ExactBoost: Directly Boosting Combinatorial and Non-decomposable Metrics

Daniel Csillag Carolina Piazza Thiago Ramos João Vitor Romano
Roberto Oliveira Paulo Orenstein

Instituto Nacional de Matemática Pura e Aplicada (IMPA)

Overview

Introduction

We have data $(X_i, y_i)_{i=1}^n$, with $X_i \in \mathbb{R}^p$, and $y_i \in \{0, 1\}$.

Introduction

We have data $(X_i, y_i)_{i=1}^n$, with $X_i \in \mathbb{R}^p$, and $y_i \in \{0, 1\}$.

We want to learn a good score function $S : \mathbb{R}^p \to [-1, 1]$.

Introduction

We have data $(X_i, y_i)_{i=1}^n$, with $X_i \in \mathbb{R}^p$, and $y_i \in \{0, 1\}$.

We want to learn a good score function $S : \mathbb{R}^p \to [-1, 1]$.

And we want to optimize notable metrics! In particular, we'll be working with

- AUC
- KS
- P@k

But our approach works for many other metrics as well.

AUC, KS and P@k

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

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

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

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.

Combinatorial and Non-Decomposable Loss Functions

Combinatorial and Non-Decomposable Loss Functions

A combinatorial loss function is one that is computed in terms of indicator functions.

Combinatorial and Non-Decomposable Loss Functions

A combinatorial loss function is one that is computed in terms of indicator functions.

A decomposable function is one where

$$\widehat{L}(S, y) = \frac{1}{n}\sum_{i=1}^{n}\widehat{L}(S_i, y_i)$$

The metrics presented are *non*-decomposable.

Combinatorial and Non-Decomposable Loss Functions

A combinatorial loss function is one that is computed in terms of indicator functions.

A decomposable function is one where

$$\widehat{L}(S, y) = \frac{1}{n} \sum_{i=1}^{n} \widehat{L}(S_i, y_i)$$

The metrics presented are *non*-decomposable.

This means that common approaches, such as

- Convex Optimization
- Stochastic Gradient Descent

Can't be readily applied!

Previous Approaches to Optimizing CND Losses

Previous Approaches to Optimizing CND Losses

- Surrogates

Previous Approaches to Optimizing CND Losses

- Surrogates
  - Specialized surrogates
  - General surrogates

Previous Approaches to Optimizing CND Losses

- Surrogates
  - Specialized surrogates
  - General surrogates
- Exact Algorithms

Previous Approaches to Optimizing CND Losses

- Surrogates
  - Specialized surrogates
  - General surrogates
- Exact Algorithms
  - RankBoost (Freund et al., 2003)
  - DMKS (Fang and Chen, 2019)
  - TopPush (Li et al., 2014)

Boosting

Boosting

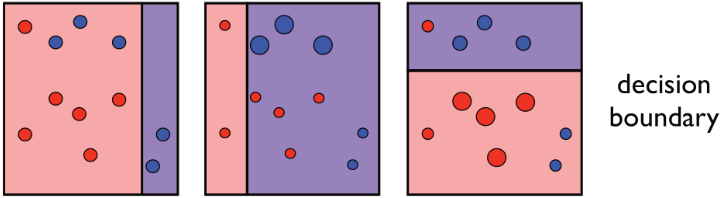


Image taken from Mehryar Mohri, et al. *Foundations of Machine Learning*, second edition, page 147

Boosting

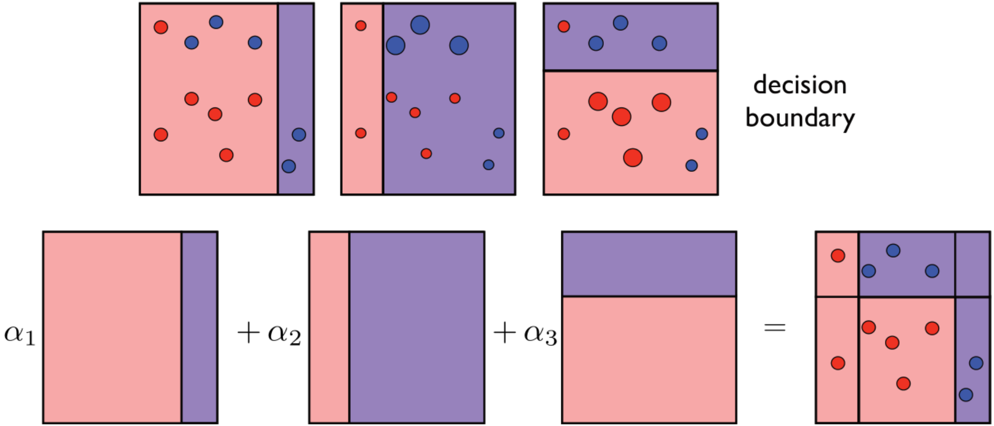


Image taken from Mehryar Mohri, et al. *Foundations of Machine Learning*, second edition, page 147

Our Boosting Framework

$$S = S_0 + \sum_i \alpha_i h_i(X), \qquad \sum_i \alpha_i = 1$$

Where

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

Our Boosting Framework

$$S = S_0 + \sum_i \alpha_i h_i(X), \qquad \sum_i \alpha_i = 1$$

Where

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

Why stumps?

Our Boosting Framework

$$S = S_0 + \sum_i \alpha_i h_i(X), \qquad \sum_i \alpha_i = 1$$

Where

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

Why stumps?

- Small complexity;

Our Boosting Framework

$$S = S_0 + \sum_i \alpha_i h_i(X), \qquad \sum_i \alpha_i = 1$$

Where

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

Why stumps?

- Small complexity;
- Fast to compute;

Our Boosting Framework

$$S = S_0 + \sum_i \alpha_i h_i(X), \qquad \sum_i \alpha_i = 1$$

Where

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

Why stumps?

- Small complexity;
- Fast to compute;
- Fast to optimize;

Our Boosting Framework

$$S = S_0 + \sum_i \alpha_i h_i(X), \qquad \sum_i \alpha_i = 1$$

Where

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

Why stumps?

- Small complexity;
- Fast to compute;
- Fast to optimize;
- Simplicity helps in preventing overfitting.

Optimizing the Weak Learners

$$(\alpha_t, h_t) = \arg\min_{\alpha, h} \widehat{L}_\theta \left(\frac{1}{1+\alpha} S_{t-1} + \frac{\alpha}{1+\alpha} h(X), y \right)$$

Optimizing the Weak Learners

$$
\begin{aligned}
(\alpha_t, h_t) &= \arg\min_{\alpha,h} \widehat{L}_\theta \left(\frac{1}{1+\alpha} S_{t-1} + \frac{\alpha}{1+\alpha} h(X), y \right) \\
&= \arg\min_{\alpha,h} \widehat{L} \left(\frac{1}{1+\alpha} S_{t-1} + \frac{\alpha}{1+\alpha} h(X) - \theta y, y \right)
\end{aligned}
$$

Optimizing the Weak Learners

$$
\begin{aligned}
(\alpha_t, h_t) &= \underset{\alpha,h}{\arg\min}\, \widehat{L}_\theta \left(\frac{1}{1+\alpha} S_{t-1} + \frac{\alpha}{1+\alpha} h(X), y \right) \\
&= \underset{\alpha,h}{\arg\min}\, \widehat{L} \left(\frac{1}{1+\alpha} S_{t-1} + \frac{\alpha}{1+\alpha} h(X) - \theta y, y \right) \\
&= \underset{\alpha,h}{\arg\min}\, \widehat{L}(S_{t-1} - \theta y + \alpha(h(X) - \theta y), y)
\end{aligned}
$$

Optimizing the Weak Learners

$$
\begin{aligned}
(\alpha_t, h_t) &= \arg\min_{\alpha,h} \widehat{L}_\theta\left(\frac{1}{1+\alpha}S_{t-1} + \frac{\alpha}{1+\alpha}h(X), y\right) \\
&= \arg\min_{\alpha,h} \widehat{L}\left(\frac{1}{1+\alpha}S_{t-1} + \frac{\alpha}{1+\alpha}h(X) - \theta y, y\right) \\
&= \arg\min_{\alpha,h} \widehat{L}(S_{t-1} - \theta y + \alpha(h(X) - \theta y), y) \\
&= \arg\min_{\alpha,h} \widehat{L}\left(S_{t-1} + a\mathbf{1}_{[X_{(j)} \le \xi]} + b\mathbf{1}_{[X_{(j)} > \xi]} - \left(1 - \frac{|b-a|}{2}\right)\theta y, y\right)
\end{aligned}
$$

Optimizing the Weak Learners

$$
\begin{aligned}
(\alpha_t, h_t) &= \operatorname*{arg\,min}_{\alpha,h} \widehat{L}_\theta \left(\frac{1}{1+\alpha} S_{t-1} + \frac{\alpha}{1+\alpha} h(X), y \right) \\
&= \operatorname*{arg\,min}_{\alpha,h} \widehat{L} \left(\frac{1}{1+\alpha} S_{t-1} + \frac{\alpha}{1+\alpha} h(X) - \theta y, y \right) \\
&= \operatorname*{arg\,min}_{\alpha,h} \widehat{L}(S_{t-1} - \theta y + \alpha(h(X) - \theta y), y) \\
&= \operatorname*{arg\,min}_{\alpha,h} \widehat{L} \left(S_{t-1} + a\mathbf{1}_{[X_{(j)} \le \xi]} + b\mathbf{1}_{[X_{(j)} > \xi]} - \left(1 - \frac{|b-a|}{2}\right) \theta y, y \right) \\
&= \operatorname*{arg\,min}_{\alpha,h} \widehat{L} \left(S_{t-1} + a\mathbf{1}_{[X_{(j)} \le \xi]} + b\mathbf{1}_{[X_{(j)} > \xi]} - M(\theta, y), y \right)
\end{aligned}
$$

Optimizing the Weak Learners

$$
\begin{aligned}
(\alpha_t, h_t) &= \arg\min_{\alpha,h} \widehat{L}_\theta\left(\frac{1}{1+\alpha}S_{t-1} + \frac{\alpha}{1+\alpha}h(X), y\right) \\
&= \arg\min_{\alpha,h} \widehat{L}\left(\frac{1}{1+\alpha}S_{t-1} + \frac{\alpha}{1+\alpha}h(X) - \theta y, y\right) \\
&= \arg\min_{\alpha,h} \widehat{L}(S_{t-1} - \theta y + \alpha(h(X) - \theta y), y) \\
&= \arg\min_{\alpha,h} \widehat{L}\left(S_{t-1} + a\mathbf{1}_{[X_{(j)} \leq \xi]} + b\mathbf{1}_{[X_{(j)} > \xi]} - \left(1 - \frac{|b-a|}{2}\right)\theta y, y\right) \\
&= \arg\min_{\alpha,h} \widehat{L}\left(S_{t-1} + a\mathbf{1}_{[X_{(j)} \leq \xi]} + b\mathbf{1}_{[X_{(j)} > \xi]} - M(\theta, y), y\right)
\end{aligned}
$$

We then update the score as follows:

$$
S_t = S_{t-1} + a\mathbf{1}_{[X_{(j)} \leq \xi]} + b\mathbf{1}_{[X_{(j)} > \xi]}.
$$

Stagewise Minimization Procedure

function $\textsc{ExactBoost}$(data (X, y), initial score S_0)
    $S \leftarrow S_0$
    for $t \in \{1, \ldots, T\}$ **do**
        for $j \in \{1, \ldots, p\}$ **do**
            $(\xi, a, b) \leftarrow \arg\min_{\xi,a,b} \widehat{L}(S + a\mathbf{1}_{[X_{(j)} \leq \xi]} + b\mathbf{1}_{[X_{(j)} > \xi]} - M(\theta, y), y)$
            $h_j \leftarrow a\mathbf{1}_{[X_{(j)} \leq \xi]} + b\mathbf{1}_{[X_{(j)} > \xi]}$
        $S' \leftarrow S + \arg\min_{h_j} \widehat{L}(S + h_j, y)$
        if $\widehat{L}(S', y) \leq \widehat{L}(S, y)$ **then**
            $S \leftarrow S'$
    return S

Subsampling

function $\textsc{ExactBoost}$(data (X, y), initial score S_0)
  $S \leftarrow S_0$
  for $t \in \{1, \ldots, T\}$ **do**
    $X^s, y^s \leftarrow$ subsample X, y
    for $j \in \{1, \ldots, p\}$ **do**
      $(\xi, a, b) \leftarrow \arg\min_{\xi,a,b} \widehat{L}(S + a\mathbf{1}_{[X^s_{(j)} \leq \xi]} + b\mathbf{1}_{[X^s_{(j)} > \xi]} - M(\theta, y), y^s)$
      $h_j \leftarrow a\mathbf{1}_{[X^s_{(j)} \leq \xi]} + b\mathbf{1}_{[X^s_{(j)} > \xi]}$
    $S' \leftarrow S + \arg\min_{h_j} \widehat{L}(S + h_j, y^s)$
    if $\widehat{L}(S', y) \leq \widehat{L}(S, y)$ **then**
      $S \leftarrow S'$
  return S

Run Averaging

function $\textsc{ExactBoost}$(data (X, y), initial score S_0)
  for $e \in \{1, \ldots, E\}$ **do**
    $S_e \leftarrow S_0$
    for $t \in \{1, \ldots, T\}$ **do**
      $X^s, y^s \leftarrow$ subsample X, y
      for $j \in \{1, \ldots, p\}$ **do**
        $(\xi, a, b) \leftarrow \arg\min_{\xi, a, b} \widehat{L}(S_e + a\mathbf{1}_{[X^s_{(j)} \leq \xi]} + b\mathbf{1}_{[X^s_{(j)} > \xi]} - M(\theta, y), y^s)$
        $h_j \leftarrow a\mathbf{1}_{[X^s_{(j)} \leq \xi]} + b\mathbf{1}_{[X^s_{(j)} > \xi]}$
      $S'_e \leftarrow S_e + \arg\min_{h_j} \widehat{L}(S_e + h_j, y^s)$
      if $\widehat{L}(S'_e, y) \leq \widehat{L}(S_e, y)$ **then**
        $S_e \leftarrow S'_e$
  return $\mathrm{mean}(S_1, \ldots, S_E)$

Optimization Procedure

function $\textsc{Optimize}$(feature $X_{(j)}$, labels y, score S)
    $\Xi \leftarrow [\min X_{(j)}, \max X_{(j)}]$
    $A \leftarrow [-1, 1]$
    $B \leftarrow [-1, 1]$
    for $k \in \{1, \ldots, c\}$ **do**
        for bisections Ξ', A', B' **do**
            $\ell_{\Xi', A', B'} \leftarrow$ estimate best metric in this subdomain
        $(\Xi, A, B) \leftarrow \arg\min_{(\Xi', A', B')} \ell_{\Xi', A', B'}$
    $(\xi_\star, a_\star, b_\star) \leftarrow \arg\min_{\xi \in \{\underline{\Xi}, \overline{\Xi}\}, a \in \{\underline{A}, \overline{A}\}, b \in \{\underline{B}, \overline{B}\}} \widehat{L}(S + a\mathbf{1}_{[X_{(j)} \leq \xi]} + b\mathbf{1}_{[X_{(j)} > \xi]}, y)$
    return $(\xi_\star, a_\star, b_\star)$

Interval Arithmetic

$$X + Y = [\underline{X}, \overline{X}] + [\underline{Y}, \overline{Y}] = [\underline{X} + \underline{Y}, \overline{X} + \overline{Y}]$$
$$X - Y = [\underline{X}, \overline{X}] - [\underline{Y}, \overline{Y}] = [\underline{X} - \overline{Y}, \overline{X} - \underline{Y}]$$

Interval Arithmetic

$$X + Y = [\underline{X}, \overline{X}] + [\underline{Y}, \overline{Y}] = [\underline{X} + \underline{Y}, \overline{X} + \overline{Y}]$$
$$X - Y = [\underline{X}, \overline{X}] - [\underline{Y}, \overline{Y}] = [\underline{X} - \overline{Y}, \overline{X} - \underline{Y}]$$
$$X \cdot Y = [\underline{X}, \overline{X}] \cdot [\underline{Y}, \overline{Y}] = [\min M, \max M]$$
$$\text{where } M = \{\underline{X} \cdot \underline{Y}, \underline{X} \cdot \overline{Y}, \overline{X} \cdot \underline{Y}, \overline{X} \cdot \overline{Y}\}$$

Interval Arithmetic

$$X + Y = [\underline{X}, \overline{X}] + [\underline{Y}, \overline{Y}] = [\underline{X} + \underline{Y}, \overline{X} + \overline{Y}]$$
$$X - Y = [\underline{X}, \overline{X}] - [\underline{Y}, \overline{Y}] = [\underline{X} - \overline{Y}, \overline{X} - \underline{Y}]$$
$$X \cdot Y = [\underline{X}, \overline{X}] \cdot [\underline{Y}, \overline{Y}] = [\min M, \max M]$$
$$\text{where } M = \{\underline{X} \cdot \underline{Y}, \underline{X} \cdot \overline{Y}, \overline{X} \cdot \underline{Y}, \overline{X} \cdot \overline{Y}\}$$
$$f(X) = f([\underline{X}, \overline{X}]) = [f(\underline{X}), f(\overline{X})], \quad f \text{ increasing}$$
$$f(X) = f([\underline{X}, \overline{X}]) = [f(\overline{X}), f(\underline{X})], \quad f \text{ decreasing}$$

Interval Arithmetic

$$X + Y = [\underline{X}, \overline{X}] + [\underline{Y}, \overline{Y}] = [\underline{X} + \underline{Y}, \overline{X} + \overline{Y}]$$

$$X - Y = [\underline{X}, \overline{X}] - [\underline{Y}, \overline{Y}] = [\underline{X} - \overline{Y}, \overline{X} - \underline{Y}]$$

$$X \cdot Y = [\underline{X}, \overline{X}] \cdot [\underline{Y}, \overline{Y}] = [\min M, \max M]$$

$$\text{where } M = \{\underline{X} \cdot \underline{Y}, \underline{X} \cdot \overline{Y}, \overline{X} \cdot \underline{Y}, \overline{X} \cdot \overline{Y}\}$$

$$f(X) = f([\underline{X}, \overline{X}]) = [f(\underline{X}), f(\overline{X})], \quad f \text{ increasing}$$

$$f(X) = f([\underline{X}, \overline{X}]) = [f(\overline{X}), f(\underline{X})], \quad f \text{ decreasing}$$

$$H(X) = H([\underline{X}, \overline{X}]) = [H(\underline{X}), H(\overline{X})]$$

$$\mathbf{1}_{[X<Y]} = \mathbf{1}_{[[\underline{X}, \overline{X}] < [\underline{Y}, \overline{Y}]]} = [\overline{X} < \underline{Y}, \underline{X} < \overline{Y}]$$

Interval Arithmetic, Applied to our Losses

Overall, we have

$$\underline{\widehat{L}(S, y)} = \widehat{L}(\overline{S}y + \underline{S}(1 - y), y)$$
$$\overline{\widehat{L}(S, y)} = \widehat{L}(\underline{S}y + \overline{S}(1 - y), y)$$

Which we know how to efficiently evaluate.

Optimization Procedure

function OPTIMIZE(feature $X_{(j)}$, labels y, score S)

$\Xi \leftarrow [\min X_{(j)}, \max X_{(j)}]$

$A \leftarrow [-1, 1]$

$B \leftarrow [-1, 1]$

for $k \in \{1, \dots, c\}$ **do**

for bisections Ξ', A', B' **do**

$S' \leftarrow S + A' \mathbf{1}_{[X_{(j)} \leq \Xi']} + B' \mathbf{1}_{[X_{(j)} > \Xi']}$

$\ell_{\Xi', A', B'} \leftarrow \widehat{L}(\overline{S'} y + \underline{S'}(1 - y), y)$

$(\Xi, A, B) \leftarrow \arg\min_{(\Xi', A', B')} \ell_{\Xi', A', B'}$

$(\xi_\star, a_\star, b_\star) \leftarrow \arg\min_{\xi \in \{\underline{\Xi}, \overline{\Xi}\}, a \in \{\underline{A}, \overline{A}\}, b \in \{\underline{B}, \overline{B}\}} \widehat{L}(S + a\mathbf{1}_{[X_{(j)} \leq \xi]} + b\mathbf{1}_{[X_{(j)} > \xi]}, y)$

return $(\xi_\star, a_\star, b_\star)$

Two Ways to Use ExactBoost

Two Ways to Use ExactBoost

As an estimator

Feature 1	Feature 2	$\cdots$	Feature $p-1$	Feature p	Label
0.2	100	$\cdots$	10	22.3	0
1.1	27	$\cdots$	5200	22.1	1
0.7	0	$\cdots$	3201	22.3	0
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$
0.6	45	$\cdots$	614	22.3	0
0.9	-21	$\cdots$	1023	22.3	1

Two Ways to Use ExactBoost

As an ensembler

Logistic Regression	k-NN	$\cdots$	AdaBoost	Random Forest	Label
0.1	0.2	$\cdots$	0.1	0.3	0
0.4	0.8	$\cdots$	0.7	0.9	1
0.2	0.4	$\cdots$	0.3	0.2	0
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$
0.3	0.6	$\cdots$	0.1	0.2	0
0.7	0.9	$\cdots$	0.6	0.8	1

Empirical and Generalization Errors

- Empirical Error is measured by $\widehat{L}(S(X), y)$ over data sample $(X_i, y_i)_{i=1}^{n} \sim \mathcal{D}$.
  - $\widehat{\mathrm{AUC}}$;
  - $\widehat{\mathrm{KS}}$;
  - $\widehat{\mathrm{P@k}}$.

Empirical and Generalization Errors

- Empirical Error is measured by $\widehat{L}(S(X), y)$ over data sample $(X_i, y_i)_{i=1}^n \sim \mathcal{D}$.
  - $\widehat{\mathrm{AUC}}$;
  - $\widehat{\mathrm{KS}}$;
  - $\widehat{\mathrm{P@k}}$.
- Generalization Error, not directly accessible, is the true error:

$$L(S) = \mathbb{E}_{\mathcal{D}}[\widehat{L}(S, y)].$$

Empirical and Generalization Errors

- Empirical Error is measured by $\widehat{L}(S(X), y)$ over data sample $(X_i, y_i)_{i=1}^n \sim \mathcal{D}$.
  - $\widehat{\mathrm{AUC}}$;
  - $\widehat{\mathrm{KS}}$;
  - $\widehat{\mathrm{P@k}}$.
- Generalization Error, not directly accessible, is the true error:

$$L(S) = \mathbb{E}_{\mathcal{D}}[\widehat{L}(S, y)].$$

- Classical results manage to bound the generalization error of decomposable losses.

AUC, KS and P@k, Revisited
Population Losses

Given $(X, X') \sim \mathcal{D}_1 \times \mathcal{D}_0$, the population losses are

$$\begin{aligned}
\mathrm{AUC}(S) &:= 1 - \Pr\{S(X) > S(X')\}, \\
\mathrm{KS}(S) &:= 1 - \sup_{t \in \mathbb{R}} \left(\Pr\{S(X') \leq t\} - \Pr\{S(X) \leq t\}\right), \\
\mathrm{P@k}_\alpha(S) &:= 1 - \Pr\{y = 1, S(X) \geq t_\alpha(S)\},
\end{aligned}$$

where $t_\alpha(S)$ denotes the $(1-\alpha)$-quantile under the population distribution, i.e.

$$t_\alpha(S) := \inf\left\{t \in \mathbb{R} : \Pr\{S(X) \leq t\} \geq 1 - \alpha\right\}.$$

Margin Theory

Intuition: high-confidence correct classifications should indicate better generalization.

Margin Theory

Intuition: high-confidence correct classifications should indicate better generalization.

Theory readily applicable in the decomposable setting; AdaBoost a classical example.

Margin Theory

Intuition: high-confidence correct classifications should indicate better generalization.

Theory readily applicable in the decomposable setting; AdaBoost a classical example.

We develop a novel extension for non-decomposable losses.

Margin Theory

Intuition: high-confidence correct classifications should indicate better generalization.

Theory readily applicable in the decomposable setting; AdaBoost a classical example.

We develop a novel extension for non-decomposable losses.

Main idea:

- Artificially decrease scores for positive labels;
- Algorithm is forced to increase the confidence when correctly classifying samples;
- Equivalent to imposing high confidence on negative cases;
- Attenuates overfitting.

Margin Theory

Intuition: high-confidence correct classifications should indicate better generalization.

Theory readily applicable in the decomposable setting; AdaBoost a classical example.

We develop a novel extension for non-decomposable losses.

Main idea:

- Artificially decrease scores for positive labels;
- Algorithm is forced to increase the confidence when correctly classifying samples;
- Equivalent to imposing high confidence on negative cases;
- Attenuates overfitting.

Notion of margin is used to bound generalization error.

AUC, KS and P@k, Revisited

Margin-Adjusted Empirical Losses

We also define θ-margin-adjusted versions of the empirical losses:

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

$$\widehat{\mathrm{KS}}_\theta(S) := 1 - \max_{t\in\mathbb{R}}\left(\frac{1}{n_0}\sum_{i\,:\,y_i=0}\mathbf{1}_{[S(X_i)\le t]} - \frac{1}{n_1}\sum_{i\,:\,y_i=1}\mathbf{1}_{[S(X_i)-\theta\le t]}\right),$$

$$\widehat{\mathrm{P@k}}_\theta(S) := 1 - \frac{1}{n}\sum_{i=1}^{n}\mathbf{1}_{[y_i=1,S(X_i)-\theta\ge\widehat{t}_\alpha(S)]},$$

with $\widehat{t}_\alpha(S)$ the sample $(1-\alpha)$-quantile.

Rademacher Complexity

Let $\{\sigma_i\}_{i=1}^n$ be iid uniform over ± 1 and independent from the data, define:

$$\mathcal{R}_n(\mathcal{H}) := \mathbb{E}_{\mathcal{D}}\left[\mathbb{E}_\sigma\left[\sup_{h\in\mathcal{H}} \frac{1}{n}\sum_{i=1}^n \sigma_i h(X_i)\right]\right],$$

$$\mathcal{R}_{n,y}(\mathcal{H}) := \mathbb{E}_{\mathcal{D}_y}\left[\mathbb{E}_\sigma\left[\sup_{h\in\mathcal{H}} \frac{1}{n_y}\sum_{i\,:\,y_i=y} \sigma_i h(X_i)\right]\right], \text{ for } y \in \{0,1\}.$$

Intuition: how well the set $\mathcal{H}$ correlates with random noise.

For the case of binary stumps, $\mathcal{R}_n(\mathcal{H}) = O(\sqrt{\log p/n})$.

Generalization Bound for AUC

Theorem

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* $\mathcal{H}$,

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

where

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

Generalization Bound for KS

Theorem

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 $\mathcal{H}$,

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

where

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

Generalization Bound for P@k

Theorem

Given $\delta \in (0,1)$*, and a class of functions* $\mathcal{H}$ *from* $\mathbb{R}^p$ *to* $[-1,1]$*, define*

$$\bar{\eta}_n(\mathcal{H}) := \sqrt{4\mathcal{R}_n(\mathcal{H}) + \frac{4}{\sqrt{n}}} + \sqrt{\frac{\log(3/\delta)}{n}},$$

Assume $\theta > 2\bar{\eta}_n(\mathcal{H})$*. Then, 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* $\mathcal{H}$*, it holds that*

$$\mathrm{P@k}(S) \leq \widehat{\mathrm{P@k}}_\theta(S) + \frac{4\mathcal{R}_{n_1,1}(\mathcal{H}) + 4/\sqrt{n_1}}{\theta - 2\bar{\eta}_n(\mathcal{H})} + \bar{\eta}_n(\mathcal{H}) + \sqrt{2\frac{\log(3/\delta)}{n_1}}.$$

Subsampling

Proposition

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 $h_\star$ the optimal stump over the entire sample $\{(X_i, y_i)\}_{i\in[n]}$. Then,

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

where the expectation is over the randomness in the choice of I.

Ensembling

Proposition

Consider the score $S_\star : \mathbb{R}^M \to \mathbb{R}$ obtained by ExactBoost over the dataset $(Z_i, y_i)_{i=1}^n$ with initial score $S_0 \equiv 0$. Then:

$$\widehat{L}_{(Z_i,y_i)_{i=1}^n}(S_*) \leq \min_{1\leq m\leq M} \widehat{L}_{(X_i,y_i)_{i=1}^n}(S_m),$$

where $\widehat{L}_{(Z_i,y_i)_{i=1}^n}(\cdot)$ and $\widehat{L}_{(X_i,y_i)_{i=1}^n}(\cdot)$ denote the loss over the ensemble and the original data.

Experimental Benchmarks

- Surrogate benchmarks:
  - AdaBoost;
  - *k*-nearest neighbors;
  - Logistic Regression;
  - Random Forest;
  - XGBoost (Gradient Boosting);
  - Neural Network (4-layer fully connected).

Experimental Benchmarks

- Surrogate benchmarks:
  - AdaBoost;
  - *k*-nearest neighbors;
  - Logistic Regression;
  - Random Forest;
  - XGBoost (Gradient Boosting);
  - Neural Network (4-layer fully connected).
- Exact benchmarks:
  - RankBoost (optimizes AUC);
  - DMKS (optimizes KS);
  - TopPush (optimizes $\mathrm{P@k}$).

Datasets

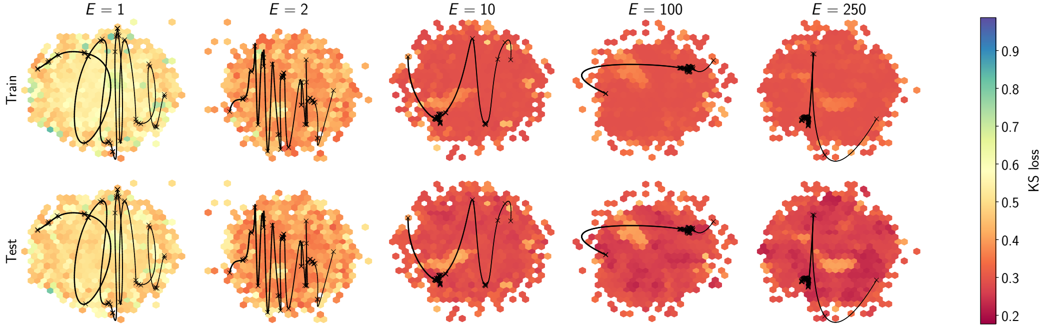
Dataset	Observations	Features	Positives	RankBoost	DMKS	TopPush	XGBoost
a1a	1605	119	24.61%	55.90x	102.78x	0.82x	0.38x
german	1000	20	70.0%	23.98x	1.28x	2.88x	0.28x
gisette	6000	5000	50.0%	OOT	55.68x	0.02x	0.34x
gmsc	150000	10	6.68%	OOT	22.89x	0.08x	0.54x
heart	303	21	45.87%	3.32x	19.00x	5.25x	0.28x
ionosphere	351	34	64.1%	3.97x	3.48x	2.69x	0.19x
liver-disorders	145	5	37.93%	1.91x	6.36x	12.53x	0.50x
oil-spill	937	49	4.38%	5.93x	7.92x	2.18x	0.28x
splice	1000	60	48.3%	49.78x	1.19x	1.20x	0.19x
svmguide1	3089	4	35.25%	220.05x	1.88x	4.27x	0.61x

Hyperparameters

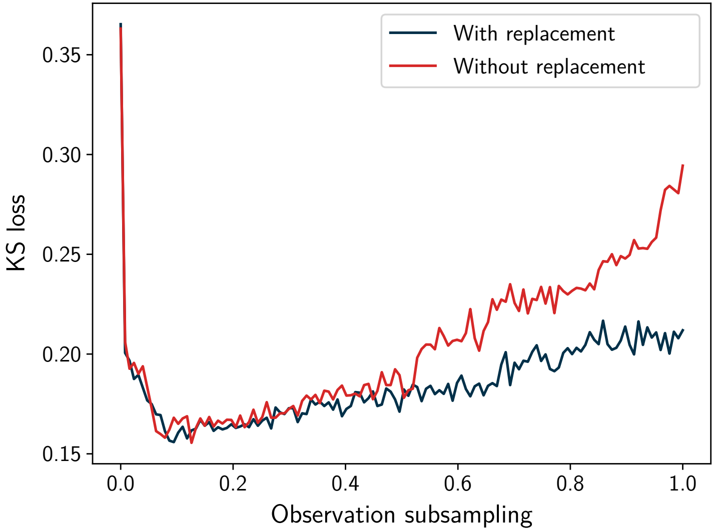
Chosen based on evidence from held-out datasets, hyperparameters were then fixed:

- Runs: 250;
- Subsampling of observations: 20%;
- Margin: 0.05;
- Rounds of boosting: 50.

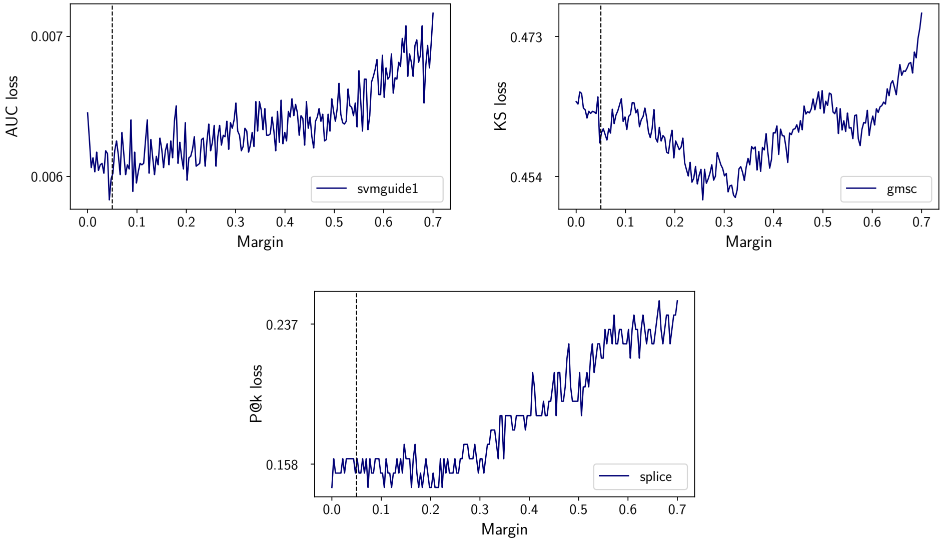
ExactBoost Hyperparameters — Run Averaging



ExactBoost Hyperparameters — Subsampling



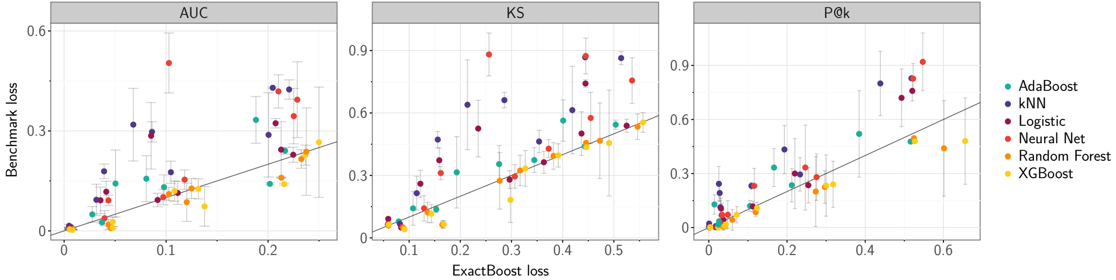
ExactBoost Hyperparameters — Margin



ExactBoost as an Estimator vs. Exact Benchmarks

Dataset	AUC ExactBoost	AUC RankBoost	KS ExactBoost	KS DMKS	P@k ExactBoost	P@k TopPush
a1a	**0.11 ± 0.0**	0.13 ± 0.0	**0.37 ± 0.0**	**0.37 ± 0.0**	**0.26 ± 0.1**	0.29 ± 0.1
german	**0.23 ± 0.0**	0.24 ± 0.0	**0.53 ± 0.0**	0.55 ± 0.0	**0.11 ± 0.0**	0.26 ± 0.1
gisette	**0.01 ± 0.0**	OOT	0.09 ± 0.0	**0.06 ± 0.0**	0.02 ± 0.0	**0.01 ± 0.0**
gmsc	**0.21 ± 0.0**	OOT	**0.44 ± 0.0**	0.45 ± 0.0	**0.52 ± 0.0**	0.96 ± 0.0
heart	**0.09 ± 0.0**	0.13 ± 0.0	0.30 ± 0.0	**0.28 ± 0.0**	**0.04 ± 0.1**	0.13 ± 0.1
iono.	**0.04 ± 0.0**	**0.04 ± 0.0**	**0.13 ± 0.0**	0.28 ± 0.0	**0.03 ± 0.0**	0.15 ± 0.1
liver	**0.22 ± 0.1**	0.32 ± 0.1	**0.45 ± 0.1**	0.50 ± 0.1	**0.23 ± 0.1**	0.47 ± 0.2
oil-spill	**0.09 ± 0.1**	**0.09 ± 0.1**	**0.25 ± 0.1**	0.45 ± 0.1	**0.52 ± 0.3**	0.96 ± 0.1
splice	0.04 ± 0.0	**0.02 ± 0.0**	**0.16 ± 0.0**	0.36 ± 0.0	**0.03 ± 0.0**	0.12 ± 0.0
svmguide1	0.01 ± 0.0	**0.00 ± 0.0**	**0.06 ± 0.0**	0.09 ± 0.0	**0.00 ± 0.0**	**0.00 ± 0.0**

ExactBoost as an Estimator vs. Surrogate Benchmarks



ExactBoost as an Ensembler

AUC

Dataset	ExactBoost	AdaBoost	Logistic	Neural Net	Rand. For.	XGBoost	Exact Bench.
a1a	**0.13 ± 0.0**	0.17 ± 0.0	0.14 ± 0.0	0.15 ± 0.0	0.27 ± 0.1	0.28 ± 0.1	0.16 ± 0.0
german	**0.23 ± 0.0**	0.32 ± 0.0	0.24 ± 0.0	0.50 ± 0.1	0.33 ± 0.0	0.35 ± 0.0	0.30 ± 0.1
gisette	**0.00 ± 0.0**	0.01 ± 0.0	0.01 ± 0.0	0.01 ± 0.0	0.03 ± 0.0	0.02 ± 0.0	0.01 ± 0.0
gmsc	0.15 ± 0.0	**0.14 ± 0.0**	0.31 ± 0.0	0.46 ± 0.0	0.42 ± 0.0	0.41 ± 0.0	0.15 ± 0.0
heart	**0.12 ± 0.0**	0.18 ± 0.1	**0.12 ± 0.0**	0.23 ± 0.1	0.19 ± 0.0	0.23 ± 0.1	0.15 ± 0.0
iono.	**0.04 ± 0.0**	0.05 ± 0.0	0.07 ± 0.0	0.07 ± 0.0	0.07 ± 0.0	0.09 ± 0.0	0.05 ± 0.0
liver	**0.30 ± 0.1**	0.34 ± 0.1	0.34 ± 0.1	0.34 ± 0.1	0.38 ± 0.0	0.38 ± 0.0	0.38 ± 0.1
oil-spill	**0.17 ± 0.1**	0.19 ± 0.1	0.29 ± 0.2	0.46 ± 0.1	0.38 ± 0.1	0.35 ± 0.2	0.19 ± 0.1
splice	**0.01 ± 0.0**	**0.01 ± 0.0**	0.08 ± 0.0	0.05 ± 0.0	0.04 ± 0.0	0.04 ± 0.0	0.02 ± 0.0
svmg1	**0.00 ± 0.0**	0.01 ± 0.0	0.01 ± 0.0	0.01 ± 0.0	0.03 ± 0.0	0.04 ± 0.0	0.01 ± 0.0

ExactBoost as an Ensembler

KS

Dataset	ExactBoost	AdaBoost	Logistic	Neural Net	Rand. For.	XGBoost	Exact Bench.
a1a	$\mathbf{0.37 \pm 0.1}$	0.44 ± 0.1	0.40 ± 0.1	0.41 ± 0.1	0.54 ± 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	0.89 ± 0.1	0.66 ± 0.0	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	0.07 ± 0.0	0.06 ± 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	0.95 ± 0.0	0.85 ± 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	0.52 ± 0.1	0.38 ± 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	0.17 ± 0.1	0.15 ± 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	0.61 ± 0.1	0.76 ± 0.1	0.76 ± 0.0	0.60 ± 0.2
oil-spill	$\mathbf{0.33 \pm 0.2}$	$\mathbf{0.33 \pm 0.2}$	0.47 ± 0.2	0.89 ± 0.1	0.76 ± 0.2	0.69 ± 0.3	0.63 ± 0.3
splice	$\mathbf{0.06 \pm 0.0}$	0.09 ± 0.0	0.28 ± 0.0	0.21 ± 0.0	0.09 ± 0.0	0.09 ± 0.0	0.28 ± 0.0
svmg1	$\mathbf{0.06 \pm 0.0}$	0.08 ± 0.0	$\mathbf{0.06 \pm 0.0}$	$\mathbf{0.06 \pm 0.0}$	0.07 ± 0.0	0.07 ± 0.0	$\mathbf{0.06 \pm 0.0}$

ExactBoost as an Ensembler

P@k

Dataset	ExactBoost	AdaBoost	Logistic	Neural Net	Rand. For.	XGBoost	Exact Bench.
a1a	**0.22 ± 0.1**	0.34 ± 0.1	0.28 ± 0.1	0.32 ± 0.1	0.34 ± 0.2	0.40 ± 0.1	0.29 ± 0.1
german	**0.13 ± 0.0**	0.16 ± 0.1	**0.13 ± 0.0**	0.33 ± 0.0	0.20 ± 0.0	0.21 ± 0.1	0.18 ± 0.0
gisette	0.01 ± 0.0	0.01 ± 0.0	**0.00 ± 0.0**	**0.00 ± 0.0**	0.02 ± 0.0	0.02 ± 0.0	0.01 ± 0.0
gmsc	0.51 ± 0.0	**0.48 ± 0.0**	0.74 ± 0.1	0.88 ± 0.0	0.65 ± 0.1	0.62 ± 0.0	0.96 ± 0.0
heart	0.07 ± 0.1	0.19 ± 0.1	**0.06 ± 0.0**	0.19 ± 0.1	0.23 ± 0.1	0.29 ± 0.1	0.14 ± 0.2
iono.	**0.03 ± 0.0**	0.04 ± 0.1	0.05 ± 0.0	0.06 ± 0.1	0.09 ± 0.1	0.10 ± 0.1	0.10 ± 0.1
liver	**0.27 ± 0.2**	0.33 ± 0.2	0.33 ± 0.2	0.40 ± 0.3	0.40 ± 0.2	0.33 ± 0.2	0.30 ± 0.2
oil-spill	**0.44 ± 0.2**	0.72 ± 0.2	0.84 ± 0.2	0.92 ± 0.1	0.72 ± 0.2	0.68 ± 0.3	0.68 ± 0.2
splice	**0.01 ± 0.0**	**0.01 ± 0.0**	s0.04 ± 0.0	0.04 ± 0.0	0.05 ± 0.0	0.05 ± 0.0	0.05 ± 0.0
svmg1	**0.00 ± 0.0**	0.01 ± 0.0	**0.00 ± 0.0**	0.01 ± 0.0	0.05 ± 0.0	0.05 ± 0.0	**0.00 ± 0.0**

Conclusion

- ExactBoost is a competitive estimator and an even better ensembler;
- There is value to be gained in working with the intended loss function;
- Novel theoretical results bound the generalization error for AUC, KS and $\mathrm{P@k}$;
- Computational implementations can be made fast through interval arithmetic;
- Paper and source code to be released.

Thank you!