# *An Efficient Boosting Algorithm for Combining Preferences*

by Y.Freund, R.Iyer, R.E.Schapire, Y.Singer

Presented by: Aivi Kaljuvee

RankBoost – boosting algorithm for combining multiple ranking functions to predict the target function

Boosting – method of producing highly accurate prediction rules by combining many “weak” rules which may be only moderately accurate

Example problems:

• movie recommendation service – based on user's ratings to movies h/she has already seen and ratings from other service users, h/she is presented a list of new movies he/she should like (collaborative filtering task)

• meta-search – results form several search engine queries are combined into one

## Formal framework

- $X$ – *instance space*
- $f_i : X \to \bar{\mathbb{R}},\ i = 1..\mathrm{n},\ \bar{\mathbb{R}} = \mathbb{R} \cup \{\iota\}$ – n *ranking features*
  $f_i(x) = \iota$ – no ranking given to x by $f_i$
  $f_i(x_1) > f_i(x_0)$ – $x_1$ is preferred over $x_0$ by $f_i$
  ties are allowed
- $H : X \to \mathbb{R}$ – *combined ranking*

## Formal framework

- $\Phi : X \times X \rightarrow \mathbb{R}$ – *feedback function*
  $\Phi(x_{0,} x_1) > 0$ – $x_1$ is preferred over $x_0$
  $\Phi(x_{0,} x_1) < 0$ – the opposite
  $\Phi(x_{0,} x_1) = 0$ – no preference
  $|\Phi(x_{0,} x_1)|$ – importance of preference
- $D(x_{0,} x_1) = c * max\{0, \Phi(x_{0,} x_1)\}$
  c is a constant so that $\sum x_{0,} x_1 D(x_{0,} x_1) = 1$
- $rloss_D(H) = \sum x_{0,} x_1 D(x_{0,} x_1) [\![ H(x_1) \leqslant H(x_0) ]\!]$
  loss function ( $[\![ \pi ]\!]$ – 1 if predicate $\pi$ holds, 0 otherwise)

## Pseudocode for RankBoost

Given: initial distribution $D$ over $X$ x $X$.

Initialize: $D_1 = D$.

For t = 1,..,T:

- Train weak learner using distribution $D_t$.
- Get weak ranking: $h_t : X \rightarrow \mathbb{R}$
- 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 (so that $D_{t+1}$ is a distribution)

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

## Finding α

- $rloss_D(H) \leqslant \prod_{t=1..\mathrm{T}} Z_t$

- $Z_t = \sum_{x_{0,}x_1} (D_t(x_{0,} x_1) \exp(\alpha_t (h_t(x_0) - h_t(x_1))))$

Finding a weak ranking

- $$h(x)=\begin{cases} f_i(x) & \text{if } f_i(x)\in\mathbb{R} \\ q_{default} & \text{if } f_i(x)=\iota \end{cases}, \quad (q_{default}\in\mathbb{R})$$
  based on actual values from ranking features, not used
- $$h(x)=\begin{cases} 1 & \text{if } f_i(x)>\theta \\ 0 & \text{if } f_i(x)\leqslant\theta \\ q_{default} & \text{if } f_i(x)=\iota \end{cases}, \quad (q_{default}, \theta\in\mathbb{R})$$
  uses relative-ordering information, used for producing weak rankings

## Experiments with the meta-search task

- queries for finding homepages for machine learning researchers and universities
- base query and extended query
- instances are pairs of base queries and URLs
- a ranking feature is the ordered list received from a request with one of the extended queries

## Results for the meta-search task

| ML Domain | Top 1 | Top 2 | Top 5 | Top 10 | Top 20 | Top 30 | Avg Rank |
|---|---|---|---|---|---|---|---|
| RankBoost | 102 | 144 | 173 | 184 | 194 | 202 | 4.38 |
| Best (Top 1) | **117** | 137 | 154 | 167 | 177 | 181 | 6.80 |
| Best (Top 10) | 112 | **147** | **172** | **179** | 185 | 187 | **5.33** |
| Best (Top 30) | 95 | 129 | 159 | 178 | **187** | **191** | 5.68 |
| University Domain | | | | | | | |
| RankBoost | 95 | 141 | 197 | 215 | 247 | 263 | 7.74 |
| Best single query | 112 | 144 | 198 | 221 | 238 | 247 | 8.17 |

Experiments with movie recommendation service

- users' ratings are ranking features, ratings from a single target user are used to construct the feedback function (half of the films rated for training, half for testing)
- compared with the following algorithms:
  - regression
  - nearest-neighbour
  - vector similarity

Performance measures:

- disagreement
  c – ordering of test movies
  N – number of test movies

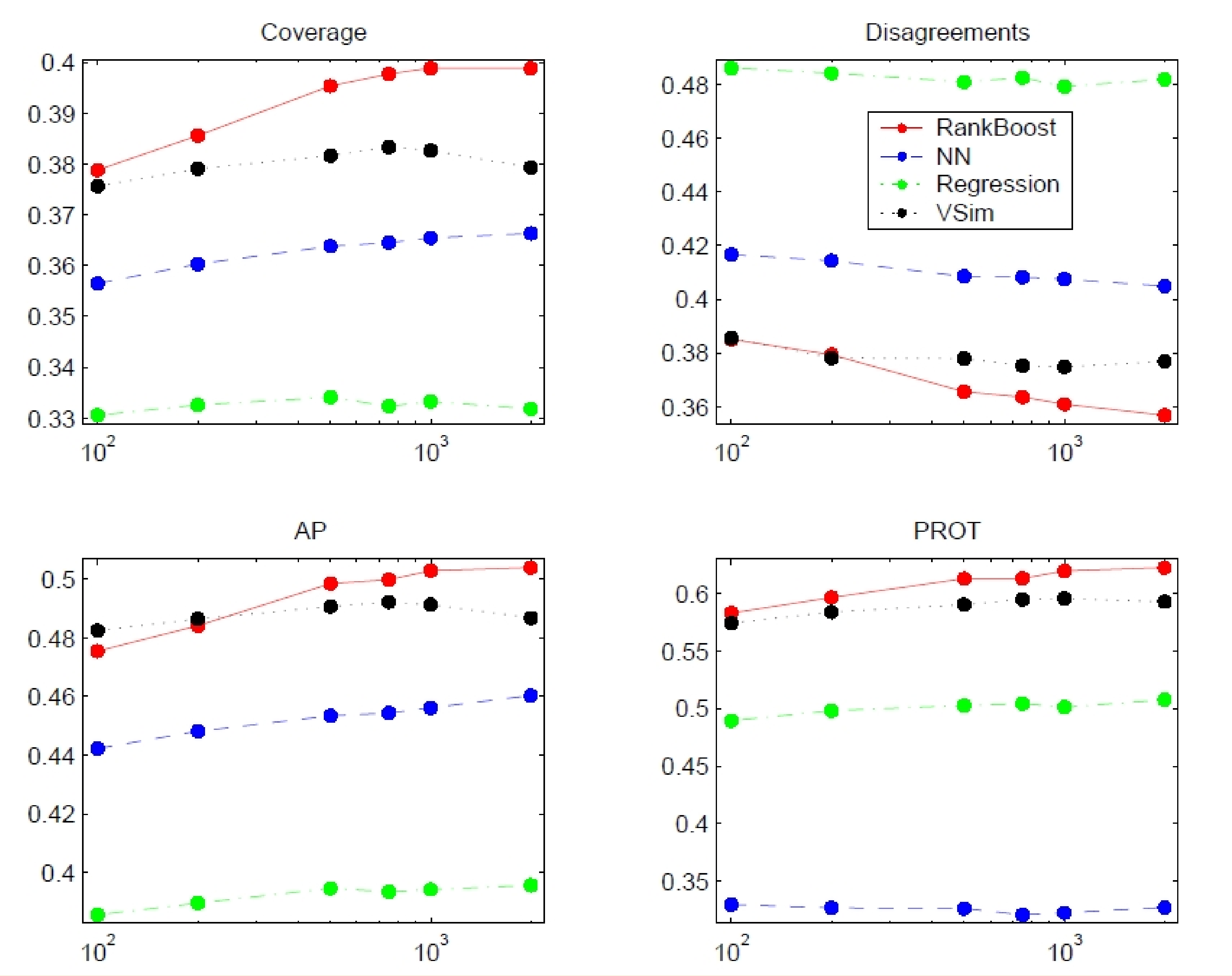
$$\frac{1}{N} \sum_{x_0, x_1 : c(x_0) < c(x_1)} \llbracket H(x_0) > H(x_1) \rrbracket$$

- average precision
  K – number of movies in
  the feedback ordering
  tk – movie
  k – its rank in the feedback ordering
  rank(tk) – movie's rank in the learned ranking

$$\frac{1}{K} \sum_{k=1..\mathrm{K}} \left( \frac{k}{(rank(t_k))} \right)$$

Performance measures:

- predicted rank of top $\dfrac{1}{(rank(t_1))}$
- coverage $\dfrac{1}{(rank(t_K))}$



Coverage
Disagreements
AP
PROT
RankBoost
NN
Regression
VSim

## Directions for future work

- reimplement algorithms used for comparison using relative-ordering information
- compare performance with AdaBoost
- apply RankBoost to information retrieval problems
- apply RankBoost to language processing tasks