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

1. Introduction

2. The Algorithm

3. Theoretical Results

4. Experimental Results

5. Conclusion
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 \(S : \mathbb{R}^p \to [-1, 1]\).

But our approach works for many other metrics as well.
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 \(S : \mathbb{R}^p \rightarrow [-1, 1]\).

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

\[ \hat{\text{AUC}}(S, y) := 1 - \frac{1}{n_1} \sum_{y_i=1} \frac{1}{n_0} \sum_{y_j=0} 1[S(X_i) > S(X_j)], \]

\[ \hat{\text{KS}}(S, y) := 1 - \max_{t \in \mathbb{R}} \sum_{i=1}^n \rho_i 1[S(X_i) \leq t], \]

\[ \hat{\text{P}@k}(S, y) := 1 - \frac{1}{n} \sum_{i=1}^n y_i 1[i \in M_k], \]

where \( \rho_i = 1/n_0 \) if \( y_i = 0 \) and \( \rho_i = -1/n_1 \) if \( y_i = 1 \), and \( M_k \) denotes the set of indices \( i = 1, \ldots, n \) achieving the highest \( k \) scores.
A combinatorial loss function is one that is computed in terms of indicator functions. A decomposable function is one where:

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

The metrics presented are non-decomposable. This means that common approaches, such as Convex Optimization and Stochastic Gradient Descent, can't be readily applied!
A combinatorial loss function is one that is computed in terms of indicator functions.

The metrics presented are non-decomposable. This means that common approaches, such as Convex Optimization and Stochastic Gradient Descent, can't be readily applied!
A combinatorial loss function is one that is computed in terms of indicator functions.

A decomposable function is one where

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

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

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

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

Boosting

\[ \alpha_1 + \alpha_2 + \alpha_3 = \]

decision boundary

Our Boosting Framework

\[ S = S_0 + \sum_i \alpha_i h_i(X), \quad \sum_i \alpha_i = 1 \]

Where

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

\[ S = S_0 + \sum_i \alpha_i h_i(X), \quad \sum_i \alpha_i = 1 \]

Where

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

Why stumps?
Our Boosting Framework

\[ S = S_0 + \sum_i \alpha_i h_i(X), \quad \sum_i \alpha_i = 1 \]

Where

\[ h_i \in \mathcal{H} = \left\{ \pm 1_{x_{ij} \leq \xi} \pm 1_{x_{ij} > \xi} : \xi \in \mathbb{R}, \ j \in [p] \right\}. \]

Why stumps?

- Small complexity;
Our Boosting Framework

\[ S = S_0 + \sum_i \alpha_i h_i(X), \quad \sum_i \alpha_i = 1 \]

Where

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

Why stumps?

- Small complexity;
- Fast to compute;
**Our Boosting Framework**

\[ S = S_0 + \sum_i \alpha_i h_i(X), \quad \sum_i \alpha_i = 1 \]

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

\[ S = S_0 + \sum_i \alpha_i h_i(X), \quad \sum_i \alpha_i = 1 \]

Where

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

Why stumps?

- Small complexity;
- Fast to compute;
- Fast to optimize;
- Simplicity helps in preventing overfitting.
Optimizing the Weak Learners

\[ (\alpha_t, h_t) = \arg\min_{\alpha, h} \hat{L}_\theta \left( \frac{1}{1 + \alpha} S_{t-1} + \frac{\alpha}{1 + \alpha} h(X), y \right) \]
Optimizing the Weak Learners

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

\[
= \arg \min_{\alpha, h} \hat{L} \left( \frac{1}{1 + \alpha} S_{t-1} + \frac{\alpha}{1 + \alpha} h(X) - \theta y, y \right)
\]
Optimizing the Weak Learners

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

\[= \arg \min_{\alpha, h, \hat{L}} \left( \frac{1}{1 + \alpha} S_{t-1} + \frac{\alpha}{1 + \alpha} h(X) - \theta y, y \right) \]

\[= \arg \min_{\alpha, h, \hat{L}} \left( S_{t-1} - \theta y + \alpha (h(X) - \theta y), y \right) \]
Optimizing the Weak Learners

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

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

\[
= \arg \min_{\alpha, h} \widehat{L}(S_{t-1} - \theta y + \alpha(h(X) - \theta y), y)
\]

\[
= \arg \min_{\alpha, h} \widehat{L} \left( S_{t-1} + a \mathbb{1}_{\{X(j) \leq \xi\}} + b \mathbb{1}_{\{X(j) > \xi\}} - \left(1 - \frac{|b - a|}{2}\right) \theta y, y \right)
\]
Optimizing the Weak Learners

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

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

\[= \arg\min_{\alpha, h} \widehat{L} (S_{t-1} - \theta y + \alpha (h(X) - \theta y), y)\]

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

\[= \arg\min_{\alpha, h} \widehat{L} \left( S_{t-1} + a \mathbf{1}_{X(j) \leq \xi} + b \mathbf{1}_{X(j) > \xi} - M(\theta, y), y \right)\]
Optimizing the Weak Learners

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

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

$$= \arg \min_{\alpha, h} \hat{L}(S_{t-1} - \theta y + \alpha(h(X) - \theta y), y)$$

$$= \arg \min_{\alpha, h} \hat{L} \left( S_{t-1} + a \mathbf{1}_{[X(j) \leq \xi]} + b \mathbf{1}_{[X(j) > \xi]} - \left( 1 - \frac{|b-a|}{2} \right) \theta y, y \right)$$

$$= \arg \min_{\alpha, h} \hat{L} \left( S_{t-1} + a \mathbf{1}_{[X(j) \leq \xi]} + b \mathbf{1}_{[X(j) > \xi]} - M(\theta, y), y \right)$$

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** `ExactBoost`(data \((X, y)\), initial score \(S_0\))

\[
S ← S_0
\]

for \(t ∈ \{1, \ldots, T\}\) do

for \(j ∈ \{1, \ldots, p\}\) do

\[(ξ, a, b) ← \arg \min_{ξ, a, b} \hat{L}(S + a1_{X(j) ≤ ξ} + b1_{X(j) > ξ} − M(θ, y), y)\]

\[h_j ← a1_{X(j) ≤ ξ} + b1_{X(j) > ξ}\]

\[S' ← S + \arg \min_{h_j} \hat{L}(S + h_j, y)\]

if \(\hat{L}(S', y) ≤ \hat{L}(S, y)\) then

\[S ← S'\]

return \(S\)
Subsampling

function $\text{ExactBoost}(\text{data } (X, y), \text{initial score } S_0)$

$S \leftarrow S_0$

for $t \in \{1, \ldots, T\}$ do

$X^s, y^s \leftarrow$ subsample $X, y$

for $j \in \{1, \ldots, p\}$ do

$(\xi, a, b) \leftarrow \arg \min_{\xi, a, b} \hat{L}(S + a1_{X^s(j) \leq \xi} + b1_{X^s(j) > \xi} - M(\theta, y), y^s)$

$h_j \leftarrow a1_{X^s(j) \leq \xi} + b1_{X^s(j) > \xi}$

$S' \leftarrow S + \arg \min_{h_j} \hat{L}(S + h_j, y^s)$

if $\hat{L}(S', y) \leq \hat{L}(S, y)$ then

$S \leftarrow S'$

return $S$
Run Averaging

function 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\(_{\xi, a, b}\) \(L(S_e + a1[X^s_{(j)} \leq \xi] + b1[X^s_{(j)} > \xi] - M(\theta, y), y^s)\)

\(h_j \leftarrow a1[X^s_{(j)} \leq \xi] + b1[X^s_{(j)} > \xi]\)

\(S'_e \leftarrow S_e + \text{arg min}_{h_j} \hat{L}(S_e + h_j, y^s)\)

if \(\hat{L}(S'_e, y) \leq \hat{L}(S_e, y)\) then

\(S_e \leftarrow S'_e\)

return mean\((S_1, \ldots, S_E)\)
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 $k \in \{1, \ldots, c\}$ do
        for bisections $\Xi', A', B'$ do
            $\ell_{\Xi', A', B'} \leftarrow$ estimate best metric in this subdomain
            $(\Xi, A, B) \leftarrow \arg\min_{(\Xi', A', B')} \ell_{\Xi', A', B'}$
            $(\xi^*, a^*, b^*) \leftarrow \arg\min_{\xi \in \Xi, a \in A, b \in B} \hat{L}(S + a 1_{X(j) \leq \xi} + b 1_{X(j) > \xi}, y)$
    return $(\xi^*, a^*, b^*)$
Interval Arithmetic

\[ X + Y = [X, X] + [Y, Y] = [X + Y, X + Y] \]
\[ X - Y = [X, X] - [Y, Y] = [X - Y, X - Y] \]
Interval Arithmetic

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

where \( M = \{X \cdot Y, X \cdot \overline{Y}, \overline{X} \cdot Y, \overline{X} \cdot \overline{Y}\} \)
Interval Arithmetic

\[ X + Y = [X, X] + [Y, Y] = [X + Y, X + Y] \]
\[ X - Y = [X, X] - [Y, Y] = [X - Y, X - Y] \]
\[ X \cdot Y = [X, X] \cdot [Y, Y] = [\min M, \max M] \]

where \( M = \{X \cdot Y, X \cdot Y, X \cdot Y, X \cdot Y\} \)

\[ f(X) = f([X, X]) = [f(X), f(X)], \quad f \text{ increasing} \]
\[ f(X) = f([X, X]) = [f(X), f(X)], \quad f \text{ decreasing} \]
Interval Arithmetic

\[ X + Y = [X, X] + [Y, Y] = [X + Y, X + Y] \]
\[ X - Y = [X, X] - [Y, Y] = [X - Y, X - Y] \]
\[ X \cdot Y = [X, X] \cdot [Y, Y] = [\min M, \max M] \]

where \( M = \{ X \cdot Y, X \cdot Y, X \cdot Y, X \cdot Y \} \)

\[ f(X) = f([X, X]) = [f(X), f(X)], \ \text{f increasing} \]
\[ f(X) = f([X, X]) = [f(X), f(X)], \ \text{f decreasing} \]

\[ H(X) = H([X, X]) = [H(X), H(X)] \]
\[ 1_{[X < Y]} = 1_{[X, X] < [Y, Y]} = [X < Y, X < Y] \]
Interval Arithmetic, Applied to our Losses

Overall, we have

\[
\hat{L}(S, y) = \hat{L}(Sy + S(1 - y), y)
\]

\[
\hat{L}(S, y) = \hat{L}(Sy + \overline{S}(1 - y), y)
\]

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 $k \in \{1, \ldots, c\}$ do

for bisections $\Xi', A', B'$ do

$S' \leftarrow S + A'1[X(j) \leq \Xi'] + B'1[X(j) > \Xi']$

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

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

$(\xi_*, a_*, b_*) \leftarrow \arg \min_{\xi \in \{\Xi, \Xi\}, a \in \{A, \bar{A}\}, b \in \{B, \bar{B}\}} \hat{L}(S + a1[X(j) \leq \xi] + b1[X(j) > \xi], y)$

return $(\xi_*, a_*, b_*)$
Two Ways to Use ExactBoost
Two Ways to Use ExactBoost

As an estimator

<table>
<thead>
<tr>
<th>Feature 1</th>
<th>Feature 2</th>
<th>...</th>
<th>Feature $p - 1$</th>
<th>Feature $p$</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2</td>
<td>100</td>
<td>...</td>
<td>10</td>
<td>22.3</td>
<td>0</td>
</tr>
<tr>
<td>1.1</td>
<td>27</td>
<td>...</td>
<td>5200</td>
<td>22.1</td>
<td>1</td>
</tr>
<tr>
<td>0.7</td>
<td>0</td>
<td>...</td>
<td>3201</td>
<td>22.3</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>:</td>
</tr>
<tr>
<td>0.6</td>
<td>45</td>
<td>...</td>
<td>614</td>
<td>22.3</td>
<td>0</td>
</tr>
<tr>
<td>0.9</td>
<td>-21</td>
<td>...</td>
<td>1023</td>
<td>22.3</td>
<td>1</td>
</tr>
</tbody>
</table>
Two Ways to Use ExactBoost

As an ensembler

<table>
<thead>
<tr>
<th>Logistic Regression</th>
<th>$k$-NN</th>
<th>...</th>
<th>AdaBoost</th>
<th>Random Forest</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>0.2</td>
<td>...</td>
<td>0.1</td>
<td>0.3</td>
<td>0</td>
</tr>
<tr>
<td>0.4</td>
<td>0.8</td>
<td>...</td>
<td>0.7</td>
<td>0.9</td>
<td>1</td>
</tr>
<tr>
<td>0.2</td>
<td>0.4</td>
<td>...</td>
<td>0.3</td>
<td>0.2</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>0.3</td>
<td>0.6</td>
<td>...</td>
<td>0.1</td>
<td>0.2</td>
<td>0</td>
</tr>
<tr>
<td>0.7</td>
<td>0.9</td>
<td>...</td>
<td>0.6</td>
<td>0.8</td>
<td>1</td>
</tr>
</tbody>
</table>
Empirical and Generalization Errors

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

- Generalization Error, not directly accessible, is the true error:
  $$L(S) = E_D[L(S, y)].$$

Classical results manage to bound the generalization error of decomposable losses.
Empirical and Generalization Errors

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

- Generalization Error, not directly accessible, is the true error:

$$L(S) = \mathbb{E}_{\mathcal{D}}[\hat{L}(S, y)].$$
Empirical and Generalization Errors

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

- Generalization Error, not directly accessible, is the true error:

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

- Classical results manage to bound the generalization error of decomposable losses.
AUC, KS and P@$\alpha$, Revisited

Population Losses

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

$$AUC(S) := 1 - \Pr\{S(X) > S(X')\},$$
$$KS(S) := 1 - \sup_{t \in \mathbb{R}} (\Pr\{S(X') \leq t\} - \Pr\{S(X) \leq t\}),$$
$$P@k_\alpha(S) := 1 - \Pr\{y = 1, S(X) \geq t_\alpha(S)\},$$

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

$$t_\alpha(S) := \inf \{ t \in \mathbb{R} : \Pr\{S(X) \leq t\} \geq 1 - \alpha \}.$$
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 $\theta$-margin-adjusted versions of the empirical losses:

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

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

\[
\widehat{\text{P@k}}_{\theta}(S) := 1 - \frac{1}{n} \sum_{i=1}^n \mathbb{1}_{[y_i = 1, S(X_i) - \theta \geq \hat{t}_\alpha(S)]},
\]

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

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

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

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

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

For the case of binary stumps, $R_n(\mathcal{H}) = O(\sqrt{\log p/n})$. 
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 $S : \mathbb{R}^p \rightarrow [-1, 1]$ obtained as convex combinations of the elements of $\mathcal{H}$,

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

where

$$\zeta_{AUC}(\mathcal{H}) = R_{\min\{n_0, n_1\}, 0}(\mathcal{H}) + R_{\min\{n_0, n_1\}, 1}(\mathcal{H}).$$
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 $S : \mathbb{R}^p \rightarrow [-1, 1]$ obtained as convex combinations of the elements of $\mathcal{H}$,

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

where

$$
\zeta_{\text{KS}}(\mathcal{H}) = \mathcal{R}_{n_0,0}(\mathcal{H}) + \mathcal{R}_{n_1,1}(\mathcal{H}) + n_0^{-1/2} + n_1^{-1/2}.
$$
Generalization Bound for $\mathcal{P}\mathcal{O}_k$

Theorem

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

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

Assume $\theta > 2\tilde{\eta}_n(\mathcal{H})$. 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

$$\mathcal{P}\mathcal{O}_k(S) \leq \mathcal{P}\mathcal{O}_k(\theta)(S) + \frac{4R_{n_1,1}(\mathcal{H}) + 4/\sqrt{n_1}}{\theta - 2\tilde{\eta}_n(\mathcal{H})} + \tilde{\eta}_n(\mathcal{H}) + \sqrt{2\frac{\log(3/\delta)}{n_1}}.$$
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,

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

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

Proposition

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

\[
\hat{L}(Z_i, y_i)_{i=1}^n (S_\star) \leq \min_{1 \leq m \leq M} \hat{L}(X_i, y_i)_{i=1}^n (S_m),
\]

where \( \hat{L}(Z_i, y_i)_{i=1}^n (\cdot) \) and \( \hat{L}(X_i, y_i)_{i=1}^n (\cdot) \) 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 P@k).
## Datasets

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Observations</th>
<th>Features</th>
<th>Positives</th>
<th>RankBoost</th>
<th>DMKS</th>
<th>TopPush</th>
<th>XGBoost</th>
</tr>
</thead>
<tbody>
<tr>
<td>a1a</td>
<td>1605</td>
<td>119</td>
<td>24.61%</td>
<td>55.90x</td>
<td>102.78x</td>
<td>0.82x</td>
<td>0.38x</td>
</tr>
<tr>
<td>german</td>
<td>1000</td>
<td>20</td>
<td>70.0%</td>
<td>23.98x</td>
<td>1.28x</td>
<td>2.88x</td>
<td>0.28x</td>
</tr>
<tr>
<td>gisette</td>
<td>6000</td>
<td>5000</td>
<td>50.0%</td>
<td>OOT</td>
<td>55.68x</td>
<td>0.02x</td>
<td>0.34x</td>
</tr>
<tr>
<td>gmsc</td>
<td>150000</td>
<td>10</td>
<td>6.68%</td>
<td>OOT</td>
<td>22.89x</td>
<td>0.06x</td>
<td>0.54x</td>
</tr>
<tr>
<td>heart</td>
<td>303</td>
<td>21</td>
<td>45.87%</td>
<td>3.32x</td>
<td>19.00x</td>
<td>5.25x</td>
<td>0.28x</td>
</tr>
<tr>
<td>ionosphere</td>
<td>351</td>
<td>34</td>
<td>64.1%</td>
<td>3.97x</td>
<td>3.48x</td>
<td>2.69x</td>
<td>0.19x</td>
</tr>
<tr>
<td>liver-disorders</td>
<td>145</td>
<td>5</td>
<td>37.93%</td>
<td>1.91x</td>
<td>6.36x</td>
<td>12.53x</td>
<td>0.50x</td>
</tr>
<tr>
<td>oil-spill</td>
<td>937</td>
<td>49</td>
<td>4.38%</td>
<td>5.93x</td>
<td>7.92x</td>
<td>2.18x</td>
<td>0.28x</td>
</tr>
<tr>
<td>splice</td>
<td>1000</td>
<td>60</td>
<td>48.3%</td>
<td>49.78x</td>
<td>1.19x</td>
<td>1.20x</td>
<td>0.19x</td>
</tr>
<tr>
<td>svmguide1</td>
<td>3089</td>
<td>4</td>
<td>35.25%</td>
<td>220.05x</td>
<td>1.88x</td>
<td>4.27x</td>
<td>0.61x</td>
</tr>
</tbody>
</table>
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

Train

Test

KS loss

0.2

0.3

0.4

0.5

0.6

0.7

0.8

0.9

E = 1

E = 2

E = 10

E = 100

E = 250
ExactBoost Hyperparameters — Subsampling

Observation subsampling with and without replacement.
ExactBoost Hyperparameters — Margin

AUC loss

KS loss

P@k loss

33 / 39
## ExactBoost as an Estimator vs. Exact Benchmarks

<table>
<thead>
<tr>
<th>Dataset</th>
<th>AUC</th>
<th>KS</th>
<th>P@k</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ExactBoost</td>
<td>RankBoost</td>
<td>ExactBoost</td>
</tr>
<tr>
<td>a1a</td>
<td>0.11 ± 0.0</td>
<td>0.13 ± 0.0</td>
<td>0.37 ± 0.0</td>
</tr>
<tr>
<td>german</td>
<td>0.23 ± 0.0</td>
<td>0.24 ± 0.0</td>
<td>0.53 ± 0.0</td>
</tr>
<tr>
<td>gisette</td>
<td>0.01 ± 0.0</td>
<td>OOT</td>
<td>0.09 ± 0.0</td>
</tr>
<tr>
<td>gmsc</td>
<td>0.21 ± 0.0</td>
<td>OOT</td>
<td>0.44 ± 0.0</td>
</tr>
<tr>
<td>heart</td>
<td>0.09 ± 0.0</td>
<td>0.13 ± 0.0</td>
<td>0.30 ± 0.0</td>
</tr>
<tr>
<td>iono.</td>
<td>0.04 ± 0.0</td>
<td>0.04 ± 0.0</td>
<td>0.13 ± 0.0</td>
</tr>
<tr>
<td>liver</td>
<td>0.22 ± 0.1</td>
<td>0.32 ± 0.1</td>
<td>0.45 ± 0.1</td>
</tr>
<tr>
<td>oil-spill</td>
<td>0.09 ± 0.1</td>
<td>0.09 ± 0.1</td>
<td>0.25 ± 0.1</td>
</tr>
<tr>
<td>splice</td>
<td>0.04 ± 0.0</td>
<td>0.02 ± 0.0</td>
<td>0.16 ± 0.0</td>
</tr>
<tr>
<td>svmguide1</td>
<td>0.01 ± 0.0</td>
<td>0.00 ± 0.0</td>
<td>0.06 ± 0.0</td>
</tr>
</tbody>
</table>
ExactBoost as an Estimator vs. Surrogate Benchmarks

AUC

KS

P@k

ExactBoost loss

Benchmark loss

0
0.1
0.2

0.3
0.4
0.5

0
0.1
0.2
0.3
0.4
0.5

0
0.3
0.6
0.9

0
0.3
0.6
0.9

0
0.2
0.4
0.6

0
0.2
0.4
0.6

AdaBoost
kNN
Logistic
Neural Net
Random Forest
XGBoost
## ExactBoost as an Ensembler

### AUC

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

<table>
<thead>
<tr>
<th>Dataset</th>
<th>ExactBoost</th>
<th>AdaBoost</th>
<th>Logistic</th>
<th>Neural Net</th>
<th>Rand. For.</th>
<th>XGBoost</th>
<th>Exact Bench.</th>
</tr>
</thead>
<tbody>
<tr>
<td>a1a</td>
<td>0.37 ± 0.1</td>
<td>0.44 ± 0.1</td>
<td>0.40 ± 0.1</td>
<td>0.41 ± 0.1</td>
<td>0.54 ± 0.1</td>
<td>0.57 ± 0.1</td>
<td>0.49 ± 0.1</td>
</tr>
<tr>
<td>german</td>
<td>0.50 ± 0.1</td>
<td>0.68 ± 0.1</td>
<td>0.53 ± 0.1</td>
<td>0.89 ± 0.1</td>
<td>0.66 ± 0.0</td>
<td>0.69 ± 0.1</td>
<td>0.53 ± 0.1</td>
</tr>
<tr>
<td>gisette</td>
<td>0.04 ± 0.0</td>
<td>0.04 ± 0.0</td>
<td>0.07 ± 0.0</td>
<td>0.07 ± 0.0</td>
<td>0.06 ± 0.0</td>
<td>0.04 ± 0.0</td>
<td>0.10 ± 0.0</td>
</tr>
<tr>
<td>gmsc</td>
<td>0.43 ± 0.0</td>
<td>0.44 ± 0.0</td>
<td>0.73 ± 0.0</td>
<td>0.95 ± 0.0</td>
<td>0.85 ± 0.0</td>
<td>0.83 ± 0.0</td>
<td>0.46 ± 0.0</td>
</tr>
<tr>
<td>heart</td>
<td>0.34 ± 0.1</td>
<td>0.38 ± 0.1</td>
<td>0.37 ± 0.1</td>
<td>0.52 ± 0.1</td>
<td>0.38 ± 0.1</td>
<td>0.46 ± 0.1</td>
<td>0.40 ± 0.0</td>
</tr>
<tr>
<td>iono.</td>
<td>0.13 ± 0.1</td>
<td>0.18 ± 0.1</td>
<td>0.18 ± 0.1</td>
<td>0.17 ± 0.1</td>
<td>0.15 ± 0.1</td>
<td>0.19 ± 0.1</td>
<td>0.27 ± 0.1</td>
</tr>
<tr>
<td>liver</td>
<td>0.53 ± 0.1</td>
<td>0.60 ± 0.2</td>
<td>0.59 ± 0.2</td>
<td>0.61 ± 0.1</td>
<td>0.76 ± 0.1</td>
<td>0.76 ± 0.0</td>
<td>0.60 ± 0.2</td>
</tr>
<tr>
<td>oil-spill</td>
<td>0.33 ± 0.2</td>
<td>0.33 ± 0.2</td>
<td>0.47 ± 0.2</td>
<td>0.89 ± 0.1</td>
<td>0.76 ± 0.2</td>
<td>0.69 ± 0.3</td>
<td>0.63 ± 0.3</td>
</tr>
<tr>
<td>splice</td>
<td>0.06 ± 0.0</td>
<td>0.09 ± 0.0</td>
<td>0.28 ± 0.0</td>
<td>0.21 ± 0.0</td>
<td>0.09 ± 0.0</td>
<td>0.09 ± 0.0</td>
<td>0.28 ± 0.0</td>
</tr>
<tr>
<td>svmg1</td>
<td>0.06 ± 0.0</td>
<td>0.08 ± 0.0</td>
<td>0.06 ± 0.0</td>
<td>0.06 ± 0.0</td>
<td>0.07 ± 0.0</td>
<td>0.07 ± 0.0</td>
<td>0.06 ± 0.0</td>
</tr>
</tbody>
</table>
## ExactBoost as an Ensembler

**P@k**

<table>
<thead>
<tr>
<th>Dataset</th>
<th>ExactBoost</th>
<th>AdaBoost</th>
<th>Logistic</th>
<th>Neural Net</th>
<th>Rand. For.</th>
<th>XGBoost</th>
<th>Exact Bench.</th>
</tr>
</thead>
<tbody>
<tr>
<td>a1a</td>
<td>0.22 ± 0.1</td>
<td>0.34 ± 0.1</td>
<td>0.28 ± 0.1</td>
<td>0.32 ± 0.1</td>
<td>0.34 ± 0.2</td>
<td>0.40 ± 0.1</td>
<td>0.29 ± 0.1</td>
</tr>
<tr>
<td>german</td>
<td>0.13 ± 0.0</td>
<td>0.16 ± 0.1</td>
<td><strong>0.13 ± 0.0</strong></td>
<td>0.33 ± 0.0</td>
<td>0.20 ± 0.0</td>
<td>0.21 ± 0.1</td>
<td>0.18 ± 0.0</td>
</tr>
<tr>
<td>gisette</td>
<td>0.01 ± 0.0</td>
<td>0.01 ± 0.0</td>
<td><strong>0.00 ± 0.0</strong></td>
<td><strong>0.00 ± 0.0</strong></td>
<td>0.02 ± 0.0</td>
<td>0.02 ± 0.0</td>
<td>0.01 ± 0.0</td>
</tr>
<tr>
<td>gmsc</td>
<td>0.51 ± 0.0</td>
<td><strong>0.48 ± 0.0</strong></td>
<td>0.74 ± 0.1</td>
<td>0.88 ± 0.0</td>
<td>0.65 ± 0.1</td>
<td>0.62 ± 0.0</td>
<td>0.96 ± 0.0</td>
</tr>
<tr>
<td>heart</td>
<td>0.07 ± 0.1</td>
<td>0.19 ± 0.1</td>
<td><strong>0.06 ± 0.0</strong></td>
<td>0.19 ± 0.1</td>
<td>0.23 ± 0.1</td>
<td>0.29 ± 0.1</td>
<td>0.14 ± 0.2</td>
</tr>
<tr>
<td>iono.</td>
<td>0.03 ± 0.0</td>
<td>0.04 ± 0.1</td>
<td>0.05 ± 0.0</td>
<td>0.06 ± 0.1</td>
<td>0.09 ± 0.1</td>
<td>0.10 ± 0.1</td>
<td>0.10 ± 0.1</td>
</tr>
<tr>
<td>liver</td>
<td>0.27 ± 0.2</td>
<td>0.33 ± 0.2</td>
<td>0.33 ± 0.2</td>
<td>0.40 ± 0.3</td>
<td>0.40 ± 0.2</td>
<td>0.33 ± 0.2</td>
<td>0.30 ± 0.2</td>
</tr>
<tr>
<td>oil-spill</td>
<td>0.44 ± 0.2</td>
<td>0.72 ± 0.2</td>
<td>0.84 ± 0.2</td>
<td>0.92 ± 0.1</td>
<td>0.72 ± 0.2</td>
<td>0.68 ± 0.3</td>
<td>0.68 ± 0.2</td>
</tr>
<tr>
<td>splice</td>
<td>0.01 ± 0.0</td>
<td><strong>0.01 ± 0.0</strong></td>
<td>0.04 ± 0.0</td>
<td>0.04 ± 0.0</td>
<td>0.05 ± 0.0</td>
<td>0.05 ± 0.0</td>
<td>0.05 ± 0.0</td>
</tr>
<tr>
<td>svmg1</td>
<td><strong>0.00 ± 0.0</strong></td>
<td>0.01 ± 0.0</td>
<td><strong>0.00 ± 0.0</strong></td>
<td>0.01 ± 0.0</td>
<td>0.05 ± 0.0</td>
<td>0.05 ± 0.0</td>
<td><strong>0.00 ± 0.0</strong></td>
</tr>
</tbody>
</table>
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 P@k;
- Computational implementations can be made fast through interval arithmetic;
- Paper and source code to be released.
Thank you!