Learning to select for a predefined ranking

Aleksei Ustimenko
Alexander Vorobev
Gleb Gusev
Pavel Serdyukov
From ranking to sorting

• Search engines typically order the items by some relevance score obtained from a ranker before presenting the items to the user

• Yet, online shops and social networks allow the user to rearrange the items using some dedicated attribute (e.g. price or time)
From ranking

To sorting
Threshold relevance?

• It was proven that filtering with a constant threshold for relevance is **suboptimal** (in terms of ranking quality metrics like DCG)

• The optimal algorithm was suggested by (Spirin et. al at SIGIR 2015), but it has quadratic complexity $O(n^2)$, where $n$ – is the list size

• Such algorithms are infeasible for search engines, we need to predict if to filter an item by **just using item features (locally), not the entire list (globally)**
LSO Problem Formulation

- We define a selection algorithm as \( F \) and the result of its application to a list \( L \) to be the selected \( L^F \).

- \( L^F \) - the same ordered list as \( L \), but with some items filtered.

- We formulate the problem of LSO as learning from \( D \) a selection algorithm \( F \) that maximizes the expected ranking quality \( Q \) of \( L^F \), where \( L \) is sampled from some \( P \):

\[
F^* = \arg \max \mathbb{E}_{L \sim P} Q(L^F)
\]
Optimal Selection Predictor

• First, we suggest to build a model $M$ that predicts the binary decision of the infeasible optimal algorithm

• Then we train a binary classifier $M$ on the training examples obtained from that algorithm $\{(x_{ij}, Opt_{ij})\}_{i: L_i \in D, j = 1..n_i}$ by minimizing logistic loss

• However, the logistic loss of such a classifier is still not directly related to ranking quality $Q$, i.e. it is not a listwise learning-to-rank algorithm
Direct Optimization of the Objective

• For a document $d$ with features vector $x_d \in \mathbb{R}^l$ we define probabilistic filtering rule by:

$$P(F(d) = 1) = \sigma(f(x_d)) = \frac{1}{1 + \exp(-f(x_d))}$$

• Assume that decisions $F(d)$ for different $d$ are independent. Denote the space of all so-defined stochastic selection algorithms by $\mathcal{F}$.

• We transform $Q$ to the $Q_{smooth}(F, L) = \mathbb{E}_{Z \sim P_F} Q(L_Z)$

• And the problem to:

$$F^* = \arg\max_{F \in \mathcal{F}} \mathbb{E}_{L \sim D} Q_{smooth}(F, L)$$
Policy Gradient Approach

• For i.i.d. samples of binary decisions $Z_1, \ldots, Z_s \sim P_F$ define the estimate (after applying the log derivative trick):

$$\frac{\partial Q_{smooth}(F, L)}{\partial f(x_j)} \approx \frac{1}{s} \sum_{i=1,s} (Q(L_{Z_i}) - b)(-p_j)^{Z_{ij}} (1 - p_j)^{1-Z_{ij}}$$

where baseline $b := Q(L_{Z_{F,0.5}}^{0.5})$ with $z_{F,k}^{0.5} = 1\{p_k > 0.5\}$

• And we use this functional gradient directly in the Gradient Boosted Decision Trees learning algorithm (with CatBoost implementation)
Pre-training

After training OSP model, we use it as a starting point for our approach.

Thus, we avoid getting stuck in local maxima.
Step by our poster #228

Learning to select for a predefined ranking

From Ranking to Sorting
Search engines typically order the items by some relevance score obtained from a ranker before presenting the items to user.

In this example, the user sees relevant results after ranking, but they become completely irrelevant after sorting by price.

Thresholding by relevance score is sub-optimal

[Sprin et al., SIGIR 2015] proposed an optimal algorithm, but it has quadratic time complexity O(n^2): infeasible for modern search engines.

LSO Problem Formulation
Consider a i.i.d. sample of lists $D = \{D_1, \ldots, D_P\}$ from a set of ordered lists $L = \{L_1, \ldots, L_n\}$ with $m$ items $d_k = (x_k, r_k)$ in $\mathbb{R} \times \mathbb{R}$.

An item $d = (x, r)$ corresponds to a context-item pair, represented by $x = (x', x'')$ of its $F$ features and assigned a relevance $r$ (unknown to the system). Assume that the items in each list $L_i$ are ordered by one of the features.

We define a selection algorithm as $F : \mathbb{R} \rightarrow \{0, 1\}$ and the result of its application to a list $L_i$ to be the selected list $L_i' = (d_{i_1}, \ldots, d_{i_{K_i}})$, where $i_1 < \ldots < i_{K_i}$ and $|L_i - \{i_1, \ldots, i_{K_i}\}| \leq K_i$.

We formulate the problem of learning to select with order (LSO) as learning from $D$ a selection algorithm $F$ that maximizes the expected ranking quality $Q$ of $L'$, where $L$ is sampled from $P$:

$$Q = \text{arg max} \ E_q[Q(L')]$$

Optimal Selection Prediction (OSP)
First, we suggest to build a model $M$ that for each item predicts the binary decision of the optimal algorithm by [Sprin et al., 2015], that is if the algorithm decided not to filter out an item.

Then we train a binary classifier $M$ on the training examples $\{(x_k, r_k), y_k\}_{k \in D}$ by minimizing logistic loss. After it, we define $y_k$ on basis of $M$ as $P(x) = \frac{1}{1 + \exp(-Q(x))}$ where $t$ is a constant hyperparameter of $P$.

However, the logistic loss is not directly related to ranking quality $Q$, i.e. it is not a safe learning-to-ranking algorithm.

Direct Optimization of the Objective
For a document $d$ with features vector $x_d \in \mathbb{R}^F$ we define probabilistic filtering rule by:

$$P(F(d) = 1) = \sigma(Q(x_d)) = \frac{1}{1 + \exp(-Q(x_d))}$$

where $\sigma$ denotes the sigmoid function.

Assume that decisions $F(d)$ for different $d$ are independent.

For the major independent evaluation of the result page relevance, we collected human relevance judgements of 5 grades (from 0 to 4)

for top 10 results of each selected list produced by the algorithms trained on train and evaluated on DCG@10, p@10, $\text{top}_{@12}$.

Finally, most representative algorithms were compared in online experiments. For evaluation we used Abandonment, MRR and $CTR@12$.

Learning Algorithm
For an ML algorithm for all the approaches, we chose GBDT as the state-of-the-art method for many practical tasks including the learning-to-rank problem in web search and click prediction.

We use GBDT implementation in the open-sourced CatBoost Python package.

Experimental Results
We pick DCG-RR($y_1, \ldots, y_5$) = $\sum \log_{p} y_{i}$ as lists quality measure. For the major independent evaluation of the result page relevance, we collected human relevance judgements of 5 grades (from 0 to 4)

for top 10 results of each selected list produced by the algorithms trained on train and evaluated on DCG@10, p@10, $\text{top}_{@12}$.

For our approach, we chose GBDT as the state-of-the-art method for many practical tasks including the learning-to-rank problem in web search and click prediction.

We use GBDT implementation in the open-sourced CatBoost Python package.

Table 2. Performance, absolute for WEAKCatoff/f and relative $\Delta$ to WEAKCatoff/f, % for other

<table>
<thead>
<tr>
<th>Approach</th>
<th>DCG@10</th>
<th>p@10</th>
<th>$\text{top}_{@12}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>WEAKCatoff/f</td>
<td>0.52</td>
<td>1.07</td>
<td>0.03</td>
</tr>
<tr>
<td>Catoff/f</td>
<td>0.05%</td>
<td>2.9%</td>
<td>1.6%</td>
</tr>
<tr>
<td>QueryCatoff/f</td>
<td>0.56%</td>
<td>6.3%</td>
<td>4.3%</td>
</tr>
<tr>
<td>USP</td>
<td>3.86%</td>
<td>20.3%</td>
<td>10.7%</td>
</tr>
<tr>
<td>OSN + LBO</td>
<td>4.17%</td>
<td>22.4%</td>
<td>12.0%</td>
</tr>
<tr>
<td>OSN + LG</td>
<td>4.33%</td>
<td>22.4%</td>
<td>12.2%</td>
</tr>
<tr>
<td>Oracle</td>
<td>34.44%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3. Online performance, relative $\Delta$ to WEAKCatoff/f, %

<table>
<thead>
<tr>
<th>Approach</th>
<th>WEAKCatoff/f</th>
<th>Abandonment</th>
<th>Abandonment</th>
<th>Catoff/f</th>
<th>QueryCatoff/f</th>
<th>USP</th>
<th>OSN + LBO</th>
<th>OSN + LG</th>
<th>Oracle</th>
</tr>
</thead>
<tbody>
<tr>
<td>WEAKCatoff/f</td>
<td>100%</td>
<td>-1.0%</td>
<td>-1.0%</td>
<td>-1.0%</td>
<td>-1.0%</td>
<td>-1.0%</td>
<td>-1.0%</td>
<td>-1.0%</td>
<td>-1.0%</td>
</tr>
<tr>
<td>Catoff/f</td>
<td>-1.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
</tr>
<tr>
<td>QueryCatoff/f</td>
<td>-1.0%</td>
<td>-1.0%</td>
<td>-1.0%</td>
<td>-1.0%</td>
<td>-1.0%</td>
<td>-1.0%</td>
<td>-1.0%</td>
<td>-1.0%</td>
<td>-1.0%</td>
</tr>
<tr>
<td>USP</td>
<td>-1.0%</td>
<td>-1.0%</td>
<td>-1.0%</td>
<td>-1.0%</td>
<td>-1.0%</td>
<td>-1.0%</td>
<td>-1.0%</td>
<td>-1.0%</td>
<td>-1.0%</td>
</tr>
<tr>
<td>OSN + LBO</td>
<td>-1.0%</td>
<td>-1.0%</td>
<td>-1.0%</td>
<td>-1.0%</td>
<td>-1.0%</td>
<td>-1.0%</td>
<td>-1.0%</td>
<td>-1.0%</td>
<td>-1.0%</td>
</tr>
<tr>
<td>OSN + LG</td>
<td>-1.0%</td>
<td>-1.0%</td>
<td>-1.0%</td>
<td>-1.0%</td>
<td>-1.0%</td>
<td>-1.0%</td>
<td>-1.0%</td>
<td>-1.0%</td>
<td>-1.0%</td>
</tr>
<tr>
<td>Oracle</td>
<td>-1.0%</td>
<td>-1.0%</td>
<td>-1.0%</td>
<td>-1.0%</td>
<td>-1.0%</td>
<td>-1.0%</td>
<td>-1.0%</td>
<td>-1.0%</td>
<td>-1.0%</td>
</tr>
</tbody>
</table>

WeakCatoff/f, “take all documents”,

Catoff/f, “filtering by global thresholding of relevance prediction”,

QueryCatoff/f, “query-wise threshold prediction + filtering by thresholding of relevance prediction”

LBO – another proposed algorithm optimizing the lower bound of $Q$ (see its description in the paper)