Learning to Rank for IR

Information Retrieval

Indian Statistical Institute

Outline

Where is learning involved in IR?

- Relevance feedback
- Parameter tuning
  (k_1, b, λ, teleportation probability, etc.)
  - simple / brute-force approach: grid search
  - large number of features / parameters $\Rightarrow$ ML techniques can help
- *Data fusion:* merging of ranked lists

Basic framework

- **Document representation**

	Feature 1	Feature 2	...	Feature k
D	x_1	x_2	...	x_k

- **Features:**
  - *Query-dependent*
    - total BM25 score for document
    - total BM25 score for title of document
    - total Query Likelihood score for document using Dirichlet smoothing
    - ... for title of document using Dirichlet smoothing
    - ... anchor text
    - etc. $\longleftarrow$ *new / improved models may be incorporated as additional dimensions*
  - *Query-independent*
    - PageRank
    - document length
    - etc.

Basic framework

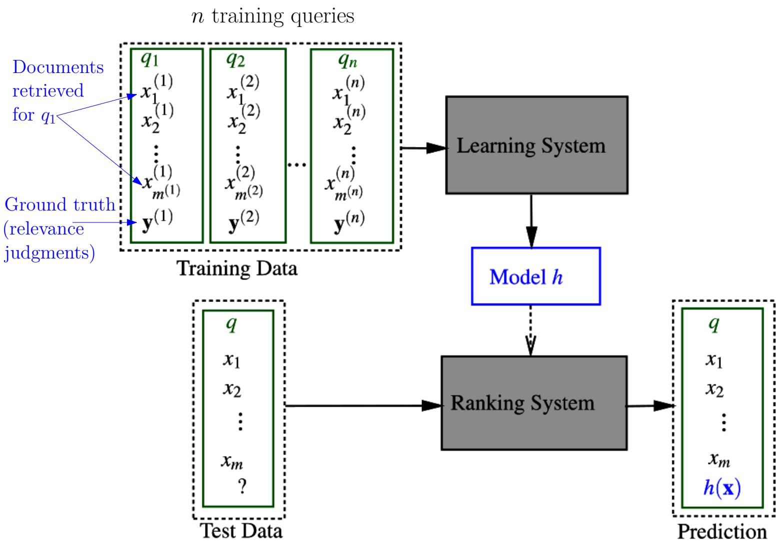
- **Document representation**

	Feature 1	Feature 2	…	Feature k
D	x_1	x_2	…	x_k

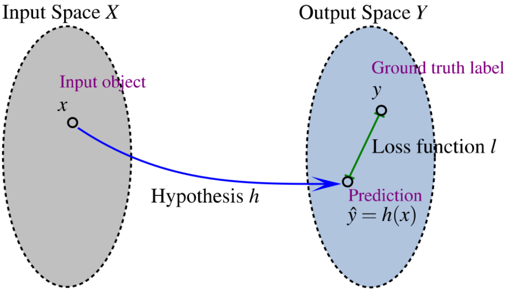
- **Features:**
  - *Query-dependent*
    - total BM25 score for document
    - total BM25 score for title of document
    - total Query Likelihood score for document using Dirichlet smoothing
    - … for title of document using Dirichlet smoothing
    - … anchor text
    - etc. $\longleftarrow$ *new / improved models may be incorporated as additional dimensions*
  - *Query-independent*
    - PageRank
    - document length
    - etc.

NB: Model parameters assumed to be fixed; only optimal way of combining features is learned.

Basic framework



Learning framework



Hypothesis: function from input space to output space

Overview of approaches

- **Pointwise approach**
  - input space: individual feature vectors
  - output space: relevance score
- **Pairwise approach**
  - input space: feature vector pairs
  - output space: $\{+1, -1\}$
- **Listwise approach**
  - input space: set of feature vectors corresponding to documents to be ranked for a particular query
  - output space: permutation / ordering of elements of input set

Outline

Logistic regression

Reference: GEY, SIGIR 94

- **Input space:** document feature vector $\mathbf{x}$
- **Output space:** probability of relevance p
- **Hypothesis:**

$$p_\theta(\mathbf{x}) \;\; (\text{or } h_\theta(\mathbf{x})) \;\; = \;\; \frac{1}{1+\exp(-\theta^T\mathbf{x})}$$

- **Loss function:**

$$J(\theta) = -\left[\, y \log p_\theta(\mathbf{x}) \; + \; (1-y)\log(1-p_\theta(\mathbf{x})) \,\right]$$

- **Training:** choose $\arg\min_\theta J(\theta)$ using gradient descent
- **Ranking of test documents:** using $p_\theta(\mathbf{x}_{test})$

Association rules

Reference: VELOSO ET AL., SIGIR 2008

- Association rules: $\mathcal{X} \rightarrow r_i$
  - *antecedent* ($\mathcal{X}$): features
  - *consequent* (r_i): relevance level
    e.g., strongly relevant (3), relevant (2), marginally relevant (1), non-relevant (0)
- *Support* $\sigma(\mathcal{X} \rightarrow r_i)$: fraction of training examples containing features $\mathcal{X}$ and having relevance level r_i
- *Confidence* $\theta(\mathcal{X} \rightarrow r_i)$: probability of observing relevance level r_i, given that $\mathcal{X}$ holds
- Thresholds
  - $\sigma_{\min}$: consider only rules that are frequently useful
  - $\theta_{\min}$: consider only rules that suggest strong implication

Example

	Query	Retrieved Documents				Relevance
		id	PageRank	BM25	*tf*	
Training Data	federal grant programs	1	[0.85-0.92]	[0.36-0.55]	[0.23-0.27]	1
		2	[0.74-0.84]	[0.36-0.55]	[0.46-0.61]	1
		3	[0.51-0.64]	[0.56-0.70]	[0.23-0.27]	0
	scholarship programs	4	[0.74-0.84]	[0.36-0.55]	[0.28-0.45]	0
		5	[0.65-0.73]	[0.56-0.70]	[0.46-0.61]	1
		6	[0.93-1.00]	[0.36-0.55]	[0.62-0.76]	0
	international trade	7	[0.74-0.84]	[0.22-0.35]	[0.12-0.22]	0
		8	[0.65-0.73]	[0.56-0.70]	[0.46-0.61]	0
		9	[0.85-0.92]	[0.71-0.80]	[0.46-0.61]	1
Test Set	after-school programs	10	[0.85-0.92]	[0.56-0.70]	[0.46-0.61]	1
		11	[0.51-0.64]	[0.36-0.55]	[0.28-0.45]	0
		12	[0.34-0.50]	[0.22-0.35]	[0.46-0.61]	1

Generated rules $(\mathcal{R})$**:** $(\sigma_{\min} = 0.2, \theta_{\min} = 0.67)$

- PageRank=[0.85,0.92] $\rightarrow\ r = 1\ \ (\theta = 1.00)$
- PageRank=[0.74,0.84] $\rightarrow\ r = 0\ \ (\theta = 0.67)$
- BM25=[0.56,0.70] $\rightarrow\ r = 0\ \ (\theta = 0.67)$
- *tf*=[0.46,0.61] $\rightarrow\ r = 1\ \ (\theta = 0.75)$

Example

	Query	Retrieved Documents				Relevance
		id	PageRank	BM25	*tf*	
Training Data	federal grant programs	1	[0.85-0.92]	[0.36-0.55]	[0.23-0.27]	1
		2	[0.74-0.84]	[0.36-0.55]	[0.46-0.61]	1
		3	[0.51-0.64]	[0.56-0.70]	[0.23-0.27]	0
	scholarship programs	4	[0.74-0.84]	[0.36-0.55]	[0.28-0.45]	0
		5	[0.65-0.73]	[0.56-0.70]	[0.46-0.61]	1
		6	[0.93-1.00]	[0.36-0.55]	[0.62-0.76]	0
	international trade	7	[0.74-0.84]	[0.22-0.35]	[0.12-0.22]	0
		8	[0.65-0.73]	[0.56-0.70]	[0.46-0.61]	0
		9	[0.85-0.92]	[0.71-0.80]	[0.46-0.61]	1
Test Set	after-school programs	10	[0.85-0.92]	[0.56-0.70]	[0.46-0.61]	1
		11	[0.51-0.64]	[0.36-0.55]	[0.28-0.45]	0
		12	[0.34-0.50]	[0.22-0.35]	[0.46-0.61]	1

Generated rules $(\mathcal{R})$**:** $(\sigma_{\min} = 0.2, \theta_{\min} = 0.67)$

- PageRank=[0.85,0.92] $\rightarrow\ r = 1\ \ (\theta = 1.00)$
- PageRank=[0.74,0.84] $\rightarrow\ r = 0\ \ (\theta = 0.67)$
- BM25=[0.56,0.70] $\rightarrow\ r = 0\ \ (\theta = 0.67)$
- *tf*=[0.46,0.61] $\rightarrow\ r = 1\ \ (\theta = 0.75)$

$\mathcal{R}_d$: rules applicable to d
(3 for $d = 10$)

Relevance estimation for document d: algorithm 1

$$\text{Vote for level } r_i : \quad s(r_i) = \frac{\sum_{\mathcal{X}\to r_i \,\in\, \mathcal{R}_d} \theta(\mathcal{X} \to r_i)}{|\mathcal{X} \to r_i \;\in\; \mathcal{R}_d|}$$

$$Score(d) = \sum_{i=0}^{k} \frac{s(r_i)}{\sum_{j=0}^{k} s(r_j)}$$

Example: document 10

$\mathcal{R}_d =$ rules 1, 3, 4

$r_i = 0$: only rule 3 applicable $\Rightarrow s(0) = 0.67$

$r_i = 1$: rules 1 and 4 applicable $\Rightarrow s(1) = (1.00 + 0.75)/2 = 0.87$

Score $= 0 \times \frac{0.67}{0.67+0.87} \; + \; 1 \times \frac{0.87}{0.67+0.87} = 0.56$

Relevance estimation for document d: algorithm 1

$$\text{Vote for level } r_i: \quad s(r_i) = \frac{\sum_{\mathcal{X} \to r_i \in \mathcal{R}_d} \theta(\mathcal{X} \to r_i)}{|\mathcal{X} \to r_i \in \mathcal{R}_d|}$$

$$Score(d) = \sum_{i=0}^{k} \frac{s(r_i)}{\sum_{j=0}^{k} s(r_j)}$$

Example: document 10

$\mathcal{R}_d =$ rules 1, 3, 4

$r_i = 0$: only rule 3 applicable $\Rightarrow s(0) = 0.67$

$r_i = 1$: rules 1 and 4 applicable $\Rightarrow s(1) = (1.00 + 0.75)/2 = 0.87$

Score $= 0 \times \frac{0.67}{0.67+0.87} + 1 \times \frac{0.87}{0.67+0.87} = 0.56$

What about document 11 ($\mathcal{R}_d = \varnothing$) ?

Relevance estimation for document d: algorithm 2

On-demand rule generation

- $\mathcal{D}_d$: *projected training data* obtained after removing examples not applicable to d
- $\mathcal{R}_d$: generated from $\mathcal{D}_d$ using $\sigma_{\min}, \theta_{\min}$ as before

Relevance estimation for document d: algorithm 2

On-demand rule generation

- $\mathcal{D}_d$: *projected training data* obtained after removing examples not applicable to d
- $\mathcal{R}_d$: generated from $\mathcal{D}_d$ using $\sigma_{\min}, \theta_{\min}$ as before

Example: document 11

Retrieved Documents				Relevance
id	PageRank	BM25	*tf*	
1	–	[0.36-0.55]	–	1
2	–	[0.36-0.55]	–	1
3	[0.51-0.64]	–	–	0
4	–	[0.36-0.55]	[0.28-0.45]	0
5	–	–	–	1
6	–	[0.36-0.55]	–	0
7	–	–	–	0
8	–	–	–	0
9	–	–	–	1
11	[0.51-0.64]	[0.36-0.55]	[0.28-0.45]	0

- $\mathcal{R}_d$ $(\sigma_{\min} = 0.2, \theta_{\min} = 0.67)$:

  BM25=[0.36,0.55] $\cap$ *tf*=[0.28,0.45] $\rightarrow r = 0$ $(\theta = 1.00)$

  PageRank=[0.51,0.64] $\rightarrow$ $r = 0$ $(\theta = 1.00)$
- $s(0) = (1.00 + 1.00)/2 = 1.00, s(1) = 0$
- *Score* $= 0 \times \frac{1.00}{1.00} + 1 \times \frac{0.00}{1.00} = 0.00$

Relevance estimation for document d: algorithm 3

- Include query terms as attributes
- Possible interpretations: (??)
  - query term attributes are Boolean

    e.g., d_{12} : Terms=*programs*, PageRank=[0.34-0.50], BM25=[0.22-0.35], *tf*=[0.46-0.61]
  - attributes like BM25, *tf* are specified on a per-term basis

    e.g.,
    d_{12} : PageRank=[0.34-0.50], BM25(*after*)=[. . .], BM25(*school*)=[. . .], BM25(*programs*)=[. . .], *tf*(*after*)=[. . .], . . .

Outline

AdaBoost

- Method for iteratively constructing *weak learning algorithms* $h_1, h_2, \ldots, h_T$ (moderately accurate classifiers with accuracy $>$ 50%) and combining them into highly accurate classifier
- Basic principle:
  - different input instances have differing difficulty levels
  - more importance should be attached to difficult input instances
  - during training, h_{t+1} attaches more importance to instances frequently misclassified by $h_1, h_2, \ldots, h_t$
  - importance modelled by probability distribution D over training instances

AdaBoost: example

Task: Document retrieval / classification

Input space: Documents $d_1, d_2, \ldots, d_m$

Output space: $Y = \{+1, -1\}$

Weak hypotheses:

1. If d_i contains "Spanish flu", output $\hat{y}_i = +1$.
2. If d_i contains "COVID-19", output $\hat{y}_i = -1$.

   ⋮

T. Standard Naive Bayes text classifier trained to classify documents as *relevant* / *non-relevant*

AdaBoost: example

Task: Document retrieval / classification

Input space: Documents $d_1, d_2, \ldots, d_m$

Output space: $Y = \{+1, -1\}$

Weak hypotheses:

1. If d_i contains "Spanish flu", output $\hat{y}_i = +1$.
2. If d_i contains "COVID-19", output $\hat{y}_i = -1$.

   ⋮

T. Standard Naive Bayes text classifier trained to classify documents as *relevant* / *non-relevant*

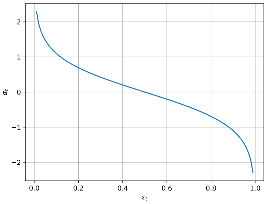
NB: For these weak classifiers, no training is required.

Inputs:

- Labelled training instances: $(x_1, y_1), \ldots, (x_m, y_m)$
- Number of iterations / rounds: T

Algorithm:

1. Initialise distribution over training instances: $D_1(i) = \frac{1}{m}, \quad (1 \leq i \leq m)$.
2. Repeat for $t = 1, 2, \ldots, T$:
   (a) Train a weak learner / construct a weak hypothesis h_t using D_t.
   (b) Compute expected error of h_t, $\varepsilon_t = \sum_{i:h_t(x_i) \neq y_i} D_t(i)$.
   (c) Compute weight of h_t: $\alpha_t \leftarrow \frac{1}{2} \ln \left(\frac{1 - \varepsilon_t}{\varepsilon_t} \right)$.



AdaBoost algorithm – slide II

(d) Update distribution

$$D_{t+1}(i) = \frac{D_t(i)\ \exp(-\alpha_t y_i h_t(x_i))}{\text{Sum-normalisation factor}}$$

$$= \frac{D_t(i)}{N_t} \times \begin{cases} e^{-\alpha_t} & \text{for correctly classified instances} \\ e^{\alpha_t} & \text{for incorrectly classified instances} \end{cases}$$

Output:

$$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

RankBoost

Reference: FREUND ET AL., JMLR 2003

- Combines *weak rankers* into single highly accurate "final" / "combined" ranker
- *Ranking feature* / *ranking function* / *scoring function* f_i:

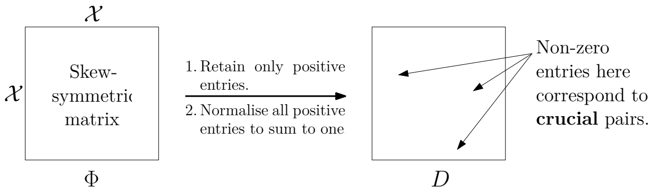
$$f_i \; : \; \mathcal{X} \to \mathbb{R} \cup \{\perp\} \text{ where } \mathcal{X} = \text{input space}$$

  - $f_i(x) > f_i(y) \;\Rightarrow\; x$ preferred to y
  - ties allowed
  - all instances **may not** be included in the ordering corresponding to a ranker
- *Ranker* $F \; : \; \mathcal{X} \to \mathbb{R}$

NB: Ranking feature vs. ranker: in ranker, ties are permitted, but no $\perp$

RankBoost: constructing the error / loss function

- Ground truth specified by function $\Phi : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$
  - $\Phi(x, y) > 0 \Rightarrow y$ should be ranked above x
  - $|\Phi(x, y)|$ specifies magnitude of preference
  - $\Phi(x, x) = 0 \qquad \forall x \in \mathcal{X}$
  - Φ : anti-symmetric (but not required to be transitive)
- $D : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ constructed from Φ as follows



- Ranking loss for ranker f

$$\sum_{(x,y)} D(x, y)\, I\left[f(y) \leq f(x)\right]$$

I – indicator function

RankBoost algorithm

Algorithm **RankBoost**
Given: initial distribution D over $\mathcal{X} \times \mathcal{X}$.
Initialize: $D_1 = D$.
For $t = 1, \ldots, T$:

- Train weak learner using distribution D_t.
- Get weak ranking $h_t : \mathcal{X} \to \mathbb{R}$.
- Choose $\alpha_t \in \mathbb{R}$.
- Update: $D_{t+1}(x_0, x_1) = \dfrac{D_t(x_0, x_1) \exp(\alpha_t (h_t(x_0) - h_t(x_1)))}{Z_t}$
  where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).

Output the final ranking: $H(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$

RankBoost algorithm

Algorithm **RankBoost**
Given: initial distribution D over $\mathcal{X} \times \mathcal{X}$.
Initialize: $D_1 = D$.
For $t = 1, \ldots, T$:

- Train weak learner using distribution D_t.
- $\boxed{\text{Get weak ranking } h_t : \mathcal{X} \to \mathbb{R}.}$
- $\boxed{\text{Choose } \alpha_t \in \mathbb{R}.}$
- Update: $D_{t+1}(x_0, x_1) = \dfrac{D_t(x_0, x_1) \exp(\alpha_t(h_t(x_0) - h_t(x_1)))}{Z_t}$
  where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).

Output the final ranking: $H(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$

Choosing α_t

Simplifying assumption: range of $h = \{0, 1\}$

Let $W_b = \sum_{(x,y)} D(x,y)\; I\big[h(x) - h(y) == b\big]$ where $b \in \{-1, 0, +1\}$

Choose $\alpha = \frac{1}{2} \ln \left(\frac{W_-}{W_+} \right)$

Weak rankings

First attempt

$$h(x) = \begin{cases} f_i(x) & \text{if } f_i(x) \in \mathbb{R} \\ q_{\text{def}} & \text{if } f_i(x) = \perp \end{cases}$$

Drawbacks:

- $h()$ depends on actual values assigned by ranking feature
- Problematic to combine ranking features with disparate scores

Weak rankings

Revised attempt

$$h(x) = \begin{cases} 1 & \text{if } f_i(x) > \theta \\ 0 & \text{if } f_i(x) \leq \theta \\ q_{\text{def}} & \text{if } f_i(x) = \perp \end{cases}$$

Best $h_t()$ may be obtained by brute force search over

- ranking features f_i
- threshold θ
- $q_{def} \in \{0, 1\}$

Outline

Listwise approaches: overview

- Input space: set of feature vectors corresponding to documents to be ranked for a particular query

  Output space: permutation / ordering of elements of input set
- Optimise loss functions *directly based on IR evaluation measures* instead of indirectly related loss functions (e.g., classification accuracy)

AdaRank

Reference: XU AND LI, SIGIR 2007

Inputs: Training data $S = \{\langle q_i, \mathbf{d}_i, \mathbf{y}_i \rangle\}_{i=1}^m$, T, evaluation measure E

Algorithm:

1. Initialise distribution over training queries: $D_1(i) = \frac{1}{m}, \quad (1 \leq i \leq m)$.
2. Repeat for $t = 1, 2, \dots, T$:
   - (a) Create *weak ranker* h_t using distribution D_t over training queries.
   - (b) Compute weight α_t of weak ranker h_t. $\longleftarrow$ depends on its “goodness”

$$\alpha_t = \frac{1}{2} \cdot \ln \frac{\sum_{i=1}^m D_t(i) \cdot \left[1 + E(\pi(q_i, \mathbf{d}_i, h_t), \mathbf{y}_i)\right]}{\sum_{i=1}^m D_t(i) \cdot \left[1 - E(\pi(q_i, \mathbf{d}_i, h_t), \mathbf{y}_i)\right]}$$

   - (c) Create *ranker* $\boxed{f_t() = \sum_{k=1}^t \alpha_k \cdot h_k().}$ $\longleftarrow$ final output at $t = T$
   - (d) Update distribution: $\longleftarrow$ emphasise hard-to-rank queries

$$D_{t+1}(i) = \frac{\exp\big(-E(\pi(q_i, \mathbf{d}_i, f_t), \mathbf{y}_i)\big)}{\sum_{j=1}^m \exp\big(-E(\pi(q_i, \mathbf{d}_i, f_t), \mathbf{y}_i)\big)}$$

AdaRank: features

- $\sum_{w \in q \cap d} \ln(1 + tf(w,d))$
- $\sum_{w \in q \cap d} \ln\left(1 + \frac{tf(w,d)}{|d|}\right)$
- $\sum_{w \in q \cap d} \ln(idf(w))$
- $\sum_{w \in q \cap d} \ln\left(1 + \frac{|C|}{cf(w,C)}\right)$
- $\sum_{w \in q \cap d} \ln\left(\frac{tf(w,d)}{|d|} \cdot idf(w) + 1\right)$
- $\sum_{w \in q \cap d} \ln\left(\frac{tf(w,d)|C|}{|d|\,cf(w,C)} + 1\right)$
- ln(BM25 score)

AdaRank: weak rankers

At each round, choose feature that maximises weighted performance.

$$\max_k \sum_{i=1}^{m} D_t(i) \cdot E\Big(\pi(q_i, \mathbf{d}_i, \boxed{x_k}), \mathbf{y}_i\Big)$$

AdaRank vs. RankBoost

- AdaRank can directly optimise any query-specific evaluation measure $\in [-1, +1]$.
- AdaRank focuses specially on top-ranked documents when used with MAP, NDCG, etc.
- AdaRank not biased against queries for which fewer documents retrieved.

Outline

Test collections

- *GOV2* / *Terabyte*
  - 2004 crawl of `.gov` domain
  - ~25 million documents (426 GB)
  - queries from *million query track*: MQ2007 (1700 queries), MQ2008 (800 queries)
- *OHSUMED*
  - subset of MEDLINE (corpus of medical scholarly publications)
  - 348,566 records from 270 medical journals (1987–1991)
  - title, abstract, MeSH indexing terms, etc.
  - 106 medical search queries
  - 16,140 relevance assessment pairs
- 3-level judgments
- each dataset partitioned into 5 parts with approx. equal number of queries

Test collections

Folds	Training set	Validation set	Test set
Fold1	{S1, S2, S3}	S4	S5
Fold2	{S2, S3, S4}	S5	S1
Fold3	{S3, S4, S5}	S1	S2
Fold4	{S4, S5, S1}	S2	S3
Fold5	{S5, S1, S2}	S3	S4

http://www.microsoft.com/en-us/research/project/letor-learning-rank-information-retrieval/

References

1. *Learning to rank for information retrieval*, Tie-Yan Liu.
   FTIR, 3(3), pages 225–331, 2009.
2. *Learning to Rank for Information Retrieval*, Tie-Yan Liu.
   Springer, 2011.