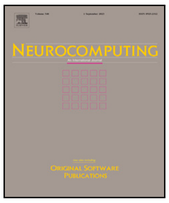
Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom



Improving rule-based classifiers by Bayes point aggregation

Luca Bergamin [a,*], Mirko Polato [b], Fabio Aiolli [a]

[a] *Department of Mathematics, University of Padova, Via Trieste 63, Padova, 35121, PD, Italy*
[b] *Department of Computer Science, University of Turin, Corso Svizzera, 185, Turin, 10149, TO, Italy*

ARTICLE INFO

Communicated by R. Zhu

Dataset link: https://github.com/BouncyButton/bayes-point-learning

Keywords:
Interpretability
Rule learning
Bayesian learning
Ensemble methods

ABSTRACT

The widespread adoption of artificial intelligence systems with continuously higher capabilities is causing ethical concerns. The lack of transparency, particularly for state-of-the-art models such as deep neural networks, hinders the applicability of such black-box methods in many domains, like the medical or the financial ones, where model transparency is a mandatory requirement, and hence white-box models are largely preferred over potentially more accurate but opaque techniques.

For this reason, in this paper, we focus on ruleset learning, arguably the most interpretable class of learning techniques. Specifically, we propose Bayes Point Rule Classifier, an ensemble methodology inspired by the Bayes Point Machine, to improve the performance and robustness of rule-based classifiers. In addition, to improve interpretability, we propose a technique to retain the most relevant rules based on their importance, thus increasing the transparency of the ensemble, making it easier to understand its decision-making process.

We also propose FIND-RS, a greedy ruleset learning algorithm that, under mild conditions, guarantees to learn hypothesis with perfect accuracy on the training set while preserving a good generalization capability to unseen data points.

We performed extensive experimentation showing that FIND-RS achieves state-of-the-art classification performance at the cost of a slight increase in the ruleset complexity w.r.t. the competitors. However, when paired with the Bayes Point Rule Classifier, FIND-RS outperforms all the considered baselines.

1. Introduction

In the early years of machine learning, rule learning dominated the research scene, with work dating back to the 1960s [1]. These rule learning systems peaked in early 2000 as one of the most popular machine learning schemes.

Nowadays, modern statistical machine learning methods, like Support Vector Machines (SVMs) and neural networks, define the go-to approach in most real-world tasks due to their performance. However, their inner workings are (usually) complex and lack transparency, which leads to poor comprehensibility [2]. Recently, there has been an increasing awareness of the importance of having the ability to explain the decisions of artificial intelligence (AI) systems [3].

With the advent of deep learning techniques, many efforts have been devoted to interpreting neural networks with numerous interesting results, especially on images and texts [4–7]. However, when working with tabular data, explanation rules are by far the most appreciated and effective in delivering a human-readable justification for the prediction outcome of AI algorithms [8].

Within *interpretable machine learning* methods, decision rule learning represents, by design, a direct approach to learning explainable models. Their inner working is transparent: each instance is classified into a class if and only if at least one rule is satisfied. These rules inherently serve as a natural and intuitive explanation for the classification outcome.

Moreover, decision rule-based classifiers lend themselves to a concise representation through a Disjunctive Normal Form (DNF) formula, essentially an OR combination of many AND conjunctive terms. This formal structure is well-established in machine learning literature [9] and is studied in discrete mathematics [10], providing a robust foundation for understanding and interpreting the model.

One distinctive advantage of decision rule learning is the inherent inspectability of the entire classifier. This attribute allows for the identification of anomalies in the learned model, contributing to the model's transparency and facilitating trust in the decision-making process. Importantly, this transparency can lead to performance improvements over time, as insights derived from the model may inform and enhance the data acquisition process [11]. Other popular models, such as SVMs or Random Forests (RFs), do not offer these important features.

The most iconic decision rule learning method is the Decision Tree (DT, [12]), which has been widely applied in many contexts, such as the medical field [13].

* Corresponding author.
E-mail address: bergamin@math.unipd.it (L. Bergamin).

https://doi.org/10.1016/j.neucom.2024.128699
Received 17 August 2023; Received in revised form 5 May 2024; Accepted 1 October 2024
Available online 16 October 2024

Besides DTs, rule set learning algorithms are also appreciated thanks to their natural interpretation [14]; however, such models usually lack effectiveness and/or scalability. Many of these rule set learning approaches are based on sequential covering [1,15], which is a general procedure that iterates over the training examples learning, rule by rule, a set of rules that covers the entire dataset.

Along with interpretability and effectiveness, robustness and stability are other essential features for a classifier, especially in scenarios with few training examples or applications, such as medicine and biology. Generally, robustness is not a strong suit for sequential covering algorithms, given their usual sensitivity to the order in which the training instances are considered. Yet, when data is scarce, the rules obtained strongly depend on the particular training data. In other words, the variance of rule-based algorithms is generally high. Aggregation methods, such as Bagging and Random Forest, have already been employed to reduce variance [16] in DTs. However, the resulting model is not interpretable.

In this paper, we propose a technique for aggregating rule sets. We summarize our contributions below.

- Leveraging Bayesian learning theory [17], we provide a principled methodology to construct a rule set aggregation that aims to improve performance while being more robust and interpretable. We call this technique the Bayes Point rule classifier, or BP classifier for short. The BP classifier embeds both the rule sets and the instances into a vector space, allowing us to model a rule set decision as a linear function. Starting from this new representation, we can aggregate the rule sets obtained with multiple executions of any rule-producing algorithm using a simple average. This technique is related to bagging: it reduces variance, thus improving the robustness, stability, and overall performance.
- To improve the interpretability of the resulting rule sets of the BP classifier, we define an importance metric that enables us to perform rule pruning safely. We provide a principled pruning heuristic that reduces the complexity of the final rule set while retaining good generalization capabilities.
- Further, to fully exploit the BP classifier, we propose FIND-RS, a novel concept learning algorithm that excels in this setting. FIND-RSis a sequential covering algorithm that always produces consistent hypotheses and produces rules with high variance. Such characteristic makes FIND-RSa very good fit when used in conjunction with the BP classifier;
- We experimentally show that a committee of rule sets generally improves the performance of rule-based classifiers. Our experiments also show that FIND-RS, when coupled with the BP classifier, achieves better performance than several state-of-the-art interpretable techniques.

2. Related works

We present a brief taxonomy of popular works used in rule induction. A summary of the main features of these methods are reported in Table 1.

Heuristic methods. Early research in rule learning focused on heuristic methods, such as Iterative Dichotomiser 3 (ID3, [18,19]) and Classification and Regression Trees (CART, [20]), which utilize a greedy partitioning approach based on different impurity measures. Sequential coverage algorithms are another class of methods that iteratively learn new rules to explain a subset of the training data and remove covered examples from the training process. Some well-known sequential covering algorithms include AQ [1], CN2 [21], and RIPPER [15].

RIPPER, based on the work of IREP [22], overcomes overfitting by employing a more effective pruning phase while maintaining computational efficiency, making it a fast method with good performance that is still considered state-of-the-art [23]. However, sequential covering algorithms may struggle with processing efficiency when dealing with modern datasets that continue to grow in size. Additionally, they cannot cover the entire search space, with OPUS being a notable exception [24].

The method proposed in its base form uses a sequential coverage strategy similar to RIPPER. It is also driven by some heuristics: in particular, we focus on extracting rules with high specificity that can be refined later in a post-processing step.

Probabilistic methods. A different class of rule-based algorithms employs Bayesian methods to evaluate the proper rules in a probabilistic framework. These rules are generally more apt to be resilient to noise and uncertainty. Notable examples are Bayesian Rule Lists (BRL, [25]), the subsequent Scalable Bayesian Rule Lists (SBRL, [26]), and Bayesian Rule Sets (BRS, [27]). Although their foundations are solid, their main weakness is the required computation of frequent itemsets, which increases training time. Moreover, high memory is a common requirement. Therefore, these methods do not scale well.

Our FIND-RS does not extract frequent item sets and can be executed to extract longer rules and to analyze high-dimensional datasets more efficiently.

The Bayes Point method proposed can be viewed in a probabilistic framework as well: it can provide a probability estimate of belonging to the positive class, enabling us to view any rule-extracting method as a probabilistic classifier.

Ensemble methods. A common way to improve machine learning methods is to use multiple instances of a simpler classifier. Both SLIPPER [28] and Random Forest [29] are algorithms that leverage this idea. The former opts for RIPPER as the base learner, while the latter uses decision trees. Decision trees are also at the core of RuleFit [30], CRE [31], and SIRUS [32], which are rule ensemble algorithms focused on improving interpretability. A voting strategy is a simple but effective way to boost performance while improving generalization.

Our Bayes Point technique leverages ensembles to boost the performance of a base rule classifier, and we provide a simple way to aggregate and prune rule sets to preserve interpretability.

Optimization-based methods. There has been growing interest in learning optimal rule lists using mathematical optimization techniques. For example, CORELS [33] uses an empirical risk minimization formulation to learn certifiably optimal rule lists by reducing the search space using a variety of computational techniques (e.g., branch and bound), which they demonstrate can outperform heuristic approaches such as RIPPER. Similarly, Yu et al. [34] use Boolean satisfiability (SAT) solvers to learn optimal decision sets and lists. These methods effectively find an optimal solution, but they can require high computing power, and usually, they constrain their search space to limited/shorter conjunctive rules.

Our FIND-RS uses a different approach: we do not optimally explore the whole DNF space, which would be infeasible for a large number of conjunctive terms, but an inductive bias drives us to find more specific rules that describe the positive training examples.

Gradient-based methods. Rule learning has recently seen a resurgence, using neural architectures based on gradient descent. Nowadays, thanks to the notable endeavors of the deep learning research community, these methods are competitive with heuristic models. These recent works span from learning DNFs using neural networks [35,36] to learning weighted DNFs (Wang et al. [37]). Using neural architectures brings advantages like reusable and modular structures. Additionally, these approaches benefit from scalability and adaptability, making them effective in capturing complex, non-linear patterns and dependencies in data.

These promising methods have typical pitfalls of deep learning, such as sensitivity to hyperparameters, high computing requirements, higher number of samples required, and convergence issues. Neural networks, primarily tailored for continuous data, face difficulties when handling discrete problems. Achieving good performance often requires additional structures like embeddings, highlighting that adapting neural networks to these problems is not trivial and can incur in additional overhead during training.

This resurgence contributes to giving new life to rule learning and unique perspectives on the nature of concept learning, e.g., DRNC [38], which considers a hierarchical rule structure. Compared to the other considered methods, DRNC favors reusable structures, akin to ID3, moving from trees to a multilayered structure.

Post-hoc interpretability methods. Many popular works were published to improve the interpretability of black-box methods. It is possible to train an interpretable model (e.g., a decision tree, or FIND-RS) on the prediction of a black-box classifier to obtain a surrogate classifier. Other methods that do not use surrogate classifiers were recently proposed; LIME [39] is a notable example. This method tackles the interpretability requirements by solving a different problem: it provides a reasonable explanation for a single instance by giving importance scores to important features used by a black-box classifier. This method has been extended to large neural networks and it has been applied to both images and text, and it is often the only viable approach to provide an amount of explainability to an existing application. Computing these scores requires additional time that can be high in non-trivial tasks.

SHAP [40] extends LIME and other interpretable methods under a joint framework. SHAP explanation is based on feature importance values. It assumes feature independence and learns an additive linear model to explain a specific input instance. Under these assumptions, game theory is used to identify some theoretical guarantees of the solution found. Its runtime can be higher compared to LIME, and some model-specific implementations have been proposed to improve speed, such as TreeSHAP [41].

Any rule learning method (and, by extension, FIND-RS) has a different approach. A post-hoc explanation will not ever be fully adherent to the underlying model but will provide an interpretation of the model outputs. A model that is explainable by design is a valid alternative where it is required to show how the model has processed a certain instance. In fact, a rule learning model can provide an exact rule (or set of rules, in the case of rule lists) that locally describes a given instance, with no additional computation required. Finally, LIME lacks a global interpretation that can be provided by rule learning methods. SHAP can provide a global interpretation by aggregating Shapley values, thus only providing a measure of feature importance, and not symbolic rules.

3. Background and notation

We consider binary classification problems with training sets $\mathcal{S} \equiv \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{n}$, where $\mathbf{x}^{(i)} \in \mathcal{X}$ are categorical feature vectors, with $\mathcal{X} \equiv \times_{j=1}^{m} \mathcal{X}_j$ for some finite (symbolic) attribute/variable domains $\mathcal{X}_j$, and $y^{(i)} \in \{-1, +1\}$.

We call $\mathcal{P} \equiv \{\mathbf{x} \mid (\mathbf{x}, y) \in \mathcal{S} \wedge y = +1\}$ the set of positive instances, and conversely, $\mathcal{N} \equiv \{\mathbf{x} \mid (\mathbf{x}, y) \in \mathcal{S} \wedge y = -1\}$ the set of negative instances. When not specified differently, we assume that $\nexists \mathbf{x} \in \mathcal{P}, \mathbf{z} \in \mathcal{N} \mid \mathbf{x} = \mathbf{z}$, i.e., the training set does not contain contradictory examples.

To handle categorical features, we employ two different encodings: attribute-value (AV) and one-hot (OH) encoding. Given an attribute (i.e., feature) with domain $\mathcal{X}_i$, $|\mathcal{X}_i| = d$, AV associates a specific value of $\mathcal{X}_i$ to the attribute, while OH converts the attribute into a binary vector of size d where only the index associated to the actual value is set to 1.

3.1. Rule learning background

To formalize the terminology used by the rule learning research community, we report some simple definitions useful to understand the rest of the paper.

Definition 3.1 (*Rule*)**.** Given an input space $\mathcal{X} \equiv \times_{j=1}^{m} \mathcal{X}_j$, a (conjunctive) rule is a set of indexed values (or constraints). It is denoted as $\mathbf{r} = \{(i, v_i), i \in C\}$, where $C \subseteq [1, m]$ is a subset of feature indices and $v_i \in \mathcal{X}_i$ is the constraint for the ith feature.

The choice of attributes' encoding (AV vs. OH) can be crucial for the success of learning. In particular, the number and nature of possible rules change.

Example. For an attribute a with three possible values $\{1, 2, 3\}$, a constraint in the AV and OH encodings are defined as in the following:

- $(a_{\text{AV}} = 1), (a_{\text{AV}} = 2), (a_{\text{AV}} = 3), (a_{\text{AV}} = ?)$ for the AV encoding,
- $a_{\text{OH}} = (1, 0, 0)$, $a_{\text{OH}} = (0, 1, 0)$, $a_{OH} = (0, 0, 1)$, $a_{\text{OH}} = (0, ?, ?)$, $a_{\text{OH}} = (?, 0, ?)$, $a_{\text{OH}} = (?, ?, 0)$, $a_{\text{OH}} = (?, ?, ?)$ for the OH encoding,

where $?$ is a wildcard that means any value for the variable. If we also include the negation, we would have rules like $a_{\text{OH}} = (?, 0, ?)$ or $a_{\text{AV}} \neq 1$. Note that constraints such as $a_{OH} = (1, 0, ?)$ are identical to $a_{OH} = (1, 0, 0)$, since a one-hot encoded value can contain only a single 1.

We call *specificity* of a rule $\mathbf{r}$ the number of conjunctive terms involved in $\mathbf{r}$, i.e., $|C|$.

Definition 3.2 (*Rule Coverage*)**.** We say that a rule $\mathbf{r}$ covers an instance $\mathbf{x} \in \mathcal{X}$ (or $\mathbf{x}$ satisfies $\mathbf{r}$) whether:

$$\mathbf{r} \geq \mathbf{x} \iff \bigwedge_{i \in C} \left(\mathbf{x}_i = v_i\right), \tag{1}$$

i.e., if $\mathbf{x}$ satisfies all the constraints in $\mathbf{r}$.

Definition 3.3 (*Rule Generalization*)**.** A rule $\mathbf{r}_1$ generalizes another rule $\mathbf{r}_2$ iff:

$$\mathbf{r}_1 \geq \mathbf{r}_2, \iff \forall \mathbf{x} \in \mathcal{X}, (\mathbf{r}_2 \geq \mathbf{x}) \Rightarrow (\mathbf{r}_1 \geq \mathbf{x}) \tag{2}$$

It is noteworthy that an instance $\mathbf{x}$ can be seen as a very specific rule in which every variable is constrained, i.e., $\forall i, \mathbf{x}_i = v_i$.

Definition 3.4 (*Rule Set*)**.** A rule set (or ruleset) is a set of rules, i.e., $\mathcal{R} \equiv \{\mathbf{r}_1, \dots, \mathbf{r}_k\}$. A set $\mathcal{R}$ of (conjunctive) rules covers an instance $\mathbf{x} \in \mathcal{X}$ iff:

$$\mathcal{R} \geq \mathbf{x} \iff \exists \mathbf{r} \in \mathcal{R} \mid \mathbf{r} \geq \mathbf{x}. \tag{3}$$

Therefore, a rule set $\mathcal{R}$ can be interpreted as a Disjunctive Normal Form (DNF) logical formula, as it consists of a disjunction of conjunctions of literals. Similarly, a rule can be seen as a conjunction of literals.

Given a rule set $\mathcal{R}$, we refer to its cardinality $|\mathcal{R}|$ as *complexity* of $\mathcal{R}$. A high complexity score means that a rule sets has many different rules.

Definition 3.5 (*Rule Set as Classification Function*)**.** A rule set h is a binary classification function $h : \mathcal{X} \to \{0, 1\}$. It classifies an instance $\mathbf{x}$ as positive iff $h \geq \mathbf{x}$, that is, at least a rule covers the given instance. Equivalently, an instance $\mathbf{x}$ is classified as negative iff there are no rules that cover the given instance.

Finally, $[\![b]\!] \in \{0, 1\}$ denotes the indicator function, which is 1 iff the condition b is true, 0 otherwise.

3.2. Bayesian learning background

A Bayesian perspective is useful to operate in noisy situations. It can be used to incorporate prior knowledge, make probabilistic predictions, and aggregate classifications of different predictors. It is also useful to develop and understand algorithms that do not explicitly manipulate probabilities, such as many concept learning methods [9].

Table 1
Features of the main rule learning models in the literature. The **Type** column uses the following acronyms. DT: Decision Tree; RS: Rule Set; RL: Rule List; H: Hierarchical; W: Weighted.

Name	Full name	Type	Opt.	Metrics	Pruning
ID3 [18]	Iterative Dichotomiser 3	DT	Divide and conquer	Information Gain	None
AQ [1]	Algorithm Quasi-Optimal	RL	Sequential covering	Coverage	Not specified
RIPPER [15]	Repeated Incremental Pruning to Produce Error Reduction	RS	Sequential covering	FOIL's Information Gain	Reduced Error Pruning
BRL [25]	Bayesian Rule List	RL	MCMC sampling	Posterior distribution scores	Prior distribution over rules
SBRL [26]	Scalable Bayesian Rule List	RL	MCMC sampling, branch and bound	Posterior distribution scores	Prior distribution over rules
BRS [27]	Bayesian Rule Set	RS	MCMC sampling, simulated annealing	Posterior distribution scores	Prior distribution over rules
CORELS [33]	Certifiably Optimal Rule Lists	RL	Branch and bound	Regularized empirical risk	Regulariz. term
Net-DNF [35]	Net Disjunctive Normal Form	RL	Backpropagation	Cross Entropy Loss	Regulariz. term
RRL [37]	Rule-based Representation Learning	WRL	Backpropagation	Cross Entropy Loss	Regulariz. term
DRNC [38]	Deep Rule Network Classifier	HRS	Greedy mini-batch learning	Accuracy	None
FIND-RS (Ours)	Find Rule Set	RS	Sequential covering	Coverage	Generalization procedure
x-BP (Ours)	Bayes Point	WRS	Ensemble learning	–	Importance-based rule pruning

Given a finite hypothesis space $\mathcal{H}$ and data $\mathcal{D}$, we can define the most probable classification and, consequently, the so-called Bayes optimal classifier as follows:

Definition 3.6 (*Bayes Optimal Classifier*)**.**

$$h_{bo}(\mathbf{x}) = \arg\max_{y\in\mathcal{Y}} \sum_{h_i\in\mathcal{H}} P(y|h_i)P(h_i|\mathcal{D}) \tag{4}$$

In practice, we can weight the classification probability of each hypothesis by its posterior probability, i.e., after seeing the data, and we can pick the most probable option. It is proven that no other classification method that uses the same hypothesis space and the same prior knowledge can outperform this method on average. This method maximizes the probability that the new instance is correctly classified given the available data, hypothesis space, and the prior probability over the hypotheses. In Eq. (4), the prior knowledge is inside the posterior probability $P(h_i|\mathcal{D})$, and we can exploit Bayes' theorem to highlight it: $P(h|\mathcal{D}) = \frac{P(\mathcal{D}|h)P(h)}{P(\mathcal{D})}$.

In the binary classification setting, when the hypothesis space is large, it is possible to resort to sampling, that is,

$$h_{bo}(\mathbf{x}) = \text{sign}\left(\mathbb{E}_{h\sim\mathcal{H}}[h(\mathbf{x})]\right) \tag{5}$$

where the expectation is taken by sampling hypotheses according to their posterior probability $P(h|\mathcal{D}) \propto P(\mathcal{D}|h)P(h)$.

When the classifiers are linear (e.g., the perceptron), it has been shown [17] that, under mild conditions, an approximation of the optimal classifier can be defined as follows.

Definition 3.7 (*Bayes Point Classifier*)**.** The Bayes point classifier is a surrogate of the optimal classifier, in which the center of mass of the version space is selected as a classifier. This corresponds to considering a uniform probability of the hypotheses W in the version space and a zero probability outside the version space. Thus, it can be shown that

$$h_{bp}(\mathbf{x}) = \text{sign}(\langle \mathbf{x}, \mathbf{w}_{cm}\rangle), \text{where } \mathbf{w}_{cm} = \mathbb{E}_W[W]. \tag{6}$$

The advantage of having such an approximation is that we obtain one classifier instead of an ensemble. To compute the expectation $\mathbb{E}_W[W]$, the authors propose running the Perceptron algorithm many times, starting from different permutations of the training dataset.

4. Bayes Point rule classifiers

As pointed out in the introduction of the paper, robustness is a key feature of rule-based classifiers. In this section, we propose two rule set aggregation techniques inspired by the Bayes setting.

4.1. Bayes optimal approximation

Ensembles are popular methods to reduce variance in classifiers [42]. For rule-based classifiers this is quite relevant, as their rules are generally learned by greedy-like algorithms that often present high variance, i.e., the rule sets obtained by different executions can differ substantially. Hence, combining these rule sets can improve the performance and stability of the final rule-based classifier. A trivial approach to lower their variance would be to average the classifications of different classifiers, similarly to Random Forests [16]. Rule-based classifiers are often used for their high interpretability, but using them naively to form an ensemble would dramatically decrease their interpretability, in exchange for higher accuracy. Therefore, our contribution is two-fold: first, we propose an ensemble method inspired by Bayes' theory, giving it a probabilistic interpretation; second, we provide a technique to improve its interpretability.

In the following, we will call a base learner "weak" learner, borrowing the term from the ensemble learning theory.

According to the Bayesian theory described in the previous section, we can consider rule sets obtained by different runs of a rule-based learner as weak hypotheses and then combine them according to the above-mentioned theory.

More formally, Bayes Optimal (BO) rule set runs T instances of the weak learner, obtaining T rule sets $\{h_1, \dots, h_T\}$. Then, an instance is classified according to the following aggregated decision:

$$H_{bo}(\mathbf{x}) = \text{sign}\left(\sum_{t=1}^{T} h_t(\mathbf{x})\right).$$

Unfortunately, this aggregated hypothesis is not interpretable. For this, in the following, we propose an alternative, inspired by the Bayes Point classifier idea, that improves interpretability of the BO rule set while maintaining its accuracy. With this purpose in mind, we need a way to embed a rule set (i.e., a weak hypothesis) in a vector space.

We consider an $(R+1)$-dimensional vector space where R is the number of possible rules. A rule set $\mathcal{R}$ can be embedded in a vector

$\hat{\mathbf{w}} \in \mathbb{R}^{R+1}$ where $\hat{w}_r = 1, 1 \leq r \leq R$ iff the rule indexed by r is present in the rule set, and $\hat{w}_0 = -\frac{1}{2}$. An instance can be represented in the same vector space as the vector $\hat{\mathbf{x}} \in \mathbb{R}^{R+1}$ where $\hat{x}_r = [\![\mathbf{r} \geq \mathbf{x}]\!], 1 \leq r \leq R$, and $\hat{x}_0 = 1$. It can be easily verified that the decision of the rule set can now be expressed as

$$h(\mathbf{x}) = \operatorname{sign}(\langle \hat{\mathbf{w}}, \hat{\mathbf{x}} \rangle) = \operatorname{sign}\left(\sum_{\mathbf{r} \in \mathcal{R}} [\![\mathbf{r} \geq \mathbf{x}]\!] - \frac{1}{2} \right),$$

i.e., $\mathbf{x}$ is classified as positive iff $\exists \mathbf{r} \in \mathcal{R} \mid \mathbf{r} \geq \mathbf{x}$.

Now, the Bayes Point (BP) rule set is obtained by taking the approximate Bayes point as the average of the hypotheses $\hat{\mathbf{w}}^{(t)}$ found by running the weak learner T times:

$$H_{bp}(\mathbf{x}) = \operatorname{sign}\left(\frac{1}{T} \sum_{t=1}^{T} (\langle \hat{\mathbf{w}}^{(t)}_{1:R}, \hat{\mathbf{x}}_{1:R} \rangle - \frac{1}{2}) \right) > 0$$

$$\Leftrightarrow \sum_{t=1}^{T} \langle \hat{\mathbf{w}}^{(t)}_{1:R}, \hat{\mathbf{x}}_{1:R} \rangle > \sum_{t=1}^{T} \frac{1}{2} = \frac{T}{2}.$$

Noticing that $\sum_t \langle \hat{\mathbf{w}}^{(t)}_{1:R}, \hat{\mathbf{x}}_{1:R} \rangle$ equals the number of rules in the T rulesets that $\mathbf{x}$ satisfies, then H_{bp} classifies a new instance as positive whenever the number of rules $\mathbf{x}$ satisfies exceeds $T/2$. Moreover, let G be the set of unique rules discovered, then the decision can be rewritten as

$$\sum_{\mathbf{r} \in G} \alpha_{\mathbf{r}} [\![\mathbf{r} \geq \mathbf{x}]\!] > 1/2 \tag{7}$$

where $0 < \alpha_{\mathbf{r}} \leq 1$ is the fraction of times that the rule $\mathbf{r}$ has been discovered.

4.1.1. Rule pruning

By leveraging the weights $\alpha_{\mathbf{r}}$ in Eq. (7), we can provide a principled method for ordering the rules discovered according to these weights. The coefficient $\alpha_{\mathbf{r}}$ represents the importance of a rule $\mathbf{r}$ in the decision, providing an indicator that can be used to perform rule pruning. Rule pruning is a common strategy to reduce the complexity of a hypothesis, thus improving the interpretability of the underlying method.

Note that when pruning is performed, for example, retaining the first K rules G_K, then the discriminant function needs to be changed accordingly. Namely, $H_K(\mathbf{x}) = +1$ iff

$$\sum_{\mathbf{r} \in G_K} \alpha_{\mathbf{r}} [\![\mathbf{r} \geq \mathbf{x}]\!] > \frac{\gamma_K}{2}, \text{ where } \gamma_K = \frac{\sum_{\mathbf{r} \in G_K} \alpha_{\mathbf{r}}}{\sum_{\mathbf{r} \in G} \alpha_{\mathbf{r}}}.$$

Note that the above formula can be rewritten as

$$v(\mathbf{x}) = \sum_{\mathbf{r} \in G_K} \bar{\alpha}_{\mathbf{r}} [\![\mathbf{r} \geq \mathbf{x}]\!] > 1, \text{ where } \bar{\alpha}_{\mathbf{r}} = 2\frac{\alpha_{\mathbf{r}}}{\gamma_K}$$

giving an interesting link between rule-based classification and probability. Consider the ratio $\frac{P(+|\mathbf{x})}{P(-|\mathbf{x})} = \frac{P(+|\mathbf{x})}{1-P(+|\mathbf{x})}$, also called the odds ratio. If we set this to be equal to $v(\mathbf{x})$, we can calculate the probability of the positive class $P(+|\mathbf{x})$ in terms of $v(\mathbf{x})$, obtaining $P(+|\mathbf{x}) = \frac{v(\mathbf{x})}{v(\mathbf{x})+1}$. In practice, this probability checks the confidence α_r of each satisfied rule in the rule set G_K. If there are more rules that match the instance, or the α_r value is high, the probability is increased.

5. FIND-RS

Common sequential covering algorithms, such as RIPPER, can find simple rules and have a moderate resiliency to noise. Hence, the variance of these rules is limited. This is beneficial for the base method but decreases its effectiveness when combined into an ensemble, such as the Bayes Point. Hence, we propose FIND-RS, a sequential covering algorithm that always produces consistent hypotheses and produces rules with high variance. Akin to decision trees in a Random Forest, high-variance classifiers can be successfully exploited to form an ensemble.

According to the Bayesian theory, hypotheses, i.e., rule sets, are sampled according to their posterior probability $P(h|\mathcal{D})$ which is proportional to the posterior $P(\mathcal{D}|h)$ and the prior $P(h)$. Put in probabilistic terms, FIND-RS will find a hypothesis such that it is always consistent, i.e., $P(\mathcal{D}|h) > 0$ iff its error is zero on the training set $\mathcal{D}$. Moreover, it will be defined in a greedy way to have an inductive bias towards simpler (less complex) rule sets, i.e., if h_1 is less complex than h_2 then $P(h_1) > P(h_2)$.

FIND-RS computes the classification rule set hypothesis by building a rule at a time. Each rule is initially defined as a randomly picked positive training instance $\mathbf{s} \in \mathcal{P}$ (not already covered by the running rule set). Then, FIND-RS tries to greedily generalize the current rule considering the remaining uncovered positive instances in a fashion that may resemble the FIND-S algorithm [9].

FIND-RS generalizes a rule by removing one or more attribute-value constraints (i.e., setting it equal to the wildcard) while keeping the overall hypothesis consistent with the negative examples. Greedily building a rule at a time, makes FIND-RS biased towards building less complex formulas, that is, with as few disjunctions as possible, thus providing more interpretable hypotheses.

It is worth noticing that, by design, FIND-RS will always find a rule set that correctly classifies all the examples in the training set if there are no contradictory examples, that is $\nexists(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2) \in \mathcal{S} \mid (\mathbf{x}_1 = \mathbf{x}_2) \wedge (y_1 \neq y_2)$. In the worst case, FIND-RS will return a hypothesis of the form $\bigvee_{\mathbf{p} \in \mathcal{P}} \mathbf{p}$ that clearly overfits the training set.

Algorithm 1 provides a detailed description of FIND-RS, while Fig. 1 depicts an overview of the algorithm execution on a toy example. The algorithm uses two lists of lists, B and D. The former, B, defines a list of bins, where each bin has all examples that share a rule. The latter, D, is used to encode the rule associated to each bin.

The algorithm follows the usual structure of a sequential covering algorithm: it processes examples from the positive dataset $\mathcal{P}$ until it is empty (line 2). A starting example $\mathbf{s}$ is chosen, and it is removed from $\mathcal{P}$ (line 3). We instantiate the starting rule as the most specific rule, i.e., a rule that matches only the chosen example. Then, for each positive example $\mathbf{p}$ left (line 7), we attempt to insert it in the new bin, creating a new generalized rule which covers the new example $\mathbf{p}$. If the new rule does not cover any negative example, the positive example $\mathbf{p}$ can be added to the bin, and the rule is updated accordingly. Otherwise, we continue. At the end of the algorithm, simplification procedures are performed, described in the following subsections.

5.1. Rule set simplification

Being FIND-RS a greedy algorithm, at the end of the training process, the produced hypothesis may contain superfluous rules, i.e., rules that can be removed without losing consistency. Thus, a simplification phase is performed to remove such redundant rules, reducing the complexity of the final rule set. To speed up this process, we rely on the following proposition, proven in Appendix.

Proposition 1. *At every iteration t of FIND-RS it holds that*

$$\forall j < i \in [k],\ \nexists \mathbf{x} \in B_i \mid \mathbf{r}_j \geq \mathbf{x},$$

where the current hypothesis is $D^t = \{\mathbf{r}_1, \dots, \mathbf{r}_k\}$.

The proof is available in Appendix. Proposition 1 provides a "backward incompatibility" between rules; however, it does not say anything in the other direction. In particular, it may happen that every instance in B_i, for some i, can be covered by other conjunctive rules $\mathbf{r}_j$ for $j > i$. In practice, this post-processing step (function `simplify` in Algorithm 1) checks, for all bins, if a bin B_i can be emptied by distributing all of its examples into any of the following bins B_j, where $j > i$. If this event occurs (line 11), it means that the corresponding conjunctive term D_i can be safely removed from the current hypothesis (line 12, 13), thus creating a more specialized one that is still consistent with the training set.

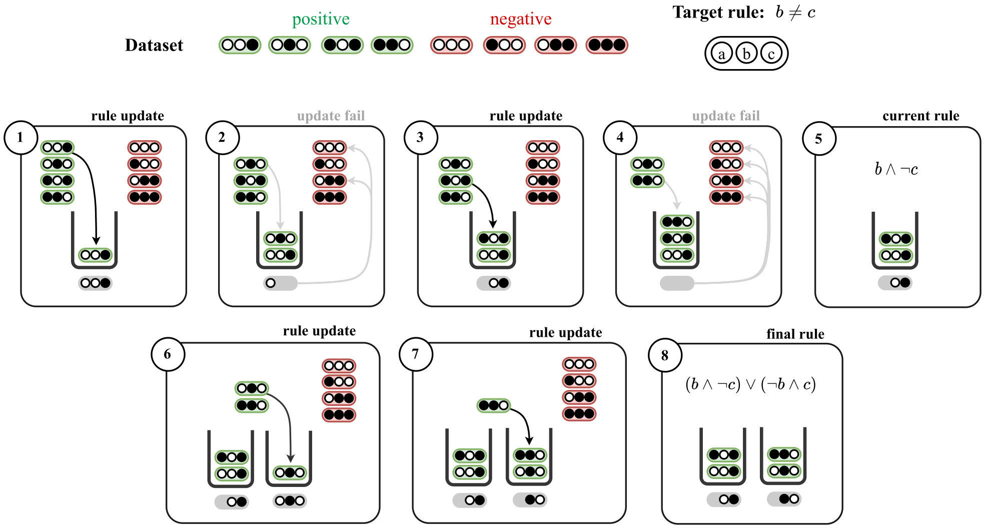


Fig. 1. We show the FIND-RS algorithm through a toy example of binary classification with 8 3-dimensional instances (a, b, c) with $a, b, c \in \{0, 1\}$, where the target concept is $\mathbf{b} \neq \mathbf{c}$, i.e., $(b \wedge \neg c) \vee (\neg b \wedge c)$.

Algorithm 1: FIND-RS

```
Input: $\mathcal{P}$: set of positive examples; $\mathcal{N}$: set of negative examples;
       $p_{gen}$: generalization probability
Output: Disjunctive Normal Form rule
1  $B, D, k \leftarrow [\,], [\,], 0$
2  while $\mathcal{P}$ is not empty do
3      $\mathbf{s} \leftarrow \text{pop}(\mathcal{P})$                      ▷ Pick a starting example
4      $B, D \leftarrow B + \{\mathbf{s}\}, D + \mathbf{s}$
5      $B_k \leftarrow B_k \cup \{\mathbf{s}\}$
6      $Q \leftarrow [\,]$                              ▷ Examples not covered by the rule
7      for $\mathbf{p} \in \mathcal{P}$ do
8          $\mathbf{r}' \leftarrow \text{generalize}(D_k, \mathbf{p})$
9          if $\nexists \mathbf{n} \in \mathcal{N} \mid \mathbf{r}' \geq \mathbf{n}$ then
10             $B_k \leftarrow B_k \cup \{\mathbf{p}\}$
11             $D_k \leftarrow \mathbf{r}'$                     ▷ Update the last rule
12         else
13             $Q \leftarrow Q + \mathbf{p}$
14     $\mathcal{P} \leftarrow Q$
15     $k \leftarrow k + 1$
16 $B, D \leftarrow \text{simplify}(B, D)$                       ▷ see Algorithm 2
17 $B, D \leftarrow \text{generalize}(\mathcal{N}, D, p_{gen})$              ▷ see Algorithm 3
18 return $B, D$
```

5.2. Rule set generalization

FIND-RS finds a partition of positive examples (i.e., the sets B_i in Algorithm 1) that is generally dependent on the order in which examples are seen. For each partition, a rule is built corresponding to the most specific hypothesis, encoding the constraints that are necessary to cover all the examples in the partition.

Since FIND-RS is a greedy algorithm, usually a small number of general bins is found. Alongside these bins, a high number of bins is found, each one having small size, yielding more specific rules. Intuitively, there is room for improvement: it is possible to generalize further, maintaining the consistency of the classifier. Observe that exhaustive algorithms are unfeasible: in fact, the set of most general hypotheses can be extracted only in exponential time (e.g., using the Candidate Elimination algorithm [9]).

Hence, we employ a simple, greedy strategy to increase generalization, relaxing each rule independently with a stochastic approach. Specifically, during the generalization process, constraints are dropped from the maximally specific rule, as described by Algorithm 3. We navigate each rule and each constraint (line 1, 3). The primary criterion for dropping a constraint is that it should not cover negative examples. Additionally, we empirically found that adding a degree of stochasticity into the generalization procedure improved performance. If a constraint can be dropped, we do so with $p \leq 1$ (line 6). The value of p can be fine-tuned as a hyperparameter during experimentation or manually set.

Although the strategy does not guarantee to extract the most general rule, it provides a good approximation in quadratic time. Additionally, we speculate that is helpful to use hypotheses that are neither maximally general nor specific: the most general rule could include unseen negative examples, while the most specific rule could not include unseen positive examples. This hypothesis is supported by the Bayes Point theory [17], since it promotes classifiers closer to the center of mass of the version space, and the size principle [43], that observes how more specific rules are more effective where there is a sufficient number of examples observed. In fact, the proposed method explores the version space, moving in the neighborhood of the initial rule set.

Finally, we report in Fig. 2 a simplifying example to illustrate both the Bayes Point, computed using FIND-RS, and the generalization procedure.

5.3. Computational complexity

The computational complexity of Algorithm 1 can be expressed in terms of the number of positive and negative examples, where P is the number of positive examples and N is the number of negative examples. Hence, the algorithm runs in $O(P^2 N d)$ time. In fact, we iterate over $\mathcal{P}$ in two nested cycles (cfr. line 2 and line 7), and iterate over $\mathcal{N}$ to check if negative examples are covered (cfr. line 9). The simplify procedure in Algorithm 2 takes $O(P^3 d)$ time, given we iterate in three nested cycles (cfr. line 2, line 3, and line 6). Finally, the generalize procedure in Algorithm 3 takes $O(PNd)$ time (cfr. line 1, line 3, and line 6). Note that these complexities are in practice lower, since the number of bins is much less than the number of positive examples unless the method drastically overfits.

We can conclude that the complexity is cubic in the number of examples, and in the worst case, it runs no worse than $O(M^3 d)$ time, where M is the total number of examples and d is the number of features.

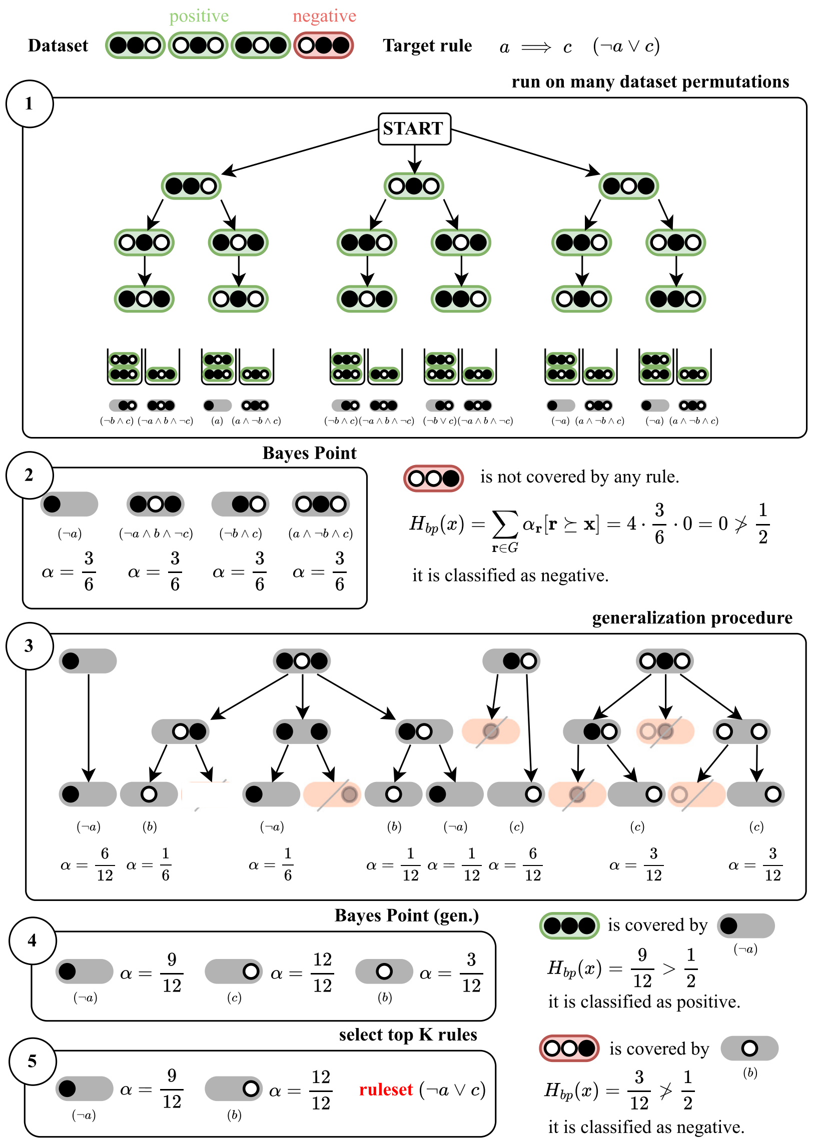

positive
negative
Dataset
Target rule $a \implies c$ $(\neg a \vee c)$
run on many dataset permutations
START
Bayes Point
is not covered by any rule.
$H_{bp}(x) = \sum_{\mathbf{r} \in G} \alpha_{\mathbf{r}}[\mathbf{r} \succeq \mathbf{x}] = 4 \cdot \frac{3}{6} \cdot 0 = 0 \not> \frac{1}{2}$
it is classified as negative.
generalization procedure
Bayes Point (gen.)
is covered by
$H_{bp}(x) = \frac{9}{12} > \frac{1}{2}$
it is classified as positive.
select top K rules
ruleset $(\neg a \vee c)$
$H_{bp}(x) = \frac{3}{12} \not> \frac{1}{2}$
it is classified as negative.

Fig. 2. The Bayes Point algorithm through a toy example. In ①, each path represents a permutation of the training examples, leading to a different ruleset. In ③, each path shows an outcome of the generalization procedure, leading to a different rule.

6. Experiments

In this section, we discuss how we tested the proposed techniques on different datasets and models. We provide all the code used to implement the models.[1]

6.1. Models

To build the baselines and the weak learners, we considered the following decision rule learning algorithms: FIND-RS (Ours), RIPPER [15], ID3 [19], AQ [1], and BRS [27].

In terms of classification performance, we also compare our methodology with state-of-the-art models such as Support Vector Machines (SVM) [44] and Random forests (RF) [29]. Additionally, we consider TabNet [45], a state-of-the-art neural model, specifically designed for tabular data. Despite these models can provide some interpretability (e.g., feature importance metrics), they cannot be described using rulesets. Therefore, their performance is just reported to quantify the loss in accuracy of rule-based classifiers.

6.2. Datasets

We selected 19 datasets from the UCI Machine Learning Repository.[2] The datasets are summarized in Table 2. They are split into four sections, according to their size. All datasets are discrete, except for `ADULT` and `MARKET`, which have also continuous features that have been discretized using 5 quantile binning. The methods have been trained using both attribute-value and one-hot encoding, if possible. We also consider a high-dimensional continuous dataset, `RNA-SEQ`, that considers a gene expression task. We discretized it using two quantile binning, considering its high number of features.

Since we work with binary classification, multiclass datasets have been converted into binary classification datasets by selecting the most frequent class as the positive class. We removed inconsistent examples, that is, pairs of identical instances that were labeled with different classes. In particular, we kept only one example for each duplicate, relabeling it according to their most frequent class (or the positive class, in case of parity).

1 https://github.com/BouncyButton/bayes-point-learning

2 https://archive.ics.uci.edu/ml/index.php

Algorithm 2: `simplify` - FIND-RS simplification procedure

```
Input: $B$: set of bins; $D$: set of rules
Output: $B$: simplified set of bins; $D$: simplified set of rules
1  $i \leftarrow 0$
2  while $i < len(D)$ do
3      for $j \in [i+1, len(D)]$ do
4          $\mathbf{r}' \leftarrow D_j$
5          $k = 0$
6          for $\mathbf{p} \in \mathcal{P}$ do
7              if $\mathbf{r}' \geq \mathbf{p}$ then
8                  $B_i$.pop($k$)
9              else
10                 $k \leftarrow k + 1$
11     if $D_i$ is empty then
12         remove($B_i$)
13         remove($D_i$)
14     else
15         $i \leftarrow i + 1$
16 return $B, D$
```

Algorithm 3: `generalize` - FIND-RS generalization procedure

```
Input: $\mathcal{N}$: set of negative examples
       $D$: set of rules
       $p_{gen}$: generalization probability (default: 0.9)
Output: $D$: generalized set of rules
1  for $D_i \in D$ do
2      $\mathbf{r} \leftarrow D_i$
3      for $c \in D_i$ do
4          $\mathbf{r}' \leftarrow \mathbf{r} - c$          ▷ Remove a constraint in a random order
5          $p \leftarrow$ random()
6          if $\nexists n \in \mathcal{N} \mid \mathbf{r}' \geq n \wedge p \leq p_{gen}$ then
7              $\mathbf{r} \leftarrow \mathbf{r}'$
8      $D_i \leftarrow \mathbf{r}$
9  return $D$
```

Table 2
Benchmark datasets for the rule learning task.

Size	Dataset	#instances	#feat.	⊕ class
Small	AUDIO	198	69	class = cochlear_a
	BREAST	265	9	class = recurrence
	CAR	1727	6	class = unacc
	COMPAS	804	27	recidiva = 1
	HIV	745	8	class = 1
	LYMPHOGRAPHY	147	18	class = 2
	MONKS1	431	6	class = 1
	MONKS2	431	6	class = 1
	MONKS3	431	6	class = 1
	PRIMARY	281	17	class = 1
	SOYBEAN	302	35	class = frog-eye-l
	SPECT	217	22	class = 1
	TTT	957	9	class = positive
	VOTE	341	16	class = republican
Medium	KR-VS-KP	3195	36	class = won
	MUSH	8123	22	class = e
Large	ADULT	32 535	14	class = ≥ 50K
	CONNECT-4	67 556	42	class = win
	MARKET	45 211	16	y = yes
High dimensional	RNA-SEQ	801	20 531	class = BRCA

6.3. Metrics

To evaluate the classification performance of the methods, we used the F1-score, defined as follows.

$$\text{F1}-\text{score} = \frac{\text{TP}}{\text{TP} + \frac{1}{2}(\text{FP} + \text{FN})}$$

where TP, TN, FP, FN are the number of true positives, true negatives, false positives, and false negatives predicted.

We argue this metric is preferable to the accuracy since it can highlight a predominance of false positives or false negatives, using a single metric. This is especially important for rule-based classifiers, which may identify rules that cover all (or no) examples.

The complexity and specificity of the rule sets are calculated according to their definitions; given a rule set $\mathcal{R}$, its complexity is $|\mathcal{R}|$, and its specificity is the average specificity of the rules in $\mathcal{R}$.

We also define some metrics to measure the effectiveness of individual rules. Rule coverage is defined as $\frac{P_\mathbf{r}}{P}$, and rule precision is defined as $\frac{P_\mathbf{r}}{P_\mathbf{r}+N_\mathbf{r}}$, considering $P_\mathbf{r}$ and $N_\mathbf{r}$ as the number of positive and negative examples classified by the rule $\mathbf{r}$, and P the number of all the positive examples.

Finally, we use two ranking strategies to compare FIND-RS to baseline rule-based classifiers.

- The overall Average Rank (AvgRank) first averages all F1-scores for each dataset and method, and then it ranks all classifiers using the averaged F1-score. The best classifier has rank 1, while the worst has rank N. Finally, we average all the rankings across different datasets.
- The run-based Average Rank (AvgRank-run) does not compute the average F1-score but considers each run independently. This second metric gives a lower score to a less robust method, i.e., it has more variability by changing its initialization only.

6.4. Experimental procedure

To train and test our baselines, we used the following procedure:

1. the data set is randomly divided into training and test sets. We considered a training-test proportion equal to 0.5;
2. the model is trained using the full training set;
3. we iterated each base method T times to create the BP classifier.
4. accuracy and F1-score are calculated using the test set;
5. the process (from (1) to (3)) is repeated 10 times to compute the average and the standard deviation of the metrics;
6. if applicable, we report the results for the best-performing encoding on average (AV or one-hot encoding).

Finally, for the biggest datasets (ADULT, CONNECT-4, and MARKET), we repeated the experiments 3 times, instead of 10, due to the high computing time.

6.5. Statistical analysis

We employed a robust statistical analysis to highlight any significant difference between FIND-RS and all the baselines. We applied a popular procedure proposed by Demšar et al. [46] and popularized by deep learning reviews [47].

First, we apply a Friedman test [48] to reject the null hypothesis ($\alpha < 0.05$). Secondly, we perform a pairwise post-hoc analysis called the Wilcoxon signed-rank test [49]. Each classifier is compared to the others, considering each independent run and each dataset, and a p-value is computed. Then, we apply Holm's correction to the p-values [50] to address the multiple testing problem and to produce the corrected p-values. Finally, we display a critical difference (CD) diagram to summarize the corrected p-values [46]. A CD diagram connects any clique of non-significantly different classifiers with a straight line. We also display the AvgRank-run metric.

Table 3
Hyperparameters considered in our experiments. For some hyper-parameters, a range of values is presented. These values are used for performing a 5-fold CV grid search.

Baseline	Hyper-parameters
FIND-RS	$p_{gen} = 0.9$
RIPPER	$k = 2$, prune_size $= 0.33$
	dl_allowance $= 64$
AQ	maxstar $= 3$
BRS	maxlen $= 3$
	max_iter $= 100$
TabNet	batch_size $= 64$, epochs $= 500$ (small)
	batch_size $= 256$, epochs $= 250$ (medium)
	batch_size $= 1024$, epochs $= 100$ (large)
RF	n_estimators $\in \{10, 100, 500\}$
	max_depth $\in \{\text{None}, 5, 10\}$
SVM	kernel $\in \{\text{linear}, \text{rbf}\}$,
	$C \in \{0.01, 0.1, 1, 10\}$
	$\gamma \in \{\text{'scale'}, \text{'auto'}\}$
	max_iter $=$ 1e6

6.6. Hyperparameters

We summarized our chosen hyperparameters selected for the methods in Table 3. For all methods, T has been set to 100. Since AQ is more computationally demanding, we used $T = 20$ and ran it only on smaller datasets. For BRS, we limit the rule length to 3, trading off the accuracy of a more specific rule with faster training. Finally, we selected $T = 20$ for the bigger datasets to speed up training.

To run Tabnet, we use the original implementation provided by the authors [45] with the parameters recommended. We adjusted the batch size and the number of epochs according to the size of the dataset.

6.7. Initialization strategies for Bayes Point classifiers

The proposed method requires many iterations of a base method. Ensuring sufficient diversity in the rules produced is essential. Therefore, each iteration preprocesses the training set with some simple techniques. We found that the following techniques were satisfactory:

- **Permutating the training set**. Covering algorithms usually consider a single example at a time and grow new rules by checking other examples. Therefore, the order of the examples can influence the rules created.
- **Using bootstrap**. Entropy-based algorithms, such as ID3 and BRS, do not grow rules starting from a single example, but compute overall statistics on the dataset. Hence, the previous strategy would be ineffective. Instead, the training set can be sampled with replacement, similarly to Random Forests.

6.8. Pruning strategy for Bayes Point classifiers

As pointed out in Section 5.1, we suggest a strategy for the selection of a good number K of rules to keep. Specifically, by ordering the rules by decreasing importance α, we can select the minimum number of rules such that the training accuracy is above the training accuracy of the BP method, multiplied by a threshold. In the following experiments, the threshold is set to 0.99.

7. Results

Here, we present a range of experiments reported in the paper. In Section 7.1, we discuss the performance of base methods, showing that FIND-RS is competitive with the state-of-the-art rule learning algorithms. In Sections 7.2 and 7.3, we consider a subset of datasets to clarify the inner workings of the model and to show both their potential and limitations. Section 7.4 discusses the impact of the hyperparameters of FIND-RS. Section 7.5 shows that the BP gives a tangible performance boost to the base methods, and in particular to FIND-RS, and in Section 7.6 we discuss how performance changes depending on the number of iterations T. In Section 7.7, we clarify what rule sets are extracted using the BP strategy. In Sections 7.8–7.10, we show how to improve the interpretability of the BP model, and the trade-off in interpretability and accuracy for the BP method. In Section 7.11 we showcase how FIND-RS works applied to a high-dimensional dataset for the gene expression domain. Finally, in Section 7.12 we compare an explanation provided by a rule vs. an explanation extracted with LIME for a selected dataset.

7.1. Performance of base learners

In order to assess the effectiveness of our proposed FIND-RS method, we compare it against several baseline methods commonly used in rule-based classification. In Table 4, we report the performance of these base methods on various datasets. We observe that RIPPER performs quite well, managing to achieve performance close to SVM and RF. However, we also notice that FIND-RS outperforms RIPPER on datasets that are governed by a noise-free rule, such as the MONKS and TTT datasets. This suggests that our approach is particularly well-suited for datasets where rules play a dominant role in classification. According to the Friedman test for the ranks, we found a significance of more than 95%, that also holds for all the following experiments.

These results highlight the strengths and weaknesses of different rule-based classifiers and demonstrate the potential benefits of using our proposed FIND-RS method in scenarios where a noise-free rule is present.

7.2. Similarity to ground truth rulesets

In the previous subsection, one could wonder if the ruleset presented corresponds to the ground truth ruleset that generated the data. While this is unfeasible for real-world datasets, it is possible to show this for artificial datasets that can be described using a DNF formula.

We employ the intersection-over-union metric (IoU), also known as the Jaccard distance, to compute the similarity of two different rule sets.

$$J(\mathcal{R}_1, \mathcal{R}_2) = \frac{|\mathcal{R}_1 \cap \mathcal{R}_2|}{|\mathcal{R}_1 \cup \mathcal{R}_2|} \tag{8}$$

In Table 5 we show how the rule sets found by different methods compare, on average, to the ground truth rule set, also considering different dataset sizes. Since each method suggests a different inductive bias, only some algorithms can be effective in discovering the underlying rules. In particular, FIND-RS is mostly effective with exact and specific rules. Finally, finding the right representation is fundamental for some datasets since a one-hot representation can expand the hypothesis space to include rules such as not x_i, as shown by the MONKS2 dataset.

Here, FIND-RS is able to find some rules that partially correspond to the target concept: as reported by Table 6, the method is able to find rules that match exactly to a portion of a target concept (i.e., four "$\neq$ 1" and two "= 1"), or rules that are more specific (i.e., two "= 1" and some more specific constraints).

7.3. Complexity of base learners

In our experiment, we evaluated the complexity of the rulesets extracted by different classification methods. Table 7 compares the average complexity of the rulesets produced by each method.

Considering the average number of rules found, our analysis showed that RIPPER and BRS tended to produce the simplest rulesets, followed

Table 4
Average F1-score of base learners. One-hot encoding was used if it was the best-performing one (marked with †). Results are averaged across 10 runs. Best results across rule-based methods are **bolded**, best results overall are underlined.

Dataset	Find-RS	RIPPER	ID3	AQ	BRS	SVM	RF	TabNet
AUDIO	**0.836** 0.09	0.830 0.08	0.804 0.07 †	0.638 0.10 †	0.802 0.10 †	0.881 0.04	0.854 0.06	0.772 0.13
BREAST	0.391 0.07 †	0.286 0.13	0.331 0.11 †	**0.438** 0.04	0.317 0.12 †	0.334 0.13	0.391 0.08	0.398 0.08
CAR	**0.989** 0.00	0.988 0.00	0.974 0.01 †	0.979 0.01 †	0.978 0.01 †	0.995 0.00	0.981 0.00	0.989 0.01
COMPAS	**0.764** 0.02	0.707 0.05	0.740 0.03	0.753 0.02	0.719 0.07 †	0.825 0.02	0.822 0.02	0.749 0.02
HIV	0.827 0.02 †	0.850 0.03 †	**0.851** 0.02 †	0.790 0.03 †	0.833 0.03 †	0.918 0.01	0.904 0.01	0.854 0.02
LYMPH.	0.788 0.07	**0.831** 0.03	0.774 0.06 †	0.781 0.07 †	0.794 0.05 †	0.836 0.04	0.859 0.02	0.765 0.05
MONKS1	**1.000** 0.00	**1.000** 0.00	0.934 0.06	0.991 0.01 †	**1.000** 0.00 †	0.998 0.00	0.958 0.03	0.975 0.05
MONKS2	**0.745** 0.09 †	0.116 0.11 †	0.476 0.10 †	0.357 0.05	0.148 0.08 †	0.089 0.18	0.148 0.12	0.894 0.08
MONKS3	0.996 0.00 †	0.996 0.00 †	0.996 0.00	0.996 0.00	**0.997** 0.00 †	0.991 0.01	0.986 0.02	0.994 0.01
PRIMARY	0.576 0.05	0.585 0.08	0.577 0.12	**0.601** 0.05	0.327 0.27 †	0.623 0.08	0.603 0.06	0.592 0.06
SOYBEAN	0.660 0.12	**0.689** 0.07	0.600 0.11	0.575 0.07 †	0.630 0.14 †	0.762 0.07	0.767 0.05	0.649 0.21
SPECT	0.885 0.04	0.655 0.15	0.888 0.02	**0.910** 0.02	0.882 0.05 †	0.936 0.02	0.936 0.01	0.892 0.03
TTT	**1.000** 0.00	0.968 0.02 †	0.902 0.03 †	0.875 0.02 †	0.977 0.02 †	0.988 0.00	0.964 0.01	0.972 0.02
VOTE	0.858 0.03 †	**0.898** 0.03 †	0.880 0.04 †	0.882 0.05 †	0.843 0.05 †	0.910 0.01	0.917 0.03	0.874 0.03
KR-VS-KP	0.988 0.00	0.985 0.00	**0.989** 0.00 †	–	0.954 0.01 †	0.987 0.02	0.987 0.00	0.894 0.19
MUSH	**1.000** 0.00	**1.000** 0.00	1.000 0.00	–	**1.000** 0.00 †	1.000 0.00	1.000 0.00	0.963 0.07
ADULT	0.532 0.01 †	0.534 0.02 †	0.542 0.01 †	–	**0.611** 0.01 †	0.319 0.00	0.675 0.00	0.470 0.20
CONNECT-4	**0.860** 0.01	0.741 0.05 †	0.824 0.02 †	–	0.754 0.02 †	0.911 0.01	0.896 0.01	0.893 0.01
MARKET	0.416 0.01 †	0.217 0.02 †	0.419 0.02 †	–	**0.489** 0.02 †	0.070 0.02	0.460 0.03	0.260 0.16
AvgRank-run	2.31	2.82	3.01	3.57	3.28			
AvgRank	2.39	2.84	3.00	3.71	3.05			

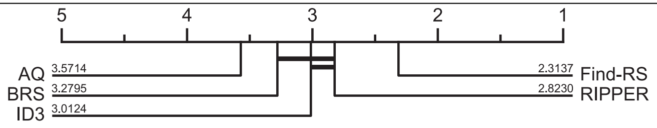


Table 5
Average ruleset similarity (as IoU) of the rulesets found by the BP methods to the ground truth rulesets, using AV encoding for all datasets except MONKS2 and MONKS3. Best results are **bolded**, second best results are underlined.

Dataset	FIND-RS	RIPPER	ID3	AQ
TTT	**1.000**	0.011	0.000	0.000
MONKS1	**1.000**	0.480	0.000	0.050
MONKS2	**0.259**	0.004	0.010	0.000
MONKS3	0.330	**0.333**	**0.333**	0.133

by FIND-RS. This observation can be attributed to the fact that these methods are designed to identify rules with imperfect precision: RIPPER uses an iterative pruning process, and BRS limits a priori the length of the rules. FIND-RS instead, finds perfect precision rules by design. As a result, RIPPER is more likely to identify simpler rulesets, whereas FIND-RS tends to identify more complex rulesets. This characteristic has important implications that will pointed out in the next experiments.

Regarding the average length of the rules found, all of the methods we tested were able to identify short conjunctive rules, except for FIND-RS. Again, this shows that the FIND-RS rule-building process, starting from a random example and progressively generalizing it, on average yields more specific rules.

7.4. Generalization of FIND-RS rules

In Fig. 3, we present the impact of removing constraints from the rules identified in FIND-RS, which leads to performance improvements. Our experiments demonstrate a consistent enhancement across various datasets and encodings compared to the baseline ($p_{gen} = 0$). With a modest p_{gen} we observe a significant improvement, and the performance gradually stabilizes around 0.8. However, in some cases, setting $p_{gen} = 1$ actually reduces the performance. This observation carries crucial implications. Even if we were to construct maximally general rules by exhaustively exploring the version space, it would result in an increased number of false positives and consequently lower performance. Instead, our study reveals the importance of selecting rules that strike a balance between generality and specificity. This methodology offers a rapid and effective solution to the problem.

7.5. Performance of Bayes Point classifiers

Table 8 reports the results of BP and non-ruleset-based methods, with FIND-RS outperforming them on eight datasets (BREAST, LYMPH., MONKS1, MONKS3, PRIMARY, TTT, KR-VS-KP, and MARKET). The performance of FIND-RS is also remarkably close to the baselines on four other datasets (CAR, SPECT, VOTE, and CONNECT-4). The baseline given by the MONKS datasets suggests that FIND-RS is able to identify concepts that can be described by a DNF. A prime example is MONKS2, whose ground truth concept is "exactly two features are equal to one" [51], which can be described by picking combinations of two one-hot features. Also, the method seems to be resilient even using noisy datasets (cfr. MONKS3, which contains 5% label noise).

Therefore, we argue that FIND-RS is best suited to data that can be described by a specific ground-truth ruleset, and can perform quite well even if this hypothesis is not met. Datasets pertaining to games (i.e., TTT, KR-VS-KP, and CONNECT-4), are particularly well suited, given their deterministic nature.

Compared to the base learners' performance, the results show a clear performance improvement across all datasets and methods. Additionally, we observed an overall reduction in the standard deviation of the F1-score, showing how using ensemble methods can reduce the overall variability in performance.

7.6. Analysis of the improvement of the Bayes Point method

In order to experimentally verify the impact of the Bayes Point method, we selected some representative datasets and varied the number of iterations T performed, as shown in Fig. 4. We observe a clear upward trend, that usually stabilizes after $T = 50$. This improvement is given by the ensemble of rules that cooperate together to provide a good classification of the test instances, as usually done by other ensemble methods (e.g., Random Forest).

Table 6
Example rules extracted from MONKS2 by FIND-RS using one-hot encoding. The ground truth concept is "exactly two variables are equal to 1". The first four rules are correct, while the last is more specific than necessary (a2 = 3).

Coverage		Precision		
Train	Test	Train	Test	Rule
0.19	0.12	1.00	1.00	a1 ≠ 1 ∧ a2 ≠ 1 ∧ a3 = 1 ∧ a4 ≠ 1 ∧ a5 ≠ 1 ∧ a6 = 1
0.04	0.02	1.00	1.00	a1 = 1 ∧ a2 ≠ 1 ∧ a3 ≠ 1 ∧ a4 ≠ 1 ∧ a5 = 1 ∧ a6 ≠ 1
0.08	0.06	1.00	1.00	a1 ≠ 1 ∧ a2 = 1 ∧ a3 ≠ 1 ∧ a4 ≠ 1 ∧ a5 ≠ 1 ∧ a6 = 1
0.08	0.08	1.00	1.00	a1 = 1 ∧ a2 ≠ 1 ∧ a3 ≠ 1 ∧ a4 ≠ 1 ∧ a5 ≠ 1 ∧ a6 = 1
0.03	0.05	1.00	1.00	a1 ≠ 1 ∧ a2 = 3 ∧ a3 ≠ 1 ∧ a4 = 1 ∧ a5 ≠ 1 ∧ a6 = 1

Table 7
Average ruleset complexity (number of rules, compl.) and average rule specificity (rule length, spec.) for each dataset using the base methods. Results are averaged across 10 runs.

Method	Find-RS		RIPPER		ID3		AQ		BRS	
dataset	compl.	spec.	compl.	spec.	compl.	spec.	compl.	spec.	compl.	spec.
AUDIO	2.4	9.7	1.7	2.5	2.0	3.8	9.9	2.5	1.4	2.9
BREAST	10.3	8.1	2.2	2.0	12.1	6.5	24.9	3.5	2.7	2.8
CAR	16.6	3.2	11.9	2.7	16.9	7.1	25.0	3.8	7.5	2.6
COMPAS	43.5	6.8	5.6	3.0	48.2	8.1	83.5	4.6	4.8	2.8
HIV	3.8	32.2	8.9	1.6	15.7	7.1	63.4	2.3	7.6	2.7
LYMPH.	7.5	3.1	2.5	1.7	4.3	3.4	11.5	2.2	2.8	2.7
MONKS1	4.0	1.8	4.7	1.9	14.5	3.4	6.7	2.4	4.0	2.8
MONKS2	16.7	5.7	1.5	4.3	24.6	7.0	52.1	5.0	2.1	3.0
MONKS3	2.2	2.2	2.0	2.0	7.2	2.2	2.2	2.2	2.0	2.4
PRIMARY	10.9	4.8	3.1	2.2	10.0	4.4	16.4	3.5	2.9	2.9
SOYBEAN	4.8	6.9	1.3	1.7	5.8	3.3	9.5	3.8	1.4	2.5
SPECT	11.2	3.3	2.1	1.3	9.1	4.8	14.7	2.2	4.6	1.9
TTT	8.0	3.0	9.5	3.3	24.9	6.3	51.5	4.1	8.3	3.0
VOTE	5.2	4.7	2.0	1.8	4.8	4.0	9.0	3.2	2.7	2.7
KR-VS-KP	11.6	7.1	9.0	4.2	11.7	7.9	–	–	4.6	2.7
MUSH	6.9	3.2	8.0	2.5	15.5	2.9	–	–	4.5	2.9
ADULT	323.5	44.4	22.2	6.7	666.2	16.8	–	–	9.0	3.0
CONNECT-4	1911.5	16.1	66.0	7.5	1796.0	17.0	–	–	13.3	3.0
MARKET	404.0	24.8	12.3	6.8	630.0	17.5	–	–	6.0	3.0
Median	10.3	5.7	4.7	2.5	14.5	6.3	15.5	3.3	4.5	2.8

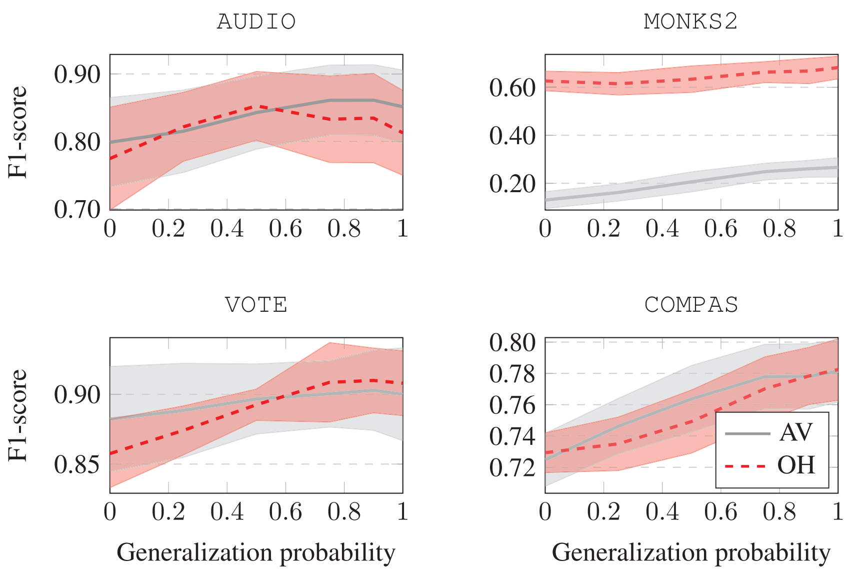


Fig. 3. Impact of dropping constraints from the rules found by FIND-RS. Each constraint is dropped according to a probability, shown in the x-axis. We report the F1-score on the test set alongside its standard deviation (2σ, e.g., 95% of the data variance), computed after 10 runs.

7.7. Ruleset visualization

In Table 9 we show the rules of a BP classifier sorted by importance, using α. We found this visualization to be effective in explaining how the classifier works. Additionally, it can provide a reasonable explanation for a specific classification. Finally, it can be used to debug a model and to check that the inferences made are reasonable. In the example reported, we find the two correct rules, and two additional rules with a very low importance score, that can be effectively pruned using the technique shown below.

7.8. Pruning of Bayes Point classifiers

We proposed a pruning strategy for the ruleset generated by our BP method, which can significantly reduce the number of rules without

Table 8
Average F1-score of BP and baseline classifiers. One-hot encoding was used if it was the best-performing one (marked with †). Results are averaged over 10 runs, with standard deviation reported in smaller size. Best results across interpretable methods are **bolded**, best results overall are underlined.

Dataset	FIND-RS	RIPPER	ID3	AQ	BRS	SVM	RF	TabNet
AUDIO	0.820$_{0.06}$	0.830$_{0.08}$ †	0.870$_{0.05}$ †	0.741$_{0.08}$ †	**0.872**$_{0.05}$ †	0.881$_{0.04}$	0.854$_{0.06}$	0.772$_{0.13}$
BREAST	0.414$_{0.05}$ †	0.280$_{0.06}$	0.273$_{0.08}$ †	**0.477**$_{0.04}$	0.260$_{0.05}$ †	0.334$_{0.13}$	0.391$_{0.08}$	0.398$_{0.08}$
CAR	**0.992**$_{0.00}$ †	0.991$_{0.00}$	0.985$_{0.00}$ †	0.991$_{0.00}$ †	0.988$_{0.00}$ †	0.995$_{0.00}$	0.981$_{0.00}$	0.989$_{0.01}$
COMPAS	0.792$_{0.02}$	**0.801**$_{0.02}$	0.798$_{0.02}$	0.797$_{0.02}$ †	0.796$_{0.02}$ †	0.825$_{0.02}$	0.822$_{0.02}$	0.749$_{0.02}$
HIV	0.883$_{0.02}$ †	**0.888**$_{0.02}$ †	0.884$_{0.01}$ †	0.832$_{0.03}$ †	0.867$_{0.03}$ †	0.918$_{0.01}$	0.904$_{0.01}$	0.854$_{0.02}$
LYMPH.	**0.874**$_{0.03}$	0.826$_{0.04}$	0.848$_{0.03}$ †	0.825$_{0.04}$ †	0.848$_{0.02}$ †	0.836$_{0.04}$	0.859$_{0.02}$	0.765$_{0.05}$
MONKS1	**1.000**$_{0.00}$	**1.000**$_{0.00}$	0.995$_{0.01}$ †	**1.000**$_{0.00}$	**1.000**$_{0.00}$ †	0.998$_{0.00}$	0.958$_{0.03}$	0.975$_{0.05}$
MONKS2	**0.809**$_{0.08}$ †	0.003$_{0.01}$ †	0.529$_{0.11}$ †	0.368$_{0.03}$	0.017$_{0.03}$ †	0.089$_{0.18}$	0.148$_{0.12}$	0.894$_{0.08}$
MONKS3	0.996$_{0.00}$ †	0.996$_{0.00}$	0.996$_{0.00}$	0.996$_{0.00}$ †	**0.997**$_{0.00}$ †	0.991$_{0.01}$	0.986$_{0.02}$	0.994$_{0.01}$
PRIMARY	0.632$_{0.04}$	0.550$_{0.10}$	**0.647**$_{0.07}$	0.599$_{0.05}$	0.488$_{0.17}$ †	0.623$_{0.08}$	0.603$_{0.06}$	0.592$_{0.06}$
SOYBEAN	0.698$_{0.07}$	0.664$_{0.06}$	0.663$_{0.06}$ †	**0.726**$_{0.05}$ †	0.722$_{0.06}$ †	0.762$_{0.07}$	0.767$_{0.05}$	0.649$_{0.21}$
SPECT	**0.929**$_{0.01}$	0.757$_{0.09}$	0.918$_{0.03}$	0.919$_{0.01}$	0.921$_{0.03}$ †	0.936$_{0.02}$	0.936$_{0.01}$	0.892$_{0.03}$
TTT	**1.000**$_{0.00}$	0.997$_{0.00}$ †	0.974$_{0.01}$ †	0.925$_{0.01}$ †	0.998$_{0.01}$ †	0.988$_{0.00}$	0.964$_{0.01}$	0.972$_{0.02}$
VOTE	**0.910**$_{0.02}$ †	0.904$_{0.02}$ †	0.905$_{0.03}$ †	0.868$_{0.03}$ †	0.909$_{0.02}$ †	0.910$_{0.01}$	0.917$_{0.03}$	0.874$_{0.03}$
KR-VS-KP	**0.994**$_{0.00}$	0.988$_{0.00}$	0.993$_{0.00}$ †	–	0.957$_{0.00}$ †	0.987$_{0.02}$	0.987$_{0.00}$	0.894$_{0.19}$
MUSH	**1.000**$_{0.00}$	**1.000**$_{0.00}$	**1.000**$_{0.00}$	–	**1.000**$_{0.00}$ †	1.000$_{0.00}$	1.000$_{0.00}$	0.963$_{0.07}$
ADULT	0.602$_{0.01}$ †	0.572$_{0.01}$ †	0.588$_{0.01}$ †	–	**0.626**$_{0.01}$ †	0.319$_{0.00}$	0.675$_{0.00}$	0.470$_{0.20}$
CONNECT-4	**0.894**$_{0.00}$	0.724$_{0.01}$ †	0.882$_{0.01}$ †	–	0.761$_{0.01}$ †	0.911$_{0.01}$	0.896$_{0.01}$	0.893$_{0.01}$
MARKET	**0.524**$_{0.01}$ †	0.206$_{0.04}$ †	0.428$_{0.01}$ †	–	0.489$_{0.01}$ †	0.070$_{0.02}$	0.460$_{0.03}$	0.260$_{0.16}$
AvgRank-run	2.30	3.18	2.95	3.41	3.17			
AvgRank	2.11	3.24	2.95	3.87	2.84			

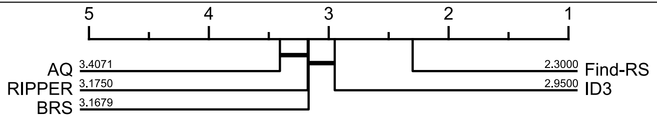


Table 9
Rules extracted from the dataset MONKS3 using FIND-RS (BP). The ground truth concept is (a2 ≠ 3 ∧ a5 ≠ 4) ∨ (a4 = 1 ∧ a5 = 3) with 5% label noise.

α	Coverage		Precision		
	Train	Test	Train	Test	Rule
1.00	0.94	0.95	1.00	1.00	a2 ≠ 3 ∧ a5 ≠ 4
0.99	0.15	0.17	1.00	1.00	a4 = 1 ∧ a5 = 3
0.01	0.11	0.15	1.00	0.74	a4 = 1 ∧ a5 ≠ 2 ∧ a5 ≠ 4 ∧ a6 = 2
0.01	0.06	0.05	1.00	1.00	a2 = 3 ∧ a4 = 1 ∧ a5 = 3
Total	1.00	0.99	1.00	0.95	

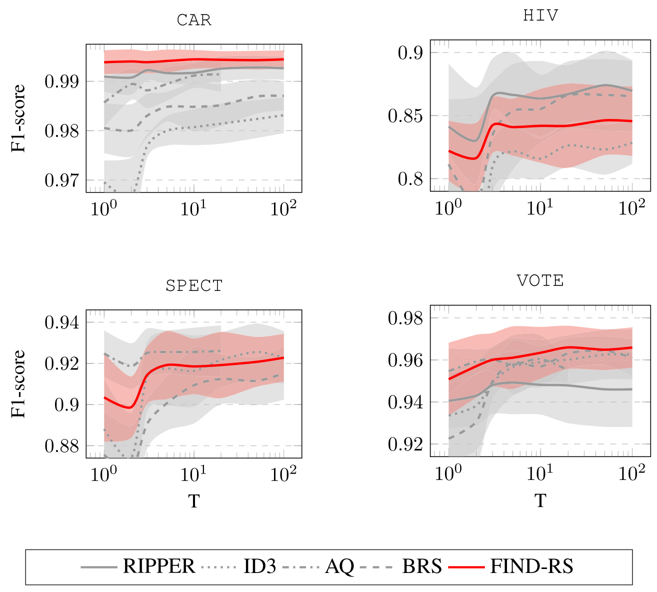


Fig. 4. Effect of changing the hyperparameter T of the Bayes Point technique for each method. In the x-axis, we report the number of iterations T in a log scale. We report the F1-score on the test set alongside its standard deviation (2σ, e.g., 95% of the data variance), computed after 10 runs. We choose the same OH encoding for all datasets and methods.

sacrificing accuracy. By removing redundant or irrelevant rules, the pruned ruleset is more interpretable and easier to understand, while also reducing the risk of overfitting.

To evaluate the effectiveness of our pruning strategy, we varied the degree of pruning progressively increasing the number of rules, and measured the resulting accuracy on both the train and test sets. Fig. 5 presents the results of this experiment for four different datasets.

Our results show that pruning the ruleset can effectively control overfitting and improve generalization performance. In particular, we observed that in three out of four datasets, the accuracy on both the training and test sets increased after adding the most relevant rules (around 10). Then, we can see that the accuracy plateaued, suggesting that our pruning strategy can help decrease the classifier's size. We also observed that, for the MONKS2 dataset, the classifier clearly overfits the data. In this case, using pruning is helpful to limit the overfitting phenomenon.

In summary, our results support the use of pruning as a useful technique for improving the interpretability and generalization performance of rule-based classifiers.

7.9. Performance of pruned Bayes Point classifiers

We compare the performance of the pruned models to the respective base learners in Table 10. We find that FIND-RS has the most noticeable improvement, followed by RIPPER and BRS. Moreover, we analyze the performance if we consider up to k rules for each dataset in Fig. 6.

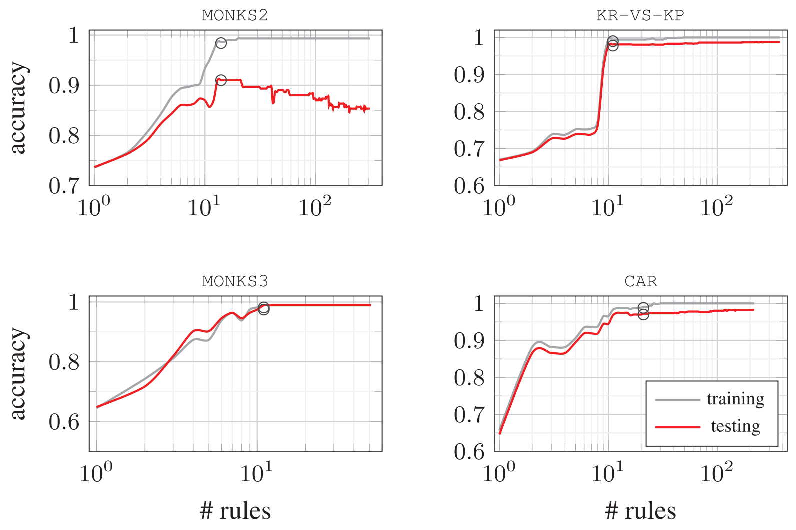


Fig. 5. The plots (log scale) show the behavior of the training and test accuracy varying the number of kept rules in the FIND-RS-BP pruning phase. Rules are sorted in decreasing order of importance. The black circle highlights the accuracy corresponding to the 99% threshold.

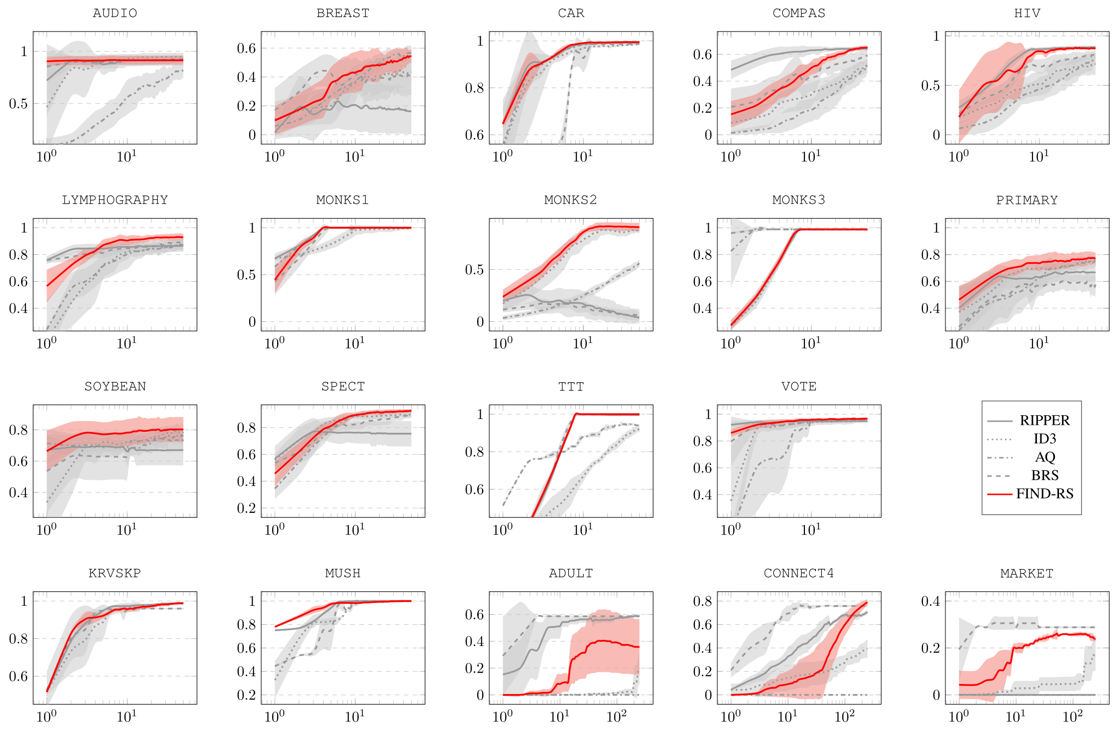


Fig. 6. F1-score (y-axis) for each dataset and method using the BP classifier with the top k rules (x-axis, logarithmic), ordered by decreasing α. FIND-RS is reported in red. Standard deviation is reported (2σ, e.g., 95% of the data variance).

We found α to be effective in giving additional information to the classifier. In fact, performance usually steadily increases and plateaus without decreasing afterward. Generally, we also found that FIND-RS is competitive with other classifiers using few rules, except for COMPAS, CONNECT-4, ADULT, and MARKET, which cannot be easily explained using short and specific DNF formulas.

Finally, we present in Table 11 how performance changes by selecting a fixed number of rules. Here, we can see that FIND-RS excels considering smaller datasets, while BRS can more easily identify general rules in larger datasets.

7.10. Complexity of Bayes Point classifiers

We present the average complexity of the extracted rules in Table 12, comparing it to the baseline and the BP. We observe that the pruning strategy successfully reduces the number of rules used, but not always to the same extent as the baseline, particularly in datasets that cannot be described by a concise DNF formula.

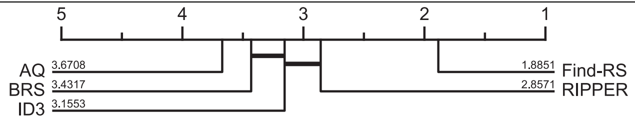
This is a reasonable outcome, as methods that mine a high number of rules do so in datasets that lack a concise ground-truth DNF formula, such as COMPAS, HIV, and KR-VS-KP. In such cases, the proposed pruning strategy may not be as effective, which explains the differences observed in the average complexity of the rules.

Table 10
Average F1-score for baseline and pruned classifiers. Pruned classifiers use a subset of rules to reach the 99% training accuracy of the BP classifier. One-hot encoding was used if it was the best-performing one (marked with †). Results are averaged over 10 runs. Best results are **bolded**.

Method	FIND-RS		RIPPER		ID3		AQ		BRS	
dataset	base	pruned	base	pruned	base	pruned	base	pruned	base	pruned
AUDIO	0.836	0.816	0.830	0.823	0.804	0.868 †	0.638	0.680 †	0.802	**0.871** †
BREAST	0.391	0.408 †	0.286	0.317	0.331	0.271 †	0.438	**0.467**	0.317	0.277 †
CAR	0.989	**0.990**	0.988	0.989	0.974	0.986 †	0.979	0.989 †	0.978	0.988 †
COMPAS	0.764	0.788	0.707	**0.808**	0.740	0.798	0.753	0.785	0.719	0.798 †
HIV	0.827	0.881 †	0.850	**0.887** †	0.851	0.885 †	0.790	0.821 †	0.833	0.866 †
LYMPH.	0.788	**0.866**	0.831	0.825	0.774	0.848 †	0.781	0.822 †	0.794	0.843 †
MONKS1	**1.000**	**1.000**	**1.000**	**1.000**	0.934	0.987	0.991	**1.000** †	**1.000**	**1.000** †
MONKS2	0.745	**0.824** †	0.116	0.130 †	0.476	0.590 †	0.357	0.340	0.148	0.066 †
MONKS3	0.996	0.992 †	0.996	0.992 †	0.996	0.992	0.996	0.992	**0.997**	**0.997** †
PRIMARY	0.576	0.635	0.585	0.535	0.577	**0.646**	0.601	0.605	0.327	0.504 †
SOYBEAN	0.660	0.688	0.689	0.690	0.600	0.653	0.575	0.701 †	0.630	**0.718** †
SPECT	0.885	**0.928**	0.655	0.759	0.888	0.921	0.910	0.914	0.882	0.917 †
TTT	**1.000**	**1.000**	0.968	0.998 †	0.902	0.970 †	0.875	0.915 †	0.977	0.998 †
VOTE	0.858	**0.912** †	0.898	0.904 †	0.880	0.902 †	0.882	0.869 †	0.843	0.900 †
KR-VS-KP	0.988	**0.994**	0.985	0.983	0.989	0.991 †	–	–	0.954	0.957 †
MUSH	**1.000**	1.000	**1.000**	0.995	1.000	0.994	–	–	**1.000**	**1.000** †
AvgRank-run		2.25		2.98		2.80		3.69		3.28
AvgRank		2.12		2.94		3.09		3.64		3.03

Table 11
Average F1-score for pruned classifiers. One-hot encoding was used if it was the best-performing one (marked with †). Results are averaged across 10 runs, with standard deviation reported in smaller size. We selected $k = 10$ rules for small datasets, $k = 25$ rules for medium datasets, and $k = 150$ rules for large datasets. Best results are **bolded**.

	Find-RS	**RIPPER**	**ID3**	**AQ**	**BRS**
AUDIO	**0.933** $_{0.05}$	0.913 $_{0.05}$ †	0.933 $_{0.04}$ †	0.624 $_{0.12}$	0.909 $_{0.05}$
BREAST	**0.577** $_{0.06}$	0.429 $_{0.07}$	0.387 $_{0.11}$ †	0.505 $_{0.07}$	0.402 $_{0.12}$
CAR	**0.992** $_{0.00}$ †	0.991 $_{0.00}$	0.971 $_{0.01}$ †	0.988 $_{0.00}$	0.949 $_{0.06}$
COMPAS	0.512 $_{0.06}$	**0.630** $_{0.03}$	0.322 $_{0.08}$	0.198 $_{0.09}$	0.432 $_{0.11}$
HIV	0.844 $_{0.03}$ †	**0.867** $_{0.02}$ †	0.518 $_{0.12}$ †	0.616 $_{0.06}$	0.692 $_{0.07}$
LYMPH.	**0.909** $_{0.03}$	0.857 $_{0.04}$	0.864 $_{0.03}$ †	0.882 $_{0.04}$ †	0.846 $_{0.04}$
MONKS1	**1.000** $_{0.00}$	**1.000** $_{0.00}$	0.978 $_{0.05}$	**1.000** $_{0.00}$ †	**1.000** $_{0.00}$
MONKS2	**0.877** $_{0.02}$ †	0.155 $_{0.13}$ †	0.805 $_{0.07}$ †	0.297 $_{0.04}$	0.139 $_{0.05}$
MONKS3	**0.988** $_{0.00}$ †	0.988 $_{0.00}$	**0.988** $_{0.00}$	**0.988** $_{0.00}$	0.988 $_{0.00}$
PRIMARY	**0.745** $_{0.06}$	0.640 $_{0.07}$	0.695 $_{0.05}$	0.636 $_{0.09}$	0.563 $_{0.07}$
SOYBEAN	**0.783** $_{0.09}$	0.688 $_{0.03}$ †	0.728 $_{0.06}$	0.726 $_{0.07}$ †	0.725 $_{0.06}$
SPECT	**0.898** $_{0.02}$	0.763 $_{0.08}$	0.885 $_{0.02}$	0.895 $_{0.03}$	0.826 $_{0.04}$
TTT	**1.000** $_{0.00}$	0.999 $_{0.00}$ †	0.828 $_{0.03}$ †	0.905 $_{0.03}$ †	0.998 $_{0.01}$
VOTE	0.959 $_{0.01}$ †	0.947 $_{0.01}$ †	0.941 $_{0.02}$ †	**0.963** $_{0.00}$ †	0.947 $_{0.02}$
KR-VS-KP	**0.990** $_{0.01}$	0.988 $_{0.00}$	0.988 $_{0.01}$ †	–	0.960 $_{0.00}$
MUSH	1.000 $_{0.00}$ †	**1.000** $_{0.00}$	**1.000** $_{0.00}$	–	**1.000** $_{0.00}$
ADULT	0.373 $_{0.22}$ †	0.579 $_{0.01}$ †	0.078 $_{0.03}$	–	**0.586** $_{0.00}$
CONNECT-4	0.708 $_{0.06}$ †	0.680 $_{0.03}$ †	0.379 $_{0.05}$	–	**0.758** $_{0.01}$
MARKET	0.259 $_{0.01}$ †	0.000 $_{0.00}$	0.064 $_{0.02}$ †	–	**0.288** $_{0.00}$
AvgRank-run	1.89	2.86	3.16	3.67	3.43
AvgRank	2.05	2.74	3.61	3.66	2.95



7.11. Application to high-dimensional datasets

We also report the performance of FIND-RS applied to a high-dimensional dataset. Here, rule-learning methods such as RIPPER and BRS do not finish because they do not scale well for a high number of features, while other methods such as TabNet struggle to converge. FIND-RS, as discussed in Section 5.3, scales linearly in the number of features. Hence, it can handle similar datasets much faster.

In Table 13 we reported the performance of FIND-RS using the F1-score along some baselines. We observed a couple of peculiarities:

- the rule found by FIND-RS is only one: this means that a single rule can effectively describe the dataset. Because of this reason, the Bayes Point technique fails to bring any improvement, since a singular bin with the same examples will lead to the same rule;
- instead, using bootstrap for initializing each single Bayes Point iteration brings a significant improvement while increasing the number of unique rules;
- the rules found involve many features (i.e., are more specific). Here, it is necessary to increase p_{gen} in order to balance the recall of the classifier (i.e., have fewer false negatives).

7.12. Comparison with LIME explanations

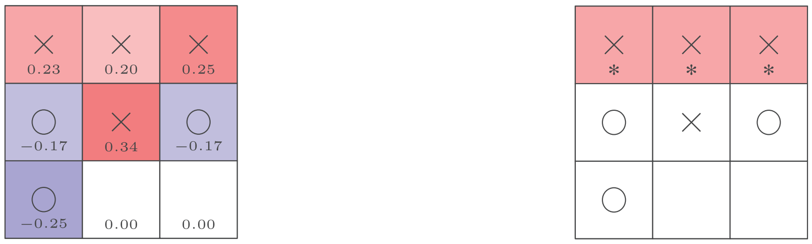
Finally, we want to report a selected example, shown in Fig. 7, where we compare an explanation given by LIME and FIND-RS, using the tic-tac-toe dataset. Explanations given by LIME are generally useful:

Table 12
Average BP ruleset complexity (number of rules) for each dataset. Results are averaged across 10 runs. Pruned classifiers use a subset of rules to reach the 99% training accuracy of the BP classifier.

Method	AQ			BRS			Find-RS			ID3			RIPPER		
dataset	base	BP	pr.	base	BP	pr.	base	BP	pr.	base	BP	pr.	base	BP	pr.
AUDIO	9.9	123.3	65.6	1.4	92.1	43.3	2.4	21.2	3.7	2.0	78.1	48.3	1.6	15.3	5.1
BREAST	24.9	418.8	240.9	2.7	192.8	166.4	10.3	877.7	398.4	12.1	1209.6	1011.4	2.2	34.6	8.4
CAR	25.0	257.2	45.0	7.5	74.2	13.6	13.1	201.6	19.2	16.9	639.3	196.2	11.9	28.5	10.2
COMPAS	85.5	1550.5	872.9	4.8	385.7	300.2	43.5	3534.9	1671.7	48.2	4340.6	3399.6	5.6	172.9	68.1
HIV	63.4	665.3	269.6	7.6	458.3	358.7	3.8	370.8	181.7	15.7	1294.0	930.0	8.9	117.2	43.3
LYMPH.	11.5	129.0	55.3	2.8	196.4	157.2	7.5	350.9	72.7	4.3	308.4	199.5	2.5	26.4	7.5
MONKS1	8.2	49.0	4.0	4.0	35.9	13.2	4.0	4.0	4.0	13.4	1044.3	462.2	4.7	23.6	4.0
MONKS2	52.1	408.2	178.4	2.1	120.4	16.4	16.7	103.8	18.3	24.6	2292.3	1690.0	1.5	89.3	11.7
MONKS3	3.1	34.2	2.1	2.0	20.5	3.2	2.2	4.3	1.9	7.2	11.2	6.9	6.8	10.0	7.0
PRIMARY	16.4	193.8	107.5	2.9	185.4	111.6	10.9	507.8	116.4	10.0	490.7	371.0	3.1	42.3	16.7
SOYBEAN	9.5	140.5	78.4	1.4	124.2	73.4	4.8	197.4	45.9	3.9	268.4	210.3	1.3	14.0	2.9
SPECT	14.7	114.4	41.8	4.6	332.1	203.9	11.2	544.1	109.5	9.1	767.0	505.2	2.1	18.6	14.2
TTT	51.5	880.0	247.5	8.3	87.3	11.3	8.0	8.3	8.0	24.9	1962.5	1135.1	9.5	180.8	9.2
VOTE	9.0	157.0	71.5	2.7	139.1	66.7	5.2	306.2	82.3	4.8	251.7	132.5	2.0	32.2	4.5
MUSH	–	–	–	4.5	36.0	15.0	6.9	295.7	47.2	15.5	61.0	12.0	8.0	28.5	6.5
KR-VS-KP	–	–	–	4.6	44.3	15.6	11.6	740.7	239.5	11.7	337.9	84.9	9.0	180.6	37.6
ADULT	–	–	–	9.0	65.2	–	323.5	6453.0	–	666.2	13 318.2	–	22.2	375.2	–
CONNECT-4	–	–	–	13.3	66.3	–	1911.5	36 546.2	–	1796.0	35 437.0	–	66.0	1087.7	–
MARKET	–	–	–	6.0	82.7	–	404.0	8172.0	–	630.0	12 334.3	–	12.3	215.3	–

Table 13
Average F1-score, rule length (specificity), and distinct rules (complexity) for the dataset RNA-SEQ, and different experimental settings. Results are averaged across 10 runs, with standard deviation reported in smaller size.

Method	F1-score	avg. rule len.	Unique rules
SVM	**0.998** 0.0017	–	–
RF	0.996 0.0025	6.063 0.0695	1895.40 71.1498
CART	0.969 0.0152	4.18 0.1659	8.2 0.8367
FIND-RS (T=1, p_{gen} = 0.0)	0.184 0.0576	9950.30 86.6808	1.00 0.0000
FIND-RS (T=1, p_{gen} = 0.9)	0.717 0.0318	991.60 23.7590	1.00 0.0000
FIND-RS (T=1, p_{gen} = 1.0)	0.921 0.0162	17.20 3.5637	1.00 0.0000
FIND-RS (T=10, p_{gen} = 1.0)	0.921 0.0162	17.20 3.5637	1.00 0.0000
FIND-RS (T=10, p_{gen} = 1.0, bootstrap)	0.993 0.0013	12.54 1.2198	10.00 0.0000
FIND-RS (T=100, p_{gen} = 1.0, bootstrap)	**0.994** 0.0038	13.35 0.4466	100.00 0.0000



(a) Explanation provided by LIME (b) Explanation provided by FIND-RS

Fig. 7. Comparison of rules given by LIME (using an SVM as a base learner) and by FIND-RS.

they provide an importance estimation of each feature of a selected test data point. Each feature can vote if one class or the other is more likely, giving a numeric score that can be added or compared. Each score is computed by trying to swap a feature value with other values. This is generally a good assumption, but, as reported by this example, can fail in the presence of hard rules, such as the three-in-a-row constraint of the game. In fact, the score of the middle cell is the highest despite not being related to the classification. This is accomplished through LIME's internal mechanism: by switching the value of the middle cell from naught (◯) to cross (×), the classifier is prompted to predict an impossible scenario where both players would win. The explanation provided by FIND-RS (and other rule learning methods) aligns more to the true rules of the game, without encoding additional constraints. Rule-based methods have the ability to capture relationships between multiple variable, and provide sparser explanations.

8. Possible extensions

Although FIND-RS has been originally designed for categorical data, it can also be applied to real-valued data after a discretization step, as it is done with the ADULT and MARKET datasets. Note that this is a standard procedure in many rule learning models, including RIPPER, BRS, and SBRL. In addition, many local explanability methods perform this type of discretization, such as LIME. Each rule can be made to encode if an attribute belongs to a bin or not. There are some works [52] which discuss different techniques for discretizing data more efficiently and can be considered as a starting point in future extensions of this work.

Another approach to deal with real-valued attributes in FIND-RS could use interval-based rules. For a given bin, the algorithm would start from the most specific interval (e.g., $1.23 \leq x_3 \leq 1.23$) corresponding to the positive seed instance in the bin, and then it would gradually grow the interval to cover more and more positive examples (e.g., $-3.21 \leq x_3 \leq 4.56$) and no negative ones. We want to consider this case for a future extension of this work.

As is, the method does not support multiclass classification. The most straightforward way to extend it to multiclass problems would be to use one classifier (a rule set) for each class, in a one-vs-rest fashion. Note that, by construction, no more than one of the rule sets associated with the different classes can cover an instance. However, this could lead to cases where the classifier is unable to give a classification as output when all the rule sets associated to the different classes do not cover the instance (i.e., having a reject option). This is plausible given the nature of the concept learning methods, which attempt to summarize only the characteristics of the positive data.

Finally, in the future, we would like to provide an enhanced version using an optimized implementation, using faster data structures and caching strategies, as often done with similar rule learning methods.

9. Conclusions

This paper proposed principled aggregation methods based on Bayesian learning theory to improve the classification performance and robustness of rule-based classifiers while ensuring high interpretability. We showed that BP aggregation increases the performance of state-of-the-art ruleset learning algorithms.

In addition, we proposed FIND-RS a novel sequential covering algorithm for ruleset learning. FIND-RS has shown to obtain the greatest benefit from BP aggregation, achieving performance close to less interpretable methods, such as SVM and RF.

High interpretability is achieved by considering only the most relevant rules based on their importance in the decision-making process. Extracting importance values is highly effective in reducing the size of the classifier while maintaining performance.

Our methodology is general enough to be applicable to any rule-based method and enhances performance without losing much in interpretability. We believe that these approaches are a step forward to gaining the trust and transparency of AI systems and boosting performance and portability.

CRediT authorship contribution statement

Luca Bergamin: Writing – review & editing, Writing – original draft, Visualization, Software, Methodology, Investigation, Data curation, Conceptualization. **Mirko Polato:** Writing – review & editing, Writing – original draft, Supervision, Methodology, Investigation, Formal analysis. **Fabio Aiolli:** Writing – review & editing, Supervision, Methodology, Investigation, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The data that support the findings of this study are openly available on Github at https://github.com/BouncyButton/bayes-point-learning.

Acknowledgment

The work of Mirko Polato has been partially supported by the spoke “FutureHPC & BigData” of the ICSC – Centro Nazionale di Ricerca in High-Performance Computing, Big Data and Quantum Computing funded by European Union – NextGenerationEU.

Appendix. Proof of Proposition 1

At every iteration t of FIND-RS holds that

$\forall j < i \in [k],\ \nexists \mathbf{x} \in B_i \mid \mathbf{r}_j \geq \mathbf{x}$,

where the current hypothesis is $D^t = \{\mathbf{r}_1, \dots, \mathbf{r}_k\}$.

Proof. We give proof by induction on the number of iterations t.

Case base: $t = 0$, then, by design, $D^0 = \emptyset$, thus Proposition 1 is hollowly true;

Inductive step: let us assume that Proposition 1 holds up until iteration t in which $D^t = \langle \mathbf{r}_1, \dots, \mathbf{r}_k \rangle$ where each conjunction $\mathbf{r}_i$ has been learnt using FIND-S over B_i. At iteration $t+1$, a new positive instance $\mathbf{x}$ is considered, and only one of the following two scenarios must be true.

(1) $D^t \ngeq \mathbf{x}$, i.e., $\nexists \mathbf{r}_i \mid \text{generalize}(\mathbf{r}_i, \mathbf{x})$ returns a hypothesis consistent with $\mathcal{N}$, thus $D^{t+1} = D^t \vee \mathbf{r}_{k+1}$ where $\mathbf{r}_{k+1} = \mathbf{x}$. Hence, by construction, $\nexists i \leq k \mid \mathbf{r}_i \geq \mathbf{x}$, so Proposition 1 holds.

(2) $D^t \geq \mathbf{x}$, i.e., $\exists \mathbf{r}_i \mid \mathbf{r}' = \text{generalize}(\mathbf{r}_i, \mathbf{x}) \geq \mathbf{x}$ and $\mathbf{r}'$ is consistent with $\mathcal{N}$. Since FIND-RS generalizes each conjunctive term in order, then $\forall j < i$, $\mathbf{r}_j$ cannot be generalized. After the generalization step, $\mathbf{r}'$ can be either unchanged (i.e., $\mathbf{r}_i \geq \mathbf{x}$) or it can be a generalization of $\mathbf{r}_i$, i.e., $\mathbf{r}' \geq \mathbf{r}_i$. In the former case, the overall hypothesis does not change and Proposition 1 holds. In the latter case, we have to show that $\mathbf{r}'$ does not cover any instance in B_j for $j > i$. Since $\mathbf{r}_i$ has been created before any $\mathbf{r}_{j>i}$, then $\mathbf{r}_i$ cannot be generalized to cover any $\mathbf{x} \in B_{j>i}$ (otherwise it would have changed previously to cover $\mathbf{x}$). Thus, Proposition 1 holds. □

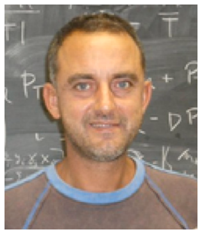
References

[1] R.S. Michalski, On the quasi-minimal solution of the general covering problem, in: International Symposium on Information Processing, 1969, pp. 125–128.
[2] D. Martens, B. Baesens, T.V. Gestel, J. Vanthienen, Comprehensible credit scoring models using rule extraction from support vector machines, New Inst. Econ. (2007).
[3] F. Doshi-Velez, B. Kim, Towards a rigorous science of interpretable machine learning, 2017, arXiv: Machine Learning.
[4] Y. Zhang, P. Tiňo, A. Leonardis, K. Tang, A survey on neural network interpretability, IEEE Trans. Emerg. Top. Comput. Intell. 5 (2020) 726–742.
[5] B. Kim, M. Wattenberg, J. Gilmer, C.J. Cai, J. Wexler, F.B. Viégas, R. Sayres, Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (TCAV), in: International Conference on Machine Learning, 2017.
[6] S. Vashishth, S. Upadhyay, G.S. Tomar, M. Faruqui, Attention interpretability across NLP tasks, 2019, ArXiv arXiv:1909.11218.
[7] M. Geva, R. Schuster, J. Berant, O. Levy, Transformer feed-forward layers are key-value memories, 2020, ArXiv arXiv:2012.14913.
[8] M. Narayanan, E. Chen, J. He, B. Kim, S.J. Gershman, F. Doshi-Velez, How do humans understand explanations from machine learning systems? An evaluation of the human-interpretability of explanation, 2018, ArXiv arXiv:1902.00006.
[9] T.M. Mitchell, Machine Learning, first ed., McGraw-Hill, Inc., USA, 1997.
[10] R. O'Donnell, Analysis of Boolean Functions, Cambridge University Press, 2014.
[11] C. Rudin, Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead, 2019, arXiv:1811.10154 [cs, stat].
[12] J.R. Quinlan, Induction of decision trees, Mach. Learn. 1 (1986) 81–106.
[13] D. Che, Q. Liu, K.M. Rasheed, X. Tao, Decision tree and ensemble learning algorithms with their applications in bioinformatics, Adv. Exp. Med. Biol. 696 (2011) 191–199.
[14] J. Fürnkranz, Rule learning, in: C. Sammut, G.I. Webb (Eds.), Encyclopedia of Machine Learning, Springer US, Boston, MA, 2010, pp. 875–879, http://dx.doi.org/10.1007/978-0-387-30164-8_738.
[15] W.W. Cohen, Fast effective rule induction, in: Machine Learning Proceedings 1995, Elsevier, 1995, pp. 115–123.
[16] L. Breiman, Bagging predictors, Mach. Learn. 24 (1996) 123–140.
[17] R. Herbrich, T. Graepel, Large scale Bayes point machines, in: Advances in Neural Information Processing Systems, vol. 13, MIT Press, 2000, pp. 528–534, URL https://proceedings.neurips.cc/paper/2000/hash/333222170ab9edca4785c39f55221fe7-Abstract.html.
[18] J.R. Quinlan, Discovering rules by induction from large collections of examples, Expert Syst. Micro Electron. Age (1979).
[19] J.R. Quinlan, Induction of decision trees, Mach. Learn. 1 (1) (1986) 81–106.

[20] L. Breiman, J.H. Friedman, R.A. Olshen, C.J. Stone, Classification and regression trees, Routledge, 2017.
[21] P. Clark, T. Niblett, The CN2 induction algorithm, Mach. Learn. 3 (4) (1989) 261–283.
[22] J. Fürnkranz, G. Widmer, Incremental reduced error pruning, in: Machine Learning Proceedings 1994, Elsevier, 1994, pp. 70–77.
[23] J. Fürnkranz, T. Kliegr, A brief overview of rule learning, in: International Symposium on Rules and Rule Markup Languages for the Semantic Web, Springer, 2015, pp. 54–69.
[24] G.I. Webb, OPUS: An efficient admissible algorithm for unordered search, J. Artificial Intelligence Res. 3 (1995) 431–465.
[25] B. Letham, C. Rudin, T.H. McCormick, D. Madigan, Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model, Ann. Appl. Stat. 9 (3) (2015) 1350–1371.
[26] H. Yang, C. Rudin, M. Seltzer, Scalable Bayesian rule lists, in: International Conference on Machine Learning, PMLR, 2017, pp. 3921–3930.
[27] T. Wang, C. Rudin, F. Doshi-Velez, Y. Liu, E. Klampfl, P. MacNeille, A Bayesian framework for learning rule sets for interpretable classification, J. Mach. Learn. Res. 18 (2017) 70:1–70:37.
[28] W.W. Cohen, Y. Singer, A simple, fast, and effective rule learner, AAAI/IAAI 99 (335–342) (1999) 3.
[29] L. Breiman, Random forests, Mach. Learn. 45 (1) (2001) 5–32.
[30] J.H. Friedman, B.E. Popescu, Predictive learning via rule ensembles, Ann. Appl. Stat. 2 (3) (2008) 916–954, http://dx.doi.org/10.1214/07-AOAS148.
[31] M. Nalenz, T. Augustin, Compressed rule ensemble learning, in: G. Camps-Valls, F.J.R. Ruiz, I. Valera (Eds.), Proceedings of the 25th International Conference on Artificial Intelligence and Statistics, PMLR, in: Proceedings of Machine Learning Research, vol. 151, 2022, pp. 9998–10014, URL https://proceedings.mlr.press/v151/nalenz22a.html.
[32] C. Bénard, G. Biau, S.D. Veiga, E. Scornet, SIRUS: Stable and Interpretable RUle Set for classification, Electron. J. Stat. 15 (1) (2021) 427–505, http://dx.doi.org/10.1214/20-EJS1792.
[33] E. Angelino, N. Larus-Stone, D. Alabi, M.I. Seltzer, C. Rudin, Learning certifiably optimal rule lists for categorical data, J. Mach. Learn. Res. 18 (2017) 234:1–234:78.
[34] J. Yu, A. Ignatiev, P.J. Stuckey, P.L. Bodic, Learning optimal decision sets and lists with SAT, J. Artificial Intelligence Res. 72 (2021) 1251–1279.
[35] L. Katzir, G. Elidan, R. El-Yaniv, Net-{dnf}: Effective deep modeling of tabular data, in: International Conference on Learning Representations, 2021, URL https://openreview.net/forum?id=73WTGs96kho.
[36] L. Dierckx, R. Veroneze, S. Nijssen, RL-Net: Interpretable Rule Learning with Neural Networks, in: H. Kashima, T. Ide, W.-C. Peng (Eds.), Advances in Knowledge Discovery and Data Mining, in: Lecture Notes in Computer Science, vol. 13935, Springer Nature Switzerland, Cham, 2023, pp. 95–107, http://dx.doi.org/10.1007/978-3-031-33374-3_8.
[37] Z. Wang, W. Zhang, N. Liu, J. Wang, Scalable rule-based representation learning for interpretable classification, Adv. Neural Inf. Process. Syst. 34 (2021).
[38] F. Beck, J. Fürnkranz, An empirical investigation into deep and shallow rule learning, 2021, arXiv preprint arXiv:2106.10254.
[39] M.T. Ribeiro, S. Singh, 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, KDD '16, Association for Computing Machinery, New York, NY, USA, 2016, pp. 1135–1144, http://dx.doi.org/10.1145/2939672.2939778.
[40] S.M. Lundberg, S.-I. Lee, A unified approach to interpreting model predictions, in: Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS '17, Curran Associates Inc., Red Hook, NY, USA, 2017, pp. 4768–4777.
[41] S.M. Lundberg, G.G. Erion, S.-I. Lee, Consistent individualized feature attribution for tree ensembles, 2019, arXiv:1802.03888.
[42] T.J. Hastie, R. Tibshirani, J.H. Friedman, The Elements of Statistical Learning, Springer, 2009, p. 282.
[43] J.B. Tenenbaum, Bayesian modeling of human concept learning, in: NIPS, 1998.
[44] B. Schölkopf, A.J. Smola, F. Bach, et al., Learning with kernels: support vector machines, regularization, optimization, and beyond, MIT Press, 2002.
[45] S.O. Arik, T. Pfister, TabNet: Attentive interpretable tabular learning, Proc. AAAI Conf. Artif. Intell. 35 (8) (2021) 6679–6687, http://dx.doi.org/10.1609/aaai.v35i8.16826, URL https://ojs.aaai.org/index.php/AAAI/article/view/16826.
[46] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (1) (2006) 1–30, URL http://jmlr.org/papers/v7/demsar06a.html.
[47] H. Ismail Fawaz, G. Forestier, J. Weber, L. Idoumghar, P.-A. Muller, Deep learning for time series classification: a review, Data Min. Knowl. Discov. 33 (4) (2019) 917–963, http://dx.doi.org/10.1007/s10618-019-00619-1.
[48] M. Friedman, A comparison of alternative tests of significance for the problem of m rankings, Ann. Math. Stat. 11 (1) (1940) 86–92, URL http://www.jstor.org/stable/2235971.
[49] F. Wilcoxon, Individual comparisons by ranking methods, Biom. Bull. 1 (6) (1945) 80–83, URL http://www.jstor.org/stable/3001968.
[50] S. Holm, A simple sequentially rejective multiple test procedure, Scand. J. Stat. 6 (2) (1979) 65–70, URL http://www.jstor.org/stable/4615733.
[51] S. Thrun, The monk's problems: A performance comparison of different learning algorithms, CMU-CS-91-197, sch, 1991.
[52] W. Zhang, Y. Liu, Z. Wang, J. Wang, Learning to binarize continuous features for neuro-rule networks, in: E. Elkind (Ed.), Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23, International Joint Conferences on Artificial Intelligence Organization, 2023, pp. 4584–4592, http://dx.doi.org/10.24963/ijcai.2023/510, Main Track.

L. Bergamin obtained his M.Sc. in Computer Science from the University of Padova and is presently enrolled in the same university's Ph.D. program in Brain, Mind, and Computer Science. He is currently focusing on the development of interpretable machine learning technologies. His research revolves around reasoning and neuro-symbolic techniques, exploring theoretical foundations and practical applications. Additionally, he actively collaborates on the development of machine learning solutions for healthcare research.

M. Polato is an Assistant Professor at the Department of Computer Science of the University of Turin. He received his M.Sc. and his Ph.D. in Brain, Mind, and Computer Science from the University of Padova (Italy) in 2013 and 2018, respectively. He served as a Program Committee member of several international conferences and as a referee for several international journals. He authored more than 40 research products, including international peer-reviewed conferences and journal papers. His research mainly focuses on federated learning and interpretable machine learning. More information about Mirko can be found at https://makgyver.github.io.



F. Aiolli received a Master's Degree and a Ph.D. in Computer Science both from the University of Pisa. He was Post-doc at the University of Pisa, Paid Visiting Scholar at the University of Illinois at Urbana-Champaign (IL), USA, and Post-doc at the University of Padova. He is currently Associate Professor at the University of Padova. His research activity is mainly in the area of Machine Learning and Information Retrieval.