Orthogonal Gradient Boosting for Simpler Additive Rule Ensembles

Fan Yang
Monash University

Pierre Le Bodic
Monash University

Michael Kamp
IKIM, University Hospital Essen

Mario Boley
Monash University

Abstract

Gradient boosting of prediction rules is an efficient approach to learn potentially interpretable yet accurate probabilistic models. However, actual interpretability requires to limit the number and size of the generated rules, and existing boosting variants are not designed for this purpose. Though corrective boosting refits all rule weights in each iteration to minimise prediction risk, the included rule conditions tend to be sub-optimal, because commonly used objective functions fail to anticipate this refitting. Here, we address this issue by a new objective function that measures the angle between the risk gradient vector and the projection of the condition output vector onto the orthogonal complement of the already selected conditions. This approach correctly approximates the ideal update of adding the risk gradient itself to the model and favours the inclusion of more general and thus shorter rules. As we demonstrate using a wide range of prediction tasks, this significantly improves the comprehensibility/accuracy trade-off of the fitted ensemble. Additionally, we show how objective values for related rule conditions can be computed incrementally to avoid any substantial computational overhead of the new method.

1 INTRODUCTION

Additive rule ensembles describe a target variable through a linear combination of conjunctions of threshold functions on individual input variables. These models, also called "rule sets", combine an interpretable syntax with high modelling flexibility. Thus, they are a useful alternative to employing opaque models with post-hoc explanations (e.g., Strumbelj and Kononenko, 2010; Ribeiro et al., 2016), which can be misleading (Rudin, 2019; Kumar et al., 2021). Indeed, like simple generalized linear models (GLMs, McCullagh and Nelder, 2019) and generalized additive models (GAMs, Hastie and Tibshirani, 1990; Lou et al., 2013), additive rule ensembles are "modular" and "simulatable" (Murdoch et al., 2019), i.e., they allow the interpretation of one term (rule) at a time, and the output of each term can be computed by a human interpreter "in her head". However, in contrast to those simpler models, rule ensembles can also model interaction effects of an arbitrary number of input coordinates. In fact, they can be regarded as a generalization of tree and forest models by representing each tree leaf through a rule. Though, given the motivation of interpretability, we are interested in finding much smaller rule ensembles than

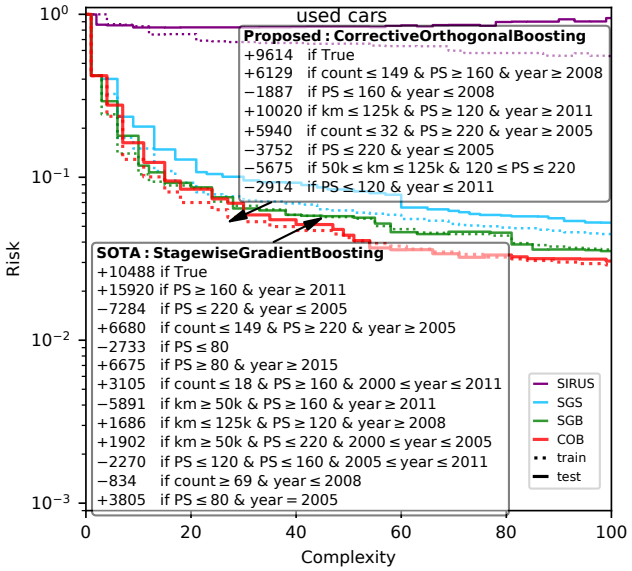

Figure 1: Risk/complexity trade-offs achieved by different rule ensemble learners for `used_cars`. Proposed approach (COB, red) reaches risk 0.135 with ensemble of complexity 27, measured as number of rules plus their total length, as opposed to 46 for best alternative.

Proceedings of the 27th International Conference on Artificial Intelligence and Statistics (AISTATS) 2024, Valencia, Spain. PMLR: Volume 238.

those typically defined by a forest. In other words, the goal in rule ensemble learning really is to optimize the trade-off between accuracy and complexity (see Fig. 1).

Algorithms for learning additive rule ensembles range from computationally inexpensive generate-and-select approaches (Friedman and Popescu, 2008; Lakkaraju et al., 2016; Bénard et al., 2021), over more expensive minimum-description length and Bayesian approaches (Wang et al., 2017), to expensive full-fledged discrete optimization methods (Dash et al., 2018; Wei et al., 2019). Within this range of options, methods based on *gradient boosting* (Friedman, 2001) are of special interest because of their good accuracy relative to their cost and their flexibility to adapt to various response variable types and loss functions. These methods identify one rule condition at a time by optimizing an objective function that aims to approximate gradient descent of the empirical risk in terms of the ensemble's prediction vector. Current boosting adaptions to rule learning (Cohen and Singer, 1999; Dembczyński et al., 2010; Boley et al., 2021) are, however, based on design choices that compromise the risk/complexity trade-off of the fitted rule ensembles. They use stagewise weight updates where rules are not revised after they are added to the ensemble, although re-optimizing all weights in every iteration in a (totally) *corrective update* (Kivinen and Warmuth, 1999; Shalev-Shwartz et al., 2010) can achieve a smaller risk for the same ensemble complexity. Interestingly, fixing this issue is not as straightforward as simply switching to a corrective update, because current boosting objective functions do not anticipate weight corrections, and, as we show here, this can lead to highly sub-optimal choices and in particular does not guarantee to find the rule condition that best approximates the inclusion of the risk gradient itself to the multi-dimensional weight search.

Here we investigate a new objective function that provides this guarantee and leads to consistent gains in the risk reduction per added rule. This function is based on considering only the part of a rule body orthogonal to the already selected rules, which takes into account that the predictions for previously covered training examples can be adjusted during weight correction. In addition to providing this refined objective function, we also derive a corresponding algorithm for incrementally evaluating sequences of related rule conditions, which is crucial for efficiently optimizing the objective function in each boosting round, whether through greedy or branch-and-bound search. As we demonstrate on a wide range of datasets, the resulting rule boosting algorithm significantly outperforms the previous boosting variants in terms of risk/complexity trade-off, which can be attributed to a better risk reduction per rule as well as an affinity to select more general rules. At the same time, the computational cost remains comparable to previous objective functions. We present these main technical contributions in Sec. 3 and their empirical evaluation in Sec. 4 and provide a concluding discussion in Sec. 5. To start, we briefly recall gradient boosting for rule ensembles in Sec. 2.

2 RULE BOOSTING

Additive rule ensembles are probabilistic models that describe the mean of a target variable Y conditional on an input variable X as $\mathbf{E}[Y \mid X = \mathbf{x}] = \mu(f(\mathbf{x}))$ where $\mu\colon \mathbb{R} \to \mathbb{R}$ is an inverse link or **mean function** and $f\colon \mathbb{R}^d \to \mathbb{R}$ is an affine linear combination of k Boolean **query functions**. That is,

$$f(\mathbf{x}) = \beta_0 + \sum_{i=1}^{k} \beta_i q_i(\mathbf{x}) \tag{1}$$

where each $q_i\colon \mathbb{R}^d \to \{0,1\}$ is a product $q_i(\mathbf{x}) = p_{i,1}(\mathbf{x}) p_{i,2}(\mathbf{x}) \dots p_{i,c_i}(\mathbf{x})$ of **propositions** $p_{i,j}(\mathbf{x}) = \mathbb{1}(s x_l \geq t)$ with $s \in \{\pm 1\}$ and $t \in \mathbb{R}$. Each term in (1) can be interpreted as an IF-THEN rule where the binary queries q_i define the rule antecedents (conditions), and the **weights** $\boldsymbol{\beta} = (\beta_1, \dots, \beta_k) \in \mathbb{R}^k$ define the rule consequents, i.e., the output of rule i for input $\mathbf{x} \in \mathbb{R}^d$ is β_i if $\mathbf{x}$ satisfies the antecedent, i.e., $q_i(\mathbf{x}) = 1$ (and 0 otherwise). The offset weight β_0 conceptually describes the output of a **background rule** with a trivial condition that is satisfied for all data points.

We are concerned with the trade-off of two properties of an additive rule ensemble f: its **complexity** $C(f) = k + \sum_{i=1}^{k} c_i$, which approximates the cognitive effort required to parse all rules, and its **prediction risk** $\mathbf{E}[l(f(X), Y)]$, which measures its accuracy with respect to some positive loss function $l(f(\mathbf{x}), y)$ for new random data sampled with respect to the joint distribution of X and Y. Here we consider loss functions that can be derived as deviance function, i.e., shifted negative log likelihood (such that $l(\mu^{-1}(y), y) = 0$), when interpreting the rule ensemble output as natural parameter of an exponential family model of the target variable. This guarantees that the loss function is strictly convex and twice differentiable (McCullagh and Nelder, 2019, p.33). Specifically, we consider the cases of $Y \mid X$ being normally distributed and $\mu(a) = a$ resulting in the **squared loss** $l_{\text{sqr}}(f(x_i), y_i) = (f(x_i) - y_i)^2$, Bernoulli distributed and $\mu(a) = 1/(1 + \exp(-a))$ resulting in the **logistic loss** $l_{\log}(f(x_i), y_i) = \log(1 + \exp(-y_i f(x_i)))$, and Poission distributed and $\mu(a) = \exp(a)$ resulting in the **Poisson loss** $l_{\text{poi}}(f(x_i), y_i) = y_i \log y_i - y_i f(x_i) - y_i + \exp(f(x_i))$.

Gradient boosting Gradient boosting methods are iterative fitting schemes for additive models based on

minimizing the (regularized) **empirical risk** $R_\lambda(f) = \sum_{i=1}^{n} l(f(\mathbf{x}_i), y_i)/n + \lambda\|\boldsymbol{\beta}\|^2/n$ taken over a training set $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$. In the context of rule learning, these methods produce a sequence of rule ensembles $f^{(0)}, f^{(1)}, \ldots, f^{(k)}$ starting with an empty $f^{(0)} = \beta_0$ where $\beta_0 = \arg\min\{R_0(\mathbf{1}\beta)\}$, and for $1 \leq t \leq k$,

$$f^{(t)}(\mathbf{x}) = \beta_0 + \beta_{t,1} q_1(\mathbf{x}) + \cdots + \beta_{t,t} q_t(\mathbf{x})$$
$$q_t = \arg\max\{\mathrm{obj}(\mathbf{q}; f^{(t-1)}) \colon q \in \mathcal{Q}\},$$

where query q_t is chosen to maximize some objective $\mathrm{obj} : \mathbb{R}^n \to \mathbb{R}$ that is defined on the query output vector $\mathbf{q} = (q(x_1), \ldots, q(x_n))$ and that additionally depends on the previous model, in particular its output vector $\mathbf{f}_{t-1} = (f^{(t-1)}(\mathbf{x}_1), \ldots, f^{(t-1)}(\mathbf{x}_n))$. Specifically, the original gradient boosting scheme (Friedman, 2001) performs an approximate gradient descent in the model output space by choosing

$$\mathrm{obj}_{\mathrm{gb}}(\mathbf{q}) = |\mathbf{g}^T\mathbf{q}|/\|\mathbf{q}\| \ , \tag{2}$$
$$\boldsymbol{\beta}_t = [\boldsymbol{\beta}_{t-1}; \arg\min_{\beta \in \mathbb{R}} R_\lambda(f^{(t-1)} + \beta q_t)], \tag{3}$$

where $\mathbf{g}$ is the **gradient vector** of the unregularized empirical risk function with components $g_i = \partial l(f(\mathbf{x}_i), y_i)/\partial f(\mathbf{x}_i)$. We refer to (2) as **gradient boosting objective** and to (3) as **stagewise weight update**, because it fits the final prediction function in "stages" corresponding to each term.

One popular variant of this scheme chooses queries that correspond to a direction in output space with maximum rate of change in risk according to its first order approximation (Mason et al., 1999). This results in the **gradient sum objective**

$$\mathrm{obj}_{\mathrm{gs}}(\mathbf{q}) = |\mathbf{g}^T\mathbf{q}| \tag{4}$$

which is guaranteed (Dembczyński et al., 2010, Thm. 1) to select rules as least as general as the gradient boosting objective in terms of the number of selected data points $\|\mathbf{q}\|_1$. This increased generality, however, can come at the expense of a reduced risk reduction when choosing $\boldsymbol{\beta}$, because the output correction of data points with large gradient elements has to be toned down to avoid over-correction of other selected data points with small gradient elements. Another variant (Chen and Guestrin, 2016; Boley et al., 2021) minimizes the second order approximation to R_λ via the **extreme boosting objective** and corresponding closed-form weight updates

$$\mathrm{obj}_{\mathrm{xgb}}(\mathbf{q}) = |\mathbf{g}^T\mathbf{q}|/\sqrt{\mathbf{h}^T\mathbf{q} + \lambda} \tag{5}$$
$$\boldsymbol{\beta}_t = [\boldsymbol{\beta}_{t-1}; -\mathbf{q}^T\mathbf{g}/(\mathbf{q}^T\mathbf{h} + \lambda)], \tag{6}$$

where $\mathbf{h} = \mathrm{diag}(\nabla^2_{f(\mathbf{x})} R(f))$ is the diagonal vector of the unregularized risk Hessian again with respect to the output vector $\mathbf{f}$. Note that this approach is well-defined for our loss functions derived from exponential family response models, which guarantee defined and positive $\mathbf{h}$. In particular for the squared loss, it is equivalent to standard gradient boosting, because the second order approximation is exact for l_{sqr} and $\mathbf{h}$ is constant.

While the closed-form solution of the weight-updates (6) reduces the update cost by some amount, it can be highly sub-optimal whenever the second order approximation is loose as, e.g., is the case with the Poisson loss. Especially if one aims for small interpretable rule ensembles and correspondingly wants to minimize the risk for each ensemble size, it appears sensible to choose $\boldsymbol{\beta}$ that minimize the actual empirical risk. The most consequent realization of this idea is given by **corrective boosting**[1] (Kivinen and Warmuth, 1999; Shalev-Shwartz et al., 2010) where the component-wise weight updates (3) are replaced by a full joint re-optimization of the weights of all selected queries, i.e.,

$$\boldsymbol{\beta}_t = \arg\min\{R_\lambda(\mathbf{Q}_t\boldsymbol{\beta}) \colon \boldsymbol{\beta} \in \mathbb{R}^t\} \ , \tag{7}$$

where $\mathbf{Q}_t = [\mathbf{q}_1, \ldots, \mathbf{q}_t]$ is the $n \times t$ **query matrix** with the output vectors of all selected queries as columns. Given that our loss function l and therefore the empirical risk R_λ are convex, the additional computational cost for solving (7) instead of (3) is negligible relative to the cost of query optimization, especially when targeting small ensembles sizes k.

Single rule optimization While the rule optimization literature can be broadly divided into branch-and-bound and greedy (or more generally beam) search (see Fürnkranz, 1999), these approaches are actually closely related: in a typical breadth-first-search implementation, both searches start with search level 0 containing only the trivial query $q(\mathbf{x}) \equiv 1$. Then, for $l = 1, 2, \ldots$ they generate search level l as all augmentations

$$q'(\mathbf{x}_0) = q(\mathbf{x}_0)\mathbb{1}(sx_{0,j} \leq sx_{i,j}) \tag{8}$$

for all q in search level $l-1$ filtered by, especially for branch-and-bound, a non-redundancy check[2] and a bound check $\mathrm{bnd}(q) \geq \mathrm{obj}(q^*)$ where q^* denotes the best query encountered so far. Restricting each search level to contain only the top w candidates for some finite **search width** w results in **beam search** with $w = 1$ corresponding to the standard greedy search. In the case of unrestricted level size ($w = \infty$), corresponding

[1] In the literature, the update (7) is often referred to as "totally corrective" or "fully corrective" whenever the unqualified term "corrective boosting" is already used to refer to the standard stagewise update (3) .

[2] In efficient implementations this non-redundancy check ensures that each query output vector $\mathbf{q}$ is at most enqueued once through exploitation of a closure system formed by the queries (see Boley et al. (2021) and references therein).

to **branch-and-bound**, finding the optimal query is guaranteed if $\mathrm{bnd}(q)$ is an upper bound to the objective value of all possible subsequent augmentations to q. In this case the bound is called **admissible**. For instance, the **selectability unaware** bounding function

$$\mathrm{bnd}(q) = \max\{\mathrm{obj}(\mathbf{q}')\colon \mathbf{q}' \leq \mathbf{q}, \mathbf{q}' \in \{0,1\}^n\} \ , \quad (9)$$

where $\mathbf{q} \leq \mathbf{q}'$ refers to the component-wise less or equals relation, is admissible, because it relaxes the problem of finding the best extension of q propositions to finding the best subset of the data points selected by q (Morishita and Sese, 2000; Grosskreutz et al., 2008).

In any search scheme like the one sketched above, a substantial computational cost is incurred for evaluating the objective function values of a large number of candidate queries, many of which are closely related. Hence, efficient implementations rely on fast incremental computations of as many of those objective values as possible. Specifically, they reduce the candidate generation and bounding to the following **prefix value problem**: for a given ordered sub-selection of m data points $\sigma\colon \{1,\ldots,m\} \to \{1,\ldots,n\}$, compute all

$$\{\mathrm{obj}(\mathbf{e}_{\sigma(1)} + \cdots + \mathbf{e}_{\sigma(i)})\colon 1 \leq i \leq m\} \quad (10)$$

where we denote by $\mathbf{e}_i$ the i-th standard unit vector. The key is to solve this problem in time $O(m)$ instead of the $O(m^2)$ resulting from separate computation of obj for all $i \in \{1,\ldots,m\}$. For candidate generation, it can then be solved once per input variable j and $s \in \{\pm 1\}$, setting σ to order $\{1,\ldots,n\}$ according to $s x_{i,j}$, to compute the objective values of all $q' = q\mathbb{1}(s x_{0,j} \geq x_{i,j})$ in time $O(n)$. This results in $O(dn)$ cost for computing all refinements (8) instead of $O(dn^2)$ incurred with separate computation. For bound computation, the problem is solved twice per candidate q to compute the selectability unaware bounding function (9) with a **prefix greedy** optimization (Lemmerich et al., 2016; Boley et al., 2017). Here, σ describes the sub-selection of the $m = \mathbf{1}^T\mathbf{q}$ data points selected by q, and the order varies based on the objective function. For $\mathrm{obj}_{\mathrm{gb}}$, correct computation of (9) is achieved by ordering with respect to $s g_i$. For $\mathrm{obj}_{\mathrm{xgb}}$ correct computation is achieved by ordering with respect to the ratio $s g_i / h_i$ (Boley et al., 2021).

3 ORTHOGONAL GRADIENT BOOSTING

Using the corrective weights updates (7) turns boosting into a form of forward variable selection for linear models. However, in contrast to conventional variable selection where all variables are given explicitly, we still have to identify a good query q_t in each boosting iteration, and it turns out that finding the appropriate query is more complicated with the corrective update than with the stagewise update (3).

3.1 Best Geometric Approximations

To be more precise, assume we are adding to the ensemble a basis function with output vector $\mathbf{v}$ in boosting round t, and let us denote by $A^{\mathrm{SU}}_{\mathbf{v}} = \mathbf{f}_{t-1} + \mathrm{span}\{\mathbf{v}\}$ the affine weight optimization subspace of the stagewise update (3) and by $A^{\mathrm{CU}}_{\mathbf{v}} = \mathrm{range}[\mathbf{Q}_{t-1}; \mathbf{v}]$ the weight optimization subspace of the corrective weight update (7). Moreover, let $A_{\mathbf{v}}$ be some family of weight optimization spaces such as either of the two defined above, and let $\mathbf{f} \in \mathbb{R}^n$ be some target vector, e.g., in our case, the empirical risk minimizer in the optimization space $A_{\mathbf{g}}$ given by the risk gradient itself. We define the **best geometric approximation** to $\mathbf{f}$ with respect to the query language $\mathcal{Q}$ as the projection of $\mathbf{f}$ onto the space $A_{\mathbf{q}}$ resulting in the smallest error among all $q \in \mathcal{Q}$, i.e.,

$$\tilde{\mathbf{f}}_{\mathcal{Q}} = \arg\min\{\|\mathbf{f} - \mathbf{f}'\|\colon \mathbf{f}' \in A_{\mathbf{q}}, q \in \mathcal{Q}\} \ .$$

In the case of a stagewise update space $A^{\mathrm{SU}}_{\mathbf{v}}$, it is easy to show that for any $\mathbf{f} = \mathbf{f}_{t-1} + \alpha\mathbf{v} \in A^{\mathrm{SU}}_{\mathbf{v}}$ the squared projection error for a given $q \in \mathcal{Q}$ is a decreasing function of the angle between $\mathbf{q}$ and $\mathbf{v}$:

$$\min_{\beta \in \mathbb{R}} \|\alpha\mathbf{v} - \beta\mathbf{q}\|^2 = \alpha^2 \left(\|\mathbf{v}\|^2 - \frac{(\mathbf{q}^T\mathbf{v})^2}{\|\mathbf{q}\|^2} \right) \ . \quad (11)$$

Since $\mathrm{obj}_{\mathrm{gb}}(\mathbf{q})$ is proportional to the cosine between $\mathbf{q}$ and $\mathbf{g}$, maximizing it thus identifies the $q \in \mathcal{Q}$ with the best geometric approximation to any target vector $\mathbf{f} \in A^{\mathrm{SU}}_{\mathbf{g}}$. Consequently, for the ideal gradient descent update at round t, i.e.,

$$\mathbf{f}^{\mathrm{GD}} = \arg\min\{R_\lambda(\mathbf{f}')\colon \mathbf{f}' \in A^{\mathrm{SU}}_{\mathbf{g}}\} \ ,$$

the best geometric approximation $\tilde{\mathbf{f}}^{\mathrm{GD}}_{\mathcal{Q}}$ is an element of $A^{\mathrm{SU}}_{\mathbf{q}}$, where $\mathbf{q}$ denotes the output vector of a query that maximizes $\mathrm{obj}_{\mathrm{gb}}$. Moreover, as the final output vector $\mathbf{f}_{\mathrm{gb}}$ is found via line search in this affine subspace, its risk is no greater than that of the best geometric approximation, i.e., $R_\lambda(\mathbf{f}_{\mathrm{gb}}) \leq R_\lambda(\tilde{\mathbf{f}}^{\mathrm{GD}}_{\mathcal{Q}})$.

However, when using the corrective update in combination with an ideal basis function with an output fully aligned with $\mathbf{g}$, the target vector becomes

$$\mathbf{f}^{\mathrm{CD}} = \arg\min\{R_\lambda(\mathbf{f}')\colon \mathbf{f}' \in A^{\mathrm{CU}}_{\mathbf{g}}\} \ . \quad (12)$$

In this case, all previously discussed objective functions fail to identify the best geometric approximation in general, resulting in a large excess risk. For the standard gradient boosting objective, this is demonstrated in the example in Fig. 2. Here, $\mathrm{obj}_{\mathrm{gb}}$ identifies in round 2 a query resulting with or without weight correction in an output vector $\mathbf{f}^{\mathrm{CD}}_{\mathrm{gb}} = \mathbf{f}^{\mathrm{GB}}_{\mathrm{gb}}$ with (unregularized) risk 24/9, whereas the best geometric approximation to the target vector $\tilde{\mathbf{f}}^{\mathrm{CD}}_{\mathcal{Q}}$ has a risk of only 1/9.

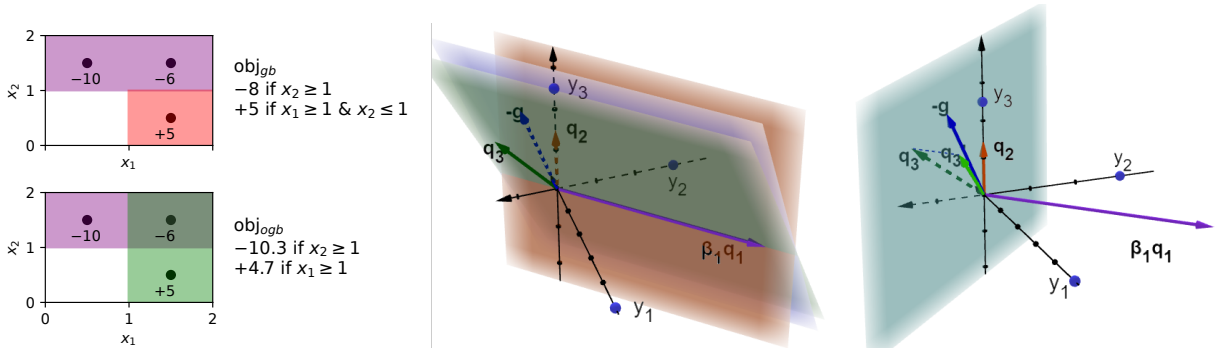

Figure 2: **Left.** Regression example with three data points with target values $\mathbf{y} = (-10, -6, 5)$ and output vectors $\mathbf{f}_{\text{gb}} = (-8, -8, 5)$ with risk 24/9 and $\mathbf{f}_{\text{ogb}} = (-10.3, -5.6, 4.7)$ with risk 1/9 of two-rule ensembles produced by corrective weight updates with obj_{gb} and obj_{ogb}, respectively. **Middle.** Approximations to target subspace (blue) spanned by $\mathbf{q}_1 = (1, 1, 0)$ and $-\mathbf{g} = (-2, 2, 5)$. The subspace (green) spanned by $\mathbf{q}_3 = (0, 1, 1)$ and $\mathbf{q}_1$ is a better approximation than the subspace (orange) spanned by $\mathbf{q}_2 = (0, 0, 1)$ and $\mathbf{q}_1$. However, the latter is selected by standard gradient boosting (obj_{gb}), because $\mathbf{q}_2$ and $-\mathbf{g}$ form a smaller angle than $\mathbf{q}_3$ and $-\mathbf{g}$. **Right.** After projection onto orthogonal complement of $\mathbf{q}_1$, angle between $\mathbf{q}_3$ and $-\mathbf{g}$ is smaller than that between $\mathbf{q}_2$ and $-\mathbf{g}$ and is thus selected by orthogonal gradient boosting objective (obj_{ogb}).

3.2 Objective for Corrective Boosting

The intuitive reason for the gradient boosting objective failing to identify the correct query in Fig. 2 is that selecting x_2 in addition to x_3 is not beneficial for the overall risk reduction *if* we are only allowed to set the weight for the newly selected query. This is because then this weight has to be a compromise between the two different magnitudes of correction required for x_2, which only needs a small positive correction, and x_3, which needs a large positive correction. If we, however, are allowed to change the weight of the previously selected query this consideration changes, because we can now balance an over-correction for x_2 by adjusting the weight of the first rule. While on first glance it seems unclear how much of such re-balancing can be applied without harming the overall risk, it turns out that this is captured by a simple criterion based on the norm of the part of the newly selected query that is orthogonal to the already selected ones (see SI for all proofs).

Proposition 1. Let $\mathbf{f}^{\text{CGD}} = \arg\min\{R_\lambda(\mathbf{f}') : \mathbf{f}' \in \text{range}[\mathbf{Q}_{t-1}; \mathbf{g}]\}$ be the output vector of the ideal corrective gradient descent update in round t and

$$q = \arg\max_{q \in \mathcal{Q}} |\mathbf{g}_\perp^T \mathbf{q}| / \|\mathbf{q}_\perp\| \tag{13}$$

where for $\mathbf{v} \in \mathbb{R}^n$ we denote by $\mathbf{v}_\perp$ its projection onto the orthogonal complement of range $\mathbf{Q}_{t-1}$. Then the output vector $\mathbf{f}_q = \arg\min\{R_\lambda(\mathbf{f}') : \mathbf{f}' \in \text{range}[\mathbf{Q}_{t-1}; \mathbf{q}]\}$ dominates the optimal geometric approximation to $\mathbf{f}$ with respect to $\mathcal{Q}$, i.e., $R_\lambda(\mathbf{f}_q) \leq R_\lambda(\tilde{\mathbf{f}}_{\mathcal{Q}}^{\text{CGD}})$.

The right hand side of Eq. (13) is undefined for redundant query vectors $\mathbf{q}$ that lie in range $\mathbf{Q}$ and therefore have $\|\mathbf{q}_\perp\| = 0$. Therefore we add a small positive value ϵ to the denominator, which can be considered an alternative **regularization parameter**. With this we define the **orthogonal gradient boosting** objective function as

$$\text{obj}_{\text{ogb}}(q) = |\mathbf{g}_\perp^T \mathbf{q}| / (\|\mathbf{q}_\perp\| + \epsilon) \ . \tag{14}$$

This function measures the cosine of the angle between the gradient vector and the orthogonal projection of a candidate query vector $\mathbf{q}$. This is in contrast to the standard gradient boosting objective, which considers the angle of the unprojected query vector instead. In the example in Fig. 2 we can observe that this difference leads to successfully identifying the best approximating subspace, as guaranteed by Prop. 1. In that example, this corresponds to a factor of 24 improvement over the empirical risk produced by the gradient boosting objective function. In fact, the potential advantage of orthogonal gradient boosting is unbounded relative to both the gradient boosting and the gradient sum objective function.

Proposition 2. There is one-dimensional input data $\mathbf{X} \in \mathbb{R}^{5\times 1}$ and a sequence of output data, $\mathbf{y}^{(m)} \in \mathbb{R}^5$, such that $\lim_{m\to\infty} R_{\text{ogb}}^{(m)} = 0$ but

$$\lim_{m\to\infty} R_{\text{gs}}^{(m)} = \lim_{m\to\infty} R_{\text{gb}}^{(m)} = \infty$$

where $R_*^{(m)} = R_0(f_*^{(3)})$ is the risk of the three-element rule ensemble f_* trained on $(\mathbf{X}, \mathbf{y}^{(m)})$ using obj_* and the corrective weight update.

Proof sketch. We define $\mathbf{X}$ as $(1, 2, 3, 4, 5)$ and $\mathbf{y}^{(m)} = (-\alpha_m - \epsilon_m, \alpha_m, -3\alpha_m - \epsilon_m, \alpha_m + \epsilon_m, 2\alpha_m + \epsilon_m)$, where

Algorithm 1 Corrective Orthogonal Boosting

Input: dataset $(\mathbf{x}_i, y_i)_{i=1}^n$, number of rules k

$\sigma_j = \text{argsort}(\{x_{i,j}\colon 1 \le i \le n\})$ **for** $j \in \{1, \dots, p\}$
$f^{(0)} = \beta_0 = \arg\min\{R_\lambda(\beta \mathbf{1})\colon \beta \in \mathbb{R}\}$
for $t = 1$ **to** k **do**
  $\mathbf{g}_t = \left(\partial l(f^{(t-1)}(\mathbf{x}_i), y_n)/\partial f^{(t-1)}(\mathbf{x}_i)\right)_{i=1}^n$
  $q_t = \text{ogb\_baselearner}(\mathbf{g}_t, \mathbf{O}_t)$
  $\mathbf{q}_{t,\perp} = \mathbf{q} - \mathbf{O}_t \mathbf{O}_t^T \mathbf{q}$
  $\mathbf{o}_t = \mathbf{q}_{t\perp}/\|\mathbf{q}_{t\perp}\|$ and $\mathbf{O}_t = [\mathbf{O}_{t-1}; \mathbf{o}_t]$
  $\boldsymbol{\beta}_t = \arg\min_{(\beta_1,\dots,\beta_t)\in\mathbb{R}^t} R_\lambda(\beta_0 + \sum_{j=1}^t \beta_j q_j)$
  $f^{(t)}(\cdot) = \beta_0 + \beta_{t,1} q_1(\cdot) + \cdots + \beta_{t,t} q_t(\cdot)$
Output: $f^{(0)}, \dots, f^{(k)}$

$\alpha_m, \epsilon_m \in \mathbb{R}_{>0}$ are two arbitrary sequences with $\alpha_m \to \infty$ and $\epsilon_m \to 0$. With this we have for the risk of the generated rule ensembles

$$
\begin{aligned}
R_{\text{gs}}^{(m)} &\ge 2\left(6\alpha_m^2 + 2\alpha_m\epsilon_m + \epsilon_m^2\right)/5 \\
R_{\text{gb}}^{(m)} &= 3\alpha_m^2/2 \\
R_{\text{ogb}}^{(m)} &= 3\epsilon_m^2/5
\end{aligned}
$$

implying the claimed limits. □

3.3 Efficient Implementation

To develop an efficient optimization algorithm for the orthogonal gradient boosting objective, we recall that projections $\mathbf{q}_\perp$ on the orthogonal complement of range $\mathbf{Q}$ can be naively computed via $\mathbf{q}_\perp = \mathbf{q} - \mathbf{Q}((\mathbf{Q}^T\mathbf{Q})^{-1}(\mathbf{Q}^T\mathbf{q}))$ where we placed the parentheses to emphasize that only matrix-vector products are involved in the computation—at least once the inverse of the Gram matrix $\mathbf{Q}^T\mathbf{Q}$ is computed. This approach allows to compute projections, and thus objective values, in time $O(nt + t^2)$ per candidate query after an initial preprocessing per boosting round of cost $O(t^2 n + t^3)$.

In a first step, this naive approach can be improved by maintaining an orthonormal basis of the range of the query matrix throughout the boosting rounds, resulting in a Gram-Schmidt-type procedure. Since the projections $\mathbf{q}_\perp$ of the selected query output vectors already form an orthogonal basis of range $\mathbf{Q}$ this only requires normalization with negligible additional cost. Formally, by storing $\mathbf{o}_t = \mathbf{q}_{t\perp}/\|\mathbf{q}_{t\perp}\|$ in all boosting rounds t, subsequent projections can be computed via $\mathbf{q}_{t+1,\perp} = \mathbf{q}_{t+1} - \mathbf{O}_t(\mathbf{O}_t^T\mathbf{q}_{t+1})$ where $\mathbf{O}_t = [\mathbf{o}_1, \dots, \mathbf{o}_t]$. This reduces the computational complexity per candidate query to $O(tn)$ without additional preprocessing.

As mentioned in Sec. 2, to achieve an acceptable scalability with n, we need to reduce this complexity further for evaluating sequences of related queries, as required by candidate generation and bounding. In particular we need to solve the prefix value problem (10), which translates for our objective function to computing:

$$
\left\{|\mathbf{g}_\perp^T \mathbf{q}^{(i)}|/(\|\mathbf{q}_\perp^{(i)}\| + \epsilon)\colon 1 \le i \le l\right\} \tag{15}
$$

given an ordered sub-selection σ where $\mathbf{q}^{(0)} = \mathbf{0}$ and $\mathbf{q}^{(i)} = \mathbf{q}^{(i-1)} + \mathbf{e}_{\sigma(i)}$. The following proof shows how the computational complexity for solving (15) can be substantially reduced compared to the direct approach above. It uses an incremental computation of projections that works directly on the available orthonormal basis vectors $\mathbf{o}$ instead of computing matrix-vector

Algorithm 2 OGB Base Learner

Input: data $(\mathbf{x}_i)_{i=1}^n$, grad. $\mathbf{g}$, orthonorm. $\mathbf{O} \in \mathbb{R}^{n,t}$

$\mathbf{g}_\perp \leftarrow \mathbf{g} - \mathbf{O}\mathbf{O}^T\mathbf{g}$
$\varphi \leftarrow \text{argsort}(\mathbf{g}_\perp)$
$v_*, q_*, I_* \leftarrow (0, 1, \{1, \dots n\})$
$\mathcal{B} \leftarrow$ empty priority queue with size limit w
enqueue$(\mathcal{B}, (v_*, q_*, I_*))$
while not empty$(\mathcal{B})$ **do**
  $\mathcal{B}' \leftarrow$ empty priority queue with size limit w
  for $(v, q, I) \in \mathcal{B}$ **do**
    $\varphi' \leftarrow \text{filter}(\varphi, I)$
    $b_+ \leftarrow \max \text{prefix\_values}(\mathbf{g}_\perp, \mathbf{O}, \varphi')$
    $b_- \leftarrow \max \text{prefix\_values}(\mathbf{g}_\perp, \mathbf{O}, \text{inverted}(\varphi'))$
    if $\max\{b_+, b_-\} \le v_*$ **or** redundant(q, I) **then**
      continue
    for $j \in \{1, \dots, d\}$ **do**
      for $s \in \{-1, 1\}$ **do**
        $\sigma \leftarrow \text{filter}(\sigma_j, I)$
        if $s = -1$ **then** $\sigma \leftarrow \text{inverted}(\sigma)$
        $\mathbf{v} = \text{prefix\_values}(\mathbf{g}_\perp, \mathbf{O}, \sigma)$
        for $i \in \{1, \dots, |\sigma|\}$ **do**
          $q'(\mathbf{x}_0) \leftarrow q(\mathbf{x}_0)\mathbb{1}(sx_{0,j} \le sx_{\sigma(i),j})$
          if $v' > v_*$ **then**
            $v_*, q_* \leftarrow v', q'$
          enqueue$(\mathcal{B}', (v', q', \{\sigma(1), \dots, \sigma(i)\}))$
  $\mathcal{B} \leftarrow \mathcal{B}'$
Output: q_*

Algorithm 3 Incremental Prefix Values

Input: proj. grad. $\mathbf{g}_\perp$, orthon. $\mathbf{O} \in \mathbb{R}^{n,t}$, order σ

$G, N_1, \dots, N_t \leftarrow (0, 0, \dots, 0)$
for $i \in \{1, \dots, |\sigma|\}$ **do**
  $G \leftarrow G + g_{\perp,\sigma(i)}$
  for $k \in \{1, \dots, t\}$ **do**
    $N_k \leftarrow N_k + o_{k,\sigma(i)}$
  $v_i = |G|/\left(\sqrt{i^2 - \sum_{k=1}^t N_k^2} + \epsilon\right)$
Output: $(v_1, \dots, v_l)$

products or, even worse, the whole projection matrix.

Proposition 3. Given a gradient vector $\mathbf{g} \in \mathbb{R}^n$, an orthonormal basis $\mathbf{o}_1, \ldots, \mathbf{o}_t \in \mathbb{R}^n$ of the subspace spanned by the queries of the first t rules, and a subselection of l candidate points $\sigma\colon [l] \to [n]$, the prefix value problem (15) can be solved in time $O(tl)$.

Proof sketch. We can write the objective value of prefix i in terms of incrementally computable quantities:

$$\frac{\left|\mathbf{g}_\perp^T \mathbf{q}^{(i)}\right|}{\|\mathbf{q}_\perp^{(i)}\| + \epsilon} = \frac{\left|\mathbf{g}_\perp^T \mathbf{q}^{(i)}\right|}{\sqrt{\|\mathbf{q}^{(i)}\|^2 - \|\mathbf{q}_\parallel^{(i)}\|^2} + \epsilon}$$
$$= \frac{\left|\mathbf{g}_\perp^T \mathbf{q}^{(i)}\right|}{\sqrt{\|\mathbf{q}^{(i)}\|^2 - \sum_{k=1}^t \|\mathbf{o}_k \mathbf{o}_k^T \mathbf{q}^{(i)}\|^2} + \epsilon}\,.$$

In particular, the t sequences of norms $\|\mathbf{o}_k \mathbf{o}_k^T \mathbf{q}^{(i)}\|$ can be computed in time $O(l)$ via cumulative summation of the k-th basis vector elements in the given order:

$$\|\mathbf{o}_k \mathbf{o}_k^T \mathbf{q}^{(i)}\| = \|\mathbf{o}_k\| \left|\sum_{j=1}^{i} \mathbf{o}_k^T \mathbf{e}_{\sigma(j)}\right| = \left|\sum_{j=1}^{i} o_{k,\sigma(j)}\right|$$

□

This result implies that the computational complexity of maximizing the objective function in boosting iteration t depends linearly on t, which introduces an asymptotic overhead of a factor of k over the previous objective functions when running boosting for k iterations. However, as we will discuss below, in practice for finite k we typically find a much smaller overhead, e.g., of around 2 for most data sets when running 10 iterations. This is due to additive preprocessing costs that are common to all objective functions and the tendency to select more general rules compared to GB/XGB.

We close this section with pseudocodes that summarize the main ideas of orthogonal gradient boosting, starting with the high level algorithm (Alg. 1), followed by the base learner for the query optimization at each boosting round (Alg. 2), and closing with the fast incremental objective function computation (Alg. 3). This specific version of Alg. 2 uses breadth-first-search for ease of exposition. Other search orders can be implemented. Moreover, it uses the following operations and notations for ordered sub-collections of data point indices $\sigma : \{1, \ldots, m\} \to \{1, \ldots n\}$, which can be represented by integer arrays of length m. The cardinality symbol refers to the size of the represented sub-collection, i.e., $|\sigma| = m$. The operation filter(σ, I) refers to the order $(\sigma(i)\colon\ 1 \leq i \leq |\sigma|, \sigma(i) \in I)$, which can be computed in time $O(m)$ assuming I is given as a Boolean array. Finally, inverted(σ) refers to the inverted order $\sigma'(i) = \sigma(m - i + 1)$. For beam search the priority queue limit w has to be set to some finite positive integer, where $w = 1$ yields the standard greedy algorithm as special case. The setting $w = \infty$ leads to branch-and-bound, for which the red lines implement the bounding part. Importantly, the computed bounding function based on a prefix greedy optimization with respect to the gradient order is just a heuristic for obj_{ogb}. Therefore, one might want to omit it for beam search. However, we find in extensive numerical experiments (see SI) that this approach leads to a 3/4-approximation algorithm with high probability.

Table 1: Datasets with size n, dimensions d, and type: classification ($^+$), Poisson (*), ordinary regression ( )

Dataset	d	n	Dataset	d	n	Dataset	d	n
ships*	4	34	breast$^+$	30	569	gender$^+$	20	3168
gdp	1	35	tic-tac-toe$^+$	27	958	digits5$^+$	64	3915
smoking*	2	36	titanic$^+$	7	1043	friedman3	4	5000
covid vic*	4	85	insurance	6	1338	demograph.	13	6876
bicycle*	4	122	banknote$^+$	4	1372	tel. churn$^+$	18	7043
iris$^+$	4	150	wage	5	1379	friedman2	4	10000
wine$^+$	13	178	ibm hr$^+$	32	1470	magic$^+$	6	16327
covid*	2	225	red wine	11	1599	videogame	5	27820
happiness	8	315	life expect.	21	1649	suicide rate	5	27820
liver$^+$	6	345	used cars	4	1770	adult$^+$	11	30162
diabetes	10	442	mobile price	20	2000			
boston	13	506	friedman1	10	2000			

4 EMPIRICAL EVALUATION

In this section, we present empirical results comparing the proposed corrective orthogonal gradient boosting (COB) to the standard gradient boosting algorithms (Dembczyński et al., 2010) using greedy optimization of obj_{gb} (SGB) and obj_{gs} (SGS) with stagewise weight update, to extreme gradient boosting (Boley et al., 2021) using branch-and-bound optimisation of obj_{xgb} with stagewise weight update (SXB), and finally to SIRUS (Bénard et al., 2021) as the state-of-the-art generate-and-filter approach. We investigate the risk/complexity trade-off, the affinity to select general rules, as well as the computational complexity. The datasets used are those of Boley et al. (2021) augmented by three additional classification datasets from the UCI machine learning repository and, to introduce a novel modelling task to the rule learning literature, five counting regression datasets from public sources. This results in a total of 34 datasets (13 for classification, 16 for regression, and 5 for counting/Poisson regression, see Tab. 1). The experiment code and further information about the datasets are available on GitHub (`https://github.com/fyan102/FCOGB`).

All algorithms were run five times on all datasets using 5 random 80/20 train/test splits to calculate robust estimates of all considered metrics. For each boosting

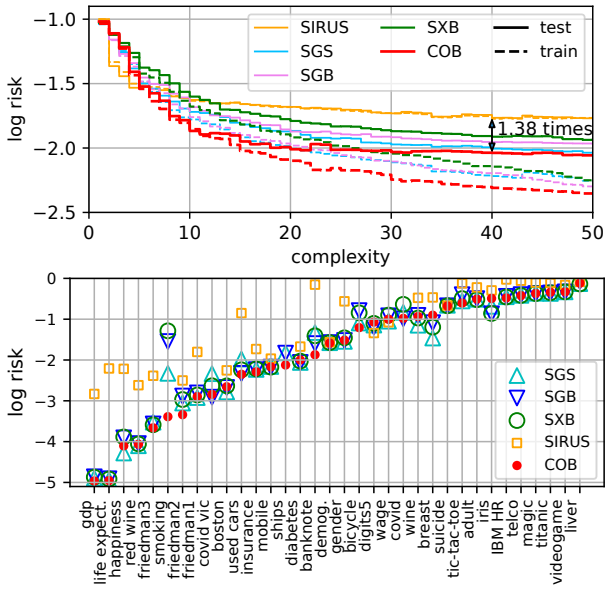

Figure 3: Method log risks across complexity levels (top) and log test risks across datasets (bottom).

variant, we consider ensembles with $k = 1, 2, \ldots$ rules until the complexity exceeds 50. For each ensemble size k (and each training split) a separate regularization parameter value λ_k is found via an internal five-fold cross-validation as

$$\arg\min\{R_{\text{CV}}(f_\lambda^{(k)}) : \lambda = 10^a, a \in \{-2, -1, 0, 1, 2\}\}$$

Note that with this procedure, the ensemble complexity is not necessarily monotone with the number of rules, i.e., because typically $\lambda_l > \lambda_k$ for $l > k$. For SIRUS, we consider different ensembles by varying the minimum occurrence frequency p_0 of a rule (in trees of the random forest) required to be added to the ensemble, incrementally decreasing p_0 until complexity 50 is exceeded.

Complexity versus risk We firstly compare the complexity/risk trade-off of SGB, SGS, SXB and COB. Fig. 3 compares the log risks in terms of the complexity of rule ensembles generated by all methods. The log risks are used such that their difference indicate risk ratios between methods, as $\log(R_A/R_B) = \log(R_A) - \log(R_B)$. Here, normalization is performed by the risk of the rule ensemble with a single empty "offset" rule. The top part compares the risks per complexity level averaged across all datasets. One can see that, for almost all complexity levels, COB has the smallest average training and test risks for all but the smallest complexity levels, where it is only beaten by SIRUS. On the other hand, SIRUS is not competitive for complexity levels greater than 15. The bottom part of Fig. 3 compares the risk per dataset averaged across all considered cognitive complexity levels from 1 to 50.

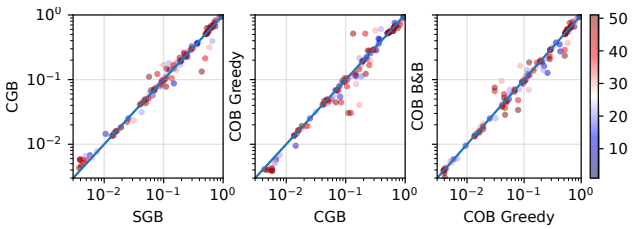

Figure 4: Comparison of risks with different complexity levels between Stepwise Gradient Boosting (SGB), Corrective Gradient Boosting (CGB), COB using Greedy search and COB using Branch-and-bound search. The colours represent the complexity of the rule ensembles.

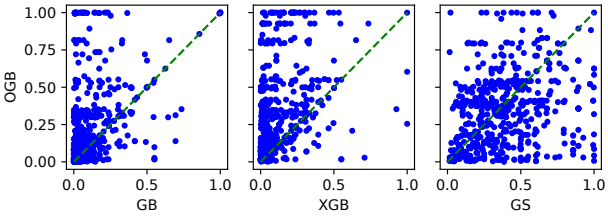

Figure 5: Coverage rate of the rules generated by Gradient Boosting, XGBoost, Gradient Sum versus OGB.

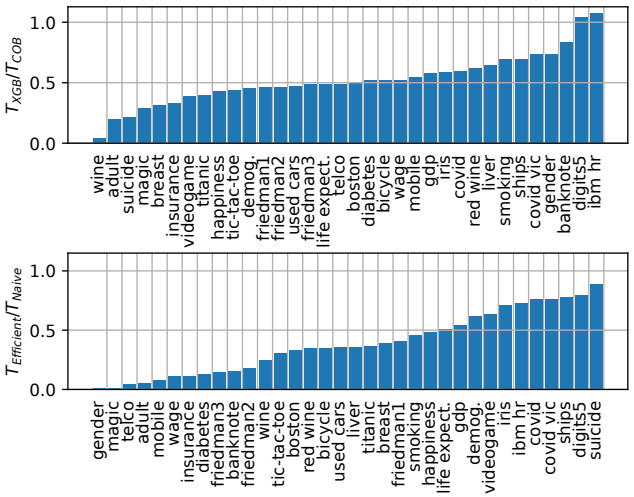

Figure 6: Running time ratio of SXB and COB (top) and naive and efficient opt. of COB (bottom).

COB generates rules which have the smallest test risks for 23 out of 34 datasets (and the smallest training risks for 26 out of 34 datasets, see Fig. 9 in SI). Moreover, COB occasionally outperforms the second-best algorithm by a wide margin (*tic-tac-toe*, *banknote*, *insurance*, *friedman2*, *used cars*, *smoking*). One-sided paired t-tests at significance level 0.05 (Bonferroni-corrected for 8 hypotheses, 4 for training risks and 4 for test risks) reveal that COB significantly outperforms all other methods for a random dataset by a margin of at least 0.001 average training risk and testing risk.

Ablation study After seeing that the developed rule learning approach yields overall significant advantages, we next investigate the necessity of its individual components. Fig. 4 compares the risk of various intermediate variants of COB for all investigated datasets and representative complexity levels. As can be seen, applying just corrective weights updates does not substantially improve standard gradient boosting with stagewise updates and greedy rule optimization, with only 57.7% of the considered ensembles improved. Correspondingly, a ones-sided paired t-test at significance level 0.05 does not reject the null hypothesis that the expected risks differ at all, i.e., without a margin, between these two methods (for a random dataset and complexity level). In contrast, using the orthogonal objective function obj_{ogb} in conjunction with corrective updates improves the risk values in 78.9% of the cases (compared to using obj_{gb} with corrective update), and optimizing obj_{ogb} with branch-and-bound search instead of greedy search improves the performance further (63.4% of ensembles are improved). Both improvements are significant, i.e., two further one-sided paired t-tests at significance level 0.05, with Bonferroni correction for three tests, reject the null hypotheses that the difference in expected risks are less than 0.001. In terms of the ensemble sizes there are no clear patterns in terms of the number of wins and losses per step. However, we can observe that larger differences tend to be attained on larger ensembles.

Effect on Condition Complexity The objective functions primarily affect the risk / complexity trade-off through the number of rules required to reach a certain complexity level. However, as a secondary effect, they might differ in their propensity to select complex queries with many propositions. While condition complexity is not directly measured by any of the investigated objective functions, it is related to the number of data points selected or "covered" by the query, because simpler queries tend to cover more data points. Here we assess the differences in **coverage rate**, $\|\mathbf{q}\|_1/n$, as follows: for each alternative objective obj_{alt} we fit rule ensembles $f_{\text{alt}}^{(t)}$ for $t = 1, \ldots, 30$ for all considered datasets (using stagewise updates). For each t, we then compare the coverage rate of the next condition (q_{t+1}) produced by obj_{alt} to the one produced by obj_{ogb} based on the same previous model $f_{\text{alt}}^{(t)}$. Importantly, we use branch-and-bound for all the objectives to avoid confounding through sub-optimal greedy solutions. As shown in Fig. 5, 91.0% of conditions identified by obj_{ogb} cover more data points than those identified by obj_{xgb}, and similarly 78.6% of the obj_{ogb} conditions cover more data points than those generated by obj_{gb}. In contrast, only 46.1% of conditions identified by obj_{ogb} cover more data points than the gradient sum objective obj_{gs}. These results are aligned with the theoretical expectation in terms of the influence of the coverage on the objective values where gradient sum is completely unaffected, whereas orthogonal gradient boosting has a denominator that tends to grow with coverage albeit less than the one of gradient boosting.

Computation time Finally, we investigate the computational overhead for generating rule ensembles with the proposed COB method with branch-and-bound search. Here, we use XGB as primary benchmark, because it uses the same more expensive search. Further, we investigate the effect of the efficient incremental computation of obj_{ogb} via Alg. 3 to assess its necessity. For both comparisons, we consider the time it takes for each test dataset to compute the largest rule ensemble generated in the main experiment (the one that exceeds complexity 50). The top of Fig 6. compares the efficient implementation of COB to XGB. We can see that the costs are in the same order of magnitude for almost all datasets. For 17 of the 34 datasets the overhead is within a factor of 2. For all but one extreme case (*wine*, overhead factor 26) the overhead is within a factor of 5. Comparing the two implementations of COB, the bottom of Fig. 6 the efficient implementation uses less than half of the time of the naive implementation for 24 out of 34 datasets. Particularly, for the datasets *gender* and *magic*, the efficient implementation improves the time more than 50-fold. Overall, the results confirm that branch-and-bound search is a practical algorithm in absolute terms: For 23 benchmark dataset, COB trains a model of complexity of 50 within one minute. Most of the other experiments run within 15 minutes except one dataset (telco churn) which require longer running time. See SI C for further details and comparisons.

5 CONCLUSION

The proposed fully corrective orthogonal boosting approach is a worthwhile alternative to previously published boosting variants for rule learning, especially when targeting a beneficial risk-complexity trade-off and an overall small number of rules. The present work provided a relatively detailed theoretical analysis of the newly developed rule objective function. However, some interesting questions were left open. While the presorting-based approach to the bounding function performs extremely well in synthetic experiments, a theoretical approximation guarantee for this algorithm has yet to be derived. Another interesting direction for future work is the extension of the introduced approximating subspace paradigm to the extreme gradient boosting approach, which, due to the utilization of higher order information, should principally be able to produce even better risk-complexity trade-offs.

Acknowledgements

The authors thank Daniel F. Schmidt for valuable discussions and the anonymous reviewers for their constructive feedback. This work was supported by the Australian Research Council (DP210100045). This work received support from the Cancer Research Center Cologne Essen (CCCE).

References

C. Bénard, G. Biau, S. Da Veiga, and E. Scornet. Interpretable random forests via rule extraction. In *International Conference on Artificial Intelligence and Statistics*, pages 937–945. PMLR, 2021.

M. Boley, B. R. Goldsmith, L. M. Ghiringhelli, and J. Vreeken. Identifying consistent statements about numerical data with dispersion-corrected subgroup discovery. *Data Mining and Knowledge Discovery*, 31(5):1391–1418, 2017.

M. Boley, S. Teshuva, P. Le Bodic, and G. I. Webb. Better short than greedy: Interpretable models through optimal rule boosting. In *Proceedings of the 2021 SIAM International Conference on Data Mining (SDM)*, pages 351–359. SIAM, 2021.

T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

W. W. Cohen and Y. Singer. A simple, fast, and effective rule learner. *AAAI/IAAI*, 99(335-342):3, 1999.

S. Dash, O. Gunluk, and D. Wei. Boolean decision rules via column generation. *Advances in neural information processing systems*, 31, 2018.

K. Dembczyński, W. Kotłowski, and R. Słowiński. Ender: a statistical framework for boosting decision rules. *Data Mining and Knowledge Discovery*, 21(1): 52–90, 2010.

J. H. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

J. H. Friedman and B. E. Popescu. Predictive learning via rule ensembles. *The annals of applied statistics*, pages 916–954, 2008.

J. Fürnkranz. Separate-and-conquer rule learning. *Artificial Intelligence Review*, 13(1):3–54, 1999.

H. Grosskreutz, S. Rüping, and S. Wrobel. Tight optimistic estimates for fast subgroup discovery. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 440–456. Springer, 2008.

T. Hastie and R. Tibshirani. *Generalized Additive Models*. Routledge, 1990. ISBN 9780203753781.

J. Kivinen and M. K. Warmuth. Boosting as entropy projection. In *Proceedings of the twelfth annual conference on Computational learning theory*, pages 134–144, 1999.

I. Kumar, C. Scheidegger, S. Venkatasubramanian, and S. Friedler. Shapley residuals: Quantifying the limits of the shapley value for explanations. *Advances in Neural Information Processing Systems*, 34:26598–26608, 2021.

H. Lakkaraju, S. H. Bach, and J. Leskovec. Interpretable decision sets: A joint framework for description and prediction. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1675–1684, 2016.

F. Lemmerich, M. Atzmueller, and F. Puppe. Fast exhaustive subgroup discovery with numerical target concepts. *Data Mining and Knowledge Discovery*, 30:711–762, 2016.

Y. Lou, R. Caruana, J. Gehrke, and G. Hooker. Accurate intelligible models with pairwise interactions. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 623–631, 2013.

L. Mason, J. Baxter, P. L. Bartlett, M. Frean, et al. Functional gradient techniques for combining hypotheses. *Advances in Neural Information Processing Systems*, pages 221–246, 1999.

P. McCullagh and J. A. Nelder. *Generalized linear models*. Routledge, 2019.

S. Morishita and J. Sese. Transversing itemset lattices with statistical metric pruning. In *Proceedings of the nineteenth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 226–236, 2000.

W. J. Murdoch, C. Singh, K. Kumbier, R. Abbasi-Asl, and B. Yu. Definitions, methods, and applications in interpretable machine learning. *Proceedings of the National Academy of Sciences*, 116(44):22071–22080, 2019.

M. T. Ribeiro, S. Singh, and C. Guestrin. "Why should I trust you?" Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

C. Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5): 206–215, 2019.

S. Shalev-Shwartz, N. Srebro, and T. Zhang. Trading accuracy for sparsity in optimization problems with sparsity constraints. *SIAM Journal on Optimization*, 20(6):2807–2832, 2010.

E. Strumbelj and I. Kononenko. An efficient explanation of individual classifications using game theory. *The Journal of Machine Learning Research*, 11:1–18, 2010.

T. Wang, C. Rudin, F. Doshi-Velez, Y. Liu, E. Klampfl, and P. MacNeille. A bayesian framework for learning rule sets for interpretable classification. *The Journal of Machine Learning Research*, 18(1):2357–2393, 2017.

D. Wei, S. Dash, T. Gao, and O. Gunluk. Generalized linear rule models. In *International Conference on Machine Learning*, pages 6687–6696. PMLR, 2019.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes]
   (b) Complete proofs of all theoretical results. [Yes]
   (c) Clear explanations of any assumptions. [Yes]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes]
   (b) The license information of the assets, if applicable. [Yes]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Yes]
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

A FULL PROOFS AND ADDITIONAL FORMAL STATEMENTS

Proposition 1. Let $\mathbf{f}^{\text{CGD}} = \arg\min\{R_\lambda(\mathbf{f}')\colon \mathbf{f}' \in \text{range}[\mathbf{Q}_{t-1}; \mathbf{g}]\}$ be the output vector of the ideal corrective gradient descent update in round t and

$$q = \arg\max_{q \in \mathcal{Q}} |\mathbf{g}_\perp^T \mathbf{q}| / \|\mathbf{q}_\perp\| \tag{13}$$

where for $\mathbf{v} \in \mathbb{R}^n$ we denote by $\mathbf{v}_\perp$ its projection onto the orthogonal complement of range $\mathbf{Q}_{t-1}$. Then the output vector $\mathbf{f}_q = \arg\min\{R_\lambda(\mathbf{f}')\colon \mathbf{f}' \in \text{range}[\mathbf{Q}_{t-1}; \mathbf{q}]\}$ dominates the optimal geometric approximation to $\mathbf{f}$ with respect to $\mathcal{Q}$, i.e., $R_\lambda(\mathbf{f}_q) \leq R_\lambda(\tilde{\mathbf{f}}_{\mathcal{Q}}^{\text{CGD}})$.

Proof. Let $\mathbf{Q} = \mathbf{Q}_{t-1}$ and $\mathbf{f} = [\mathbf{Q}; \mathbf{g}]\boldsymbol{\alpha}$ and $\tilde{\mathbf{f}} = [\mathbf{Q}; \mathbf{q}]\boldsymbol{\beta}$ for some arbitrary coefficient vectors $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathbb{R}^t$. Denoting by $\mathbf{v}_\parallel$ the projection of $\mathbf{v} \in \mathbb{R}^n$ onto the column space of $\mathbf{Q}$ and its orthogonal complement by $\mathbf{v}_\perp$, we can decompose the squared norm of the difference $\mathbf{f} - \tilde{\mathbf{f}}$ as

$$\begin{aligned}
\|\mathbf{f} - \tilde{\mathbf{f}}\|^2 &= \|[\mathbf{Q}; \mathbf{g}]\boldsymbol{\alpha} - [\mathbf{Q}; \mathbf{q}]\boldsymbol{\beta}\|^2 \\
&= \|[\mathbf{Q}; \mathbf{g}_\parallel + \mathbf{g}_\perp]\boldsymbol{\alpha} - [\mathbf{Q}; \mathbf{q}_\parallel + \mathbf{q}_\perp]\boldsymbol{\beta}\|^2 \\
&= \|[\mathbf{Q}; \mathbf{g}_\parallel]\boldsymbol{\alpha} + \alpha_t \mathbf{g}_\perp - [\mathbf{Q}; \mathbf{q}_\parallel]\boldsymbol{\beta} + \beta_t \mathbf{q}_\perp\|^2 \\
&= \|[\mathbf{Q}; \mathbf{g}_\parallel]\boldsymbol{\alpha} - [\mathbf{Q}; \mathbf{q}_\parallel]\boldsymbol{\beta}\|^2 + \|\alpha_t \mathbf{g}_\perp - \beta_t \mathbf{q}_\perp\|^2
\end{aligned}$$

where the last step follows from the Pythagorean theorem and the fact that $\alpha_t \mathbf{g}_\perp - \beta_t \mathbf{q}_\perp$ is an element from the orthogonal complement of range$[\mathbf{Q}; \mathbf{g}_\parallel] = \text{range}[\mathbf{Q}; \mathbf{q}_\parallel] = \text{range}\,\mathbf{Q}$. The equality of these ranges also implies that $\beta_1, \ldots, \beta_{t-1}$ can, for all choices of β_t, be chosen such that the left term of the error decomposition is 0. Setting $\gamma = \beta_t / \alpha_t$, it follows for the squared projection error of $\mathbf{f}$ onto range$[\mathbf{Q}, \mathbf{q}]$ that

$$\begin{aligned}
\min_{\boldsymbol{\beta} \in \mathbb{R}^t} \|\mathbf{f} - \tilde{\mathbf{f}}\|^2 &= \min_{\beta \in \mathbb{R}^t} \|\alpha_t \mathbf{g}_\perp - \beta_t \mathbf{h}_\perp\|^2 \\
&= \min_{\gamma \in \mathbb{R}^t} \alpha_t^2 \|\mathbf{g}_\perp - \gamma \mathbf{q}_\perp\|^2 \\
&= \min_{\gamma \in \mathbb{R}^t} \alpha_t^2 (\|\mathbf{g}_\perp\|^2 - 2\gamma \mathbf{q}_\perp^T \mathbf{g}_\perp + \gamma^2 \|\mathbf{q}_\perp\|^2)
\end{aligned}$$

and plugging in the minimizing $\gamma = \mathbf{q}_\perp^T \mathbf{g}_\perp / \|\mathbf{q}_\perp\|^2$

$$= \alpha^2 (\|\mathbf{g}_\perp\| - (\mathbf{g}_\perp^T \mathbf{q}_\perp)^2 / \|\mathbf{q}_\perp\|^2) \ ,$$

from which, noting that $\mathbf{g}_\perp^T \mathbf{q}_\perp = \mathbf{g}_\perp^T \mathbf{q}$, it follows that a query that maximizes $|\mathbf{g}_\perp^T \mathbf{q}| / \|\mathbf{q}_\perp\|$ minimizes the projection error. Hence, by choosing $\mathbf{f} = \mathbf{f}^{\text{GCD}}$ the ideal corrective gradient descent update, we have that $\tilde{\mathbf{f}}_{\mathcal{Q}}^{\text{GCD}} \in \text{range}[\mathbf{Q}; \mathbf{q}]$ and by definition of $\mathbf{f}_q$ we have $R_\lambda(\mathbf{f}_q) \leq R_\lambda(\tilde{\mathbf{f}}_{\mathcal{Q}}^{\text{GCD}})$ as required. □

Proposition 2. There is one-dimensional input data $\mathbf{X} \in \mathbb{R}^{5 \times 1}$ and a sequence of output data, $\mathbf{y}^{(m)} \in \mathbb{R}^5$, such that $\lim_{m \to \infty} R_{\text{ogb}}^{(m)} = 0$ but

$$\lim_{m \to \infty} R_{\text{gs}}^{(m)} = \lim_{m \to \infty} R_{\text{gb}}^{(m)} = \infty$$

where $R_*^{(m)} = R_0(f_*^{(3)})$ is the risk of the three-element rule ensemble f_* trained on $(\mathbf{X}, \mathbf{y}^{(m)})$ using obj_* and the corrective weight update.

Proof. We define $\mathbf{X}$ as $(1, 2, 3, 4, 5)$ and $\mathbf{y}^{(m)} = (-\alpha_m - \epsilon_m, \alpha_m, -3\alpha_m - \epsilon_m, \alpha_m + \epsilon_m, 2\alpha_m + \epsilon_m)$, where $\alpha_m, \epsilon_m \in \mathbb{R}$ are two arbitrary sequences with $\alpha_m \to \infty$ and $\epsilon_m \to 0$. We calculate the values of obj_{gs},obj_{gb} and obj_{ogb} for all possible queries to select the first, second and the third queries, as shown in Table 2 and Table 3. We use the query vectors to represent the queries in this proof.

For the gradient sum objective, according to Table 2, the first query identified is either the one with outputs $\mathbf{q}_{\text{gs}}^{(1)} = (1, 1, 1, 0, 0)$ or the one with output $\mathbf{q}_{\text{gs}}^{(1)'} = (0, 0, 0, 1, 1)$ for the five data points. In the first case, the weight of the query is $\beta_{\text{gs}}^{(1)} = (-\alpha_m - 2\epsilon_m/3)$. The gradient vector after adding this rule is $\mathbf{g}_{\text{gs}}^{(1)} =$

Table 2: Calculation of the objective functions for the first and the second query in proof of Proposition 2

		1ST QUERY		2ND QUERY						
$\mathbf{q}$	$\|\mathbf{q}\|$	$\text{obj}_{\text{gs}}(\mathbf{q})$ $\lvert\mathbf{q}^T\mathbf{g}^{(0)}\rvert$	$\text{obj}_{\text{gb}}(\mathbf{q})$ $\text{obj}_{\text{ogb}}(\mathbf{q})$	$\mathbf{q}_\perp$	$\|\mathbf{q}_\perp\|$	$\lvert\mathbf{q}^T\mathbf{g}^{(1)}_{\text{gb}}\rvert$ $\lvert\mathbf{q}^T_\perp\mathbf{g}^{(1)}_{\text{ogb}\perp}\rvert$	$\text{obj}_{\text{gs}}(\mathbf{q})$ 1ST CASE	$\text{obj}_{\text{gs}}(\mathbf{q})$ 2ND CASE	$\text{obj}_{\text{gb}}(\mathbf{q})$	$\text{obj}_{\text{ogb}}(\mathbf{q})$
$(1,0,0,0,0)$	1	$\alpha_m+\epsilon_m$	$\alpha_m+\epsilon_m$	$(1,0,0,0,0)$	1	$\alpha_m+\epsilon_m$	$\epsilon_m/3$	$\alpha_m+\epsilon_m$	$\alpha_m+\epsilon_m$	$\alpha_m+\epsilon_m$
$(0,1,0,0,0)$	1	α_m	α_m	$(0,1,0,0,0)$	1	α_m	$2\alpha_m+\frac{2\epsilon_m}{3}$	α_m	α_m	α_m
$(0,0,1,0,0)$	1	$3\alpha_m+\epsilon_m$	$3\alpha_m+\epsilon_m$	$(0,0,0,0,0)$	0	0	$2\alpha_m+\frac{\epsilon_m}{3}$	$3\alpha_m+\epsilon_m$	0	0
$(0,0,0,1,0)$	1	$\alpha_m+\epsilon_m$	$\alpha_m+\epsilon_m$	$(0,0,0,1,0)$	1	$\alpha_m+\epsilon_m$	$\alpha_m+\epsilon_m$	$\frac{\alpha_m}{2}$	$\alpha_m+\epsilon_m$	$\alpha_m+\epsilon_m$
$(0,0,0,0,1)$	1	$2\alpha_m+\epsilon_m$	$2\alpha_m+\epsilon_m$	$(0,0,0,0,1)$	1	$2\alpha_m+\epsilon_m$	$2\alpha_m+\epsilon_m$	$\frac{\alpha_m}{2}$	$2\alpha_m+\epsilon_m$	$2\alpha_m+\epsilon_m$
$(1,1,0,0,0)$	$\sqrt{2}$	ϵ_m	$\epsilon_m/\sqrt{2}$	$(1,1,0,0,0)$	$\sqrt{2}$	ϵ_m	$2\alpha_m+\epsilon_m/3$	ϵ_m	$\epsilon_m/\sqrt{2}$	$\epsilon_m/\sqrt{2}$
$(0,1,1,0,0)$	$\sqrt{2}$	$2\alpha_m+\epsilon_m$	$\frac{2\alpha_m+\epsilon_m}{\sqrt{2}}$	$(0,1,0,0,0)$	1	α_m	$\epsilon_m/3$	$2\alpha_m+\epsilon_m$	$\alpha_m/\sqrt{2}$	α_m
$(0,0,1,1,0)$	$\sqrt{2}$	$2\alpha_m$	$\sqrt{2}\alpha_m$	$(0,0,0,1,0)$	1	$\alpha_m+\epsilon_m$	$\alpha_m-\frac{2\epsilon_m}{3}$	$\frac{7\alpha_m}{2}+\epsilon_m$	$\frac{\alpha_m+\epsilon_m}{\sqrt{2}}$	$\alpha_m+\epsilon_m$
$(0,0,0,1,1)$	$\sqrt{2}$	$3\alpha_m+2\epsilon_m$	$\frac{3\alpha_m+2\epsilon_m}{\sqrt{2}}$	$(0,0,0,1,1)$	$\sqrt{2}$	$3\alpha_m+2\epsilon_m$	$3\alpha_m+2\epsilon_m$	0	$\frac{3\alpha_m+2\epsilon_m}{\sqrt{2}}$	$\frac{3\alpha_m+2\epsilon_m}{\sqrt{2}}$
$(1,1,1,0,0)$	$\sqrt{3}$	$3\alpha_m+2\epsilon_m$	$\frac{3\alpha_m+2\epsilon_m}{\sqrt{3}}$	$(1,1,0,0,0)$	$\sqrt{2}$	ϵ_m	0	$3\alpha_m+2\epsilon_m$	$\frac{\epsilon_m}{\sqrt{3}}$	$\frac{\epsilon_m}{\sqrt{2}}$
$(0,1,1,1,0)$	$\sqrt{3}$	α_m	$\frac{\alpha_m}{\sqrt{3}}$	$(0,1,0,1,0)$	$\sqrt{2}$	$2\alpha_m+\epsilon_m$	$\alpha_m+\frac{4\epsilon_m}{3}$	$\frac{5\alpha_m}{2}+\epsilon_m$	$\frac{2\alpha_m+\epsilon_m}{\sqrt{3}}$	$\frac{2\alpha_m+\epsilon_m}{\sqrt{2}}$
$(0,0,1,1,1)$	$\sqrt{3}$	ϵ_m	$\frac{\epsilon_m}{\sqrt{3}}$	$(0,0,0,1,1)$	$\sqrt{2}$	$3\alpha_m+2\epsilon_m$	$\alpha_m+\frac{5\epsilon_m}{3}$	$3\alpha_m+\epsilon_m$	$\frac{3\alpha_m+2\epsilon_m}{\sqrt{3}}$	$\frac{3\alpha_m+2\epsilon_m}{\sqrt{2}}$
$(1,1,1,1,0)$	2	$2\alpha_m+\epsilon_m$	$\alpha_m+\frac{\epsilon_m}{2}$	$(1,1,0,1,0)$	$\sqrt{3}$	α_m	$\alpha_m+\epsilon_m$	$\frac{7\alpha_m}{2}+2\epsilon_m$	$\frac{\alpha_m}{2}$	$\frac{\alpha_m}{\sqrt{3}}$
$(0,1,1,1,1)$	2	$\alpha_m+\epsilon_m$	$\frac{\alpha_m+\epsilon_m}{2}$	$(0,1,0,1,1)$	$\sqrt{3}$	$4\alpha_m+2\epsilon_m$	$3\alpha_m+\frac{7\epsilon_m}{3}$	$2\alpha_m+\epsilon_m$	$2\alpha_m+\epsilon_m$	$\frac{4\alpha_m+\epsilon_m}{\sqrt{3}}$
$(1,1,1,1,1)$	$\sqrt{5}$	0	0	$(1,1,0,1,1)$	2	$3\alpha_m+\epsilon_m$	$3\alpha_m+2\epsilon_m$	$3\alpha_m+2\epsilon_m$	$\frac{3\alpha_m+\epsilon_m}{\sqrt{5}}$	$\frac{3\alpha_m+\epsilon_m}{2}$

$(-\epsilon_m/3, 2\alpha_m+2\epsilon_m/3, -2\alpha_m-\epsilon_m/3, \alpha_m+\epsilon_m, 2\alpha_m+\epsilon_m)$. The second query selected is $\mathbf{q}^{(2)}_{\text{gs}} = (0,1,1,1,1)$ according to the objective values calculated in Table 2. After adding this query, the corrected weight vector is $\beta^{(2)}_{\text{gs}} = (-7\alpha_m/4 - 5\epsilon_m/4, 9\alpha_m/8 + 7\epsilon_m/8)$, and the gradient vector is $\mathbf{g}^{(2)}_{\text{gs}} = ((3\alpha_m+\epsilon_m)/4, (13\alpha_m + 3\epsilon_m)/8, -(19\alpha_m+5\epsilon_m)/8, -(\alpha_m-\epsilon_m)/8, (7\alpha_m+\epsilon_m)/8)$. Then, we calculate the values of $\text{obj}_{\text{gs}}(\mathbf{q})$ in Table 3, and the third query selected is $\mathbf{q}^{(3)}_{\text{gs}} = (0,0,1,1,0)$. The corrected weight vector is $\beta^{(3)}_{\text{gs}} = (-(7\alpha_m+5\epsilon_m)/4, (19\alpha_m+9\epsilon_m)/8, -(5\alpha_m+\epsilon_m)/2)$. The output vector of the rule ensemble is

$$-\frac{7\alpha_m+5\epsilon_m}{4}\begin{pmatrix}1\\1\\1\\0\\0\end{pmatrix}+\frac{19\alpha_m+9\epsilon_m}{8}\begin{pmatrix}0\\1\\1\\1\\1\end{pmatrix}-\frac{5\alpha_m+\epsilon_m}{2}\begin{pmatrix}0\\0\\1\\1\\0\end{pmatrix}=\begin{pmatrix}-(7\alpha_m+5\epsilon_m)/4\\(5\alpha_m-\epsilon_m)/8\\-5(3\alpha_m+\epsilon_m)/8\\-(\alpha_m-5\epsilon_m)/8\\(19\alpha_m+9\epsilon)/8\end{pmatrix}.$$

The gradient vector is $\mathbf{g}^{(3)}_{\text{gs}} = ((3\alpha_m+\epsilon_m)/4, (3\alpha_m+\epsilon_m)/8, -(9\alpha_m+3\epsilon_m)/8, (9\alpha_m+3\epsilon_m)/8, -(3\alpha_m+\epsilon_m)/8)$. The empirical risk after adding three rules into the rule ensemble is

$$R^{(m)}_{\text{gs}} = \frac{3}{8}(3\alpha_m+\epsilon_m)^2.$$

In the second case ($\mathbf{q}^{(1)'}_{\text{gs}} = (0,0,0,1,1)$), the weight of the first query is $3\alpha_m/2+\epsilon_m$, and the gradient vector is $\mathbf{g}^{(1)'}_{\text{gs}} = (-\alpha_m-\epsilon_m, \alpha_m, -3\alpha_m-\epsilon_m, -\alpha_m/2, \alpha_m/2)$. The second query selected is $\mathbf{q}^{(2)'}_{\text{gs}} = (1,1,1,1,0)$. The corrected weight vector is $(2\alpha_m+9\epsilon_m, -\alpha_m-4\epsilon_m/7)$. The gradient vector after adding two rules is $\mathbf{g}^{(2)'}_{\text{gs}} = (-3\epsilon_m/7, 2\alpha_m+4\epsilon_m/7, -2\alpha_m-3\epsilon/7, 2\epsilon/7, -2\epsilon/7)$. The third query selected is $\mathbf{q}^{(3)'}_{\text{gs}} = (0,1,0,0,0)$. The

Table 3: Calculation of the objective functions for the third query in proof of Proposition 2

$\mathbf{q}$	$\|\mathbf{q}\|$	$\mathbf{q}_\perp$	$\|\mathbf{q}_\perp\|_2$	$\left\|\mathbf{q}_\perp^T \mathbf{g}_{\text{ogb}\perp}^{(1)}\right\|$	$\left\|\mathbf{q}^T \mathbf{g}_{\text{gb}}^{(2)}\right\|$	$\text{obj}_{\text{gs}}(\mathbf{q})$ 1ST CASE	$\text{obj}_{\text{gs}}(\mathbf{q})$ 2ND CASE	$\text{obj}_{\text{gb}}(\mathbf{q})$	$\text{obj}_{\text{ogb}}(\mathbf{q})$
$(1,0,0,0,0)$	1	$(1,0,0,0,0)$	1	$\alpha_m+\epsilon_m$	$\alpha_m+\epsilon_m$	$(3\alpha_m+\epsilon_m)/4$	$3\epsilon_m/7$	$\alpha_m+\epsilon_m$	$\alpha_m+\epsilon_m$
$(0,1,0,0,0)$	1	$\left(0,\frac{2}{3},0,-\frac{1}{3},-\frac{1}{3}\right)$	$\sqrt{\frac{2}{3}}$	$\frac{\alpha_m+2\epsilon_m}{3}$	α_m	$\frac{13\alpha_m+3\epsilon_m}{8}$	$2\alpha_m+\frac{4\epsilon_m}{7}$	α_m	$\frac{\alpha_m+2\epsilon_m}{\sqrt{6}}$
$(0,0,1,0,0)$	1	$(0,0,0,0,0)$	0	0	0	$\frac{19\alpha_m+5\epsilon_m}{8}$	$2\alpha_m+\frac{3\epsilon_m}{7}$	0	0
$(0,0,0,1,0)$	1	$\left(0,-\frac{1}{3},0,\frac{2}{3},-\frac{1}{3}\right)$	$\sqrt{\frac{2}{3}}$	$\frac{\alpha_m-\epsilon_m}{3}$	$\frac{\alpha_m}{2}$	$\frac{\alpha_m-\epsilon_m}{8}$	$\frac{2\epsilon_m}{7}$	$\frac{\alpha_m}{2}$	$\frac{\alpha_m-\epsilon_m}{\sqrt{6}}$
$(0,0,0,0,1)$	1	$\left(0,-\frac{1}{3},0,-\frac{1}{3},\frac{2}{3}\right)$	$\sqrt{\frac{2}{3}}$	$\frac{2\alpha_m+\epsilon_m}{3}$	$\frac{\alpha_m}{2}$	$\frac{7\alpha_m+\epsilon_m}{8}$	$\frac{2\epsilon_m}{7}$	$\frac{\alpha_m}{2}$	$\frac{2\alpha_m+\epsilon_m}{\sqrt{6}}$
$(1,1,0,0,0)$	$\sqrt{2}$	$\left(1,\frac{2}{3},0,-\frac{1}{3},-\frac{1}{3}\right)$	$\sqrt{\frac{5}{3}}$	$\frac{4\alpha_m+5\epsilon_m}{3}$	ϵ_m	$\frac{19\alpha_m+5\epsilon_m}{8}$	$2\alpha_m+\frac{\epsilon_m}{7}$	$\frac{\epsilon_m}{\sqrt{2}}$	$\frac{4\alpha_m+5\epsilon_m}{\sqrt{15}}$
$(0,1,1,0,0)$	$\sqrt{2}$	$\left(0,\frac{2}{3},0,-\frac{1}{3},-\frac{1}{3}\right)$	$\sqrt{\frac{2}{3}}$	$\frac{\alpha_m+2\epsilon_m}{3}$	α_m	$\frac{3\alpha_m+\epsilon_m}{4}$	$\frac{\epsilon_m}{7}$	$\frac{\alpha_m}{\sqrt{2}}$	$\frac{\alpha_m+2\epsilon_m}{\sqrt{6}}$
$(0,0,1,1,0)$	$\sqrt{2}$	$\left(0,-\frac{1}{3},0,\frac{2}{3},-\frac{1}{3}\right)$	$\sqrt{\frac{2}{3}}$	$\frac{\alpha_m-\epsilon_m}{3}$	$\frac{\alpha_m}{2}$	$\frac{5\alpha_m+\epsilon_m}{2}$	$2\alpha_m+\frac{\epsilon_m}{7}$	$\frac{\alpha_m}{2\sqrt{2}}$	$\frac{\alpha_m-\epsilon_m}{\sqrt{6}}$
$(0,0,0,1,1)$	$\sqrt{2}$	$\left(0,-\frac{2}{3},0,\frac{1}{3},\frac{1}{3}\right)$	$\sqrt{\frac{2}{3}}$	$\frac{\alpha_m+2\epsilon_m}{3}$	0	$\frac{3\alpha_m+\epsilon_m}{4}$	0	0	$\frac{\alpha_m+2\epsilon_m}{\sqrt{6}}$
$(1,1,1,0,0)$	$\sqrt{3}$	$\left(1,\frac{2}{3},0,-\frac{1}{3},-\frac{1}{3}\right)$	$\sqrt{\frac{5}{3}}$	$\frac{4\alpha_m+5\epsilon_m}{3}$	ϵ_m	0	$\frac{2\epsilon_m}{7}$	$\frac{\epsilon_m}{\sqrt{3}}$	$\frac{4\alpha_m+5\epsilon_m}{\sqrt{15}}$
$(0,1,1,1,0)$	$\sqrt{3}$	$\left(0,\frac{1}{3},0,\frac{1}{3},-\frac{2}{3}\right)$	$\sqrt{\frac{2}{3}}$	$\frac{2\alpha_m+\epsilon_m}{3}$	$\frac{\alpha_m}{2}$	$\frac{7\alpha_m+\epsilon_m}{8}$	$\frac{3\epsilon_m}{7}$	$\frac{\alpha_m}{2\sqrt{3}}$	$\frac{2\alpha_m+\epsilon_m}{\sqrt{6}}$
$(0,0,1,1,1)$	$\sqrt{3}$	$\left(0,-\frac{2}{3},0,\frac{1}{3},\frac{1}{3}\right)$	$\sqrt{\frac{2}{3}}$	$\frac{\alpha_m+2\epsilon_m}{3}$	0	$\frac{13\alpha_m+3\epsilon_m}{8}$	$2\alpha_m+\frac{3\epsilon_m}{7}$	0	$\frac{\alpha_m+2\epsilon_m}{\sqrt{6}}$
$(1,1,1,1,0)$	2	$\left(1,\frac{1}{3},0,\frac{1}{3},-\frac{2}{3}\right)$	$\sqrt{\frac{5}{3}}$	$\frac{5\alpha_m+4\epsilon_m}{3}$	$\frac{\alpha_m}{2}+\epsilon_m$	$\frac{\alpha_m-\epsilon_m}{8}$	0	$\frac{\alpha_m}{4}+\frac{\epsilon_m}{2}$	$\frac{5\alpha_m+4\epsilon_m}{\sqrt{15}}$
$(0,1,1,1,1)$	2	$(0,0,0,0,0)$	0	0	α_m	0	$\epsilon/7$	$\alpha_m/2$	0
$(1,1,1,1,1)$	$\sqrt{5}$	$(1,0,0,0,0)$	1	$\alpha_m+\epsilon_m$	ϵ_m	$(3\alpha_m+\epsilon_m)/4$	$2\epsilon/7$	$\epsilon_m/\sqrt{5}$	$\alpha_m+\epsilon_m$

corrected weight vector is $((12\alpha_m+7\epsilon_m)/5, -(9\alpha_m+4\epsilon_m)/5, 2(7\alpha_m+2\epsilon_m)/5)$. The output vector of the rule ensemble is

$$\frac{12\alpha_m+7\epsilon_m}{5}\begin{pmatrix}0\\0\\0\\1\\1\end{pmatrix}-\frac{9\alpha_m+4\epsilon_m}{5}\begin{pmatrix}1\\1\\1\\1\\0\end{pmatrix}+\frac{2}{5}(7\alpha_m+2\epsilon_m)\begin{pmatrix}0\\1\\0\\0\\0\end{pmatrix}=\begin{pmatrix}-(9\alpha_m+4\epsilon_m)/5\\\alpha_m\\-(9\alpha_m+4\epsilon_m)/5\\3(\alpha_m+\epsilon_m)/5\\(12\alpha_m+7\epsilon_m)/5\end{pmatrix}.$$

The empirical risk after adding three rules is

$$R_{\text{gs}}^{(m)}=\frac{2}{5}\left(6\alpha_m^2+2\alpha_m\epsilon_m+\epsilon_m^2\right).$$

For gradient boosting objective, the first query selected is $\mathbf{q}_{\text{gb}}^{(1)}=(0,0,1,0,0)$ according to Table 2. Its weight is $-3\alpha_m-\epsilon_m$. The gradient after adding this rule is $\mathbf{g}_{\text{gb}}^{(1)}=(-\alpha_m-\epsilon_m,\alpha_m,0,\alpha_m+\epsilon_m,2\alpha_m+\epsilon_m)$. The second query selected by the gradient boosting objective is $\mathbf{q}_{\text{gb}}^{(2)}=(0,0,0,1,1)$. The weight vector after correction is $(-3\alpha_m-\epsilon_m, 3\alpha_m/2+\epsilon_m)$. The gradient becomes $\mathbf{g}_{\text{gb}}^{(2)}=(-\alpha_m-\epsilon_m,\alpha_m,0,-\alpha_m/2,\alpha_m/2)$ after adding the second rule. Then, according to Table 3, the third query selected by gradient boosting objective is $\mathbf{q}_{\text{gb}}^{(3)}=(1,0,0,0,0)$. The corrected weight vector is $(-3\alpha_m-\epsilon_m, 3\alpha_m/2+\epsilon_m, -\alpha_m-\epsilon_m)$. The output vector of the rule ensemble is

$$-(3\alpha_m+\epsilon_m)\begin{pmatrix}0\\0\\1\\0\\0\end{pmatrix}+\left(\frac{3\alpha_m}{2}+\epsilon_m\right)\begin{pmatrix}0\\0\\0\\1\\1\end{pmatrix}-(\alpha_m+\epsilon_m)\begin{pmatrix}1\\0\\0\\0\\0\end{pmatrix}=\begin{pmatrix}-\alpha_m-\epsilon_m\\0\\-3\alpha_m-\epsilon_m\\3\alpha_m/2+\epsilon_m\\3\alpha_m/2+\epsilon_m\end{pmatrix}.$$

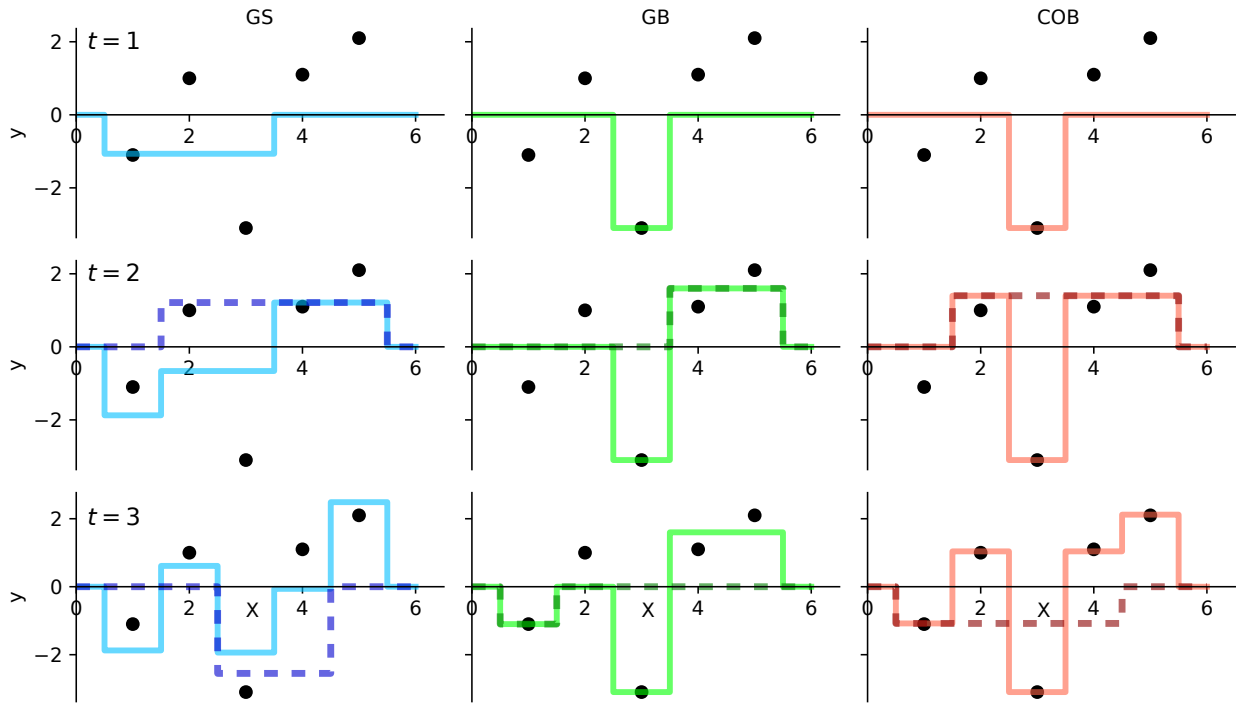

Figure 7: Visualisation of the generating of rules by different objective functions (in column) in each iteration (row) for the dataset used in proof of Proposition 2. The solid lines show the output of rule ensembles in each iteration. The dashed lines show the output of the rule generated in each iteration.

The gradient vector is $\mathbf{g}_{\text{gb}}^{(3)} = (0, \alpha_m, 0, -\alpha_m/2, \alpha_m/2)$. The empirical risk after adding three rules into the ensemble is

$$R_{\text{gb}}^{(m)} = 3\alpha_m^2/2.$$

In the first iteration, the orthogonal boosting objective selects the same query as the gradient boosting, since their objective values are the same, so their weights, outputs, gradients are also the same. We calculate the orthogonal projections $\mathbf{q}_\perp$, their lengths $\|\mathbf{q}_\perp\|$ and the objective values $\text{obj}_{\text{ogb}}(\mathbf{q})$ as Table 2. The second query selected is $\mathbf{q}_{\text{ogb}}^{(2)} = (0, 1, 1, 1, 1)$, and the weight vector after correction is $(-(13\alpha_m + 5\epsilon_m)/3, (4\alpha_m + 2\epsilon_m)/3)$. The gradient vector after adding the first two rules is $\mathbf{g}_{\text{ogb}}^{(2)} = (-(\alpha_m + \epsilon_m), -(\alpha_m + \epsilon_m)/3, 0, -(\alpha_m - \epsilon_m)/3, (2\alpha_m + \epsilon_m)/3)$. According to Table 3, the third query selected by the orthogonal boosting objective is $\mathbf{q}_{\text{ogb}}^{(3)} = (1, 1, 1, 1, 0)$. The weight vector is now $(-4\alpha_m - 7\epsilon_m/5, 2\alpha_m + 6\epsilon_m/5, -\alpha_m - 4\epsilon_m/5)$. The output vector of the rule ensemble is

$$-\left(4\alpha_m + \frac{7\epsilon_m}{5}\right)\begin{pmatrix}0\\0\\1\\0\\0\end{pmatrix} + \left(2\alpha_m + \frac{6\epsilon_m}{5}\right)\begin{pmatrix}0\\1\\1\\1\\1\end{pmatrix} - \left(\alpha_m + \frac{4\epsilon_m}{5}\right)\begin{pmatrix}1\\1\\1\\1\\0\end{pmatrix} = \begin{pmatrix}-\alpha_m - 4\epsilon_m/5\\ \alpha_m + 2\epsilon_m/5\\ -3\alpha_m - \epsilon_m\\ \alpha_m + 2\epsilon_m/5\\ 2\alpha_m + 6\epsilon_m/5\end{pmatrix}.$$

The gradient vector is $\mathbf{g}_{\text{ogb}}^{(3)} = (-\epsilon_m/5, -2\epsilon_m/5, 0, 3\epsilon_m/5, -\epsilon_m/5)$. The empirical risk after adding three rules into the ensemble is

$$R_{\text{ogb}}^{(m)} = 3\epsilon_m^2/5.$$

Since $\alpha_m \to \infty$ and $\epsilon_m \to 0$,

$$\lim_{m\to\infty} R_{\text{gs}}^{(m)} = \lim_{m\to\infty} 3\left(3\alpha_m + \epsilon_m\right)^2/8 = \infty,$$

or

$$\lim_{m\to\infty} R_{\text{gs}}^{(m)} = \lim_{m\to\infty} 2\left(6\alpha_m^2 + 2\alpha_m\epsilon_m + \epsilon_m^2\right)/5 = \infty,$$

$$\lim_{m\to\infty} R_{\text{gb}}^{(m)} = \lim_{m\to\infty} 3\alpha_m^2/2 = \infty,$$

and

$$\lim_{m\to\infty} R_{\text{ogb}}^{(m)} = \lim_{m\to\infty} 3\epsilon_m^2/5 = 0.$$

The rules generated by each objective function in each step is visualised as Figure 7.

□

Proposition 3. Given a gradient vector $\mathbf{g} \in \mathbb{R}^n$, an orthonormal basis $\mathbf{o}_1, \ldots, \mathbf{o}_t \in \mathbb{R}^n$ of the subspace spanned by the queries of the first t rules, and a sub-selection of l candidate points $\sigma\colon [l] \to [n]$, the prefix value problem (15) can be solved in time $O(tl)$.

Proof. To see the claim, we first rewrite the objective value for the i-th prefix as

$$\frac{\left|\mathbf{g}_\perp^T \mathbf{q}^{(i)}\right|}{\|\mathbf{q}_\perp^{(i)}\| + \epsilon} = \frac{\left|\mathbf{g}_\perp^T \mathbf{q}^{(i)}\right|}{\sqrt{\|\mathbf{q}^{(i)}\|^2 - \|\mathbf{q}_\parallel^{(i)}\|^2} + \epsilon} \ .$$

The value of $\|\mathbf{q}^{(i)}\|^2$ is trivially given as $|\mathrm{I}(\mathbf{q}^{(i)})| = i$, and $\mathbf{g}^T\mathbf{q}^i$ can be easily computed for all $i \in [l]$ in time $O(n)$ via cumulative summation. Finally we can reduce the problem of computing the (squared) norms of the l projected prefixes to computing the t (squared) norms of the prefixes on the subspaces given by the individual orthonormal basis vectors via

$$\|\mathbf{q}_\parallel^{(i)}\|^2 = \left\| \sum_{k=1}^{t} \mathbf{o}_k \mathbf{o}_k^T \mathbf{q}^{(i)} \right\|^2 = \sum_{k=1}^{t} \|\mathbf{o}_k \mathbf{o}_k^T \mathbf{q}^{(i)}\|^2 \ .$$

Each of these t sequences of (squared) norms can be computed in time $O(n)$ by rewriting

$$\begin{aligned}
\|\mathbf{o}_k \mathbf{o}_k^T \mathbf{q}^{(i)}\| &= \left\| \mathbf{o}_k \mathbf{o}_k^T \left(\sum_{j=1}^{i} \mathbf{e}_{\sigma(j)} \right) \right\| \\
&= \|\mathbf{o}_k\| \left| \sum_{j=1}^{i} \mathbf{o}_k^T \mathbf{e}_{\sigma(j)} \right| \\
&= \left| \sum_{j=1}^{i} o_{k,\sigma(j)} \right|
\end{aligned}$$

where the last equality shows how an $O(n)$-computation is achieved via cumulative summation of the k-th basis vector elements in the order given by σ. □

Proposition 4. Let $\mathbf{g}$ be the gradient vector after the application of the weight correction step (7) for selected queries $\mathbf{q}_1, \ldots, \mathbf{q}_t$. If the regularisation parameter is 0, then $\mathbf{g} \perp \operatorname{span}\{\mathbf{q}_1, \ldots, \mathbf{q}_t\}$.

Proof. After the weight correction step $\boldsymbol{\beta}$ is a stationary point of $R(\mathbf{Q}(\cdot))$, i.e., we have for all $j \in [t]$

$$0 = \frac{\partial R(\mathbf{Q}\boldsymbol{\beta})}{\partial \beta_j} = \sum_{i=1}^{n} \frac{\partial\, l(\tilde{\mathbf{q}}_i^T \boldsymbol{\beta}, y_i)}{\partial \beta_j} = \sum_{i=1}^{n} q_{ij} \underbrace{\frac{\partial\, l(\tilde{\mathbf{q}}_i^T \boldsymbol{\beta}, y_i)}{\partial\, \tilde{\mathbf{q}}_i^T \boldsymbol{\beta}}}_{g_i} = \mathbf{q}_j^T \mathbf{g} \ .$$

□

Proposition 5. Let $\mathbf{Q} = [\mathbf{q}_1, \ldots, \mathbf{q}_{t-1}] \in \mathbb{R}^{n\times(t-1)}$ be the selected query matrix and $\mathbf{g}$ the corresponding gradient vector after full weight correction, and let us denote by $\mathbf{q} = \mathbf{q}_\perp + \mathbf{q}_\parallel$ the orthogonal decomposition of $\mathbf{q}$ with respect to range $\mathbf{Q}$. Then we have for a maximizer $\mathbf{q}^*$ of the **orthogonal gradient boosting objective** $\mathrm{obj}_{\text{ogb}}(q) = |\mathbf{g}_\perp^T \mathbf{q}|/(\|\mathbf{q}_\perp\| + \epsilon)$:

a) For $\epsilon \to 0$, $\text{span}\{\mathbf{q}_1, \ldots, \mathbf{q}_{t-1}, \mathbf{q}^*\}$ is the best approximation to $\text{span}\{\mathbf{q}_1, \ldots, \mathbf{q}_{t-1}, \mathbf{g}\}$.

b) For $\epsilon \to \infty$, $\mathbf{q}^*$ maximizes obj_{gs} and any maximizer of obj_{gs} maximizes obj_{ogb}.

c) For $\epsilon = 0$ and $\|\mathbf{q}_\perp\| > 0$, the ratio $(\text{obj}_{\text{ogb}}(q)/\text{obj}_{\text{gb}}(q))^2$ is equal to $1 + (\|\mathbf{q}_\parallel\|/\|\mathbf{q}_\perp\|)^2$.

d) The objective value $\text{obj}_{\text{ogb}}(q)$ is upper bounded by $\|\mathbf{g}_\perp\|$.

Proof. a) If $\epsilon \to 0$, then $\text{obj}_{\text{ogb}}(q) \to \dfrac{|g_\perp^T q|}{\|q_\perp\|}$. If $\mathbf{q}^*$ is a maximizer of obj_{ogb}, then as shown in Lemma 4.1, $\mathbf{q}^*$ minimises the minimum distance from all

$$\mathbf{f} \in \text{span}\{\mathbf{q}_1, \cdots, \mathbf{q}_{t-1}, \mathbf{g}\}$$

to the subspace of

$$\text{span}\{\mathbf{q}_1, \cdots, \mathbf{q}_{t-1}, \mathbf{q}^*\}.$$

Therefore, the subspace spanned by $[\mathbf{q}_1, \cdots, \mathbf{q}_{t-1}, \mathbf{q}^*]$ is the best approximation to the subspace spanned by $[\mathbf{q}_1, \cdots, \mathbf{q}_{t-1}, \mathbf{g}]$.

b) Let q_1 and q_2 be any two queries and denote by $\text{obj}_{\text{ogb}}^{(\epsilon)}(q)$ the obj_{ogb}-value of q for a specific ϵ. Then

$$\begin{aligned}
&\lim_{\epsilon\to\infty} \epsilon \left(\text{obj}_{\text{ogb}}^{(\epsilon)}(q_1) - \text{obj}_{\text{ogb}}^{(\epsilon)}(q_2)\right) \\
=&\lim_{\epsilon\to\infty} \epsilon \left(\frac{|g_\perp^T \mathbf{q}_1|}{\|\mathbf{q}_1^\perp\| + \epsilon} - \frac{|g_\perp^T \mathbf{q}_2|}{\|\mathbf{q}_2^\perp\| + \epsilon}\right) \\
=&\lim_{\epsilon\to\infty} \left(\frac{|g_\perp^T \mathbf{q}_1|}{\|\mathbf{q}_1^\perp\|/\epsilon + 1} - \frac{|g_\perp^T \mathbf{q}_2|}{\|\mathbf{q}_2^\perp\|/\epsilon + 1}\right) \\
=&|g_\perp^T \mathbf{q}_1| - |g_\perp^T \mathbf{q}_2| \\
=&\text{obj}_{\text{gs}}(q_1) - \text{obj}_{\text{gs}}(q_2)
\end{aligned}$$

Thus for large enough ϵ, the signs of $\text{obj}_{\text{ogb}}^{(\epsilon)}(q_1) - \text{obj}_{\text{ogb}}^{(\epsilon)}(q_2)$ and $\text{obj}_{\text{gs}}(q_1) - \text{obj}_{\text{gs}}(q_2)$ agree. Therefore, a query q is a obj_{gs}-maximizer, i.e., $\text{obj}_{\text{gs}}(q) \geq \text{obj}_{\text{gs}}(q')$ for all $q' \in \mathcal{Q}$, if and only if q is a obj_{ogb}-maximizer, i.e., $\text{obj}_{\text{ogb}}(q) \geq \text{obj}_{\text{ogb}}(q')$ for all $q' \in \mathcal{Q}$.

c) If $\epsilon = 0$ and $\|q_\perp\| > 0$, then

$$\begin{aligned}
\left(\frac{\text{obj}_{\text{ogb}}(q)}{\text{obj}_{\text{gb}}(q)}\right)^2 &= \frac{\dfrac{|\mathbf{g}_\perp^T \mathbf{q}|^2}{\|\mathbf{q}_\perp\|^2}}{\dfrac{|\mathbf{g}_\perp^T \mathbf{q}|^2}{\|\mathbf{q}\|^2}} = \frac{\|\mathbf{q}\|^2}{\|\mathbf{q}_\perp\|^2} \\
&= \frac{\|\mathbf{q}_\parallel\|^2 + \|\mathbf{q}_\perp\|^2}{\|\mathbf{q}_\perp\|^2} \\
&= 1 + \left(\frac{\|\mathbf{q}_\parallel\|}{\|\mathbf{q}_\perp\|}\right)^2
\end{aligned}$$

d) If we divide the numerator and denominator of $\text{obj}_{\text{ogb}}(\mathbf{q})$ with $\|\mathbf{q}_bot\|$, then we can get

$$\begin{aligned}
\text{obj}_{\text{ogb}}(\mathbf{q}) &= \frac{|\mathbf{g}_\perp^T \mathbf{q}|}{\|q_\perp\| + \epsilon} \\
&= \frac{\dfrac{|\mathbf{g}_\perp^T \mathbf{q}_\perp|}{\|\mathbf{q}_\perp\|}}{1 + \dfrac{\epsilon}{\|\mathbf{q}_\perp\|}}
\end{aligned}$$

according to the Cauchy–Schwarz inequality, $\frac{|\mathbf{g}_\perp^T \mathbf{q}|}{\|\mathbf{q}_\perp\|} \leq \frac{\|\mathbf{g}_\perp\|\|\mathbf{q}_\perp\|}{\|\mathbf{q}_\perp\|} = \|\mathbf{g}_\perp\|$, so,

$$\text{obj}_{\text{ogb}}(\mathbf{q}) \leq \frac{\|\mathbf{g}_\perp\|}{1 + \frac{\epsilon}{\|\mathbf{q}_\perp\|}}$$

as $\|\mathbf{q}_\perp\|$ is upper bounded by the number of data points n,

$$\text{obj}_{\text{ogb}}(\mathbf{q}) \leq \frac{\|\mathbf{g}_\perp\|}{1 + \frac{\epsilon}{n}}$$
$$\text{obj}_{\text{ogb}}(\mathbf{q}) \leq \|\mathbf{g}_\perp\|.$$

□

B GREEDY APPROXIMATION TO BOUNDING FUNCTION

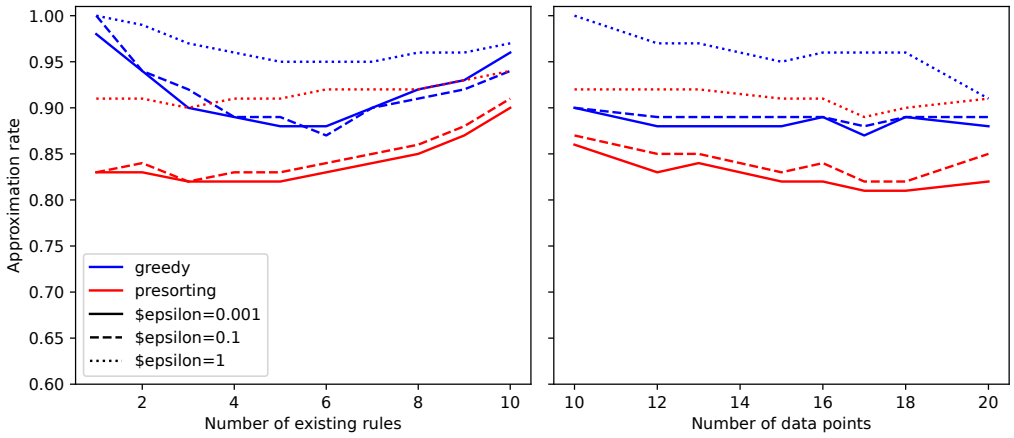

Figure 8: The approximation rates for different number of existing rules (left) and data points (right) with 99% success rate.

The branch-and-bound search described in Section 3.3 requires an efficient way of calculating the value of $\text{bnd}(\mathbf{q}) = \max\{obj(\mathbf{q}') : I(\mathbf{q}') \subseteq I(\mathbf{q}), \mathbf{q}' \in \{0,1\}^n\}$, where $I(\mathbf{q}) = \{i : \mathbf{q}(x_i) = 1, 1 \leq i \leq n\}$. It is too expensive to enumerate all possible $\mathbf{q}'$s as there are 2^n cases in the worst case. One solution to this problem is that we can relax the constraint $\mathbf{q}' \in \{0,1\}^n$ to $\mathbf{q}' \in [0,1]^n$ and it can be solved by quadratic programming. However, this would render the branch-and-bound search computationally very expensive. Instead, we investigate here using greedy approximations to identifying the optimal point sets. This does not yield an admissible bounding function, and thus does not guarantee to identify the optimal query, but can still lead to good approximation ratios in practice and is computationally inexpensive.

In particular, we are investigating two greedy variants: the fast prefix greedy approach described in the main text, and a slower full greedy approach that works as follows. Given a query $\mathbf{q}'^{(t-1)} \leq \mathbf{q}$, we need to find the data point selected by $\mathbf{q}$ which maximise the objective function, and use it with $\mathbf{q}'^{(t-1)}$ to form a $\mathbf{q}'(t)$.

$$i_*^{(t)} = \underset{i \in I(q) - I(q'^{(t-1)})}{\arg\max} \frac{\mathbf{g}^T \left(\mathbf{q}'^{(t-1)} + \mathbf{e}_i\right)}{\|\left(\mathbf{q}'^{(t-1)} + \mathbf{e}_i\right)_\perp\| + \epsilon}.$$

Table 4: The ratio of instances (15 data points and 5 existing rules) which reaches certain approximation rates.

Approx. rate	ϵ=0.001	ϵ=0.1	ϵ=1
75%	100.00%	100.00%	100.00%
80%	99.66%	99.71%	100.00%
85%	96.21%	98.05%	99.93%
90%	88.11%	90.80%	99.34%
95%	65.54%	70.15%	92.43%
100%	33.20%	36.87%	63.28%

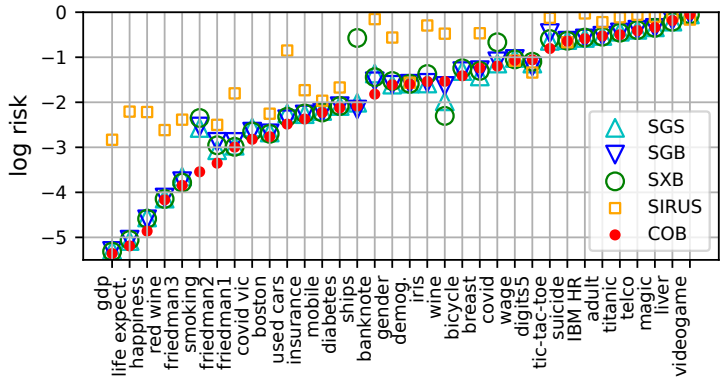

Figure 9: Comparison of log training risks over different datasets. The datasets are ordered by the training risks of COB.

where $0 \leq t \leq |I(\mathbf{q})|$, $\mathbf{q}'^{(0)} = \mathbf{0}$ and $\mathbf{q}'^{(t)} = \mathbf{q}'^{(t-1)} + \mathbf{e}_{i_*^{(t)}}$. We use the maximum value of $\text{obj}(\mathbf{q}'^{(t)})$ as the bounding value for query $\mathbf{q}$. The computation time complexity level of this approach is $O(n^2)$ for each query.

We now investigate how the number of existing rules and number of data points affect the approximation ratios achieved by both greedy approaches. To analyse the impact of number of existing rules, we fix the number of data points as 15, and we vary the number of existing rules from 1 to 10. To show the trend of the approximation ratios in terms of number of data points, we make the number of existing queries as 5, and the numbers of data points are from 10 to 20. For each pair of number of data points and number of existing rules, we generate 2000 groups of initial queries and initial gradient vectors. In each initial query, the output of each data point has a probability of 0.5 being 1 and 0. The initial gradient vector is generated by a standard normal distribution whose dimension is the same as the number of data points, and then it is projected onto the subspace orthogonal to the existing queries. We test these 2000 instances to see the difference between the approximation of $\text{bnd}(\mathbf{q})$ obtained by the full greedy approach, the pre-sorting greedy approach, and the actual optimal objective values (obtained by a brute-force approach). We choose three different values of ϵ: 0.001, 0.1 and 1.

Figure 8 shows the approximation rate in terms of number of existing rules and number of data points. For the presorting greedy approach, if there are more rules existing, the approximation is closer to the true value. Although the approximation rate is decreasing slightly with more number of data points, there is still a trend that the decreasing is getting smaller when the size of dataset is increasing. The full greedy approach approximates the true bounding function better than the presorting greedy approach. However, since the fully greedy approach costs more time than the presorting greedy, it is still reasonable to use the presorting greedy approach to get higher efficiency.

C ADDITIONAL DETAILS OF EMPIRICAL EVALUATION

All experiments in this paper are conducted on a computer with CPU 'Intel(R) Core(TM) i5-10300H CPU @ 2.50GHz' and memory of 24G.

To further show the difference between the proposed corrective orthogonal boosting and the other methods, we

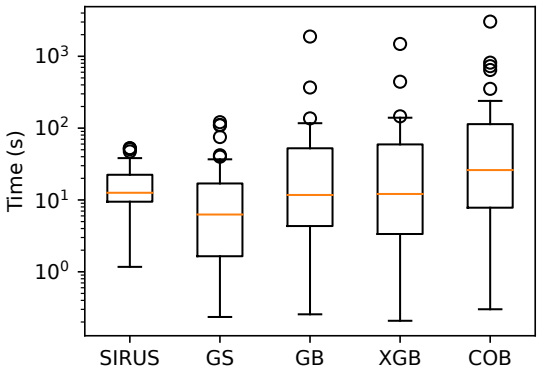

Figure 10: The distribution of computation times across all test datasets to reach complexity level 50 for different algorithms. GS and GB are using greedy search, while XGB and COB are using branch-and-bound.

provide additional details of empirical evaluation. Table 5 shows the normalised average training risk, test risk and computation time of the rule ensembles generated by SIRUS, SGS, SGB, SXB and COB using greedy search and branch-and-bound search over complexity levels from 1 to 50 for the 34 datasets used in the experiments of this paper. In Table 5, we bold the lowest training and test risks for each dataset, and the texts with red colours indicate the COB approach using greedy search or branch-and-bound search have lower risks than all the other methods. Figure 9 compares the normalised average logged training risks over complexity levels from 1 to 50 for different datasets. According to Fig. 9 and Table 5, COB generates lower training risks than the other algorithms for 26 out of 34 datasets.

For the COB with greedy search, there are 29 out of 34 datasets whose training risks are lower than the other methods. However, it has only 15 out of 34 datasets whose test risks are lower than other methods. Therefore, using branch-and-bound search generates better rule ensembles than greedy search. The One-sided T-test at significance level 0.05 with Bonferroni-correction for 8 hypotheses (4 for training and 4 for test) also shows the same results with the branch-and-bound search: the COB using greedy search generates rule ensembles with significantly less risk than the other methods by at least 0.001 of the normalised training and test risks.

Figure 10 shows the box plot of the running time for generating rule ensembles with complexity level 50 spent by different algorithms on the 34 datasets. Although the overall running time of COB is higher than the other methods, they are still at the same scale.

Furthermore, Figure 11 shows more comparisons of the risk / complexity tradeoff for SIRUS, gradient sum, gradient boosting, XGBoost and Orthogonal Gradient Boosting for 6 datasets. We compare the rule ensembles generated by COB with complexity around 20 and the first method whose risk value is compatitive with the COB rule ensemble.

Table 5: Comparison of normalised risks and computation times for rule ensembles, averaged over cognitive complexities between 1 and 50, using SIRUS(SRS), Gradient Sum(SGS), Gradient boosting (SGB), XGBoost (SXB) and COB (using greedy search and branch-and-bound search), for benchmark datasets of classification (upper), regression (middle) and Poisson regression problems (lower).

Dataset	d	n	Train risks						Test risks						Computation times				
			SRS	SGS	SGB	SXB	COB_G	COB_B	SRS	SGS	SGB	SXB	COB_G	COB_B	SRS	SGS	SGB	SXB	COB_B
titanic	7	1043	.895	.653	.656	.646	.639	**.616**	.894	.695	**.707**	.711	.713	.717	7.077	2.624	9.858	10.21	25.71
tic-tac-toe	27	958	.892	.555	.641	.577	.650	**.492**	.885	.596	.682	.629	.721	**.566**	12.59	3.971	10.34	6.09	13.99
iris	4	150	.685	.261	.261	.332	**.192**	.251	.745	.521	.440	.459	.531	**.424**	11.02	0.775	1.099	1.453	2.487
breast	30	569	.569	**.277**	.310	.314	.290	.304	.627	**.269**	.362	.338	.383	.349	11.48	6.744	74.43	74.83	239.2
wine	13	178	.578	.216	.250	.192	.191	**.183**	.621	.346	.431	.409	.483	**.265**	9.456	1.530	4.432	2.154	55.183
ibm hr	32	1470	.980	.567	.560	.571	**.558**	**.558**	.974	.640	.645	.636	.652	**.621**	11.15	17.24	10.99	12.92	12.03
telco churn	18	7043	.944	.679	.682	.678	**.664**	.668	.945	.663	.677	.665	**.650**	.660	50.83	40.01	1883	1485	3039
gender	20	3168	.566	.230	.230	.249	.224	**.224**	.570	**.243**	.247	.263	.246	.246	22.42	22.73	25.49	24.27	32.95
banknote	4	1372	.854	.304	.267	.290	.253	**.228**	.858	.311	.268	.299	.264	**.229**	8.933	6.298	5.648	7.060	8.444
liver	6	345	.908	.815	.834	.814	**.802**	.834	.917	.879	.927	**.873**	.940	.891	9.734	1.997	99.72	124.1	193.9
magic	10	19020	.906	.718	.708	.710	.707	.707	.903	.698	.693	.693	.688	**.688**	1.364	75.14	89.18	101.9	352.2
adult	11	30162	.804	.594	.599	.588	**.575**	.576	.802	.603	.615	.601	**.589**	.589	2.169	121.0	136.7	146.0	728.3
digits5	64	3915	**.248**	.332	.312	.344	.329	.315	**.262**	.329	.314	.341	.320	.315	52.60	110.8	72.74	101.5	97.4
insurance	6	1338	.169	.130	.142	.144	**.120**	.123	.177	.132	.145	.147	**.127**	.128	14.06	7.507	15.94	12.98	39.53
friedman1	10	2000	.180	.089	.074	.068	.067	.068	.165	.091	.077	.075	**.070**	.072	16.79	2.514	4.302	3.171	6.915
friedman2	4	10000	.082	.133	.119	.115	.770	**.075**	.082	.135	.120	.115	**.077**	**.077**	47.33	11.79	17.56	13.18	28.4
friedman3	4	5000	.093	.045	.042	.042	.041	**.041**	.092	.048	.047	.046	.045	**.045**	29.86	6.243	10.61	8.559	17.65
wage	5	1379	.427	.370	.362	.359	.352	.354	**.341**	.358	.405	.411	.368	.365	14.18	5.605	12.12	13.17	25.19
demographics	13	6876	.219	.214	.214	.214	.212	**.212**	**.209**	.216	.217	.217	.214	.215	38.24	36.80	29.40	33.04	72.42
gdp	1	35	.063	.020	.020	.020	.024	**.020**	.059	.020	.020	.020	.027	**.020**	7.974	.261	.351	.282	.488
used cars	4	1770	.373	.139	.123	.132	**.113**	**.121**	.427	.171	.131	.141	**.116**	.130	15.00	8.371	12.10	9.484	20.27
diabetes	10	442	.156	.138	.142	.139	.132	.134	.188	.141	.141	.147	**.136**	.149	10.50	2.204	3.574	3.920	7.591
boston	13	506	.101	.086	.087	.086	**.080**	.082	.105	**.079**	.087	.089	.088	.087	10.96	3.055	6.731	5.285	10.44
happiness	8	315	.109	.031	.031	.031	.029	**.029**	.109	.033	.039	.039	.035	**.033**	6.344	1.160	11.37	11.31	26.43
life expect.	21	1649	.109	.026	.026	.026	.026	**.026**	.110	.027	.027	.027	.026	**.026**	21.44	16.16	58.43	63.82	131.2
mobile prices	20	2000	.148	.131	.137	.137	**.122**	.126	.140	.134	.143	.143	**.126**	.132	33.81	15.03	367.7	442.5	815.4
suicide rate	5	27820	.547	.543	.540	.540	.540	.532	.514	.521	.521	.521	.519	.512	52.35	109.6	117.1	139.6	644.6
videogame	6	16327	**.895**	.953	.953	.953	.953	.953	.850	**.720**	**.720**	**.720**	**.720**	**.720**	1.171	41.91	34.38	45.90	119.1
red wine	11	1599	.072	.034	.035	.034	.034	.034	.073	.035	.036	.036	.035	**.035**	19.94	9.149	15.32	21.99	35.34
covid vic	4	85	NA	.153	.121	.132	.105	**.086**	NA	.182	**.100**	.133	.104	.086	NA	.523	.600	.628	.854
covid	2	225	NA	.344	.371	.891	.343	**.321**	NA	.459	.411	.741	.417	**.395**	NA	.701	.690	.682	1.143
bicycle	4	122	NA	.317	.324	.337	.296	**.275**	NA	.366	.478	.457	.529	**.320**	NA	.695	1.103	1.105	2.124
ships	4	34	NA	.174	.181	.146	**.125**	.168	NA	.197	.203	.199	.464	**.155**	NA	.235	.296	.311	.448
smoking	2	36	NA	.127	.128	.163	**.078**	.072	NA	.136	.250	.322	.121	**.084**	NA	.266	.256	.208	.301

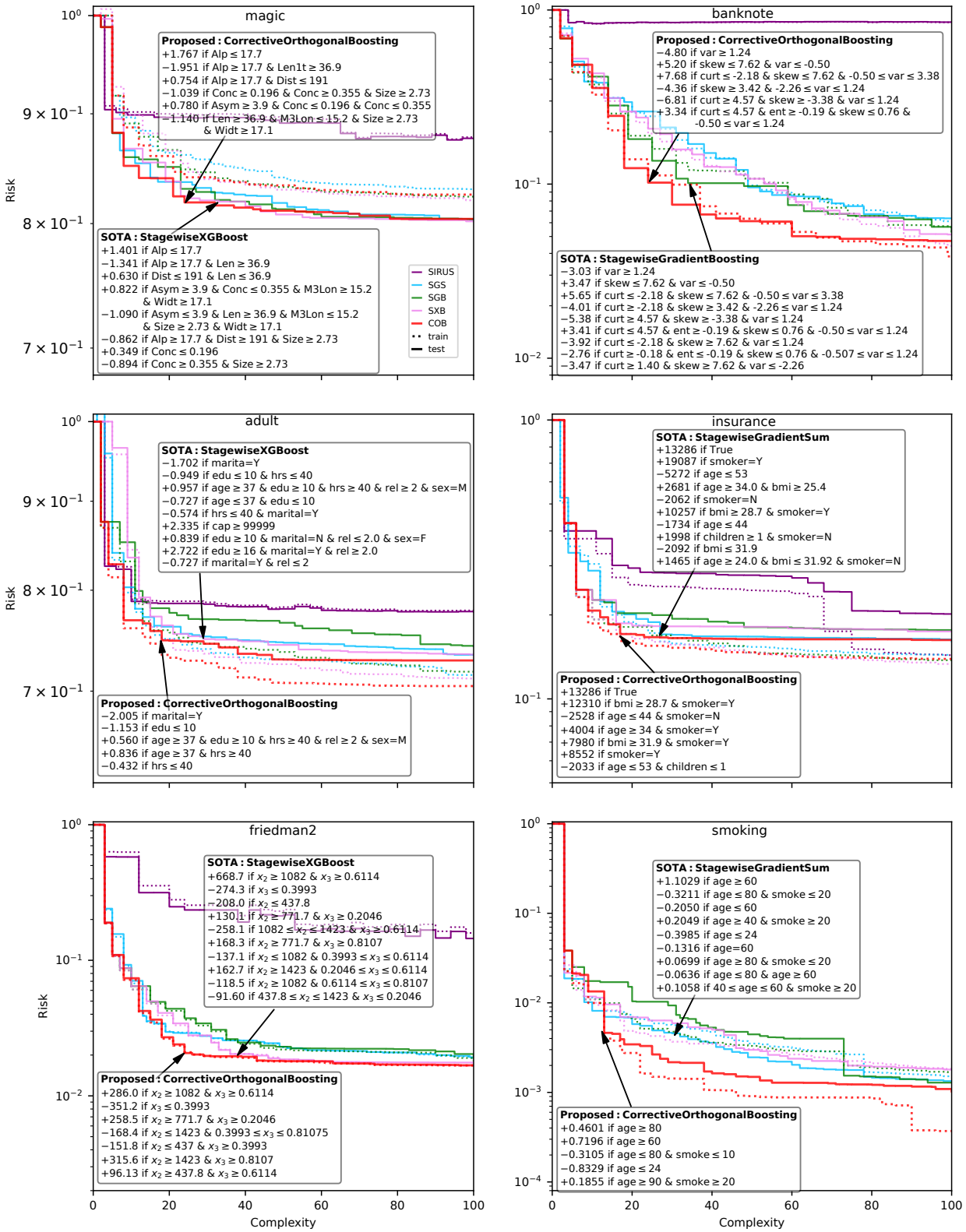

Figure 11: Risk/complexity curves of proposed approach (red) compared to alternatives for `magic`, `banknotes`, `adult`, `insurance`, `friedman 2` and `smoking`. Annotated rule ensembles have equivalent risk but substantially reduced complexity for the proposed method.