Noname manuscript No.
(will be inserted by the editor)

Improving Ranking Performance with Cost-sensitive Ordinal Classification via Regression

Yu-Xun Ruan · Hsuan-Tien Lin · Ming-Feng Tsai

Received: date / Accepted: date

Abstract This paper proposes a novel ranking approach, Cost-sensitive Ordinal Classification via Regression (COCR), which respects the discrete nature of ordinal ranks in real-world data sets. In particular, COCR applies a theoretically-sound reduction from ordinal to binary classification and solves the binary classification sub-tasks with point-wise regression. Furthermore, COCR allows specifying mis-ranking costs to further improve the ranking performance. Such an opportunity is exploited by deriving a corresponding cost for a popular ranking criterion, Expected Reciprocal Rank (ERR). The resulting ERR-tuned COCR boasts the benefits of both efficiency by using point-wise regression and top-rank prediction accuracy from the ERR criterion. Evaluations on two large-scale data sets, including "Yahoo! Learning to Rank Challenge" and "Microsoft Learning to Rank", verify the significant superiority of COCR over commonly-used simple regression approaches. In addition, even better performance can be achieved by the ERR-tuned COCR approach. More specifically, when comparing on the larger data set of the Yahoo! Learning to Rank Challenge, we verify that ERR-tuned COCR can be a promising choice for ensemble learning—improving over a set of popularly-used LambdaMART models and reaching comparable performance to the top-three entries in the challenge.

Keywords list-wise ranking · cost-sensitive · regression · reduction

Yu-Xun Ruan
Graduate Institute of Networking and Multimedia, National Taiwan University, No. 1, Sec. 4, Roosevelt Rd., Taipei 10617, Taiwan
E-mail: r98944042@csie.ntu.edu.tw

Hsuan-Tien Lin
Department of Computer Science & Information Engineering, National Taiwan University, No. 1, Sec. 4, Roosevelt Rd., Taipei 10617, Taiwan
E-mail: htlin@csie.ntu.edu.tw

Ming-Feng Tsai
Department of Computer Science, National Chengchi University, No. 64, Sec. 2, ZhiNan Rd., Wenshan District, Taipei 11605, Taiwan
E-mail: mftsai@nccu.edu.tw

1 Introduction

In web-search engines and recommendation systems, a common practical need is to learn an effective ranking function for information retrieval. In particular, given a query, the ranking function can be used to order a list of related documents, web pages or items by relevance and show users the relevant items on the top of the ranking list. In recent years, the need has drawn much research attention in the information retrieval and machine learning communities (Liu, 2009; Lv et al, 2011; Richardson et al, 2006).

Three important characteristics of the practical need will be considered in this paper. First, the real-world data sets for ranking are usually huge—containing millions of documents or web pages. This paper focus on such a *large-scale* ranking problem. Secondly, many of the real-world benchmark data sets for learning to rank are labeled by human with *ordinal ranks*—that is, qualitative and discrete judgments like {`highly irrelevant`, `irrelevant`, `neural`, `relevant`, `highly relevant`}. We shall focus on learning to rank from such ordinal data sets. Thirdly, the effectiveness of the ranking function is often evaluated by the order of the items in the resulting ranking list, in which those items on the top of the ranking list are emphasized. Such *list-wise evaluation criteria* match users' perception when using the ranking function for information retrieval. We shall study learning to rank under the list-wise evaluation criteria.

To tackle the large-scale ranking problem, many learning-based ranking algorithms are based on a long-lasting method in statistics and machine learning: *regression*. In particular, these algorithms treat the ordinal ranks as real-valued scores and learn a scoring function by regression for ranking. Theoretical connections between regression and list-wise ranking criteria have been studied by Cossock and Zhang (2006). The benefit of regression is that there are some standard and mature tools that can efficiently deal with large-scale data sets. Nevertheless, standard regression tools often require some metric assumptions on the real-valued scores (e.g., rank 4 is twice as large as rank 2), while the assumptions do not naturally fit the characteristics of the ordinal ranks. A few other studies thus try to resort to ordinal *classification*, which is more aligned with the qualitative and discrete nature of the ordinal ranks. Some theoretical connections between classification and list-wise ranking criteria have been established by Li et al (2007).

In this work, we improve and combine the regression and the classification approaches to tackle the ranking problem. In particular, we connect the problem with *cost-sensitive* ordinal classification, a more sophisticated setting than usual ordinal classification. Cost-sensitive classification penalizes different kinds of mispredictions differently; therefore, it can express the list-wise evaluation criteria much better. We study theoretical guarantee that allows using cost-sensitive classification to embed a popular list-wise ranking criterion—*expected reciprocal rank* (ERR) (Chapelle et al, 2009). Furthermore, we apply an existing method to reduce the cost-sensitive ordinal classification problem to a batch of binary classification tasks (Lin and Li, 2012). The reduction method respects the qualitative and discrete nature of the ordinal ranks. Finally, we keep the benefit of the regression tools by using them as soft learners for the batch of binary classification tasks. We name the whole framework *Cost-sensitive Ordinal Classification via Regression* (COCR). The framework not only enables us to use the well-established regression tools without relying on the metric assumptions that the ordinal ranks do not carry, but also allows us to better match the list-wise evaluation criteria by embedding them as costs.

Evaluations on two large-scale data sets, including “Yahoo! Learning to Rank Challenge” and “Microsoft Learning to Rank”, verify the significant superiority of COCR over conventional regression approaches. Experimental results show that COCR can perform better than simple regression approach with some commonly-used costs. The results demonstrate the importance of treating the ordinal ranks as discrete rather than continuous. Moreover, when adding ERR-based costs, COCR can reach even better performance, thereby demonstrating the advantages of connecting ranking to cost-sensitive ordinal classification. We further demonstrate that ERR-tuned COCR can be an useful too for ensemble learning. In particular, when compared on the larger data set of the Yahoo! Learning to Rank Challenge, ERR-tuned COCR improves over a set of popularly-used LambdaMART models and reaches comparable performance to the top-three entries in the challenge.

This paper is organized as follows. First, we introduce the ranking problem and illustrate related works in Section 2. We formulate the COCR framework in Section 3. Section 4 derives the cost corresponding to the ERR criterion. We present the experimental results on some large-scale data sets and conduct several comparisons in Section 5. Finally, we conclude in Section 6.

2 Setup and Related work

We work on the following ranking problem. For a given query with index q, consider a set of documents $\{\mathbf{x}_{q,i}\}_{i=1}^{N(q)}$, in which $N(q)$ is the number of documents related to q and each document $\mathbf{x}_{q,i}$ is encoded as a vector in $\mathcal{X} \subseteq \mathbb{R}^D$. For the task, we attempt to order all $\mathbf{x}_{q,i}$ according to their relevances to q. In particular, each $\mathbf{x}_{q,i}$ is assumed to be associated with an ideal ordinal relevance value $y_{q,i} \in \mathcal{Y} = \{0, 1, 2, \cdots, K\}$. Below is a data set containing Q queries with labeled document-relevance examples:

$$\mathcal{D} = \big\{(\mathbf{x}_{q,i}, y_{q,i})\colon q = 1, 2, \cdots, Q; i = 1, 2, \cdots, N(q)\big\}.$$

The goal of the ranking problem is to use $\mathcal{D}$ to obtain a scoring function (ranker) $r(\mathbf{x})\colon \mathcal{X} \to \mathbb{R}$, and the ranker can obtain an ordering introduced by the predicted value of $r(\mathbf{x}_{q,i})$ that is close to the ordering by target value $y_{q,i}$.

For simplicity, we use n to denote the abstract pair (q, i), let $N = \sum_{q=1}^{Q} N(q)$, and name $\mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^{N}$. Learning-based approaches for the ranking problem can be classified into the following three categories (Liu, 2009):

– Point-wise: The approaches aim at directly predicting the score of $\mathbf{x}$. In other words, they learn $r(\mathbf{x}_{q,i}) \approx y_{q,i}$ to make the orderings introduced by r and that by y as close as possible. When y is real-valued, the goal is similar to the traditional regression; thus, several well-established tools in regression can be applied directly. A representative regression approach for point-wise ranking has been studied by Cossock and Zhang (2006). When the target value y is within an ordinal set $\{0, 1, \cdots, K\}$, the ranking problem can be reduced to ordinal regression (also called ordinal classification) such as McRank (Li et al, 2007) and Pranking (Crammer and Singer, 2002).

– Pair-wise: In this category, the ranking problem is transformed into a binary classification one that decides whether $\mathbf{x}_{q,i}$ is preferred over $\mathbf{x}_{q,j}$. In other words, the aim is to learn a ranker r such that

$$\operatorname{sign}\big((\mathbf{x}_{q,i}) - r(\mathbf{x}_{q,j})\big) \approx \operatorname{sign}\big(y_{q,i} - y_{q,j}\big),$$

which captures the local comparison nature of ranking. The approaches usually need to construct pairs $(\mathbf{x}_{q,i}, \mathbf{x}_{q,j})$ between those examples with different y. Then, given a query with $N(q)$ documents, the number of pairs can be as many as $\Omega\big(N(q)^2\big)$, which makes pair-wise approaches inefficient for large-scale data sets. Representative approaches in this category include RankSVM (Joachims, 2002), RankBoost (Freund et al, 2003) and RankNet (Burges et al, 2005).

– List-wise: While point-wise ranking considers scoring each instance $\mathbf{x}_{q,i}$ by itself and pair-wise ranking tries to predict the local ordering of the pair $(\mathbf{x}_{q,i}, \mathbf{x}_{q,j})$, list-wise ranking targets at the total ordering of $\{\mathbf{x}_{q,i}\}_{i=1}^{N(q)}$ introduced by the ranker r. The approaches attempt to find the best ranker r by optimizing some objective function that can evaluate the effectiveness of different permutations/orderings introduced by different rankers. The objective function can be a list-wise ranking criterion, and the direct optimization allows the learning process to take the structure of all $\{\mathbf{x}_{q,i}\}$ into account. However, since there are $\big(N(q)\big)!$ possible permutations over $N(q)$ documents, list-wise ranking can be computationally even more expensive than pair-wise ranking. Some representative work includes LambdaRank (Burges et al, 2006), BoltzRank (Volkovs and Zemel, 2009), and NDCGBoost (Valizadegan et al, 2000).

This study focuses on how to improve point-wise ranking by incorporating structural information. In specific, we propose to transform the list-wise ranking criterion as cost, and then put the cost into the reduction process for ordinal ranking. The proposed approach not only inherits the benefit of point-wise ranking in terms of dealing with large-scale data sets, but also possesses the advantage of list-wise ranking to take the structure of the whole ranking list into account.

3 Cost-sensitive Ordinal Classification via Regression

In this section, we formulate the COCR framework. We first describe how to reduce a ranking problem from cost-sensitive ordinal classification to binary classification, which is based on the work of (Lin and Li, 2012). Then, we discuss how the reduction method can be extended to pair with regression algorithms instead of binary classification ones.

3.1 Reduction to Binary Classification

We first introduce the reduction method in (**?**), which belongs to point-wise ranking and solves the ordinal classification problem. Considering a data set $\mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^{N}$ and possible ordinal ranks $\mathcal{Y} = \{0, 1, \cdots, K\}$, the reduction method learns a ranker $r \colon \mathcal{X} \to \mathcal{Y}$ from $\mathcal{D}$ such that $r(\mathbf{x})$ is close to $y \in \mathcal{Y}$. In the work, the task of learning a ranker r is decomposed to K simpler sub-tasks, and each sub-task learns a binary classifier $g_k \colon \mathcal{X} \to \{0, 1\}$, where $k = 1, 2, \cdots, K$. In specific, the k-th sub-task is to determine the following question:

"Is $\mathbf{x}$ ranked higher than or equal to rank k?"

Each binary classifier g_k is learned from the transformed data set:

$$\mathcal{D}^{(k)} = \left\{ \left(\mathbf{x}_n, b_n^{(k)} \right) \right\}_{n=1}^{N},$$

where

$$b_n^{(k)} = [\![y_n \geq k]\!] \tag{1}$$

encodes the desired answer for each $\mathbf{x}_n$ on the associated question. If all binary classifiers g_k answer most of the associated questions correctly, it has been theoretically proved (**?**) that a simple "counting" ranker

$$r_g(\mathbf{x}) = \sum_{k=1}^{K} g_k(\mathbf{x}) \tag{2}$$

can also predict the rank y closely.

In addition to the reduction from the ordinal classification task to binary classification ones, the method also allows to specify the costs charged on different kinds of mis-ranking errors. In particular, each example $(\mathbf{x}_n, y_n)$ can be coupled with a cost vector $\mathbf{c}_n$ whose k-th component $\mathbf{c}_n[k]$ denotes the penalty for scoring $\mathbf{x}_n$ as k. The value of $\mathbf{c}_n[k]$ reflects the degree of the difference between y_n and k. Thus, it is common to assume that $\mathbf{c}_n[k] = 0$ when $k = y_n$. In addition, the cost $\mathbf{c}_n[k]$ is assumed to be larger when k is further away from y_n. Two common functions satisfy the requirements and have been widely used in practice:

– absolute cost vectors:
$$\mathbf{c}_n[k] = |y_n - k|\,. \tag{3}$$
– squared cost vectors:
$$\mathbf{c}_n[k] = (y_n - k)^2. \tag{4}$$

For instance, suppose that the highest relevance $K = 4$. Given an example $(\mathbf{x}_n, y_n)$ with $y_n = 3$, the absolute cost is $(3, 2, 1, 0, 1)$ and the squared cost is $(9, 4, 1, 0, 1)$. Note that the squared cost charges more than the absolute cost when k is further away from y_n. The cost vectors give the learning algorithm some additional information about the preferred ranking criterion, and can be used to boost ranking performance if they are chosen or designed carefully.

The reduction method transforms the cost vector $\mathbf{c}_n$ to the weight of each binary example $\left(\mathbf{x}_n, b_n^{(k)}\right)$ to indicate its importance. The weight is defined as

$$w_n^{(k)} = \Big| \mathbf{c}_n[k] - \mathbf{c}_n[k-1] \Big|. \tag{5}$$

Intuitively, when the difference between the k-th and the $(k-1)$-th costs is large, a ranker will attempt to answer the k-th associated question correctly. The theoretical justification for using the weights is shown by (**?**). The weights $w_n^{(k)}$ are included as an additional piece of information when training g_k. Many existing binary classification can take the weights into account by some simple changes in algorithm or by sampling.

Therefore, the reduction method from cost-sensitive ordinal classification to weighted binary classification takes a cost-sensitive data set of the form $\mathcal{D} = \{(\mathbf{x}_n, y_n, \mathbf{c}_n)\}_{n=1}^{N}$ and transforms $\mathcal{D}$ to weighted binary classification data sets $\mathcal{D}^{(k)} = \left\{\left(\mathbf{x}_n, b_n^{(k)}, w_n^{(k)}\right)\right\}_{n=1}^{N}$, each of which is used to learn a binary classifier g_k that will be combined to get the ranker r_g in (2). Note that the absolute cost simply results in $w_n^{(k)} = 1$ (equal weights) and leads to the simple weight-less version introduced in this section. Many existing approaches (Li et al, 2007; Mohan et al, 2011) also decompose the ordinal classification problem to a batch of binary classification sub-tasks in a weight-less way and thus implicitly considers only the absolute cost. The reduction method, on the other hand, provides the opportunity to use a broader range of costs in a principled way.

3.2 Replacing Binary Classification with Regression

The reduction method learns a *hard* ranker r_g from $\mathcal{X}$ to $\mathcal{Y} = \{0, 1, 2, \cdots, K\}$; that is, many different instances $\mathbf{x}_{q,i}$ can be mapped to a same rank. While such a ranker carries a strong theoretical guarantee, it results in ties of ordering, and thus is usually not preferred in practice. Next, we discuss how we can obtain a *soft* ranker from $\mathcal{X}$ to $\mathbb{R}$ instead.

The basic idea is as follows. We replace $g_k \colon \mathcal{X} \to \{0, 1\}$ with soft binary classifiers $h_k \colon \mathcal{X} \to [0, 1]$, where $[\![h_k(\mathbf{x}) \geq 0.5]\!]$ is the hard classifier $g_k(\mathbf{x})$ in prediction while the value of $|h_k(\mathbf{x}) - 0.5|$ represents the confidence of the prediction. Note that the hard ranker r_g in the reduction method is composed by a batch of hard binary classifiers g_k. To use the detailed confidence information after getting h_k, we propose to keep Equation (2) unchanged. That is, the soft ranker will be constructed by

$$r_h(\mathbf{x}) = \sum_{k=1}^{K} h_k(\mathbf{x}). \tag{6}$$

Below we show that r_h can be a reasonable ranker by using the above equation. The common way to learn the soft binary classifiers h_k is to use regression. Traditional least-squared regression, when applied on the binary classification problem from $\mathbf{x}$ to some binary label $b \in \{0, 1\}$, can be viewed as learning an estimator of the posterior probability $P(b = 1|\mathbf{x})$. Following the same argument, each soft binary classifier $h_k(\mathbf{x})$ in our proposed approach estimates the posterior probability $P(y \geq k|\mathbf{x})$. Let us first assume that each h_k is perfectly accurate on the estimation. That is, let $P_k = P(y = k|\mathbf{x})$,

$$\begin{aligned} P_1 + P_2 + \cdots + P_K &= h_1(\mathbf{x}) \\ P_2 + \cdots + P_K &= h_2(\mathbf{x}) \\ \cdots &= \cdots \\ P_K &= h_K(\mathbf{x}). \end{aligned}$$

Take a summation on both sides of the equations,

$$P_1 + 2P_2 + \cdots + KP_K = \sum_{k=1}^{K} h_k(\mathbf{x}) = r_h(\mathbf{x}).$$

Note that the left-hand-side is the expected rank:

$$E(y|\mathbf{x}) = \sum_{k=0}^{K} k \cdot P(y = k|\mathbf{x}).$$

In other words, when all soft binary classifiers $h_k(\mathbf{x})$ perfectly estimate $P(y \geq k|\mathbf{x})$, the soft ranker $r_h(\mathbf{x})$ can also perfectly estimate the expected rank given $\mathbf{x}$.

In practice, however, soft binary classifiers $h_k(\mathbf{x})$ may not be perfect and can make errors in estimating $P(y \geq k|\mathbf{x})$. In such a case, the next theorem shows that $r_h(\mathbf{x})$ is still guaranteed to be close to the expected rank given $\mathbf{x}$.

Theorem 1 *Consider any binary classifiers $h_k \colon \mathcal{X} \to \mathbb{R}$ for $k = 0, 1, \cdots, K$. Assume that*

$$\sum_{k=1}^{K} \Big(h_k(\mathbf{x}) - P(y \geq k|\mathbf{x}) \Big)^2 \leq \epsilon^2.$$

Then,

$$\big(r_h(\mathbf{x}) - E(y|\mathbf{x}) \big)^2 \leq K\epsilon^2.$$

Proof

$$\begin{aligned}
&\big(r_h(\mathbf{x}) - E(y|\mathbf{x}) \big)^2 \\
&= \left(\sum_{k=1}^{K} h_k(\mathbf{x}) - \sum_{k=1}^{K} P(y \geq k|\mathbf{x}) \right)^2 \\
&\leq \left(\sum_{k=1}^{K} 1^2 \right) \left(\sum_{k=1}^{K} \Big(h_k(\mathbf{x}) - P(y \geq k|\mathbf{x}) \Big)^2 \right) \qquad (7) \\
&\leq K\epsilon^2.
\end{aligned}$$

Note that (7) comes from the Cauchy-Schwarz inequality.

Theorem 1 shows that when soft binary classifiers h_k can estimate the posterior probability $P(y \geq k|\mathbf{x})$ correctly, the soft ranker r_h will also obtain the expected rank of $\mathbf{x}$ closely. According to the theorem, we propose to replace the binary classification algorithm in the reduction method with a base regression algorithm $\mathcal{A}_r$. The base regression algorithm attempts to learn soft binary classifiers h_k and obtain a soft ranker r_h by using Equation (6). Algorithm 1 summarizes the process of the proposed *Cost-sensitive Ordinal Classification via Regression* (COCR) framework.

4 Costs For ERR Criterion

In this section, we study how to embed a list-wise ranking criterion as the costs in the COCR framework. We focus on a popular criterion: Expected Reciprocal Rank (ERR), which has been widely used in the *Yahoo! Learning to Rank Challenge*[1].

[1] `http://learningtorankchallenge.yahoo.com/index.php`

Algorithm 1 The COCR Framework

```
Input: $\mathcal{D} = \{(\mathbf{x}_n, y_n, \mathbf{c}_n)\}_{n=1}^N$
  for $k = 1, 2, \cdots, K$ do
    1. Transform the cost-sensitive data set to a weighted binary classification data set
       $\mathcal{D}^{(k)} = \left\{\left(\mathbf{x}_n, b_n^{(k)}, w_n^{(k)}\right)\right\}_{n=1}^N$ with (1) and (5).
    2. Apply a base regression algorithm $\mathcal{A}_r$ on $\mathcal{D}^{(k)}$ to get a soft binary classifier $h_k(\mathbf{x})$.
  end for
  return $r_h$ with (6).
```

4.1 ERR: Expected Reciprocal Rank

ERR (Chapelle et al, 2009) is an evaluation criterion for multiple relevance judgments. Consider a ranker r that defines an ordering:

$$\pi \colon \{1, 2, \cdots, N(q)\} \to \{1, 2, \cdots, N(q)\},$$

where $\pi(i)$ is the position of example $(\mathbf{x}_{q,i}, y_{q,i})$ in the ordering introduced by r, and the document at the top position 1 has the largest $r(\mathbf{x}_{q,i})$. For simplicity we use $\sigma(i)$ to denote $\pi^{-1}(i)$; then, the ERR criterion can be defined as follows:

$$\begin{aligned} ERR(r,q) &= \sum_{i=1}^{N(q)} \frac{1}{i} R\big(y_{q,\sigma(i)}\big) \prod_{j=1}^{i-1} \Big(1 - R\big(y_{q,\sigma(j)}\big)\Big), \\ \text{with } R(y) &= \frac{2^y - 1}{2^K},\ y \in \{0, 1, \cdots, K\}. \end{aligned} \tag{8}$$

The function $R(y)$ maps the ordinal rank y to a probability term that models whether the user would stop at the associated document $\mathbf{x}$. When y is large (highly relevant), $R(y)$ is close to 1; instead, when y is small (highly irrelevant), $R(y)$ is close to 0. Note that for ERR higher values indicate better performance. Considering the ordering produced by r on query q, the criterion can be represented as:

$$ERR(r,q) = \sum_{i=1}^{N(q)} \frac{1}{i} P(\text{user stops at position } i \text{ of ordering}).$$

As suggested by Chapelle et al (2009), ERR reflects users' search behaviors and can be used to quantify users' satisfaction. The main difference between ERR and other position-based metrics such as RBP (Moffat and Zobel, 2008) and NDCG (Järvelin and Kekäläinen, 2002) is that the discount term $\frac{1}{i}\prod_{j=1}^{i-1}\Big(1 - R\big(y_{q,\sigma(j)}\big)\Big)$ of ERR depends not only on the position information $\frac{1}{i}$, but also on whether there are highly relevant instances appearing before position i.

Next, we derive an error bound on the ERR criterion. To simplify the derivation, we work on a single query and remove the query index q from the notation. In addition, we can permute the index in (8) with π and get an equivalent definition of ERR as:

$$ERR(r) = \sum_{i=1}^{N} \frac{1}{\pi(i)} R(y_i) \prod_{j=1}^{\pi(i)-1} \Big(1 - R(y_{\sigma(j)})\Big). \tag{9}$$

4.2 An Error Bound on ERR

In (Cossock and Zhang, 2006), the DCG criterion is shown to be bounded by some regression loss functions; in (Li et al, 2007), the DCG criterion in list-wise ranking is also shown to be bounded by a scaled error rate in multi-class classification. Inspired by the two studies, we derive a bound for ERR in order to find suitable costs for COCR. Note that (Mohan et al, 2011) makes a similar attempt with some different derivation steps and shows that ERR is bounded by a scaled error rate in multi-class classification. Our bound, on the other hand, will reveal that ERR is approximately bounded by some costs in cost-sensitive ordinal classification.

Let $\boldsymbol{\beta}$ be a vector with $\boldsymbol{\beta}[i] = \frac{1}{i}$. In addition, define $\mathbf{y}$ as a vector with $\mathbf{y}[i] = y_i$ and $\hat{\mathbf{y}}$ as a vector $\hat{\mathbf{y}}[i] = r(\mathbf{x}_i)$. For any vector $\tilde{\mathbf{y}}$ of length N, any permutation

$$\tilde{\pi} \colon \{1, 2, \cdots, N\} \to \{1, 2, \cdots, N\}$$

and its inverse permutation $\tilde{\sigma}$, we define

$$F(\tilde{\pi}, \tilde{\mathbf{y}}) = R\Big(\mathbf{y}\big[i\big]\Big) \prod_{j=1}^{\tilde{\pi}(i)-1} \Big(1 - R\Big(\tilde{\mathbf{y}}\big[\tilde{\sigma}(j)\big]\Big)\Big).$$

We see that (9) simplifies to

$$ERR(r) = \sum_{i=1}^{N} \boldsymbol{\beta}[\pi(i)] \cdot F_i(\pi, \mathbf{y}). \tag{10}$$

We now use the above definitions to derive the upper-bound of the difference between $ERR(r)$ and the ERR of a perfect ranker.

Theorem 2 *For a given set of examples $\{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$, consider a perfect ranker p such that $p(\mathbf{x}_i) = y_i$. Assume that the perfect ranker introduces an ordering ρ. Then,*

$$\begin{aligned} & ERR(p) - ERR(r) \\ \le & \left(\sum_{i=1}^{N} \Big(\boldsymbol{\beta}\big[\rho(i)\big] - \boldsymbol{\beta}\big[\pi(i)\big] \Big)^2 \right)^{\frac{1}{2}} \left(\sum_{i=1}^{N} \Big(F_i(\rho, \mathbf{y}) - F_i(\pi, \hat{\mathbf{y}}) \Big)^2 \right)^{\frac{1}{2}} \end{aligned}$$

Proof

From the definition in Equation (10),

$$\begin{aligned} & ERR(r) \\ = & \sum_{i=1}^{N} \boldsymbol{\beta}[\pi(i)] \cdot F_i(\pi, \mathbf{y}) \\ = & \sum_{i=1}^{N} \boldsymbol{\beta}[\pi(i)] \cdot F_i(\pi, \hat{\mathbf{y}}) + \sum_{i=1}^{N} \boldsymbol{\beta}[\pi(i)] \cdot \Big(F_i(\pi, \mathbf{y}) - F_i(\pi, \hat{\mathbf{y}}) \Big). \end{aligned}$$

Note that π is the ordering constructed by $\hat{\mathbf{y}}$. Thus, the sequence $F_i(\pi, \hat{\mathbf{y}})$ is non-decreasing with $\boldsymbol{\beta}[\pi(i)]$. By the rearrangement inequality,

$$\sum_{i=1}^{N} \boldsymbol{\beta}[\pi(i)] \cdot F_i(\pi, \hat{\mathbf{y}}) \geq \sum_{i=1}^{N} \boldsymbol{\beta}[\rho(i)] \cdot F_i(\pi, \hat{\mathbf{y}}). \tag{11}$$

In addition, ρ is the ordering constructed by $\mathbf{y}$. Thus, for all i,

$$F_i(\pi, \mathbf{y}) \geq F_i(\rho, \mathbf{y}). \tag{12}$$

From (11) and (12),

$$\begin{aligned}
&ERR(r) \\
&\geq \sum_{i=1}^{N} \boldsymbol{\beta}\big[\rho(i)\big] \cdot F_i(\pi, \hat{\mathbf{y}}) + \sum_{i=1}^{N} \boldsymbol{\beta}\big[\pi(i)\big] \cdot \Big(F_i(\rho, \mathbf{y}) - F_i(\pi, \hat{\mathbf{y}})\Big) \\
&\geq \sum_{i=1}^{N} \boldsymbol{\beta}\big[\rho(i)\big] \cdot F_i(\rho, \mathbf{y}) + \\
&\quad \sum_{i=1}^{N} \Big(\boldsymbol{\beta}\big[\pi(i)\big] - \boldsymbol{\beta}\big[\rho(i)\big]\Big) \cdot \Big(F_i(\rho, \mathbf{y}) - F_i(\pi, \hat{\mathbf{y}})\Big) \\
&= ERR(p) + \sum_{i=1}^{N} \Big(\boldsymbol{\beta}\big[\pi(i)\big] - \boldsymbol{\beta}\big[\rho(i)\big]\Big) \cdot \Big(F_i(\rho, \mathbf{y}) - F_i(\pi, \hat{\mathbf{y}})\Big).
\end{aligned}$$

Then, by the Cauchy-Schwarz inequality.

$$\begin{aligned}
&ERR(p) - ERR(r) \\
&\leq \left(\sum_{i=1}^{N} \Big(\boldsymbol{\beta}\big[\rho(i)\big] - \boldsymbol{\beta}\big[\pi(i)\big]\Big)^2\right)^{\frac{1}{2}} \left(\sum_{i=1}^{N} \Big(F_i(\rho, \mathbf{y}) - F_i(\pi, \hat{\mathbf{y}})\Big)^2\right)^{\frac{1}{2}}
\end{aligned}$$

4.3 Optimistic ERR Cost

Next, we use the bound in Theorem 2 to derive the costs for the ERR criterion. In specific, we attempt to minimize the right-hand-side of the bound with respect to r. The term

$$\sum_{i=1}^{N} \Big(\boldsymbol{\beta}[\rho(i)] - \boldsymbol{\beta}[\pi(i)]\Big)^2$$

in the bound depends on the total ordering introduced by r and is difficult to be calculated in a point-wise manner by COCR. Thus, we turn to minimize the term:

$$\begin{aligned}
&\Big(F_i(\rho, \mathbf{y}) - F_i(\pi, \hat{\mathbf{y}})\Big)^2 \\
&= \left(\frac{2^{\mathbf{y}[i]} - 1}{2^K} \prod_{j=1}^{\rho(i)-1} \left(1 - \frac{2^{\mathbf{y}[\rho^{-1}(j)]} - 1}{2^K}\right)\right. \\
&\quad \left. - \frac{2^{\hat{\mathbf{y}}[i]} - 1}{2^K} \prod_{j=1}^{\pi(i)-1} \left(1 - \frac{2^{\hat{\mathbf{y}}[\pi^{-1}(j)]} - 1}{2^K}\right)\right)^2.
\end{aligned}$$

Consider an assumption that we consider only rankers that produce correct rank $r(\mathbf{x}_i) \approx y_i$. Then, the ordering π introduced by r will be close to the ordering ρ introduced by the prefect ranker p. Thus,

$$\prod_{j=1}^{\rho(i)-1}\left(1-\frac{2^{\mathbf{y}[\rho^{-1}(j)]}-1}{2^K}\right) \approx \prod_{j=1}^{\pi(i)-1}\left(1-\frac{2^{\hat{\mathbf{y}}[\pi^{-1}(j)]}-1}{2^K}\right)$$

If the ranker predicts $r(\mathbf{x}_i) = k$,

$$\begin{aligned}
&\Big(F_i(\rho,\mathbf{y}) - F_i(\pi,\hat{\mathbf{y}})\Big)^2 \\
\approx &\left(\left(\frac{2^{\mathbf{y}[i]}-1}{2^K} - \frac{2^{2^K}-1}{2^K}\right)\cdot\prod_{j=1}^{\pi(i)-1}\left(1-\frac{2^{\hat{\mathbf{y}}[\pi^{-1}(j)]}-1}{2^K}\right)\right)^2 \\
\propto &\left(2^{y_i} - 2^k\right)^2
\end{aligned}$$

To minimize the bound in Theorem 2, we define the optimistic ERR (oERR) cost vector by:

$$\mathbf{c}_i[k] = \left(2^{y_i} - 2^k\right)^2. \tag{13}$$

When the optimistic ERR cost is minimized, each

$$\Big(F_i(\rho,\mathbf{y}) - F_i(\pi,\hat{\mathbf{y}})\Big)^2$$

is approximately minimized and the right-hand-side of the bound in Theorem 2 is small. Then, $ERR(r)$ would be close to the ideal ERR of the perfect ranker. For $K = 4$, given an example $(\mathbf{x}_n, y_n)$ with $y_n = 3$, the squared cost is $(9, 4, 1, 0, 1)$ and the oERR cost is $(64, 36, 16, 0, 64)$. We see that, when normalized by the largest component in the cost, the oERR vector charges more penalty for mis-ranking errors than the squared cost. Next, we conduct experiments to validate the effectiveness of the oERR cost when it is integrated with the COCR framework.

5 Experiments

We carry out several experiments to verify the following claims:

1. For large-scale list-wise ranking problems with ordinal ranks, with a same base regression approach, the proposed COCR framework can outperform a direct use of regression (Cossock and Zhang, 2006).
2. The derived oERR cost in (13) can be coupled with COCR to boost the quality of ranking in terms of the ERR criterion.
3. The COCR framework with the proposed oERR cost can also be applied for ensemble learning to get even better ranking performance.

We will first introduce the data sets and the base regression algorithms used in our experiments. Then, we compare COCR with different costs and discuss the results. In the end, we validate that COCR with oERR cost is also a valuable and efficient approach for ensemble learning.

5.1 Data sets

Four benchmark real-world large-scale data sets are used in our experiments. The statistics of the benchmark data sets are described below.

- *Yahoo! Learning To Rank Challenge Data sets*:[2]
  In 2010, Yahoo! held the Learning to Rank Challenge for improving the ranking quality in web-search systems. There were two data sets in the competition: the bigger one was used for track 1 and named LTRC1 in our experiments; the smaller one (LTRC2) was used for track 2. Both LTRC1 and LTRC2 are divided into the three parts—training, validation and test. For training/validation/test respectively,
  - LTRC1 contains Q=19,944/2,994/6,983 queries and N=473,134/71,083/165,660 examples
  - LTRC2 contains Q=1,266/1,266/3,798 queries and N=4,815/34,881/103,174 examples

  In both two data sets, the number of features D is 700 and all of the features have been scaled to $[0, 1]$. The relevance values y_n range from 0 to $K = 4$, with 0 meaning irrelevant and 4 meaning highly relevant.
- *Microsoft Learning to Rank Data sets*:[3]
  The data sets were released by Microsoft Research in 2010. There are two data sets MS10K and MS30K.
  - MS10K contains Q=10,000 queries and N=1,200,192 examples.
  - MS30K contains Q=31,531 queries and N=3,771,125 examples.

  The MS10K data set is constructed by a random sampling of 10,000 queries from MS30K. There are $D = 136$ features and we normalize the features to $[0, 1]$. Each data set is divided to five standard parts for cross-validation. The ordinal relevance values in the data sets also range from 0 to 4.

5.2 Base Regression Algorithms

Three base regression algorithms are considered in our experiments, including *linear regression* (Hastie et al, 2003), *M5' decision tree* (M5P) (Wang and Witten, 1997) to *Gradient Boosted Regression Trees* (GBRT) (Friedman, 2001). In the experiments, we use WEKA (Hall et al, 2009) for the linear regression and M5P, and use RT-Rank (Mohan et al, 2011) implementation for GBRT.

- *Linear regression* is arguably one of the most widely-used algorithm for regression. It learns a simple linear model that combines the numerical features in $\mathbf{x}$ to make the prediction. We take the standard least-squared formulation of linear regression (Hastie et al, 2003) as the baseline algorithm in our experiments.
- *M5P* is a decision tree algorithm based on an earlier M5 (Quinlan, 1992) algorithm. M5P produces a regression tree such that each leaf node consists of

[2] `http://learningtorankchallenge.yahoo.com/datasets.php`
[3] `http://research.microsoft.com/en-us/projects/mslr/default.aspx`

a linear model for combining the numerical features. M5P can do nonlinear regression with the partitions provided by the internal nodes and is thus more powerful than linear regression. We will consider a single M5P tree as well as multiple M5P trees combined by the popular *Booststrap Aggregation* (bagging) method (Breiman, 1996).
- *GBRT* aggregates multiple decision trees with gradient boosting to improve the regression performance (Friedman, 2001). The aggregation procedure generates diverse decision trees by taking the regression errors (residuals) into account, and can thus produce a more powerful regressor than bagging-M5P. GBRT is a leading algorithm in the Yahoo! Learning To Rank Challenge (Mohan et al, 2011) and is thus taken into our comparisons.

Below we conduct several comparisons using the above base regression algorithms under the COCR framework with different costs.

5.3 Comparison Using Linear Regression

Table 1 shows the average test ERR of direct regression and three COCR settings on the four data sets when using linear regression as the base algorithm. Bold fonts indicate that the COCR setting significantly outperforms direct regression. The corresponding p-values are also listed in the table for reference. First of all, we see that COCR with the squared cost is better than direct regression on all the data sets. COCR with the absolute cost, which is similar to the McRank approach (Li et al, 2007), can also achieve a higher ERR over direct regression on some data sets. The results verify that it is important to respect the discrete nature of the ordinal-valued y_n instead of directly treating them as real values for regression. In particular, the reduction method within COCR properly takes the discrete nature into account and should thus be preferred over direct regression on the data sets with ordinal ranks.

From Table 1, COCR with the oERR cost not only is better than direct regression, but can further boost the ranking performance over the absolute and the squared costs to reach the best ERR on all data sets except the smallest data set LTRC2. On larger data sets like MS10K and MS30K, the difference is especially large and significant. We further compare COCR with different costs to COCR with the oERR cost with a one-tailed t-test and list the corresponding p-values in Table 3(a). The results show that COCR with the oERR cost is definitely the best choice within the three COCR settings on LTRC1, MS10K and MS30K. The results justify the usefulness of the proposed oERR cost over the commonly-used absolute or square costs.

Another metric for list-wise ranking is Normalized DCG (NDCG) (Järvelin and Kekäläinen, 2002). In order to verify if COCR can also enhance NDCG, we list the NDCG@10 results in Table 2. Note that higher NDCG values indicate better performance. For NDCG@10, COCR with the squared cost is better than direct regression on all data sets. In addition, COCR with the squared cost is better than COCR with the absolute cost on MS10K and MS30K, and better than COCR with the oERR cost on LTRC1 and LTRC2. The findings suggest that COCR with the squared cost is the best of the three settings. On the other hand, COCR with the oERR cost is less strong in terms of the NDCG criterion. Thus, the flexibility of COCR in plugging in different costs is important. More specifically, the flexibility

Table 1 ERR Comparison Using Linear Regression

data set	direct	COCR		
	regression	absolute, p-value	squared, p-value	oERR, p-value
LTRC1	0.4470	$\mathbf{0.4484}$, $\mathbf{3.00 * 10^{-4}}$	$\mathbf{0.4490}$, $\mathbf{2.23 * 10^{-5}}$	$\mathbf{0.4505}$, $\mathbf{3.65 * 10^{-6}}$
LTRC2	0.4440	$\mathbf{0.4465}$, $\mathbf{3.00 * 10^{-4}}$	$\mathbf{0.4472}$, $\mathbf{1.00 * 10^{-4}}$	$\mathbf{0.4461}$, $\mathbf{1.42 * 10^{-2}}$
MS10K	0.2643	0.2642, $5.67 * 10^{-2}$	$\mathbf{0.2697}$, $\mathbf{2.25 * 10^{-20}}$	$\mathbf{0.2792}$, $\mathbf{1.12 * 10^{-35}}$
MS30K	0.2748	0.2748, $2.88 * 10^{-1}$	$\mathbf{0.2828}$, $\mathbf{2.38 * 10^{-116}}$	$\mathbf{0.2942}$, $\mathbf{1.09 * 10^{-161}}$

Table 2 NDCG@10 Comparison Using Linear Regression

data set	direct	COCR		
	regression	absolute, p-value	squared, p-value	oERR, p-value
LTRC1	0.7638	$\mathbf{0.7652}$, $\mathbf{5.34 * 10^{-5}}$	$\mathbf{0.7652}$, $\mathbf{1.20 * 10^{-3}}$	0.7636, $4.05 * 10^{-1}$
LTRC2	0.7519	$\mathbf{0.7552}$, $\mathbf{5.96 * 10^{-6}}$	$\mathbf{0.7562}$, $\mathbf{3.01 * 10^{-6}}$	0.7518, $4.74 * 10^{-1}$
MS10K	0.3916	0.3915, $3.02 * 10^{-1}$	$\mathbf{0.3945}$, $\mathbf{1.90 * 10^{-11}}$	$\mathbf{0.3931}$, $\mathbf{5.79 * 10^{-2}}$
MS30K	0.4025	0.4026, $2.33 * 10^{-1}$	$\mathbf{0.4061}$, $\mathbf{1.44 * 10^{-49}}$	$\mathbf{0.4060}$, $\mathbf{3.40 * 10^{-11}}$

Table 3 One-Tailed Test that Compare the oERR Cost to Other Costs

(a) Linear Regression

data set	absolute	squared
LTRC1	$\mathbf{2.70 * 10^{-3}}$	$\mathbf{1.51 * 10^{-2}}$
LTRC2	$3.12 * 10^{-1}$	$7.48 * 10^{-2}$
MS10K	$\mathbf{1.85 * 10^{-36}}$	$\mathbf{7.73 * 10^{-22}}$
MS30K	$\mathbf{8.12 * 10^{-161}}$	$\mathbf{3.51 * 10^{-81}}$

(b) M5P

data set	absolute	squared
LTRC1	$2.80 * 10^{-1}$	$\mathbf{6.03 * 10^{-2}}$
LTRC2	$\mathbf{1.03 * 10^{-5}}$	$1.62 * 10^{-1}$
MS10K	$\mathbf{8.10 * 10^{-3}}$	$\mathbf{4.56 * 10^{-8}}$
MS30K	$\mathbf{3.85 * 10^{-2}}$	$\mathbf{1.37 * 10^{-5}}$

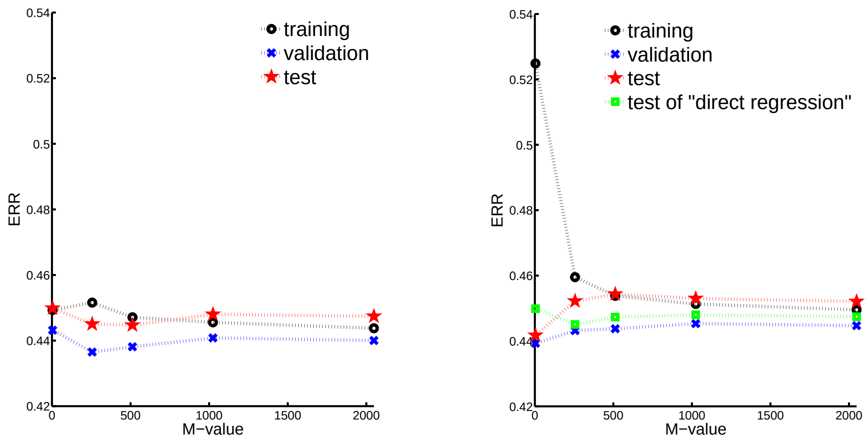
allows us to obtain better rankers towards the application needs (NDCG or ERR) by tuning the costs appropriately.

5.4 Comparison Using M5P

The M5P decision tree comes with a parameter M, which stands for the minimum number of instances per leaf when constructing the tree. A smaller M results in a more complex (possibly deeper) tree while a larger M results in a simpler one. Figure 1 shows the results of tuning M when applying M5P in direct regression and COCR with the oERR cost on the LTRC1 data set. The M-values of 4, 256, 512, 1024 and 2048 are examined. The default M in WEKA is 4.

Figure 1(a) shows that direct regression with M5P can reach the best test performance on the default value of $M = 4$. However, as shown in Figure 1(b), COCR with the oERR cost can overfit when $M = 4$. Its training ERR is dramatically high, but the test ERR is extremely low. The findings suggest a careful selection of the M parameter. We conduct a fair selection scheme using the validation ERR.

Fig. 1 Effects of Tuning the Parameter M of M5P on LTRC1



(a) direct regression (b) COCR with oERR cost

Table 4 ERR Results of Tuning the Parameter M of M5P on LTRC1

parameter M	direct regression	COCR absolute	COCR squared	COCR oERR
4 (validation)	0.4432*	0.4381	0.4381	0.4393
256 (validation)	0.4365	0.4410	0.4425	0.4432
512 (validation)	0.4382	0.4426	0.4437	0.4438
1024 (validation)	0.4408	0.4445*	0.4453*	0.4453*
2048 (validation)	0.4400	0.4426	0.4431	0.4447
test by best validation	0.4499	**0.4526**	**0.4521**	**0.4530**
p-value		$\mathbf{2.60 * 10^{-3}}$	$\mathbf{1.05 * 10^{-2}}$	$\mathbf{8.00 * 10^{-4}}$

(* represents the best validation result)

In particular, we check the models constructed by $M = 4$, 256, 512, 1024 and 2048; pick the model that comes with the highest validation ERR; report its corresponding test ERR. The results on LTRC1 is listed in Table 4. The first five rows show the validation results of different algorithms, and the last row show the test result when using the best model in validation. The results demonstrate that when M is carefully selected, all COCR settings outperform direct regression significantly on LTRC1 and COCR with the oERR achieves the best ERR of the three settings.

With the parameter selection scheme above, Table 5 lists the test ERR on the four data sets. The results in the table further confirms that almost all COCR settings are significantly better than direct regression on all data sets, except COCR with the absolute cost on the smallest LTRC2. Furthermore, COCR with oERR cost achieves the best ERR performance on all data sets. After comparing COCR with the oERR cost to COCR with other costs using a one-tailed t-test, as shown

Table 5 ERR Comparison Using M5P

data set	direct	COCR		
	regression	absolute, p-value	squared, p-value	oERR, p-value
LTRC1	0.4499	$\mathbf{0.4526, 2.60 * 10^{-3}}$	$0.4521, 1.05 * 10^{-2}$	$\mathbf{0.4530, 8.00 * 10^{-4}}$
LTRC2	0.4489	$0.4499, 2.26 * 10^{-1}$	$\mathbf{0.4533, 7.00 * 10^{-4}}$	$\mathbf{0.4538, 2.00 * 10^{-4}}$
MS10K	0.3014	$\mathbf{0.3129, 3.11 * 10^{-13}}$	$\mathbf{0.3101, 3.04 * 10^{-8}}$	$\mathbf{0.3156, 3.38 * 10^{-17}}$
MS30K	0.3298	$\mathbf{0.3438, 4.32 * 10^{-54}}$	$\mathbf{0.3423, 4.23 * 10^{-43}}$	$\mathbf{0.3451, 1.74 * 10^{-59}}$

Table 6 NDCG@10 Comparison Using M5P

data set	direct	COCR		
	regression	absolute, p-value	squared, p-value	oERR, p-value
LTRC1	0.7680	$\mathbf{0.7695, 9.01 * 10^{-2}}$	$\mathbf{0.7698, 1.26 * 10^{-2}}$	$\mathbf{0.7698, 7.26 * 10^{-2}}$
LTRC2	0.7535	$0.7519, 2.26 * 10^{-1}$	$\mathbf{0.7565, 7.49 * 10^{-2}}$	$\mathbf{0.7567, 6.39 * 10^{-2}}$
MS10K	0.4233	$\mathbf{0.4327, 8.07 * 10^{-11}}$	$\mathbf{0.4295, 1.31 * 10^{-5}}$	$\mathbf{0.4284, 4.00 * 10^{-4}}$
MS30K	0.4545	$\mathbf{0.4645, 5.32 * 10^{-34}}$	$\mathbf{0.4614, 6.79 * 10^{-18}}$	$\mathbf{0.4589, 1.20 * 10^{-7}}$

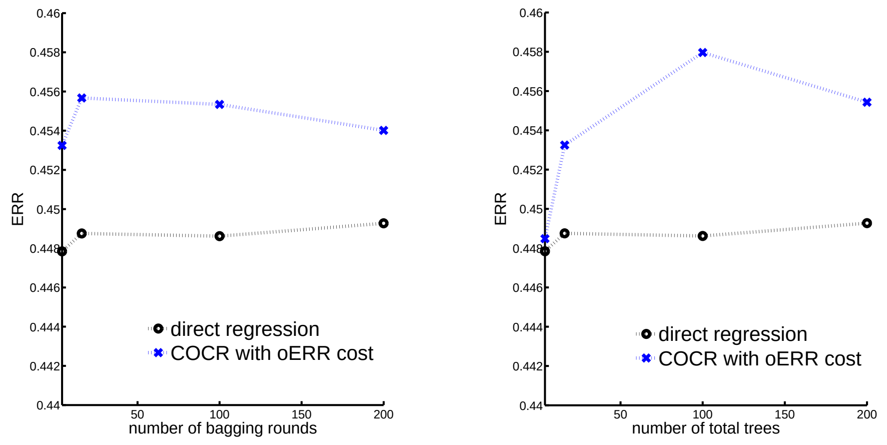
in Table 3(b), we verify that the differences are mostly significant, especially on MS30K and MS10K. The results again confirm that the oERR cost is a competitive choice in the COCR settings.

Table 6 shows the test NDCG results. Both COCR with the squared and the oERR costs can be better than direct regression on all data sets. In addition, COCR with the absolute cost can be better than direct regression on all data sets except the smallest LTRC2. The finding echoes the results in Table 2 about the benefit of COCR on improving NDCG with a carefully chosen cost.

5.5 Comparison Using Bagging-M5P

One concern about the comparison using M5P is that the COCR framework appears to be combining K decision trees while direct regession only uses a single tree. To understand more about the effect on different number of trees, we combine the bagging algorithm (Breiman, 1996) with M5P. In particular, we run T rounds of bagging. In each round, a boostrapped 10% of the training data set is used to obtain a M5P decision tree. After T rounds, the trees are averaged to form the final prediction. Then, bagging-M5P for direct regression generates T decision trees and bagging-M5P for COCR generates TK trees. Figure 2(a) compares direct bagging-M5P to COCR-bagging-M5P with the oERR cost under the same T. That is, for the same horizontal value, the corresponding point on the COCR curve uses K times more trees than the point on the direct regression curve. The figure shows that the whole performance curve of COCR is always better than direct regression. On the other hand, Figure 2(b) compares the two algorithms under the same total number of trees. That is, COCR with T rounds of bagging-M5P is compared to direct regression with TK rounds of bagging-M5P. The figure suggests that the performance of COCR with the oERR cost is still much better

Fig. 2 Effects of Number-of-rounds and Number-of-trees in Bagging-M5P



(a) Comparison under the Same Number-of-rounds

(b) Comparison under the Same Number-of-trees

Table 7 ERR Comparison Using GBRT (LTRC1)

step size	direct	COCR		
	regression	absolute, p-value	squared, p-value	oERR, p-value
0.1	0.4590	0.4595, $1.74 * 10^{-1}$	**0.4602**, $\mathbf{1.42 * 10^{-2}}$	**0.4603**, $\mathbf{2.38 * 10^{-2}}$
0.05	0.4576	**0.4587**, $\mathbf{2.82 * 10^{-2}}$	**0.4596**, $\mathbf{2.46 * 10^{-4}}$	**0.4602**, $\mathbf{2.02 * 10^{-4}}$
0.02	0.4547	**0.4566**, $\mathbf{4.87 * 10^{-5}}$	**0.4575**, $\mathbf{2.16 * 10^{-8}}$	**0.4583**, $\mathbf{6.96 * 10^{-7}}$

Table 8 ERR Comparison Using GBRT (LTRC2)

step size	direct	COCR		
	regression	absolute, p-value	squared, p-value	oERR, p-value
0.1	0.4563	0.4571, $2.26 * 10^{-1}$	**0.4579**, $\mathbf{6.87 * 10^{-2}}$	**0.4597**, $\mathbf{9.00 * 10^{-4}}$
0.05	0.4584	0.4586, $3.72 * 10^{-1}$	**0.4599**, $\mathbf{3.97 * 10^{-2}}$	**0.4598**, $\mathbf{5.29 * 10^{-2}}$
0.02	0.4601	0.4603, $3.59 * 10^{-1}$	0.4599, $4.67 * 10^{-1}$	0.4600, $4.75 * 10^{-1}$

than direct regression. The results demonstrate that COCR with the oERR cost is consistently a better choice than direct regression using bagging-M5P, regardless of whether we compare under the same number of bagging rounds or the same number of M5P trees.

5.6 Comparison Using GBRT

Next, we compare COCR settings with direct regression using GBRT (Friedman, 2001) as the base regression algorithm. Because GBRT requires longer to train

Table 9 Values Setted of Parameters

parameter	value
numbers of leaves	from 50 to 200 in steps of 30
min percentage of observations per leaf	$0.12, 0.25, 0.5$
learning rate	$0.01, 0.05, 0.1, 0.3, 0.5$
sub-sampling rate	$0.3, 0.5, 1.0$
feature-sampling rate	$0.1, 0.3, 0.5, 1.0$
number of trees	3000

Table 10 LambdaMART ERR Results for Ensemble Learning (LTRC1)

model	ERR result
LambdaMART (i)	0.4652
LambdaMART (ii)	0.4648
LambdaMART (iii)	0.4648
LambdaMART (iv)	0.4659
LambdaMART (v)	0.4656
LambdaMART (vi)	0.4663

than bagging-M5P, M5P or linear regression, we only conduct the experiments on the data sets: LTRC1, LTRC2. We follow the setting in (Mohan et al, 2011) for the parameters of GBRT—the number of iterations is set to 1000; the depth of every decision tree is set to 4; the step size of each GBRT iteration is set to either 0.1, 0.05 or 0.02. Table 7 and Table 8 respectively show the ERR results on LTRC1 and LTRC2. In Table 7, no matter what the value of step size, COCR with any costs perform significantly better than direct regression with GBRT in most case. In Table 8, when using a larger step size 0.1 or 0.05, COCR with the squared or the oERR costs perform significantly better than direct regression with GBRT; COCR with the absolute cost is similar to direct regression with GBRT. When using a smaller step size 0.02, however, all four algorithms in Table 8 can reach similar ERR on the small data set. Because COCR with the oERR cost setting always enjoys a similar or better performance than direct regression or COCR with other costs, it can be a useful first-hand choice for a sophisticated base regression algorithm like GBRT.

5.7 Ensemble learning using COCR with oERR cost

The experimental results above validate that the proposed oERR-tuned COCR is effective for enhancing the ranking performance. Next, we demonstrate that oERR-tuned COCR is also a promising choice for ensemble learning. We consider a popularly-used ranking model, LambdaMART (Burges, 2010), as the base learner for ensemble learning, and run the experiments on the Yahoo! LTRC1 data set. We

Table 11 ERR Results for Ensemble Learning (LTRC1)

combined model set	ERR result
Set I	**0.4663**
Set II	0.4662
Set III	**0.4673**

Table 12 The Top 5 Accuracies on the Leader Board of LTRC (track 1)

rank	ERR result
1	0.468605
2	0.467857
3	0.466954
4	0.466776
5	0.466157

use the open-source library[4] to training LambdaMART, and follow the settings in (Burges et al, 2011; Ganjisaffar et al, 2011) for tuning the parameters (see Table 9).

We select some LambdaMART models from the various parameter combinations. The corresponding test ERR of those models are shown in Table 10. We then take oERR-tuned COCR with linear regression to combine the models within those sets. Then, we consider three sets of LambdaMART models: Set I (LambdaMART model (i), (ii), (iii), (iv), (v)), Set II (LambdaMART model (i), (ii), (iii), (iv), (v), (vi)), Set III (LambdaMART model (i), (iii), (iv), (vi)).

Table 11 shows the test ERR of the three sets. Several findings are highlighted as follows:

– The ensemble from Set I, which contains some "weaker" LambdaMART models, is comparable to the best single LambdaMart (vi). The result demonstrate that oERR-tuned COCR is a reasonable approach for ensemble learning.
– The ensemble from Set II, which contains Set I plus the "strong" LambdaMART (vi), there is no significant improvement over Set I.
– The ensemble from Set III, which comes from selecting a subset of models in Set II, we can reach significantly better performance. The result of 0.4673 in ERR beats the best single LambdaMART model with a p-value $1.18 * 10^{-2}$.

Table 12 lists the top five ERR results on the competition of "Yahoo! Learning to Rank Challenge". As shown in the table, our experimental result on Set III (0.4673) reaches comparable performance to the top-three entries on the competition data set. Therefore, it demonstrates that oERR-tuned COCR can be an effective technique for combining weaker, possibly cost-insensitive base learners.

[4] http://code.google.com/p/jforests/

6 CONCLUSIONS

We propose a novel COCR framework for ranking. The framework consists of three main ingredients: decomposing the ordinal ranks to binary classification labels to respect the discrete nature of the ranks; allowing different costs to express the desired ranking criterion; using mature regression tools to not only deal with large-scale data sets but also provide good estimates of the expected rank. In addition to the sound theoretical guarantee of the proposed COCR, a series of empirical results with different base regression algorithms demonstrate the effectiveness of COCR. In particular, COCR with the squared cost can usually do better than direct regression, a commonly-used baseline, on both the ERR criterion and the NDCG criterion.

In addition, we prove an upper bound of the ERR criterion and derive the optimistic ERR cost from the bound. Experimental results suggest that COCR with the optimistic ERR cost not only outperforms direct regression but also reaches better ERR than COCR with the absolute or the squared costs often. Furthermore, we verify that COCR with the optimistic ERR improves over a set of popularly-used LambdaMART models and is a promising for ensemble learning. Future work includes coupling the proposed COCR framework with other well-known regression algorithms and deriving costs that correspond to other pair-wise or list-wise ranking criteria of interest.

References

Breiman L (1996) Bagging predictors. Machine Learning 24(2):123–140

Burges C (2010) From ranknet to lambdarank to lambdamart: An overview. Tech. rep., Microsoft Research

Burges C, Shaked T, Renshaw E, Lazier A, Deeds M, Hamilton N, Hullender G (2005) Learning to rank using gradient descent. In: Proceedings of ICML '05, ACM, pp 89–96

Burges C, Ragno R, Le QV (2006) Learning to rank with nonsmooth cost functions. In: Advances in Neural Information Processing Systems (NIPS), MIT Press, vol 18, pp 193–200

Burges C, Svore KM, Bennett PN, Pastusiak A, Wu Q (2011) Learning to rank using an ensemble of lambda-gradient models. Journal of Machine Learning Research: Workshop and Conference Proceedings 14:25–35

Chapelle O, Metlzer D, Zhang Y, Grinspan P (2009) Expected reciprocal rank for graded relevance. In: Proceedings of CIKM '09, ACM, pp 621–630

Cossock D, Zhang T (2006) Subset ranking using regression. In: Proceedings of COLT '06, Springer, pp 605–619

Crammer K, Singer Y (2002) Pranking with ranking. In: Advances in Neural Information Processing Systems (NIPS), MIT Press, vol 14, pp 641–647

Freund Y, Iyer R, Schapire RE, Singer Y (2003) An efficient boosting algorithm for combining preferences. Journal of Machine Learning Research 4:933–969

Friedman JH (2001) Greedy function approximation: A gradient boosting machine. Annals of Statistics 29:1189–1232

Ganjisaffar Y, Caruana R, Lopes C (2011) Bagging gradient-boosted trees for high precision, low variance ranking models. In: Proceedings of SIGIR '11, ACM, pp 85–94

Hall M, Frank E, Holmes G, Pfahringer B, Reutemann P, Witten IH (2009) The weka data mining software: an update. SIGKDD Explorations Newsletter 11(1):10–18

Hastie T, Tibshirani R, Friedman J (2003) The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer

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

Joachims T (2002) Optimizing search engines using clickthrough data. In: Proceedings of KDD '02, ACM, pp 133–142

Li P, Burges C, Wu Q, Platt JC, Koller D, Singer Y, Roweis S (2007) Mcrank: Learning to rank using multiple classification and gradient boosting. In: Advances in Neural Information Processing Systems (NIPS), MIT Press, vol 19

Lin HT, Li L (2012) Reduction from cost-sensitive ordinal ranking to weighted binary classification. Neural Computation To appear

Liu TY (2009) Learning to rank for information retrieval. Foundations and Trends in Information Retrieval 3:225–331

Lv Y, Moon T, Kolari P, Zheng Z, Wang X, Chang Y (2011) Learning to model relatedness for news recommendation. In: Proceedings of WWW '11, ACM, pp 57–66

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

Mohan A, Chen Z, Weinberger KQ (2011) Web-search ranking with initialized gradient boosted regression trees. Journal of Machine Learning Research Workshop and Conference Proceedings 14:77–89

Quinlan RJ (1992) Learning with continuous classes. In: Proceedings of IJCAI '92, World Scientific, pp 343–348

Richardson M, Prakash A, Brill E (2006) Beyond PageRank: machine learning for static ranking. In: Proceedings of WWW '06, ACM, pp 707–715

Valizadegan H, Jin R, Zhang R, Mao J (2000) Learning to rank by optimizing ndcg measure. In: Proceedings of SIGIR '00, ACM, pp 41–48

Volkovs MN, Zemel RS (2009) Boltzrank: learning to maximize expected ranking gain. In: Proceedings of ICML '09, ACM, pp 1089–1096

Wang Y, Witten IH (1997) Induction of model trees for predicting continuous classes. In: Proceedings of ECML '97, Springer, pp 128–137