Gradient Boosting versus Mixed Integer Programming for Sparse Additive Modeling

Fan Yang[1](✉)[0009−0009−5291−7594], Pierre Le Bodic[1][0000−0003−0842−9533] and Mario Boley[2,1][0000−0002−0704−4968]

[1] Department of Data Science and AI, Monash University, Clayton, VIC, 3800, Australia {fan.yang1,pierre.lebodic,mario.boley}@monash.edu
[2] Department of Information Systems, University of Haifa, Haifa, 3498838, Israel mboley@is.haifa.ac.il

Abstract. Gradient boosting is a widely used algorithm for fitting sparse additive models over flexible classes of basis functions. Despite its popularity, the performance of gradient boosting as an approximation algorithm to the empirical risk minimizing model with a specific number k of selected basis functions is poorly understood. We provide a theoretical lower bound of $1/2 - 1/(4k-2)$ on the worst-case approximation ratio for the risk reduction that gradient boosting achieves relative to the optimal model when both are limited to k terms. This result reveals an inherent limitation in boosting's ability to approximate the best possible sparse additive model, raising the question of how tight and representative this bound is in practice. To empirically answer this question, we employ mixed integer programming (MIP) to approximate the optimal additive models on 21 real datasets. The experimental results do not show larger gaps than the theoretical analysis, indicating that the theoretical lower bound is tight. Moreover, for twelve datasets, the approximation gaps are of the same order of magnitude as the theoretical lower bound, which shows the representativeness of the theoretical bound. To that end, the study also has the practical implication that the presented MIP approach frequently offers notable improvements over gradient boosting.

Keywords: Gradient boosting · Mixed integer programming · Approximation gap · Additive model · Rule learning

1 Introduction

Gradient boosting is a widely used algorithm for training additive models [10], particularly in tasks that require a balance between predictive accuracy and model complexity. By iteratively fitting weak learners to residual errors, boosting builds powerful predictive models while maintaining a level of interpretability often lacking in more complex machine learning approaches [21,4,1,16]. However, a fundamental limitation arises due to its greedy nature: at each iteration, boosting selects the basis function that most reduces the empirical risk in the immediate step, rather than making globally optimal selections over all iterations [7]. This property leads to an approximation gap, defined as the relative difference between

the risk achieved by models generated by boosting and the theoretical optimum for a given model class and number of terms [16].

To better understand the approximation gap of greedy approaches like boosting, prior work has analyzed the performance of greedy algorithms relative to optimal models. For instance, according to [7], the ratio between the risk reductions of models generated by greedy approaches and the optimal model is lower bounded by $1 - \exp(-\lambda_{\min})$, where $\lambda_{\min}$ is the smallest eigenvalue of the covariance matrix of input variables, i.e. it shows an upper bound of the gap between risks of greedy approaches and optimal models. However, in general, this upper bound can be arbitrarily large, indicating that the potential for improvement over boosting for general inputs remains unknown.

In this work, we show that there exists datasets for which k-term models generated by boosting have an approximation gap of $1/2 - 1/(4k-2)$. Since there may be datasets for which boosting performs worse, we can state that the worst-case approximation gap of boosting is at least $1/2 - 1/(4k-2)$. This bound formalizes a fundamental limitation of boosting and quantifies its suboptimality. Furthermore, to evaluate the bound empirically, we employ Mixed Integer Programming (MIP) to attempt to improve the additive models generated by boosting on real-world datasets. Our experiments confirm that the observed approximation gaps align with the theoretical bound, indicating practical relevance.

A key goal of our study is to evaluate whether the theoretical bound on the boosting approximation gap established is both tight (i.e., accurately characterizing the worst-case gap) and representative (i.e., aligning with typical empirical cases). Figure 1 shows the cumulative distribution of the empirical risk gaps across 21 datasets, along with reference lines that partition the space into four interpretive quadrants. As shown in Figure 1, for twelve out of 21 datasets, the approximation gaps have the same magnitude as the theoretical worst-case bound (Optimistic), and for the eleven datasets where the optimal additive models are achieved by MIP (Conservative), there are seven datasets whose approximation gaps are of the same magnitude as the theoretical analysis. This result shows the representativeness and tightness of the theoretical analysis. These results suggest that the theoretical analysis captures not only the worst-case behavior but also reflects typical empirical scenarios.

The remainder of this paper is structured as follows: Section 2 provides background on gradient boosting and additive models. Section 3 presents the lower bound of the worst-case approximation gap of boosting. Section 4 uses a MIP approach to approximate the optimal additive models. With the empirical evaluation using 21 real-world datasets, we show the tightness and the representativeness of our theoretical analysis. Section 5 concludes the paper with key takeaways and future research directions.

2 Background

Throughout this paper we denote by Y the target random variable and by X the input variable that take values in $\mathcal{Y} \subseteq \mathbb{R}$ and $\mathcal{X} \subseteq \mathbb{R}^d$, respectively. Moreover,

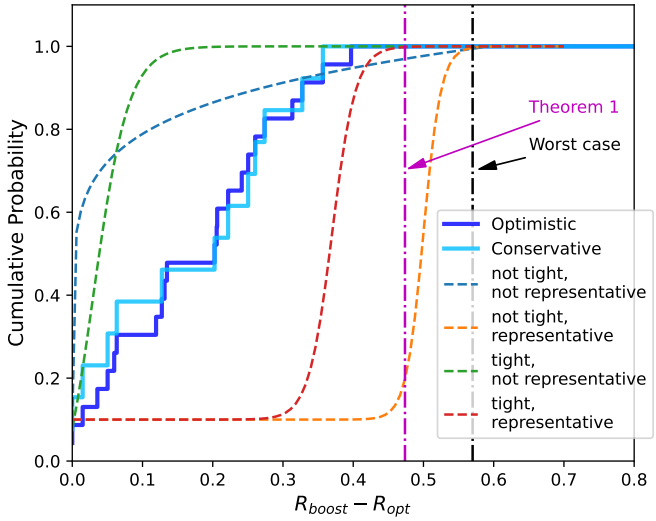

Fig. 1. The cumulative probabilities of the distribution of risk differences between models generated by boosting and MIP where MIP provides a feasible solution (Optimistic) and MIP obtains optimal solutions (Conservative). The dashed curves depict four scenarios regarding the analysis in Theorem 1 and possible worst-case gaps: whether the bound is tight or not to the actual worst-case gap, and whether the worst case is representative of real-world datasets or not.

we assume a dataset $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ drawn from the joint distribution of Y and X. A set of basis functions $\mathcal{H} \subseteq \mathbb{R}^d \to \mathbb{R}$ and an activation or inverse link function $\mu : \mathbb{R} \to \mathbb{R}$ define a family of additive models for predicting Y given X, where an individual model correspond to a finite sub-selection $H = \{h_1, \ldots, h_k\} \subseteq \mathcal{H}$. The H defines an **additive model** as a linear function

$$f(\mathbf{x}) = \sum_{i=1}^{k} \beta_i h_i(\mathbf{x}) = \boldsymbol{\beta}^T \mathbf{h}(x) \tag{1}$$

parameterized by $(\beta_1, \ldots, \beta_k) \in \mathbb{R}^k$ that defines $\hat{y}(\mathbf{x}) = \mu(f(\mathbf{x}))$ as a prediction for Y given $X = \mathbf{x}$.

As an instance of additive models, an **additive rule ensemble** [11,9], or a **rule set** has basis functions in the space of boolean query functions $\mathcal{H}_{\text{query}}$. An additive rule ensemble can be represented as a set of "IF ... THEN ..." rules, which can be easily understood by humans. These additive rule ensembles can be interpretable if the number of rules is not large [14].

2.1 Measurements of Performance of Machine Learning Models

Given a loss function $l\colon \mathbb{R} \times \mathbb{R} \to \mathbb{R}_+$ that quantifies the cost of misprediction and a prescribed number of terms k, our goal is then to select a model $H = (h_1, \ldots, h_k)$

and estimate optimal corresponding parameters $\beta_1, \ldots, \beta_k$ that minimize the **expected loss** or prediction **risk** $R(\boldsymbol{\beta}; H) = \mathbf{E}[l(Y, \hat{y}(X))]$. Absent any further information about the joint distribution of X and Y, this goal is typically pursued by minimizing the **regularized empirical risk**:

$$\hat{R}_\lambda(\boldsymbol{\beta}; H) = \sum_{i=1}^{n} l(y_i, \hat{y}(\mathbf{x}_i))/n + \lambda \Omega(\boldsymbol{\beta})$$

as a surrogate, where $\lambda \in \mathbb{R}_+$ is a regularization parameter and Ω is a positive penalty function that penalizes model parameters according to their magnitude.

An important special case are loss functions that are derived as deviance functions when modeling the target variable as member of a canonical exponential family [13] with natural parameter f, i.e., when considering the probabilistic additive model where the target variable $Y \,|\, X = \mathbf{x}$ follows a probability density or mass function given by

$$p(y \,|\, f(\mathbf{x})) = \exp\left(\frac{y f(\mathbf{x}) - b(f(\mathbf{x}))}{\phi} - c(y, \phi)\right)$$

with some positive dispersion parameter ϕ and scaled log normalizer $b\colon \mathbb{R} \to \mathbb{R}$. In this case, the loss function

$$l(y, \hat{y}) = \log p(y \,|\, \mu^{-1}(y)) - \log p(y \,|\, \mu^{-1}(\hat{y})) \tag{2}$$

is convex in $\hat{y} = f(\mathbf{x})$ [13].

2.2 Forward Selection and Boosting

To learn additive models, forward selection and boosting approaches are commonly used [18,3,10,5]. Forward selection methods construct model sequences $H_0 \subset H_1 \subset \cdots \subset H_k$ with $H_0 = \{\}$ and $H_{t+1} \setminus H_t = \{h_{t+1}\}$ via greedily optimizing a selection score $\mathfrak{S}(\cdot; H_t, \boldsymbol{\beta}^{(t)})$ depending on the previous model and parameter fit:

$$\begin{aligned} h_{t+1} &= \arg\max\{\mathfrak{S}(h; H_t, \boldsymbol{\beta}^{(t)})\colon\ h \in \mathcal{H}\} \\ \boldsymbol{\beta}^{(t+1)} &= \arg\min\{\hat{R}_\lambda(\boldsymbol{\beta}; H_t \cup \{h_{t+1}\})\colon\ \boldsymbol{\beta} \in \mathbb{R}^{t+1}\}\ . \end{aligned}$$

The traditional forward selection methods [3] assume finite sets of basis functions that are small enough to explicitly compute a model fit for each candidate $h \in \mathcal{H} \setminus H_t$, i.e., they use

$$\mathfrak{S}_{\text{fs}}(\varphi; \Phi_t) = -\min\{\hat{R}_\lambda([\boldsymbol{\Phi}_\mathbf{t}; \boldsymbol{\phi}]\boldsymbol{\beta})\colon\ \boldsymbol{\beta} \in \mathbb{R}^{t+1}\}\ . \tag{3}$$

Gradient boosting [10] instead implicitly searches $\mathcal{H}$ by exactly or approximately optimizing a score that quantifies the alignment of the output vector of $h \in \mathcal{H}$ defined by $\mathbf{h} = (h(\mathbf{x}_1), \ldots, h(\mathbf{x}_n))$ with the empirical risk gradient vector $\mathbf{g}$ of the current model fit, i.e., at iteration t

$$g_i = \frac{\partial}{\partial f_t(\mathbf{x}_i)} l(y_i, \mu(f_t(\mathbf{x}_i)))\ \ ,\ \ i = 1, \ldots, n\ . \tag{4}$$

Algorithm 1 Boosting for Additive Models

Input: dataset $(\mathbf{X}, \mathbf{y}) = (x_i, y_i)_{i=1}^n$, number of terms k,
Initialize $f_0 = 0$, $\Phi_0 = []$.
for $t = 1 \cdots k$ **do**
  $h_t = \arg\max_{\mathbf{h}} \text{obj}(\mathbf{h})$
  $\Phi_t = [\Phi_{t-1}; \mathbf{h}_t(\mathbf{X})]$
  $\boldsymbol{\beta}^{(t)} = \text{WeightCalculate}(\Phi_t)$
  $f_t(x) = \boldsymbol{\beta}^{(t)T}\mathbf{h}(x)$
Output the additive model f_k

Starting with an empty model $f_0(\mathbf{x}) = 0$, **idealized boosting** adds a basis function $h_t^* \in \mathcal{H}$ into the additive model to achieve the largest decrease in (regularized) empirical risk for the training dataset in the t-th iteration. Then an updated weight vector $\boldsymbol{\beta}^{(t)}$ is calculated, resulting in a sequence of models $f_1, \cdots, f_k$, where

$$f_t(x) = \sum_{j=1}^{t} \beta_j^{(t)} h_j(x) \qquad \text{, for } t \in \{1, \ldots, k\}. \tag{5}$$

To efficiently choose basis functions, the commonly-used boosting variants approximate h_t^* based on an objective function $\text{obj}(\mathbf{h}) : \mathbb{R}^n \to \mathbb{R}$, which is a function of the output vector of the basis function $\mathbf{h} = (h(x_1), \cdots, h(x_n))$. For instance, we have the gradient boosting [10] objective $\text{obj}_{\text{gb}}(\mathbf{h}) = |\mathbf{h}^T\mathbf{g}|/\|\mathbf{h}\|_2$ and the gradient sum [6,9] objective function $\text{obj}_{\text{gs}}(\mathbf{h}) = |\mathbf{h}^T\mathbf{g}|$.

To calculate the weight vector $\boldsymbol{\beta}$, the **stepwise weight update** method calculates the weight only for the last basis function added into the model [10,9].

$$\boldsymbol{\beta}^{(t)} = \left[\boldsymbol{\beta}^{(t-1)}; \arg\min_{\beta} \hat{R}_\lambda \left(f_{t-1} + \beta h_t\right)\right]. \tag{6}$$

The stepwise weight update method keeps the weight of the previously-generated basis functions unchanged. The **corrective weight update** method [17,21] recalculates the weights of all queries:

$$\boldsymbol{\beta}^{(t)} = \arg\min_{\boldsymbol{\beta}} \hat{R}_\lambda \left(\boldsymbol{\Phi}_t \boldsymbol{\beta}\right), \tag{7}$$

where $\boldsymbol{\Phi}_t = [\mathbf{h}_1, \ldots, \mathbf{h}_t]$ is the $n \times t$ **output matrix** of selected basis functions. The corrective weight update method further reduces the risk of the models, yielding models with higher accuracy for given number of terms. We summarize the general framework of boosting for learning additive models in Algorithm 1.

3 Lower Bound on Boosting Risk Gap

In this section, we give a lower bound to the risk difference between the optimal k-term additive model and the boosting model f_k, i.e., the **approximation gap**

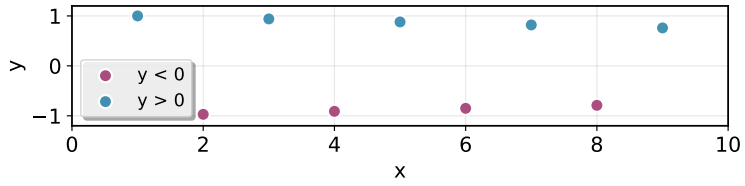

Fig. 2. An example alternating regression dataset with $k = 5$ and $\Delta = 0.03$.

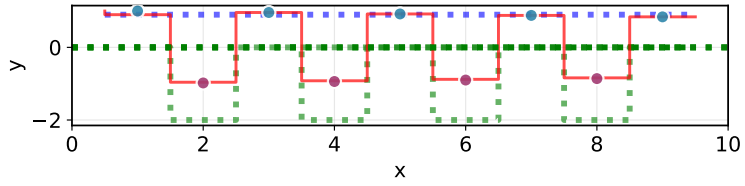

Fig. 3. The reference rule ensemble for $D_{5,0.03}$. The blue dotted line is the rule covering all data points, and the green dotted curves are the other rules. The red curve show the predicted values of the additive rule ensemble.

$R_\lambda(f_k) - R_\lambda(f_k^*)$, in the worst-case across all input datasets. In particular, this bound will apply to the special case of rule base-learners, i.e., $\mathcal{H} = \mathcal{H}_{\text{rules}}$, and the squared loss $l(y, \hat{y}) = l_{\text{sq}}(y, \hat{y}) = (y - \hat{y})^2$. It turns out that the bounds can be proved using a simple construction of datasets with an alternating sequence of positive and negative data points with target values of decreasing magnitude (see Fig. 2 for an illustration). Formally, for a positive integer k, the alternating regression dataset $D_{k,\Delta} \subseteq (\mathbb{R} \times \mathbb{R})^{2k-1}$ is defined by

$$D_{k,\Delta} = \{(x_1, y_1), \ldots, (x_{2k-1}, y_{2k-1})\} \tag{8}$$

$$x_i = i \tag{9}$$

$$y_i = (-1)^{i-1}(a_i) \tag{10}$$

$$a_i = 1 - (i-1)\Delta \tag{11}$$

where $1 \leq i \leq 2k-1$ is the index of data points, and $\Delta > 0$ is used for controlling the sequence of selected queries. For $D_{k,\Delta}$, we can construct a reference rule ensemble f_k^* with k rules as follows: The first rule $w_1 q_1(x)$ has the query $q_1(x_i) = 1$, where $i = 1, \ldots, 2k-1$; for $j = 2, \ldots, k$, the rules $w_j q_j$ are defined as $q_j(x_i) = \mathbb{1}(i = 2(j-1))$, as shown in Figure 3. With this reference rule ensemble, we have the following lemma:

Lemma 1. *For $D_{k,\Delta}$, there exists an additive rule ensemble $f_k^\dagger$ containing k rules whose risk is*

$$R(f_k^\dagger) = \frac{\Delta^2 k(k^2-1)}{3(2k-1)} \quad .$$

Proof. We calculate the risk of the reference rule ensemble f_k^* as

$$\min_{\beta} R(f_k^*) = \min_{\beta} \frac{1}{2k-1} \left(\sum_{i=1}^{k} (\beta_1 - y_{2i-1})^2 + \sum_{i=1}^{k-1} (\beta_1 + \beta_{i+1} - y_{2i})^2 \right)$$

setting $\beta_{i+1} = y_{2i} - \beta_1$ minimizes term i of the second sum, hence we get

$$\begin{aligned} &= \min_{\beta_1} \frac{1}{2k-1} \left(\sum_{i=1}^{k} (\beta_1 - (1-(2i-2)\Delta))^2 \right) \\ &= \min_{\beta_1} \frac{k\left((\beta_1-1)^2 + 2\Delta(\beta_1-1)(k-1) + \frac{2}{3}\Delta^2(k-1)(2k-1)\right)}{2k-1} \end{aligned}$$

which is a quadratic function in $\beta_1 - 1$ which is minimised at $\beta_1 - 1 = -\Delta(k-1)$

$$= \frac{\Delta^2 k(k^2-1)}{3(2k-1)} \ . \tag{12}$$

Therefore, f_k^* satisfies the condition in the lemma. □

Next, we show the behavior of boosting algorithms on the dataset $D_{k,\Delta}$. The following lemma shows the risk of a rule ensemble generated by boosting for $D_{k,\Delta}$ with k rules. Using the risks of the reference rule ensemble and the risk of the rule ensemble generated by boosting, we are able to obtain the lower bound of the worst case approximation gap of the boosting algorithm.

Lemma 2. *For any $D_{k,\Delta}$ with $0 < \Delta < 1/(2k-4+\sqrt{2}/2)$, the total loss of an ensemble f_k of k basis functions fitted by idealized boosting is*

$$R(f_k) = (k-1)\left(\frac{(1-\Delta(3k-2))}{2k-1} + \frac{\Delta^2(6-19k+14k^2)}{6(2k-1)} \right) \ .$$

Proof. We start by examining the matrix Q_t with entries $q_{i,j} = q_i(x_j)$ of the rule ensemble f_k produced by boosting at iteration $t \in \{0, 1, \ldots, 2k-1\}$. Specifically, for $0 < \Delta < 1/(2k-4+\sqrt{2}/2)$, we will prove that the matrix Q_t has the form

$$Q_t = \begin{pmatrix} 1 & 0 & \ldots & \ldots & \ldots & \ldots & \ldots & 0 \\ b_{21} & 1 & 0 & \ldots & \ldots & \ldots & \ldots & 0 \\ b_{31} & b_{32} & 1 & 0 & \ldots & \ldots & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ b_{t1} & b_{t2} & \ldots & b_{t,t-1} & 1 & 0 & \ldots & 0 \end{pmatrix} \ , \tag{13}$$

for all $1 \le i \le t$, $0 \le b_{i,j} \le b_{i,t+1} \le 1$ and $b_{i,j} \in \{0, 1\}$. Once this statement is proven, we can readily show that f_t covers the first t data points with a loss of 0, and that the other data points are uncovered. Formally,

1. For $i \le t$, the loss $l(f_t(x_i), y_i) = 0$;
2. For $i > t$, the loss $l(f_t(x_i), y_i) = (1-(i-1)\Delta)^2$.

For Property 1, since the matrix Q_t has full rank, the equation system $Q_t\beta = \mathbf{y}$ always has a unique solution, hence the corrective weight update ensures that $f_t(x_i) = y_i$ for $1 \leq i \leq t$. For Property 2, if $i > t$, $q_{i,j} = 0$ for all $j \in \{1, \ldots, 2k-1\}$, so $f_t(x_i) = 0$, thus $l(f_t(x_i), y_i) = (1 - (i-1)\Delta)^2$.

We now prove by induction that (13) holds. In the base case $t = 0$, the model is empty. Therefore, the matrix Q_0 is empty, so the base case is true. The induction hypothesis is that (13), and therefore Properties 1 and 2, hold at the t-th iteration. We denote β the weight of the $(t+1)$-th rule generated by boosting, and the ordered set A (resp. B) the indices of positive (resp. negative) data points uncovered by rules q_1 to q_t, and covered by rule q_{t+1}, ordered by increasing x_i. (Note that data points covered by rule $t+1$ can include data points already covered, but Property 1 will ensure that their loss remain 0.) Formally, we define A and B as following:

$$A = \{i | q_{t+1}(x_i) = 1, y_i > 0, q_j(x_i) = 0, \text{ for all } j = 1, \ldots, t\} \ , \tag{14}$$

$$B = \{i | q_{t+1}(x_i) = 1, y_i < 0, q_j(x_i) = 0, \text{ for all } j = 1, \ldots, t\} \ , \tag{15}$$

and A_t (resp. B_t) represents the t-th element of A (resp. B).

We denote $\alpha = |A|$ and $\gamma = |B|$. Since a rule can only cover contiguous data points, and that positive and negative points alternate, we have $|\alpha - \gamma| \leq 1$.

For idealized boosting, the risk reduction of the i-th data point is $(a_i)^2 - (a_i - \beta)^2 = -\beta^2 + 2\beta a_i$ if $y_j > 0$, and $(-a_i)^2 - (-a_i - \beta)^2 = -\beta^2 - 2\beta a_i$ if $y_i < 0$. Then, the total risk reduction produced by the $(t+1)$-th rule is

$$\begin{aligned}
\max_\beta n(R(f_t) - R(f_{t+1})) &= \max_\beta \sum_{k=1}^{\alpha} \left(-\beta^2 + 2\beta a_{A_k}\right) + \sum_{k=1}^{\gamma} \left(-\beta^2 - 2\beta a_{B_k}\right) \\
&= \max_\beta -(\alpha+\gamma)\beta^2 + 2\beta \left(\sum_{k=1}^{\alpha} a_{A_k} - \sum_{k=1}^{\gamma} a_{B_k}\right)
\end{aligned}$$

which is a quadratic function in β whose maximizer is $\beta = \dfrac{\sum_{k=1}^{\alpha} a_{A_k} - \sum_{k=1}^{\gamma} a_{B_k}}{\alpha + \gamma}$

$$= \frac{1}{\alpha+\gamma} \left(\sum_{k=1}^{\alpha} a_{A_k} - \sum_{k=1}^{\gamma} a_{B_k}\right)^2 \ ,$$

where $n = 2k - 1$. Since $a_i = 1 - (i-1)\Delta$,

$$\max n(R(f_t) - R(f_{t+1})) = \max \frac{\left(\alpha a_{A_1} - \gamma a_{B_1} - \alpha(\alpha-1)\Delta + \gamma(\gamma-1)\Delta\right)^2}{\alpha + \gamma}$$

If $\alpha = \gamma$, then $|a_{A_1} - a_{B_1}| = \Delta$, and

$$\max n(R(f_t) - R(f_{t+1})) = \frac{\alpha\Delta^2}{2} \ . \tag{16}$$

Since the first t data points are already covered, the maximum of the risk reduction in the case $\alpha = \gamma$ is $(k - \lfloor (t+1)/2 \rfloor)\Delta^2/2$.

If $\alpha = \gamma + 1$, then $B_1 = A_1 - 1$, $a_{B_1} = a_{A_1} - \Delta$, $\alpha \geq 1$, and

$$\max n(R(f_t) - R(f_{t+1})) = \max \frac{(a_{A_1} - (\alpha - 1)\Delta)^2}{2\alpha - 1} \tag{17}$$

In this expression, the numerator increases if α decreases, and the denominator decreases if α decreases. Therefore, the maximum value of the risk reduction in the case $\alpha = \gamma + 1$ is $a_{A_1}^2$, and it is achieved when $\alpha = 1$. It means that the query which covers only one positive data point reduces the most risk. The maximum risk reduction is achieved at (x_i, y_i) where $i = \min\{j : j > t, y_j > 0\}$.

If $\alpha = \gamma - 1$, then $B_1 = A_1 + 1$, $a_{B_1} = a_{A_1} + \Delta$, $\alpha \geq 0$, and

$$\max n(R(f_t) - R(f_{t+1})) = \max \frac{(a_{B_1} - \alpha\Delta)^2}{2\alpha + 1} \tag{18}$$

Similar to the case of $\alpha = \gamma + 1$, the maximum value of the risk reduction in the case $\alpha = \gamma - 1$ is $a_{B_1}^2$, which is achieved when $\gamma = 1$. Therefore, the query which covers only one negative data point reduces the most risk, and its maximum value is achieved at (x_i, y_i) where $i = \min\{j : j > t, y_j < 0\}$. From the above discussion, if $|\alpha - \gamma| = 1$, then the query covering only (x_{t+1}, y_{t+1}) reduces the most risk, and the maximum risk reduction is $(1 - t\Delta)^2$. If

$$\Delta < \sqrt{2}/(k + \sqrt{2}(t-1) - \lfloor (t+1)/2 \rfloor) \ , \tag{19}$$

i.e. $(1 - t\Delta)^2 > (k - \lfloor (t+1)/2 \rfloor)\Delta^2/2$, the query selected by boosting covers the data points (x_i, y_i) with largest $|y_i|$, which is the $(t+1)$-th data point. To make sure (19) is satisfied for all iterations until the whole dataset is covered, we need

$$0 < \Delta < \frac{1}{2k - 4 + \sqrt{2}/2} \ . \tag{20}$$

According to Properties 1 and 2, with t iterations, the losses of data points $(x_1, y_1), \ldots, (x_t, y_t)$ are 0. Therefore, the risk of the t-term rule ensemble is

$$\begin{aligned} R(f_t) &= \frac{1}{2k-1} \sum_{i=t+1}^{2k-1} (1 - (i-1)\Delta)^2 \\ &= \frac{(2k - t - 1)(6 - 6\Delta(2k + t - 2) + \Delta^2(2t^2 + 4kt + 8k^2 - 5t - 14k + 6))}{6(2k-1)} \ . \end{aligned}$$

If $t = k$, then

$$R(f_k) = \frac{(k-1)(6 - 6\Delta(3k-2) + \Delta^2(14k^2 - 19k + 6))}{6(2k-1)} \ .$$

□

Lemma 2 highlights a key observation regarding the structure of rule ensembles generated by boosting for the alternating dataset $D_{k,\Delta}$. Specifically, the rule

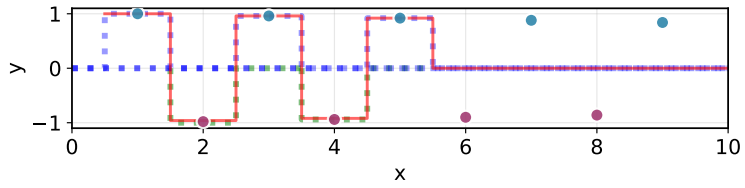

Fig. 4. The additive rule ensemble generated by boosting for $D_{5,0.03}$. The dotted lines represent each rule generated by boosting, and the red solid line represent the output of the model.

ensembles produced by boosting tend to cover only individual data points rather than capturing the broader structure of the dataset. As shown in Figure 4, the rules generated by boosting focus on minimizing the local risk for each point rather than modelling the overall pattern, leading to suboptimal performance when compared to the reference rule ensemble.

With Lemmas 1 and 2, we can calculate the gap between the risks of the models generated by boosting and the reference solution. Formally, the lower bound to the worst-case additive risk gap is given by the following theorem.

Theorem 1. *For $k > 0$ and any $\epsilon > 0$, the worst case additive approximation gap between the k-term optimal model f_k^* and boosting model f_k satisfies*

$$R(f_k) - R(f_k^*) \geq \frac{1}{2} - \frac{1}{4k-2} \ . \tag{21}$$

Proof. Using Lemmas 1 and 2, we obtain for the approximation gap for $D_{k,\Delta}$

$$\begin{aligned} R(f_k) - R(f_k^*) &= \frac{(k-1)(2-\Delta(6k-4)+\Delta^2(4k^2-7k+2))}{2(2k-1)} \\ &= \frac{1}{2} - \frac{1}{4k-2} - \underbrace{\frac{(k-1)(\Delta(6k-4)-\Delta^2(4k^2-7k+2))}{2(2k-1)}}_{\xi(\Delta)} \ . \end{aligned} \tag{22}$$

For $k = 1$, $\xi(\Delta)$ is the constant zero function, hence, the claim is true. For $k > 1$, ξ is a concave quadratic function in Δ with maximum value at $\Delta^* = (3k-4)/(4k^2-7k+2) > 0$. The claim is true for $\epsilon \geq \xi(\Delta^*)$. For $0 < \epsilon < \xi(\Delta^*)$, we observe that $\xi(0) = 0$. Therefore, we can find a positive Δ' such that $\xi(\Delta') = \epsilon$ by the intermediate value theorem, which concludes the proof. □

Note that this result is consistent with the optimality of boosting in the special case of $k = 1$. For $k = 2$, we obtain a lower bound to the approximation gap of $1/3$. In the limit, for arbitrarily large k, the bound converges to $1/2$. Further, it is worth noting that, while the proof of the result uses idealized boosting (forward selection), it can be adapted to the other discussed variants.

4 Empirical Evaluation of the Boosting Risk Gap via MIP

In Section 3, we analyze the lower bound of the worst-case gap between the risks of boosting and optimal models theoretically. However, to answer the question of how tight this bound is, and how representative the worst case is for the real-world datasets, we need to explore alternative approaches of generating rule ensembles with lower risks. In this section, we adopt a MIP approach to approximate the optimal models for different datasets.

MIP involves modelling a problem into a formulation containing decision variables, constraints and an objective function, and solving the formulation using existing solvers like Gurobi [2,12]. In a typical MIP setting, a part of variables are constrained to be integers or discrete values, while others can take continuous values [20]. MIP allows us to find the optimal solution by maximizing or minimizing an objective function subject to a set of constraints. MIP is particularly well-suited for problems where decisions (such as whether a query is selected) are binary or integer-based, while other parameters, like rule weights, are continuous, making it commonly used to obtain or approximate the optimal additive models [8,17,19].

A MIP formulation consists of several key components [20]. First, *decision variables* represent the choices to be made, with some restricted to integers (e.g., binary decisions) and others take continuous values. The *objective function* is a mathematical expression that the MIP seeks to optimize (either minimize or maximize), often involving a cost or risk function. The problem is subject to *constraints*, which are typically linear equations or inequalities that limit the possible values of the decision variables, ensuring that solutions are feasible. Previous works presented in literatures [15,12,17] solve the machine learning problem using MIP solvers, but they usually adopt column generation approaches to model the problem, such as IPBoost [15]. Instead, in our approach, we model the problem of learning additive rule ensembles directly.

4.1 The MIP formulation

First, we introduce the MIP formulation for approximating optimal additive rule ensembles. Below is a list of variables used in this formulation:

- $l_{t,j}$ (resp. $u_{t,j}$): the lower (resp. upper) bound of t-th rule on j-th dimension ($t = 1, \ldots, k$, $j = 1, \ldots, d$, same below) demonstrated in Figure 5.
- β_t: the weight of the t-th rule, as shown in Figure 5.
- $s_{i,t,j}$: A binary variable indicating whether the i-th data point is between the lower bound and the upper bound on the j-th dimension of the t-th rule ($i = 1, \ldots, n$, same below).
- $z_{i,t}$: A binary variable indicating whether the i-th data point is selected by the t-th rule. Figure 6 demonstrates the usage of the variables $s_{i,t,j}$ and $z_{i,t}$.
- $r_{i,t,j}$ (resp. $p_{i,t,j}$): a binary variable indicating whether the j-th coordinate value of the i-th data point is greater (resp. less) than $l_{t,j}$.

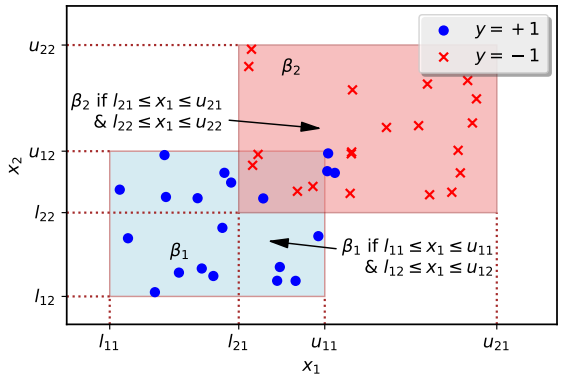

Fig. 5. An illustration of the decision variables used in the MIP formulation for learning additive rule ensembles, including the lower and upper bounds and weights of rules. There are two rules, and the query of each rule is a conjunction of two propositions.

The auxiliary variables $r_{i,t,j}$ and $p_{i,t,j}$ facilitate the expressing of $s_{i,t,j}$ as linear functions of other variables. Specifically, $r_{i,t,j}$ (resp. $p_{i,t,j}$) indicates whether the i-th data point is greater (resp. less) than the lower (resp. upper) bound on the dimension j. The value of $s_{i,t,j}$ is 1 only if both $r_{i,t,j}$ and $p_{i,t,j}$ are 1, meaning the point is within the specified range. A data point is covered by the t-th rule if $s_{i,t,j} = 1$ for all dimensions, in which case $z_{i,t} = 1$. Totally, there are $O(ndk)$ decision variables in this formulation.

To reduce the number of constraints, rather than directly using lower and upper bounds for each rule in every dimension, we represent rules based on the data points they cover. The dataset is sorted along each dimension, and a point is selected by a rule if it has a neighboring point that is also selected. Specifically: If a selected data point exists to the left (resp. right) of the i-th data point on dimension j, then $r_{i,t,j}$ (resp. $p_{i,t,j}$) is 1. If both $r_{i,t,j}$ and $p_{i,t,j}$ are 1, the i-th data point is covered by the t-th rule on dimension j.

The list of constraints is described below:

- On dimension j, if both neighboring points of the i-th data point are covered by the t-th rule, then the i-th data point is also covered by the t-th rule:

$$s_{i,t,j} \ge s_{i_<,t,j} + s_{i_>,t,j} - 1 \qquad \forall t, j,$$

  where $i_<$ (resp. $i_>$) refer to the data point whose value is smaller (resp. greater) on dimension j, if any.
- If neither neighbors of the i-th data point are covered by the t-th rule on dimension j, the i-th data point is not, either:

$$s_{i,t,j} \le s_{i_<,t,j} + s_{i_>,t,j} \qquad \forall t, j.$$

- If two points i and $i_=$ have the same coordinate on dimension j, then

$$s \text{ (resp. } r, p)_{i,t,j} = s \text{ (resp. } r, p)_{i_=,t,j} \qquad \forall t, j \ .$$

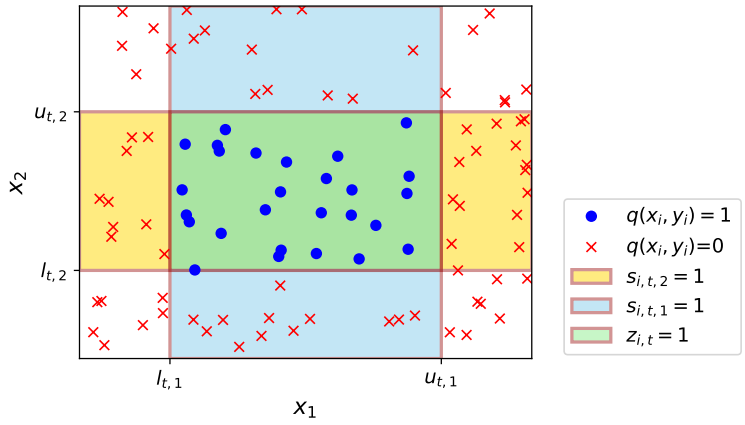

Fig. 6. The constraints in the MIP Formulation for learning rule ensembles. These constraints guarantee that data points satisfying the condition of a query are covered by the query. In this figure, the query q is represented as a conjunction of two propositions: "$l_{t,1} \leq x_1 \leq u_{t,1}$ **and** $l_{t,2} \leq x_2 \leq u_{t,2}$".

- $r_{i,t,j}$ and $p_{i,t,j}$ are no less than $s_{i,t,j}$:

$$r \text{ (resp. } p)_{i,t,j} \geq s_{i,t,j} \qquad \forall i, t, j.$$

- $r_{i,t,j}$ (resp $p_{i,t,j}$) is monotonically non-decreasing:

$$r_{i,t,j} \geq r_{i<,t,j} \ , \ p_{i,t,j} \leq p_{i<,t,j} \qquad \forall i, t, j.$$

- $s_{i,t,j} = 1$ if and only if both $r_{i,t,j} = 1$ and $p_{i,t,j} = 1$:

$$s_{i,t,j} = r_{i,t,j} + p_{i,t,j} - 1 \qquad \forall i, t, j.$$

  This constraint ensures that if a data point is between the lower bound and upper bound of a proposition, then it satisfies the condition of the proposition.
- The data point x_i is in the box of the t-th rule if

$$z_{i,t} \leq s_{i,t,j}, \quad \forall j \ ,$$
$$z_{i,t} \geq \sum_j s_{i,t,j} - d - 1 \ .$$

  This constraint indicates that if a data point satisfies all the propositions in a query, then it is covered by this query, as demonstrated in Figure 6.

The objective of this optimization problem is the regularized empirical risk, which is calculated as:

$$\hat{R}_\lambda = \frac{1}{n} \sum_{i=1}^{n} l \left(y_i, \sum_{t=1}^{k} z_{i,t} \beta_t \right) + \frac{\lambda}{n} \sum_{t=1}^{k} \Omega(\beta_t), \tag{23}$$

where n is the number of data points, k is the number of rules, λ is the regularization parameter, l is the loss function, and Ω is a regularization function. The risk function is the objective in this optimization formulation. The objective of this optimization formulation is to minimize the risk value. After solving the optimization problem, we determine the upper and lower bounds of the t-th rule on the j-th dimension according to the values of $s_{i,t,j}$ for all $i = 1, \ldots, n$.

Based on this MIP formulation, we can easily implement the idealized boosting algorithm (MIP-boosting). In the m-th iteration, we fix the weights $\beta_t^{(m)}$, lower and upper bounds $l_{t,j}^{(m)}$ and $u_{t,j}^{(m)}$, $s_{i,t,j}^{(m)}$ and $z_{i,t}^{(m)}$ of the t-th rule ($t < m$) generated in previous iterations as constants. We take $s_{i,t,j}^{(m)}$ as an example:

$$s_{i,t,j}^{(m)} = s_{i,t,j}^{(m-1)} \quad , t < m \quad .$$

Only the decision variables with $t = m$, such as $\beta_m^{(m)}$, $s_{i,m,j}^{(m)}$ and $z_{i,m}^{(m)}$, related to the new rule to be generated in the m-th iteration are included as variables in the MIP formulation in the m-th iteration. Therefore, in such a configuration, only one more rule is generated by the optimization solver. There are totally $O(nd)$ decision variables and $O(nd)$ constraints in this formulation. With the MIP-boosting implementation, the problem of learning additive rule ensembles generated by boosting are easily solved optimally, where the MIP bounds of the training risks are the same as the solutions. In each iteration, this implementation adds one rule which minimizes the empirical risk into the model.

4.2 Experiments

To answer the question whether the lower bound of the worst-case approximation gap of boosting shown in Theorem 1 is tight and representative, we compare the risks of models generated by MIP-boosting and the original MIP formulation. To obtain the optimal models, the original MIP formulations are solved using starting values provided by the boosting implementation. With this setting, the models generated by MIP are always better than boosting models. All the experiments use Gurobi to solve the MIP formulations. The time budget for generating each rule for MIP and boosting are set as 600 second for all datasets. All experiments in this paper are conducted on a computer with "Intel(R) Core(TM) i5-10300H CPU @ 2.50GHz" and memory of 72GB. All code and datasets used in our experiments are provided in the supplementary materials and in the GitHub repository `https://github.com/fyan102/MIPRule` to ensure full reproducibility.

We compare the empirical risks of additive rule ensembles with 10 rules generated by boosting for 21 datasets, and the optimal model implemented using MIP approach in Table 1, as well as the MIP bounds. The *MIP bound* is the lower bound of the objective value that can be achieved by the MIP solver, indicating how close the solution found by the solver is to the true optimum. In the table, we observe that for eleven datasets, such as `IBM HR`, `gdp`, `fitness`, and `salary`, the MIP bound matches the training risk achieved by MIP, indicating that the solver has successfully proved optimality for these models. For these datasets, the

Table 1. Comparison of training risks of additive rule ensembles trained via boosting (b, implemented with MIP) and globally optimal models obtained via the MIP approach (M) for both classification ($^+$) and regression tasks. The middle section shows the maximum difference in training risk between the two approaches (bolded), the iteration k^* where this occurs, and the corresponding risks. The right section reports the training risks of both approaches, objective bounds (bnd) of MIP, and runtimes (t, in seconds) for 10 rules. Optimal MIP risk values and their bounds are in red.

dataset	d	n	Max risk difference				Training 10 rules				
			$\text{diff}_{\max}$	k^*	R_{b,k^*}	R_{M,k^*}	$R_{\text{b},10}$	$R_{\text{M},10}$	$\text{bnd}_{\text{M},10}$	$t_{\text{b},10}$/s	$t_{\text{M},10}$/s
tic-tac-toe$^+$	27	958	**.397**	9	.835	.438	.793	.422	.000	2579	5082
salary	1	30	**.357**	2	.501	.144	.050	.050	.050	.214	3.08
study time	1	96	**.328**	2	.669	.341	.069	.069	.069	.654	75.2
demog.	13	6876	**.313**	10	.919	.606	.919	.606	.000	3061	5814
iris$^+$	4	150	**.274**	2	.490	.216	.095	.028	.028	133	4081
IBM HR$^+$	32	1470	**.260**	10	.848	.587	.847	.587	.587	340	1779
student marks	2	100	**.250**	2	.636	.386	.050	.045	.045	1.48	1762
titanic$^+$	7	1043	**.241**	1	.788	.547	.593	.387	.000	1767	4728
income	2	20	**.222**	2	.512	.290	.018	.016	.016	.371	1757
happiness	8	315	**.206**	2	.628	.421	.093	.071	.000	49.2	2059
used cars	4	1770	**.206**	3	.622	.417	.216	.202	.000	1045	2903
gdp	1	35	**.203**	2	.687	.484	.385	.385	.385	.964	141
boston	13	506	**.135**	2	.695	.560	.326	.307	.000	261	2364
headbrain	3	237	**.131**	2	.743	.612	.395	.387	.000	3.918	1800
fitness	3	30	**.128**	2	.573	.446	.115	.108	.1078	.641	1755
mobile	20	2000	**.119**	2	.842	.722	.602	.542	.000	2617	5340
wine$^+$	13	178	**.063**	3	.145	.082	.031	.013	.013	4719	5606
insurance	6	1338	**.060**	3	.383	.323	.196	.185	.000	1166	3547
social media	2	63	**.050**	4	.585	.535	.500	.490	.490	.871	1761
wage	5	1379	**.036**	3	.756	.720	.614	.594	.000	1109	2903
breast$^+$	30	569	**.015**	10	.392	.378	.392	.378	.378	240	4590

solution obtained is guaranteed to be the global optimum, which highlights the effectiveness of the MIP approach in learning optimal rule ensembles for datasets with a small size and dimensions. However, there are ten datasets with large number of data points and features (like `titanic`, `tic-tac-toe` and `insurance`) where the optimality of the solutions cannot be guaranteed. For these datasets, the optimal models may still have a lower risk than the risks shown in Table 1.

In Table 1, we show the maximum difference between the risks of rule ensembles generated by boosting and MIP over all the iterations between 1 and 10 in the middle part of the table, as well as the iteration number the maximum difference occurs and their risk values. We plot the distribution of the largest risk differences between the boosting and optimal models in Figure 1. There are totally twelve datasets where the largest risk gaps are of the same magnitude as the theoretical lower bound, indicating that the worst-case scenario we analyze is representative. Furthermore, since we have not found any datasets whose largest risk difference

between boosting and MIP models are larger than the theoretical analysis, the experimental results show that the actual approximation gaps are usually less than the theoretical bound, suggesting that the lower bound of the worst-case approximation gap is tight. This implies that boosting generally provides a good approximation to the optimal solutions, with performance significantly better than the worst-case bound in most real-world cases.

Notably, the MIP approach generally incurs a higher computational cost compared to boosting. This is primarily due to its increased complexity: the number of decision variables and constraints in the full MIP formulation scales as $O(nkd)$. In contrast, the MIP-boosting implementation only involves $O(nd)$ variables and constraints, since it incrementally adds one rule at a time. As the number of rules increases, solving the full MIP formulation becomes increasingly time-consuming, often reaching the pre-specified time limit. Nevertheless, as shown in Table 1, for nine datasets the running times of the MIP and MIP-boosting methods are of the same order of magnitude, demonstrating that MIP can remain tractable for moderate-sized problems.

5 Conclusion

This paper examines the approximation gap by boosting to generate additive models theoretically and empirically. We provide a constructive proof showing the lower bound of the worst-case risk gap between boosting and the optimal solutions, which is approximately 50% of the initial risk. This gap arises from the greedy nature of boosting, highlighting the need for alternative approaches to obtain better accuracy-complexity trade-offs. To empirically explore the gap, we introduce MIP formulations to approximate optimal additive models. By reducing redundancy in candidate solutions, MIP finds optimal models for small datasets. The experimental results confirm that the observed risk gap aligns with our theoretical analysis, suggesting the tightness and representative of the worst-case bound. Future work may further explore the theoretical upper bound of the risk gap of boosting. Another interesting direction is to improve the scalability of MIP formulations to learn additive models for larger, more complex datasets.

Acknowledgements This work was supported by the Australian Research Council (DP210100045). We thank the anonymous reviewers for their valuable feedback that led to a substantial improvement of this work.

References

1. Bénard, C., Biau, G., Da Veiga, S., Scornet, E.: Interpretable random forests via rule extraction. In: International Conference on Artificial Intelligence and Statistics. pp. 937–945. PMLR (2021)
2. Bertsimas, D., Dunn, J.: Optimal classification trees. Machine Learning **106**(7), 1039–1082 (2017)

3. Blanchet, F.G., Legendre, P., Borcard, D.: Forward selection of explanatory variables. Ecology **89**(9), 2623–2632 (2008)
4. Boley, M., Teshuva, S., Le Bodic, P., Webb, G.I.: Better short than greedy: Interpretable models through optimal rule boosting. In: Proceedings of the 2021 SIAM International Conference on Data Mining (SDM). pp. 351–359. SIAM (2021)
5. Chen, T., Guestrin, C.: Xgboost: A scalable tree boosting system. In: Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining. pp. 785–794 (2016)
6. Cohen, W.W., Singer, Y.: A simple, fast, and effective rule learner. AAAI/IAAI **99**(335-342), 3 (1999)
7. Das, A., Kempe, D.: Approximate submodularity and its applications: Subset selection, sparse approximation and dictionary selection. Journal of Machine Learning Research **19**(3), 1–34 (2018)
8. Dash, S., Gunluk, O., Wei, D.: Boolean decision rules via column generation. Advances in neural information processing systems **31** (2018)
9. Dembczyński, K., Kotłowski, W., Słowiński, R.: Ender: a statistical framework for boosting decision rules. Data Mining and Knowledge Discovery **21**(1), 52–90 (2010)
10. Friedman, J.H.: Greedy function approximation: a gradient boosting machine. Annals of statistics pp. 1189–1232 (2001)
11. Friedman, J.H., Popescu, B.E.: Predictive learning via rule ensembles. The annals of applied statistics pp. 916–954 (2008)
12. Günlük, O., Kalagnanam, J., Li, M., Menickelly, M., Scheinberg, K.: Optimal decision trees for categorical data via integer programming. Journal of global optimization **81**(1), 233–260 (2021)
13. McCullagh, P., Nelder, J.A.: Generalized linear models. Routledge (2019)
14. Murdoch, W.J., Singh, C., Kumbier, K., Abbasi-Asl, R., Yu, B.: Definitions, methods, and applications in interpretable machine learning. Proceedings of the National Academy of Sciences **116**(44), 22071–22080 (2019)
15. Pfetsch, M., Pokutta, S.: Ipboost–non-convex boosting via integer programming. In: International Conference on Machine Learning. pp. 7663–7672. PMLR (2020)
16. Shalev-Shwartz, S., Srebro, N., Zhang, T.: Trading accuracy for sparsity in optimization problems with sparsity constraints. SIAM Journal on Optimization **20**(6), 2807–2832 (2010)
17. Shen, C., Li, H., Van Den Hengel, A.: Fully corrective boosting with arbitrary loss and regularization. Neural Networks **48**, 44–58 (2013)
18. Sutter, J.M., Kalivas, J.H.: Comparison of forward selection, backward elimination, and generalized simulated annealing for variable selection. Microchemical journal **47**(1-2), 60–66 (1993)
19. Wei, D., Dash, S., Gao, T., Gunluk, O.: Generalized linear rule models. In: International Conference on Machine Learning. pp. 6687–6696. PMLR (2019)
20. Wolsey, L.A.: Mixed integer programming. Wiley Encyclopedia of Computer Science and Engineering pp. 1–10 (2007)
21. Yang, F., Le Bodic, P., Kamp, M., Boley, M.: Orthogonal gradient boosting for simpler additive rule ensembles. In: International Conference on Artificial Intelligence and Statistics. pp. 1117–1125. PMLR (2024)