Machine Learning (2020) 109:51–78
https://doi.org/10.1007/s10994-019-05826-x

RANKBOOST+: an improvement to RANKBOOST

Harold Connamacher[1] · Nikil Pancha[1] · Rui Liu[1] · Soumya Ray[1]

Received: 28 January 2018 / Revised: 20 June 2019 / Accepted: 5 July 2019 / Published online: 12 August 2019

Abstract
RANKBOOST is a well-known algorithm that iteratively creates and aggregates a collection of "weak rankers" to build an effective ranking procedure. Initial work on RANKBOOST proposed two variants. One variant, that we call RB-D and which is designed for the scenario where all weak rankers have the binary range $\{0, 1\}$, has good theoretical properties, but does not perform well in practice. The other, that we call RB-C, has good empirical behavior and is the recommended variation for this binary weak ranker scenario but lacks a theoretical grounding. In this paper, we rectify this situation by proposing an improved RANKBOOST algorithm for the binary weak ranker scenario that we call RANKBOOST+. We prove that this approach is theoretically sound and also show empirically that it outperforms both RANKBOOST variants in practice. Further, the theory behind RANKBOOST+ helps us to explain why RB-D may not perform well in practice, and why RB-C is better behaved in the binary weak ranker scenario, as has been observed in prior work.

Keywords Ranking · Boosting · Ensemble methods · Rankboost

1 Introduction

In a ranking task, a learner is given a set of preferences, often organized pairwise, over instances. For instance, in a movie recommendation task, the learner might be told that Alice likes "2001: A Space Odyssey" better than "Interstellar," and similar facts. The learner needs to produce a function that then correctly ranks novel instances. In this example, this function

Editors: Jesse Davis, Elisa Fromont, Derek Greene, and Bjorn Bringmann.

✉ Harold Connamacher
harold.connamacher@case.edu

Nikil Pancha
nikil.pancha@case.edu

Rui Liu
rui.liu4@case.edu

Soumya Ray
sray@case.edu

1 Department of Electrical Engineering and Computer Science, Case Western Reserve University, 10900 Euclid Ave, Cleveland, OH 44106, USA

would be expected to rank new movies in Alice's order of preference. Ranking has many applications in areas such as drug design (Agarwal et al. 2010), information retrieval (Nuray and Can 2006) and of course recommendation systems (Freund et al. 2003).

An elegant way to solve the ranking task is through a "boosting" algorithm. In classification, algorithms such as AdaBoost (Freund and Schapire 1995) iteratively learn and aggregate a collection of "base learners" into a solution that is very accurate. Base learners are only required to satisfy a "weak learning" criterion, which means they only need to be slightly better than chance on the learning task. This is attractive for at least two reasons. First, for many domains, finding a classifier that satisfies weak learning can be easier than finding one that is very accurate. Second, boosting algorithms for classification are known to possess desirable theoretical properties (Freund and Schapire 1995). We may hope to extend these characteristics to the ranking problem if a suitable boosting algorithm for ranking can be designed.

Indeed, AdaBoost was directly extended to the ranking problem by a framework called RankBoost (Freund et al. 2003), which is our focus in this paper. RankBoost shows how to combine a collection of weak rankers into an effective ranking procedure, as described below. A version of RankBoost, which we call Rb-d in this paper, has been shown to possess good theoretical properties (Freund et al. 2003; Mohri et al. 2012). A different version, which we call Rb-c in this paper, has been used to solve many ranking tasks with good results (Cortes and Mohri 2004; Cao et al. 2007; Zheng et al. 2008; Agarwal et al. 2012; Aslam et al. 2009). These implementations are described in the next section.

While RankBoost is a well-established algorithm for boosting for ranking, there is a gap in our understanding of the approach. In particular, the theoretical understanding of RankBoost applies to Rb-d; however, experiments in the original paper show it underperforms Rb-c, which does *not* have a similar theoretical justification, on the very ranking scenarios for which Rb-d was designed. We verify this behavior in our experiments on real data. Thus the version that has a theoretical justification does not work very well in practice, while the version that works better in practice has limited theoretical support.

In this paper, we address this gap. We propose an approach we call RankBoost+, which is built on the RankBoost framework. We show that RankBoost+ has the good theoretical properties of Rb-d. However, we show that RankBoost+ is also closely related to Rb-c, and our experiments show that it significantly outperforms both Rb-d and Rb-c in a number of real-world ranking tasks. Further, the theory of RankBoost+ also gives insight into why Rb-d underperforms in practice, and explains why Rb-c outperforms Rb-d in the ranking scenario and metric that Rb-d was designed to optimize.

In the following sections, we first describe RankBoost and the two variants, Rb-d and Rb-c. Next we motivate and present the RankBoost+ algorithm. Then we discuss the theoretical properties of RankBoost+ and explain why Rb-c outperforms Rb-d in practice. Finally, we evaluate these approaches empirically on several real-world ranking problems and show that the empirical results are in excellent agreement with what the theory predicts.

2 RankBoost

Let $\mathscr{X}$ be an instance space, and let $c : \mathscr{X} \times \mathscr{X} \rightarrow \{-1, 0, 1\}$ be a target labeling function defined as:

$$c(x, x') = \begin{cases} +1 & \text{if} \quad x' \text{ is ranked higher than } x, \\ -1 & \text{if} \quad x \text{ is ranked higher than } x', \text{ and} \\ 0 & \text{if} \quad \text{no preference.} \end{cases} \tag{1}$$

Note that this ranking scenario does not permit us to specify that x and x' should be ranked the same. Also note that we do not assume that the order induced by c is transitive.

Let $\mathscr{H}$ be the set of ranking functions $h : \mathscr{X} \rightarrow \mathbb{R}$. If $h(x) > h(y)$ we state that h ranks x higher than y. Let D be the distribution over $\mathscr{X} \times \mathscr{X}$. The original RankBoost paper (Freund et al. 2003) defines the generalization error of ranker h on distribution D to be the probability that, given a pair of elements, we have a preference about how they are ranked but h does not correctly rank the pair.

$$R_1(h) = \underset{(x,x')\sim D}{Pr}[(c(x,x') \neq 0) \wedge (c(x,x')(h(x') - h(x)) \leq 0)]. \tag{2}$$

RankBoost is a supervised learning algorithm that is trained on a subset of data drawn from $\mathscr{X} \times \mathscr{X}$ using distribution D, and RankBoost is given the correct ranking for this subset. More formally, consider the labeled sample S defined as $S = \{(x_1, x'_1, y_1), \ldots, (x_m, x'_m, y_m)\}$ where, for each $i \in \{1, \ldots, m\}$, $y_i = c(x_i, x'_i)$, and (x_i, x'_i) is drawn i.i.d. according to distribution D. Any pair (x_i, x'_i) with $y_i \neq 0$ is called a *critical pair*. As is standard in the literature, we simplify the presentation by assuming S contains only critical pairs.

Next we define *rank loss functions* on the sample that are analogous to the generalization error. The original RankBoost paper (Freund et al. 2003) defines two different rank loss functions for ranker h on sample S. We denote these functions as $\hat{R}_1$ and $\hat{R}_2$.

$$\hat{R}_1(h) = \frac{1}{m}\sum_{i=1}^{m} \mathbf{1}_{y_i(h(x'_i)-h(x_i))\leq 0}. \tag{3}$$

$$\hat{R}_2(h) = \frac{1}{m}\sum_{i=1}^{m} \mathbf{1}_{y_i(h(x'_i)-h(x_i))<0} + \frac{1}{2}\mathbf{1}_{y_i(h(x'_i)-h(x_i))=0}. \tag{4}$$

$\hat{R}_1$ treats ties as equally bad as reverse rankings. The justification for $\hat{R}_1$ is that it is the sample estimate of R_1 on ranker h, and in Freund et al. (2003) the authors design RankBoost to minimize $\hat{R}_1$. We highlight in this paper some advantages to minimizing $\hat{R}_1$ as well some issues that result from this choice. $\hat{R}_2$, on the other hand, gives ties half the error as a reverse ranking. The intuition, given in Freund et al. (2003), for why $\hat{R}_2$ is also a good rank loss function is that if we break tie pairs randomly in our output ranking, we expect half of the tied critical pairs to receive the correct ranking. Therefore, $\hat{R}_2$ is the sample estimate of R_1 on the ranking produced by h if we break ties randomly. While RankBoost is designed to minimize $\hat{R}_1$, Freund et al. (2003) uses $\hat{R}_2$ to evaluate RankBoost on real world datasets.

Both $\hat{R}_1$ and $\hat{R}_2$ are non-convex functions, and minimizing either is an NP-complete problem (Cohen et al. 1999). RankBoost is a direct adaptation of the seminal classification boosting algorithm AdaBoost (Freund and Schapire 1995), and $\hat{R}_1$ is a direct analog to the empirical loss function used with AdaBoost. Just as with AdaBoost, RankBoost does not minimize $\hat{R}_1$ directly. Instead, it minimizes an upper bound on $\hat{R}_1$ that is the following *exponential loss function*.

$$\hat{E}_1(h) = \frac{1}{m}\sum_{i=1}^{m} e^{-y_i(h(x'_i)-h(x_i))}. \tag{5}$$

Although (5) is a better behaved function than $\hat{R}_1$, it is typically difficult to find the element of $\mathscr{H}$ that minimizes (5). Instead, we assume that we have access to a routine that can only find *weak rankers*, elements of $\mathscr{H}$ so named because the ranking they produce is only weakly correlated to the desired ranking. The goal of RankBoost is to learn a linear combination of these weak rankers that is highly accurate. A common ranking scenario, and the one considered in this paper, is to further restrict the weak rankers used by RankBoost to be *binary weak rankers*: members of $\mathscr{H}$ that map each element of $\mathscr{X}$ to only two values, {0, 1}. Formally, let $f_1, \ldots, f_N$ be the finite set of binary weak rankers of $\mathscr{H}$ that RankBoost considers during its execution. Let $\boldsymbol{\eta} = [\eta_1, \ldots, \eta_N]$ be an element of $\mathbb{R}^N$. We define the function $F_1 : \mathbb{R}^N \to \mathbb{R}$ as:

$$F_1(\boldsymbol{\eta}) = \hat{E}_1\left(\sum_{s=1}^{N} \eta_s f_s\right) = \frac{1}{m}\sum_{i=1}^{m} e^{\sum_{s=1}^{N} y_i \eta_s (f_s(x_i') - f_s(x_i))}. \tag{6}$$

As (6) is a convex function, it has a unique minimum. The goal of RankBoost is to iteratively adjust the η_s values in order to quickly converge to $\boldsymbol{\eta^*}$, the vector that achieves the minimum of (6).

RankBoost achieves this minimization by iteratively building an ensemble ranker that is a linear combination of binary weak rankers. At iteration t, the ensemble ranker is $g_{t-1} = \sum_{j=1}^{t-1} \alpha_j h_j$, where $h_j \in \{f_1, \ldots f_N\}$ is the weak ranker chosen at iteration j. During the execution, RankBoost maintains a distribution $D_t(i)$ over the critical pairs (x_i', x_i) at iteration t. The idea is for critical pairs not correctly ranked by g_{t-1} to have higher probability in D_t than those pairs correctly ranked. RankBoost learns a weak ranker $h_t : \mathscr{X} \to \{0, 1\}$ with small expected error over D_t. Ranker h_t is added to the ensemble ranker with weight α_t. Then a new distribution D_{t+1} is created by multiplying the probability for each pair by a scaling function ω_t, that will be defined later, that increases the probability of the pairs h_t misranks and decreases the probability of the pairs h_t correctly ranks:

$$D_{t+1}(i) = \frac{D_t(i)\boldsymbol{\omega}_t(i)}{Z_t}. \tag{7}$$

Z_t is a normalizing factor to insure that the probabilities add to one.

RankBoost is an algorithm framework, listed below as Algorithm 1, and it can be instantiated in multiple ways. In this paper, we focus on two proposed versions from prior work (Freund et al. 2003). We denote as Rb-d, for *discrete* RankBoost, the version of RankBoost where the only weak rankers considered by the algorithm are binary weak rankers. This version uses Eq. (14) on line 7 of Algorithm 1 to calculate the appropriate weight α_i for weak ranker h_i. Rb-d is the version of RankBoost used in the theoretical analysis of ranking in prior work (Freund et al. 2003). We denote as Rb-c, for *continuous* RankBoost, the version of RankBoost where the weak rankers map each element of $\mathscr{X}$ to [0, 1]. This version uses Eq. (19) on line 7 to calculate the appropriate weight α_i for weak ranker h_i. As mentioned above, this paper is focused on the ranking scenario where we only have binary weak rankers, but even in this scenario Rb-c generally outperforms Rb-d. Rb-c is the version of RankBoost most often used in practice.

2.1 The discrete variation of RankBoost

As noted above, Rb-d is the version of RankBoost about which we have theoretical results, and the values assigned in lines 6, 7, and 9 of Rb-d minimize $\hat{E}_1\left(\sum_{s=1}^{t} \alpha_s h_s\right)$. We define the values ε_t^{+1}, ε_t^{0}, and ε_t^{-1} as

$$\varepsilon_t^{\tau} = \sum_{i=1}^{m} D_t(i)\mathbf{1}_{y_i(h_t(x_i') - h_t(x_i)) = \tau}, \tag{8}$$

Algorithm 1 Template Pseudocode for RB-D, RB-C and RANKBOOST+

```
1: function RANKBOOST((x_1, x'_1, y_1), ··· , (x_m, x'_m, y_m))
2:   for i = 1 to m do
3:     D_1(i) = 1/m
4:   end for
5:   for t = 1 to T do
6:     h_t = best weak ranker (Eq. (13) for RB-D, RB-C or Eq. (32) for RANKBOOST+)
7:     Compute α_t (Eq. (14) for RB-D, Eq. (19) for RB-C, or Eq. (34) for RANKBOOST+)
8:     for i = 1 to m do
9:       Compute D_{t+1}(i) (Eq. (10) for RB-D, RB-C, or Eq. (26) for RANKBOOST+)
10:    end for
11:  end for
12:  g = Σ_{t=1}^T α_t h_t
13:  return g
14: end function
```

and we let ε_t^+ and ε_t^- stand for ε_t^{+1} and ε_t^{-1}, respectively. Line 9 of RB-D uses Eq. (7) to set D_{t+1}. Just as with ADABOOST, the RB-D scaling function maps the i^{th} critical pair to its contribution to $\hat{E}_1(\alpha_t h_t)$.

$$\omega_t(i) = \begin{cases} e^{-\alpha_t} & \text{if pair } i \text{ correctly ranked by } h_t, \\ e^{\alpha_t} & \text{if pair } i \text{ reverse ranked by } h_t, \\ 1 & \text{if pair } i \text{ tied by } h_t. \end{cases} \tag{9}$$

Thus, Eq. (7) is equivalent to

$$D_{t+1}(i) = \frac{D_t(i)\exp\left(-\alpha_t y_i (h_t(x_i') - h_t(x_i))\right)}{Z_t} \tag{10}$$

where Z_t, the normalization factor, is

$$Z_t = \varepsilon_t^0 + \varepsilon_t^+ e^{-\alpha_t} + \varepsilon_t^- e^{\alpha_t}. \tag{11}$$

A simple induction argument with (10) gives

$$\hat{E}_1\left(\sum_{s=1}^{t} \alpha_s h_s\right) = \prod_{s=1}^{t} Z_s. \tag{12}$$

Equation (12) is fundamental to RB-D. It proves that if we can minimize each Z_t, the hypothesis output by RB-D minimizes $\hat{E}_1$. Therefore, at line 6 RB-D chooses the weak ranker that minimizes the expression $\delta(h_t) = \varepsilon_t^- - \varepsilon_t^+$,

$$h_t = \arg\min_{h \in \mathscr{H}'} \delta(h) \tag{13}$$

where $\mathscr{H}'$ is the set of weak rankers, and at line 7, RB-D defines

$$\alpha_t = \frac{1}{2} \log \frac{\varepsilon_t^+}{\varepsilon_t^-}. \tag{14}$$

Such an h_t and α_t produce the minimum Z_t value.

$\hat{E}_1$ has a nice mathematical property exploited by RB-D. The error contribution of pair i is $\exp[-y_i(h(x_i') - h(x_i))]$, and

$$e^{-y_i((\alpha_1 h_1 + \alpha_2 h_2)(x_i') - (\alpha_1 h_1 + \alpha_2 h_2)(x_i))} = e^{-y_i \alpha_1 h_1(x_i')} e^{-y_i \alpha_2 h_2(x_i')} e^{y_i \alpha_1 h_1(x_i)} e^{y_i \alpha_2 h_2(x_i)}. \tag{15}$$

This property allows RB-D to be simple. For example, RB-D can reuse the same ranker in different iterations without any adjustments to the algorithm. Also, this property is used in the *bipartite ranking scenario* where $\mathscr{X}$ is partitioned into two sets $\mathscr{X}_0$ and $\mathscr{X}_1$ where all elements of $\mathscr{X}_0$ are ranked above all elements of $\mathscr{X}_1$, and the ranking makes no distinction on pairs otherwise. In this ranking scenario, we exploit (15) to replace the distribution over pairs with distributions over instances, and the result is an efficient algorithm that gives the same result as RB-D. See Freund et al. (2003) for details. This paper, though, is not considering the bipartite ranking scenario.

We present two important theoretical properties of RB-D. Observation 1 is from Rudin et al. (2005), and it follows from the fact that $\hat{E}_1$ is a convex function and thus has a unique global minimum. Theorem 1, found in Mohri et al. (2012), is a direct analog of an equivalent theorem for ADABOOST of Freund and Schapire (1995). The theorem states that the ranking loss of RB-D decreases exponentially with respect to boosting rounds.

Observation 1 (Rudin et al. 2005) ***Assuming that*** RB-D ***allows negative*** α ***values and assuming that*** (14) ***is well-defined in all rounds of the algorithm,*** RB-D *is a coordinate descent algorithm that converges to the global minimum of* $\hat{E}_1$ *on the vector space spanned by the weak rankers of* $\mathscr{H}$.

Theorem 1 (Mohri et al. 2012) ***Assuming that*** (14) ***is well-defined in all rounds of the algorithm,*** *the empirical error of the hypothesis g returned by* RB-D *satisfies*

$$\hat{R}_1(g) \leq \exp\left[-2\sum_{t=1}^{T}\left(\frac{\delta(h_t)}{2}\right)^2\right] = \exp\left[-2\sum_{t=1}^{T}\left(\frac{\varepsilon_t^+ - \varepsilon_t^-}{2}\right)^2\right]. \tag{16}$$

It is useful to note that the assumptions in boldface above are often unstated in the literature, or in some cases do not hold (for example, p. 6 of Rudin et al. 2005 and p. 218 and Fig. 9.1, line 4 of Mohri et al. 2012 assume nonnegative α values for RB-D). If the assumptions do not hold, we can create ranking scenarios that would violate the theoretical guarantees. We provide these counterexamples in Sect. B in the appendix for completeness.

2.2 Opportunities for improvement

We noted above that Freund et al. (2003) identifies two rank loss functions. The fact that Freund et al. (2003) uses $\hat{R}_2$ to evaluate the performance of RB-D suggests that there are many ranking scenarios for which $\hat{R}_2$ may be a better measure of rank loss than $\hat{R}_1$. In addition, there are issues with using $\hat{E}_1$ as an upper bound for $\hat{R}_1$. Note that $\hat{E}_1$ treats ties differently than $\hat{R}_1$. Specifically, $\hat{E}_1$ decreases the weight of ties relative to both reverse ranked and correctly ranked pairs. This causes $\hat{R}_1$ and $\hat{E}_1$ to order the weak rankers differently.

The functions $\hat{R}_1$ and $\hat{E}_1$ each give a quasiordering on the rankers of $\mathscr{H}$. Ideally, we would like them to induce the same ordering. It may be unrealistic for an exponential loss function to give the same ordering as its corresponding rank loss function for all rankers in $\mathscr{H}$. However, since Algorithm 1 is iteratively finding a minimal binary weak ranker to add to the ensemble, we argue that a reasonable property for the exponential loss function is for it to give the same ordering *on binary weak rankers* as the rank loss function. Because $\hat{R}_1$ and $\hat{E}_1$ treat ties differently, there are natural datasets for which $\hat{R}_1$ and $\hat{E}_1$ do not induce the same quasiordering on the set of all binary weak rankers. The proof of this proposition is in Sect. A in the appendix.

Proposition 1 *There exist datasets and binary weak rankers h_1 and h_2 such that $\hat{R}_1(h_1) > \hat{R}_1(h_2)$ but $\hat{E}_1(h_1) < \hat{E}_1(h_2)$, $\hat{E}_1(\alpha_{h_1} h_1) < \hat{E}_1(\alpha_{h_2} h_2)$ where α_{h_1} and α_{h_2} are the weights assigned by* RB-D *given distribution D_1, and $\hat{E}_1(\alpha^c_{h_1} h_1) < \hat{E}_1(\alpha^c_{h_2} h_2)$ where $\alpha^c_{h_1}$ and $\alpha^c_{h_2}$ are the weights assigned by* RB-C *given distribution D_1.*

2.3 The continuous variation of RANKBOOST

While RB-D is the version of RANKBOOST used in theoretical analysis, RB-C is the version most often used in practice. In RB-C, the weak rankers can map elements to the range [0, 1], and the only change in Algorithm 1 is in the computation of α_t on line 6. While we know of no theorems explicitly about the behavior of RB-C, RB-C generally outperforms RB-D even in the scenario of binary weak rankers for which RB-D was designed (Freund et al. 2003).

RB-C still has $\hat{E}_1$ as the error approximation that should be minimized. However, to find the value for α_t, RB-C does not minimize Z_t but an upper bound on Z_t. Let,

$$r_t = \sum_{i=1}^{m} D_t(i) y_i (h_t(x'_i) - h_t(x_i)). \tag{17}$$

The upper bound for Z_t used by RB-C is

$$Z_t \leq \left(\frac{1 - r_t}{2}\right) e^{\alpha_t} + \left(\frac{1 + r_t}{2}\right) e^{-\alpha_t}, \tag{18}$$

and the α_t that minimizes the right hand side of (18) is

$$\alpha_t = \frac{1}{2} \log \frac{1 + r_t}{1 - r_t}. \tag{19}$$

A nice property of RB-C is that (19) is well defined in all but perfect rankers. However, the theoretical guarantees of Observation 1 and Theorem 1 are not known to hold for RB-C. RB-C generally outperforms RB-D in practice, and in Sect. 3.4 below we give mathematical intuition for why we see this behavior.

3 Improving RANKBOOST

In this section, we propose RANKBOOST+, an improvement to RANKBOOST. The inspiration for RANKBOOST+ comes from the issues observed above with using $\hat{E}_1$ as an upper bound for $\hat{R}_1$ and the observation in Freund et al. (2003) that in many ranking scenarios $\hat{R}_2$ may be a better loss function for ranking. In this section, we first derive $\hat{E}_2$, an exponential loss function that orders the weak rankers the same as $\hat{R}_2$. We then follow the same RANKBOOST framework and identify the weak ranker, the weight to give it, and the distribution update rule that minimizes $\hat{E}_2$. Doing this gives us the same theoretical guarantees as RB-D, but without the often unstated assumptions and without the issues of $\hat{E}_1$. In addition, this work provides mathematical justification for why RB-C outperforms RB-D. In the next section we test RANKBOOST+ on real world data sets and demonstrate that RANKBOOST+ outperforms both RB-D and RB-C. To keep the presentation in this section clean, the proofs of the propositions and theorems are placed in Sect. A of the appendix.

3.1 Defining $\hat{E}_2$

In designing RankBoost+, we keep to the framework of RankBoost. Equation (6) above defines F_1 on the vector $\boldsymbol{\eta} = \{\eta_1, \ldots, \eta_N\}$. We define an analogous loss function F_2 on $\boldsymbol{\eta}$, and then from F_2 we get an equivalent loss function $\hat{E}_2$ defined on the ensemble ranker $\sum_{s=1}^{N} \eta_s f_s$. For the purpose of defining F_2, we write F_1 in an equivalent but slightly different manner:

$$F_1(\boldsymbol{\eta}) = \frac{1}{m} \sum_{i=1}^{m} e^{\sum_{s=1}^{N} \ln \omega_1^*(i, f_s, \eta_s)} \tag{20}$$

where

$$\omega_1^*(i, f_s, \eta_s) = \begin{cases} e^{-\eta_s} & \text{if pair } i \text{ correctly ranked by } f_s, \\ e^{\eta_s} & \text{if reverse ranked by } f_s, \text{ and} \\ 1 & \text{if tied by } f_s. \end{cases} \tag{21}$$

We use (20) as our template to define F_2, and we define F_2 to use the same exponential loss values as F_1 for pairs $\eta_s f_s$ either correctly ranks or reverse ranks: $e^{-\eta_s}$ and e^{η_s}, respectively. The only difference between F_2 and F_1 is on tied pairs. For ties F_2 follows the behavior of $\hat{R}_2$ and sets the exponential rank loss to be the average of the correct and reverse rank values: $\frac{1}{2}\left(e^{-\eta_s} + e^{\eta_s}\right) = \cosh(\eta_s)$. It is straightforward to see that using this average is necessary for F_2, and by extension $\hat{E}_2$, to induce the same order on the weak rankers as $\hat{R}_2$. Suppose some weak ranker makes a reverse rankings and b ties. Another ranker with $a - c$ reverse rankings and $b + 2c$ ties, for any c, will have the same $\hat{R}_2$ error. However, any loss function that does not assign to ties the average of the misranking and correct ranking values will give a different loss to these two rankers. In Proposition 3 below, we prove that using the average is also sufficient.

Using the same form as (21), we have

$$F_2(\boldsymbol{\eta}) = \frac{1}{m} \sum_{i=1}^{m} e^{\sum_{s=1}^{N} \ln \omega_2^*(i, f_s, \eta_s)} \tag{22}$$

where function ω_2^* is defined as

$$\omega_2^*(i, f_s, \eta_s) = \begin{cases} e^{-\eta_s} & \text{if pair } i \text{ correctly ranked by } f_s, \\ e^{\eta_s} & \text{if reverse ranked by } f_s, \text{ and} \\ \cosh(\eta_s) & \text{if tied by } f_s. \end{cases} \tag{23}$$

Note that just like F_1, F_2 is a convex function. Therefore, once we define RankBoost+, we can use convexity and Eq. (29) below, the analogous equation to (12) for Rb-d, to prove that RankBoost+ has theoretical properties analogous to Observation 1 and Theorem 1.

We can now define $\hat{E}_2$ as a function on the ensemble ranker produced at each iteration of Algorithm 1. Given ensemble $g = \sum_{s=1}^{t} \alpha_s h_s$, let $\eta' = [\eta'_1, \ldots, \eta'_N]$ be the element of $\mathbb{R}^N$ where η'_i is the sum of α_j for which $f_i = h_j$. Then,

$$\hat{E}_2\left(\sum_{s=1}^{t} \alpha_s h_s\right) = F_2\left(\boldsymbol{\eta}'\right). \tag{24}$$

For completeness, the next proposition proves that $\hat{E}_2$ is an upper bound for $\hat{R}_2$. The following proposition proves that $\hat{E}_2$ has the nice property that it induces the same quasiordering on the weak rankers as $\hat{R}_2$.

Proposition 2 *Let $g = \sum_{s=1}^{N} \eta_s f_s$. Then $\hat{E}_2(g) \geq \hat{R}_2(g)$.*

Proposition 3 *For all weak rankers h_1 and h_2, if $\hat{R}_2(h_1) < \hat{R}_2(h_2)$, then $\hat{E}_2(h_1) < \hat{E}_2(h_2)$.*

Proposition 3 is also only concerned with the ordering of weak rankers. Minimizing $\hat{R}_2$ is NP-complete (Cohen et al. 1999), and we are therefore unlikely to have an exponential loss function that induces the same ordering as $\hat{R}_2$ on all possible rankers.

3.2 The RankBoost+ algorithm

RankBoost+ has the same framework as RankBoost (Algorithm 1). To describe it, we reuse the notations Z_t, α_t, and D_t but change their definitions.

Just as with RankBoost, RankBoost+ maintains a distribution D_t on the critical pairs, and after choosing a weak ranker h_t and weight α_t to add to the ensemble ranker, RankBoost+ updates the distribution weight for pair i by multiplying the weight by a scaling function that we denote ω_t^+.

Recall that the scaling function for Rb-d, Eq. (9), maps pair i to its contribution to $\hat{E}_1(\alpha_t h_t)$. In defining the scaling function ω_t^+ for RankBoost+, we note that when running Algorithm 1 the same weak ranker may be chosen in multiple iterations. Consider the ensemble ranker created after iteration $t-1$: $g_{t-1} = \sum_{s=1}^{t-1} \alpha_s h_s$ and the ensemble ranker created after iteration t: $g_t = \sum_{s=1}^{t} \alpha_s h_s$. Let $\eta' \in \mathbb{R}^N$ be the vector such that $F_2(\eta') = \hat{E}_2(g_{t-1})$, and let $\eta \in \mathbb{R}^N$ be the vector such that $F_2(\eta) = \hat{E}_2(g_t)$. Using this notation, η'_j is the cumulative weight of ranker f_j in the ensemble at the start of iteration t and η_j is the cumulative weight after iteration t. Suppose that weak ranker f_j is chosen at iteration t: $h_t = f_j$, and suppose that f_j was chosen at earlier iterations. The contribution of f_j to F_2, and thus to $\hat{E}_2$, for pair i is $\omega_2^*(i, f_j, \eta'_j)$ before iteration t and $\omega_2^*(i, f_j, \eta_j)$ after. We define ω_t^+, the scaling function for RankBoost+ in iteration t, such that $\omega_t^+(i) \cdot \omega_2^*(i, f_j, \eta'_j) = \omega_2^*(i, f_j, \eta_j)$. To be consistent with our use of η for a vector of $\mathbb{R}^N$ and α for the weights of the ensemble, we define α'_t to be the total weight assigned to f_j in the ensemble at the start of iteration t. Thus, $\alpha'_t = \eta'_j$, and

$$\omega_t^+(i) = \begin{cases} e^{-\alpha_t} & \text{if pair } i \text{ correctly ranked by } h_t, \\ e^{\alpha_t} & \text{if reverse ranked by } h_t, \text{ and} \\ \frac{\cosh(\alpha_t + \alpha'_t)}{\cosh(\alpha'_t)} & \text{if tied by } h_t. \end{cases} \tag{25}$$

Defining ω_t^+ in this manner gives us Eq. (29) below that is analogous to Eq. (12). Recall that (12) is fundamental to Rb-d. In fact, (12) is used to prove Theorem 1, and (12) and the convexity of $\hat{E}_1$ is needed for Observation 1.

Line 9 of RankBoost+ is very similar to line 9 of RankBoost:

$$D_{t+1}(i) = \frac{D_t(i)\omega_t^+(i)}{Z_t} \tag{26}$$

with

$$Z_t = \varepsilon_t^+ e^{-\alpha_t} + \varepsilon_t^- e^{\alpha_t} + \varepsilon_t^0 \frac{\cosh(\alpha_t + \alpha'_t)}{\cosh \alpha'_t}. \tag{27}$$

Just as with Rb-d, RankBoost+ chooses the weak ranker h_t and weight α_t that minimizes $\hat{E}_2$. At iteration t we have $g_{t-1} = \sum_{s=1}^{t-1} \alpha_s h_s$. We calculate the derivative of $g_{t-1} + \alpha_t h_t$ with respect to α_t.

Using induction on (26) gives

$$\prod_{s=1}^{t} Z_s = D_1(i) \prod_{s=1}^{t} \omega_s^+(i), \tag{28}$$

Combining (28) with (24), the definition of $\hat{E}_2$, gives

$$\hat{E}_2\left(\sum_{s=1}^{t} \alpha_s h_s\right) = \prod_{s=1}^{t} Z_s. \tag{29}$$

Therefore, we have the derivative

$$\begin{aligned} \frac{d\hat{E}_2(g_{t-1} + \alpha_t h_t)}{d\alpha_t} &= \frac{dZ_t}{d\alpha_t} \prod_{s=1}^{t-1} Z_s \\ &= \left(-\varepsilon_t^+ e^{-\alpha_t} + \varepsilon_t^- e^{\alpha_t} + \varepsilon_t^0 \frac{\sinh(\alpha_t + \alpha_t')}{\cosh \alpha_t'}\right) \prod_{s=1}^{t-1} Z_s \end{aligned} \tag{30}$$

where $\sinh(\alpha) = \frac{1}{2}\left(e^{\alpha} - e^{-\alpha}\right)$.

At iteration t, we want to add the weak ranker h_t that will cause the greatest decrease in $\hat{E}_2$. Consider the directional derivative

$$\left.\frac{d\hat{E}_2(g_{t-1} + \alpha_t h_t)}{d\alpha_t}\right|_{\alpha_t = 0} = \left[-\varepsilon_t^+ + \varepsilon_t^- + \varepsilon_t^0 \frac{\sinh \alpha_t'}{\cosh \alpha_t'}\right] \prod_{s=1}^{t-1} Z_s. \tag{31}$$

As we are not restricting α_t to be positive, the h_t that achieves the greatest decrease in $\hat{E}_2$ corresponds to the direction in (31) with the greatest slope. Therefore, on line 6 of Algorithm 1 RankBoost+ chooses h_t such that

$$h_t = \arg\max_{h \in \mathscr{H}'} |\delta(h)|, \tag{32}$$

where $\mathscr{H}'$ is the set of binary weak rankers and

$$\delta(h_t) = \varepsilon_t^- - \varepsilon_t^+ + \varepsilon_t^0 \frac{\sinh \alpha_t'}{\cosh \alpha_t'}. \tag{33}$$

Setting $\frac{d\hat{E}_2(g_{t-1}+\alpha_t h_t)}{d\alpha_t} = 0$ and solving for α_t gives line 7 of RankBoost+:

$$\alpha_t = \frac{1}{2} \log \frac{\varepsilon_t^+ + \varepsilon_t^0 \frac{\exp(-\alpha_t')}{2\cosh \alpha_t'}}{\varepsilon_t^- + \varepsilon_t^0 \frac{\exp \alpha_t'}{2\cosh \alpha_t'}}. \tag{34}$$

Now that we have defined the weights α_t used by RankBoost+, we give a corollary to Proposition 3 to show that, in contrast to Rb-d and Rb-c (see Proposition 1 above), the weighted weak rankers used by RankBoost+ in the first iteration of the algorithm have the same quasiordering with respect to both $\hat{R}_2$ and $\hat{E}_2$.

Corollary 1 *For all weak rankers h_1 and h_2, if $\hat{R}_2(h_1) < \hat{R}_2(h_2) < \frac{1}{2}$, then $\hat{E}_2(\alpha_{h_1} h_1) < \hat{E}_2(\alpha_{h_2} h_2)$ where α_{h_1} and α_{h_2} are the weights assigned by* RankBoost+ *given distribution D_1.*

In Corollary 1, it is sufficient to consider weak rankers h with $\hat{R}_2(h) < \frac{1}{2}$. Suppose we have ranker h with $\hat{R}_2(h) > \frac{1}{2}$ and the "inverse ranker" h' that reverses every ranking choice in h. Let D be any distribution on the critical pairs. Then $\hat{R}_2(h') = 1 - \hat{R}_2(h)$ and $\hat{E}_2(\alpha' h') = \hat{E}_2(\alpha h)$ where α' and α are the weights selected by RankBoost+ for h' and h, respectively, given distribution D [see Eq. (34)].

We close this section by noting some subtle issues that arise from using $\cosh(\alpha_t)$ for the exponential loss of tied pairs. One change from RankBoost, as indicated by the scaling function (25), is that RankBoost+ will need to keep track of the accumulated weight assigned to each unique weak ranker. For a second difference from RankBoost, let $\mathscr{H}'$ be a set of binary weak rankers and let $\mathscr{G}$ be a proper subset of $\mathscr{H}'$. $\mathscr{G}$ induces a different convex function F_2 than $\mathscr{H}'$ does. On the other hand, the convex function F_1 is only different when $\mathrm{span}(\mathscr{G}) \neq \mathrm{span}(\mathscr{H}')$. For example, consider the simple scenario where we have three weak rankers f_1, f_2, and f_3 where $f_2 = f_3$. As noted above, F_2 is a convex function and Theorem 3 below proves that RankBoost+ converges to the unique minimum of this convex function. While the rankers $\eta_1 f_1 + \eta_2 f_2 + \eta_3 f_3$ and $\eta_1 f_1 + (\eta_2 + \eta_3) f_2$ are identical, the convex function defined by F_2 on f_1, f_2, and f_3 is different than that defined on f_1 and f_2. In particular, the former is

$$a_1 e^{-\eta_1-\eta_2-\eta_3} + a_2 e^{-\eta_1-\eta_2+\eta_3} + a_3 e^{-\eta_1+\eta_2-\eta_3} + a_4 e^{-\eta_1+\eta_2+\eta_3}$$
$$+ a_5 e^{\eta_1-\eta_2-\eta_3} + a_6 e^{\eta_1-\eta_2+\eta_3} + a_7 e^{\eta_1+\eta_2+\eta_3} + a_8 e^{\eta_1+\eta_2-\eta_3}$$

where $a_1, \ldots, a_8$ are constants defined by the behavior of f_1, f_2, and f_3. If we instead, restrict the weak rankers to be just f_1 and f_2, and if we assign the weight $\eta_2 + \eta_3$ to weak ranker f_2, then F_2 defines a different convex function

$$(a_1 + a_2) e^{-\eta_1-\eta_2-\eta_3} + (a_3 + a_4) e^{-\eta_1+\eta_2+\eta_3} + (a_5 + a_6) e^{\eta_1-\eta_2-\eta_3} + (a_7 + a_8) e^{\eta_1+\eta_2+\eta_3}.$$

This leads to the question: what is the effect on F_2 of having a linearly dependent set of weak rankers? Recall that the only difference between F_1 and F_2 is the treatment of ties. Adding a weak ranker that is a linear combination of other weak rankers has the effect of replacing a $\cosh(x)$ term in F_2 with a $\cosh(x_1)\cosh(x_2)$ term where $x = x_1 + x_2$. The minimum of $\cosh(x_1)\cosh(x_2)$ occurs where $x_1 = x_2 = \frac{x}{2}$, and $\cosh^2(x/2) < \cosh(x)$. As a result, adding a linearly dependent weak ranker has the effect of decreasing the weight of ties. As $\cosh(x) \geq 1$, $F_2(\boldsymbol{\eta}) \geq F_1(\boldsymbol{\eta})$ for all $\boldsymbol{\eta}$, and as $\lim_{k\to\infty} \cosh^k(x/k) = 1$, we see that as we add more linearly dependent weak rankers to our set, the vector that minimizes F_2 will converge to the vector that minimizes F_1.

This analysis suggests that, in the ranking scenarios where $\hat{R}_2$ is a better loss function for ranking than $\hat{R}_1$, we should get optimal behavior if we define F_2 on a linearly independent set of weak rankers. Doing so will achieve two benefits. It will maximize the difference between F_1 and F_2, and thus the difference between the behavior of RankBoost and RankBoost+. Also, it will have F_2 most closely match the property of $\hat{R}_2$ where the weight of a tied pair is equal to the average of the weights of the correctly ranked and incorrectly ranked pairs.

3.3 Theoretical properties of RankBoost+

The following two theorems show that RankBoost+ has similar theoretical guarantees as those described above for Rb-d: its ranking loss decreases exponentially in the number of boosting rounds, and it converges to the global minimum of $\hat{E}_2$ on the space of all linear combinations of weak rankers. Please see Sect. A of the appendix for the proofs.

Theorem 2 *The empirical error of the hypothesis g returned by* RANKBOOST+ *satisfies*

$$\hat{R}_2(h) \leq \exp\left[-2\sum_{t=1}^{T}\left(\frac{\delta(h_t)}{2}\right)^2\right] = \exp\left[-2\sum_{t=1}^{T}\left(\frac{\varepsilon_t^+ - \varepsilon_t^- - \varepsilon_t^0 \frac{\sinh(\alpha_t')}{\cosh(\alpha_t')}}{2}\right)^2\right]. \quad (35)$$

Furthermore, if there exists γ *such that for all* $t \in [1, T]$, $0 < \gamma \leq \frac{\delta(h_t)}{2}$, *then*

$$\hat{R}_2(g) \leq \exp(-2\gamma^2 T) \quad (36)$$

Theorem 3 *Let* $\mathscr{H}'$ *be a set of binary weak rankers.* RANKBOOST+ *is a coordinate descent algorithm that converges to the global minimum of* F_2 *on the vector space spanned by* $\mathscr{H}'$.

3.4 Why does RB-C outperform RB-D in practice?

Our analysis of RANKBOOST+ sheds light on why RB-C performs better than RB-D in the binary weak ranker scenario, and why we expect RANKBOOST+ to perform better than RB-C.

Recall that RB-C minimizes (18), and the α_t value is chosen with Eq. (19). Let

$$\begin{aligned} r_t^+ &= \textstyle\sum_{i\in\{j|y_j(h_t(x_j')-h_t(x_j))>0\}} D_t(i) y_i (h_t(x_i') - h_t(x_i)), \\ r_t^- &= -\textstyle\sum_{i\in\{j|y_j(h_t(x_j')-h_t(x_j))<0\}} D_t(i) y_i (h_t(x_i') - h_t(x_i)), \text{ and} \\ r_t^0 &= \textstyle\sum_{i\in\{j|y_j(h_t(x_j')-h_t(x_j))=0\}} D_t(i). \end{aligned} \quad (37)$$

If we restrict ourselves to the binary weak ranker scenario of RB-D, $h_t(x_j') - h_t(x_j) \in \{-1, 0, 1\}$, then $r_t^+ + r_t^- + r_t^0 = 1$. Plugging this equation into (18) shows that the value RB-C minimizes is

$$\left(\frac{1-r_t}{2}\right)e^{\alpha_t} + \left(\frac{1+r_t}{2}\right)e^{-\alpha_t} = r_t^+ e^{-\alpha_t} + r_t^- e^{\alpha_t} + r_t^0 \cosh(\alpha_t), \quad (38)$$

and the α_t value of (19) for RB-C is

$$\frac{1}{2}\log\frac{1+r_t}{1-r_t} = \frac{1}{2}\log\frac{r_t^+ + \frac{1}{2}r_t^0}{r_t^- + \frac{1}{2}r_t^0}. \quad (39)$$

Note that the right hand side of (38) is equivalent to (27) with $\alpha_t' = 0$, and (19) for determining RB-C's α_t value is equivalent to (34) with $\alpha_t' = 0$. Furthermore, Eq. (13) used by RB-C to choose ranker h_t is equivalent to (32) with $\alpha_t' = 0$. Thus in the binary weak ranker scenario, RB-C is finding the weak ranker and weight that minimizes $\hat{E}_2$ instead of $\hat{E}_1$. Under our hypothesis that minimizing $\hat{E}_2$ is a better metric for ranking than minimizing $\hat{E}_1$ we expect RB-C to perform better than RB-D.

However, RB-C does not achieve the minimum for $\hat{E}_2$ because of two reasons. First, it uses distribution update rule (10) rather than (26) so it is not scaling the pairs by the error contribution to $\hat{E}_2$. Second, on reusing a ranker from the ensemble ranker or when using a ranker that is a linear combination of rankers already used in the ensemble, RB-C is calculating an α value that is larger in magnitude than the corresponding directional derivative of E_2. As a result, in this situation RB-C may not find the weak ranker and weight that minimizes $\hat{E}_2$, and may even choose a ranker and weight pair that increases $\hat{E}_2$.

As noted above, to have the maximal separation between the minimizers of F_2 and F_1, and thus $\hat{E}_2$ and $\hat{E}_1$, we need to define F_2 on a maximal linearly independent set of weak rankers.

In the limit, adding linearly dependent rankers to the ensemble will cause RankBoost+ and Rb-d to converge to the same ensemble ranker. This behavior is unlikely to be observed in practice because it requires a very large number of linearly dependent rankers and rounds of the algorithm. In practice, we expect the behavior of RankBoost+ to converge to the behavior of Rb-c as we include more linearly dependent weak rankers in the ensemble. Anytime RankBoost+ chooses to add a new weak ranker to the ensemble that is linearly dependent to rankers already in the ensemble, it will have $\alpha'_t = 0$ in that round. RankBoost+ will effectively be using Rb-c's (19) to assign the weights to the rankers. Even with the extreme case of $\alpha'_t = 0$ in all rounds of the algorithm, the behavior of RankBoost+ should not be exactly the same as Rb-c because the two algorithms use different distribution update rules.

To summarize, we develop our approach using a different rank loss function, $\hat{R}_2$. We derive an exponential loss function $\hat{E}_2$ that produces consistent orderings on weak rankers as $\hat{R}_2$. We then develop the RankBoost+ algorithm that minimizes $\hat{E}_2$, and we prove that RankBoost+ has good theoretical properties: it exponentially decreases $\hat{R}_2$ and converges to the global minimum of $\hat{E}_2$.

4 Implementing RankBoost+

In this section we describe some differences in how RankBoost+ is implemented relative to Rb-d and Rb-c. While all the algorithms share the same template (Algorithm 1), to achieve the best results with RankBoost+ requires that the set of rankers chosen should be linearly independent, as discussed in the end of Sect. 3.2.

We start by considering each h_j to be a vector

$$h_j = \begin{pmatrix} y_0(h_j(x'_0) - h_j(x_0)) \\ y_1(h_j(x'_1) - h_j(x_1)) \\ \vdots \\ y_m(h_j(x'_m) - h_j(x_m)) \end{pmatrix}$$

At iteration t, we will have a set of linearly independent weak rankers S_t already chosen by the model, as well as a set R of candidate rankers, which represent all possible directions for the next step of boosting. We can imagine S_t to be a matrix, with each column being some h_j, allowing us to give weights η_t to each member of S_t, so the overall model prediction on training data at iteration t is $S_t \eta_t$. For each candidate ranker h at iteration $t+1$, we check if it is in the span of S. If it is not, we compute the score defined in Eq. (32) with $\alpha'_s = 0$. If it is, then for some vector β, $S_t \beta = h$. Then, rather than minimizing $\hat{E}_2(g_t + \alpha_{t+1} h_{t+1})$ over α_{t+1}, we must minimize $\hat{E}_2(g_t + \alpha_{t+1} \sum_{i=1}^{|(S_t)_i|} (S_t)_i \beta_i)$. If we consider $\hat{E}_2$ to be a function of the weights and rankers separately, we can write this as $F_2(\eta_t + \alpha_{t+1}\beta)$, where the underlying set of rankers is S_t. To select the best ranker [Eq. (32)] for dependent h, we must take the derivative of F_2 at $\alpha_t = 0$ in the direction specified by β: $\left|\nabla_\beta F_2(\eta_t + \alpha_{t+1}\beta)\right| = \left|\nabla F_2(\eta_t) \cdot \frac{\beta}{\|\beta\|_2}\right|$. We can compute $\nabla \hat{E}_2$ as specified in Eq. (30) without significant computational cost, but the cost of determining which rankers are dependent and then computing the β vector has a complexity of $O(m|S_t|^2)$, as this step requires a least squares solve every iteration.

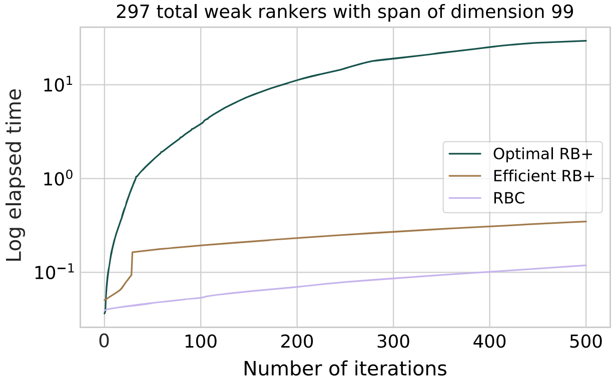

Fig. 1 Comparison of speed for implementations of RankBoost+ on synthetic data. We call RankBoost+ using all possible directions "Optimal Rb+" and Algorithm 2 we call "Efficient Rb+"

Because of this, we have an alternative, faster version that does not as significantly affect the convergence speed or generalization, which is described in Algorithm 2. This implementation still maintains the independent set S_t, but does not consider descent over rankers linearly dependent to the set of already used rankers. To do this, we compute the gradients over each direction in S_t, the set of rankers already selected (line 8), and then separately compute gradients over every direction in R, the set of all candidate rankers, while making the assumption that no ranker in R is in the span of S_t (line 10). We select the candidate that has the highest gradient, and if it is not in the span of S_t, we check to see if it is linearly independent, and add it to S if it is. The ranker can then be removed from R (line 13), as it is now known to be a linear combination of some rankers in S_t. This removal can be done by setting the column to be zero in R, meaning it cannot be reselected. As the size of S grows, we still have the issue of the cost of a least squares solve. However, since the highest weights are assigned to the first several rankers, so, after several iterations, we prune R by selecting an independent set of rankers from it. In this operation, we ensure that the chosen subset contains S_t, as we have determined that the rankers in S_t can account for the greatest decrease in loss. This also has a one-time cost of $O(m|R|^2)$, which will be more efficient than potentially requiring an $O(m|S_t|^2)$ computation every iteration. The iteration when we prune R is once the first dependent ranker is chosen by the algorithm. On larger datasets, this step could be done after a set number of iterations to limit the number of least squares solves performed. This can be seen in line 14 of Algorithm 2. The function select-independent-subset chooses a maximal linearly independent subset $T \subseteq R$ which has no elements that lie in the span of S, and has $\text{span}(T) \subseteq \text{span}(R) - \text{span}(S)$. We implement this by examining the QR decomposition (Q', R') of the concatenation $[S, \text{permute}(R)]$, and discarding any element from the set where $\text{diag}(R') = 0$. Here $\text{permute}(R)$ randomly permutes the columns of R. This operation will produce a set that is maximal with high probability.

The main added costs of this method over Rb-d or Rb-c are choosing a linearly independent subset of $R \cup S$, and checking if a ranker is in the span of S_t. The latter will only be needed for the first few iterations, until S is fixed, so there is not significant computational cost. In Fig. 1, we show a comparison of the speeds of the two different implementations of RankBoost+ on synthetic data. At around 30 iterations, the more efficient implementation has one iteration that takes longer, which is the step where a linearly independent subset is chosen, but then it runs at the approximately the same speed as Rb-d or Rb-c.

Algorithm 2 Pseudocode for efficient RANKBOOST+ implementation

function RankboostPlus(R)
  $R \in \{-1, 0, 1\}^{m \times n}$ is the set of all candidate rankers (in a matrix), with no duplicate columns
  $D_0 \leftarrow \left(\frac{1}{m}, \ldots, \frac{1}{m}\right)^T \in \mathbb{R}^m$
  Set η_0 to be an empty vector, and S_0 to be an empty matrix
  greedy-regime $\leftarrow$ True {Tracks whether or not we are considering directions outside those in S}
  for $t = 1$ **to** T **do**
    Create matrix M_t with $(M_t)_{ij} = \mathbb{I}[(S_t)_{ij} = 0]$
    gradient-over-S $\leftarrow \left|-D_t^T S_t + D_t^T M_t \circ \tanh(\eta_t)\right|$ {$\circ$ denotes elementwise multiplication}
    if greedy-regime **then**
      gradient-over-R $\leftarrow \left|-D_t^T R\right|$
      if max(gradient-over-R) $>$ max(gradient-over-S) **then**
        $\alpha_t' \leftarrow 0$ {Adding a new ranker}
        Set h_t to the best ranker in R, and set the corresponding column of R to be zero.
        if $h_t \in \text{span}(S_t)$ **then**
          $S_t \leftarrow [S_t, \text{select-independent-subset}(R, S_t)]$ {Making S static}
          Append zeros to η_t
          Create matrix M_t with $(M_t)_{ij} = \mathbb{I}[(S_t)_{ij} = 0]$
          gradient-over-S $\leftarrow \left|-D_t^T S_t + D_t^T M_t \circ \tanh(\eta_t)\right|$
          greedy-regime $\leftarrow$ False
        end if
      end if
    end if
    if $\neg$greedy-regime **or** max(gradient-over-R) $\leq$ max(gradient-over-S) **then**
      $i \leftarrow \arg\max$(gradient-over-S)
      $h_t \leftarrow (S_t)_i, \alpha_t' \leftarrow (\eta_t)_i$
    end if
    Compute α_t [Eq. (34)]
    if $h_t \notin S_t$ **then**
      $S_{t+1} \leftarrow [S_t, h_t], \eta_{t+1} \leftarrow [\eta_t, \alpha_t]$
    else
      $S_{t+1} \leftarrow S_t, \eta_{t+1} \leftarrow \eta_t$
      $(\eta_{t+1})_i \leftarrow (\eta_{t+1})_i + \alpha_t$
    end if
    Update D according to Eq. (26)
  end for
  Define $g(x) = \sum_{k=1}^{|S_{T+1}|} (\eta_{T+1})_k (S_{T+1})_k(x)$
  return g
end function

5 Empirical evaluation

In this section, we empirically study the behavior of RB-D, RB-C and RANKBOOST+ on real ranking problems. While our theoretical analysis indicates that RANKBOOST+ should be the most "well-behaved" of the three, this analysis is on training samples, so we still need to verify that it is reflected in the generalization performance. Our empirical hypothesis is that RANKBOOST+ will significantly outperform RB-D on real data. We hypothesize that it will also outperform RB-C but possibly with a smaller margin.[1]

[1] Our implementations of RANKBOOST+, RB-C, and RB-D can be found at https://github.com/rail-cwru/rankboostplus.

The ranking tasks we use are as follows.[2] MovieLens (Harper and Konstan 2016; Weimer et al. 2008; Guiver and Snelson 2009) is a movie recommendation task where users rate movies using a scale of 1 (lowest) to 5 (highest). It contains 943 users, 1682 movies and 100,000 ratings. Each user induces a separate ranking task using the movie ratings from other users as features. To get meaningful results, we only consider users who have rated at least 100 movies, from which we run experiments on 367 ranking tasks. In each such task, we select features *a priori* by removing other users (features) that have over 50% missing ratings on the movies in the task under consideration. The rationale is that such features are likely to have low information with respect to the target, and unlikely to be selected by the base learner.

LETOR (Qin and Liu 2013; Qin et al. 2010; Xia et al. 2008; Valizadegan et al. 2009) is a benchmark collection for research on learning to rank, released by Microsoft Research Asia. We use the version of MSLR-WEB10K with 10,000 queries. In this dataset, queries and URLs are represented by IDs, and 136-dimensional continuous feature vectors are extracted from query-URL pairs along with relevance judgment labels. The relevance judgments are obtained from a retired labeling set of a commercial web search engine (Microsoft Bing), which take 5 values from 0 (irrelevant) to 4 (perfectly relevant). For our experiments, we use 300 queries as separate datasets. These are selected by taking the 300 queries that would produce the most critical pairs, but have less than 50,000 critical pairs in total.

For the third set of tasks, we use the datasets MQ2007 (about 10,000 critical pairs) and MQ2008 (about 37,500 critical pairs) from LETOR 4.0 (Qin and Liu 2013). These datasets come from the Million Query track of TREC 2007 and TREC 2008. Each has 46 continuous features, as well as a relevance judgment, which is used to extract critical pairs.

The base ranker we use for all tested algorithms is a decision stump, which is a common choice (Freund et al. 2003). For each ranker, the leaf node labels are chosen using weighted majority vote. Missing feature values are given a value lower than all known values of the feature in their column. In the case that there are more than 255 candidate thresholds for a single feature, we randomly subsample them to determine a set of 255, as suggested by Qin et al. (2010).

A final detail is in the computation of $\hat{E}_2$ over training data for RB-D and RB-C. To measure $\hat{E}_2$, we track $F_2(\eta)$ over a linearly independent set of weak rankers accumulated in each iteration. A new ranker introduced in a succeeding iteration may either be in the span of the accumulated set, or not. If it is independent, we add it to the accumulated set, and update the η vector accordingly. If it lies in the span of the existing rankers, we find the update to η such that the resulting prediction by the independent ensemble is equal to the prediction made by the original, linearly dependent ensemble at that iteration. F_2 is calculated with these η values.

5.1 Results and discussion

We perform 5-fold cross validation on each of the 667 ranking tasks from MovieLens and LETOR (MSLR-WEB10K). For each task, we compute test ranking errors $\hat{R}_1$ and $\hat{R}_2$ (averaged over 5 folds) for RANKBOOST+, RB-D and RB-C, as well as Normalized Discounted Cumulative Gain (NDCG) (Järvelin and Kekäläinen 2002) at positions 3, 5, and 7. For each fold of a task, we take three folds for training, one fold for validation, and one for testing. Then, when reporting results, for each metric, we choose the iteration at which that metric

[2] Unfortunately, the datasets used in the paper introducing RANKBOOST Freund et al. (2003) are no longer available.

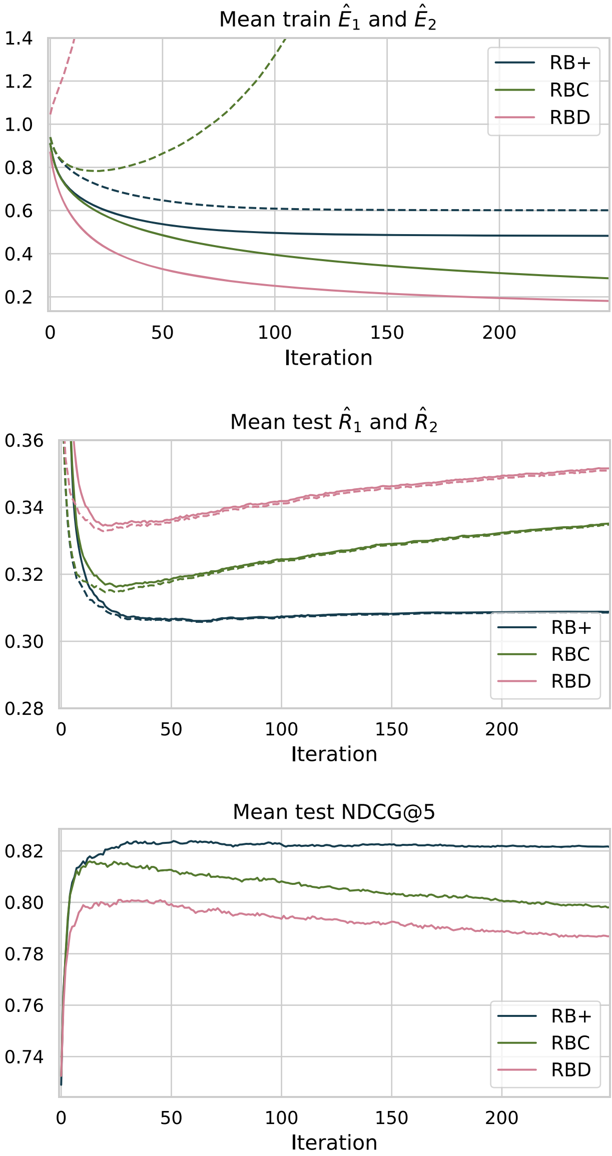

Fig. 2 Convergence rates on MovieLens (top: test $\hat{R}_1$ and $\hat{R}_2$, middle: train $\hat{E}_1$ and $\hat{E}_2$, bottom: Test NDCG@5) $\hat{R}_1$ and $\hat{E}_1$ are denoted by solid lines, and $\hat{R}_2$ and $\hat{E}_2$ are dashed lines

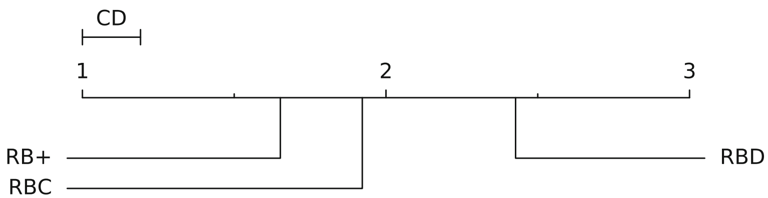

Fig. 3 Critical difference diagram for test $\hat{R}_2$ on WEB10K

Table 1 Ranks of algorithms on real-world domains w.r.t $\hat{R}_1$, $\hat{R}_2$ and NDCG

Dataset	Metric	Algorithm			CD
		RANKBOOST+	RB-C	RB-D	
MovieLens	$\hat{R}_1$	**1.314** (32)	1.938 (31)	2.748 (26)	0.1754
	$\hat{R}_2$	**1.356** (25)	1.952 (21)	2.692 (18)	
	NDCG@3	**1.696** (4)	2.125 (2)	2.179 (3)	
	NDCG@5	**1.559** (7)	2.211 (2)	2.230 (4)	
	NDCG@7	**1.476** (11)	2.268 (2)	2.256 (4)	
Web10k	$\hat{R}_1$	**1.543** (33)	1.907 (37)	2.550 (28)	0.1914
	$\hat{R}_2$	**1.652** (20)	1.922 (20)	2.427 (16)	
	NDCG@3	2.038 (6)	1.942 (7)	2.020 (10)	
	NDCG@5	1.952 (9)	1.955 (11)	2.093 (17)	
	NDCG@7	1.927 (12)	1.963 (16)	2.110 (20)	

The column CD contains the critical difference length based on the Nemenyi test. Performances that are at least one critical difference length better than the second best are boldfaced. Median stopping iteration is indicated by the number in parentheses

Table 2 Ranks of various metrics (by fold) on LETOR 4.0

Dataset	Metric	Algorithm			CD
		RANKBOOST+	RB-C	RB-D	
LETOR 4.0	$\hat{R}_1$	1.3	1.9	2.8	1.048
	$\hat{R}_2$	1.1	2.1	2.8	
	NDCG@3	1.9	1.8	2.3	
	NDCG@5	1.7	1.6	2.7	
	NDCG@7	1.7	1.5	2.8	

was minimized on the validation set, and then for each task, we average these over all 5 folds. To summarize the results across the tasks from each domain, we rank these algorithms in increasing order of test ranking errors on each task, i.e. the algorithm with smallest test error gets rank 1 and the one with largest test error gets rank 3. We then average scores from all datasets to get a final average rank for each algorithm. These are shown in Table 1. For MQ2007 and MQ2008, we use the QueryLevelNorm version with the specified folds, and rank algorithms using NDCG as suggested in Qin and Liu (2013), using the Mean Average Precision to select the optimal iteration when NDCG is measured. These results are shown in Table 2. The average performance measures for different datasets are shown in Appendix C.

From Table 1, we observe that, over the two domains, RANKBOOST+ has the smallest average rank while RB-D has the largest average rank, with RB-C in between, but generally closer to RANKBOOST+ than to RB-D. This occurs for *both* $\hat{R}_1$, optimized by RB-D, and $\hat{R}_2$, optimized by RANKBOOST+ and (imperfectly) by RB-C. A hypothesis test at a significance level of 0.05 on the results for WEB10K for $\hat{R}_2$ using a critical difference diagram (Demšar 2006) is shown in Fig. 3. We see that RANKBOOST+ significantly outperforms both RB-D and RB-C on $\hat{R}_2$, indicated by the absence of connecting bars between algorithms. The same figure is obtained for MovieLens and also if $\hat{R}_1$ is used, indicating that RANKBOOST+ outperforms the baselines under all conditions. These results align well with the theoretical analysis above and confirm our empirical hypothesis. Our results also confirm the observation that RB-C indeed performs better in practice than RB-D, as suggested by the authors of the RANKBOOST paper (Freund et al. 2003); however, it still does not outperform RANKBOOST+ because, as explained above, it only approximately optimizes $\hat{E}_2$. Table 2 shows that for MQ2007 and MQ2008, again, both RANKBOOST+ and RB-C generally outperform RB-D. In these two datasets, RANKBOOST+ is generally as good as or better than RB-C with the exception of NDCG on MQ2008. However, since we used the single specified train/test/validation partitions for these datasets, these differences are not statistically significant.

In Fig. 2, we show the rates of convergence for $\hat{R}_1$, $\hat{R}_2$, and NDCG@5 averaged over the test sets, as well as $\hat{E}_1$ and $\hat{E}_2$ averaged over the train sets on MovieLens. On the train sets, we observe that RANKBOOST+ reduces $\hat{E}_2$ quickly, as the theory suggests. RB-D in fact *increases* $\hat{E}_2$, though it does minimize $\hat{E}_1$. RB-C decreases $\hat{E}_2$ initially; however, because it does not minimize $\hat{E}_2$ exactly, eventually $\hat{E}_2$ starts increasing. As expected, on $\hat{E}_1$, RB-D and RB-C are better minimizers than RANKBOOST+. On the test sets, RANKBOOST+ produces the smallest $\hat{R}_1$ and $\hat{R}_2$, and RB-D the largest, with RB-C in between. The figures also shows that when measuring test set $\hat{R}_2$, RANKBOOST+ converges around the same speed as RB-D and RB-C, but the average error does not significantly increase after a certain point, which indicates that RANKBOOST+ overfits less severely than RB-D or RB-C. The same effect is observed in the NDCG results. RANKBOOST+ achieves a high NDCG on the test set and tends not to overfit, while both RB-D and RB-C start overfitting after a few iterations.

Finally, one might wonder why RANKBOOST+ outperforms RB-D on $\hat{R}_1$ when it is optimizing an upper bound of $\hat{R}_2$. From the figure, we observe that this is because $\hat{R}_2$ and $\hat{R}_1$ converge over boosting iterations. This happens because they differ only on ties. Recall that the datasets only have critical pairs and as an ensemble grows, we may expect the number of ties to decrease. Thus, a lower $\hat{R}_2$ also means a lower $\hat{R}_1$.

Taken together, these results show that the theoretical results for RANKBOOST+ also have an impact in practice.

6 Conclusion

In this paper, we addressed a gap in the literature for the RANKBOOST framework. Prior work proposed two variants: RB-D, which was theoretically well motivated but did not work well in practice, and RB-C, which outperformed RB-D in practice but had limited theoretical support. We have proposed RANKBOOST+, which has good theoretical support and outperforms RB-D and RB-C in practice. Further, the theory behind the approach helps explain why RB-D underperforms in practice and why RB-C is more competitive. We have also clarified some assumptions made in previous theoretical results for RANKBOOST. Finally, we demonstrate empirically that the theoretical conclusions carry over to improvements on real ranking tasks. In future work, we plan to look at the bipartite version of RANKBOOST, as well as study the generalization properties of RANKBOOST+ theoretically.

A Proofs of the theorems and propositions in the paper

Proof of Proposition 1 Let the dataset be the set of all subsets of $\{a, b, c\}$, and the true ordering on the dataset is the partial ordering defined by the subset operation. Specifically, the critical pairs are (x, y) where $x \subsetneq y$. Let h_1 and h_2 be weak rankers. h_1 maps the set $\{a, b\}$ to

1 and it maps all other sets to 0. h_2 maps the sets $\emptyset$, $\{a, c\}$, and $\{a, b, c\}$ to 1 and it maps all other sets to 0. h_1 correctly orders 3 of the critical pairs of subsets, misorders 1 of the pairs, and ties 15. h_2 correctly orders 7 of the critical pairs of subsets, misorders 5 of the pairs, and ties 7. Straightforward calculation shows $\hat{R}_1(h_1) = \frac{16}{19}$ and $\hat{R}_1(h_2) = \frac{12}{19}$. However, $\hat{E}_1(h_1) \approx 0.990627$ and $\hat{E}_1(h_2) \approx 1.21929$. Likewise, $\hat{E}_1(\alpha_{h_1} h_1) \approx .971795$ and $\hat{E}_1(\alpha_{h_2} h_2) \approx .991166$. Also, $\hat{E}_1(\alpha^c_{h_1} h_1) \approx .990034$ and $\hat{E}_1(\alpha^c_{h_2} h_2) \approx .992386$. □

Proof of Proposition 2 To show that $\hat{E}_2(g)$ is an upper bound to $\hat{R}_2(g)$, we will consider the contribution of a single pair (x'_i, x_i) to both $\hat{E}_2$ and $\hat{R}_2$. Assume w.l.o.g. that $y_i = 1$. The case that $\sum_{s=1}^N \eta_s f_s(x'_i) > \sum_{s=1}^N \eta_s f_s(x_i)$ is trivial so assume $\sum_{s=1}^N \eta_s f_s(x'_i) \le \sum_{s=1}^N \eta_s f_s(x_i)$.

$$\begin{aligned}
0 &\le \sum_{s=1}^N \eta_s f_s(x_i) - \sum_{s=1}^N \eta_s f_s(x'_i) \\
&= \sum_{s=1}^N -\eta_s(f_s(x'_i) - f_s(x_i)) \\
&= \sum_{s: f_s(x'_i) \ne f_s(x_i)} -\eta_s(f_s(x'_i) - f_s(x_i)) \\
&\le \sum_{s: f_s(x'_i) \ne f_s(x_i)} -\eta_s(f_s(x'_i) - f_s(x_i)) + \sum_{s: f_s(x'_i) = f_s(x_i)} \ln\cosh(\eta_s) \\
&= \sum_{s=1}^N \ln \omega_2^*(i, f_s, \eta_s).
\end{aligned}$$

Thus, $e^{\sum_{s=1}^N \ln \omega_2^*(i, f_s, \eta_s)} \ge 1$ for each critical pair misranked by g, and $\hat{E}_2(g) \ge \hat{R}_2(g)$. □

Proof of Proposition 3 Consider ranker h_1 with weight α_1.

$$\begin{aligned}
\hat{R}_2(\alpha_1 h_1) &= \frac{1}{m} \sum_{i=1}^m \frac{\omega_2^*(i, h_1, \alpha_1) - e^{-\alpha_1}}{e^{\alpha_1} - e^{-\alpha_1}} \\
&= \frac{\frac{1}{m}\sum_{i=1}^m \omega_2^*(i, h_1, \alpha_1) - \frac{1}{m}\sum_{i=1}^m e^{-\alpha_1}}{e^{\alpha_1} - e^{-\alpha_1}} \\
&= \frac{\hat{E}_2(\alpha_1 h_1) - e^{-\alpha_1}}{e^{\alpha_1} - e^{-\alpha_1}}.
\end{aligned} \tag{40}$$

If $\hat{R}_2(\alpha_1 h_1) < \hat{R}_2(\alpha_2 h_2)$, from (40), we have $\frac{\hat{E}_2(\alpha_1 h_1) - e^{-\alpha_1}}{e^{\alpha_1} - e^{-\alpha_1}} < \frac{\hat{E}_2(\alpha_2 h_2) - e^{-\alpha_2}}{e^{\alpha_2} - e^{-\alpha_2}}$. If we let $\alpha_1 = \alpha_2 = 1$, then $\hat{R}_2(h_1) < \hat{R}_2(h_2)$ implies $\hat{E}_2(h_1) < \hat{E}_2(h_2)$. □

Proof of Corollary 1 From Eq. (34), the weights assigned by RANKBOOST+ to h_1 distribution D_1 is $\alpha_{h_1} = \frac{1}{2} \ln\left(\frac{1 - \hat{R}_2(h_1)}{\hat{R}_2(h_1)}\right)$. If $\hat{R}_2(h_1) < \frac{1}{2}$ then $\alpha_{h_1} > 0$ and $\hat{R}_2(\alpha_{h_1} h_1) = \hat{R}_2(h_1)$. Combining these equalities with (40) gives

$$\begin{aligned}
\hat{E}_2(\alpha_{h_1} h_1) &= \hat{R}_2(\alpha_{h_1} h_1)\left(e^{\alpha_{h_1}} - e^{-\alpha_{h_1}}\right) + e^{-\alpha_{h_1}} \\
&= \hat{R}_2(h_1)\left(\sqrt{\frac{1 - \hat{R}_2(h_1)}{\hat{R}_2}} - \sqrt{\frac{\hat{R}_2(h_1)}{1 - \hat{R}_2(h_1)}}\right) + \sqrt{\frac{\hat{R}_2(h_1)}{1 - \hat{R}_2(h_1)}}
\end{aligned} \tag{41}$$

$$
\begin{aligned}
&= \frac{\hat{R}_2(h_1) - 2\hat{R}_2(h_1)^2 + \hat{R}_2(h_1)}{\sqrt{\hat{R}_2(h_1)(1-\hat{R}_2(h_1))}} \\
&= 2\sqrt{\hat{R}_2(h_1)(1-\hat{R}_2(h_1))}. \qquad (42)
\end{aligned}
$$

If $\hat{R}_2(h_1) < \hat{R}_2(h_2) < \frac{1}{2}$, then $2\sqrt{\hat{R}_2(h_1)(1-\hat{R}_2(h_1))} < 2\sqrt{\hat{R}_2(h_2)(1-\hat{R}_2(h_2))}$, and from (42), $\hat{E}_2(\alpha_{h_1} h_1) < \hat{E}_2(\alpha_{h_2} h_2)$. □

Proof of Theorem 2 First note that if there exists a round t with $\varepsilon_t^- = 0$ and $\varepsilon_t^0 = 0$, then we have a perfect ranker, that ranker will get infinite weight, and $\hat{R}_2(g) = 0$.

Otherwise, using Eq. (29) we have $\hat{R}_2(g) \leq \hat{E}_2(g) = \prod_{t=1}^T Z_t$ and

$$
\begin{aligned}
\prod_{t=1}^T Z_t &= \prod_{t=1}^T \left(\varepsilon_t^+ e^{-\alpha_t} + \varepsilon_t^- e^{\alpha_t} + \varepsilon_t^0 \frac{\cosh(\alpha_t + \alpha_t')}{\cosh \alpha_t'}\right) \\
&= \prod_{t=1}^T \frac{1}{\cosh \alpha_t'} \left(\varepsilon_t^+ \cosh(\alpha_t') e^{-\alpha_t} + \varepsilon_t^- \cosh(\alpha_t') e^{\alpha_t} + (1 - \varepsilon_t^+ - \varepsilon_t^-) \cosh(\alpha_t + \alpha_t')\right) \\
&= \prod_{t=1}^T \frac{1}{2\cosh \alpha_t'} \Big(\varepsilon_t^+ \left(e^{-\alpha_t + \alpha_t'} - e^{\alpha_t + \alpha_t'}\right) \\
&\quad + \varepsilon_t^- \left(e^{\alpha_t - \alpha_t'} - e^{-\alpha_t - \alpha_t'}\right) + e^{\alpha_t + \alpha_t'} + e^{-\alpha_t - \alpha_t'}\Big) \\
&= \prod_{t=1}^T \frac{1}{2\cosh(\alpha_t')} \left(e^{\alpha_t} \left((1 - \varepsilon_t^+) e^{\alpha_t'} + \varepsilon_t^- e^{-\alpha_t'}\right) + e^{-\alpha_t} \left(\varepsilon_t^+ e^{\alpha_t'} + (1 - \varepsilon_t^-) e^{-\alpha_t'}\right)\right).
\end{aligned}
$$

Note that

$$
\begin{aligned}
\alpha_t &= \frac{1}{2} \ln \frac{\varepsilon_t^+ + \varepsilon_t^0 \frac{\exp(-\alpha_t')}{2\cosh \alpha_t'}}{\varepsilon_t^- + \varepsilon_t^0 \frac{\exp(\alpha_t')}{2\cosh \alpha_t'}} \\
&= \frac{1}{2} \ln \frac{(e^{\alpha_t'} + e^{-\alpha_t'})\varepsilon_t^+ + e^{-\alpha_t'}(1 - \varepsilon_t^+ - \varepsilon_t^-)}{(e^{\alpha_t'} + e^{-\alpha_t'})\varepsilon_t^- + e^{\alpha_t'}(1 - \varepsilon_t^+ - \varepsilon_t^-)} \\
&= \frac{1}{2} \ln \frac{e^{\alpha_t'} \varepsilon_t^+ + e^{-\alpha_t'}(1 - \varepsilon_t^-)}{e^{\alpha_t'}(1 - \varepsilon_t^+) + e^{-\alpha_t'} \varepsilon_t^-}.
\end{aligned}
$$

Thus

$$
\begin{aligned}
\prod_{t=1}^T Z_t &= \prod_{t=1}^T \frac{1}{\cosh(\alpha_t')} \sqrt{\left((1 - \varepsilon_t^+) e^{\alpha_t'} + \varepsilon_t^- e^{-\alpha_t'}\right)\left(\varepsilon_t^+ e^{\alpha_t'} + (1 - \varepsilon_t^-) e^{-\alpha_t'}\right)} \\
&= \prod_{t=1}^T \frac{1}{\cosh(\alpha_t')} \sqrt{\left((1 - \varepsilon_t^+)(\cosh(\alpha_t') + \sinh(\alpha_t')) + \varepsilon_t^- (\cosh(\alpha_t') - \sinh(-\alpha_t'))\right)} \\
&\quad \times \sqrt{\left(\varepsilon_t^+ (\cosh(\alpha_t') + \sinh(\alpha_t')) + (1 - \varepsilon_t^-)(\cosh(\alpha_t') - \sinh(\alpha_t'))\right)} \\
&= \prod_{t=1}^T \frac{1}{\cosh(\alpha_t')} \sqrt{\left((1 - \varepsilon_t^+ + \varepsilon_t^-) \cosh(\alpha_t') + (1 - \varepsilon_t^+ - \varepsilon_t^-) \sinh(\alpha_t')\right)}
\end{aligned}
$$

$$
\begin{aligned}
&\times \sqrt{\left((1+\varepsilon_t^+ - \varepsilon_t^-)\cosh(\alpha_t') - (1-\varepsilon_t^+ - \varepsilon_t^-)\sinh(\alpha_t')\right)} \\
&= \prod_{t=1}^{T} \sqrt{\left(1-\varepsilon_t^+ + \varepsilon_t^- + \varepsilon_t^0 \frac{\sinh(\alpha_t')}{\cosh(\alpha_t')}\right)\left(1+\varepsilon_t^+ - \varepsilon_t^- - \varepsilon_t^0 \frac{\sinh(\alpha_t')}{\cosh(\alpha_t')}\right)} \\
&\le \prod_{t=1}^{T} \sqrt{\exp\left[-\left(\varepsilon_t^+ - \varepsilon_t^- - \varepsilon_t^0 \frac{\sinh(\alpha_t')}{\cosh(\alpha_t')}\right)^2\right]} \\
&= \prod_{t=1}^{T} \exp\left[-\frac{1}{2}\left(\varepsilon_t^+ - \varepsilon_t^- - \varepsilon_t^0 \frac{\sinh(\alpha_t')}{\cosh(\alpha_t')}\right)^2\right] \\
&= \exp\left[-2\sum_{t=1}^{T}\left(\frac{\varepsilon_t^+ - \varepsilon_t^- - \varepsilon_t^0 \frac{\sinh(\alpha_t')}{\cosh(\alpha_t')}}{2}\right)^2\right]
\end{aligned}
$$

□

Proof of Theorem 3 The proof uses the framework of Rudin et al. (2005) and shows convergence for F_2. Convergence of $\hat{E}_2$ trivially follows. Let $\mathscr{H}'$ be the finite set of weak rankers RankBoost+ considers during its execution. Let $N = |\mathscr{H}'|$, and let $\boldsymbol{\eta} = [\eta_1, \ldots, \eta_N]$ be an element of $\mathbb{R}^N$. From Eq. (24),

$$F_2(\boldsymbol{\eta}) = \frac{1}{m}\sum_{i=1}^{m}\prod_{s=1}^{N}\frac{1}{2}\left(e^{-\eta_s y_i r_s(x_i')} + e^{\eta_s y_i r_s(x_i)}\right) \tag{43}$$

where $r_s(x) = 2f_s(x) - 1$. We can think of RankBoost+ as starting with $\boldsymbol{\eta}_0 = [0, \ldots, 0]$, and at each iteration it chooses one of the coordinates, η_j, to adjust in order to minimize F_2. Let $\boldsymbol{\eta}_t = [\eta_1, \ldots, \eta_N]$ represent the accumulated weights for each weak ranker after iteration t of RankBoost+. Let $\mathbf{e}_j$ be the elementary unit vector of $\mathbb{R}^N$ that has a one in the jth coordinate.

$$\frac{\partial F_2(\boldsymbol{\eta}_t + \lambda \mathbf{e}_j)}{\partial \lambda} = \left(-\varepsilon_j^+ e^{-\lambda} + \varepsilon_j^- e^{\lambda} + \varepsilon_j^0 \frac{\sinh(\lambda + \eta_j)}{\cosh \eta_j}\right) F_2(\boldsymbol{\eta}_t). \tag{44}$$

As

$$\left.\frac{\partial F_2(\boldsymbol{\eta}_t + \lambda \mathbf{e}_j)}{\partial \lambda}\right|_{\lambda=0} = \left[-\varepsilon_t^+ + \varepsilon_t^- + \varepsilon_t^0 \frac{\sinh \eta_j}{\cosh \eta_j}\right] F_2(\boldsymbol{\eta}_t), \tag{45}$$

and $\frac{\partial F_2(\boldsymbol{\eta}_t + \lambda \mathbf{e}_j)}{\partial \lambda} = 0$ at

$$\lambda = \frac{1}{2}\log \frac{\varepsilon_t^+ + \varepsilon_t^0 \frac{\exp(-\eta_j)}{2\cosh \eta_j}}{\varepsilon_t^- + \varepsilon_t^0 \frac{\exp \eta_j}{2\cosh \eta_j}}, \tag{46}$$

we see that RankBoost+ at iteration $t+1$ finds the weak ranker and weight that achieves the largest decrease in F_2 at $\boldsymbol{\eta}_t$. Since F_2 is convex and thus does not have any local minima, RankBoost+ is equivalent to a coordinate descent that algorithm that converges to the global minimum of F_2. □

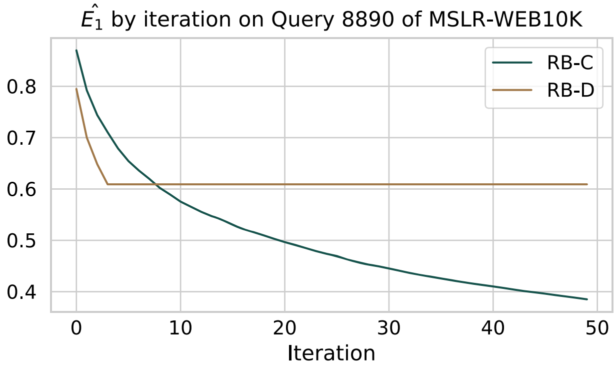

Fig. 4 An example of early stopping by RB-D on a real dataset

B Analysis of the theory for RB-D

In Sect. 2, we present two well known theoretical guarantees for RB-D: that the rank loss of the ensemble produced by RB-D decreases exponentially with each iteration of RB-D and that RB-D converges to the global minimum of $\hat{E}_1$. We give these as Observation 1 and Theorem 1. We noted that there are assumptions needed for these guarantees, and the assumptions are not always included in the literature. In this section, we prove that these assumptions are required.

First, we show that the theoretical results may not hold if (14) is not well-defined. Equation (14) is not well-defined if we have a weak ranker that makes no reverse rankings, but it is not a perfect ranker because it ties some pairs. In the proofs below, we give synthetic ranking scenarios in order to create a situation where (14) is undefined. However, this situation does occur in practice. Figure 4 demonstrates a real dataset in which there exists a ranker that makes no reverse rankings, and RB-D chooses that ranker. In Fig. 4, we halt RB-D and return the ensemble ranker that existed prior to the iteration that makes (14) undefined. This ensemble is clearly not at the minimum $\hat{E}_1$ value as the plot for RB-C demonstrates.

With (14) undefined, it is not clear what RB-D should do. Figure 4 demonstrates that halting RB-D and returning the ensemble without this ranker violates Observation 1 and Theorem 1. We now show that the other options for dealing with this situation are also problematic. One possible trick is to halt RB-D and give the ranker with $\varepsilon_t^- = 0$ infinite weight in the ensemble. This is what we do with ADABOOST. Recall that Theorem 1 states that the rank loss of the ensemble is upper bounded by $\exp\left[-2\sum_{t=1}^{T}\left(\frac{\varepsilon_t^+ - \varepsilon_t^-}{2}\right)^2\right]$.

Lemma 1 *Let g be the ensemble ranker produced by* RB-D *after T iterations of the algorithm. Let $U(g) = \exp\left[-2\sum_{t=1}^{T}\left(\frac{\varepsilon_t^+ - \varepsilon_t^-}{2}\right)^2\right]$. Suppose $\varepsilon_t^- > 0$ for $t < T$ and $\varepsilon_T^- = 0$. If we define $\left(\sum_{t=1}^{T-1}\alpha_t h_t + \infty h_T\right)(x) = (\infty h_T)(x)$, then $\hat{R}_1(g)$ can exceed $U(g)$ by $\Theta(1)$.*

Proof Let $\mathscr{H} = \{h_1, h_2\}$. The input to RB-D will be $(n+1)^2$ critical pairs drawn from $2n+2$ elements, and we set $T = 2$. Figure 5 shows the counterexample. In Fig. 5 elements $a_1 \ldots, a_{n+1}$ are represented as red circles and elements $b_1, \ldots, b_{n+1}$ as blue stars. The critical pairs are $\{(a_i, b_j), i \in \{1, \ldots, n+1\}, j \in \{1, \ldots, n+1\}\}$, and any element a_i should be ranked higher than any element b_j.

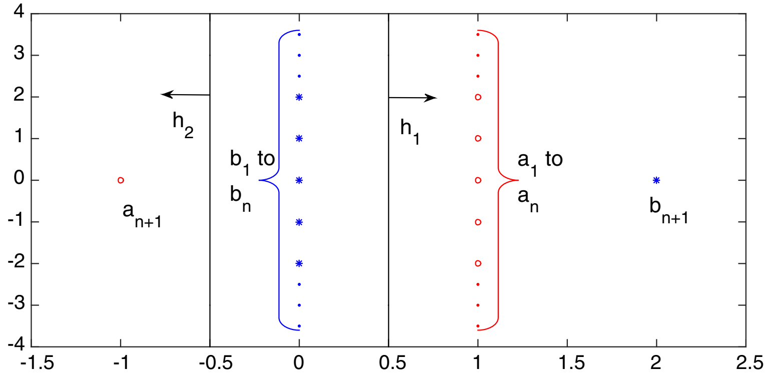

Fig. 5 Counterexample used in Lemma 1

We depict the two weak rankers h_1 and h_2 of $\mathscr{H}$ in Fig. 5. The arrows indicates the subset of elements mapped to 1 by the weak ranker. h_1 has n^2 critical pairs correctly ranked, 1 critical pair ranked in the reverse order, and $2n$ critical pairs tied. h_2 has $n+1$ pairs correctly ranked, 0 pairs reverse ranked, and n^2+n pairs tied.

On line 6 of Algorithm 1, RB-D will choose the ranker with the minimal $\varepsilon^- - \varepsilon^+$ value. For h_1, this value is $\frac{1-n^2}{n^2+2n+1}$, and for h_2 this value is $\frac{-n-1}{n^2+2n+1}$. If $n > 2$ RB-D will choose h_1. We have $\varepsilon_1^+ = \frac{n^2}{(n+1)^2}$, $\varepsilon_1^- = \frac{1}{(n+1)^2}$, and $\varepsilon_1^0 = \frac{2n}{(n+1)^2}$. As a result,

$$\alpha_1 = \frac{1}{2}\log\frac{\varepsilon_1^+}{\varepsilon_1^-} = \log n.$$

Now, the algorithm creates the new distribution D_2. Note that

$$Z_1 = \varepsilon_1^0 + 2\sqrt{\varepsilon_1^+\varepsilon_1^-} = \frac{2n}{(n+1)^2} + 2\sqrt{\frac{n^2}{(n+1)^4}} = \frac{4n}{(n+1)^2}.$$

For each pair (a_i, b_j) correctly ranked by h_1, $D_2(\cdot) = \left(\frac{1}{(n+1)^2}\frac{1}{n}\right) \div \left(\frac{4n}{(n+1)^2}\right) = \frac{1}{4n^2}$. For the pair (a_{n+1}, b_{n+1}) reverse ranked by h_1, $D_2(\cdot) = \left(\frac{1}{(n+1)^2}n\right) \div \left(\frac{4n}{(n+1)^2}\right) = \frac{1}{4}$. For each pair (a_{n+1}, b_j) and (a_i, b_{n+1}) tied by h_1, $D_2(\cdot) = \left(\frac{1}{n+1}\right) \div \left(\frac{4n}{(n+1)^2}\right) = \frac{1}{4n}$.

In the second round, RB-D will again choose the ranker with minimal $\varepsilon^- - \varepsilon^+$. For h_1, this value is 0. Since h_2 has no reverse ranked pairs, its value is negative. RB-D will choose h_2 as the minimal ranker for distribution D_2. h_2 will give $n+1$ critical pairs correct rankings, and of those pairs one was reverse ranked by h_1 and the rest were tied by h_1. h_2 reverse ranks no critical pair. Therefore, $\varepsilon_2^+ = \frac{n}{4n} + \frac{1}{4} = \frac{1}{2}$ and $\varepsilon_2^- = 0$. RB-D now halts.

$$\begin{aligned} U(h) &= \exp\left\{-\frac{1}{2}\left[\left(\frac{n^2}{(n+1)^2 - \frac{1}{(n+1)^2}}\right)^2 + \left(\frac{1}{2}\right)^2\right]\right\} \\ &= \exp\left\{-\frac{1}{2}\left[\left(\frac{n-1}{n+1}\right)^2 + \frac{1}{4}\right]\right\}, \qquad (47) \\ \hat{R}_1(h) &= \hat{R}_1(\log n\, h_1 + \infty\, h_2) \end{aligned}$$

$$= 1 - \frac{n+1}{(n+1)^2} = \frac{n}{n+1}. \tag{48}$$

As $\lim_{n\to\infty} \hat{R}_1(h) = 1$ and $\lim_{n\to\infty} U(h) = \exp(-\frac{5}{8})$, $\hat{R}_1(h)$ could be $\Theta(1)$ greater than the upper bound. In addition, plugging values for n into (47) and (48) shows that Theorem 1 can be violated on as few as 10 elements. □

A second common option for dealing with the $\log \frac{1}{0}$ values is to replace $\log \frac{1}{0}$ with a large constant in line 12 of Algorithm 1. This option will fix the specific situation of Lemma 1. However, if there are multiple weak rankers with $\varepsilon_t^- = 0$, then we could violate both Theorem 1 and Eq. (12). Recall that Eq. (12) is the key property used to prove the theoretical results of RB-D.

Lemma 2 *Let g and U be defined as in Lemma 1. Suppose that whenever we have $\varepsilon_t^- = 0$ we give h_t weight $\hat{\infty}$ in g, for some large positive constant $\hat{\infty}$. Then $\hat{R}_1(g)$ can exceed $U(g)$ by $\Theta(1)$, and $\hat{E}_1(g)$ can exceed $\prod_{t=1}^T Z_t$ by $\Theta(1)$.*

Proof Assume we have $10n$ elements and $25n^2$ critical pairs of the form (a_i, b_j) for $1 \le i, j \le 5n$. As before, the ground truth has element a_i ranked higher than each element b_j. Let $\mathcal{H} = \{h_1, h_2\}$. Let h_1 assign 1 to $a_1, \ldots, a_{4n}$ and 0 to all other elements. Let h_2 assign 1 to $a_{4n+1}, \ldots, a_{5n}$ and $b_1, \ldots, b_{4n}$ and 0 to all other elements. h_1 has $20n^2$ correctly ranked pairs, no reverse ranked pairs, and $5n^2$ tied pairs. h_2 has n^2 correctly ranked pairs, $16n^2$ reverse ranked pairs, and $8n^2$ tied pairs.

We will set $T = 2$. RB-D will choose h_1 as the weak ranker in the first iteration, and since α_1 is undefined, we assume RB-D uses $\hat{\infty}$ as the weight for h_1 in g. We have $\hat{R}_1(\hat{\infty}h_1) = \hat{E}_1(\hat{\infty}h_1) = Z_1 = \frac{1}{5}$.

In the second round, the $20n^2$ critical pairs correctly ranked by h_1, which includes all $16n^2$ pairs reverse ranked by h_2, have probability 0. The remaining critical pairs each has probability $\frac{1}{5n^2}$. RB-D will choose h_2 as the weak ranker in second iteration, and α_2 will be undefined. Z_2 is the proportion of critical pairs with positive probability that are tied by h_2. $Z_2 = \frac{4}{5}$.

Let $g = \hat{\infty}h_1 + \hat{\infty}h_2$. Note that $g(a_i) = \hat{\infty}$ for all $i \in \{1, \ldots, 5n\}$, $g(b_j) = \hat{\infty}$ for all $j \in \{1, \ldots, 4n\}$, and $g(b_j) = 0$ for all $j \in \{4n+1, \ldots, 5n\}$. Thus g correctly ranks $\frac{1}{5}$ of the critical pairs, and ties the remaining pairs.

$$R_1(g) = \hat{E}_1(g) = \frac{4}{5}, \text{ but}$$

$$Z_1 Z_2 = \frac{4}{25}, \text{ and}$$

$$U(g) = \exp\left[-2\left(\left(\frac{4/5}{2}\right)^2 + \left(\frac{1/5}{2}\right)^2\right)\right]$$

$$= \exp(-17/50) \approx 0.712.$$

□

A trick that will allow Theorem 1 is to change the definition of g in Theorem 1 and in line 12 of Algorithm 1 from $g = \sum_{t=1}^T \alpha_t h_t$ to

$$g = \lim_{(d_1,\ldots,d_t)\to(\varepsilon_1^-,\ldots,\varepsilon_T^-)} \sum_{t=1}^T \left(\frac{1}{2} \ln \frac{\varepsilon_t^+}{d_t}\right) h_t. \tag{49}$$

The change needed for the proof of Theorem 1 is straightforward, but the change needed for RB-D is non-trivial and, in our opinion, not elegant. However, any trick that gives h_t finite weight when $\varepsilon_t^- = 0$ violates Observation 1.

Observation 2 *If $\varepsilon_t^- = 0$, and if we give h_t a finite weight in the ensemble, then* RB-D *does not converge to the global minimum of $\hat{E}_1$.*

Proof Note that if $\varepsilon_t^- = 0$ then $\hat{E}_1$ does not contain any terms of the form e^{α_t}. Thus the minimum of $\hat{E}_1$ occurs as α_t goes to ∞. □

Finally, we demonstrate that if RB-D does not allow the α weights to be negative, then RB-D will not converge to the minimum of $\hat{E}_1$ in all ranking scenarios. We note that Mohri et al. (2012) suggests that RANKBOOST gets better generalization results if the α values are restricted to be positive, and if $\mathscr{H}$ contains at least one weak ranker that makes more correct rankings than reverse rankings, then equation (13), used to choose the next weak ranker, will not choose a ranker that requires negative α.

Lemma 3 *If* RB-D *is restricted to only allow positive α values, then there exists a set of weak rankers such that* RB-D *does not converge to the global minimum of $\hat{E}_1$ on the vector space spanned by the weak rankers.*

Proof Consider the set $\{1, \ldots, 6\}$ with the true order being $1 > \cdots > 6$. Let h_1 and h_2 be our only two weak rankers. Suppose h_1 partitions the set as $\{\{1, 2, 3, 6\}, \{4, 5\}\}$ and h_2 partitions the set as $\{\{2\}, \{1, 3, 4, 5, 6\}\}$. In both cases, the left subset elements are ranked above the right subset elements. So, $h_2(2) = 1$ and $h_2(x) = 0$ for $x \neq 2$.

In the first iteration of RB-D, RB-D will choose ranker h_1 and assign it weight $\alpha_1 = \frac{1}{2} \ln 3$. In the second iteration, RB-D will choose ranker h_2 and assign it weight $\frac{1}{2} \ln \frac{2+2\sqrt{3}}{\sqrt{3}}$. RB-D will halt after the second iteration because for both rankers we now have $\varepsilon^+ \leq \varepsilon^-$. Therefore RB-D outputs the ensemble ranker $\frac{1}{2}\left(\ln 3h_1 + \frac{2+2\sqrt{3}}{\sqrt{3}} h_2\right) \approx .54931h_1 + .57445h_2$.

$$\hat{E}_1(\alpha_1 h_1 + \alpha_2 h_2) = \frac{1}{15}\left(2e^{\alpha_1} + 4e^{-\alpha_1} + e^{\alpha_2} + 2e^{-\alpha_2} + 2e^{-\alpha_1-\alpha_2} + 4\right).$$

Thus, $\hat{E}_1\left(\frac{1}{2}\left(\ln 3h_1 + \frac{2+2\sqrt{3}}{\sqrt{3}} h_2\right)\right) \approx .888387$. However, the minimum of $\hat{E}_1(\alpha_1 h_1 + \alpha_2 h_2)$ is $.88703\ldots$ with $\alpha_1 = .46894\ldots$ and $\alpha_2 = .58953\ldots$. □

C Result tables

See Tables 3 and 4.

Table 3 Raw mean performance of algorithms over all datasets

Dataset	Metric	Algorithm		
		RANKBOOST+	RB-C	RB-D
MovieLens	$\hat{R}_1$	0.3100	0.3218	0.3394
	$\hat{R}_2$	0.3114	0.3218	0.3376
	NDCG@3	0.7920	0.7642	0.7620
	NDCG@5	0.8019	0.7684	0.7675
	NDCG@7	0.8104	0.7744	0.7749
Web10k	$\hat{R}_1$	0.3763	0.3804	0.3918
	$\hat{R}_2$	0.3762	0.3790	0.3880
	NDCG@3	0.4628	0.4667	0.4847
	NDCG@5	0.4836	0,4847	0.4881
	NDCG@7	0.4972	0.4989	0.4911

Table 4 $\hat{R}_1$, $\hat{R}_2$, and NDCG measured on LETOR 4.0

Dataset	Metric	Algorithm		
		RANKBOOST+	RB-C	RB-D
MQ2007	$\hat{R}_1$	0.2974	0.2980	0.3049
	$\hat{R}_2$	0.2985	0.2978	0.3098
	NDCG@3	0.4121	0.4104	0.4034
	NDCG@5	0.4186	0.4169	0.4126
	NDCG@7	0.4280	0.4276	0.4218
MQ2008	$\hat{R}_1$	0.2082	0.2153	0.2190
	$\hat{R}_2$	0.2027	0.2161	0.2192
	NDCG@3	0.4190	0.4250	0.4185
	NDCG@5	0.4684	0.4702	0.4582
	NDCG@7	0.4887	0.4899	0.4819

$\hat{R}_1$ and $\hat{R}_2$ are measured at the best validation iteration of the specific metric, and NDCG is measured at the iteration with lowest Mean Average Precision

References

Agarwal, A., Raghavan, H., Subbian, K., Melville, P., Lawrence, R. D., Gondek, D. C., & Fan, J. (2012). Learning to rank for robust question answering. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management* (pp. 833–842). ACM.

Agarwal, S., Dugar, D., & Sengupta, S. (2010). Ranking chemical structures for drug discovery: A new machine learning approach. *Journal of Chemical Information and Modeling*, *50*(5), 716–731.

Aslam, J. A., Kanoulas, E., Pavlu, V., Savev, S., & Yilmaz, E. (2009). Document selection methodologies for efficient and effective learning-to-rank. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 468–475). ACM.

Cao, Z., Qin, T., Liu, T. Y., Tsai, M. F., & Li, H. (2007). Learning to rank: from pairwise approach to listwise approach. In: *Proceedings of the 24th International Conference on Machine Learning* (pp. 129–136). ACM.

Cohen, W. W., Schapire, R. E., & Singer, Y. (1999). Learning to order things. *Journal of Artificial Intelligence Research*, *10*, 243–270.

Cortes, C., & Mohri, M. (2004). AUC optimization vs. error rate minimization. In *Proceedings of the 16th International Conference on Neural Information Processing Systems* (pp. 313–320). MIT Press.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, *7*(Jan), 1–30.
Freund, Y., Iyer, R., Schapire, R. E., & Singer, Y. (2003). An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, *4*(Nov), 933–969.
Freund, Y., & Schapire, R. E. (1995). A decision-theoretic generalization of on-line learning and an application to boosting. In *European Conference on Computational Learning Theory* (pp. 23–37). Springer.
Guiver, J., & Snelson, E. (2009). Bayesian inference for Plackett–Luce ranking models. In *Proceedings of the 26th Annual International Conference on Machine Learning* (pp. 377–384). ACM.
Harper, F. M., & Konstan, J. A. (2016). The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, *5*(4), 19.
Järvelin, K., & Kekäläinen, J. (2002). Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems*, *20*(4), 422–446. https://doi.org/10.1145/582415.582418.
Mohri, M., Rostamizadeh, A., & Talwalkar, A. (2012). *Foundations of Machine Learning*. Cambridge: MIT Press.
Nuray, R., & Can, F. (2006). Automatic ranking of information retrieval systems using data fusion. *Information Processing & Management*, *42*(3), 595–614.
Qin, T., & Liu, T. (2013). Introducing LETOR 4.0 datasets. CoRR. http://arxiv.org/abs/1306.2597
Qin, T., Liu, T. Y., Xu, J., & Li, H. (2010). LETOR: A benchmark collection for research on learning to rank for information retrieval. *Information Retrieval*, *13*(4), 346–374.
Rudin, C., Cortes, C., Mohri, M., & Schapire, R. E. (2005). Margin-based ranking meets boosting in the middle. In *International Conference on Computational Learning Theory* (pp. 63–78). Springer.
Valizadegan, H., Jin, R., Zhang, R., & Mao, J. (2009). Learning to rank by optimizing NDCG measure. In *Proceedings of the 22nd International Conference on Neural Information Processing Systems* (pp. 1883–1891). Curran Associates Inc.
Weimer, M., Karatzoglou, A., Le, Q. V., & Smola, A. (2007). COFIRANK maximum margin matrix factorization for collaborative ranking. In *Proceedings of the 20th International Conference on Neural Information Processing Systems* (pp. 1593–1600). Curran Associates Inc.
Xia, F., Liu, T. Y., Wang, J., Zhang, W., & Li, H. (2008). Listwise approach to learning to rank: Theory and algorithm. In *Proceedings of the 25th International Conference on Machine Learning* (pp. 1192–1199). ACM.
Zheng, Z., Zha, H., & Sun, G. (2008). Query-level learning to rank using isotonic regression. In *2008 46th Annual Allerton Conference on Communication, Control, and Computing* (pp. 1108–1115). IEEE.