Multi-objective Relevance Ranking

Michinari Momma
michi@amazon.com
Amazon.com Inc.
Seattle, WA

Alireza Bagheri Garakani
alirezg@amazon.com
Amazon.com Inc.
Seattle, WA

Yi Sun
yisun@amazon.com
Amazon.com Inc.
Seattle, WA

ABSTRACT

In this paper, we introduce an Augmented Lagrangian based method in a search relevance ranking algorithm to incorporate the multi-dimensional nature of relevance and business constraints, both of which are the requirements for building relevance ranking models in production. The off-the-shelf solutions cannot handle such complex objectives and therefore, modelers are left hand-tuning of parameters that have only indirect impact to the objectives, attempting to incorporate multiple objectives (MO) in a model. This process is time-consuming and tends to face sub-optimality. The proposed method is designed to systematically solve the MO problem in a constrained optimization framework, which is integrated with a popular Boosting algorithm and is, by all means, a novel contribution. Furthermore, we propose a procedure to specify the constraints to achieve business goals and the exploration scales *linearly* in the number of constraints, while existing methodology can scale *exponentially*. The experimental results show that the method successfully builds models that achieve MO criteria much more efficiently than existing methods. The potential impact includes significant reduction in model development time and allows for automation of model refresh even with presence of several MO criteria, in real world production system scale with hundreds of millions of records.

KEYWORDS

Learning to rank; Multi-objective ranking optimization; Product search; Web search

ACM Reference Format:
Michinari Momma, Alireza Bagheri Garakani, and Yi Sun. 2019. Multi-objective Relevance Ranking. In *Proceedings of the SIGIR 2019 Workshop on eCommerce (SIGIR 2019 eCom)*, 8 pages.

1 INTRODUCTION

Relevance in information retrieval (IR) has been extensively studied and applied to various areas such as web search, product search and recommendation, etc. The concept of relevance is multi-dimensional, dynamic and evolutional [1, 15]. In product search, a ranking of products is modeled by customer's historical behavior. Typically, signals such as purchase, add to cart or click are used as a target variable and models are tuned to optimize rankings based on such behaviors. To address the multi-dimensional nature of relevance, various features that represent relevance dimensions are used as input features to a model. For example, Amazon Search [18, 19] has a large number of features to capture relevance with a feature repository consisting of *product* features such as sales and customer reviews, *queries / context* features such as query specificity or customer status, as well as *textual similarity* features between query and products.

Business constraints are additional requirements in production modeling. Some are derived from existing relevance metrics proven to be effective over time and are sought to retain in model refresh to ensure avoiding churn [14]. Some are derived from relevance such as latency to ensure quick search responses, and some are strategic and examples include minimum %-gain to consider experimentation / launch and avoiding adult items for media products. Reduction of search defects that are the search results that do not match the query in various aspects, can be considered for both as it gives better customer experience and business requirement being strict as to ensure the quality of search results. An example of search defect is showing a cheap zirconium ring for a query "diamond ring", which could give customers the impression that the search is broken, or the e-commerce site is more like a flea market, damaging brand image of the service [19].

As discussed above, relevance ranking modeling faces challenges of dealing with multiple metrics of relevance and / or business constraints, i.e., multiple-objectives (MO). Off-the-shelf machine learning solutions [5, 10, 13], cannot handle such complicated objectives in a systematic manner. Although the effect can be limited, one may try to employ either over-weighting (OW) [7] by exemplars and/or by search impressions, a set of query-product pairs for a session on a given day, to influence the objective function in a desired way. Tuning of weight values is required in this approach and it can suffer combinatorial exploration to search for a best combination of weightings, which becomes prohibitive as the number of objectives grows.

To address the issue, we propose a constrained optimization method applied to the Gradient Boosting Tree (GBT). Specifically, we introduce a constraint optimization in the LambdaMART algorithm, the most prominent method for relevance modeling today. LambdaMART is based on GBT [8] using CART [2] and employs optimization on IR metrics such as the normalized cumulative discounted gains (NDCG) or the mean reciprocal rank (MRR). One of our goals is to propose a *practical* approach to the MO problem / task in a production setting, which is constrained by memory and model size, i.e., the number to trees, and efficiency of the learning algorithm to avoid regression of latency in scoring and model development timeline.

In order to meet the requirements and limitations, adaptation of the Augmented Lagrangian (AL) method [17] to LambdaMART (AL-LM) is proposed. AL converts the original constrained problem into an unconstrained problem by adding penalty terms that

In: J. Degenhardt, S. Kallumadi, U. Porwal, A. Trotman (eds.):
Proceedings of the SIGIR 2019 eCom workshop, July 2019, Paris, France, published at http://ceur-ws.org

penalize constraint violations. The Lagrange multipliers, i.e., dual variables are estimated at each iteration. The advantage of AL for incorporating into Boosting framework is its simplicity and smoothness. With AL, we introduce dual variables in Boosting. The dual variables are iteratively optimized and fit well within the Boosting iterations. The Boosting objective function is replaced by the unconstrained AL problem and the gradient is readily derived using the LambdaMART gradients. With the gradient and updates of dual variables, we solve the optimization problem by jointly iterating AL and Boosting steps. To the best of our knowledge, our work is the first to explicitly introduce constrained optimization problem in Boosting and the first to apply it search relevance problems. Although in this paper, the proposed method is implemented in a Boosting algorithm and applied for relevance modeling (ranking problems), it is naturally applicable to other algorithms such as logistic regressions, SVM, and neural networks with various applications in classification and regression domain.

Our code is currently incorporated in GBM in R. We plan to implement the algorithm into XGBoost and / or LightGBM and make them publicly available.

Section 2 introduces existing work in multi-objective optimization in relevance ranking. Section 3 gives formulations and algorithm of our proposed method. Section 4 illustrates experimental results and Section 5 concludes this paper.

2 RELATED WORK

The MO problem in search relevance literature has been a popular research topic and there have been two major directions: combining multiple objectives in a single model and aggregate multiple models tuned for each objective. As for the single model approach, Dong et al. [7] proposes an over-weighting model, based on GBrank [22], to adjust importance weighing of examples coming from different data sources, for incorporating recency into the web search relevance ranking. Svore et al. [20] optimizes both human-labeled relevance and click, with the former being prioritized. Their objectives are based on NDCG that are modeled by the LambdaMART-loss. The combined objective function is a weighted linear combination of the two, which is in fact popular in literature [16, 21].

Another popular approach is to use aggregation of multiple rankers. Dai et al. [6] proposes a hybrid approach of label aggregation and mixture of experts for achieving recency and general ranking objectives. Kang et al. [12] relies on editorial grading for each metric and relative preference to optimize both label aggregation and model aggregation in a supervised manner. Each component model is trained for each objective. They apply the method for optimizing three objectives that are general ranking (i.e. matching), distance and reputation. These methods require separate models for each objective and are not directly applicable in our setting where we deploy a single model in production.

Most of the existing algorithms rely on heuristics, such as weighting or linear combination of objectives / models, and require manual tuning of hyper-parameters, which becomes prohibitive as the number of objectives becomes large. Our approach, in contrast, automatically yields a model that satisfies minimal requirements over MO's, alleviating the effort of hand-tuning of hyper-parameters, which makes a clear distinction from the existing methods.

In terms of application of constrained optimization to more generic problems, Bayesian Optimization (BO) with constraints [9, 11] could be applicable. However, in our problem, objective functions and gradients are available, even if it is a surrogate function, and exploiting gradients in optimization should be more efficient. Incorporation of BO on top of our approach to fine tune metrics or hyper parameters in Boosting component could be an interesting direction in the future.

3 FORMULATION AND ALGORITHM

In this section, we provide formulation and an algorithm of the proposed method in details. As a method for production modeling, there are some requirements in design: *latency* in scoring and the *computational cost* in training. For the former, we should avoid constructing large number of trees and complex structure of each individual tree as it translates into latency degradation. For the latter, we should avoid large number of iterations, or nested iterations, as the objective function evaluation can be expensive in search application.

3.1 LambdaMART objective and gradient

First, let us review the LambdaMART formulation. Suppose we have a set of queries $Q = \{q\}$ and an index set of documents (products in product search) associated with each query I^q. A document is denoted by d_i with an index $i \in I^q$. A set of pairs of document indices by $P^q = \{(i, j)\}$ with the relation $R_i \rhd^q R_j$: d_i is more relevant than d_j for a given query q. Suppose also we have a single cost function to optimize, referred to as *primary cost*. Given a relevance model f, for a query and a document, a score of the document is computed as a function of its input features that could be query dependent: $s_i^q = f(x_i^q)$. A probability of the relevance relationship is modeled by a sigmoid function:

$$\begin{aligned} Prob(R_i \rhd^q R_j) &= Prob\left((i,j) \in P^q\right) \\ &= \left(1 + e^{-\sigma\left(s_i^q - s_j^q\right)}\right)^{-1} \end{aligned} \tag{1}$$

where σ is a parameter to determine the shape of the sigmoid function. LambdaMART [3, 16] is based on the pairwise cost that is the cross-entropy with a rank-dependent weight to incorporate importance of high ranks. For a pair $(i, j) \in P^q$, the cross entropy is given by

$$c^{pm}\left(s_i^q, s_j^q\right) = \left|\Delta Z_{i,j}^{pm,q}\right| \log\left(1 + e^{-\sigma\left(s_i^q - s_j^q\right)}\right) \tag{2}$$

where $\Delta Z_{i,j}^{pm,q}$ is the difference of a metric such as NDCG between the (i, j) pair in one order and one that is flipped. For a metric like NDCG, relevant documents in higher ranked position gets a high value and that in lower a low value. Therefore, the weight $\Delta Z_{i,j}^{pm,q}$ tends to be large when a high ranked position is involved in the pair. For a query, the total cost $c(\boldsymbol{s}^q)$ with $\boldsymbol{s}^q$ being a vector containing all documents for the query, i.e., $\boldsymbol{s}^q = [s_1, .., s_{I^q}]$, is given by

$$c^{pm}\left(\boldsymbol{s}^q\right) = \sum_{(i,j)\in P^q} c^{pm}\left(s_i^q, s_j^q\right) \tag{3}$$

The primary objective function of LambdaMART is a sum over all queries:

$$C^{pm}(\boldsymbol{s}) = \sum_{q\in Q}\sum_{(i,j)\in P^q} c^{pm}(s_i^q, s_j^q) = \sum_{q\in Q} c^{pm}(\boldsymbol{s}^q) \tag{4}$$

where $\boldsymbol{s}$ is a concatenated vector containing scores for all queries: $\{\boldsymbol{s}^q | q \in Q\}$.

Boosting in LambdaMART is based on the GBT, which generates a linear combination of base functions that are the decision trees, which are learned to fit the gradient. Typically, in GBT, once a tree is constructed, the function value of each leaf is computed by an estimate of the Newton step for the node. More formally, at an n^{th} iteration, leaf nodes $\{R_{nl}\}_{l=1}^{L}$, where L is the number of leaves of a tree, are generated by CART for given input features of a query and documents, and the gradient as a target: $\{(x_i^q, g_n(f(x_i^q)))\}_{q\in Q,\ i\in I^q}$, with g_n being the gradient with respect to a score. The function value for each node is given as: $\gamma_{nl} = -\left(\sum_{x_i^q\in R_{nl}} \partial^2 C^{pm}/\partial s_i^q\right)\left(\sum_{x_i^q\in R_{nl}} \partial C^{pm}/\partial\left(s_i^q\right)^2\right)^{-1}$. Therefore, the score, or the prediction function is given as follows:

$$s_i^q = f(x_i^q) = f_0(x_i^q) + \sum_{n=1}^{N}\sum_{l=1}^{L}\gamma_{nl}\delta(x_i^q \in R_{nl}) \tag{5}$$

where $f_0(x_i^q)$ is an initial base function that could be provided by prior knowledge and δ is the Kronecker delta. To derive the gradient, it is handy to rewrite the query cost as follows:

$$\begin{aligned} c^{pm}(\boldsymbol{s}^q) &= \sum_{i\in I^q} c^{pm}(s_i^q) \\ &= \sum_{i\in I^q}\left(\sum_{j:(i,j)\in P^q} c^{pm}(s_i^q, s_j^q) + \sum_{j:(j,i)\in P^q} c^{pm}(s_j^q, s_i^q)\right) \end{aligned} \tag{6}$$

with $c^{pm}(s_i^q) \equiv \sum_{j:(i,j)\in P^q} c^{pm}(s_i^q, s_j^q) + \sum_{j:(j,i)\in P^q} c^{pm}(s_j^q, s_i^q)$. The first term computes the pairwise cost between d_i and others that are less relevant than d_i. The second term computes that between d_i and others that are more relevant than d_i. The gradient used in LambdaMART [3, 4] is readily computed by taking derivative with respect to the current score. By defining $\lambda_i^{pm,q} \equiv \partial c_q^{pm}/\partial s_i$ and $\lambda_{ij}^{pm,q} \equiv -\sigma\left|\Delta Z_{i,j}^{pm,q}\right|\left(1 + e^{\sigma\left(s_i^q - s_j^q\right)}\right)^{-1}$, we have the gradient formula in terms of λ's.

$$\lambda_i^{pm,q} = \sum_{j:(i,j)\in P^q}\lambda_{ij}^{pm,q} - \sum_{j:(j,i)\in P^q}\lambda_{ji}^{pm,q} \tag{7}$$

Similarly, the second order derivative, defined as $\rho_i^{pm,q} \equiv \partial^2 c^{pm}/\partial\left(s_i^q\right)^2$ is given as follows:

$$\begin{aligned} \rho_i^{pm,q} &= \sigma^2 \sum_{j:(i,j)\in P^q}\left|\Delta Z_{i,j}^{pm,q}\right|\rho_{ij}^{pm,q}\left(1-\rho_{ij}^{pm,q}\right) \\ &\quad - \sigma^2 \sum_{j:(j,i)\in P^q}\left|\Delta Z_{i,j}^{pm,q}\right|\rho_{ji}^{pm,q}\left(1-\rho_{ji}^{pm,q}\right) \end{aligned} \tag{8}$$

where we define $\rho_{ij}^{pm,q} \equiv \left(1 + e^{\sigma\left(s_i^q - s_j^q\right)}\right)^{-1}$.

3.2 Incorporating Augmented Lagrangian method in Boosting

Suppose all objectives are given in terms of cost functions. Just like the primary objective function reviewed in 3.1, we use surrogated cost functions to optimize the metrics we desire. For example, suppose we want to set minimum criteria in NDCG. In this case, we use LambdaMART costs for NDCG and set the cost no greater than the given upper-bound (UB) b, i.e., $C^t(\boldsymbol{s}) \le b^t,\ t = 1, \ldots, T$. Given the constraints represented in terms of cost functions, we have the following constraint optimization problem:

$$\min_{\boldsymbol{s}} C^{pm}(\boldsymbol{s})\ \ s.t.\ C^t(\boldsymbol{s}) \le b^t,\ t = 1, ..., T. \tag{9}$$

The Lagrangian is written by

$$\mathcal{L}(\boldsymbol{s}, \boldsymbol{\alpha}) = C^{pm}(\boldsymbol{s}) + \sum_{t}^{T}\alpha^t\left(C^t(\boldsymbol{s}) - b^t\right) \tag{10}$$

where $\boldsymbol{\alpha} = \left[\alpha^1, ..., \alpha^T\right]$ is a vector of dual variables. The Lagrangian is solved by minimizing with respect to the primal variables $\boldsymbol{s}$ and maximizing with respect to the dual variables $\boldsymbol{\alpha}$. AL iteratively solves the constraint optimization while alleviating non-smoothness in $\boldsymbol{\alpha}$ arising in the dual. In our problem, the Lagrangian at iteration k is written as follows:

$$\begin{aligned} \mathcal{L}_k(\boldsymbol{s}, \boldsymbol{\alpha}) &= C^{pm}(\boldsymbol{s}) + \sum_{t}^{T}\alpha^t\left(C^t(\boldsymbol{s}) - b^t\right) \\ &\quad - \sum_{t}^{T}\frac{1}{2\mu_k}\left(\alpha^t - \alpha_{k-1}^t\right)^2 \end{aligned} \tag{11}$$

where α_{k-1}^t is a solution in the previous iteration and a constant in the current iteration k. μ_k^t is a sufficiently large constant associated with each dual variable α^t. Note that the last term is newly added as compared with Eq. (10) and it gives proximal minimization with iterates α_{k-1}^t, to make the optimization *smooth*.

We maximize the Lagrangian with respect to $\boldsymbol{\alpha} \ge 0$ and minimize with respect to $\boldsymbol{s}$.

$$\max_{\boldsymbol{\alpha}\ge 0}\min_{\boldsymbol{s}}\mathcal{L}_k(\boldsymbol{s}, \boldsymbol{\alpha}) \tag{12}$$

From the stationary condition $\partial\mathcal{L}_k/\partial\alpha^t = 0$, we obtain the update formula for α:

$$\alpha_k^t = \max\left(0, \mu_k^t\left(C^t(\boldsymbol{s}) - b^t\right) + \alpha_{k-1}^t\right). \tag{13}$$

At an iteration k, if the constraint t is not satisfied, i.e., $C^t(\boldsymbol{s}) > b^t$, we have $\alpha_k^t > \alpha_{k-1}^t$, which means the Lagrange multiplier α^t increases unless the constraint is already satisfied. Intuitively, we can consider it as weighting that is adjusted each iteration to overweight an unsatisfied constraint that is associated the cost we want to improve. Note if a constraint should be strictly satisfied at optimality, α^t should take value 0 to maximize the Lagrangian. If a constraint should be satisfied with equality, α^t can take a finite value. By restricting the solution to the former case, we can push the dual variable to 0 whenever the constraint is satisfied:

$$\alpha_k^t = \begin{cases} 0, & \text{if } C^t(\boldsymbol{s}) - b^t < 0 \\ \mu_k^t\left(C^t(\boldsymbol{s}) - b^t\right) + \alpha_{k-1}^t, & \text{otherwise} \end{cases} \tag{14}$$

Intuitively, we can avoid unnecessary iterations to find out $\alpha = 0$ by simply pushing α to zero whenever the constraint is met. Our

preliminary study shows the update Eq. (14) works better than that in Eq. (13) in both primary and sub-objectives. Throughout this paper, we adopt Eq. (14) as the update scheme for dual variables.

As for the primal variables, the first order derivatives are given as follows:

$$\frac{\partial \mathcal{L}_k}{\partial s_i^q} = \sum_{q\in Q, i\in I^q} \left(\lambda_i^{pm,q} + \sum_t \alpha^t \lambda_t^{t,q}\right) = \sum_{q\in Q, i\in I^q} \lambda_i^q \quad (15)$$

where we define $\lambda_i^q \equiv \lambda_i^{pm,q} + \sum_t \alpha^t \lambda_i^{t,q}$. Similarly, by defining $\rho_t^q \equiv \rho_i^{pm,q} + \sum_t \alpha^t \rho_i^{t,q}$, the second order derivative with respect to a score is simply given as follows:

$$\frac{\partial^2 \mathcal{L}_k}{\partial \left(s_i^q\right)^2} = \sum_{q\in Q, i\in I^q} \left(\rho_i^{pm,q} + \sum_t \alpha^t \rho_i^{t,q}\right) = \sum_{q\in Q, i\in I^q} \rho_i^q \quad (16)$$

One interpretation of the AL method is a penalty method on constraint violations. To see it clearly, we can plug the stationary condition Eq. (13) into Eq. (11), yielding

$$\begin{aligned}\mathcal{L}_k(\boldsymbol{s}, \boldsymbol{\alpha}) = C^{pm}(\boldsymbol{s}) &+ \sum_{t\in\{\alpha^t>0\}} \alpha^t \left(C^t(\boldsymbol{s}) - b^t\right) \\ &+ \sum_{t\in\{\alpha^t>0\}} \frac{\mu_k^t}{2} \left(C^t(\boldsymbol{s}) - b^t\right)^2 + \text{const.}\end{aligned} \quad (17)$$

It is evident that it penalizes constraint violations associated with $\alpha^t > 0$ with quadratic function. Therefore, setting a high value of μ_k^t imposes the constraint more strictly but on the other hand, setting a too high value may introduce a non-smoothness behavior which AL tries to avoid.

3.3 An approach to Augmented Lagrangian with LambdaMART

Both AL and LambdaMART algorithms run by iterations. We propose a simple approach to integrate the two; Conduct an AL step calculation to update α at each Boosting iteration, which means the AL iteration index k is set equal to the Boosting iteration index n. As for the AL parameter μ_k^t, we take a fixed policy on iteration: $\mu_k^t = \mu^t$. A variant of algorithm would increase the value as iterations proceeds. Algorithm 1 shows the steps of the algorithm.

Most notably, the additional component to the original LambdaMART at the n-th iteration step is the gradient computation: λ_i^q,ρ_i^q and the update of α_n^t. The modification to existing solvers such as GBM in R [10], XGBoost [5] , and LightGBM [13] should be a minimal effort.

4 EXPERIMENTS

In this section, we analyze the algorithm by a numerical study. The dataset we use is a product search data collected and generated in an e-commerce service. We first study how to set the optimization parameter μ and the cost upper-bounds to ultimately achieve modeling goals using smaller sampled dataset. Then we apply the settings to the full dataset for building a production-ready model. Our prototyping code for AL-LM is currently incorporated in GBM in R and used throughout of this section.

Input: Number of trees N, number of leaves per tree L, learning rate η, AL parameter μ^t. Initial Lagrange multiplier estimate $\alpha_0^t = 0, t = 1, .., T$. Given initial BaseModel

foreach $q \in Q$ **do**
  $f_0(x_i^q) = BaseModel(x_i^q), i \in I^q$
  /* If BaseModel is empty, set $f_0(x_i^q) = 0$ */
end
for *n=1 to N* **do**
  foreach $q \in Q, i \in I^q$ **do**
    $\lambda_i^q = \lambda_i^{pm,q} + \sum_{t=1}^T \alpha_{n-1}^t \lambda_i^{t,q}$,
    $\rho_i^q = \rho_i^{pm,q} + \sum_{t=1}^T \alpha_{n-1}^t \rho_i^{t,q}$
  end
  $\{R_{nl}\}_{l=1}^L$
  /* Create an L leaf tree on $\{(x_i^q, \lambda_i^q)\}_{q\in Q, i\in I^q}$ */
  $\gamma_{nl} = -\frac{\sum_{q\in Q, x_i^q \in R_{nl}} \lambda_i^q}{\sum_{q\in Q, x_i^q \in R_{nl}} \rho_i^q}$
  /* Assign leaf values on Newton step estimate */
  foreach $q \in Q, i \in I^q$ **do**
    $f_n(x_i^q) = f_{n-1}(x_i^q) + \eta \sum_l \gamma_{nl} \delta\left(x_i^q \in R_{nl}\right)$ /* Take step with learning rate η */
  for *t=1 to T* **do**
    Compute cost $C^t(\boldsymbol{s})$ with $\{s_i^q = f_n\left(x_i^q\right)\}_{q\in Q, i\in I^q}$
    Update α_n^t via Eq. (13) or Eq. (14)
  end
end
end

Algorithm 1: AL-LambdaMART (AL-LM)

4.1 Dataset and objectives

The dataset consists of search queries, input features (e.g., query, product, and query-product dependent features such as product sales, customer review, textual matches between a query and products), as well as customer's purchase decision. We follow the basic modeling practice described in [18, 19]. We collect training data for a month worth of the data followed by a week worth of the data for evaluation. For this study, we focus on purchase as a primary objective.

As for sub-objectives, we identify four. The first one ($t1$) is to surface set of products that have relatively good quality and popular to a certain customer segment (e.g., customers with subscriptions or customers who are more interested in trending products.); the insight being promoting popular products of high quality for such a segment while others still have the impressions on such products. The 2nd – 4th ones are products that contain additional benefit/service to such a customer segment; 2nd ($t2$) offers some lowest benefit to the customer of the three and 3rd ($t3$) and 4th ($t4$) offer superior benefits in an increasing order. Generally, benefits are hierarchical and coverage is inversely proportional to it.

Table 1: Baseline unconstrained results.

	purchase		$t1$		$t2$	
	cost	NDCG	cost	NDCG	cost	NDCG
train	0.106	0.826	0.038	0.932	0.047	0.842
valid	0.133	0.809	0.038	0.930	0.045	0.845

Table 2: Upper bounds (b^{t1}, b^{t2}) set by cost reduction rate from baseline.

	%-cost reduction					
	5%	10%	20%	30%	40%	50%
$t1$	0.036	0.034	0.030	0.026	0.023	0.019
$t2$	0.045	0.043	0.038	0.033	0.028	0.024

4.2 Multi-objective ranking illustration

First, we show how the algorithm progresses by iterations. In this study, we use two constraints for simplicity. We randomly sample $20K$ search impressions from the dataset and split into 50% for training and validation sets. We run the algorithm up to 100 iterations for illustration purpose. The UB b^t are set based on the unconstrained baseline; we run the unconstraint problem first, which is exactly the same as the original LambdaMART algorithm. Table 1 shows cost and NDCG for the purchase, $t1$ and $t2$. Then the UB values are set based on a cost reduction rate, such as 5, 10, ..., 50%, from the unconstrained baselines. Table 2 shows actual values of UB's used in the experiments.

4.2.1 Cost curve illustration. Figure 1 (a) illustrates curves of costs that are the objectives in the problem. The value of α_n^t is also shown in Figure 1 (b). We set UB for 20% cost reduction against the unconstrained baseline and set $\mu = 10K$, which is a setting that is found to be large enough as studied in next subsection. In cost curves, solid curves correspond to the cost values on the training data and dotted curves those on the validation data. Note solid lines represent the UB's. As seen in the Figure 1 (a), the initial few iterations try to satisfy constraints aggressively. Then spend a number of iterations in the over-satisfied region, gradually challenging the UB's. The behavior of α can be seen in (b). α is pushed to zero when the constraint is met during the iterations. Then when a constraint violation occurs, a finite value of α kicks in again. At iteration 80, α^{t2} actually gets 0.79, as seen in Figure 1 (b). Validation results are also shown as dotted curves in Figure 1 (a). Constraint satisfaction in training tend to be generalizable to that in validation set.

4.2.2 Optimization parameter μ. In AL, the optimization parameter μ_k^t can be *any* sufficiently large value. In our experiments, we set μ_n^t to be a constant across all iterations and constraint. We vary the value from $(10, 100, 1K, 10K)$ and see how the constraints are satisfied, over different cost UB reduction rates that ranges from 5% through 50%, see Table 1 for the values used for each objective. As a metric to measure constrained satisfaction, we use relative margin that is defined as $(b^t - C^t)/b^t$. Negative value means constraint violation. In Table 4 and 3, obviously, the more UB values are set aggressively, the less chance the constraints can be satisfied. For both constraints, when UB's are set 5%, all constraints are met even for μ is as small as 10. However, as UB is set up to 30%, only larger

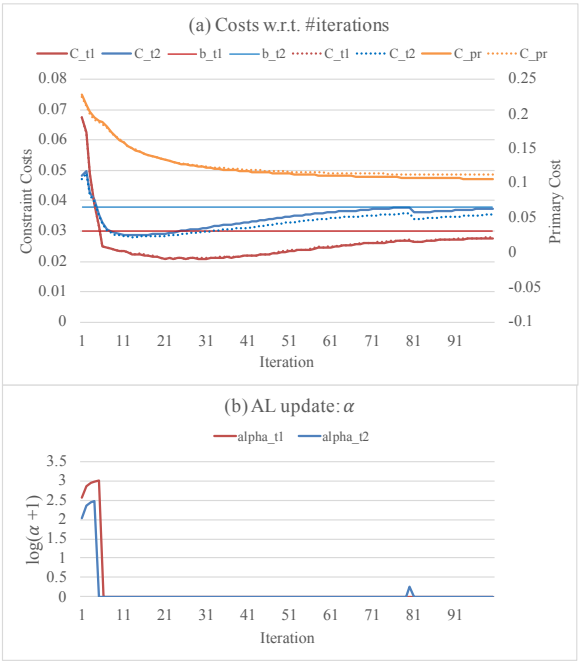

Figure 1: (a) Cost curves in AL-LM. μ is set to $10K$. Solid curves represent training results and dotted validation. Solid lines represent UB's. (b) how α changes over iterations shown in log-scale.

Table 3: Relative margin by UB reduction rates and μ, for $t1$

μ	data	UB reduction rate (%)					
		5%	10%	20%	30%	40%	50%
10000	tr	19.28	13.87	6.40	3.72	2.45	-0.28
	val	21.13	15.30	7.33	4.20	3.24	0.31
1000	tr	17.81	13.22	6.62	0.02	0.70	-0.09
	val	19.05	13.84	6.87	1.00	1.03	0.80
100	tr	18.31	14.02	5.80	-0.06	-0.65	-1.66
	val	18.82	13.90	5.98	0.94	-0.05	-1.17
10	tr	13.08	12.15	6.97	-0.39	-1.56	-3.28
	ts	14.25	13.17	7.08	-0.55	-1.28	-2.98

value of μ, i.e., $\mu = 1K$ or $10K$ can satisfy the constraints. For above 40% reduction, even $\mu = 10K$ suffers from infeasibility. As $\mu = 10K$ satisfies most constraints for the cost reduction rate for the training data, a sensible choice for the value of μ would be at least $10K$.

4.2.3 Cost reduction and NDCG gain. Now, we look at the cost reduction rates and NDCG gains over the unconstrained cases as baseline. Table 5 shows for each objective, how the attained cost reduction rates and NDCG gains vary with different UB settings. Note for the constraints, the cost values are consistent with Table 4 and 3, as they are computed based on values in Table 5.

As we have tighter UB values, cost associated with the constraint is reduced, which improves the NDCG gain. For the primary objective, purchase, the behavior is opposite, which is all expected as we are tightening constraints. An important observation is the

Table 4: Relative margin by UB reduction rates and μ, for $t2$

μ	data	UB reduction rate (%)					
		5%	10%	20%	30%	40%	50%
10000	tr	3.09	2.01	0.65	6.19	-0.98	-0.18
	val	3.34	2.11	1.50	7.22	0.39	0.29
1000	tr	2.16	1.31	0.10	0.07	-0.45	-1.15
	val	1.97	1.51	0.63	0.92	0.61	-0.90
100	tr	2.61	-0.09	0.46	-1.14	-0.97	-2.10
	val	2.57	-0.51	0.96	-0.59	0.00	-1.54
10	tr	0.70	-0.46	-1.57	-1.64	-3.20	-5.26
	ral	0.47	-0.04	-1.55	-1.50	-2.64	-5.24

Table 5: Cost reduction (%) and NDCG-gain for each objective, by different UB's, with μ=10K

		training / upper-bound reduction%						**validation** / upper-bound reduction%					
		5%	10%	20%	30%	40%	50%	5%	10%	20%	30%	40%	50%
	Cost red.	24.28	23.87	26.40	33.72	42.45	49.72	26.13	25.30	27.33	34.20	43.24	50.31
t1	NDCG	1.62	1.57	1.74	2.32	3.06	3.74	1.76	1.66	1.81	2.34	3.16	3.74
	Cost red.	8.09	12.01	20.65	36.19	39.02	49.82	8.34	12.11	21.50	37.22	40.39	50.29
t2	NDCG	0.92	1.68	3.49	6.58	7.09	9.30	1.06	1.87	3.46	6.76	7.21	9.34
pur-	Cost red.	0.02	-0.22	-0.02	-1.25	-2.54	-5.18	0.21	0.18	0.29	-0.82	-1.88	-3.60
chase	NDCG	-0.23	-0.01	-0.21	-0.63	-1.03	-1.58	0.18	0.18	0.11	-0.20	-0.23	-0.96

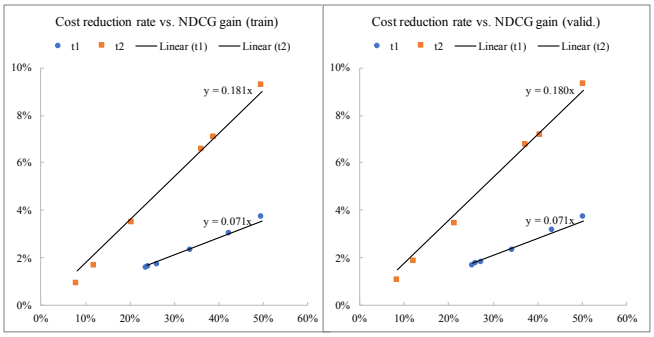

Figure 2: Relationship between cost reduction and NDCG gain

validation results are *consistent* with those of training, which means the constraint satisfaction in training generalizes well at least for the dataset examined. Note constraint $t2$ is over-satisfied with large margin in cost. This is due to the correlation between the target values associated with $t1$ and $t2$, as that of $t1$ is available only if the target value of $t2$ is positive, which partially explains $t1$ is a tighter constraint to be satisfied than $t2$.

The relationship between the cost reduction and the NDCG gains is illustrated in Figure 2. The two metrics are quite consistent and fit well by linear lines. This means, we can estimate cost reduction value for a given NDCG gain requirement, by using the approximately liner relationship. Once we have the cost reduction rate, we can set it as the UB. This observation is quite useful in practice where NDCG gain values would likely be the criteria for offline modeling.

Table 6: Production scale modeling results in %-gain. For over-weighting (OW), 7 models achieve +1% gain criteria out of 144 trials in grid search. Min, max and avg performances of the 7 models are shown. For AL-LM, *one* denotes use of estimated UB's (one-shot model) and *adjust* applying adjustment over the one-shot model. The bold letter shows statistically significance in comparing OW (max) and AL-ML (one).

	pur-	$t1$		$t2$		$t3$		$t4$	
	chase	@5	@22	@5	@22	@5	@22	@5	@22
OW (min)	-0.10	2.59	1.98	1.15	1.00	1.66	1.19	2.01	1.54
OW (max)	-0.03	**3.52**	**2.67**	1.66	1.41	2.33	1.67	2.74	2.08
OW (avg)	-0.05	2.90	2.20	1.27	1.10	1.79	1.30	2.26	1.68
AL-LM (one)	-0.07	3.21	1.83	**4.28**	**3.08**	**3.38**	**2.29**	**4.19**	**2.77**
AL-LM (adj.)	-0.03	2.12	1.26	3.13	2.16	2.73	1.97	3.36	2.64

4.3 Production-scale modeling

4.3.1 Applying AL-ML in the full dataset. In this subsection, we show results on a larger sampled data to simulate a production modeling with more sub-objectives. To this end, we increase the samples to ~1MM search impressions for the training and ~500K search impressions for the hold-out evaluation data sets and use the full four sub-objectives. Each sub-objective is computed by a binary target value, just like the purchase target. The goal of the modeling is to achieve at least 1% gain in the number of products with the positive sub-objective value, i.e., presence of products eligible for some customer benefit in top-5 and 22 ranks, while minimizing the impact on the purchase NDCG. We use relationship between the metrics (top-K) and cost as illustrated in Figure 2 to estimate the values of UB's by using smaller number of samples. In other words, as long as we know the relationship between the metric value and cost value, we can achieve the goal of +1% gain by just setting the UB's. Namely, we choose 14% reduction for b^{t1} and $1/0.181 = 5.5\%$ reduction for b^{t2} and adjust if the constraint is too tight to satisfy, or over-satisfied. Note as we find $t3$ and $t4$ follows very similar pattern as $t2$, we use the same setting as $t2$ to them.

4.3.2 OW method as a baseline. As a baseline method to compare against, we tune models by over-weighting (OW) over the sub-objectives. Basically, in the OW method, we identify search impressions that contain products with the positive sub-objective value for applying over-weightings. We introduce weights on the pairwise cost computation so that we can influence the cost function depending on products that matches a query. For example, if a product has a value one in the target $t1$, i.e., the product is eligible for some benefit, and we want to optimize $t1$ over other sub-objectives, we put high weight on the cost associated with $t1$ so the ranking is more likely to optimized for the search impressions containing products with $t1$ as a feature. When there are multiple sub-objectives, we need to combinatorially tune multiple weighting schemes to concurrently achieve multiple objectives.

The weight values are tuned first by manually finding a reasonably good weighting parameter ranges and run grid search to fine tune. As there are four dimensional space to explore and too time consuming for the problem size, we only search weighting parameters for $t1$ and $t2$; we rely on correlation between $t2$ and $t3$ or $t4$ to

optimize overall metrics. Despite the search space reduction, we end up building 144+ models.

4.3.3 Results. Table 6 shows the results on the evaluation set from OW and AL-LM, which are measured by %-gain with respect to the unconstrained baseline. For OW, only 7 out of 144 trials over the grid are found as feasible solutions. We report *min*, *max* and *avg* among the solutions. For AL-LM, we already have estimates of each constraint from Figure 2. A similar preliminary experiment gives estimate of other cost UB's.

Both methods achieve the 1% gain criteria for all metrics. While AL-LM achieves higher gains for $t2 - t4$ and purchase NDCG is flat, the model shows slight over satisfaction and we apply some adjustment (relax UB's by 20%: 14% reduction for b^{t1} to 11% and 4%), yielding slight improvement on purchase (insignificant) with less margin on $t1$. In terms of efficiency, AL-LM is a clear win as it requires only two model builds, one for getting the baseline cost value and the other by applying cost reduction UB's estimated by smaller sample. We could put efforts on adjustment if the one-shot model has issues, which can be resolved slight tightening or relaxing constraints. OW, on the other hand, requires exploration of the weighting schemes and the success rate is $7/144 = 4.8\%$, which requires 21 trials on average and 61 trials at 95% confidence.

Addition of further sub-objectives, such as search defect and adult products tend to be much less correlated with the constraints we studied in this paper, which means more combinatorial exploration (*exponential* in T) is needed for existing methodologies. AL-LM is the choice for MO modeling, as it only requires setting UB's and some adjustment around the one-shot model: (*linear* in T).

5 CONCLUSION AND FURTHER WORK

In this paper, we introduced an Augmented Lagrangian based method to address challenges with the multi-objective optimization in relevance modeling. Specifically, we incorporated a constraint optimization method into Boosting so that modelers can use it very easily as an extension to the LambdaMART, one of the most popular Boosting methods in this domain. The explicit introduction of constrained optimization is a novel contribution of this paper and thus its application to relevance ranking is novel. Experimental results showed the proposed AL-LM can indeed efficiently resolve the MO problem. The impact of the outcome would be not limited to reducing the model development lead time; modelers can make more effort in developing new features by using the time saved and an automated model refresh can be realizable by incorporating various constraints automatically. In fact, we have been quite successful in apply the multiple objective models built with AL-LM to modeling projects in production.

Our code is currently incorporated in GBM in R. We plan to implement the algorithm into XGBoost and / or LightGBM and make it publicly available.

Although we applied the method on relevance modeling, it is applicable to wider problem domains, i.e., classification and regressions, and other machine learning methods such as neural networks, etc. Extension to wider application and methodology is a future work as well.

REFERENCES

[1] Pia Borlund. 2003. The Concept of Relevance in IR. *J. Am. Soc. Inf. Sci. Technol.* 54, 10 (Aug. 2003), 913–925. https://doi.org/10.1002/asi.10286

[2] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. 1984. *Classification and Regression Trees.* Wadsworth and Brooks, Monterey, CA.

[3] Chris J.C. Burges. 2010. *From RankNet to LambdaRank to LambdaMART: An Overview.* Technical Report. https://www.microsoft.com/en-us/research/publication/from-ranknet-to-lambdarank-to-lambdamart-an-overview/

[4] Christopher J. Burges, Robert Ragno, and Quoc V. Le. 2007. Learning to Rank with Nonsmooth Cost Functions. In *Advances in Neural Information Processing Systems 19*, B. Schölkopf, J. C. Platt, and T. Hoffman (Eds.). MIT Press, 193–200. http://papers.nips.cc/paper/2971-learning-to-rank-with-nonsmooth-cost-functions.pdf

[5] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16).* ACM, New York, NY, USA, 785–794. https://doi.org/10.1145/2939672.2939785

[6] Na Dai, Milad Shokouhi, and Brian D. Davison. 2011. Learning to Rank for Freshness and Relevance. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '11).* ACM, New York, NY, USA, 95–104. https://doi.org/10.1145/2009916.2009933

[7] Anlei Dong, Yi Chang, Zhaohui Zheng, Gilad Mishne, Jing Bai, Ruiqiang Zhang, Karolina Buchner, Ciya Liao, and Fernando Diaz. 2010. Towards Recency Ranking in Web Search. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining (WSDM '10).* ACM, New York, NY, USA, 11–20. https://doi.org/10.1145/1718487.1718490

[8] Jerome H. Friedman. 2001. Greedy function approximation: A gradient boosting machine. *Ann. Statist.* 29, 5 (10 2001), 1189–1232. https://doi.org/10.1214/aos/1013203451

[9] Jacob R. Gardner, Matt J. Kusner, Zhixiang Xu, Kilian Q. Weinberger, and John P. Cunningham. 2014. Bayesian Optimization with Inequality Constraints. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32 (ICML'14).* JMLR.org, II–937–II–945. http://dl.acm.org/citation.cfm?id=3044805.3044997

[10] Ridgeway Greg. 2013. *gbm: Generalized Boosted Regression Models.* https://cran.r-project.org/web/packages/gbm/index.html

[11] José Miguel Hernández-Lobato, Michael A. Gelbart, Ryan P. Adams, Matthew W. Hoffman, and Zoubin Ghahramani. 2016. A General Framework for Constrained Bayesian Optimization Using Information-based Search. *J. Mach. Learn. Res.* 17, 1 (Jan. 2016), 5549–5601. http://dl.acm.org/citation.cfm?id=2946645.3053442

[12] Changsung Kang, Xuanhui Wang, Yi Chang, and Belle Tseng. 2012. Learning to Rank with Multi-aspect Relevance for Vertical Search. In *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining (WSDM '12).* ACM, New York, NY, USA, 453–462. https://doi.org/10.1145/2124295.2124350

[13] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.). Curran Associates, Inc., 3146–3154. http://papers.nips.cc/paper/6907-lightgbm-a-highly-efficient-gradient-boosting-decision-tree.pdf

[14] Mahdi Milani Fard, Quentin Cormier, Kevin Canini, and Maya Gupta. 2016. Launch and Iterate: Reducing Prediction Churn. In *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett (Eds.). Curran Associates, Inc., 3179–3187. http://papers.nips.cc/paper/6053-launch-and-iterate-reducing-prediction-churn.pdf

[15] S. Mizzaro. 1998. How many Relevances in Information Retrieval? *Interacting With Computers* 10, 3 (1998), 305–322.

[16] Phong Nguyen, John Dines, and Jan Krasnodebski. 2017. A Multi-Objective Learning to re-Rank Approach to Optimize Online Marketplaces for Multiple Stakeholders. *CoRR* abs/1708.00651 (2017). arXiv:1708.00651 http://arxiv.org/abs/1708.00651

[17] J. Nocedal and S. Wright. 2006. *Numerical Optimization* (2 ed.). Springer. http://books.google.com.tr/books?id=VbHYoSyelFcC,/bib/nocedal/nocedal2006numerical/%28Springer%20series%20in%20operations%20research%29%20Jorge%20Nocedal%2C%20Stephen%20Wright-Numerical%20Optimization-Springer%20%282006%29.pdf,http://www.bioinfo.org.cn/~wangchao/maa/Numerical_Optimization.pdf

[18] Daria Sorokina. 2016. Amazon Search: The Joy of Ranking Products. https://mlconf.com/mlconf-2016-sf/.

[19] Daria Sorokina and Erick Cantu-Paz. 2016. Amazon Search: The Joy of Ranking Products. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '16).* ACM, New York, NY, USA, 459–460. https://doi.org/10.1145/2911451.2926725

[20] Krysta M. Svore, Maksims N. Volkovs, and Chris J.C. Burges. 2011. Learning to Rank with Multiple Objective Functions, In Proceedings of WWW 2011. https://www.microsoft.com/en-us/research/publication/learning-to-rank-with-multiple-objective-functions/

[21] Lidan Wang, Paul N. Bennett, and Kevyn Collins-Thompson. 2012. Robust Ranking Models via Risk-sensitive Optimization. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '12)*. ACM, New York, NY, USA, 761–770. https://doi.org/10.1145/2348283.2348385

[22] Zhaohui Zheng, Hongyuan Zha, Tong Zhang, Olivier Chapelle, Keke Chen, and Gordon Sun. 2008. A General Boosting Method and its Application to Learning Ranking Functions for Web Search. In *Advances in Neural Information Processing Systems 20*, J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis (Eds.). Curran Associates, Inc., 1697–1704. http://papers.nips.cc/paper/3305-a-general-boosting-method-and-its-application-to-learning-ranking-functions-for-web-search.pdf