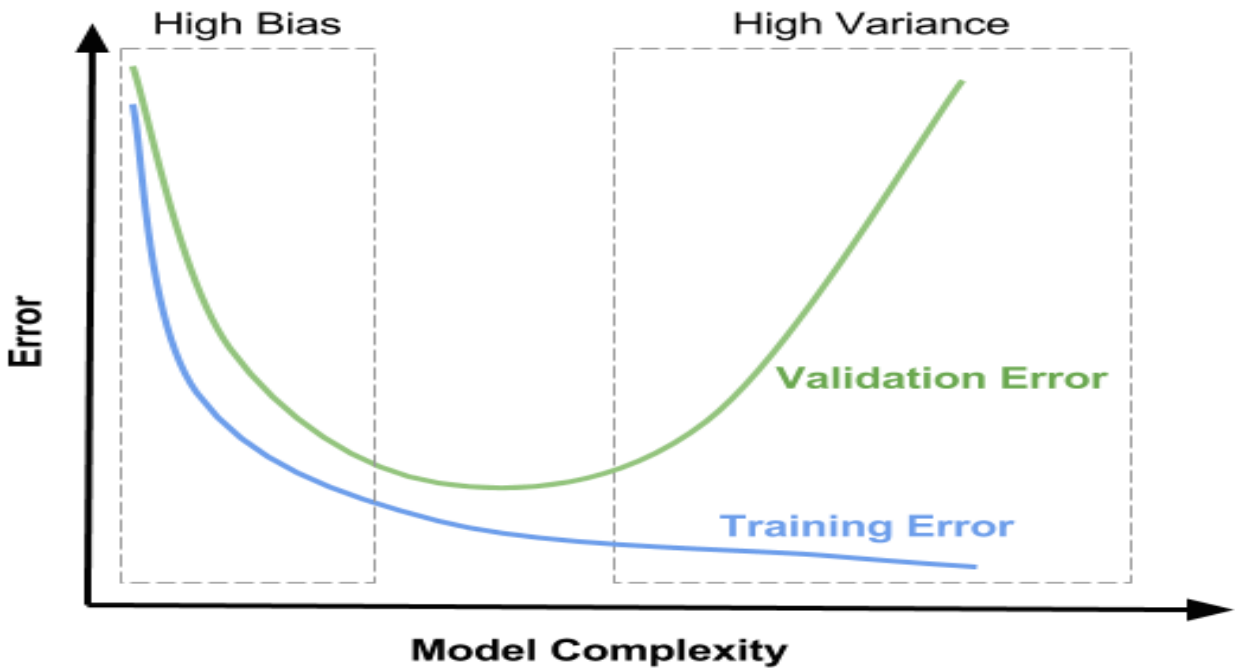
$$\text{argmin}_{f \in \mathcal{F}} \text{Remp}(f; \mathcal{D}) = \text{argmin}_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} L(y^{(i)}, f(x^{(i)}))$$

Justifying the empirical risk minimization principle

When the number of examples increases ($\rightarrow +\infty$), the empirical risk converges to the true risk



In practice: Finite samples and the bias-variance tradeoff



Overtraining problem

Regularizing the empirical risk

Aiming at a tradeoff between minimizing the empirical risk and avoiding too complex models

$$\text{argmin}_{f \in \mathcal{F}} \overbrace{\sum_{i=1}^{n} L(y^{(i)}, f(x^{(i)}))}^{\text{Remp}(f;\mathcal{D})} + \lambda C_{\mathcal{F}}(f)$$

- λ is an hyperparameter that controls this tradeoff
- $C_{\mathcal{F}}(f)$ measures the complexity of $f \in \mathcal{F}$

Regularizing the empirical risk – illustration

- Regression problem ($y \in \mathbb{R}$)
- $\mathcal{F}$: $w_0 + w_1 x_1 + \cdots w_p x_p$ (family of linear functions, parametrized by $w \in \mathbb{R}^{p+1}$)
- L2 loss
- Complexity measures by $||w||_2^2 = \sum_{l=0}^{p} w_l^2$

Then:

$$\text{argmin}_{f \in \mathcal{F}} \sum_{i=1}^{n} L(y^{(i)}, f(x^{(i)})) + \lambda C_{\mathcal{F}}(f)$$

$\Leftrightarrow$

$$\text{argmin}_{w \in \mathbb{R}^{p+1}} \sum_{i=1}^{n} (y^{(i)} - w_0 + w_1 x_1 + \cdots w_p x_p)^2 + \lambda \sum_{l=0}^{p} w_l^2$$

A few words on $\mathcal{F}$

The estimation-approximation tradeoff

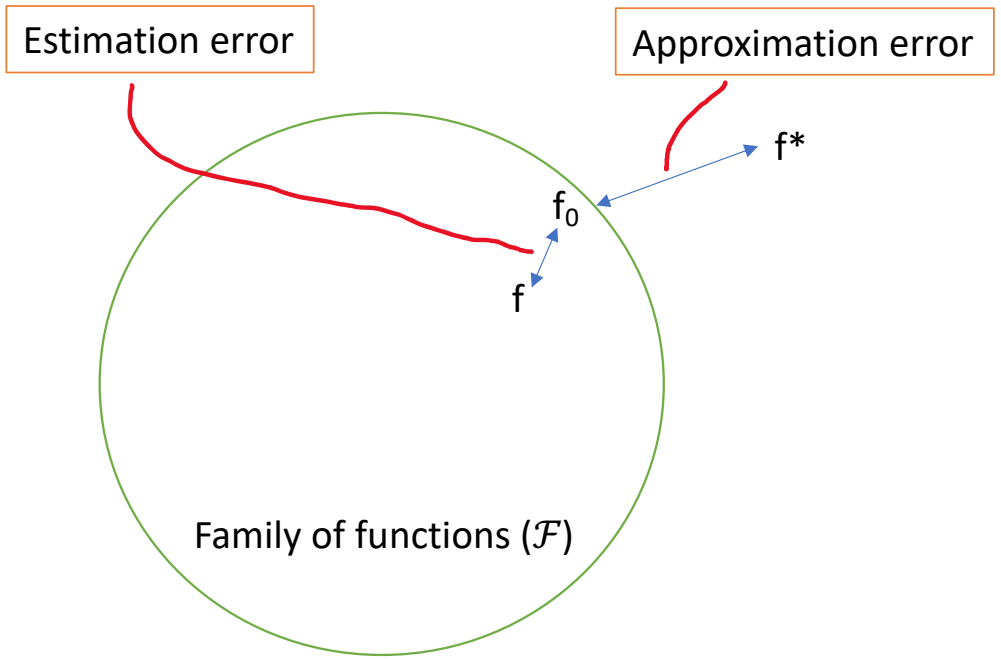


Table of content

Introduction

1. In many applications (as IR), the main goal is to **rank** objects
2. A simple, however not optimal, way to to do so is to learn a score for an object
3. Learning a regression function
   - Neural nets, SVR, gradient boosted trees, ...
   - Typical losses *mean square error (MSE or L_2 loss), mean absolute error (MAE or L_1 loss)*
4. Learning a classification function
   - Many objects have binary class labels, with some ordering (class 1, C_1, is preferred over class 0)
   - Typical loss *cross-entropy loss (CE)*

MSE: $(\hat{y}_i - y_i)^2$; CE: $-y_i \log(p_i) - (1 - y_i) \log(1 - p_i)$, $p_i = P(x_i \in C_1)$

Illustration

The goal is to find the ranking (o'_2, o'_3, o'_1) from some training data in which objects are assigned scores or ordered class labels

1. Learn regressor on training data; compute scores for o'_1, o'_2 and o'_3; rank them according to their score
2. Do the same with the classifier, using the score of class 1

In IR, binary relevance judgements (training data) in the form:
$q_1 : (d^{(1)}, 1); (d^{(2)}, 0); \cdots$
$q_2 : \cdots$
$\cdots$

What is an example (an $x \in \mathbb{R}^p$)? A document? A query?

Modeling IR as a binary classification problem

$\rightarrow$ A (query,doc) pair: $x = (q, d) \in \mathbb{R}^p$

General coordinates (p is a few tens)

$$x_1 = \sum_{w \in q \bigcap d} \log(tf_w^d)$$
$$x_2 = \sum_{w \in q \bigcap d} \log(\text{idf}_w)$$
...
$$x_{17} = q \cdot d$$
$$x_{18} = BM25(q, d)$$
...
$$x_{30} = \text{PageRank}(d)$$
...

Modeling IR as a binary classification problem (cont'd)

Each example $x(=(q,d))$ containing a document relevant (resp. non relevant) to the query is in C_1 (resp. C_0)

Remarks

1. One uses the value of the decision function to obtain an order on documents
2. Drawbacks (both classificaiton and regression)
   - Method that assigns a score to a document independently of other documents ($\rightarrow$ *pointwise method*)
   - Loss function not built from evaluation function (true objective)

Preference pairs and ranking

1. In many cases, one does not have a complete ranking of objects to learn from, but rather a partial ordering which can take the form of preference pairs
2. For instance, relevance is not an absolute notion and it is easier to compare the relative relevance of say two documents
3. In such cases, one is looking for a function f that preserves partial order between objects:

$$x^{(i)} \succ x^{(j)} \iff f(x^{(i)}) > f(x^{(j)})$$

Remark: Preference pairs can be obtained from (binary) relevance judgements as well as from complete or partial orderings

How to learn from such pairs?

Idea: Transform a preference pair information into a classification information by forming the difference between objects in the pair

$$x^{(i,j)} = (x^{(i)} - x^{(j)})$$
$$y^{(i,j)} = \begin{cases} +1 \text{ if } x^{(i)} \succ (j) \\ 0 \text{ if } x^{(j)} \succ x^{(i)} \end{cases}$$

One can then use any classifier as before ($\rightarrow$ *pairwise method*)

Remark: In the case of IR, $x^{(i)}$ and $x^{(j)}$ are (doc,query) pairs with the same query ($x^{(i)} \succ x^{(j)}$ means $d^{(i)}$ is more relevant than $d^{(j)}$ for the query)

In practice

How to apply a classifier learnt on preference pairs?

- For each object to be ranked, form a pair with the *null* object (null vector)
- The score obtained for each such pair is an indication of the score of the object
- Objects are thus ranked acc. to this score (optimal in some case)

Remarks:

- No difference between errors made at the top or at the middle of the list ($\rightarrow$ adaptation of the loss)
- Still not aligned with evaluation functions for ranking

In practice

How to apply a classifier learnt on preference pairs?

- For each object to be ranked, form a pair with the *null* object (null vector)
- The score obtained for each such pair is an indication of the score of the object
- Objects are thus ranked acc. to this score (optimal in some case)

Remarks:

- No difference between errors made at the top or at the middle of the list ($\rightarrow$ adaptation of the loss)
- Still not aligned with evaluation functions for ranking

In practice

How to apply a classifier learnt on preference pairs?

- For each object to be ranked, form a pair with the *null* object (null vector)
- The score obtained for each such pair is an indication of the score of the object
- Objects are thus ranked acc. to this score (optimal in some case)

Remarks:

- No difference between errors made at the top or at the middle of the list ($\rightarrow$ adaptation of the loss)
- Still not aligned with evaluation functions for ranking

Extensions/General remarks

1. *Listwise* approach
   - Directly treat lists as examples
   - Possibility to use (approximations of) ranking evaluation functions as objectives/losses
2. Methods that require training data

Table of content

Relevance judgements

- Binary judgements: the doc is relevant (1) or not relevant (0) to the query
- Multi-valued judgements: $Perfect > Excellent > Good > Correct > Bad$
- Preference pairs: doc d_A more relevant than doc d_B to the query

Several (large) collections with many (> 30) queries and associated (binary) relevance judgements: TREC collections (trec.nist.gov), CLEF (www.clef-campaign.org), FIRE (fire.irsi.res.in)

Common evaluation measures

- MAP (Mean Average Precision)
- MRR (Mean Reciprocal Rank)
  - For a given query q, let r_q be the rank of the first relevant document retrieved
  - MRR: mean of $\frac{1}{r_q}$ over all queries
- WTA (Winner Takes All)
  - If the first retrieved doc is relevant, $s_q = 1$; $s_q = 0$ otherwise
  - WTA: mean of s_q over all queries
- NDCG (Normalized Discounted Cumulative Gain)

NDCG

- NDCG at position k:

$$N(k) = \overbrace{Z_k}^{\text{normalization}} \underbrace{\sum_{j=1}^{k}}_{\text{cumul}} \overbrace{(2^{r(j)} - 1)}^{\text{gain}} \; / \; \underbrace{\log_2(j+1)}_{\text{position discount}}$$

- Averaged over all queries

G : Gain

Relevance	Value (gain)
Perfect (5)	$31 = 2^5 - 1$
Excellent (4)	$15 = 2^4 - 1$
Good (3)	$7 = 2^3 - 1$
Correct (2)	$3 = 2^2 - 1$
Bad (0)	$0 = 2^1 - 1$

DCG : Discounted CG

Discounting factor: $\frac{\ln(2)}{\ln(j+1)}$

Doc. (rank)	Rel..	Gain	CG	DCG
1	*Perf. (5)*	31	31	31
2	*Corr. (2)*	3	34	$32.9 = 31 + 3 \times 0.63$
3	*Exc. (4)*	15	49	$40,4$
4	*Exc. (4)*	15	64	$46,9$
...	...	...	...	...

Ideal ranking: max DCG

Document (rank)	Relevance	Gain	max DCG
1	*Perfect (5)*	31	31
2	*Excellent (4)*	15	40.5
3	*Excellent (4)*	15	48
4	*Correct (2)*	3	49.3
...	...	...	...

Normalized DCG

Doc. (rank)	Rel.	Gain	DCG	max DCG	NDCG
1	*Perfect (5)*	31	31	31	1
2	*Correct (2)*	3	32.9	40.5	0.81
3	*Excellent (4)*	15	40.4	48	0.84
4	*Excellent (4)*	15	46.9	49.3	0.95
...	...	...	...	...	

Remarks on evaluation measures

- Measures for a given position (e.g. list of 10 retrieved documents)

- NDCG is more general than MAP (multi-valued relevance vs binary relevance)

- Non continuous (and thus non derivable), but can be approximated!

Table of content

Building training data

- Several annotated collections exist
  - TREC (TREC-vidéo)
  - CLEF
  - NTCIR

- For new collections, as intranets of companies, such collections do not exist and it may be difficult to build them $\rightarrow$ standard models, with little training

- What about the web?

Training data on the web

- An important source of information; click data from users
  - Use clicks to infer preferences between docs (preference pairs)
  - In addition, and if possible, use eye-tracking data
- What can be deduced from clicks?

Joachims et al. *Accurately Interpreting Clickthrough Data as Implicit Feedback*, SIGIR 2005

Exploiting clicks (1)

Clicks can not be used to infer absolute relevance judgements; they can nevertheless be used to infer relative relevance judgements. Let $(d_1, d_2, d_3, \cdots)$ be an ordered list of documents retrieved for a particular query and let C denote the set of clicked documents. The following strategies can be used to build relative relevance judgements:

1. If $d_i \in C$ and $d_j \notin C$, $d_i \succ_{rel_q} d_j$
2. If d_i is the last clicked doc, $\forall j < i$, $d_j \notin C$, $d_i \succ_{rel_q} d_j$
3. $\forall i \geq 2, d_i \in C, d_{i-1} \notin C, d_i \succ_{rel_q} d_{i-1}$
4. $\forall i, d_i \in C, d_{i+1} \notin C, d_i \succ_{rel_q} d_{i+1}$

Exploiting clicks (2)

- The above strategies yield a partial order between docs
- Leading to a very large training set on which one can deploy learning to rank methods
- IR on the web has been characterized by a "data rush":
  - Index as many pages as possible
  - Get as many click data as possible

Letor

http://research.microsoft.com/en-us/um/beijing/projects/letor/

Tao Qin, Tie-Yan Liu, Jun Xu, and Hang Li. *LETOR: A Benchmark Collection for Research on Learning to Rank for Information Retrieval*, Information Retrieval Journal, 2010

Table of content

Conclusion

- Approaches aiming at exploiting all the available information (60 features for the *gov* collection for example - including scores of standard IR models)
- Approaches aiming at "ranking" documents (*pairwise*, *listwise*)
- Many proposals (neural nets, *boosting* and ensemble methods, ...); no clear difference on all collections
- State-of-the-art methods when many features available

References

• Burges et al. *Learning to Rank with Nonsmooth Cost Functions*, NIPS 2006

• Cao et al. *Adapting Ranking SVM to Document Retrieval*, SIGIR 2006

• Cao et al. *Learning to Rank: From Pairwise to Listwise Approach*, ICML 2007

• Workshop LR4IR, 2007 (Learning to Rank for Information Retrieval).

• Liu *Learning to Rank for Information Retrieval*, tutoriel, 2008.