Combining Ordinal Preferences by Boosting

Hsuan-Tien Lin[1] and Ling Li[2]

[1] Department of Computer Science and Information Engineering
National Taiwan University
htlin@csie.ntu.edu.tw
[2] Department of Computer Science
California Institute of Technology
ling@caltech.edu

Abstract. We analyze the relationship between ordinal ranking and binary classification with a new technique called reverse reduction. In particular, we prove that the regret can be transformed between ordinal ranking and binary classification. The proof allows us to establish a general equivalence between the two in terms of hardness. Furthermore, we use the technique to design a novel boosting approach that improves any cost-sensitive base ordinal ranking algorithm. The approach extends the well-known AdaBoost to the area of ordinal ranking, and inherits many of its good properties. Experimental results demonstrate that our proposed approach can achieve decent training and test performance even when the base algorithm produces only simple decision stumps.

1 Introduction

We work on a supervised learning task called *ordinal ranking*, which is also referred to as ordinal regression [1] or ordinal classification [2]. The task, which aims at predicting the ranks (i.e., ordinal class labels) of future inputs, is closely related to multi-class classification and metric regression. Somehow it is different from the former because of the ordinal information encoded in the ranks, and is different from the latter because the metric distance between the ranks is not explicitly defined. Since rank is a natural representation of human preferences, the task lends itself to many applications in social science and information retrieval.

Many ordinal ranking algorithms have been proposed from a machine learning perspective in recent years. For instance, Herbrich et al. [3] designed an approach with support vector machines based on comparing training examples in a pairwise manner. Nevertheless, such a pairwise comparison perspective may not be suitable for large-scale learning because the size of the associated optimization problem is quadratic to the number of training examples.

There are some other approaches that do not lead to such a quadratic expansion, such as perceptron ranking [4, PRank] and support vector ordinal regression [1, SVOR]. Li and Lin [5] proposed a reduction method that unified these approaches using the extended binary classification (EBC) perspective,

and showed that any binary classification algorithm can be casted as an ordinal ranking one with EBC. Still some other approaches fall into neither of the perspective above, such as Gaussian process ordinal regression [6].

Given the wide availability of ordinal ranking algorithms, a natural question is whether their performance could be further improved by a generic method. In binary classification, there are many *boosting* algorithms that serve the purpose. They usually work by combining the hypotheses that are produced from a base binary classification algorithm [7], including the well-known adaptive boosting (AdaBoost) approach [8]. There are also some boosting-related approaches for ordinal ranking. For example, Freund et al. [9] introduced the RankBoost approach based on the pairwise comparison perspective. Lin and Li [10] proposed ordinal regression boosting (ORBoost), which is a special instance of the EBC perspective. However, both approaches take a base binary classification algorithm rather than a base ordinal ranking one. In other words, they cannot be directly used to improve the performance of existing ordinal ranking algorithms such as PRank or SVOR.

In this paper, we propose a novel boosting approach for ordinal ranking. The approach improves the performance of any cost-sensitive ordinal ranking algorithm, including PRank and SVOR. Our approach directly extends the original AdaBoost, and inherits many of its good properties. The approach is designed with a technique called *reverse reduction*, which complements the reduction method of Li and Lin [5]. The technique not only helps in designing our proposed approach, but also reveals strong theoretical connections between ordinal ranking and binary classification.

The paper is organized as follows. In Section 2, we introduce the basic setup as well as the reduction method of Li and Lin [5]. Then, we discuss the reverse reduction technique and its theoretical implications in Section 3. We use the technique to design and analyze the proposed AdaBoost.OR approach in Section 4. Finally, we show the experimental results in Section 5 and conclude in Section 6.

2 Background

We shall first define the ordinal ranking problem. Then, we introduce the reduction method of Li and Lin [5] and the consequent error bound.

2.1 Problem Setup

In the ordinal ranking problem, the task is to predict the rank y of some input $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^D$, where y belongs to a set $\mathcal{K}$ of consecutive integers $1, 2, \cdots, K$. We shall adopt the cost-sensitive setting, in which a cost vector $\mathbf{c} \in \mathbb{R}^K$ is generated with $(\mathbf{x}, y)$ from some fixed but unknown distribution $\mathcal{P}$ on $\mathcal{X} \times \mathcal{K} \times \mathbb{R}^K$. The k-th element $\mathbf{c}[k]$ of the cost vector represents the penalty when predicting the input vector $\mathbf{x}$ as rank k. We naturally assume that $\mathbf{c}[y] = 0$ and $\mathbf{c}[k] \geq 0$ for all $k \in \mathcal{K}$. An ordinal ranking problem comes with a given training

set $\mathcal{S} = \{(\mathbf{x}_n, y_n, \mathbf{c}_n)\}_{n=1}^{N}$, whose elements are drawn i.i.d. from $\mathcal{P}$. The goal of the problem is to find an ordinal ranker $r\colon \mathcal{X} \to \mathcal{K}$ such that its generalization error $\pi(r) \equiv \mathcal{E}_{(\mathbf{x},y,\mathbf{c})\sim\mathcal{P}}\, \mathbf{c}[r(\mathbf{x})]$ is small.

Note that if we replace "rank" with "label", the setup above is the same as the cost-sensitive K-class classification problem [11]. The rank, however, carries extra ordinal information, which suggests that the mislabeling penalty depend on the "closeness" of the prediction. Hence, the cost vector $\mathbf{c}$ should be *V-shaped* with respect to y [5], i.e.,

$$\begin{cases} \mathbf{c}[k-1] \geq \mathbf{c}[k]\,, & \text{for } 2 \leq k \leq y\,; \\ \mathbf{c}[k+1] \geq \mathbf{c}[k]\,, & \text{for } y \leq k \leq K-1. \end{cases}$$

We shall assume that every cost vector $\mathbf{c}$ generated from $\mathcal{P}$ is V-shaped with respect to its associated y.

2.2 Reduction Method and Error Bound

Li and Lin [5] proposed a reduction method from ordinal ranking to binary classification. The reduction method constitutes of two stages: training and prediction. During the training stage, each ordinal example is extended to $K-1$ weighted binary examples. Then, the binary examples are used to train a set of $K-1$ closely related binary classifiers, or equivalently, one joint binary classifier $g(\mathbf{x}, k)$. Then, during the prediction stage, the ordinal ranker $r_g(\mathbf{x})$ is constructed from $g(\mathbf{x}, k)$ by a counting method:[3]

$$r_g(\mathbf{x}) \equiv 1 + \sum_{k=1}^{K-1} [\![g(\mathbf{x}, k) > 0]\!]\,. \tag{1}$$

Although Li and Lin [5] dealt with a more restricted cost-sensitive setting, the error bound theorem [5, Theorem 3], which is one of their key results, can be easily extended for our setting. The extension is based on the following distribution $\mathcal{P}_b$ that generates weighted binary examples $(\mathbf{x}, k, z, w)$:

1. Draw a tuple $(\mathbf{x}, y, \mathbf{c})$ from $\mathcal{P}$, and draw k uniformly within $\{1, 2, \cdots, K-1\}$.
2. Let $\begin{cases} z = 2 \cdot [\![k < y]\!] - 1 \\ w = (K-1) \cdot \left| \mathbf{c}[k+1] - \mathbf{c}[k] \right| \end{cases}$. (2)

With distribution $\mathcal{P}_b$, the generalization error of any binary classifier g is

$$\pi_b(g) \equiv \mathop{\mathcal{E}}_{(\mathbf{x},k,z,w)\sim\mathcal{P}_b} w \cdot [\![z \neq g(\mathbf{x}, k)]\!]\,.$$

Then, we can obtain the extended error bound theorem with a proof similar to the one from Li and Lin [5].

[3] $[\![\cdot]\!]$ is 1 if the inner condition is true, and 0 otherwise.

Theorem 1. *If $g(\mathbf{x}, k)$ is rank-monotonic, i.e.,*

$$g(\mathbf{x}, k-1) \geq g(\mathbf{x}, k), \text{ for } 2 \leq k \leq K-1,$$

or if every cost vector $\mathbf{c}$ is convex, i.e.,

$$\mathbf{c}[k+1] - \mathbf{c}[k] \geq \mathbf{c}[k] - \mathbf{c}[k-1], \text{ for } 2 \leq k \leq K-2,$$

then $\pi(r_g) \leq \pi_b(g)$.

Proof. The details can be found in the work of Lin [12].

3 Reverse Reduction Technique

Theorem 1 indicates that if there exists a decent binary classifier g, we can obtain a "good" ordinal ranker r_g. Nevertheless, it does not guarantee how good r_g is in comparison with other ordinal rankers. If we denote g_* as the optimal binary classifier under $\mathcal{P}_b$, and r_* as the optimal ordinal ranker under $\mathcal{P}$, does a small regret $\big(\pi_b(g) - \pi_b(g_*)\big)$ in binary classification translate to a small regret $\big(\pi(r_g) - \pi(r_*)\big)$ in ordinal ranking? In particular, is $\pi(r_{g_*})$ close to $\pi(r_*)$? Next, we introduce the reverse reduction technique, which helps to answer the questions above.

3.1 Reverse Reduction

The reverse reduction technique works on the binary classification problems generated by the reduction method described in Section 2. We can use the technique to not only understand more about the theoretical nature of ordinal ranking, but also design better ordinal ranking algorithms. Reverse reduction goes through each stage of the reduction method in a different direction. In the training stage, instead of starting with the ordinal examples $(\mathbf{x}_n, y_n, \mathbf{c}_n)$, reverse reduction deals with the weighted binary examples $(\mathbf{x}_n, k, z_{nk}, w_{nk})$. It first combines each set of binary examples sharing the same $\mathbf{x}_n$ to an ordinal example by

$$\begin{cases} y_n = 1 + \sum_{k=1}^{K-1} [\![z_{nk} > 0]\!] \ ; \\ \mathbf{c}_n[k] = \sum_{\ell=1}^{K-1} \frac{w_{n\ell}}{K-1} \cdot [\![y_n \leq \ell < k \text{ or } k < \ell \leq y_n]\!] . \end{cases} \tag{3}$$

It is easy to verify that (3) is the exact inverse transform of (2) on the training examples. These ordinal examples are then given to an ordinal ranking algorithm to obtain an ordinal ranker r. In the prediction stage, reverse reduction works by decomposing the prediction $r(\mathbf{x})$ to $K-1$ binary predictions, each as if coming from a joint binary classifier

$$g_r(\mathbf{x}, k) = 2\,[\![k < r(\mathbf{x})]\!] - 1. \tag{4}$$

Then, a lemma on the generalization ability of g_r immediately follows.

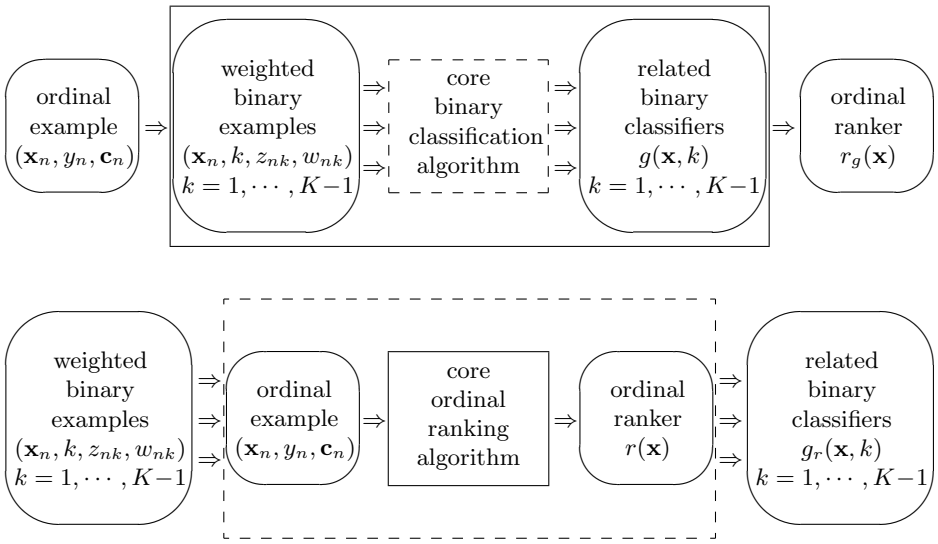


Fig. 1. Top: Reduction; Bottom: Reverse Reduction

Lemma 1. *For every ordinal ranker r, $\pi(r) = \pi_b(g_r)$.*

Proof. Because every cost vector $\mathbf{c}$ from $\mathcal{P}$ is V-shaped,

$$\begin{aligned}\pi(r) &= \mathop{\mathcal{E}}_{(\mathbf{x},y,\mathbf{c})\sim\mathcal{P}} \sum_{[\![k<y]\!]\neq[\![k<r(\mathbf{x})]\!]} \Big|\mathbf{c}[k+1] - \mathbf{c}[k]\Big| \\ &= \mathop{\mathcal{E}}_{(\mathbf{x},k,z,w)\sim\mathcal{P}_b} w \cdot [\![z \neq g_r(\mathbf{x}, k)]\!] \\ &= \pi_b(g_r).\end{aligned}$$

The steps of reduction and reverse reduction are illustrated in Figure 1. Note that if the reduction block is plugged into the reverse reduction block, we recover the underlying binary classification algorithm (and vice versa). This observation may suggest that reverse reduction is trivial and useless. Nevertheless, as we will show next, reverse reduction is a perfect complement of the reduction method, and allows us to draw a strong theoretical connection between ordinal ranking and binary classification. In addition, reverse reduction is useful in designing boosting methods for ordinal ranking, which will be demonstrated in Section 4.

3.2 Regret Bound via Reverse Reduction

Without loss of generality, we use the following definition for the optimal ordinal ranker r_* and the optimal binary classifier g_*, with ties arbitrarily broken and

sign(0) assumed to be +1.

$$r_*(\mathbf{x}) \equiv \operatorname*{argmin}_{\ell} \mathop{\mathcal{E}}_{\mathbf{c}\sim\mathcal{P}(\cdot|\mathbf{x})} \mathbf{c}[\ell]\,,$$
$$g_*(\mathbf{x},k) \equiv \operatorname{sign}\left(\mathop{\mathcal{E}}_{(w,z)\sim\mathcal{P}_b(\cdot|\mathbf{x},k)} (w\cdot z)\right).$$

It is not hard to prove that r_* and g_* are optimal, i.e., for any ordinal ranker r and any binary classifier g,

$$\pi(r) \geq \pi(r_*), \quad \pi_b(g) \geq \pi_b(g_*)\,. \tag{5}$$

With the definitions of r_* and g_*, the reverse reduction technique allows a simple proof of the following regret bound.

Theorem 2. *If $g(\mathbf{x},k)$ is rank-monotonic, or if every cost vector $\mathbf{c}$ is convex, then*

$$\pi(r_g) - \pi(r_*) \leq \pi_b(g) - \pi_b(g_*).$$

Proof.

$$\begin{array}{lcll} \pi(r_g) - \pi(r_*) & \leq & \pi_b(g) - \pi(r_*) & \text{(from Theorem 1)} \\ & = & \pi_b(g) - \pi_b(g_{r_*}) & \text{(from Lemma 1)} \\ & \leq & \pi_b(g) - \pi_b(g_*) & \big(\text{from (5)}\big) \end{array}$$

An immediate implication of the regret bound is as follows. If there exists one optimal binary classifier g_+ that is rank-monotonic, both the right-hand-side and the left-hand-side are 0. That is, every optimal binary classifier under $\mathcal{P}_b$ leads to an optimal ordinal ranker under $\mathcal{P}$. In other words, locating an optimal ordinal ranker is "no harder than" locating an optimal binary classifier. On the other hand, binary classification is also "no harder than" ordinal ranking, because the former is a special case of the latter with $K = 2$. Therefore, if there is a rank-monotonic g_+, ordinal ranking is equivalent to binary classification in hardness.[4] In the following theorem, we show a general sufficient condition for the equivalence.

Theorem 3. *Assume that the effective cost*

$$\mathbf{c}_{\mathbf{x}}[k] = \mathop{\mathcal{E}}_{\mathbf{c}\sim\mathcal{P}(\cdot|\mathbf{x})} \mathbf{c}[k] - \min_{\ell} \mathop{\mathcal{E}}_{\mathbf{c}\sim\mathcal{P}(\cdot|\mathbf{x})} \mathbf{c}[\ell]$$

is V-shaped with respect to $y_{\mathbf{x}} = \operatorname{argmin}_\ell \mathbf{c}_{\mathbf{x}}[\ell] = r_*(\mathbf{x})$ on every point $\mathbf{x} \in \mathcal{X}$. Let $g_+(\mathbf{x},k) \equiv 2\,[\![k < y_{\mathbf{x}}]\!] - 1$. Then g_+ is rank-monotonic and optimal for $\mathcal{P}_b$.

[4] Note that the equivalence in hardness here is qualitative and considers neither the number of independent examples N needed nor the number of classes K.

Proof. By construction g_+ is rank-monotonic. The key is to show $g_*(\mathbf{x}, k) = \text{sign}(\mathbf{c}_\mathbf{x}[k+1] - \mathbf{c}_\mathbf{x}[k])$. Then, because $\mathbf{c}_\mathbf{x}$ is V-shaped, $g_+(\mathbf{x}, k) = g_*(\mathbf{x}, k)$ for all $(\mathbf{x}, k)$ except when $\mathbf{c}_\mathbf{x}[k+1] - \mathbf{c}_\mathbf{x}[k] = 0$. Therefore, $\pi_b(g_+) = \pi_b(g_*)$ and g_+ is optimal for $\mathcal{P}_b$.

Note that if every cost vector $\mathbf{c}$ is convex, the effective cost $\mathbf{c}_\mathbf{x}$ would also be convex, and hence V-shaped. Thus, the convexity of $\mathbf{c}$ is also a (weaker) sufficient condition for the equivalence in hardness between ordinal ranking and binary classification.

As can be seen from the definition of r_*, the effective cost $\mathbf{c}_\mathbf{x}$ conveys sufficient information for determining the optimal prediction at $\mathbf{x}$. Because ordinal ranking predictions should take "closeness" into account (see Section 2), it is reasonable to assume that $\mathbf{c}_\mathbf{x}$ is V-shaped. Hence, in general (with such a minor assumption), ordinal ranking is equivalent to binary classification in terms of hardness.

4 AdaBoost for Ordinal Ranking

We now use reduction and reverse reduction to design a novel boosting approach for ordinal ranking. We shall first introduce the ideas behind the approach. In the training stage, we apply the reduction technique, and take AdaBoost as the core binary classification algorithm. AdaBoost would then train a base binary classifier $\hat{g}_t$ with weighted binary examples in its t-th iteration. We use the reverse reduction technique to replace $\hat{g}_t$ with g_{r_t}, and let the approach train a base ordinal ranker r_t with cost-sensitive ordinal examples instead.

After the training steps above, our approach returns an ensemble of ordinal rankers $H = \{(r_t, v_t)\}_{t=1}^{T}$, where $v_t \geq 0$ is the weight associated with the ordinal ranker r_t. In the prediction stage, we first apply the reverse reduction technique in (4) to cast each ordinal ranker r_t as a joint binary classifier $\hat{g}_t = g_{r_t}$. The weighted votes from all the binary classifiers in the ensemble are gathered to form binary predictions. Then, the reduction technique comes into play, and constructs an ordinal prediction from the binary ones by (1). Combining the steps above, we get the following prediction rule for an ordinal ranking ensemble H:

$$r_H(\mathbf{x}) \equiv 1 + \sum_{k=1}^{K-1} \left[\!\!\left[\sum_{t=1}^{T} v_t \left[\!\left[k < r_t(\mathbf{x}) \right]\!\right] \geq \frac{1}{2} \sum_{t=1}^{T} v_t \right]\!\!\right]. \tag{6}$$

The steps of going back and forth between reduction and reverse reduction may seem complicated. Nevertheless, we can simplify many of them with careful derivations, which are illustrated below.

4.1 Prediction Steps

We shall start with the prediction steps, and derive a simplified form of (6) as follows.

Theorem 4. *(prediction with the weighted median) For any ordinal ranking ensemble* $H = \{(r_t, v_t)\}_{t=1}^{T}$*, assume that* $v_t \geq 0$ *and* $\sum_{t=1}^{T} v_t = 1$*. Then,*

$$r_H(\mathbf{x}) = \min \left\{ k \colon \sum_{t=1}^{T} v_t \left[\!\left[k \geq r_t(\mathbf{x}) \right]\!\right] > \frac{1}{2} \right\}. \tag{7}$$

Proof. Let $k^* = \min \left\{ k \colon \sum_{t=1}^{T} v_t \left[\!\left[k \geq r_t(\mathbf{x}) \right]\!\right] > \frac{1}{2} \right\}$. Then,

$$\sum_{t=1}^{T} v_t \left[\!\left[k \geq r_t(\mathbf{x}) \right]\!\right] > \frac{1}{2}$$

if and only if $k^* \leq k$. That is, $\sum_{t=1}^{T} v_t \left[\!\left[k < r_t(\mathbf{x}) \right]\!\right] \geq \frac{1}{2}$ if and only if $k < k^*$.

Thus, $r_H(\mathbf{x}) = 1 + k^* - 1 = k^*$.

Therefore, the prediction rule (6) that goes back and forth between reduction and reverse reduction can be equivalently performed by computing a simple and intuitive statistic in (7): the weighted median. Note that the rule in (7) is not specific for our approach. It can be applied to ordinal ranking ensembles produced by any ensemble learning approaches, such as bagging [13].

4.2 Training Steps

We now look at the training steps. The steps of the original AdaBoost are listed in Algorithm 1. After plugging AdaBoost into reduction and a base ordinal ranking algorithm into reverse reduction, we can equivalently obtain Algorithm 2: AdaBoost.OR. The equivalence is based on maintaining the following invariance in each iteration.

Lemma 2. *Substitute the indices* m *in Algorithm 1 with* (n, k)*. That is,* $\hat{\mathbf{x}}_m = (\mathbf{x}_n, k)$*,* $\hat{z}_m = \hat{z}_{nk}$*, and* $\hat{w}_m = \hat{w}_{nk}$*. Take* $\hat{g}_t(\mathbf{x}, k) = g_{r_t}(\mathbf{x}, k)$*, and assume that in Algorithms 1 and 2,*

$$\mathbf{c}_n^{(\tau)}[k] = \sum_{\ell=1}^{K-1} \frac{\hat{w}_{n\ell}^{(\tau)}}{K-1} \cdot \left[\!\left[y_n \leq \ell < k \text{ or } k < \ell \leq y_n \right]\!\right] \tag{8}$$

is satisfied for $\tau = t$ *with* $\hat{w}_{n\ell}^{(\tau)} \geq 0$*. Then, equation* (8) *is satisfied for* $\tau = t + 1$ *with* $\hat{w}_{n\ell}^{(\tau)} \geq 0$*.*

Proof. Because (8) is satisfied for $\tau = t$ and $\hat{w}_{n\ell}^{(t)} \geq 0$, the cost vector $\mathbf{c}_n^{(t)}$ is V-shaped with respect to y_n and $\mathbf{c}_n^{(t)}[y_n] = 0$. Thus,

$$\sum_{n=1}^{N} \left(\mathbf{c}_n^{(t)}[1] + \mathbf{c}_n^{(t)}[K] \right) = \sum_{n=1}^{N} \sum_{k=1}^{K-1} \hat{w}_{nk}^{(t)}.$$

Algorithm 1 AdaBoost [8]

Input: examples $\{(\hat{\mathbf{x}}_m, \hat{z}_m, \hat{w}_m)\}_{m=1}^{M}$

Initialize $\hat{w}_m^{(1)} = \hat{w}_m$ for all m.
For $t = 1$ **to** T

1. Obtain $\hat{g}_t$ from the base binary classification algorithm.
2. Compute the weighted training error $\hat{\epsilon}_t$.

$$\hat{\epsilon}_t = \left(\sum_{m=1}^{M} \hat{w}_m^{(t)} \cdot [\![\hat{z}_m \neq \hat{g}_t(\hat{\mathbf{x}}_m)]\!]\right) \Big/ \left(\sum_{m=1}^{M} \hat{w}_m^{(t)}\right)$$

   If $\hat{\epsilon}_t > \frac{1}{2}$, set $T = t - 1$ and abort loop.
3. Let $\hat{v}_t = \frac{1}{2} \log \frac{1-\hat{\epsilon}_t}{\hat{\epsilon}_t}$.
4. Let $\hat{\Lambda}_t = \exp(2\hat{v}_t) - 1$.

$$\hat{w}_m^{(t+1)} = \hat{w}_m^{(t)} + \begin{cases} 0, & \hat{z}_m = \hat{g}_t(\hat{\mathbf{x}}_m) \ ; \\ \hat{\Lambda}_t \hat{w}_m^{(t)}, & \hat{z}_m \neq \hat{g}_t(\hat{\mathbf{x}}_m) \ . \end{cases}$$

Algorithm 2 AdaBoost.OR

Input: examples $\{(\mathbf{x}_n, y_n, \mathbf{c}_n)\}_{n=1}^{N}$

Initialize $\mathbf{c}_n^{(1)}[k] = \mathbf{c}_n[k]$ for all n, k.
For $t = 1$ **to** T

1. Obtain r_t from the base ordinal ranking algorithm.
2. Compute the weighted training error ϵ_t.

$$\epsilon_t = \left(\sum_{n=1}^{N} \mathbf{c}_n^{(t)}[r_t(\mathbf{x})]\right) \Big/ \left(\sum_{n=1}^{N} \mathbf{c}_n^{(t)}[1] + \mathbf{c}_n^{(t)}[K]\right)$$

   If $\epsilon_t > \frac{1}{2}$, set $T = t - 1$ and abort loop.
3. Let $v_t = \frac{1}{2} \log \frac{1-\epsilon_t}{\epsilon_t}$.
4. Let $\Lambda_t = \exp(2v_t) - 1$.
   If $r_t(\mathbf{x}_n) \geq y_n$, then

$$\mathbf{c}_n^{(t+1)}[k] = \mathbf{c}_n^{(t)}[k] + \begin{cases} 0, & k \leq y_n \ ; \\ \Lambda_t \cdot \mathbf{c}_n^{(t)}[r_t(\mathbf{x}_n)] \ , & k > r_t(\mathbf{x}_n) \ ; \\ \Lambda_t \cdot \mathbf{c}_n^{(t)}[k] \ , & \text{otherwise} \ . \end{cases}$$

   Else, switch $>$ to $<$ and vice versa.

In addition, since $\hat{g}_t(\mathbf{x}, k) = g_{r_t}(\mathbf{x}, k)$, by a proof similar to Lemma 1,

$$\sum_{n=1}^{N} \mathbf{c}_n^{(t)}[r_t(\mathbf{x})] = \sum_{n=1}^{N} \sum_{k=1}^{K-1} \hat{w}_{nk}^{(t)} \cdot [\![z_{nk} \neq \hat{g}_t(\mathbf{x}_n, k)]\!] \ .$$

Therefore, $\hat{\epsilon}_t = \epsilon_t$, $\hat{v}_t = v_t$, and $\hat{\Lambda}_t = \Lambda_t$.

Because $\hat{g}_t(\mathbf{x}_n, k) \neq z_{nk}$ if and only if $r_t(\mathbf{x}_n) \leq k < y_n$ or $y_n < k \leq r_t(\mathbf{x}_n)$,

$$\hat{w}_{nk}^{(t+1)} = \begin{cases} \hat{w}_{nk}^{(t)} + \hat{\Lambda}_t \hat{w}_{nk}^{(t+1)}, & y_n < k \leq r_t(\mathbf{x}_n) \\ & \text{or } y_n < k \leq r_t(\mathbf{x}_n) \ ; \\ \hat{w}_{nk}^{(t)}, & \text{otherwise} \ . \end{cases} \tag{9}$$

It is easy to check that $\hat{w}_{nk}^{(t+1)}$ are non-negative. Furthermore, we see that the update rule in Algorithm 2 is equivalent to combining (9) and (8) with $\tau = t+1$. Thus, equation (8) is satisfied for $\tau = t + 1$.

Then, by mathematical induction from $\tau = 1$ up to T with Lemma 2, plugging AdaBoost into reduction and a base ordinal ranking algorithm into reverse

reduction is equivalent to running AdaBoost.OR with the base algorithm. AdaBoost.OR takes AdaBoost as a special case of $K = 2$, and inherits many of its good properties, as discussed below.

4.3 Properties

AdaBoost.OR can take any cost-sensitive base algorithm that produces individual ordinal rankers r_t with errors $\epsilon_t \leq \frac{1}{2}$. In binary classification, the $\frac{1}{2}$ error bound can be naturally achieved by a constant classifier or a fair coin flip. For ordinal ranking, is $\frac{1}{2}$ still easy to achieve? The short answer is yes. In the following theorem, we demonstrate that there always exists a constant ordinal ranker that satisfies the error bound.

Theorem 5. *Define constant ordinal rankers $\tilde{r}_k$ by $\tilde{r}_k(\mathbf{x}) \equiv k$ for all $\mathbf{x}$. For any set $\{\mathbf{c}_n\}_{n=1}^N$, there exists a constant ranker with $k \in \mathcal{K}$ such that*

$$\tilde{\epsilon}_k = \left(\sum_{n=1}^{N} \mathbf{c}_n[\tilde{r}_k(\mathbf{x})]\right) \Big/ \left(\sum_{n=1}^{N} \mathbf{c}_n[1] + \mathbf{c}_n[K]\right) \leq \frac{1}{2}$$

Proof. Either $\tilde{r}_1$ or $\tilde{r}_K$ achieves error $\leq \frac{1}{2}$ because by definition $\tilde{\epsilon}_1 + \tilde{\epsilon}_K = 1$.

Therefore, even the simplest deterministic ordinal rankers can always achieve the desired error bound.[5] If the base ordinal ranking algorithm produces better ordinal rankers, the following theorem bounds the normalized training cost of the final ensemble H.

Theorem 6. *Suppose the base ordinal ranking algorithm produces ordinal rankers with errors $\epsilon_1, \cdots, \epsilon_T$, where each $\epsilon_t \leq \frac{1}{2}$. Let $\gamma_t = \frac{1}{2} - \epsilon_t$, the final ensemble r_H satisfies the following error bound:*

$$\frac{\sum_{n=1}^{N} \mathbf{c}_n[r_H(\mathbf{x}_n)]}{\sum_{n=1}^{N} \mathbf{c}_n[1] + \mathbf{c}_n[K]} \leq \prod_{t=1}^{T} \sqrt{1 - 4\gamma_t^2} \leq \exp\Big(-2\sum_{t=1}^{T} \gamma_t^2\Big).$$

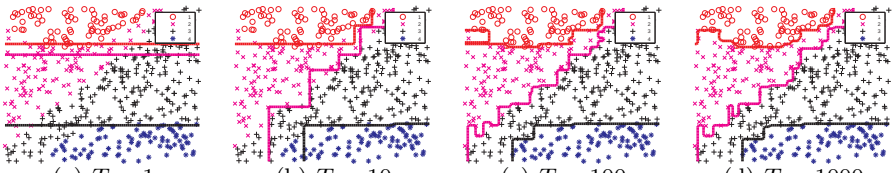
Proof. Similar to the proof for Lemma 2, the left-hand-side of the bound equals

$$\left(\sum_{n=1}^{N}\sum_{k=1}^{K-1} w_{nk} \left[\!\left[z_{nk} \neq g_{\hat{H}}(\mathbf{x}_n, k) \right]\!\right]\right) \Big/ \left(\sum_{n=1}^{N}\sum_{k=1}^{K-1} w_{nk}\right) ,$$

where $\hat{H}$ is a binary classification ensemble $\{(\hat{g}_t, v_t)\}_{t=1}^T$ with $\hat{g}_t = g_{r_t}$. Then, the bound is a simple consequence of the well-known AdaBoost bound [8].

Theorem 6 indicates that if the base algorithm always produces an ordinal ranker with $\epsilon_t \leq \frac{1}{2} - \gamma$ for $\gamma > 0$, the training cost of H would decrease

[5] Similarly, the error bound can be achieved by a randomized ordinal ranker which returns either 1 or K with equal probability.



(a) $T = 1$ (b) $T = 10$ (c) $T = 100$ (d) $T = 1000$

Fig. 2. decision boundaries produced by AdaBoost.OR on an artificial data set

exponentially with T. That is, AdaBoost.OR can rapidly improve the training performance of such a base algorithm.

We can also extend the generalization error bounds of AdaBoost to AdaBoost.OR, including the T-dependent bound [8] and the margin-based ones [14]. The steps for proving these bounds are similar to those for the SVOR bound derived by Li and Lin [5].

5 Experiments

We now demonstrate the validity of AdaBoost.OR. We will first illustrate its behavior on an artificial data set. Then, we test its training and test performance on benchmark data sets.

5.1 Artificial Data

We generate 500 input vectors $\mathbf{x}_n \in [0,1] \times [0,1]$ uniformly, and rank them with $\mathcal{K} = \{1,2,3,4\}$ based on three quadratic boundaries. Then, we apply AdaBoost.OR on these examples with the absolute cost, i.e., $\mathbf{c}[k] \equiv |y-k|$ with respect to the associated y.

We use a simple base algorithm called ORStump, which solves the following optimization problem efficiently with dynamic programming:

$$\begin{aligned} \min_{\theta,d,q} \quad & \sum_{n=1}^{N} \mathbf{c}_n[r(\mathbf{x}_n,\theta,d,q)] \\ \text{subject to} \quad & \theta_1 \le \theta_2 \le \cdots \le \theta_{K-1}, \\ \text{where} \quad & r(\mathbf{x},\theta,d,q) \equiv \max\{k\colon q \cdot (\mathbf{x})_d < \theta_k\}\,. \end{aligned}$$

The ordinal ranking decision stump $r(\cdot,\theta,d,q)$ is a natural extension of the binary decision stump [15]. Note that the set of all possible ordinal ranking decision stumps includes constant ordinal rankers. Therefore, ORStump can always achieve $\epsilon_t \le \frac{1}{2}$.

The decision boundaries generated by AdaBoost.OR with ORStump using $T = 1, 10, 100, 1000$ are shown in Figure 2. The case of $T = 1$ is the same as applying ORStump directly on the artificial set, and we can see that its resulting decision boundary cannot capture the full characteristic of the data. As T gets larger, however, AdaBoost.OR is able to boost up ORStump to form

more sophisticated boundaries that approximate the underlying quadratic curves better.

5.2 Benchmark Data

Next, we run AdaBoost.OR on eight benchmark data sets[6] with the absolute cost. We keep the splits provided by Chu and Keerthi [1], and also average the results over 20 trials. Thus, our results can be fairly compared with their benchmark SVOR ones.

We couple AdaBoost.OR with three base algorithms: ORStump, PRank [4], and a reduction-based formulation of SVOR [1] proposed by Li and Lin [5]. For the PRank algorithm, we adopt the SiPrank variant, and make it cost-sensitive by presenting random examples $(\mathbf{x}_n, y_n)$ with probability proportional to $\max_{k \in \mathcal{K}} \mathbf{c}_n[k]$. In addition, we apply the pocket technique with ratchet [16] for 2000 epochs to get a decent training cost minimizer. For SVOR, we follow the same setting of Li and Lin [5] except for the choice of parameter. In particular, we set the parameter C in the t-th iteration as the smallest number within $\left\{2^{-20}, 2^{-18}, \cdots, 2^{20}\right\}$ that makes $\epsilon_t \leq 0.3$. The setup guarantees that SVOR produces a decent training cost minimizer without overfitting the training examples too much.

We run AdaBoost.OR for $T = 1000, 100, 10$ iterations for ORStump, PRank, and SVOR respectively. Such a setup is intended to compensate the computational complexity of each individual base algorithm. Nevertheless, a more sophisticated choice of T should further improve the performance of AdaBoost.OR.

For each algorithm, the average training cost as well as its standard error is reported in Table 1; the average test cost and its standard error is reported in Table 2. For each pair of single and AdaBoost.OR (short-handed AB.OR) entries, we mark those within one standard error of the lowest in bold. We also list the benchmark SVOR results from Chu and Keerthi [1] in Table 2, and mark the entries better than the benchmark ones with †.

From the tables, we see that AdaBoost.OR almost always improves both the training and test performance of the base algorithm significantly, especially for ORStump and SVOR. It is harder for AdaBoost.OR to improve the performance of PRank, because it sometimes cannot produce a good r_t in terms of minimizing the training cost even with the pocket technique.

Note that a single-shot execution of the SVOR base algorithm is much worse than the benchmark SVOR in terms of test cost. The difference can be explained by looking at their parameter selection procedures. The SVOR base algorithm chooses its parameter by only the training cost to guarantee $\epsilon_t \leq \frac{1}{2}$, which is the condition that allows AdaBoost.OR to work (see Theorem 6). On the other hand, the benchmark SVOR goes through a complete parameter selection procedure using cross-validation, which justifies its good test performance but is quite time-consuming. On the other hand, AdaBoost.OR (especially with ORStump) is

[6] They are pyrimdines, machineCPU, boston, abalone, bank, computer, california, and census [1].

Table 1. average cost of ordinal ranking algorithms on the training set

data set	ORStump single	ORStump AB.OR	PRank single	PRank AB.OR	SVOR single	SVOR AB.OR
pyr.	1.76 ± 0.02	$\mathbf{0.02 \pm 0.01}$	0.46 ± 0.03	$\mathbf{0.27 \pm 0.05}$	2.44 ± 0.04	$\mathbf{0.18 \pm 0.02}$
mac.	1.12 ± 0.02	$\mathbf{0.12 \pm 0.01}$	0.88 ± 0.01	$\mathbf{0.86 \pm 0.01}$	2.49 ± 0.03	$\mathbf{0.36 \pm 0.02}$
bos.	1.05 ± 0.01	$\mathbf{0.00 \pm 0.00}$	0.85 ± 0.01	$\mathbf{0.83 \pm 0.01}$	2.43 ± 0.01	$\mathbf{0.29 \pm 0.02}$
aba.	1.53 ± 0.01	$\mathbf{1.05 \pm 0.01}$	$\mathbf{1.44 \pm 0.01}$	$\mathbf{1.44 \pm 0.01}$	2.63 ± 0.01	$\mathbf{0.39 \pm 0.01}$
ban.	1.98 ± 0.01	$\mathbf{1.14 \pm 0.00}$	1.51 ± 0.00	$\mathbf{1.47 \pm 0.00}$	1.63 ± 0.07	$\mathbf{0.18 \pm 0.02}$
com.	1.18 ± 0.00	$\mathbf{0.50 \pm 0.00}$	$\mathbf{0.66 \pm 0.00}$	$\mathbf{0.66 \pm 0.00}$	2.51 ± 0.01	$\mathbf{0.35 \pm 0.01}$
cal.	1.62 ± 0.00	$\mathbf{0.88 \pm 0.00}$	$\mathbf{1.21 \pm 0.00}$	$\mathbf{1.21 \pm 0.00}$	2.61 ± 0.01	$\mathbf{0.52 \pm 0.01}$
cen.	1.83 ± 0.00	$\mathbf{1.11 \pm 0.00}$	1.58 ± 0.01	$\mathbf{1.56 \pm 0.01}$	2.51 ± 0.00	$\mathbf{0.43 \pm 0.02}$

(results that are as significant as the best one of each pair are marked in bold)

Table 2. average cost of ordinal ranking algorithms on the test set

data set	ORStump single	ORStump AB.OR	PRank single	PRank AB.OR	SVOR single	SVOR AB.OR	benchmark result
pyr.	1.91 ± 0.09	$\mathbf{1.24 \pm 0.05}^{\dagger}$	1.57 ± 0.07	$\mathbf{1.42 \pm 0.07}$	2.63 ± 0.10	$\mathbf{1.36 \pm 0.05}$	1.294
mac.	1.29 ± 0.04	$\mathbf{0.84 \pm 0.02}^{\dagger}$	0.97 ± 0.01	$\mathbf{0.93 \pm 0.02}^{\dagger}$	2.62 ± 0.04	$\mathbf{0.93 \pm 0.03}^{\dagger}$	0.990
bos.	1.17 ± 0.01	$\mathbf{0.89 \pm 0.01}$	0.91 ± 0.01	$\mathbf{0.89 \pm 0.01}$	2.46 ± 0.03	$\mathbf{0.80 \pm 0.01}$	0.747
aba.	1.59 ± 0.00	$\mathbf{1.48 \pm 0.00}$	$\mathbf{1.48 \pm 0.01}$	$\mathbf{1.48 \pm 0.01}$	2.65 ± 0.02	$\mathbf{1.53 \pm 0.01}$	1.361
ban.	2.00 ± 0.00	$\mathbf{1.53 \pm 0.00}$	1.54 ± 0.00	$\mathbf{1.50 \pm 0.00}$	1.68 ± 0.07	$\mathbf{1.48 \pm 0.00}$	1.393
com.	1.20 ± 0.00	$\mathbf{0.63 \pm 0.00}$	$\mathbf{0.66 \pm 0.00}$	$\mathbf{0.66 \pm 0.00}$	2.51 ± 0.01	$\mathbf{0.67 \pm 0.01}$	0.596
cal.	1.64 ± 0.00	$\mathbf{1.00 \pm 0.00}^{\dagger}$	$\mathbf{1.21 \pm 0.00}$	$\mathbf{1.21 \pm 0.00}$	2.61 ± 0.01	$\mathbf{1.10 \pm 0.00}$	1.008
cen.	1.85 ± 0.00	$\mathbf{1.25 \pm 0.00}$	1.60 ± 0.01	$\mathbf{1.58 \pm 0.00}$	2.51 ± 0.01	$\mathbf{1.25 \pm 0.01}$	1.205

(results that are better than the benchmark one are marked with †)

(results that are as significant as the best one of each pair are marked in bold)

faster in training and can achieve a decent performance without resorting to the cross-validation steps. The efficiency along with the comparable performance can make AdaBoost.OR a promising alternative for some application needs.

6 Conclusion

We presented the reverse reduction technique between ordinal ranking and binary classification. The technique complemented the reduction method of Li and Lin [5], and allowed us to derive a novel regret bound for ordinal ranking. Furthermore, we used the technique to prove that ordinal ranking is generally equivalent to binary classification in hardness.

We also used reduction and reverse reduction to design a novel boosting approach, AdaBoost.OR, to improve the performance of any cost-sensitive base ordinal ranking algorithm. We showed the parallel between AdaBoost.OR and AdaBoost in algorithmic steps and in theoretical properties. Experimental results validated that AdaBoost.OR indeed improved both the training and test performance of existing ordinal ranking algorithms.

Acknowledgment

We thank Yaser Abu-Mostafa, Amrit Pratap, and the anonymous reviewers for valuable suggestions. Hsuan-Tien Lin was partly sponsored by the Caltech Division of Engineering and Applied Science Fellowship when initiating this project, and the continuing work was supported by the National Science Council of Taiwan via the grant NSC 98-2218-E-002-019.

References

[1] Wei Chu and S. Sathiya Keerthi. New approaches to support vector ordinal regression. In Luc De Raedt and Stefan Wrobel, editors, *Proceedings of ICML 2005*, pages 145–152. ACM, 2005.

[2] Eibe Frank and Mark Hall. A simple approach to ordinal classification. In Luc De Raedt and Peter Flach, editors, *Proceedings of ECML 2001*, volume 2167 of *Lecture Notes in Artificial Intelligence*, pages 145–156. Springer-Verlag, 2001.

[3] Ralf Herbrich, Thore Graepel, and Klaus Obermayer. Large margin rank boundaries for ordinal regression. In Alexander J. Smola, Peter J. Bartlett, Bernhard Schökopf, and Dale Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 115–132. MIT Press, 2000.

[4] Koby Crammer and Yoram Singer. Online ranking by projecting. *Neural Computation*, 17:145–175, 2005.

[5] Ling Li and Hsuan-Tien Lin. Ordinal regression by extended binary classification. In Bernhard Schölkopf, John C. Platt, and Thomas Hofmann, editors, *Proceedings of NIPS 2006*, volume 19, pages 865–872. MIT Press, 2007.

[6] Wei Chu and Zoubin Ghahramani. Gaussian processes for ordinal regression. *Journal of Machine Learning Research*, 6:1019–1041, 2005.

[7] Ron Meir and Gunnar Rätsch. An introduction to boosting and leveraging. In S. Mendelson and A. J. Smola, editors, *Advanced Lectures on Machine Learning*, volume 2600 of *Lecture Notes in Computer Science*, pages 118–183. Springer-Verlag, 2003.

[8] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[9] Yoav Freund, Raj Iyer, Robert E. Shapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, 4:933–969, 2003.

[10] Hsuan-Tien Lin and Ling Li. Large-margin thresholded ensembles for ordinal regression: Theory and practice. In José L. Balcazár, Philip M. Long, and Frank Stephan, editors, *Proceedings of ALT 2006*, volume 4264 of *Lecture Notes in Artificial Intelligence*, pages 319–333. Springer-Verlag, 2006.

[11] Naoki Abe, Bianca Zadrozny, and John Langford. An iterative method for multi-class cost-sensitive learning. In Won Kim, Ron Kohavi, Johannes Gehrke, and William DuMouchel, editors, *Proceedings of KDD 2004*, pages 3–11. ACM, 2004.

[12] Hsuan-Tien Lin. *From Ordinal Ranking to Binary Classification*. PhD thesis, California Institute of Technology, 2008.

[13] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[14] Robert E. Schapire, Yoav Freund, Peter Bartlett, and Wee Sun Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics*, 26(5):1651–1686, 1998.

[15] Robert C. Holte. Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, 11(1):63–91, 1993.
[16] Stephen I. Gallant. Perceptron-based learning algorithms. *IEEE Transactions on Neural Networks*, 1(2):179–191, 1990.