Optimizing Combinatorial and Non-decomposable Metrics with ExactBoost

Daniel Csillag [*1] **Carolina Piazza** [*2] **Thiago Ramos** [*1] **João Vitor Romano** [*1]
Roberto Oliveira [1] **Paulo Orenstein** [1]

Abstract

Many classification algorithms require the use of surrogate losses when the intended loss function is combinatorial or non-decomposable. We introduce a fast and exact stagewise optimization algorithm, dubbed ExactBoost, that boosts stumps to the actual loss function. By developing a novel extension of margin theory to the non-decomposable setting, it is possible to bound the generalization error of ExactBoost for many metrics with different levels of non-decomposability.

1. Introduction

Several classification tasks involve combinatorial and non-decomposable (CND) loss functions. A combinatorial metric is computed in terms of indicator functions, while non-decomposable metrics cannot be reduced to a sum of loss functions on each sample point. Note approaches based on convex optimization or stochastic gradient descent are not readily applicable without resorting to surrogate losses.

Assume the data comes as independent and identically distributed (iid) points $(X_i, y_i)_{i=1}^n$, with features $X_i \in \mathbb{R}^p$ and binary labels $y_i \in \{0, 1\}$, and the goal is to learn score functions $S : \mathbb{R}^p \to [-1, 1]$ that minimize a certain loss function. Three important CND losses are:

$$\widehat{\mathrm{AUC}}(S, y) = 1 - \frac{1}{n_1} \sum_{y_i=1} \frac{1}{n_0} \sum_{y_j=0} \mathbf{1}_{[S(X_i) > S(X_j)]}, \quad (1)$$

$$\widehat{\mathrm{KS}}(S, y) = 1 - \max_{t \in \mathbb{R}} \sum_{i=1}^{n} \rho_i \mathbf{1}_{[S(X_i) \le t]}, \quad (2)$$

$$\widehat{\mathrm{P@k}}(S, y) = 1 - \frac{1}{n} \sum_{i=1}^{n} y_i \mathbf{1}_{[i \in \mathcal{M}_k]}, \quad (3)$$

where $\rho_i = 1/n_0$ if $y_i = 0$ and $\rho_i = -1/n_1$ if $y_i = 1$, and $\mathcal{M}_k$ denotes the set of indices $i = 1, \ldots, n$ achieving the highest k scores. In this paper, we focus on the Kolmogorov-Smirnov (KS) and area under the ROC cuve (AUC) losses, with the analogous results for the precision at k (P@k) loss relegated to the Supplementary Material.

This paper considers a stagewise derivative-free optimization procedure via boosted stumps tailored to the exact loss function with a margin condition, called ExactBoost. While margin theory is readily applicable in the decomposable setting (Zhai et al., 2013; Schapire & Freund, 2013), a novel extension is developed here for CND losses.

Given scores $\mathbf{S} = \{S(X_1), \ldots, S(X_n)\}$, labels $\mathbf{y} = \{y_1, \ldots, y_n\}$ and loss function $\widehat{L} : [-1, 1]^n \times \{0, 1\}^n \to \mathbb{R}$, ExactBoost minimizes $\hat{L}(\mathbf{S}, \mathbf{y})$ by iteratively solving

$$\min_h \widehat{L}_\theta(\mathbf{S} + h, \mathbf{y}) \quad \text{s.t.} \quad h(X) = a\mathbf{1}_{[X_{(j)} \le \xi]} + b\mathbf{1}_{[X_{(j)} > \xi]},$$

with $a, b, \xi \in \mathbb{R}, j \in \{1, \ldots, p\}$, and where $\widehat{L}_\theta$ is a margin-adjustment over $\widehat{L}$, $h : \mathbb{R}^p \to \mathbb{R}$ is a stump and $X_{(j)}$ denotes the jth feature. Using the discrete structure of CND losses and interval arithmetic, ExactBoost is of order $O(pn \log n)$.

Contributions We propose a fast, exact stagewise optimization algorithm for CND losses using interval arithmetic. The method comes with provable optimality bounds on its generalization error due to a novel extension of margin theory. Its empirical performances is comparable or superior to general-purpose and loss-specific procedures available.

2. Overview of ExactBoost

The main idea behind ExactBoost is stagewise optimization tailored to a margin-adjusted empirical loss. Take $(X_1, y_1), \ldots, (X_n, y_n) \overset{\text{iid}}{\sim} \mathcal{D}$ and a loss $\widehat{L}$ that is invariant under rescaling and translation in its first argument. The goal is to find a score function where higher scores $S(X_i)$ should indicate higher likelihood of $y_i = 1$. It will be assumed that, after T rounds, a score has the form

$$S(X_i) = \sum_{t=1}^{T} w_t h_t(X_i), \quad (4)$$

[*]Equal contribution [1]Instituto de Matemática Pura e Aplicada, Rio de Janeiro, Brazil [2]Princeton University, Princeton, USA. Correspondence to: Paulo Orenstein <pauloo@impa.br>, Roberto Oliveira <rimfo@impa.br>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

with $w_t \geq 0, \sum_{t=1}^{T} w_t = 1, h_t \in \mathcal{H}$, and the set of weak learners $\mathcal{H}$ are binary stumps of the form

$$\mathcal{H} = \left\{ \pm \mathbf{1}_{[X_{(j)} \leq \xi]} \pm \mathbf{1}_{[X_{(j)} > \xi]} \, : \, \xi \in \mathbb{R}, j \in [p] \right\}. \quad (5)$$

For stagewise minimization, at iteration t, set $(\alpha_t, h_t) = \operatorname{argmin}_{\alpha,h} \widehat{L}\left((\mathbf{S}_{t-1} + \alpha \mathbf{h})/(1+\alpha), \mathbf{y}\right)$, where $\alpha \geq 0$, $h \in \mathcal{H}$, $\mathbf{S}_{t-1} = (S_{t-1}(X_1), \ldots, S_{t-1}(X_n))$, $\mathbf{h} = (h(X_1), \ldots, h(X_n))$ and $\mathbf{y} = (y_1, \ldots, y_n)$. Then, let $S_t = (S_{t-1} + \alpha_t h_t)/(1+\alpha_t)$ or, as the loss is scaling-invariant, $S_t = S_{t-1} + \alpha_t h_t$, and go to the next iteration.

To attenuate overfitting, consider the margin-adjusted loss:

$$\widehat{L}_\theta(\mathbf{S}, \mathbf{y}) = \widehat{L}(\mathbf{S} - \theta \mathbf{y}, \mathbf{y}), \quad (6)$$

where $\theta > 0$ is a margin parameter. Scores for positive labels are artificially decreased, forcing the algorithm to increase the confidence when correctly classifying samples. More precisely, Theorem 2 below shows that the minimizing $\hat{L}_\theta$ via ExactBoost provides an upper bound on the value of the populational KS loss. Similar results are presented for the AUC and P@k losses in the supplementary material. Each of these losses is emblematic of a different level of non-decomposability and the corresponding proof techniques may be employed for losses of similar structure. These results recover a guarantee available to decomposable losses (Bartlett et al., 1998; Koltchinskii & Panchenko, 2002) and lead to algorithms that generalize well, as Section 4 shows.

The algorithm must iteratively solve

$$(\alpha_t, h_t) = \operatorname{argmin}_{\alpha,h} \widehat{L}\left(\mathbf{S}_{t-1} - \theta \mathbf{y} + \alpha(\mathbf{h} - \theta \mathbf{y}), \mathbf{y}\right),$$

for which it is enough to pick (ξ_t, j_t, a_t, b_t) that solves

$$\min_{\xi \in \mathbb{R}, j \in [p], \tilde{a}, \tilde{b} \in \mathbb{R}} \widehat{L}\Big(\mathbf{S}_{t-1} + \tilde{a}\mathbf{1}_{[X_{(j)} \leq \xi]} + \tilde{b}\mathbf{1}_{[X_{(j)} > \xi]} - (1 + |\tilde{b} - \tilde{a}|/2)\theta \mathbf{y}, \mathbf{y}\Big), \quad (7)$$

and set $\mathbf{S}_t = \mathbf{S}_{t-1} + a_t \mathbf{1}_{[X_{(j_t)} \leq \xi_t]} + b_t \mathbf{1}_{[X_{(j_t)} > \xi_t]}$. Note the discrete nature of CND losses allows the problem above to be solved by only considering a finite set of ξ, $\tilde{a}$ and $\tilde{b}$: for ξ, it suffices to look at the unique values of feature $X_{(j)}$, $j \in [p]$, and for a and b the unique values of $S(X_i)$, $i \in [n]$.

The resulting algorithm is called ExactBoost, as it is based on the exact loss function provided rather than a surrogate. Algorithm 1 includes the full pseudocode. Note it takes as input an initial set of scores which could be any starting point, including scores trained by other learning models.

In order to solve (7), we use a space subdivision scheme similar to bisection: for each iteration, we use interval arithmetic (IA) to evaluate an upper bound on the value of the loss for each subdivision, and further iterate only on the cell with the highest upper bound. This optimization procedure is of order $O(cf(n))$, where $f(n)$ is the cost of evaluating the loss with IA, and c is the desired number of bits of precision in our result. This makes ExactBoost $O(pn\log(n))$.

To avoid overfitting, subsampling is used (see Subsection 3 for theoretical guarantees). Also, randomized runs of the algorithm are averaged, similar in spirit to random forests, and can be trivially parallelized. More details about the implementation are in the Supplementary Material.

Algorithm 1 ExactBoost

function ExactBoost(data $(\mathbf{X}, \mathbf{y})$, initial scores S_0, margin θ, bins b, iterations T, estimator runs E)
for $e \in \{1, \ldots, E\}$ **do**
  $S_e \leftarrow S_0$
  for $t \in \{1, \ldots, T\}$ **do**
    $\mathbf{X}^s, \mathbf{y}^s \leftarrow$ subsample $\mathbf{X}, \mathbf{y}$
    for $j \in \{1, \ldots, p\}$ **do**
      $\widehat{L}(h) \leftarrow \widehat{L}_\theta(S_e(\mathbf{X}^s_{(j)}) + h(\mathbf{X}^s_{(j)}), \mathbf{y}^s)$
      $h_j \leftarrow \operatorname{argmin}_h \widehat{L}(h)$ //Algorithm 2
    end for
    $h \leftarrow \operatorname{argmin}_{h_j} \widehat{L}_\theta(S_e(\mathbf{X}^s) + h_j(\mathbf{X}^s), \mathbf{y}^s)$
    $S'_e \leftarrow S_e + h$
    if $\widehat{L}_\theta(S'_e(\mathbf{X}), \mathbf{y}) \leq \widehat{L}_\theta(S_e(\mathbf{X}), \mathbf{y})$ **then**
      $S_e \leftarrow S'_e$
      $S_e \leftarrow (S_e - \min S_e)/(\max S_e - \min S_e)$
    end if
  end for
end for
return $\operatorname{mean}(S_1, \ldots, S_E)$

Algorithm 2 Iterative Minimization

function Min(loss $\hat{L}_\theta$, data $\mathbf{X}_{(j)}$, labels $\mathbf{y}$, scores S, margin θ)
$\Xi \leftarrow [\min \mathbf{X}_{(j)}, \max \mathbf{X}_{(j)}]$; $A \leftarrow [-1, 1]$; $B \leftarrow [-1, 1]$
for $k \in \{1, \ldots, c\}$ **do**
  $l_\star \leftarrow +\infty$
  for bisections $(\Xi^{(b)}, A^{(b)}, B^{(b)})$ **do**
    for $i \in \{1, \ldots, n\}$ **do**
      $s \leftarrow S + A^{(b)}\mathbf{1}_{[\mathbf{X}_{(j)} \leq \Xi^{(b)}]} + B^{(b)}\mathbf{1}_{[\mathbf{X}_{(j)} > \Xi^{(b)}]}$
      $\mathbf{s}_i \leftarrow \underline{s}$ if $y_i = 0$ otherwise $\overline{s}$
    end for
    if $\widehat{L}_\theta(\mathbf{s}, \mathbf{y}) < l_\star$ **then**
      $l_\star \leftarrow \widehat{L}_\theta(\mathbf{s}, \mathbf{y})$ // $\widehat{L}_\theta(\mathbf{s}, \mathbf{y}) = [\underline{\widehat{L}_\theta(S, y)}, \overline{\widehat{L}_\theta(S, y)}]$
    end if
  end for
end for

3. Theoretical Results

Assume the score function is a convex combination of simple functions coming from a general family $\mathcal{H}$, as in (4). Then the population error of S can be upper bounded by the sum of a margin-adjusted sample error of S plus an error depending on $\mathcal{H}$. Crucially, the stochastic error is controlled uniformly over S and only depends on the class of simple functions $\mathcal{H}$. For ExactBoost, $\mathcal{H}$ is the set of stumps in (5),

and, from the results below, one can allow for a number of features that is nearly as large as an exponential in the number of positive and negative examples.

The theoretical results presented are based on the KS and AUC losses, but hold in far greater generality. For example, see the Supplementary Material for P@k results. While the guarantees for each loss are slightly different, the margin adjustment on each loss is essentially the same.

The results below follow previous work in obtaining empirical bounds for classification tasks (Schapire et al., 1998; Bartlett & Mendelson, 2002), though that work is not directly applicable to CND losses. The results presented here also differ in spirit from those obtained via surrogate losses, such as (Kar et al., 2015; Agarwal, 2013). Surrogate metrics can provide upper bounds of the desired loss but often lack a natural quantitative interpretation. The theorems below, on the other hand, guarantee that minimizing a margin-adjusted empirical loss leads, with high probability, to a small population loss. This is the main idea powering ExactBoost.

Notation. Let $\mathcal{D}$ be a probability distribution over $(X, y) \in \mathbb{R}^p \times \{0, 1\}$, and $\mathcal{D}_0$ (respectively, $\mathcal{D}_1$) denote the conditional distribution of X when $y = 0$ (respectively, 1). Conditionally on n_1 and n_0 respectively, the subsamples $\mathbf{X}_1 \coloneqq (X_i : i \in [n],\, y_i = 1)$ and $\mathbf{X}_0 \coloneqq (X_i : i \in [n], y_i = 0)$ are iid from $\mathcal{D}_1$ and $\mathcal{D}_0$. Score functions $S : \mathbb{R}^p \to [-1, 1]$ are convex combinations of elements in a family of measurable functions $\mathcal{H} : \mathbb{R}^p \to [-1, 1]$. Denote by $\mathcal{R}_n(\mathcal{H})$ the Rademacher complexity $\mathcal{H}$ with respect to $\mathcal{D}$ and $\mathcal{R}_{n,y}(\mathcal{H})$ for $y \in \{0, 1\}$ the complexities with respect to $\mathcal{D}_0$ and $\mathcal{D}_1$. Note $\mathcal{R}_n(\mathcal{H}) = O(\sqrt{\log p/n})$ when $\mathcal{H}$ is as in (5).

Define the populational AUC as

$$\mathrm{AUC}(S) \coloneqq 1 - \Pr\{S(X) > S(X')\}$$

and the populational KS loss as

$$\mathrm{KS}(S) = 1 - \sup_{t\in\mathbb{R}} \left(\Pr_{X\sim\mathcal{D}_0} \{S(X) \le t\} - \Pr_{X\sim\mathcal{D}_1} \{S(X) \le t\} \right).$$

Theorem 1. *Given $\theta > 0$, $\delta \in (0, 1)$, and a class of functions $\mathcal{H}$ from $\mathbb{R}^p$ to $[-1, 1]$, the following holds with probability at least $1-\delta$: for all score functions $S : \mathbb{R}^p \to [-1, 1]$ obtained as convex combinations of the elements of $\mathcal{H}$,*

$$\mathrm{AUC}(S) \le \widehat{\mathrm{AUC}}_\theta(S) + \frac{4}{\theta}\zeta_{\mathrm{AUC}}(\mathcal{H}) + \sqrt{\frac{2\log(1/\delta)}{\min\{n_0, n_1\}}},$$

where $\zeta_{\mathrm{AUC}}(\mathcal{H}) = \mathcal{R}_{\min\{n_0,n_1\},0}(\mathcal{H}) + \mathcal{R}_{\min\{n_0,n_1\},1}(\mathcal{H})$.

In the particular case of ExactBoost, where $\mathcal{H}$ is given by (5), the theorem implies that the score S produced by the algorithm satisfies $\mathrm{AUC}(S) \le \widehat{\mathrm{AUC}}_\theta(S) + o(1)$ with high probability whenever $\min\{n_0, n_1\} \gg \theta^{-2} \log p$. This result can be extended to similar pairwise losses.

Theorem 2. *Given $\theta > 0$, $\delta \in (0, 1)$, and a class of functions $\mathcal{H}$ from $\mathbb{R}^p$ to $[-1, 1]$, the following holds with probability at least $1-\delta$: for all score functions $S : \mathbb{R}^p \to [-1, 1]$ obtained as convex combinations of the elements of $\mathcal{H}$,*

$$\mathrm{KS}(S) \le \widehat{\mathrm{KS}}_\theta(S) + \frac{8}{\theta}\zeta_{\mathrm{KS}}(\mathcal{H}) + \sqrt{\frac{\log(2/\delta)}{2}}\left(\frac{1}{\sqrt{n_0}} + \frac{1}{\sqrt{n_1}}\right),$$

where $\zeta_{\mathrm{KS}}(\mathcal{H}) = \mathcal{R}_{n_0,0}(\mathcal{H}) + \mathcal{R}_{n_1,1}(\mathcal{H}) + n_0^{-1/2} + n_1^{-1/2}$.

In words, a score that achieves a small margin-adjusted KS loss should, with high probability, have good performance on the population. In the specific case of ExactBoost, the theorem above yields with probability greater than $1 - \delta$,

$$\mathrm{KS}(S) \le \widehat{\mathrm{KS}}_\theta(S) + C\sqrt{\frac{\theta^{-2}(1 + \log p) + \log(2/\delta)}{\min\{n_0, n_1\}}},$$

with $C > 0$ a universal constant. Treating δ as fixed, good training performance on the margin-adjusted loss leads to good generalization when the number of positive and negative examples in the data are much larger than $\theta^{-2} \log p$.

Subsampling Subsampling can help ExactBoost avoid overfitting. The next proposition is helpful in controlling its impact in the optimization procedure for some losses.

Proposition 1. *Let $\widehat{L}$ be either the $\widehat{\mathrm{KS}}$ or the $\widehat{\mathrm{AUC}}$ loss. Consider a subset of indices $I = I_0 \cup I_1 \subset [n]$ chosen independently and uniformly at random with equal number of positive and negative cases, $|I_0| = |I_1| = k$. Let h_R be the optimal stump over the reduced sample $\{(X_j, y_j)\}_{j\in I}$ and score S and h_* the optimal stump over the entire sample $\{(X_i, y_i)\}_{i\in[n]}$. Then,*

$$\mathbb{E}[\widehat{L}(S + h_R)] \le \widehat{L}(S + h_*) + \frac{e}{k},$$

where the expectation is over the choice of I.

Hence, using random subsets of observations in ExactBoost leads to an expected error close to optimal when the subset has a balanced proportion of positive and negative examples.

Ensembling For some losses, it is possible to guarantee that the empirical loss of the ensembler is smaller than the empirical loss of each ensemble member. See the Supplementary Material for a formal result.

4. Experiments

To test its performance, ExactBoost is compared against 10 exact and surrogate-based algorithms, both as a standalone estimator and as an ensembler, on 30 heterogeneous datasets. For ease of presentation, KS and AUC losses results of 6 representative datasets are shown in the main paper; the rest are in the Supplementary Material. Table 1 displays the

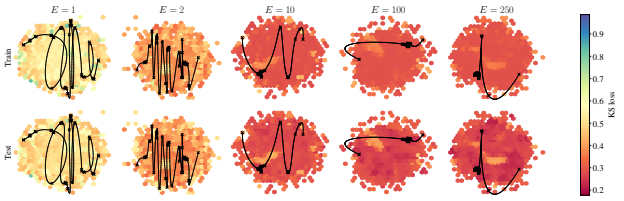

Figure 1. KS loss landscape highlighting ExactBoost's optimization trajectories on dataset heart via UMAP, going from left to right. More runs E lead to better train and test losses.

Dataset	Obs.	Features	Positives	RankBoost	DMKS
a1a	1605	119	24.6%	3.10x	34.50x
german	1000	20	70.0%	11.10x	2.00x
gisette	6000	5000	50.0%	OOT	36.70x
gmsc	150000	10	6.7%	OOT	87.50x
heart	303	21	45.9%	1.90x	23.70x
iono	351	34	64.1%	2.90x	4.70x
liver	145	5	37.9%	3.50x	17.40x

Table 1. Dataset properties and timings of algorithms relative to ExactBoost. ExactBoost is always faster ($> 1\times$). OOT indicates the time budget of 5 days was exceeded (see the Supplementary Material for details on computational setup and budget).

main characteristics of each dataset, which span various applications, and range from balanced to imbalanced. Sources for the data can be found in the Supplementary Material.

Benchmarks ExactBoost is compared to many learning algorithms: AdaBoost, k-nearest neighbors, logistic regression and random forest (via their Scikit-Learn implementation, (Pedregosa et al., 2011)), gradient boosting (via XGBoost, (Chen & Guestrin, 2016)) and a 4-layer connected neural net (via TensorFlow, (Abadi et al., 2015)). Methods that specifically optimize the performance metric are also considered. For KS, the baseline is DMKS (Fang & Chen, 2019). For AUC, it is RankBoost (Freund et al., 2003).

Hyperparameters Hyperparameters were fixed throughout the experiments. Baseline models were trained with the package-provided hyperparameters; see the Supplementary Material. Aided by experimental evidence on held-out

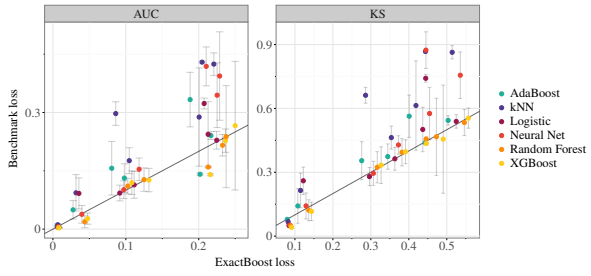

Figure 2. Test error for ExactBoost vs surrogate methods as estimators. Alternatives are generally worse than ExactBoost or statistically indistinguishable.

Dataset	AUC ExactBoost	AUC RankBoost	KS ExactBoost	KS DMKS
a1a	$\mathbf{0.11 \pm 0.0}$	0.13 ± 0.0	$\mathbf{0.37 \pm 0.0}$	$\mathbf{0.37 \pm 0.0}$
german	$\mathbf{0.23 \pm 0.0}$	0.24 ± 0.0	$\mathbf{0.53 \pm 0.0}$	0.55 ± 0.0
gisette	$\mathbf{0.01 \pm 0.0}$	OOT	0.09 ± 0.0	$\mathbf{0.06 \pm 0.0}$
gmsc	$\mathbf{0.21 \pm 0.0}$	OOT	$\mathbf{0.44 \pm 0.0}$	0.45 ± 0.0
heart	$\mathbf{0.09 \pm 0.0}$	0.13 ± 0.0	0.30 ± 0.0	$\mathbf{0.28 \pm 0.0}$
iono	$\mathbf{0.04 \pm 0.0}$	$\mathbf{0.04 \pm 0.0}$	$\mathbf{0.13 \pm 0.0}$	0.28 ± 0.0
liver	$\mathbf{0.22 \pm 0.1}$	0.32 ± 0.1	$\mathbf{0.45 \pm 0.1}$	0.50 ± 0.1

Table 2. Evaluation of loss-specific optimizers. ExactBoost has the best performance, it is faster and uses less memory.

Dataset	ExactBoost	AdaBoost	Logistic	XGBoost	RankBoost
a1a	$\mathbf{0.13 \pm 0.0}$	0.17 ± 0.0	0.14 ± 0.0	0.28 ± 0.1	0.16 ± 0.0
german	$\mathbf{0.23 \pm 0.0}$	0.32 ± 0.0	0.24 ± 0.0	0.35 ± 0.0	0.30 ± 0.1
gisette	$\mathbf{0.00 \pm 0.0}$	0.01 ± 0.0	0.01 ± 0.0	0.02 ± 0.0	0.01 ± 0.0
gmsc	0.15 ± 0.0	$\mathbf{0.14 \pm 0.0}$	0.31 ± 0.0	0.41 ± 0.0	0.15 ± 0.0
heart	$\mathbf{0.12 \pm 0.0}$	0.18 ± 0.1	$\mathbf{0.12 \pm 0.0}$	0.23 ± 0.1	0.15 ± 0.0
iono.	$\mathbf{0.04 \pm 0.0}$	0.05 ± 0.0	0.07 ± 0.0	0.09 ± 0.0	0.05 ± 0.0
liver	$\mathbf{0.30 \pm 0.1}$	0.34 ± 0.1	0.34 ± 0.1	0.38 ± 0.0	0.38 ± 0.1

Table 3. Evaluation of AUC ensemblers. ExactBoost is generally the top performer. The full table is in the Supplementary Material.

Dataset	ExactBoost	AdaBoost	Logistic	XGBoost	DMKS
a1a	$\mathbf{0.37 \pm 0.1}$	0.44 ± 0.1	0.40 ± 0.1	0.57 ± 0.1	0.49 ± 0.1
german	$\mathbf{0.50 \pm 0.1}$	0.68 ± 0.1	0.53 ± 0.1	0.69 ± 0.1	0.53 ± 0.1
gisette	$\mathbf{0.04 \pm 0.0}$	$\mathbf{0.04 \pm 0.0}$	0.07 ± 0.0	$\mathbf{0.04 \pm 0.0}$	0.10 ± 0.0
gmsc	$\mathbf{0.43 \pm 0.0}$	0.44 ± 0.0	0.73 ± 0.0	0.83 ± 0.0	0.46 ± 0.0
heart	$\mathbf{0.34 \pm 0.1}$	0.38 ± 0.1	0.37 ± 0.1	0.46 ± 0.1	0.40 ± 0.0
iono.	$\mathbf{0.13 \pm 0.1}$	0.18 ± 0.1	0.18 ± 0.1	0.19 ± 0.1	0.27 ± 0.1
liver	$\mathbf{0.53 \pm 0.1}$	0.60 ± 0.2	0.59 ± 0.2	0.76 ± 0.0	0.60 ± 0.2

Table 4. Evaluation of KS ensemblers. ExactBoost is always the top performer. The full table is in the Supplementary Material.

datasets, ExactBoost uses $E = 250$ runs, $T = 50$ rounds, subsampling of 20% and margin of $\theta = 0.05$. See Figure 1.

Results ExactBoost is generally better than loss-specific optimizers, as illustrated in Table 2. Table 1 includes some timing comparisons showing that ExactBoost scales well even to large datasets, while Figure 2 shows that ExactBoost also has good performance against surrogate alternatives. Full results are included the Supplementary Material.

Ensembling ExactBoost is also a great ensembler. Six base models were used: AdaBoost, k-nearest neighbors, logistic regression, neural network, random forest and XGBoost. These models were trained on training folds, and their predictions on test folds were used as features for the ensemble models. Tables 3 and 4 show the results of using different surrogate and exact models as ensemblers. The surrogate ensemblers were AdaBoost, logistic regression and XGBoost, while the exact benchmarks were given by RankBoost (for AUC) and DMKS (for KS).

As shown in Tables 3 and 4, ExactBoost is generally the best ensembler available. In fact, it is also robust to noisy features coming from poorly performing base models. This is particularly attractive because, given the discrete nature of combinatorial losses, it is often the case that the best performing optimizer changes from dataset to dataset.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org.

Agarwal, S. Surrogate regret bounds for the area under the roc curve via strongly proper losses. In *Conference on Learning Theory*, pp. 338–353. PMLR, 2013.

Bartlett, P. and Mendelson, S. Rademacher and Gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3(Nov):463–482, 2002. URL https://eprints.qut.edu.au/43936/.

Bartlett, P., Freund, Y., Lee, W. S., and Schapire, R. E. Boosting the margin: a new explanation for the effectiveness of voting methods. *The Annals of Statistics*, 26(5):1651 – 1686, 1998. doi: 10.1214/aos/1024691352. URL https://doi.org/10.1214/aos/1024691352.

Chen, T. and Guestrin, C. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pp. 785–794, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4232-2. doi: 10.1145/2939672.2939785. URL http://doi.acm.org/10.1145/2939672.2939785.

Fang, F. and Chen, Y. A new approach for credit scoring by directly maximizing the kolmogorov–smirnov statistic. *Computational Statistics & Data Analysis*, 133:180–194, 2019. ISSN 0167-9473. URL https://doi.org/10.1016/j.csda.2018.10.004.

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

Kar, P., Narasimhan, H., and Jain, P. Surrogate functions for maximizing precision at the top. In *International Conference on Machine Learning*, pp. 189–198. PMLR, 2015.

Koltchinskii, V. and Panchenko, D. Empirical margin distributions and bounding the generalization error of combined classifiers. *Ann. Statist.*, 30(1):1–50, 02 2002. doi: 10.1214/aos/1015362183. URL https://doi.org/10.1214/aos/1015362183.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Schapire, R. E. and Freund, Y. Boosting: Foundations and algorithms. *Kybernetes*, 2013.

Schapire, R. E., Freund, Y., Bartlett, P., and Lee, W. S. Boosting the margin: a new explanation for the effectiveness of voting methods. *Ann. Statist.*, 26(5):1651–1686, 10 1998. doi: 10.1214/aos/1024691352. URL https://doi.org/10.1214/aos/1024691352.

Zhai, S., Xia, T., Tan, M., and Wang, S. Direct 0-1 loss minimization and margin maximization with boosting. In Burges, C. J. C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems 26*, pp. 872–880. Curran Associates, Inc., 2013.