Learning to Select for a Predefined Ranking

Aleksei Ustimenko [*1 2 3] **Aleksandr Vorobev** [*1] **Gleb Gusev** [1 4] **Pavel Serdyukov** [1]

Abstract

In this paper, we formulate a novel problem of learning to select a set of items maximizing the quality of their ordered list, where the order is *predefined* by some explicit rule. Unlike the classic information retrieval problem, in our setting, the predefined order of items in the list may not correspond to their quality in general. For example, this is a dominant scenario in personalized news and social media feeds, where items are ordered by publication time in a user interface. We propose new theoretically grounded algorithms based on direct optimization of the resulting list quality. Our offline and online experiments with a large-scale product search engine demonstrate the overwhelming advantage of our methods over the baselines in terms of all key quality metrics.

1. Introduction

Modern search systems traditionally face the problem of providing users with ordered lists of items best responding their needs in different contexts. One particularly important case is serving a user with a *list* of relevant items, yet ordered not by their relevance, but by a *dedicated* item attribute (often chosen by the user after formulating the initial query). Examples of such scenarios are time-based ordering on a social media website (Hasanain et al., 2015) or price-based ordering of offers on an e-commerce website (e.g., Amazon [1]). In that case, the algorithmic challenge essentially comes down to *selecting* items to include in the list to be shown to the user.

[*]Equal contribution [1]Yandex, Moscow, Russia [2]Skoltech University, Moscow, Russia [3]Faculty of Computer Science, Higher School of Economics, Moscow, Russia [4]Department of Innovation and High Technology, Moscow Institute of Physics and Technology, Dolgoprudny, Russia. Correspondence to: Aleksei Ustimenko <austimenko@yandex-team.ru>, Aleksandr Vorobev <alvorobyev88@gmail.com>, Gleb Gusev <gleb57@gmail.com>, Pavel Serdyukov <pavser@yandex-team.ru>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

[1]https://www.amazon.com

It is generally a poor choice to select all retrieved items, since the top ones by the dedicated attribute can be totally irrelevant to the user needs: e.g., the cheapest offer is often by no means the most relevant to the user query. In practice, selection algorithms are often far from perfect, and numerous users of, e.g., popular product or job search engines regularly suffer from poor quality of results ordered by price or publication time. For example, see (Spirin et al., 2015, Section 1) for illustrative statistics and Figure 1 in the supplementary material for a result page at one of the most popular product search system.

Despite a large body of learning-to-rank literature, surprisingly, there are no papers about learning to select for a predefined ranking. One straightforward selection method referred to as the *cutoff* approach below is (1) to learn a relevance prediction model, (2) to select only items whose predicted relevance is higher than some threshold (Hasanain et al., 2015). As it was demonstrated by Spirin et al. (2015), this approach is too far from the optimal solution in general. Indeed, on one hand, we want to show the most relevant items at the top of the ordered list; therefore, we should exclude all items above, even if they are slightly less relevant. On the other hand, a threshold suitable for that purpose will also hide many relevant results ranked below from the user. For example, for the list of items (d_1, d_2, d_3) (ordered by a dedicated attribute) with respective relevance values $(2, 7, 1)$, the maximum Discounted Cumulative Gain[2] (Järvelin & Kekäläinen, 2002) of top-3 results (DCG@3) is achieved on the selected set (d_2, d_3), where d_3 is selected and d_1 is not, though having the higher relevance (see Table 1 for details). At the same time, the cutoff approach with the best relevance threshold selects only d_2, and it is much closer to the original list than to the best solution in terms of DCG@3.

Table 1. **Example of suboptimality of the cutoff approach**

	Original list	**Best cutoff**	**Best selection**
List	d_1, d_2, d_3	d_2	d_2, d_3
Attribute values	$1, 2, 3$	2	$2, 3$
Relevance values	$2, 7, 1$	7	$7, 1$
DCG@3 of list	6.92	7	7.63

[2]$DCG@k(r_1, \dots, r_k) = \sum_{i=1}^{k} \frac{r_i}{\log_2(i+1)}$, r_i - relevance of the item at position i

If we assume that relevances are known, the selection task becomes a nice combinatorial problem, which has an exact solution with quadratic time complexity in the number of candidate items (Spirin et al., 2015). Unfortunately, this solution is inapplicable in practice for highly-loaded systems due to its high time complexity and its incompatibility with distributed system architectures of large-scale search engines (see Section 2.1 for details).

In this paper, we formulate the selection task as a machine learning problem, where we need to train a *selection model* F, which decides for each item d whether to select it ($F(d) = 1$) or not ($F(d) = 0$). Unlike usual classification tasks, the objective is to maximize a given relevance-based *ranking quality measure* of the selected set of items ordered in a predefined way. We call this formulation *learning to select with order* (LSO) problem below, by analogy with learning to rank (LTR) one. To solve this problem, we propose practically feasible and effective selection algorithms.

Overall, **the contribution of this paper** is four-fold:
(1) We formulate a learning-to-select-with-order problem (LSO), a novel machine learning task with a specific objective function.
(2) We propose two approaches to LSO, which include smoothing the objective function and two methods for gradient estimation: policy gradient and utilization of a lower bound of the objective, see Section 4.2.
(3) To avoid local extremes of the non-convex smoothed objective, we combine its optimization with optimization of its convex approximation, see Section 4.3.
(4) In offline and online experiments with a large-scale e-commerce search-engine (see Section 6), our approaches with gradient boosted decision trees (GBDT) remarkably outperform the existing solutions to the LSO problem.

2. Problem Setting

2.1. Local vs Global Selection Algorithms

First, we describe the state-of-the-art system architecture of modern search engines, which basically determines our problem setting for the selection task described in Section 1. Large commercial search systems with billions of items in their databases and millions of users served daily typically rely on distributed query processing. That means running several search engines in parallel on separate *buckets* of candidate items (the first ranking stage) and aggregating their top ranked items into the final search result page at a meta-search engine (the second ranking stage) (Glover et al., 1999; Patel & Shah, 2011; Wang et al., 2011).

In the selection task, the top ranked items are chosen by the dedicated attribute. Therefore, it is critical to apply selection at the first stage before choosing the top ranked items: otherwise, the meta-search engine will receive mostly irrelevant items from each first stage engine. At the meta-search level, since having less candidate items, by analogy with hierarchical web search ranking algorithms (Glover et al., 1999; Wang et al., 2011), one can apply an additional selection based on more complex features.

At the same time, since only a subset of all the candidate items is available at each first stage search engine where selection must take place, a selection algorithm has no access to aggregated information about all the candidate items. It means that any *global* selection algorithms (by analogy to global ranking approaches (Ji et al., 2009)) that need the entire list of the candidate items to make the decision on one of them, are not applicable. In particular, we cannot apply the quadratic-time optimization method from Spirin et al. (2015), see more details in Section 3.

Our setting is to learn a *local* selection algorithm which makes decisions on each item in parallel with no direct knowledge of the other items. Such local approach allows to minimize latency of the system response, what is critical for user satisfaction and, eventually, for search system revenue (Schurman & Brutlag, 2009). Thereby, state-of-the-art LTR models are local (Liu et al., 2009a).

2.2. LSO Problem Formulation

Consider a set $D = \{L_i\}_{i=1}^m$ of ordered sets (referred to as *lists* below). Each list $L_i = (d_{i,1}, \dots, d_{i,n_i})$ consists of n_i items $d_{i,j} = (x_{i,j}, r_{i,j}) \in \mathbb{R}^l \times \mathbb{R}$. In terms of practice, list L_i corresponds to the set of candidate *items* retrieved in a particular context of user-system interaction; an item $d = (x, r)$ corresponds to a context-item pair represented by the vector $x = (x^0, \dots, x^l)$ of its l features and assigned to a relevance value r unknown to the system; the items in L_i are distributed among multiple first stage search engines. For convenience of notations, we assume that the items of each set L_i are ordered by increasing (w.l.o.g.) a dedicated attribute x^0 (i.e., $x^0_{i,1} \le \dots \le x^0_{i,n_i}$), which may correspond to publication time for news and social media feeds or to price for product search engine. For popular contexts, the number of items n_i may achieve millions. Lists L_i from D are i.i.d. samples from some distribution $\mathbb{P}$ on the space $\mathcal{L} := \coprod_{n=1}^{\infty} (\mathbb{R}^l \times \mathbb{R})^n$ of lists of any length. Besides, we are provided with some *objective measure* Q on lists, which corresponds to a ranking quality measure in practice, e.g., DCG, Expected Reciprocal Rank (ERR) (Chapelle et al., 2009), Rank-Biased Precision (RBP) (Moffat & Zobel, 2008). Its value is determined by only the vector of item relevance values: $Q((x_1, r_1), \dots, (x_n, r_n)) = q_n(r_1, \dots, r_n)$ for some $q_n \colon \mathbb{R}^n \to \mathbb{R}$ which is naturally required to be non-decreasing w.r.t. each argument r_i [3].

[3] Note that r_i is relevance to the user needs, which, in particular, may include a user intent to see results with better values of x^0_i. Therefore, r_i may correlate with x^0_i to some extent.

We define a *selection algorithm* as a feature-based binary decision rule $F : \mathbb{R}^l \to \{0, 1\}$ and define the result of its application to a list L_i to be the *selected* list $L_i^F := (d_{i_1}, \ldots, d_{i_k})$, where $i_1 < \ldots < i_k$ and $\{i_1, \ldots, i_k\} = \{i \in [1, n] : F(x_i) = 1\}$. 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 of selected list L^F, where the original list L is sampled from $\mathbb{P}$:

$$F^* = \arg\max_F \mathbb{E}_{L\sim\mathbb{P}}\, Q(L^F) \tag{1}$$

2.3. Analysis

In the usual LTR problem with most ranking quality measures, the optimal list is the list of items ordered by decreasing relevances. In LSO problem, the structure of the optimal solution is not so simple (see the example from Introduction). Here we provide some intuition on its properties.

In Section 1 of the supplementary material, we formulate some natural requirements for an objective measure Q and prove the following proposition under them:

Proposition 1 *The optimal selection for a set $L = (d_1, \ldots, d_n)$ has the form $F(d_i) = \mathbb{1}\{r_i > t(i)\}$, where $t(\cdot)$ is a **non-increasing** threshold function.*

The following corollary can be immediately derived:

Corollary 1 *If some item d_i is selected, all items d_j with lower positions (i.e. $x_j^0 > x_i^0$) that have at least the same relevance ($r_j \geq r_i$) should also be selected.*

Informally, the optimal selection algorithm should be generally more selective at the top of the list than at the bottom.

3. Related Work

To the best of our knowledge, the only studied selection algorithm applicable within large-scale systems is the *cutoff* approach that selects either top-k items by predicted relevance or items with a predicted relevance higher than a threshold b. This approach has linear time complexity and is widely used in different application tasks, including microblog retrieval (Hasanain et al., 2015), static index pruning (Carmel et al., 2001) or pruning items in a retrieval process (Wang et al., 2011). The cutoff approach can often be strengthened by tuning it specifically for each context (e.g., a user query). This is commonly performed by utilizing context features to predict the optimal value of either k (*rank cutoff*) (Hasanain et al., 2015; Wang et al., 2011) or b (*score cutoff*) (Carmel et al., 2001; Wang et al., 2011). According to our analysis in Section 2.3, even the best cutoff threshold is too crude: its decision depends only on relevance r_i of the item d_i and does not take into account its position i (see Proposition 1 and the example in Section 1).

In the assumption that relevances are known, the LSO problem can be seen as a combinatorial optimization task.[4] In the case of additive objective measure Q, Spirin et al. (2015) proposed an exact solving algorithm to this task, with time complexity $O(n^2)$ where n is the number of candidate items. We refer to this algorithm as *Exact Quadratic Complexity Algorithm (EQA)*. In a dynamic programming fashion, it goes from top to bottom over a list $L = (d_1, \ldots, d_n)$ and obtains, for each $i, j \in [1, n]$, the best sublist $L_{i,j}$ of length j of the list $(d_1, \ldots, d_i)$ of length i. The list $L_{i,j}$ is obtained as the best one among $L_{i-1,j}$ and $L_{i-1,j-1} \frown (d_i)$ ("$\frown$" means concatenation). For the practical setting of unknown relevances, Spirin et al. (2015) applied EQA to predicted relevances learned using MSE loss. This approach is global and, moreover, has quadratic complexity in the number of items, what makes it inapplicable within a distributed system architecture (see Section 2.1). In the setting of global selection problem this approach is still not necessarily the best possible solution, because it is not end-to-end. In particular, the choice of the loss function (MSE in Spirin et al. (2015)) could be adapted to the listwise objective function Q. In LTR, listwise algorithms significantly outperform pointwise ones (Liu et al., 2009a). In this paper, we propose a direct optimization approach to the local LSO problem.

4. Approach

In the below-described approaches, we propose several loss functions designed for LSO problem. These losses can be used in combination with any gradient-based machine learning algorithm including gradient boosting and any parametric model (neural networks, linear models, and others). For this section, it is important that such algorithms rely on calculating the gradient of the loss function w.r.t. predictions of the currently built model on training examples.

4.1. Prediction of the Optimal Decision

First, we suggest to build a model M that predicts, based on item features, the decision of EQA on this item (see Section 3). For this purpose, we can apply EQA offline to the item sets of a training dataset D with known relevance labels $r_{i,j}$ (recall that running EQA online is impossible). As a result, we obtain the optimal binary decision $Opt_{i,j}$ for each item $d_{i,j}$. Then we train a binary classifier M on the training examples $\{(x_{i,j}, Opt_{i,j})\}_{i:\, L_i\in D,\, j=1..n_i}$ to be a LogitBoost model (Friedman et al., 2000) (state-of-the-art classification model) by maximizing log-likelihood (i.e., minimizing logistic loss).

Having the model M learned, we define the selection algorithm F_M on its basis as $F_M(x) = \mathbb{1}\{M(x) > t_{\text{const}}^{\text{rel}}\}$, where $M(x)$ is the model prediction for the feature vector

[4] As LTR with known relevances reduces to a sorting task.

x, the threshold $t^{\text{rel}}_{\text{const}}$ is a constant hyperparameter of F_M, and $\mathbb{1}\{A\}$ denotes the indicator of the event A.

In comparison with the cutoff approach with a predicted threshold described in Section 3, this approach, referred further as OSP (Optimal Selection Predictor), uses the optimal selection as a target, instead of the best cutoff selection. Thus, we expect its performance in terms of the objective $\mathbb{E}_{L\sim\mathbb{P}}\, Q(L^F)$ of LSO problem (1) to be higher than one of the cutoff approach. However, the loss function optimized by OSP (namely, logistic loss) is still not directly related to the objective $\mathbb{E}_{L\sim\mathbb{P}}\, Q(L^F)$. In particular, the value of the loss on an item does not depend on the item position in the selected list (while depends only on the binary decision of EQA and on the probability predicted by OSP). At the same time, the impact of an error on the objective measure decreases with i (e.g., as $\frac{1}{\log(i+1)}$ in the case of DCG). This effect resembles the weak points of pointwise approach to LTR (Liu et al., 2009b, Section 2.5.2).

4.2. Direct Optimization of the Objective Measure

Smoothing the objective measure Motivated by the above-described drawback of the cutoff and OSP approaches, we propose to construct selection algorithms F by direct optimization of the objective $\mathbb{E}_{L\sim\mathbb{P}}\, Q(L^F)$. To apply gradient-based optimization methods, we first need to allow scores $F(x)$ to take any values in some continuous space, while $F(x) \in \{0, 1\}$ in Equation 1. A straightforward solution is to define the selected list L^F on the basis of such scores $M(x)$ by a *threshold-based selection rule*:

$$F(x) = \mathbb{1}\{M(x) > t\} \tag{2}$$

for some $t \in \mathbb{R}$, i.e., items with scores above the threshold t are selected (as it was proposed in the OSP algorithm). However, in this case, the objective measure Q is a discontinuous function of the scores $M(x)$, while we need it to be differentiable. The same problem for ranking quality measures in LTR was addressed in previous papers (Chapelle & Wu, 2010; Wu et al., 2009; Valizadegan et al., 2009). But, though a given ranking quality measure (e.g., DCG) provides one function of **relevances** of the items of the resulting list, it corresponds to different functions of the vector of candidate item **scores** in LTR and LSO problems. Indeed, the resulting list of items and, thus, of their relevances, is formed on the basis of that vector differently: by ranking by item scores for LTR and by threshold-based selection for LSO. So, a direct application of the proposed smoothing methods to LSO is impossible.

Note that a selection algorithm $F(x)$ is a binary classifier, which decides, upon item features x, whether to select the item out. This encourages us to adapt to our problem the smoothing technique being state-of-the-art for the binary classification problem and used as a component of the LogitBoost and logistic regression models (Chapelle et al., 2015). Namely, we assume the decision $F(d)$ on an item d with features x_d to be a Bernoulli random variable with sigmoid dependence of the parameter on the score function $f : \mathbb{R}^l \to \mathbb{R}$ associated with the selection algorithm F:

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

Besides, we naturally assume that decisions $F(d)$ for different items d are generated independently. We denote the space of so-defined stochastic selection algorithms by $\mathcal{F}$.

Then, for a selection algorithm $F \in \mathcal{F}$ and a list $L = (d_1, \ldots, d_n)$ of items $d_i = (x_i, r_i)$, the selected list $L_Z := \{d_i : Z_i = 1, i = 1, \ldots, n\}$ is random with the following distribution P_F of the random vector of selection decisions $Z = (Z_1, \ldots, Z_n)$, $Z_i \in \{0, 1\}$:

$$P_F(z) = \prod_{i=1}^{n} p_i^{z_i}(1 - p_i)^{1-z_i} \tag{3}$$

Further, the objective measure Q is naturally extended to the measure $Q_{\text{smooth}} : \mathcal{F} \times \mathcal{L} \to \mathbb{R}$ by taking the expectation of Q w.r.t. randomness of the selected list L_Z:

$$Q_{\text{smooth}}(F, L) = \mathbb{E}_{Z\sim P_F}\, Q(L_Z) = \sum_{z\in\{0,1\}^n} Q(L_z) P_F(z)$$

Consequently, the LSO problem is transformed to maximization of the expectation of $Q_{\text{smooth}}(F, L)$ w.r.t. randomness of the original list L on the space $\mathcal{F}$:

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

The gradient of Q_{smooth} w.r.t. the model prediction $f(x_j)$ on an item $d_j, j = 1..n$, required by the GD-based optimization algorithm is expressed as follows:

$$\frac{\partial Q_{\text{smooth}}(F, L)}{\partial f(x_j)} = \mathbb{E}_{Z\sim P_F} Q(L_Z)(-p_j)^{1-Z_j}(1-p_j)^{Z_j} \tag{4}$$

Since precise calculation of this expression includes summation of 2^n components, its execution for each training item d_j at each iteration of the GD-based optimization algorithm is infeasible in a standard production environment. To address this problem, we suggest two approaches.

Policy Gradient The first approach aims to estimate the expectation in Equation 4 by Monte Carlo sampling of lists L_Z according to distribution $P_F(z)$ from Equation 3. This method is known as *likelihood ratio method* (Glynn, 1990) in the case of a more general problem which is to estimate the gradient of the expectation of a *reward* function (Q in

the case of LSO) of a random variable (list L_Z in the case of LSO). The optimization algorithm based on this method is known as *policy gradient* (Williams, 1992).

Since such an estimate of the gradient has a high variance, the trick with using *reward baseline* b was proposed (Williams, 1992): by considering the reward $Q'(L) = Q(L) - b$ instead of $Q(L)$ in the gradient estimation procedure, we keep this estimation unbiased but enable reducing its variance by using an appropriate baseline b. Indeed, intuitively, the variable $(Q(L_Z) - b)(-p_j)^{1-Z_j}(1-p_j)^{Z_j}$ from Equation 4 has a lower variance when the mean of the variable $(Q(L_Z) - b)$ is closer to zero, since the multiplier $(-p_j)^{1-Z_j}(1-p_j)^{Z_j}$ changes its sign with Z_j, while the value $Q(L_Z)$ weakly depends on Z_j. See (Kimura et al., 1995) for the detailed analysis.

It motivates us to consider the expectation or the median of the distribution of $Q(L_Z)$ for a baseline, which would be specific for each selection algorithm F and each original list L. However, their precise estimation, again, requires many samples of L_Z and, thus, is infeasible. Hence, we use for b the mode of the distribution of $Q(L_Z)$, which is equal to $Q(L_{z_F^{0.5}})$, where $z_F^{0.5}$ is the vector of the most probable selection decisions for each item according to the predicted probabilities p_i, i.e., $z_{F,i}^{0.5} = \mathbb{1}\{p_i > 0.5\}$. Noteworthy, $L_{z_F^{0.5}}$ is the selected list we would choose if we apply to the list L the threshold-based selection relying on the predicted probability $\sigma(f(x))$ as on $M(x)$ in Equation 2 with the threshold $t = 0.5$. With such defined baseline, our estimate of the gradient on one sampled list L_z is equal to $(Q(L_z) - Q(L_{z_F^{0.5}}))(-p_j)^{1-z_j}(1-p_j)^{z_j}$. Naturally, it votes for shifting the probability p_j towards the decision z_j and, thus, for increasing the probability of the list L_z iff its quality $Q(L_z)$ is higher than the quality $Q(L_{z_F^{0.5}})$ of the currently chosen list for the threshold $t = 0.5$.

As a result, the Policy Gradient approach relies on the following gradient estimate:

$$\frac{\partial \hat{Q}_{\text{smooth}}}{\partial f(x_j)} = \frac{1}{s}\sum_{i=1}^{s}(Q(L_{z^{(i)}}) - Q(L_{z_F^{0.5}}))(-p_j)\left(\frac{1-p_j}{-p_j}\right)^{z_j^{(i)}} \tag{5}$$

where lists $L_{z^{(1)}}, \dots, L_{z^{(s)}}$ are sampled according to $P_F(z)$ independently and s is the number of samples to be a hyper-parameter of the gradient estimation procedure.

Lower Bound Optimization Another approach is to approximate the objective Q_{smooth} by another function of the model predictions $f(x_j)$ with a gradient exactly calculated with low time complexity. For this purpose, we consider the most practical case of an additive objective measure $Q(d_1, \dots, d_n) = \sum_{i=1}^{n} r_i w_i$ with w_i being a convex function of the position i. For example, DCG and RBP satisfy this condition. In this case, by transferring the idea suggested in (Valizadegan et al., 2009) for LTR, the objective Q_{smooth} can be lower-bounded by the following measure:

$$Q^{\text{low}}_{\text{smooth}}(F, L) = \sum_{i=1}^{n} r_i p_i w(1 + \sum_{j<i} p_j),$$

where $w(i)$ is an alternative notation for w_i. Indeed, using the expression $1 + \sum_{j<i} F(x_j)$ for the position of the item d_i in the selected list (in the case of including the item in the list) and applying the Jensen inequality provide

$$\begin{aligned} Q_{\text{smooth}} &= \sum_{i=1}^{n} r_i p_i \mathbb{E}_F w(1 + \sum_{j<i} F(x_j)) \\ &\geq \sum_{i=1}^{n} r_i p_i w(\mathbb{E}_F(1 + \sum_{j<i} F(x_j))) = Q^{\text{low}}_{\text{smooth}} \end{aligned}$$

The gradients of the objective $Q^{\text{low}}_{\text{smooth}}$ follow the equation:

$$\begin{aligned} \frac{\partial Q^{\text{low}}_{\text{smooth}}}{\partial f(x_j)} = \sigma'(f(x_j))\Bigg(& r_j w(1 + \sum_{l<j} p_l) \\ & + \sum_{i=j+1}^{n} r_i p_i w'(1 + \sum_{l<i} p_l)\Bigg) \end{aligned} \tag{6}$$

Notably, all the gradients $\{\frac{\partial Q^{\text{low}}_{\text{smooth}}}{\partial f(x_j)}\}_{j=1}^{n}$ can be calculated with the same asymptotic time complexity as one of them, $O(n)$: we use $O(n)$ steps to obtain $\{\sum_{l<i} p_l\}_{i=1}^{n}$, then $O(n)$ operations to calculate $\{\sum_{i=j+1}^{n} r_i p_i w'(1 + \sum_{l<i} p_l)\}_{j=1}^{n}$ and, finally, $O(n)$ operations to obtain the gradients.

Overall, this approach estimates gradients of the measure Q_{smooth} with a non-zero bias, but with the zero variance, while the Policy Gradient approach does it unbiasedly with a non-zero variance. Thus, it is not clear a priori which approach is more effective.

Having the model $f(x)$ learned, we no longer need to model the selection decision $F(x)$ stochastically. Hence, we define it deterministically by threshold-based rule from Equation 2, where we use $\sigma(f(x))$ for $M(x)$.

Our implementation of all the proposed approaches is described in Section 5.

4.3. Fighting Local Extremum

However, in the LSO problem, both above-described approaches of direct optimization face an important issue: both objectives Q_{smooth} and $Q^{\text{low}}_{\text{smooth}}$ as functions in the space of model predictions $f(x_i)$ on the training examples may have local maximum points which do not correspond to the global maximum[5]. As our experimental results demonstrate

[5]For a parametric model $f(x_i) = g(w, x_i)$, a point f_0 of extremum in the space of model predictions $f(x_i)$ corresponds to an extremum in the space of parameters w if $g(w, x)$ is continuous w.r.t. w and the point f_0 lays in the value domain of $g(w, x)$

(see Section 6), this issue is crucial for effective learning of selection algorithms described in Section 4.2. The following theorem shows that such local maximum points are possible even in the case of the only list, i.e. $\mathbb{P}$ is concentrated at it (see Section 2 of the supplementary for the proof).

Theorem 1 *In the LSO problem with a quality measure* $Q(d_1, \dots, d_n) = \sum_{i=1}^{n} r_i w_i$*:*
1) If the weight w_i *is a logarithmically convex (concave) function of the position* i*, i.e.,* $w_i/w_{i+1} > (<)\ w_{i+1}/w_{i+2}$ $\forall i \in \mathbb{N}$*, there exist a list* $L = (d_1, \dots, d_n)$ *and a selection algorithm* F *such that* F *provides a local but not the global maximum for* $G_1(f_1, \dots, f_n) = Q_{\text{smooth}}(F, L)$ *and* $G_2(f_1, \dots, f_n) = Q^{\text{low}}_{\text{smooth}}(F, L)$ *(where* $f_i = f(x_i)$*) in the space* $\mathbb{C}^1(\mathbb{R}^n)$*, where the distribution* $\mathbb{P}$ *of the original list is concentrated at the list* L*, i.e.,* $\mathbb{P}(L) = 1$*.*
2) If $\{w_i, i \in \mathbb{N}\}$ *is the geometric progression, i.e.,* $w_i/w_{i+1} = const$*, then for any list* L*, in the case* $\mathbb{P}(L) = 1$*, the global maximum of* $G_1(f_1, \dots, f_n)$ *and* $G_2(f_1, \dots, f_n)$ *is the only local one.*

In particular, the conditional of logarithmically convex weights is valid for DCG. Thus, GD-based approaches of its direct optimization may wrongly converge to points of local maximum. Note that, in the case of algorithms scoring each item individually, this problem is specific for LSO in comparison with LTR, where any loss function has the only local minimum. Indeed, for a point L (a ranked list of items) different from the global optimum point L^*, there is a pair of neighbor items in L ranked differently from L^*. Then we can swap only these two items by changing only the model prediction for one of them and thus improve the ranking quality. Thus, the point L is not a local minimum.

At the same time, in the OSP approach, the loss function (logistic loss) is convex w.r.t. predictions $f(x_i)$ and the target is the globally optimal decision in terms of Q_{smooth} too. Thus, the opposite to the loss of OSP can be considered as a convex approximation of Q_{smooth} in the space of model predictions $f(x_i)$. Motivated by this, we suggest a combined approach to the LTR problem: first, to learn OSP selection algorithm and then to improve it by direct optimization. Intuitively, OSP learns a model close to minimum of its loss function (on Figure 1, this moves us from initial point F_0 to the point F_1) ignoring points of local maximum of Q_{smooth} located near to its path, and then direct optimization climbs the closest local (ideally, also the global) maximum of Q_{smooth} (in terms of Figure 1, this moves us from the point F_1 to the point F_2). In order to regulate the confidence of the direct optimization procedure in the prior decisions on the training items provided by OSP, we start this procedure from scaled model predictions $f'(x_i) = M \cdot f(x_i)$, where $f(x_i)$ is the prediction of OSP and M is a scale parameter to be fitted.

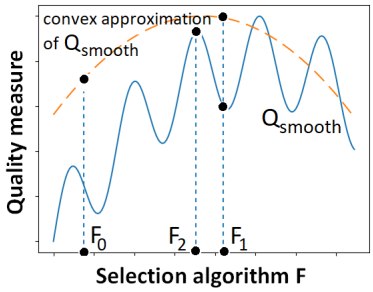

Figure 1. Schematic illustration of two steps of learning: (i) OSP: optimize the convex approximation of Q_{smooth} (logistic loss) and (ii) directly optimize Q_{smooth}

5. Experimental Settings

Data We collected our data from the live stream of search queries submitted in October-November 2017 to a popular product search engine Yandex.Market[6] aggregating offers of products from online shops and serving millions of user queries daily. We considered only queries for which the user chose ordering by price in ascending order and sampled a set D of 30K of them. In terms of Section 2, an *offer* corresponds to an item and the *offer price* corresponds to the dedicated attribute. For each query, we considered only top-500 cheapest candidate offers from tens of thousands of offers (60K in average) chosen by a weak production selection algorithm, which selects all the offers having at least one word from the query in the name or in the description. Further, we randomly split all the queries from the collected dataset D into 5 parts of equal size to run 5-fold cross-validation: at each i of five runs, 80% of the queries (D^i_{train}) are used for training, 10% (D^i_{valid}) are used for tuning hyperparameters of the algorithms and 10% (D^i_{test}) are used for testing. The data with labels used for learning all the models and for evaluation by DCG-RR (see description of labels and metrics below) is available in open source [7].

Features In our experiments, we use (in all methods including baselines) all the features of the Yandex.Market ranker (see Section 3 of the supplementary for details).

However, in LSO, according to the form of the optimal selection algorithm (see Proposition 1), it is useful to implicitly predict not only the item relevance r_i (as in LTR), but also the position i of this item in the original ordered list L and the threshold $t(i)$ for this list, which depends on the aggregated relevance of items located below in the list L. Thus, we are motivated to construct features encoding distributions of relevance and price over offers in L. Hence, we added four features (for all methods including baselines) to our experiments: models trained to predict the average

[6] https://market.yandex.ru
[7] https://research.yandex.com/datasets/market

price (relevance) of top-1 and top-10 offers by production relevance are referred as $AvPricePred$ ($AvRelPred$).

Labels We use the relevance score of each item assigned by the production ranker (referred as *production relevance* below) as the predicted relevance in the baseline approaches and as ground truth relevance in our approaches. See Section 5 of the supplementary for a detailed description.

Metrics For the preliminary analysis of the algorithms to be compared, we, first, evaluate result pages with maximum length of 500 items. As the objective measure of the result page quality in this case, we use the product search specific version (reflecting the fact that users explore more items than in web search) of DCG with position weights equal to reciprocal positions (reciprocal ranks) and with production relevance r_i used as relevance gains: $DCG\text{-}RR(r_1, \dots, r_k) = \sum_{i=1}^{k} r_i/i$.

For the major independent evaluation of the result page relevance, we collected human relevance judgments of 5 grades (from 0 to 4) for top-10 results of each selected list produced by the tested algorithms trained on D^1_{train} and applied to 2K queries from D^1_{test}. On this basis, we measure DCG@10 and precision@10 (p@10) (the share of at least partly relevant offers, i.e., ones with relevance labels 1, 2, 3 and 4, among top-10) and stup@12 (the share of stupid[8] results among results of the first page).

Finally, most representative algorithms were compared in online experiments that lasted for 3 days (resulted in 23K unique users and 100K queries for each algorithm). For evaluation, the following key production metrics are used: click MRR (Craswell, 2009) (inverse of the position of the first click; 0, if there are no clicks), abandonment (share of queries without clicks) and CTR@12 (share of clicked results among top-12). Due to proprietary reasons, we present only relative values of the online metrics.

Machine-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 (Chapelle & Chang, 2011; Burges, 2010) and click prediction (Konig et al., 2009; Trofimov et al., 2012). We use GBDT implementation in open-sourced CatBoost[9] Python package, since it outperforms the most popular alternatives, XGBoost (Chen & Guestrin, 2016) and LightGBM (Ke et al., 2017), according to the comparison by Prokhorenkova et al. (2018).

To obtain a formal description of the learning algorithm for a particular loss function, one should replace L in Algorithm 3 in (Prokhorenkova et al., 2018) by that loss. The whole source code of our learning algorithm and its difference from CatBoost release 0.10.04 are available[10]. Later, it was added to CatBoost as StochasticFilter loss[11]. See Section 4 of the supplementary for CatBoost parameters we used.

Algorithms to be tested

– $Oracle$: finds the optimal list by applying EQA (see Section 3) to ground truth relevance labels for optimizing DCG-RR. $Oracle$ provides the ideal list. However, in practice, we do not have ground truth labels and cannot apply EQA to a large database.

Baselines described in Sections 1 and 3:

– $WeakCutoff$: selects all the items from L_i. Note that they were retrieved by the above-described weak production cutoff-based selection.

– $ConstCutoff$: selects items with relevance predictions above a constant threshold $t^{\text{rel}}_{\text{const}}$ fitted on $D^i_{\text{train}} \cup D^i_{\text{valid}}$ (see Table 3 in the supplementary for fitted values of this and below-described thresholds);

– $QueryCutoff$: for a query q, selects items with relevance predictions above the threshold $t^{\text{rel}}(q)$, which is predicted using query features by a model trained by minimizing MSE on $D^i_{\text{train}} \cup D^i_{\text{valid}}$ with the optimal relevance thresholds as targets [12].

Our approaches, proposed in Section 4, each selects items with the predicted probability above a constant threshold $t^{\text{prob}}_{\text{const}}$ and uses DCG-RR for objective measure Q:

– OSP (Optimal Selection Predictor): the model trained to predict decisions of the optimal selection (see Section 4.1).

– PG (Policy Gradient): the model of prediction trained by direct optimization of Q_{smooth}, a smoothed modification of the original measure Q, with sampling-based gradient estimation, starting from predictions $f(x_i) = 0.01$ for all the training examples (see Section 4.2).

– LBO (Lower Bound Optimization): the model of prediction trained by direct optimization of $Q^{\text{low}}_{\text{smooth}}$, a lower bound for Q_{smooth}, starting from predictions $f(x_i) = 0.01$ (see Section 4.2).

– $OSP + PG$ / $OSP + LBO$: the model of prediction trained as OSP during first $n_1 \leq 500$ iterations and then, after prediction transformation, trained as PG/LBO during the remaining $n_2 \leq 500$ iterations (n_1 and n_2 to be fitted), see Section 4.3. Thus, this approach uses the same limit of 1000 training iterations as other ML-based approaches. The fitted values of the scale parameter M are 2 and 1 respectively. The number s of sampled selected lists per each training list in the gradient estimation by Equation 5 was set to $s = 1$, since several samples did not improve the quality in our experiments.

[8] a special sub-class of irrelevant results, which represent extreme/confusing irrelevance

[9] https://catboost.ai

[10] https://github.com/TakeOver/catboost/tree/0.10.4_release

[11] https://github.com/catboost/catboost/commit/df18d16

[12] The optimal threshold is defined as the average of the minimum relevance among items selected by $Oracle$ and the maximum relevance among the excluded ones

Table 2. Performance, absolute for $WeakCutoff$ and relative Δ to $WeakCutoff$, % for others

Approach	DCG-RR	DCG@10	p@10	stup@12
$WeakCutoff$	0.52	1.07	0.73	0.06
$ConstCutoff$	0.05%	2.9%	1.6%	-11.7%
$QueryCutoff$	0.56%	6.3%	4.3%	-17.5%
OSP	3.86%	20.3%	10.7%	-33.2%
$OSP + LBO$	4.17%	**22.4%**	12.0%	**-37.6%**
$OSP + PG$	**4.33%**	**22.4%**	**12.2%**	-36.8%
$Oracle$	14.44%			

Table 3. Online performance, relative Δ to $WeakCutoff$, %

Approach	Abandonment	CTR@12	MRR
$QueryCutoff$	-1.4%	8.7%	5.7%
OSP	-4.5%	19.7%	24.6%
$OSP + PG$	**-5.1%**	**24.8%**	**36.3%**

6. Experimental Results

Analysis of the local extremum problem First, we report that the LBO and PG algorithms with the fitted parameters differ from $WeakCutoff$ negligibly: their training processes started from the decisions of $WeakCutoff$ ($f(x) = 0.01$) and almost did not change these decisions (the learned algorithms exclude 0 and 0.2 items per query respectively). When we initialized decisions by $f(x) = -0.01$, both learned algorithms did not achieve even the quality of $WeakCutoff$. To demonstrate the problem of local extremum, discussed in Section 4.3, more explicitly, we define *hard* items d_i for a given stochastic selection algorithm as ones for which the selection decision is rather confident with $|p_i - 0.5| > 0.25$, is globally suboptimal (i.e., does not coincide with one of $Oracle$), but locally optimal (i.e., optimal given fixed selection decisions on other items). The average number of hard items per list N_{hard} is presented in Table 3 in the supplementary. Clearly, LBO and PG greatly suffer from hard items: while $Oracle$ excludes 78.9 items per list, 62.6 and 58.9 of them are hard for that filters respectively. Thus, the local extremum problem strongly prevents effective direct optimization of Q_{smooth}. Since $N_{\text{hard}} = 16.9$ is rather small for OSP, it seems attractive to start direct optimization from this algorithm. Notably, starting from OSP, both LBO and PG even decrease the number of hard items ($N_{\text{hard}} = 6.1$ and 10.6), i.e., correct mistakes of OSP.

Main results The results of offline experiments are presented in Table 2. Obviously, $ConstCutoff$ approach is superior in comparison with $WeakCutoff$ in terms of all the metrics. However, the additional gain from upgrading it to $QueryCutoff$ is even more impressive, what emphasizes importance of query features in LSO. Indeed, they contain information about relevance and price distributions over candidate offers, which is useful in LSO according to our intuition for feature construction described in Section 5.

Next, OSP provides an outstanding increase of all the metrics. The reason is that it has the optimal selection as a target and is not restricted by any crude assumptions about dependence $t(i)$ of relevance threshold on the item position i (in terms of Proposition 1), in contrast to the cutoff approaches assuming $t(i) = const$ for each given query.

Finally, both combined approaches $OSP + LBO$ and $OSP + PG$ *remarkably* outperform OSP, confirming our intuition about the advantages of their components described in Section 4.3. Among two combined approaches, $OSP + PG$ performs slightly better with significant difference only for p@10 [13]. Thus, variance of the PG gradient estimation reduced by means of our reward baseline, is a smaller issue than the bias of the gradient estimate in LBO. All the other pairwise comparisons between algorithms in Table 2 are significant [13]. Being normalized w.r.t. $Oracle$, quality of $OSP + PG$ in terms of DCG-RR is 0.91.

In our online experiments, we compare $QueryCutoff$ approach as the strongest baseline, OSP as an ML-based approach without restrictive model assumptions and $OSP + PG$ as our best approach. The results are presented in Table 3 and confirm all the previous conclusions. All the pairwise comparisons are significant [13], except for $OSP + PG$ vs OSP for Abandonment metric.

Feature analysis See Section 3 of the supplementary for feature analysis. In particular, it shows that the new features, we constructed specifically for the LSO task, remarkably increase the quality of $OSP + PG$. This confirms our intuition for feature construction proposed in Section 5.

7. Conclusions

Based on a new machine-learning setting of the selection task, we proposed two effective and practically feasible theoretically-based approaches to LSO. Their construction includes (i) smoothing the original objective function Q by Q_{smooth}, (ii) approximation of Q_{smooth} by a convex surrogate to avoid local extremes and (iii) two ways to overcome an exponential (w.r.t. the number of training examples) complexity of calculation of the Q_{smooth} gradient for its direct optimization. The selection algorithms produced by these approaches for the large-scale product search engine dramatically outperform the baselines in our offline and online experiments in terms of all key search quality metrics.

[13] p-value< 0.05 for one-tailed paired t-test

References

Burges, C. J. From ranknet to lambdarank to lambdamart: An overview. Technical report, June 2010.

Carmel, D., Cohen, D., Fagin, R., Farchi, E., Herscovici, M., Maarek, Y. S., and Soffer, A. Static index pruning for information retrieval systems. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 43–50. ACM, 2001.

Chapelle, O. and Chang, Y. Yahoo! learning to rank challenge overview. In *Yahoo! Learning to Rank Challenge*, pp. 1–24, 2011.

Chapelle, O. and Wu, M. Gradient descent optimization of smoothed information retrieval metrics. *Information retrieval*, 13(3):216–235, 2010.

Chapelle, O., Metlzer, D., Zhang, Y., and Grinspan, P. Expected reciprocal rank for graded relevance. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pp. 621–630. ACM, 2009.

Chapelle, O., Manavoglu, E., and Rosales, R. Simple and scalable response prediction for display advertising. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 5(4):61, 2015.

Chen, T. and Guestrin, C. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794. ACM, 2016.

Craswell, N. Mean reciprocal rank. In *Encyclopedia of Database Systems*, pp. 1703–1703. Springer, 2009.

Friedman, J., Hastie, T., Tibshirani, R., et al. Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *The annals of statistics*, 28(2):337–407, 2000.

Glover, E. J., Lawrence, S., Birmingham, W. P., and Giles, C. L. Architecture of a metasearch engine that supports user information needs. In *Proceedings of the eighth international conference on Information and knowledge management*, pp. 210–216. ACM, 1999.

Glynn, P. W. Likelihood ratio gradient estimation for stochastic systems. *Communications of the ACM*, 33 (10):75–84, 1990.

Hasanain, M., Elsayed, T., and Magdy, W. Improving tweet timeline generation by predicting optimal retrieval depth. In *Asia Information Retrieval Symposium*, pp. 135–146. Springer, 2015.

Järvelin, K. and Kekäläinen, J. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446, 2002.

Ji, S., Zhou, K., Liao, C., Zheng, Z., Xue, G.-R., Chapelle, O., Sun, G., and Zha, H. Global ranking by exploiting user clicks. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pp. 35–42. ACM, 2009.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., and Liu, T.-Y. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, pp. 3149–3157, 2017.

Kimura, H., Yamamura, M., and Kobayashi, S. Reinforcement learning by stochastic hill climbing on discounted reward. In *Machine Learning Proceedings 1995*, pp. 295–303. Elsevier, 1995.

Konig, A. C., Gamon, M., and Wu, Q. Click-through prediction for news queries. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pp. 347–354. ACM, 2009.

Liu, T.-Y. et al. Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval*, 3(3): 225–331, 2009a.

Liu, T.-Y. et al. Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval*, 3(3): 225–331, 2009b.

Moffat, A. and Zobel, J. Rank-biased precision for measurement of retrieval effectiveness. *ACM Transactions on Information Systems (TOIS)*, 27(1):2, 2008.

Patel, M. B. and Shah, D. D. Meta search ranking strategies. *International Journal of Information and Computing Technology (RESEARCH@ ICT) ISSN*, 976(5999):24–25, 2011.

Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., and Gulin, A. Catboost: unbiased boosting with categorical features. In *Advances in Neural Information Processing Systems*, pp. 6639–6649, 2018.

Schurman, E. and Brutlag, J. The user and business impact of server delays, additional bytes, and http chunking in web search. In *Velocity Web Performance and Operations Conference*, 2009.

Spirin, N. V., Kuznetsov, M., Kiseleva, J., Spirin, Y. V., and Izhutov, P. A. Relevance-aware filtering of tuples sorted by an attribute value via direct optimization of search quality metrics. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 979–982. ACM, 2015.

Trofimov, I., Kornetova, A., and Topinskiy, V. Using boosted trees for click-through rate prediction for sponsored search. In *Proceedings of the Sixth International Workshop on Data Mining for Online Advertising and Internet Economy*, pp. 2. ACM, 2012.

Valizadegan, H., Jin, R., Zhang, R., and Mao, J. Learning to rank by optimizing ndcg measure. In Bengio, Y., Schuurmans, D., Lafferty, J. D., Williams, C. K. I., and Culotta, A. (eds.), *Advances in Neural Information Processing Systems 22*, pp. 1883–1891. Curran Associates, Inc., 2009. URL http://papers.nips.cc/paper/3758-learning-to-rank-by-optimizing-ndcg-measure.pdf.

Wang, L., Lin, J., and Metzler, D. A cascade ranking model for efficient ranked retrieval. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pp. 105–114. ACM, 2011.

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

Wu, M., Chang, Y., Zheng, Z., and Zha, H. Smoothing dcg for learning to rank: A novel approach using smoothed hinge functions. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pp. 1923–1926. ACM, 2009.