Web-Search Ranking with Initialized Gradient Boosted Regression Trees

Ananth Mohan MOHANA@WUSTL.EDU
Zheng Chen ZHENG.CHEN@WUSTL.EDU
Kilian Weinberger KILIAN@WUSTL.EDU
Department of Computer Science & Engineering
Washington University in St. Louis
St. Louis, MO 63130, USA

Editor: Olivier Chapelle, Yi Chang, Tie-Yan Liu

Abstract

In May 2010 Yahoo! Inc. hosted the *Learning to Rank Challenge*. This paper summarizes the approach by the highly placed team *Washington University in St. Louis*. We investigate Random Forests (RF) as a low-cost alternative algorithm to Gradient Boosted Regression Trees (GBRT) (the de facto standard of web-search ranking). We demonstrate that it yields surprisingly accurate ranking results — comparable to or better than GBRT. We combine the two algorithms by first learning a ranking function with RF and using it as *initialization* for GBRT. We refer to this setting as iGBRT. Following a recent discussion by Li et al. (2007), we show that the results of iGBRT can be improved upon even further when the web-search ranking task is cast as classification instead of regression. We provide an upper bound of the Expected Reciprocal Rank (Chapelle et al., 2009) in terms of classification error and demonstrate that iGBRT outperforms GBRT *and* RF on the Microsoft Learning to Rank and Yahoo Ranking Competition data sets with surprising consistency.

Keywords: Ranking, Decision Trees, Boosting, Random Forests

1. Introduction

The success of search engines such as Google, Yahoo! and Bing has lead to an increased interest in algorithms for automated web search ranking. Web search ranking is often treated as a supervised machine learning problem (Burges et al., 2005; Li et al., 2007; Zheng et al., 2007b): each query-document pair is represented by a high-dimensional feature vector and its label indicates the document's degree of relevance to the query. A machine learning algorithm is trained to predict the relevance from the feature vector, and during test time the documents are ranked according to these predictions.

The past years have seen many different approaches to web search ranking, including adaptations of support vector machines (Joachims, 2002; Chapelle and Keerthi, 2010), neural networks (Burges et al., 2005) and gradient boosted regression trees (GBRT) (Zheng et al., 2007b). The latter has arguably established itself as the current state-of-the-art learning paradigm (Li et al., 2007; Gao et al., 2009; Burges, 2010).

Irrespective of which learning algorithm is used, the various ranking settings fall into three categories: point-wise, pair-wise, and list-wise. Point-wise algorithms predict the relevance of a document to a query by minimizing a regression loss (e.g. the squared loss). Pair-wise approaches learn a classifier that predicts if one document is more relevant than another; these approaches include RankBoost (Freund et al., 2003), FRank (Tsai et al., 1999), and GBRank (Zheng et al., 2007a). List-wise approaches, such as AdaRank (J., 2007), PermuRank (Xu et al., 2008), tend to iteratively optimize a specialized ranking performance measure, for example NCDG.

In this paper we describe the point-wise ranking approach of the team *Washington University in St. Louis* for the *Yahoo Learning To Rank Challenge* in May 2010. Most of the decisions we made throughout the competition were heavily influenced by our limited computational resources. We focussed on developing a light-weight algorithm that can be trained fully on a single multi-core desktop within reasonable amount of time.

We investigate the use of Random Forests (RF) (Breiman, 2001) for web search ranking and demonstrate that it can be a powerful low-cost alternative to GBRT. Although RF is also based on tree averaging, it has several clear advantages over GBRT: 1. It is particularly insensitive to parameter choices; 2. It is known to be very resistant to overfitting; 3. It is "embarrassingly" parallelizable. In addition, we demonstrate on several real world benchmark data sets that RF can match (or outperform) GBRT with surprising consistency.

As a second contribution, we address a particular weakness of GBRT. In gradient boosting, there exists an inherent trade-off between the step-size and early stopping. To obtain the true global minimum, the step-size needs to be infinitesimally small and the number of iterations very large. Of course, this is unrealistic and common practice is to use a reasonably small step-size (≈ 0.1) with roughly 1000 iterations. We show that, as RF often outperforms GBRT and is very resistant to overfitting, its predictions can be used as a starting-point for the gradient boosting. In this setting GBRT starts-off at a point that is very close to the global minimum and merely refines the already good predictions.

We refer to the resulting algorithm as *initialized Gradient Boosted Regression Trees (iGBRT)*. iGBRT is insensitive to parameters and when web-search ranking is cast as a classification problem, as suggested by Li et al. (2007), it outperforms both GBRT and Random Forests on *all* Yahoo and Microsoft benchmark datasets. As a final contribution, in addition to the empirical evaluation, we provide an upper bound of the ERR metric with respect to the classification error of a ranker.

This paper is organized as follows: In section 2 we briefly review the web-searching ranking setting and define necessary notation. In section 3 we introduce RF. In section 4 we introduce GBRT. Both algorithms are combined in section 5 as initialized gradient boosted regression trees (iGBRT). In section 6 we review the classification setting introduced by Li et al. (2007), adapt iGBRT to this framework and provide an upper bound of the ERR in terms of the classification error. Finally, we provide an extensive evaluation of all algorithms in section 7.

2. Notation and Setup

We assume that we are provided with data of triples $D = \{(\mathbf{x}_1, q_1, y_1), \ldots, (\mathbf{x}_n, q_n, y_n)\}$, consisting of documents ($\mathbf{x}_i \in \mathcal{R}^f$), queries ($q_i \in \{1, \ldots, n_q\}$) and labels ($y_i \in \{0, \ldots, 4\}$). The label y_i indicates to what degree document $\mathbf{x}_i$ is relevant to the query q_i and ranges from $y_i = 0$ ("irrelevant") to $y_i = 4$ ("perfect match"). There are fewer queries than samples

Algorithm 1 Random Forests

```
Input: $D = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, Parameters: $K : 0 < K \leq f$, $M_{RF} : M_{RF} > 0$
for $t = 1$ to $M_{RF}$ do
    $D_t \subseteq D$                      #Sample with replacement, $|D_t| = |D|$.
    $h_t \leftarrow \mathrm{Cart}(D_t, K, \infty)$    #Build full ($d = \infty$) Cart with $K \leq f$ randomly chosen features at each split.
end for
$T(\cdot) = \frac{1}{M_{RF}} \sum_{t=1}^{M_{RF}} h_t(\cdot)$.
return $T(\cdot)$
```

($n_q < n$). In this paper, our algorithmic setup is not affected by the number of queries. We assume that a document vector $\mathbf{x}_i$ is a f dimensional feature vector that incorporates all the sufficient statistics about the query and the document as features. For example, one feature could be "the number of occurrences of the query in the document". To simplify notation we will assume that all documents belong to the same query, *i.e.* $n_q = 1$ and with a slight abuse of notation let $D = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$. However, the techniques work for sets with multiple queries, and in fact the data we use for experiments does contain many queries.

Point-wise machine-learned ranking trains a predictor $T(\cdot)$ such that $T(\mathbf{x}_i) \approx y_i$. If $T(\cdot)$ is accurate, the ordering π_h of all documents according to values of $T(\mathbf{x}_i)$ should be close to the desired ordering π_y according to y_i. We can evaluate the quality of the ranking function either with the *root mean squared error* (RMSE) or with ranking specific metrics such as *normalized discounted cumulative gain* (NDCG) (Järvelin and Kekäläinen, 2002) or *expected reciprocal rank* (ERR) (Chapelle et al., 2009). (Please note that for RMSE *lower* values are better, whereas for NDCG and ERR *higher* values indicate better performance.)

All the algorithms in this paper are based on Classification and Regression Trees (CART) (Breiman, 1984). We assume that we have an efficient implementation of a slightly modified version of CART that greedily builds a regression tree to minimize the squared-loss, but at each split uniformly samples k features and only evaluates those as candidates for splitting.

$$\mathrm{Cart}(S, k, d) \approx \operatorname*{argmin}_{h \in \mathcal{T}_d} \sum_{(\mathbf{z}_i, r_i) \in S} (h(\mathbf{z}_i) - r_i)^2. \tag{1}$$

The three parameters of our CART algorithm (1) are: 1. a set $S \subseteq D$; 2. an integer $k \leq f$ which determines how many uniformly picked features are considered at each split; 3. an integer $d > 0$ that defines the maximum depth of the resulting tree (in (1) $\mathcal{T}_d$ denotes the set of all CART trees of maximum depth d).

The *Yahoo Learning to Rank Challenge* was based on two data sets of unequal size: Set 1 with 473134 and Set 2 with 19944 documents. We use the smaller Set 2 for illustration throughout the paper. In section 7 we report a thorough evaluation on both Yahoo data sets and the five folds of the Microsoft MSLR data set.

3. Random Forests

In this section we briefly introduce Random Forests (Breiman, 2001). The fundamental concept underlying Random Forests is bagging (Breiman, 1984). In bagging, a learning algorithm is applied multiple times to a subset of D and the results are averaged. Each time

Algorithm 2 Gradient Boosted Regression Trees (Squared Loss)

```
Input: data set $D = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, Parameters: $\alpha$, $M_B$, $d$
Initialization: $r_i = y_i$ for $i = 1$ to $n$
for $t = 1$ to $M_B$ do
    $T_t \leftarrow \text{Cart}(\{(\mathbf{x}_1, r_1), \ldots, (\mathbf{x}_n, r_n)\}, f, d)$   #Build Cart of depth $d$, with all $f$ features and targets $\{r_i\}$
    for $i = 1$ to $n$ do
        $r_i \leftarrow r_i - \alpha T_t(\mathbf{x}_i)$   #Update the residual of each sample $\mathbf{x}_i$.
    end for
end for
$T(\cdot) = \alpha \sum_{t=1}^{M_{RF}} T_t(\cdot)$.   #Combine the Regression Trees $T_1, \ldots, T_{M_B}$.
return $T(\cdot)$
```

the algorithm is trained, $n = |D|$ data points are sub-sampled with replacement from D, so that the individual classifiers vary slightly. This process reduces overfitting by averaging classifiers that are trained on different data sets from the same underlying distribution. Random Forests is essentially bagging applied to CART with full depth ($d = \infty$), where at each split only K uniformly chosen features are evaluated to find the best splitting point. The construction of a single tree is independent from earlier trees, thus making Random Forests an inherently parallel algorithm. Algorithm 1 implements the regression version of the Random Forests algorithm. Only two parameters need to be tuned. M_{RF} specifies the number of trees in the forest and K determines the number of features that each node considers to find the best split. As Random Forests is based on bagging, it does not overfit with increasing M_{RF}, so we set it to be very large ($M_{RF} = 10000$). The algorithm is only sensitive to one parameter, K. A common rule of thumb is to set K to 10% of the number of features (*i.e.* $K = 0.1f$). We follow this rule throughout the paper.

4. Gradient Boosted Regression Trees

Similar to Random Forests, Gradient Boosted Regression Trees (GBRT) (Friedman, 2001) is a machine learning technique that is also based on tree averaging. However, instead of training many full ($d = \infty$) high variance trees that are averaged to avoid overfitting, GBRT sequentially adds small trees ($d \approx 4$), each with high bias. In each iteration, the new tree to be added focuses explicitly on the documents that are responsible for the current remaining regression error. Empirical results have shown that GBRT is especially well-suited for web-search ranking (Zheng et al., 2007b; Burges, 2010). In fact, all the winning teams of the *Yahoo Learning to Rank Challenge* used some variation of GBRT[1].

Let $T(\mathbf{x}_i)$ denote the current prediction of sample $\mathbf{x}_i$. Furthermore, assume we have a continuous, convex and differentiable loss function $\mathcal{L}(T(\mathbf{x}_1), \ldots, T(\mathbf{x}_n))$ which reaches its minimum if $T(\mathbf{x}_i) = y_i$ for all $\mathbf{x}_i$. Throughout the paper we use the square loss: $\mathcal{L} = \frac{1}{2}\sum_{i=1}^{n}(T(\mathbf{x}_i) - y_i)^2$. In each iteration, a new tree $h(\cdot)$ is added to the existing classifier $T(\cdot)$. The best $h(\cdot)$ is found with a first-order Taylor expansion of $\mathcal{L}(T + h_t)$, which is minimized with respect to $h_t(\cdot)$. See Zheng et al. (2007b) for a detailed derivation.

[1] See the official ICML workshop homepage http://learningtorankchallenge.yahoo.com/workshop.php.

Algorithm 3 Initialized Gradient Boosted Regression Trees (Squared Loss)

```
Input: data set $D = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, Parameters: $\alpha$, $M_B$, $d$, $K_{RF}$, $M_{RF}$
$F \leftarrow RandomForests(D, K_{RF}, M_{RF})$
Initialization: $r_i = y_i - F(\mathbf{x}_i)$ for $i = 1$ to $n$
for $t = 1$ to $M_B$ do
  $T_t \leftarrow \text{Cart}(\{(\mathbf{x}_1, r_1), \ldots, (\mathbf{x}_n, r_n)\}, f, d)$  #Build Cart of depth $d$, with all $f$ features, and targets $\{r_i\}$.

  for $i = 1$ to $n$ do
    $r_i \leftarrow r_i - \alpha T_t(\mathbf{x}_i)$            #Update the residual of each sample $\mathbf{x}_i$.
  end for
end for
$T(\cdot) = F(\cdot) + \alpha \sum_{t=1}^{M_B} T_t(\cdot)$.         #Combine the Regression Trees $T_1, \ldots, T_M$ with the RF $F$.
return $T(\cdot)$
```

Intuitively, GBRT performs gradient descent in the instance space $\mathbf{x}_1, \ldots, \mathbf{x}_n$ *i.e.* during each iteration the current prediction $T(\mathbf{x}_i)$ is updated with a gradient step

$$T(\mathbf{x}_i) \leftarrow T(\mathbf{x}_i) - \alpha \frac{\mathcal{L}}{T(\mathbf{x}_i)}, \tag{2}$$

where $\alpha > 0$ denotes the learning rate. The negative gradient $-\frac{\partial \mathcal{L}}{\partial T(\mathbf{x}_i)}$ is approximated with the prediction of the regression tree $h_t(\mathbf{x}_i)$ which satisfies:

$$h_t \approx -\underset{h \in \mathcal{T}_d}{\text{argmin}} \sum_{i=1}^{n} \left(h_t(\mathbf{x}_i) - r_i\right)^2, \quad \text{where: } r_i = \frac{\partial \mathcal{L}}{\partial T(\mathbf{x}_i)}. \tag{3}$$

In the case where $\mathcal{L}$ is the squared loss, the gradient for a document $\mathbf{x}_i$ becomes the residual from the previous iteration, *i.e.* $r_i = y_i - T(\mathbf{x}_i)$. Each iteration t, we use the standard CART algorithm (1), with $K = f$, to find a solution to (3). GBRT depends on three parameters: The learning rate $\alpha > 0$, the tree-depth d, and the number of iterations M_B. Based on experiments from previous research (Friedman, 2001; Zheng et al., 2007b), we set $d = 4$ and pick M_B and α with a validation data set. Smaller learning rates tend to result in better accuracy but require more iterations. If M_B is too large, the algorithm starts overfitting. Figure 1 shows how Boosting compares to RF on the *Yahoo Learning to Rank Challenge* data Set 2 for various settings of α and M_B. The figure shows a clear trend that RF outperforms *all* settings of GBRT according to *all* three metrics (ERR, NDCG, RMSE). Although not shown here, boosting for even more iterations, up to $M_B = 5000$, did not change that result at any time — on the contrary, GBRT started to overfit, widening the performance gap between RF and GBRT. Similar results were obtained on the much larger Yahoo Set 1, whereas the results on the Microsoft Learning to Rank data set were mixed with no clear winner (see Table 3 in section 7). RF was averaged over $M_{RF} = 10000$ iterations and $K = 70$.

5. Initialized Gradient Boosted Regression Trees

GBRT, as described in the previous section, is traditionally initialized with the all-zero function $T_0(\mathbf{x}_i) = 0$. Consequently, the initial residual is $r_i = y_i$. As the loss function $\mathcal{L}$ is

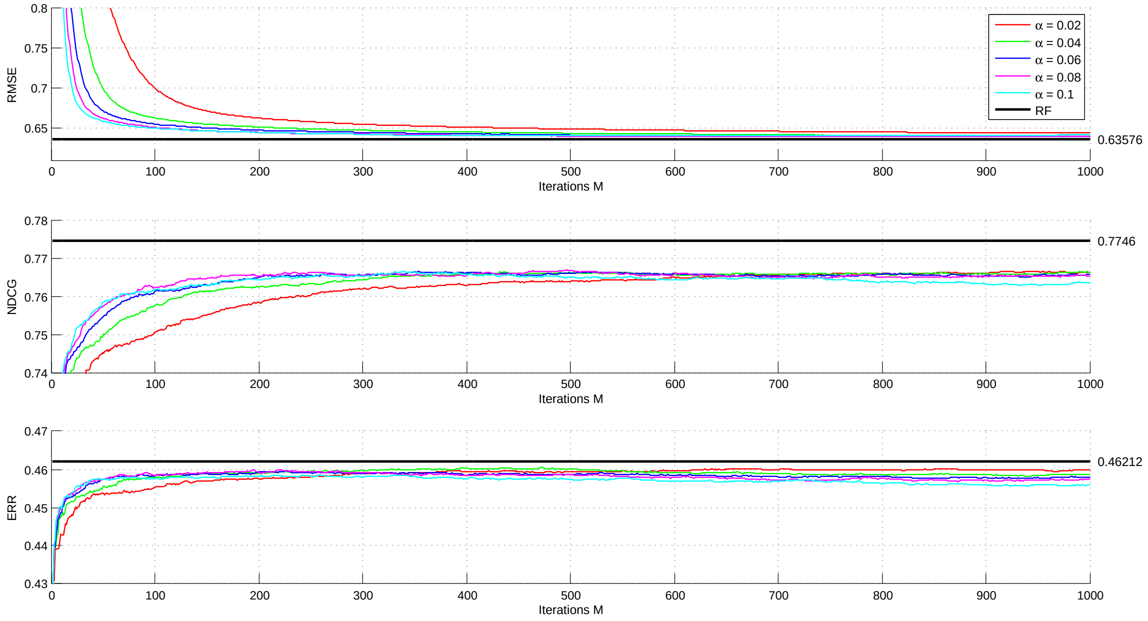

Figure 1: Results of GBRT (with varying step-size α) compared to RF (bold black line) on the Yahoo Set 2. RF outperform boosted regression trees with respect to RMSE, ERR and NDCG.

M_{RF}	0	100	1000	10000
RMSE	0.64132	0.63416	0.63097	**0.63047**
ERR	0.45669	0.45956	0.46170	**0.46257**
NDCG	0.76575	0.77080	0.77712	**0.77805**

Table 1: Performance of iGBRT with varying initializations on Yahoo Set 2.

convex, the gradient descent approach from GBRT should converge to its global minimum irrespective of its initialization (with an appropriate learning-rate α) (Shor, 1970). However these theoretical results tend to not hold in practice for two reasons: 1. in each iteration the gradient is only *approximated*; 2. for true convergence, the learning-rate α needs to be infinitesimally small — requiring an unrealistically large number of iterations $M_B \gg 0$. We therefore propose to *initialize* GBRT with the predictions of RF from section 3. RF is a good initialization for several reasons: 1. RF is known to be very resistant towards overfitting and therefore makes a good optimization starting point; 2. RF is insensitive to parameter settings and does not require additional parameter tuning.

Algorithm 3 shows the pseudo-code implementation of iGBRT. The main differences to GBRT (Algorithm 2) lie in the initial settings of r_i, which are set to the residual of the RF predictions: $r_i \leftarrow y_i - F(\mathbf{x}_i)$, and the final boosted classifier which is added to the initial

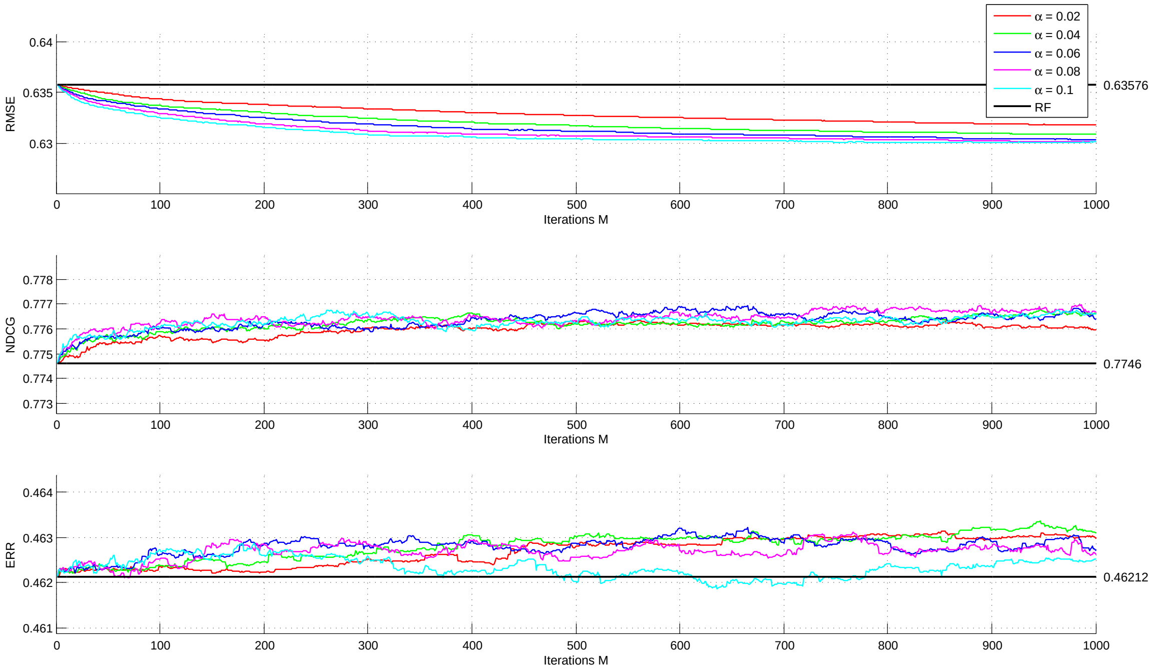

Figure 2: Results of iGBRT on Yahoo test set 2. The predictions were initialized with the predictions of Random Forests (RF). iGBRT improves over RF on all metrics, with some overfitting in the settings with large learning-rates ($\alpha = 0.1$).

results of RF. Figure 2 shows the traces of iGBRT under various step-sizes (on the test partition of yahoo Set 2). In contrast to the standard GBRT, iGBRT improves consistently over Random Forests. In fact, iGBRT outperforms RF and GBRT for all settings of α and $M_B \leq 1000$ in RMSE, NDCG, and ERR (except for a very brief period in ERR with $\alpha = 0.1$).

Table 1 shows ranking results of iGBRT, under varying amounts of trees M_{RF} for the Random Forests initialization. The column with $M_{RF} = 0$ is identical to standard GBRT. The number of boosting iterations M_B was chosen on a validation data set (up to $M_B = 1000$). The table shows clearly that the boosted results were heavily influenced by their initialization. Strikingly, even with only 100 averaged trees as initialization, iGBRT outperforms standard GBRT ($M_{RF} = 0$) with respect to all three metrics.

6. Classification vs. Regression

So far, all our algorithms used regression to approximate the relevance of a document. Recently, Li et al. (2007) proposed a learning to rank paradigm that is based on classification instead of regression. Instead of learning a function $T(\mathbf{x}_i) \approx y_i$, the authors utilize the fact that the original relevance scores are discrete, $y_i \in \{0, 1, 2, 3, 4\}$, and generate four binary classification problems indexed by $c = 1, \ldots, 4$. The c^{th} classification problem predicts if the document is *less* relevant than c. More formally, we denote the binary label of document $\mathbf{x}_i$ for problem $c \in \{1, \ldots, 4\}$ as $b_i^c \equiv (y_i < c)$. For each of these binary classification problems, we train a classifier $T^c(\cdot)$. We carefully choose classifiers $T^c(\cdot)$ to return well defined probabilities (*i.e.* $0 \leq T^c(\cdot) \leq 1$) and $T^c(\mathbf{x})$ can be interpreted as the probability of document $\mathbf{x}$ being less relevant than c. More formally, $T^c(\mathbf{x}) = P(\mathrm{rel}(\mathbf{x}) < c)$. If we define the constant functions $T^0(\cdot) = 0$ and $T^5(\cdot) = 1$ (by definition relevance is non-negative and all documents are less relevant than 5), we can combine all classifiers $T^0, \ldots, T^5$ to compute the probability that a document $\mathbf{x}_i$ has a relevance of $r \in \{0, \ldots, 4\}$:

$$\begin{aligned} P(\mathrm{rel}(\mathbf{x}_i) = r) &= P(\mathrm{rel}(\mathbf{x}_i) < r + 1) - P(\mathrm{rel}(\mathbf{x}_i) < r) \\ &= T^{r+1}(\mathbf{x}_i) - T^r(\mathbf{x}_i). \end{aligned}$$

Li et al. (2007) show that GBRT is well-suited for this setup. Regression trees, minimizing the squared-loss, predict the average label of documents at a leaf. If these contain only binary labels $\{0, 1\}$, the predictions are within the interval $[0, 1]$. The same holds for Random Forests, which are essentially averaged regression trees. We can therefore use RF and iGBRT as binary classifiers for this framework[2]. We evaluate the classification paradigm in the following section.

In an attempt to explain our empirical results, which clearly favor classification over regression, we show that the ERR error is bounded by the classification error in Appendix A. Our current bound shows a clear relationship between ERR and classification performance, however is probably too loose to be of considerable practical value.

Theorem 1 *Given n documents indexed by $\{1, \cdots, n\}$. Suppose a classifier, assigns a relevance score to each document, denoted by $\hat{y}_1, \ldots, \hat{y}_n \in \{0, 1, 2, 3, 4\}$. A ranker, π, ranks documents according to $\hat{y}_i$ such that $\pi(i) < \pi(j)$ if $\hat{y}_i > \hat{y}_j$ (ties are broken arbitrarily). Let g be a perfect ranker and let the ERR scores of g and π be ERR_g and ERR_π, respectively. The ERR error of π, $ERR_g - ERR_\pi$, is bounded by the square root of the classification error:*

$$ERR_g - ERR_\pi \leq \frac{15\pi}{16\sqrt{6}} \sqrt{\sum_{i=1}^{n} 1_{y_i \neq \hat{y}_i}}$$

7. Results

We evaluate all algorithms on several data sets from the Yahoo Ranking competition (two sets from different countries) and the five splits of the Microsoft MSLR data sets[3]. Both data sets contain pre-defined train/validation/test splits. Table 2 summarizes various statistics about all the data sets.

We experimented with several ways to deal with missing features (which were present in all data sets): splitting three ways during tree construction and substituting missing

[2] Depending on the step-size, there might be small violations of the probability assumptions in the case of boosting. However, in our experience this does not seem to hurt the performance.

[3] http://research.microsoft.com/en-us/projects/mslr/

	Yahoo LTRC		MSLR Folds				
TRAIN	Set 1	Set 2	F1	F2	F3	F4	F5
# Features	700	700	136	136	136	136	136
# Documents	473134	34815	723412	716683	719111	718768	722602
# Queries	19944	1266	6000	6000	6000	6000	6000
Avg # Doc per Query	22.723	26.5	119.569	118.447	118.852	118.795	119.434
% Features Missing	0.68178	0.67399	0.37228	0.37331	0.37263	0.37163	0.37282

TEST	Set 1	Set 2	F1	F2	F3	F4	F5
# Documents	165660	103174	241521	241988	239093	242331	235259
# Queries	6983	3798	6000	6000	6000	6000	6000
Avg # Doc per Query	22.723	26.165	119.761	119.994	118.547	120.167	116.6295
% Features Missing	0.68113	0.67378	0.37368	0.36901	0.37578	0.37204	0.37215

Table 2: Statistics of the Yahoo Competition and Microsoft Learning to Rank data sets.

ERR	Regr./	Yahoo LTRC		MSLR Folds				
method	Class.	Set 1	Set 2	F1	F2	F3	F4	F5
GBRT	R	0.45304	0.45669	0.35914	0.35539	0.35579	0.36039	0.37076
RF	R	0.46349	0.46212	0.35481	0.35458	0.34775	0.35853	0.36853
iGBRT	R	0.46301	**0.46303**	0.35787	0.35985	0.35383	0.36491	0.37422
GBRT	C	0.45448	0.46008	**0.36264**	0.36168	**0.35990**	0.36498	0.37549
RF	C	0.46308	0.46200	0.35868	0.35677	0.35003	0.36364	0.37052
iGBRT	C	**0.46360**	0.46246	0.36232	**0.36198**	0.35486	**0.36744**	**0.37430**

NDCG	Regr./	Yahoo LTRC		MSLR Folds				
method	Class.	Set 1	Set 2	F1	F2	F3	F4	F5
GBRT	R	0.76991	0.76587	0.47958	0.48014	0.47477	0.48302	0.48705
RF	R	0.79575	0.77552	0.47493	0.48105	0.47177	0.48238	0.49027
iGBRT	R	0.79575	**0.77725**	0.47863	0.48847	0.47842	0.48927	**0.49688**
GBRT	C	0.77246	0.77132	**0.48385**	0.48477	0.47705	0.48908	0.49383
RF	C	0.79544	0.77373	0.47761	0.48202	0.47208	0.48563	0.48935
iGBRT	C	**0.79672**	0.77591	0.48366	**0.48858**	**0.47868**	**0.49084**	0.49581

Table 3: Performance of Gradient Boosted Regression Trees (GBRT), Random Forests (RF) and Initialized Gradient Boosted Regression Trees (iGBRT). All results are evaluated in ERR (upper table) and NDCG (lower table). We investigated both, the regression setting (R - in second column) and classification (C). The parameters were set by cross validation on the pre-defined validation data.

features with zeros, infinity, the mean or the median feature value. Results on validation splits showed that substituting zeros for missing values tends to outperform the alternative approaches across most data sets. We trained the respective rankers on the training set, then ranked the validation and test sets. The performance of the learners are judged by ERR and NDCG. For Boosting, we used learning rates $\alpha \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$. The reported numbers are the performances from the test sets, with the parameter settings that best performed on the validation sets. For RF, we used the rule of thumb and fixed $K = 0.1f$ and $M_{RF} = 10000$.

Running Time	RF	RF	GBRT	iGBRT
Iterations M	10000	1000	1000	10000/500
Regression	130m	16m	103m	181m
Classification	768m	77m	388m	842m

Table 4: Run Times (in minutes) for RF, GBRT and iGBRT on the Yahoo Ranking Competition Set 2.

We performed all experiments on a standard workstation with 8 cores[4]. Table 4 summarizes the training times. All implementations were parallelized. The RF implementation distributed the construction of the M_{RF} trees onto the various cores, during boosting the splits were performed by different threads in parallel. With a comparable number of trees ($M_{RF} = M_B$), RF was by far the fastest algorithm despite that its tree-sizes are much larger. This can be attributed to the fact that RF scales linearly with the number cores and resulted in a much better CPU utilization. For best possible ranking results we used RF with $M_{RF} = 10000$, which was a bit slower than GBRT. Although the differences between $M_{RF} = 10000$ and $M_{RF} = 1000$ were not necessarily significant, we included them as they did matter for the competition leaderboard. iGBRT was the slowest algorithm as it involves both RF and GBRT sequentially (given our limited resources, we set $M_B = 500$). In the classification setting, boosting became faster than RF as the different classification problems could be learned in parallel and the 8 cores were utilized more effectively. Both boosting algorithms required additional time for the parameter search by cross validation. Our own open-source C++ implementation of all algorithms and links to the data sets are available at the url: http://research.engineering.wustl.edu/~amohan/.

In addition to regression, we also used the classification setting from section 6 on all three algorithms (iGBRT, RF and GBRT). Table 3 summarizes the results under ERR and NDCG. We observed the following trends: 1. Classification reliably outperforms regression across all methods. 2. iGBRT (with classification) outperforms RF and GBRT on most of the data sets according to both ERR and NDCG. 3. RF (classification) yielded better results than GBRT on all data sets except MSLR Fold 3. This is particularly impressive, as no parameter sweep was performed for RF. In the Yahoo Competition iGBRT(C) would have achieved 11^{th} place on Set 1 and iGBRT(R) 4^{th} place on Set 2 (with differences in ERR from the winning solution on the order of 10^{-3} and 10^{-4} respectively). Please note that the higher ranked submissions tended to average over millions of trees and were trained for weeks on over hundred computers (Burges, 2010), whereas our models could be trained on a single desktop in one day.

8. Conclusion

In this paper we compared three algorithms for machine learned web-search ranking in a regression and classification setting. We showed that Random Forests, with parameters picked according to simple rules of thumb, is a very competitive algorithm and reliably outperforms tuned Gradient Boosted Regression Trees. We introduced initialized gradient boosted regression trees (iGBRT), which uses GBRT to further refine the results of Random Forests. Finally, we demonstrated that classification tends to be a better paradigm for

[4] Intel Xeon L5520@2.27GHz

web-search ranking than regression. In fact, iGBRT in a classification setting consistently achieves state-of-the-art performance on all publicly available web-search data sets that we are aware of.

Appendix A.

Let us define a ranker as a bijective function $\pi : \{1, \cdots, n\} \rightarrow \{1, \cdots, n\}$, which maps a given document's index i into its ordering position π_i. Also, let us denote the inverse mapping of π by $\sigma_i = \sigma(i) = \pi^{-1}(i)$.

Expected Reciprocal Rank (ERR) (Chapelle et al., 2009) was designed to measure the performance for web-search ranking algorithms. It simulates a person who looks through the output of a web-search engine. The model assumes that the web-surfer goes through the documents in order of increasing rank and stops the moment he/she is satisfied. $R(y_i)$ denotes the probability that the person is satisfied with the i^{th} result. The ranker obtains a payoff of $\frac{1}{i}$ if the person is satisfied with the document in the i^{th} position. ERR computes the expected payoff under this model. The formal definition is as follows:

$$ERR_\pi = \sum_{i=1}^{n} C_\pi(i) R(y_{\sigma_i}) = \sum_{i=1}^{n} C_\pi(\pi_i) R(y_i) \tag{4}$$

where:

$$C_\pi(i) = \frac{1}{i} \prod_{j=1}^{i-1} (1 - R(y_{\pi^{-1}(j)})) = \frac{1}{i} \prod_{j=1}^{i-1} (1 - R(y_{\sigma_j})) \tag{5}$$

and $R(y) = \frac{2^y - 1}{16}$. Here $C_\pi(i)$ is a product of the payoff $\frac{1}{i}$ and the probability that the surfer was not satisfied with all previous documents, $\prod_{j=1}^{i-1}(1 - R(y_{\sigma_j}))$.

Definition 1 ***Perfect ranker:*** *A ranker, g is a perfect ranker, if it ranks documents according to their original label y_i, i.e. $g(i) < g(j)$ if $y_i > y_j$. When $y_i = y_j$, then document i and document j are arbitrarily ranked. Also denote the inverse mapping of g by $\phi_i = \phi(i) = g^{-1}(i)$.*

Now we denote the corresponding ERR score of the perfect ranker g as ERR_g. Then for a given ranker π, the ERR error is $ERR_g - ERR_\pi$. Theorem 1 states that the ERR error can be bounded by the classification error of the documents. The proof is below:

Proof This proof follows a line of reasoning inspired by Li et al. (2007). By the definition of ERR we have

$$\begin{aligned} ERR_\pi &= \sum_{i=1}^{n} C_\pi(\pi_i) R(y_i) = \sum_{i=1}^{n} C_\pi(\pi_i) \frac{2^{y_i} - 1}{16} \\ &= \sum_{i=1}^{n} C_\pi(\pi_i) \frac{2^{\hat{y}_i} - 1}{16} + \sum_{i=1}^{n} C_\pi(\pi_i) \frac{2^{y_i} - 2^{\hat{y}_i}}{16}. \end{aligned} \tag{6}$$

According to the rearrangement inequality (because π is the ideal ranking for $\hat{y}_i$) we can show that,

$$
\begin{aligned}
\sum_{i=1}^{n} C_\pi(i) R(\hat{y}_{\sigma_i}) &\geq \sum_{i=1}^{n} C_g(i) R(\hat{y}_{\phi_i}) \\
\sum_{i=1}^{n} C_\pi(\pi_i) \frac{2^{\hat{y}_i}-1}{16} &\geq \sum_{i=1}^{n} C_g(g_i) \frac{2^{\hat{y}_i}-1}{16}.
\end{aligned} \tag{7}
$$

Combining (6) and (7) leads to

$$
\begin{aligned}
ERR_\pi &\geq \sum_{i=1}^{n} C_g(g_i) \frac{2^{\hat{y}_i}-1}{16} + \sum_{i=1}^{n} C_\pi(\pi_i) \frac{2^{y_i}-2^{\hat{y}_i}}{16} \\
&= \sum_{i=1}^{n} C_g(g_i) \frac{2^{y_i}-1}{16} - \sum_{i=1}^{n} C_g(g_i) \frac{2^{y_i}-2^{\hat{y}_i}}{16} + \sum_{i=1}^{n} C_\pi(\pi_i) \frac{2^{y_i}-2^{\hat{y}_i}}{16} \\
&= ERR_g + \sum_{i=1}^{n} (C_\pi(\pi_i) - C_g(g_i)) \frac{2^{y_i}-2^{\hat{y}_i}}{16}.
\end{aligned}
$$

Consequently we obtain

$$
ERR_g - ERR_\pi \leq \frac{1}{16} \sum_{i=1}^{n} (C_\pi(\pi_i) - C_g(g_i))(2^{y_i} - 2^{\hat{y}_i}).
$$

Applying Cauchy-Schwarz inequality leads to

$$
\begin{aligned}
ERR_g - ERR_\pi &\leq \frac{1}{16} \left(\sum_{i=1}^{n} (C_\pi(\pi_i) - C_g(g_i))^2 \right)^{1/2} \left(\sum_{i=1}^{n} (2^{y_i} - 2^{\hat{y}_i})^2 \right)^{1/2} \\
&\leq \frac{15\pi}{16\sqrt{6}} \sqrt{\sum_{i=1}^{n} 1_{y_i \neq \hat{y}_i}},
\end{aligned}
$$

where we use the fact that $\sum_{i=1}^{n} (C_\pi(\pi_i) - C_g(g_i))^2 \leq \sum_{i=1}^{n} \frac{1}{i^2} \leq \frac{\pi^2}{6}$, and $2^4 - 2^0 = 15$. ■

References

L. Breiman. *Classification and regression trees.* Chapman & Hall/CRC, 1984.

L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

C. Burges. From RankNet to LambdaRank to LambdaMART: An Overview. *Microsoft Research Technical Report MSR-TR-2010-82*, 2010.

C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*, ICML '05, pages 89–96, New York, NY, USA, 2005. ACM.

O. Chapelle and S. S. Keerthi. Efficient algorithms for ranking with svms. *Inf. Retr.*, 13: 201–215, June 2010. ISSN 1386-4564.

O. Chapelle, D. Metlzer, Y. Zhang, and P. Grinspan. Expected reciprocal rank for graded relevance. In *Proceeding of the 18th ACM conference on Information and knowledge management*, CIKM '09, pages 621–630, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-512-3.

Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *J. Mach. Learn. Res.*, 4:933–969, 2003. ISSN 1532-4435.

J. H. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, pages 1189–1232, 2001. ISSN 0090-5364.

J. Gao, Q. Wu, C. Burges, K. Svore, Y. Su, N. Khan, S. Shah, and H. Zhou. Model adaptation via model interpolation and boosting for web search ranking. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2*, pages 505–513. Association for Computational Linguistics, 2009.

Xu. J. A boosting algorithm for information retrieval. In *Proceedings of the 30th Annual ACM Conference on Research and Development in Information Retrieval*, 2007.

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

T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM International conference on Knowledge discovery and data mining (SIGKDD)*, page 142. ACM, 2002.

P. Li, C. Burges, and Q. Wu. Learning to rank using classification and gradient boosting. In *Proceedings of the International Conference on Advances in Neural Information Processing Systems (NIPS)*, 2007.

N. Shor. Convergence rate of the gradient descent method with dilatation of the space. *Cybernetics and Systems Analysis*, 6(2):102–108, 1970.

M. Tsai, T. Liu, H. Chen, and W. Ma. Frank: A ranking method with fidelity loss. Technical Report MSR-TR-2006-155, Microsoft Research, November 1999.

J. Xu, T. Liu, M. Lu, H. Li, and W. Ma. Directly optimizing evaluation measures in learning to rank. In *Proceedings of the 31th Annual ACM Conference on Research and Development in Information Retrieval.* ACM Press, 2008.

Z. Zheng, H. Zha, K. Chen, and G. Sun. A regression framework for learning ranking functions using relative relevance judgements. *Proceedings of the 30th Annual ACM Conference on Research and Development in Information Retrieval*, 2007a.

Z. Zheng, H. Zha, T. Zhang, O. Chapelle, K. Chen, and G. Sun. A general boosting method and its application to learning ranking functions for web search. *Advances in Neural Information Processing Systems*, 19, 2007b.