Neural Rule Lists: Learning Discretizations, Rules, and Order in One Go

Sascha Xu[*], **Nils Philipp Walter**[*] **and Jilles Vreeken**
CISPA Helmholtz Center for Information Security
{sascha.xu, nils.walter, vreeken}@cispa.de

Abstract

Interpretable machine learning is essential in high-stakes domains like healthcare. Rule lists are a popular choice due to their transparency and accuracy, but learning them effectively remains a challenge. Existing methods require feature pre-discretization, constrain rule complexity or ordering, or struggle to scale. We present NEURULES, a novel end-to-end framework that overcomes these limitations. At its core, NEURULES transforms the inherently combinatorial task of rule list learning into a differentiable optimization problem, enabling gradient-based learning. It simultaneously discovers feature conditions, assembles them into conjunctive rules, and determines their order—without pre-processing or manual constraints. A key contribution here is a gradient shaping technique that steers learning toward sparse rules with strong predictive performance. To produce ordered lists, we introduce a differentiable relaxation that, through simulated annealing, converges to a strict rule list. Extensive experiments show that NEURULES consistently outperforms combinatorial and neural baselines on binary as well as multi-class classification tasks across a wide range of datasets.

1 Introduction

Rule lists [Cohen, 1995] are conceptually simple classifiers that are inherently interpretable and surprisingly effective [Rudin, 2019]. As the name implies, rule lists are ordered lists of rules of the kind "**if** `Thalassemia` = normal **and** `Resting BP` < 151 **then predict** *yes*". Each rule consists of a set of conditions, when met, trigger a specific prediction. The model evaluates the rules in order and applies the first one whose conditions are satisfied, yielding a simple and interpretable decision path. We give an example rule list in Fig. 1. Because of their transparency requirements, rule lists are often used in high-stakes applications, e.g. stroke prediction [Letham et al., 2015], credit risk evaluation [Bhatore et al., 2020], and criminal justice [Lakkaraju and Rudin, 2017].

Inferring rule lists is a challenging combinatorial problem in which we face a super-exponential search space with little exploitable structure. First, there is the problem of which atomic rule conditions, or predicates, to use. Existing strategies rely on pre-discretization, which incurs a loss of information. Second, there is the problem of combining these predicates into rule-heads, i.e. the set of conditions that specify when a rule triggers. Existing approaches are either combinatorial, and have to restrict the search space in favor of runtime [Cohen, 1995, Yang et al., 2017, Proenca and van Leeuwen, 2020, Angelino et al., 2018], or use differentiable relaxations that often lead to verbose rule-heads [Wang et al., 2021, Qiao et al., 2021, Dierckx et al., 2023]. Third, there is the order of the list. Except for Angelino et al. [2018], existing methods learn rules greedily or consider a fixed list order [Dierckx et al., 2023], either of which can result in subpar accuracy.

[*]Equal contribution

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

if not (`Chest pain`) **and not** (`Exercise chest pain`) **and** (`0.88 < ST depression < 5.24`)
  then predict *yes*
else if not (`Chest pain`) **and** (`45.5 < Age < 66.4`) **and** (`Sex = female`)
  then predict *yes*
else if (`Resting BP < 151`) **and** (`ST depression < 2.65`) **and** (`Thal = normal`)
  then predict *no*
  ...
else predict *yes*

Figure 1: First three rules of the rule list learned with NEURULES on the Heart Disease dataset. NEURULES jointly optimizes thresholds, rule aggregation, and list order.

In this paper, we propose a novel approach that *jointly* addresses all of the above challenges. We introduce NEURULES, a differentiable end-to-end architecture that learns predicates, conjunctive rules, and the order of the rule list in a single unified framework—without manual pre-processing or structural constraints. Unlike prior work, we relax the discretization of features, the construction of rules, and the ordering of the rule list, allowing all components to be learned jointly. To overcome the limitations of existing methods, NEURULES builds on four key innovations:

(i) *Soft predicate learning*. Instead of relying on fixed discretization, we learn soft predicates that converge to crisp thresholds via simulated annealing.

(ii) *Differentiable rule composition*. We use a conjunction layer that supports robust composition of predicates while mitigating vanishing gradients.

(iii) *Gradient shaping for sparsity*. A targeted mechanism promotes the learning of rule heads that are both sparse and accurate.

(iv) *Learned rule ordering*. We model rule order as a soft priority, which is annealed into a strict ordering by the end of training.

Together, these elements yield a crisp and succinct rule list upon convergence. NEURULES thus provides a holistic differentiable analog of rule lists that scales well and natively supports binary and multi-class classification. Extensive experiments show that NEURULES consistently outperforms combinatorial and neuro-symbolic baselines developed over the past 30 years.

2 Preliminaries

We consider a dataset of n pairs $\{(\mathbf{x}, y)\}_{k=1}^{n}$ consisting of a *descriptive feature vector* $\mathbf{x} \in \mathbb{R}^d$ of d real-valued features and a discrete-valued *target label* $y \in \mathcal{Y}$. We assume each sample $(\mathbf{x}, y)$ to be a realization of a pair of random variables $(\mathbf{X}, Y) \sim P_{\mathbf{X},Y}$, drawn iid.

Rules. We consider predictive *rules* $r : \mathcal{X} \to \mathcal{Y}$ that consist of a *rule head* $h : \mathcal{X} \to \{0, 1\}$ and a *consequent* $c \in \mathcal{Y}$. A rule head consists of (a logical conjunction of) binary predicates $\pi : \mathbb{R} \to \{0, 1\}$ that represent the presence of a characteristic, e.g. "$18 <$ `Age` < 65". We consider predicates π to be threshold functions on a feature $x \in \mathbb{R}$ parameterized by a lower and upper bound, $a, b \in \mathbb{R}$, as

$$\pi\left(x; a, b\right) = \mathbb{1}[a < x < b]\ , \tag{1}$$

where $a = -\infty$ resp. $b = +\infty$ are interpreted as no condition. Our formulation also works on binary features, so that we can accommodate both categorical features through one-hot encoding and learned predicates with more complex structure [Wang et al., 2015]. We allow every rule its own predicates. That is, every rule r_i has its own thresholds a_{ij}, b_{ij} per feature x_j. We use the shorthand π_{ij} to denote $\pi(x_j; a_{ij}, b_{ij})$. We write $\boldsymbol{\pi}_i$ to denote all predicates for rule r_i, i.e. $\boldsymbol{\pi}_i = \{\pi_{ij}\}_{j=1}^{d}$.

We say a rule head *holds* iff $h_i(\mathbf{x}; \theta_i) = \bigwedge_j \pi_{ij}(x_j; a_{ij}, b_{ij}) = 1$, where $\theta_i = \{(a_{ij}, b_{ij})\}_{j=1}^{d}$ are the parameters of the predicates. A rule r maps input $\mathbf{x}$ to label c if the rule head holds, otherwise, its output is undefined ($\bar{c}$). Formally,

$$r(\mathbf{x}; c, \theta) = \begin{cases} c & \text{if } h(\mathbf{x}; \theta) = 1 \\ \bar{c} & \text{else} \end{cases} \tag{2}$$

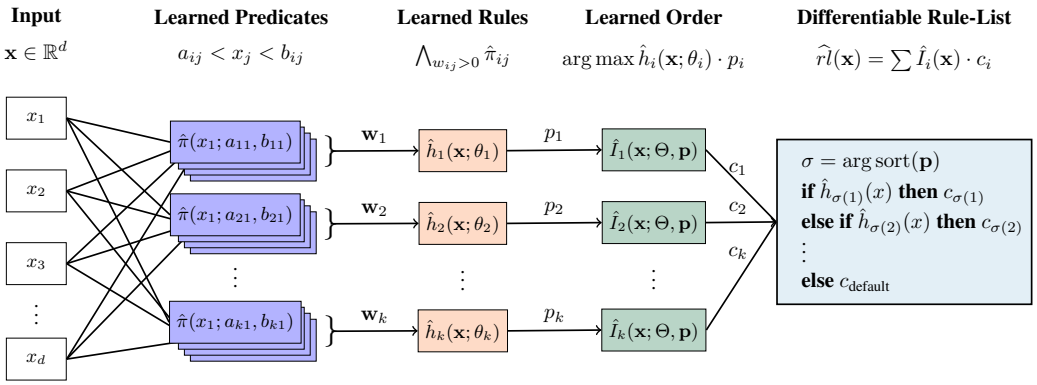


Figure 2: NEURULES architecture: We learn binary predicates $\hat{\pi}(x_j, a_{ij}, b_{ij})$ per feature j and rule i. We combine these into k rules $\hat{h}_i(\mathbf{x};\theta_i)$, turning off unnecessary predicates through sparse weights $\mathbf{w}_i$. We predict the label of a sample $\mathbf{x}$ as c_i using the rule that has highest priority p_i and is active, i.e. $\arg\max_i \hat{h}_i(\mathbf{x};\theta_i) \cdot p_i$. NEURULES is end-to-end differentiably optimizable.

Rule Lists. [Cohen, 1995] are if-then-else classifiers made out of rules. A *rule list* rl is an ordered set of k rules $\{(h_i, c_i)\}_{i=1}^k$ with parameters $\Theta = \{\theta_i\}_{i=1}^k$. To predict the label for input $\mathbf{x}$, we traverse the list from top to bot and return the consequent c_i of the first rule h_i that holds, with the final rule h_k serving as an "else" case to ensure every sample can be assigned a label. Formally,

$$rl(\mathbf{x};\Theta) = \begin{cases} c_1 & \text{if } h_1(\mathbf{x};\theta_1) = 1 \\ c_2 & \text{else if } h_2(\mathbf{x};\theta_2) = 1 \\ \vdots & \\ c_k & \text{else} \end{cases} \tag{3}$$

In the following, we will refer to binary-valued predicates π and rule heads h as strict, and to their differentiable relaxations $\hat{\pi}$ and $\hat{h}$ to the interval $[0, 1]$ as *soft*.

3 Differentiable Rule Lists

We now introduce NEURULES, short for **Neu**ral **Rule** Lists. We show its architecture in Fig. 2. The first module learns **binary predicates** $\hat{\pi}(x_j; a_{ij}, b_{ij})$ for each feature x_j and rule i. This gives NEURULES the freedom to discretize the same feature differently for each rule according to its needs, e.g. "$18 <$ `Age` < 30" for one and "$16 <$ `Age` < 24" for another.

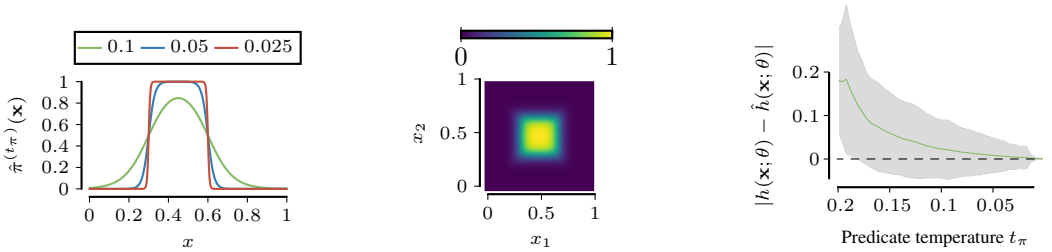
The second module combines these into **sparse rules** $\hat{h}_i(\mathbf{x};\theta_i)$. These are differentiable functions that, for all practical purposes, behave like a logical conjunction. By learning a weight vector $\mathbf{w}_i$ per rule, we turn off unnecessary predicates which functions as implicit feature selection.

The third module orders these into a **rule list** $\widehat{rl}(\mathbf{x};\Theta,\mathbf{p})$. We learn a priority p_i per rule to govern the order in the rule list. We use the Gumbel-Softmax [Jang et al., 2017] to approximate the strict indicator $I_i(\mathbf{x})$ that is 1 iff rule i is the highest priority rule that is active, i.e. $\hat{h}_i(\mathbf{x};\theta_i) \cdot p_i$ is maximal.

Next, we describe each of these modules in detail. Together, they form a differentiable relaxation that allows us to jointly optimize a rule list from predicates, rules, up to rule list order. We show that all our approximations converge towards strict logical operators at the end of training, i.e. NEURULES converges to a strict rule list, and that through gradient shaping we achieve sparse rules.

3.1 Learning Predicates

We begin with how to learn the predicates that act as the building blocks of the "**if** ..." condition of a rule. In particular, we consider thresholding functions $\pi(x; a, b) = \mathbb{1}[a < x < b]$ on individual features x_j. To avoid the need for pre-discretization, we seek to differentiably optimize per rule



(a) Soft predicate $\hat{\pi}^{(t_\pi)}(x)$ under a decreasing temperature t_π.

(b) Differentiable logical conjunction $\hat{h}(\mathbf{x};\theta)$ over two features.

(c) Softness of rule $\hat{h}(\mathbf{x};\theta)$ for different t_π (variance in gray).

Figure 3: The soft predicate approaches the true thresholding with decreasing temperature $t_\pi \to 0$ **(a)**. Multiple soft predicates are combined into a conjunctive rule **(b)**. Decreasing the temperature t_π results in increasingly strict, binary rules where $\hat{h}(\mathbf{x};\theta) \approx h(\mathbf{x};\theta)$ **(c)**.

i respectively feature j the thresholds a_{ij} and b_{ij} as part of the model. To this end, we require a differentiable equivalent of the thresholding function, as it is non-continuous at the bounds and hence unsuitable for gradient based optimization. In its place, we use an approximation that is differentiable, but yields soft predicates $\hat{\pi} : \mathbb{R} \to [0, 1]$ [Yang et al., 2018]. For a single feature x_j, we have

$$\hat{\pi}(x_j; a_{ij}, b_{ij}, t_\pi) = \frac{1}{1 + e^{\frac{1}{t_\pi}(a_{ij} - x_j)} + e^{\frac{1}{t_\pi}(x_j - b_{ij})}} \,, \tag{4}$$

where the temperature t_π controls the smoothness of the approximation. In Fig. 3a we show $\hat{\pi}$ using different temperatures t_π. As we can see, in the limit $t_\pi \to 0$ the soft predicate converges to the true thresholding function $\pi(x_j; a_{ij}, b_{ij})$ (see Appx. A.1), i.e.

$$\lim_{t_\pi \to 0} \hat{\pi}(x_j; a_{ij}, b_{ij}, t_\pi) = \pi(x_j; a_{ij}, b_{ij}) \,. \tag{5}$$

We start training with a high temperature t_π, resulting in smoother predicates $\hat{\pi}$ and avoids exploding/vanishing gradients with respect to a_{ij} and b_{ij}. To obtain a strict predicates at the end of training, we use temperature annealing to continuously decrease the temperature t_π as the training progresses. In the limit, this leaves us strict predicates where $\forall i, j : \hat{\pi}_{ij} \in \{0, 1\}$, where we write $\hat{\pi}_{ij}$ for $\hat{\pi}(x_j; a_{ij}, b_{ij}, t_\pi)$ whenever clear from context.

In sum, by using soft, differentiable predicates $\hat{\pi}$ with temperature annealing instead of static preprocessing, we are able to learn the rule and feature-specific predicates through backpropagation.

3.2 Learning Logical Conjunctions

Effectively, we need the soft $\hat{h}_i$ to behave like a logical conjunction that is differentiable, while at the same time rewarding sparsity. Formally, we require

$$\hat{h}_i : \mathcal{X} \to [0, 1], \quad \forall j \in [d] : \hat{\pi}_{ij} = 1 \Rightarrow \hat{h}_i(\mathbf{x}; \theta_i) = 1 \,, \quad \exists j \in [d] : \hat{\pi}_{ij} = 0 \Rightarrow \hat{h}_i(\mathbf{x}; \theta_i) = 0 \,. \tag{6}$$

This means the differentiable conjunction evaluates to 1 if all predicates are active, and to 0 if at least one predicate is inactive. At first glance, the weighted harmonic mean $\hat{h}_i(\mathbf{x}; \theta_i) = \frac{\sum_{j=1}^{d} w_{ij}}{\sum_{j=1}^{d} w_{ij} \hat{\pi}_{ij}^{-1}}, \quad w_{ij} \in \mathbb{R}_0^+$ introduced by [Xu et al., 2024] is a promising candidate as it fulfills the outlined criteria. If any predicate $\hat{\pi}_{ij} = 0$, then the reciprocal $\hat{\pi}_{ij}^{-1} \to \infty$ and the rule evaluates to 0. If all predicates are active, i.e. $\forall j \in [d] : \hat{\pi}_{ij} = 1$, then $\hat{h}_i(\mathbf{x}; \theta_i) = 1$. The weights $w_{ij} \in \mathbb{R}_0^+$ control predicate inclusion, where setting $w_{ij} = 0$ disables the predicate $\hat{\pi}_{ij}$. However, this formulation leads to vanishing gradients when predicates become inactive. Specifically, when any predicate $\hat{\pi}_{ik} \to 0$ with $w_{ik} > 0$, the gradients with respect to predicates and their weights approach zero, hindering learning. Thus predicates evaluating to 0 cannot be deactivated, leading to verbose rules and poor performance, as the inactive predicate zeros out the entire conjunction.

Relaxed Conjunctions. To overcome this, we propose an elegant trick that resolves vanishing gradients while simultaneously acting as a gradient shaping mechanism that promotes sparse rules. To do so, we introduce a dynamic slack η_i per rule,

$$\hat{h}_i(\mathbf{x};\theta_i) = \frac{\sum_{j=1}^{d} w_{ij}}{\sum_{j=1}^{d} w_{ij}\frac{1+\eta_i}{\hat{\pi}_{ij}+\eta_i}}, \qquad \eta_i = \frac{\epsilon}{\sum_{j=1}^{d} w_{ij}} \;, \tag{7}$$

where ϵ determines the slack magnitude. With the above, the conjunction no longer collapses to zero when a predicate is inactive but assumes a value in $\hat{h}_i(\mathbf{x};\theta_i) \in [0, C]$. Thus, a gradient gate remains open. The value C depends on ϵ and the associated weight. In the worst case, it is bounded by

$$\hat{h}_i(\mathbf{x};\theta_i) \leq \frac{\epsilon}{w_{il}+\epsilon} \;. \tag{8}$$

For example, with $\epsilon = 10^{-3}$ and $w_{il} \geq 0.099$, then $C \leq 0.01$. More generally, for a desired level of slack C, we require $w_{il} = 0$ or $w_{il} > \epsilon/C$. We provide a formal derivation in Appx. A.3.

The slack formulation also guarantees *stable* gradients. The term $w_{ij}(1+\eta_i)/(\hat{\pi}_{ij}+\eta_i)$ is always bounded and non-zero and as a result, the gradient $\partial\hat{h}_i/\partial w_{ij}$ remains well-defined. More importantly, it induces a gradient shaping effect that promotes sparse rules. If $\hat{\pi}_{ij} = 0$, the gradient is strictly negative, and the corresponding weight is reduced. If $\hat{\pi}_{ij} = 1$, the gradient is strictly positive, increasing the weight. For intermediate values of $\hat{\pi}_{ij}$ between 0 and 1, the gradient balances both effects: weakly activated predicates are discouraged while more strongly activated ones are reinforced. We give a detailed derivation in Appx. B.2.

This relaxation not only enables reliable gradient flow but also introduces a principled, self-regularizing mechanism for differentiable rule learning. By implicitly pruning unused predicates and amplifying relevant ones, it yields shorter rules than other neural approaches without the need for any explicit regularization term. We evaluate this in Section 5.3 where we find that the relaxed conjunction leads to consistent improvements in performance.

Differentiable Rules. With the predicates and the logical conjunction in place, the entire rule head

$$\lim_{t_\pi \to 0} \hat{h}_i(\mathbf{x};\theta_i) = \begin{cases} 1 & \text{if } \forall j : w_{ij} = 0 \vee \hat{\pi}_{ij} = 1 \\ 0^+ & \text{else} \end{cases} \tag{9}$$

can be differentiably learned, where 0^+ denotes a value slightly bigger than 0 due to the slack. However, this does not matter in practice since the activations of the rules only depend on $\hat{h}(\mathbf{x}) = 1$, which is exactly met. As the rule consequent, which is the "**then** ..." part of a rule, we use a logit vector $c_i \in \mathbb{R}^m$ in the classification setting with m classes. A single rule is then learned as

$$\textbf{if } \hat{h}_i(\mathbf{x};\theta_i) = 1 \textbf{ then } \underset{l \in \{1,\ldots,m\}}{\arg\max}\; c_{i,l} \;. \tag{10}$$

Next, we describe how the model learns the order of individual rules.

3.3 Learning Rule List Order

To complete NEURULES, we learn the order in which the rules should be applied, rather than assuming a fixed sequence or relying on greedy construction. We achieve this by introducing a learnable priority vector $\mathbf{p} \in \mathbb{R}^k_{>0}$, where each p_j defines the position of rule j in the list. A sample $\mathbf{x}$ is classified using the prediction c_i of the first applicable rule with the highest priority, i.e.

$$rl(\mathbf{x};\Theta,\mathbf{p}) = c_i \,, \quad \text{such that} \quad h_i(\mathbf{x};\theta_i) = 1 \wedge (\forall j \neq i : h_j(\mathbf{x}) = 0 \vee p_i > p_j) \;. \tag{11}$$

To cast this combinatorial problem into a differentiable form, we rewrite the rule list as a linear combination of the rule consequents c_i, using the product of $h_i(\mathbf{x};\theta_i) \cdot p_i$ to determine which c_i to select. Under mild conditions, this formulation is equivalent to the original rule list (see Appx. A.4).

Proposition 1 *Given binary rule heads $h_i(\mathbf{x};\theta_i) \in \{0,1\}$ and unique priorities $p_i \in \mathbb{R}_{>0}$, i.e. $\forall i \neq j : p_i \neq p_j$, the rule list $rl(\mathbf{x};\Theta,\mathbf{p})$ from Eq. 11 is equivalent to weighted sum of*

$$rl(\mathbf{x};\Theta,\mathbf{p}) = \sum_{i=1}^{k} c_i \cdot \mathbb{1}\left[i = \arg\max_j (h_j(\mathbf{x};\theta_j) \cdot p_j)\right] \;.$$

Differentiable Relaxation. While $\arg\max$ can be used with gradient routing, it produces zero gradients for all but the selected rule, hindering learning. Moreover, relying solely on the current top rule can cause the model to converge prematurely to suboptimal solutions. To address both issues, we relax the indicator function using the Gumbel-Softmax as

$$\hat{I}_i(\mathbf{x}; \Theta, \mathbf{p}, t_{rl}) = \text{softmax}_i \left(\frac{\hat{\mathbf{h}}(\mathbf{x}; \Theta) \circ \mathbf{p} + \mathbf{g}}{t_{rl}} \right), \quad \mathbf{g} \sim \text{Gumbel}(0, 1)^k . \tag{12}$$

The Gumbel-Softmax yields a differentiable approximation to the $\arg\max$ and converges to it with high probability as $t_{rl} \to 0$. We anneal t_{rl} during training to gradually sharpen rule selection. This relaxation allows gradients to flow to multiple rules early in training, improving stability and enabling exploration of different rule configurations. Even rules with low current priority continue to receive updates and can become active later. Altogether, NEURULES models a differentiable rule list as

$$\widehat{rl}(\mathbf{x}; \Theta, \mathbf{p}, t_{rl}) = \sum_{i=1}^{k} \hat{I}_i(\mathbf{x}; \Theta, \mathbf{p}, t_{rl}) \cdot \hat{c}_i , \tag{13}$$

At convergence, we convert the model into a strict rule list by fixing the learned discretization parameters a_{ij} and b_{ij}, and construct each rule $r_i(\mathbf{x}; \theta_i)$ as a conjunction over all predicates with $w_{ij} > 0$. We then order the rules by their learned priorities p_i, and compute predictions using $c_i = \text{softmax}(\hat{c}_i)$ over learned class logits, yielding the final discrete rule list $rl(\mathbf{x}; \Theta, \mathbf{p})$.

3.4 Overall Objective

We train NEURULES in a supervised learning setting, optimizing parameters based on labeled data samples $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ from the data distribution $P_{X,Y}$. We minimize a classification loss $\ell(\hat{y}, y)$, for which we choose the cross-entropy loss. Formally, we minimize the classification loss l and regularization term with respect to all rule list parameters $\Theta = (a_1, b_1, \ldots, a_k, b_k, \mathbf{w}_1, \ldots, \mathbf{w}_k)$ and $\mathbf{p}$, where t_{rl} and t_π are controlled by an annealing schedule. The overall objective is given by

$$\arg\min_{\Theta, \mathbf{p}}; \frac{1}{n} \sum_{i=1}^{n} \ell(\widehat{rl}(\mathbf{x}_i; \Theta, \mathbf{p}, t_{rl}), y_i) + \lambda \mathcal{R}(\Theta, \mathbf{p}, t_{rl}). \tag{14}$$

To avoid overfitting and encourage meaningful rules, we add a coverage-based regularizer akin to the minimum support criterion in decision trees. To this end, we define the soft coverage of each rule j as the average activation over the training set, using the soft rule selection indicator $\hat{I}_j(\mathbf{x}_i; \Theta, \mathbf{p}, t_{rl})$. Formally, we define it as $cov_j = \frac{1}{n} \sum_{i=1}^{n} \hat{I}_j(\mathbf{x}_i; \Theta, \mathbf{p}, t_{rl})$.

We then construct a regularizer $\mathcal{R}(\Theta, \mathbf{p}, t_{rl})$ out of a user-specified minimum and maximum coverage $cov_{\min}$ and $cov_{\max}$ that discourages rules that are only rarely or too frequently active as per

$$\mathcal{R}(\Theta, \mathbf{p}, t_{rl}) = \frac{1}{k} \sum_{j=1}^{k} \max\left(0, cov_{\min} - cov_j\right)^2 + \max\left(0, cov_j - cov_{\max}\right)^2 , \tag{15}$$

where we use the $\max(\cdot)$ operator to ensure that rules whose coverage is within the interval $[cov_{\min}, cov_{\max}]$ incur no penalty. The regularizer activates only when coverage moves outside this desired range, applying a quadratic penalty that encourages rule coverage to return to the targeted interval. This ensures minimal interference while effectively promoting the desired behavior.

4 Related Work

Rule lists were introduced by Cohen [1995] and often used in practice, e.g. in healthcare [Deo, 2015], criminal justice [Angelino et al., 2018], and credit risk evaluation [Bhatore et al., 2020], based on their competitive performance and inherent interpretability [Rudin, 2019]. Rule lists are closely related to decision rule learners for unordered DNF rule sets [Dash et al., 2018]. The main difference is the inclusion of an order that makes rule lists more compact and thus easier to grasp.

Combinatorial Optimization. Rule lists are typically optimized using heuristic, combinatorial search [Cohen, 1995]. Wang et al. [2017] propose Bayesian rule lists—probabilistic models whose

complexity is controlled by user-defined priors. These priors typically yield compact rule lists but may degrade performance if poorly chosen. Proenca and van Leeuwen [2020] and Papagianni and van Leeuwen [2023] propose to use MDL to address the trade-off between complexity and accuracy. Other approaches use branch-and-bound or ILP to learn optimal rule lists [Angelino et al., 2018, Aivodji et al., 2022], but these exact methods are computationally costly and limited to small datasets. In general, combinatorial methods explicitly limit rule length and rule count to prevent exponential explosion of the search space. Moreover, they rely on pre-discretizing continuous features, which is prone to remove important signal from the data.

Differentiable Optimization. Recently, neural methods have been proposed for learning rule-based classifiers. Qiao et al. [2021] introduced the first end-to-end approach, using a neural architecture with continuous relaxations of logical operators. Drawing from fuzzy logic, this allows differentiable optimization of operations like negation, disjunction, and conjunction [van Krieken et al., 2022]. After training, rules are extracted from the network. Wang et al. [2021] extend this to deeper architectures capable of learning more complex rules. For binary data, Walter et al. [2024] use a binary autoencoder to optimize both predictive and descriptive strength. Unlike NEURULES, none of these are equipped with a mechanism that removes unnecessary predicates and hence often return overly complex rules.

To learn rule lists, Dierckx et al. [2023] build on Qiao et al. [2021] by imposing a fixed rule hierarchy, but their approach requires feature pre-discretization—a limitation shared with combinatorial methods. Kusters et al. [2022] propose a differentiable rule learning method based on linear decision boundaries, but these cannot be directly interpreted as threshold-based rules. In contrast, NEURULES jointly learns threshold-based predicates per rule and feature, learns sparse rule heads that are strict logical conjunctions after training, and optimizes the order of the rules in the list, all in one go.

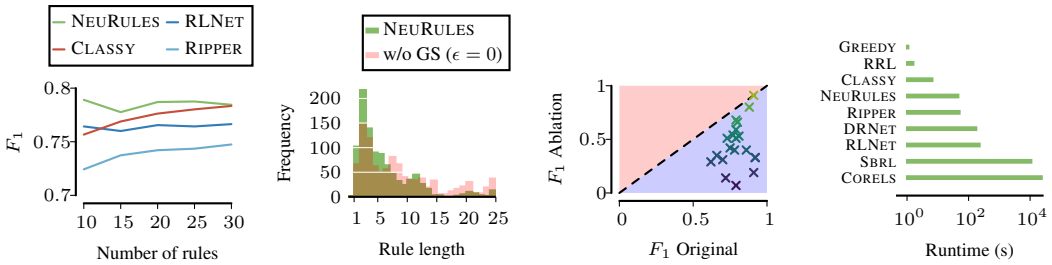
	NEURULES	RLNET	RRL	DRNET	GREEDY	CLASSY	RIPPER	CORELS	SBRL
Adult	0.80 ± 0.01	0.76 ± 0.01	0.77 ± 0.03	0.78 ± 0.01	0.75 ± 0.0	0.81 ± 0.0	0.80 ± 0.0	0.80 ± 0.0	**0.82** ± 0.0
Android Malware	0.93 ± 0.01	**0.95** ± 0.01	0.92 ± 0.03	**0.95** ± 0.0	0.86 ± 0.0	0.94 ± 0.0	0.86 ± 0.03	0.33 ± 0.01	*n/a ± n/a*
COMPAS	**0.67** ± 0.02	0.65 ± 0.02	0.59 ± 0.02	0.58 ± 0.03	0.66 ± 0.02	**0.67** ± 0.02	0.65 ± 0.01	0.65 ± 0.01	**0.67** ± 0.01
Covid ICU	0.57 ± 0.08	0.60 ± 0.05	0.63 ± 0.03	0.48 ± 0.07	0.63 ± 0.02	0.60 ± 0.06	**0.64** ± 0.02	0.63 ± 0.03	**0.64** ± 0.04
Credit Card	0.79 ± 0.01	0.77 ± 0.02	0.75 ± 0.05	0.76 ± 0.01	0.79 ± 0.01	0.79 ± 0.01	0.76 ± 0.05	*n/a ± n/a*	**0.80** ± 0.0
German Credit	**0.72** ± 0.04	0.71 ± 0.04	0.71 ± 0.03	0.15 ± 0.02	0.67 ± 0.04	0.67 ± 0.06	0.71 ± 0.04	0.54 ± 0.2	0.59 ± 0.01
Credit Screening	**0.86** ± 0.02	0.84 ± 0.02	0.82 ± 0.03	0.43 ± 0.04	**0.86** ± 0.02	0.85 ± 0.02	**0.86** ± 0.02	0.72 ± 0.04	0.75 ± 0.05
Diabetes	0.71 ± 0.05	0.70 ± 0.03	0.73 ± 0.07	0.44 ± 0.14	0.71 ± 0.04	0.70 ± 0.04	**0.74** ± 0.07	0.71 ± 0.06	0.71 ± 0.04
Drug Response	0.73 ± 0.04	0.74 ± 0.03	**0.76** ± 0.01	0.65 ± 0.03	0.67 ± 0.04	0.70 ± 0.04	0.69 ± 0.02	*n/a ± n/a*	*n/a ± n/a*
Electricity	0.75 ± 0.01	0.69 ± 0.03	0.63 ± 0.09	0.61 ± 0.01	0.75 ± 0.0	0.59 ± 0.01	0.75 ± 0.01	0.73 ± 0.0	**0.78** ± 0.01
FICO	**0.70** ± 0.01	0.67 ± 0.02	0.64 ± 0.03	0.56 ± 0.02	0.69 ± 0.01	0.67 ± 0.02	**0.70** ± 0.01	*n/a ± n/a*	**0.70** ± 0.01
Heart Disease	**0.80** ± 0.03	0.74 ± 0.02	0.72 ± 0.04	0.51 ± 0.1	0.71 ± 0.05	0.77 ± 0.09	**0.80** ± 0.05	0.65 ± 0.05	0.64 ± 0.04
Hepatitis	**0.81** ± 0.06	0.77 ± 0.07	0.78 ± 0.08	0.18 ± 0.04	0.73 ± 0.08	0.74 ± 0.07	0.75 ± 0.09	0.74 ± 0.1	0.73 ± 0.1
Juvenile	**0.89** ± 0.02	0.87 ± 0.01	0.88 ± 0.01	**0.89** ± 0.01	0.83 ± 0.04	0.88 ± 0.01	0.03 ± 0.01	*n/a ± n/a*	*n/a ± n/a*
Magic	0.81 ± 0.01	0.77 ± 0.01	0.72 ± 0.03	0.73 ± 0.05	0.75 ± 0.01	0.74 ± 0.01	0.77 ± 0.0	0.69 ± 0.0	**0.82** ± 0.0
Phishing	0.90 ± 0.03	**0.93** ± 0.0	0.83 ± 0.06	**0.93** ± 0.01	0.89 ± 0.0	0.92 ± 0.01	0.89 ± 0.0	0.27 ± 0.01	0.40 ± 0.01
Phoneme	0.80 ± 0.01	0.71 ± 0.01	0.72 ± 0.03	0.64 ± 0.02	0.76 ± 0.01	0.79 ± 0.02	0.77 ± 0.01	0.73 ± 0.01	**0.81** ± 0.01
QSAR	0.81 ± 0.02	**0.83** ± 0.02	0.80 ± 0.01	0.52 ± 0.02	0.74 ± 0.03	0.82 ± 0.03	0.79 ± 0.03	*n/a ± n/a*	0.81 ± 0.01
Ring	**0.92** ± 0.01	0.81 ± 0.01	0.83 ± 0.04	0.33 ± 0.02	0.56 ± 0.02	0.65 ± 0.03	0.74 ± 0.04	*n/a ± n/a*	0.83 ± 0.01
Titanic	0.75 ± 0.02	0.74 ± 0.03	0.69 ± 0.06	0.45 ± 0.09	**0.78** ± 0.02	0.77 ± 0.02	0.75 ± 0.03	0.66 ± 0.03	0.66 ± 0.04
Tokyo	**0.92** ± 0.02	0.91 ± 0.02	0.91 ± 0.01	0.25 ± 0.09	0.88 ± 0.01	**0.92** ± 0.02	**0.92** ± 0.03	*n/a ± n/a*	0.90 ± 0.02
TuanDromd	0.96 ± 0.01	0.98 ± 0.01	0.97 ± 0.01	**0.99** ± 0.0	0.92 ± 0.01	0.98 ± 0.0	0.92 ± 0.01	0.07 ± 0.0	*n/a ± n/a*
Avg. F1	**0.79** ± 0.02	0.77 ± 0.02	0.75 ± 0.04	0.56 ± 0.04	0.75 ± 0.02	0.76 ± 0.03	0.73 ± 0.03	0.63 ± 0.04	0.73 ± 0.02
Rank	**2.59**	4.50	5.18	7.00	5.34	4.14	4.07	6.53	4.06

Table 1: **Binary Classification**. F_1 scores for 22 real-world datasets. Timed-out experiments (>24h) are indicated by *n/a*. NEURULES consistently achieves the best or close to the best performance, obtaining the highest overall F_1 (0.79) and rank (2.59).

5 Experiments

In this section, we empirically evaluate the performance of NEURULES by comparing it against a range of rule-based and neuro-symbolic baselines, including both classical and state-of-the-art models. These include optimal rule lists (CORELS, Angelino et al. 2018), scalable Bayesian rule lists (SBRL, Yang et al. 2017), MDL-based rule lists (CLASSY, Proenca and van Leeuwen 2020), RIPPER (RIPPER, Cohen 1995), and greedily constructed rule lists (GREEDY). We also consider the state-of-the-art neural rule list model RLNET [Dierckx et al., 2023], as well as two recent neuro-symbolic rule set methods—RRL [Wang et al., 2021] and DRNET [Qiao et al., 2021].

As implementations, we use the imodels library [Singh et al., 2021] for CORELS, SBRL, and GREEDY; Weka [Hall et al., 2009] for RIPPER; and the original implementations by the authors for CLASSY, RLNET, RRL, and DRNET. We tune hyperparameters for all methods using grid search on held-out datasets (see Appx. E). All results are averaged over five-fold cross-validation. All source code and datasets used in our experiments are included in the Supplementary Material.



(a) Average F_1 score over all datasets for increasing no. of rules.

(b) NeuRules rule lengths with and without gradient shaping ($\epsilon = 0$).

(c) NeuRules F_1 scores with vs. without relaxed conjunctions.

(d) Average runtime over all benchmarked datasets. (log scale)

Figure 4: NeuRules is accurate for both short and long rule lists **(a)**. The learned rules lists **(b)**, mostly succinct and some detailed rules. The relaxed conjunction $\hat{h}(\mathbf{x})$ is always better (blue area) and improves the F_1 score on average by **0.35 (c)**. We report the avg. runtime in **(d)**.

5.1 Binary Classification

We start with a comprehensive evaluation on 22 real-world binary classification datasets that span a variety of domains. We provide characteristics and sources in Appx. C, report weighted F_1 scores (accounting for class imbalance) for a budget of 10 rules in Table 1, accuracies in Appx. Table 5, and average runtimes in Fig. 4d.

Overall. NeuRules demonstrates consistently strong performance, achieving the best overall average F_1 of **0.79** and best overall average rank of **2.59**. In terms of F_1, its closest competitors are RLNet and Classy (0.76), and in terms of rank Sbrl (4.06). Despite extensive hyperparameter tuning, Corels underperforms and does not terminate within a reasonable time for 6 datasets. In terms of runtime, NeuRules is the 4-th fastest with $46s$ on average. On the same hardware, it compares favorably to RLNet ($228s$), in spite of RLNet neither learning list order nor predicates. NeuRules outperforms the next best method on the `Ring` dataset by **0.09** F_1 points. `Ring` consists entirely of continuous features, highlighting the advantage of NeuRules in optimizing thresholds.

Number of Rules. Next, we evaluate how each method performs under a varying rule budget. We compare their results across increasing limits ($n = 10, \ldots, 30$) and show the results in Fig. 4a. We see that NeuRules consistently performs best across all budget settings, maintaining a comfortable lead over the best neural approach, RLNet. The greedy methods, Ripper and Classy, show increasing performance for larger budgets; Classy needs three times as many rules as NeuRules to match its performance in terms of F_1, showing the advantage of jointly optimizing the rule list.

Rule Length. Finally, we analyze the lengths of the rule heads NeuRules learns, i.e. the number of predicates. We give the histogram of the lengths of the rules NeuRules learned over all datasets in Fig. 4b, showing both those with and without gradient shaping. We see that the vast majority of rules consist of few predicates, i.e. they are sparse, and that gradient shaping plays an essential role in doing so. In Appx. Table 7 we see that gradient shaping also strongly improves performance, and reduces average rule length compared to other neural methods (Appx. Table 6).

5.2 Multi-Class Classification

Next, we evaluate NeuRules on 6 real-world multi-class classification datasets (see Appx. C), for which we set the dimension of the consequent $c \in \mathbb{R}^m$ to the number of classes m. We compare to RLNet and Classy as they are strong competitors that support multi-class classification, with Classy being combinatorial and RLNet continuous. We allocate each method a budget of 5 rules *per class*, i.e. $5m$ rules in total for m-class classification.

	NeuRules	RLNet	Classy
Car	0.83 ± 0.03	**0.84** ± 0.02	0.83 ± 0.03
Ecoli	**0.84** ± 0.04	0.78 ± 0.05	0.75 ± 0.04
Iris	**0.95** ± 0.04	0.81 ± 0.08	0.94 ± 0.03
Penguins	**0.99** ± 0.01	**0.99** ± 0.01	0.96 ± 0.03
Sat. image	**0.81** ± 0.01	0.68 ± 0.04	0.43 ± 0.11
Yeast	**0.55** ± 0.02	0.39 ± 0.04	0.52 ± 0.03
Avg. F1	**0.88** ± 0.02	0.82 ± 0.04	0.78 ± 0.04
Avg. Rank	**1.17**	2.33	2.50

Table 2: **Multi-class classification.** F_1 scores for six real-world datasets.

We give the results in Table 2. We find that NeuRules obtains the best average F_1 of **0.88** and best average rank of **1.17**, giving it a comfortable lead over its closest competitor, RLNet, which achieves

an average F_1 of 0.82 and average rank of 2.33. The results demonstrate that NEURULES handles multi-class problems with ease whilst delivering state-of-the-art performance.

5.3 Ablation Studies

To show the necessity and impact of the core components of NEURULES, i.e. learned predicates, relaxed conjunction, and learned rule ordering, we perform an ablation study.

Learned Discretization. We first examine the effect of learning thresholds as opposed to pre-defining them. To this end, we re-run the experiments from Section 5.1 with pre-discretized data. We provide the results on all datasets in Table 7 in the Appendix. On average, we find that the F_1 of NEURULES is degraded by 0.04 and 0.03 resp. for a uniform and k-means-based threshold selection. Furthermore, the difference is highly dataset and discretization dependent. For example, the performance on the Credit Card dataset drops by 0.05 using k-means, but by 0.08 for uniform thresholding. In general, while fixed binning can sometimes achieve reasonable results, it requires users to tune the discretization manually. On the other hand, the learned discretization by NEURULES performs at least as well or better as the fixed binning on most datasets.

Relaxed Conjunctions. Next, to test the efficacy of the relaxation of the logical conjunction, we re-run NEURULES without slack, i.e. $\epsilon = 0$ in Eq. (7). We plot the difference in F_1 score in Figure 4c. On most datasets, the relaxed conjunction outperforms the strict one by a large margin, and on average by 0.35 F_1 points. In addition, the strict conjunction performs worse on every tested dataset. Using the strict conjunction leads at best to a drop in performance, and in the worst case to numerical instability resulting in overflows. The extent of improvement stresses the importance of non-vanishing gradients and shows the necessity of relaxing the logical conjunction.

Rule Ordering. Lastly, we examine the impact of NEURULES's flexible rule ordering, by using fixed priorities p_i, i.e. they are *not* updated, which is similar to RLNET. We provide the results in Table 7 in the Appendix. We find that the F_1 of NEURULES degrades by 0.05 on average, with the largest drop on the `Credit Card` and `Adult` datasets. These datasets contain many samples ($n > 30\,000$) and likely allow to learn many rules with sufficient coverage. Based on this evidence, we postulate that learning the order of the rules is especially important for large datasets.

6 Conclusion

We propose NEURULES, a differentiable relaxation of the rule list learning problem that converges to a strict rule list through temperature annealing. NEURULES jointly learns feature discretization, the construction of conjunctive rules, and their ordering without requiring any pre-processing or manual constraints. By assigning a learnable priority score to each rule, NEURULES learns the ordering as part of the end-to-end training process. This enables it to flexibly learn both simple and complex rules, and arrange them to maximize predictive performance. As a result, NEURULES produces rule lists that are succinct and accurate, supporting trustworthy decision-making across a wide range of applications. Extensive experiments on real-world datasets show that NEURULES consistently outperforms both combinatorial and neuro-symbolic baselines.

Limitations. NEURULES requires the rule list length to be specified in advance. Learning this length differentiably is not trivial as adding or removing a rule would require a gradient signal that captures how such a change influences the ordering and the selection of predicates across the list. Additionally, learning rule lists is an inherently combinatorial problem. Addressing it through a differentiable and scalable approach necessarily introduces a set of hyperparameters, including those controlling the temperature schedule used to produce discrete rule lists. NEURULES shares standard hyperparameters common to deep learning models, such as learning rate and batch size. While these parameters must be set, our ablations in Appendix E.1 show NEURULES is robust to their variation.

Future Work. Extending NEURULES to regression tasks opens up a wide range of new applications to benefit from interpretable rule lists. In that context, we also plan to derive a non-conformity score from the rule list model for conformal prediction. Another exciting direction of future work is the adaptation of NEURULES to other data types, such as images or graphs. With appropriate predicate functions that extract meaningful concepts in those domains, rule list models could be used as more interpretable and accountable deep learning models.

References

U. Aivodji, Julien Ferry, Sebastien Gambs, Marie-Jose Huguet, Mohamed, and Siala. Leveraging integer linear programming to learn optimal fair rule lists. In *Integration of AI and OR Techniques in Constraint Programming*, 2022.

Elaine Angelino, Nicholas Larus-Stone, Daniel Alabi, Margo Seltzer, and Cynthia Rudin. Learning certifiably optimal rule lists for categorical data. *Journal of Machine Learning Research*, 18(234): 1–78, 2018.

Siddharth Bhatore, Lalit Mohan, and Y Raghu Reddy. Machine learning techniques for credit risk evaluation: a systematic literature review. *Journal of Banking and Financial Technology*, 4(1): 111–138, 2020.

William W Cohen. Fast effective rule induction. In *Machine learning proceedings 1995*, pages 115–123. Elsevier, 1995.

Sanjeeb Dash, Oktay Gunluk, and Dennis Wei. Boolean decision rules via column generation. *Advances in neural information processing systems*, 31, 2018.

Rahul C Deo. Machine learning in medicine. *Circulation*, 132(20):1920–1930, 2015.

Dua Dheeru and Karra Taniskidou Efi. Uci machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`. Accessed: 2023-10-01.

Lucile Dierckx, Rosana Veroneze, and Siegfried Nijssen. Rl-net: Interpretable rule learning with neural networks. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 95–107. Springer, 2023.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18, 2009.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*, 2017.

Remy Kusters, Yusik Kim, Marine Collery, Christian de Sainte Marie, and Shubham Gupta. Differentiable rule induction with learned relational features. In *International Workshop on Neural-Symbolic Learning and Reasoning*, 2022.

Himabindu Lakkaraju and Cynthia Rudin. Learning cost-effective and interpretable treatment regimes. In *Artificial intelligence and statistics*, pages 166–175. PMLR, 2017.

Benjamin Letham, Cynthia Rudin, Tyler H McCormick, and David Madigan. Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model. 2015.

Ioanna Papagianni and Matthijs van Leeuwen. Discovering rule lists with preferred variables. In *International Symposium on Intelligent Data Analysis*, pages 340–352. Springer, 2023.

Hugo M Proenca and Matthijs van Leeuwen. Interpretable multiclass classification by mdl-based rule lists. *Information Sciences*, 512:1372–1393, 2020.

Litao Qiao, Weijia Wang, and Bill Lin. Learning accurate and interpretable decision rule sets from neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 4303–4311, 2021.

Samuel Romano, Randal S. Olson, Jeremy Green, and Jason H. Moore. Pmlb: A large benchmark suite for machine learning evaluation and comparison, 2016. URL `https://github.com/EpistasisLab/pmlb`. Accessed: 2023-10-01.

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

Chandan Singh, Keyan Nasseri, Yan Shuo Tan, Tiffany Tang, and Bin Yu. imodels: a python package for fitting interpretable models, 2021. URL `https://doi.org/10.21105/joss.03192`.

Emile van Krieken, Erman Acar, and Frank van Harmelen. Analyzing differentiable fuzzy logic operators. *Artificial Intelligence*, 302:103602, 2022.

Nils Philipp Walter, Jonas Fischer, and Jilles Vreeken. Finding interpretable class-specific patterns through efficient neural search. In *Proceedings of the 38th Annual AAAI Conference on Artificial Intelligence*. AAAI, 2024.

Tong Wang, Cynthia Rudin, Finale Doshi-Velez, Yimin Liu, Erica Klampfl, and Perry MacNeille. A bayesian framework for learning rule sets for interpretable classification. *Journal of Machine Learning Research*, 18(70):1–37, 2017.

William Yang Wang, Kathryn Mazaitis, and William W Cohen. A soft version of predicate invention based on structured sparsity. In *IJCAI*, pages 3918–3924, 2015.

Zhuo Wang, Wei Zhang, Ning Liu, and Jianyong Wang. Scalable rule-based representation learning for interpretable classification. *Advances in Neural Information Processing Systems*, 34:30479–30491, 2021.

Sascha Xu, Nils Philipp Walter, Janis Kalofolias, and Jilles Vreeken. Learning exceptional subgroups by end-to-end maximizing kl-divergence. In *Forty-first International Conference on Machine Learning*, 2024.

Hongyu Yang, Cynthia Rudin, and Margo Seltzer. Scalable bayesian rule lists. In *International conference on machine learning*, pages 3921–3930. PMLR, 2017.

Yongxin Yang, Irene Garcia Morillo, and Timothy M Hospedales. Deep neural decision trees. *arXiv preprint arXiv:1806.06988*, 2018.

A Convergence of Continuous Relaxations

We show that our continuous relaxations for predicate, logical conjunction, and rule list converge to their discrete counterparts.

A.1 Predicate

We restate the proof of Xu et al. [2024]. The soft predicate for a single feature x_i is defined as

$$\hat{\pi}(x_j; a_{ij}, b_{ij}, t_\pi) = \frac{1}{1 + e^{\frac{1}{t_\pi}(a_{ij} - x_j)} + e^{\frac{1}{t_\pi}(x_j - b_{ij})}} , \tag{16}$$

We now show that the soft predicate converges to the hard predicate as $t_\pi \to 0$, which is defined as

$$\pi(x_i; a, b) = \begin{cases} 1 & \text{if } x_i \in [a_{ij}, b_{ij}] \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

Proof: *Consider the following four cases:*

1. $x_i < a_{ij}$*: Then for the lower bound it holds that*

$$\lim_{t_\pi \to 0} e^{\frac{1}{t_\pi}(a_{ij} - x_i)} = e^{\infty} = \infty , \tag{18}$$

while for the upper bound it, which must be $b_{ij} > a_{ij} > x_i$, it holds that

$$\lim_{t_\pi \to 0} e^{\frac{1}{t_\pi}(x_i - b_{ij})} = e^{-\infty} = 0 . \tag{19}$$

Hence, in the limit of $t_\pi \to 0$, we have

$$\lim_{t_\pi \to 0} \frac{1}{1 + e^{\frac{1}{t_\pi}(a_{ij} - x_j)} + e^{\frac{1}{t_\pi}(x_j - b_{ij})}} = 0 .$$

2. $x_i > b_{ij}$*: Then for the lower bound it holds that*

$$\lim_{t_\pi \to 0} e^{\frac{1}{t_\pi}(a_{ij} - x_i)} = e^{-\infty} = 0 \,, \tag{20}$$

while for the upper bound it, which must be $b_{ij} < x_i$*, it holds that*

$$\lim_{t_\pi \to 0} e^{\frac{1}{t_\pi}(x_i - b_{ij})} = e^{\infty} = \infty \,. \tag{21}$$

Hence, in the limit of $t_\pi \to 0$*, we have*

$$\lim_{t_\pi \to 0} \frac{1}{1 + e^{\frac{1}{t_\pi}(a_{ij} - x_j)} + e^{\frac{1}{t_\pi}(x_j - b_{ij})}} = 0 \,.$$

3. $a_{ij} < x_i < b_{ij}$*: Then for the lower bound it holds that*

$$\lim_{t_\pi \to 0} e^{\frac{1}{t_\pi}(a_{ij} - x_i)} = e^{-\infty} = 0 \,, \tag{22}$$

while for the upper bound it, which must be $b_{ij} > x_i$*, it holds that*

$$\lim_{t_\pi \to 0} e^{\frac{1}{t_\pi}(x_i - b_{ij})} = e^{-\infty} = 0 \,. \tag{23}$$

Hence, in the limit of $t_\pi \to 0$*, we have*

$$\lim_{t_\pi \to 0} \frac{1}{1 + e^{\frac{1}{t_\pi}(a_{ij} - x_j)} + e^{\frac{1}{t_\pi}(x_j - b_{ij})}} = 1 \,.$$

4. $x_i = a_{ij}$ *or* $x_i = b_{ij}$*: If* $x_i = a_{ij}$*, then*

$$\lim_{t_\pi \to 0} e^{\frac{1}{t_\pi}(a_{ij} - x_i)} = e^{0} = 1 \quad \textit{and} \quad \lim_{t_\pi \to 0} e^{\frac{1}{t_\pi}(x_i - b_{ij})} = e^{-\infty} = 0 \,, \tag{24}$$

whereas if $x_i = b_{ij}$*, then*

$$\lim_{t_\pi \to 0} e^{\frac{1}{t_\pi}(a_{ij} - x_i)} = e^{-\infty} = 0 \quad \textit{and} \quad \lim_{t_\pi \to 0} e^{\frac{1}{t_\pi}(x_i - b_{ij})} = e^{0} = 1 \,. \tag{25}$$

In both cases, we have

$$\lim_{t_\pi \to 0} \frac{1}{1 + e^{\frac{1}{t_\pi}(a_{ij} - x_j)} + e^{\frac{1}{t_\pi}(x_j - b_{ij})}} = \frac{1}{1+1} = \frac{1}{2} \,. \tag{26}$$

To obtain the desired behavior at the boundaries, i.e. $\hat{\pi}(x_i) = 1$ *or* $\hat{\pi}(x_i) = 0$*, the output must thus be either ceiled or floored, which corresponds to choosing strict or non-strict inequalities (*$a_{ij} < x_i < b_{ij}$ *or* $a_{ij} \leq x_i \leq b_{ij}$*).*

□

A.2 Logical Conjunction

The soft logical conjunction for a set of predicates $(\hat{\pi}_{i1}, \ldots, \hat{\pi}_{id})$, where $\hat{\pi}_{ij} = \hat{\pi}(x_j; a_{ij}, b_{ij})$, is defined as

$$\hat{h}_i(\mathbf{x}; \theta_i) = \frac{\sum_{j=1}^{d} w_{ij}}{\sum_{j=1}^{d} w_{ij} \hat{\pi}_{ij}^{-1}} \,. \tag{27}$$

Given a set of non-negative weights $\mathbf{w}_i \in [0, \infty)^d$, with at least one weight w_{ij} being positive, we first show that the soft logical conjunction takes values in $[0, 1]$ given d predicates $\hat{\pi}_{ij} \in [0, 1]$.

The domain of the reciprocal is $\hat{\pi}_{ij}^{-1} \in [1, \infty)$. This implies for the weighted sum of reciprocals that $\sum_{j=1}^{d} w_{ij} \hat{\pi}_{ij}^{-1} \geq \sum_{j=1}^{d} w_{ij} > 0$, since there exists a weight $w_{ij} > 0$, Then the soft logical conjunction is bounded by

$$0 \leq \frac{\sum_{j=1}^{d} w_{ij}}{\sum_{j=1}^{d} w_{ij} \hat{\pi}_{ij}^{-1}} = \hat{h}_i(\mathbf{x}; \theta_i) \leq 1 \,. \tag{28}$$

In particular, if and only if all active predicates are 1, i.e. $\forall j, w_{ij} > 0 : \hat{\pi}_{ij} = 1$, the rule $\hat{h}_i(\mathbf{x}; \theta_i) = 1$. In that case, $\hat{\pi}_{ij}^{-1} = 1$ such that $\sum_{j=1}^{d} w_{ij} \hat{\pi}_{ij}^{-1} = \sum_{j=1}^{d} w_{ij}$, hence showing that $\hat{h}_i(\mathbf{x}; \theta_i) = 1$.

On the other hand, if there exists an index j where $w_{ij} > 0$, then $\hat{\pi}_{ij}^{-1} = \infty \leftrightarrow \hat{\pi}_{ij} = 0$ and the rule $\hat{h}_i(\mathbf{x}; \theta_i) = 0$.

Let us now consider the limit $t_\pi \to 0$, for which in Appendix A.1 we have shown that $\hat{\pi}_{ij} \in \{0, 1\}$ for all j, i.e. the predicates are binary. Above, we have shown that

1. $\forall j \in [d] : \hat{\pi}_{ij} = 1 \vee w_{ij} = 0 \implies \hat{h}_i(\mathbf{x}; \theta_i) = 1$
2. $\exists j \in [d] : w_{ij} > 0 \wedge \hat{\pi}_{ij} = 0 \implies \hat{h}_i(\mathbf{x}; \theta_i) = 0$

Then, $\hat{h}_i(\mathbf{x}; \theta_i) \in \{0, 1\}$, as either all predicates are 1 or at least one predicate is 0, and $\hat{h}_i$ mimics the logical conjunction of the predicates, i.e. $\hat{h}_i(\mathbf{x}; \theta_i) = \bigwedge_{j=1, w_{ij}>0}^{d} \pi_{ij}$.

A.3 Relaxed Conjunction

The relaxed conjunction $\hat{h}_i(\mathbf{x}; \theta_i)$ is defined as

$$\eta_i = \frac{\epsilon}{\sum_{j=1}^{d} w_{ij}} , \quad \hat{h}_i(\mathbf{x}; \theta_i) = \frac{\sum_{j=1}^{d} w_{ij}}{\sum_{j=1}^{d} w_{ij} \frac{1+\eta_i}{\hat{\pi}_{ij}+\eta_i}} . \tag{29}$$

We first show that for $\hat{\pi}_{il} = 0$ the resulting soft conjunction is upper bounded by $\frac{\epsilon}{\epsilon + w_{il}}$.

Proof: *Let $\hat{\pi}_{il} = 0$. Then the relaxed conjunction is*

$$\hat{h}_i(\mathbf{x}; \theta_i) = \frac{\sum_{j=1}^{d} w_{ij}}{\sum_{j \neq l} w_{ij} \frac{1+\eta_i}{\hat{\pi}_{ij}+\eta_i} + w_{il} \frac{1+\eta_i}{\hat{\pi}_{il}+\eta_i}} \tag{30}$$

$$\hat{h}_i(\mathbf{x}; \theta_i) = \frac{\sum_{j=1}^{d} w_{ij}}{\sum_{j \neq l} w_{ij} \frac{1+\eta_i}{\hat{\pi}_{ij}+\eta_i} + w_{il} \frac{1+\eta_i}{\eta_i}} \tag{31}$$

$$\tag{32}$$

To upper bound $\hat{h}_i(\mathbf{x})$, we lower bound the denominator. Consider the minimum value of the denominator, which is obtained when all other predicates are active, i.e. $\hat{\pi}_{ij} = 1$ for all $j \neq l$, where

$$\sum_{j \neq l} w_{ij} \frac{1+\eta_i}{\hat{\pi}_{ij}+\eta_i} + w_{il} \frac{1+\eta_i}{\eta_i} \geq \sum_{j \neq l} w_{ij} \frac{1+\eta_i}{1+\eta_i} + w_{il} \frac{1+\eta_i}{\eta_i} = \sum_{j \neq l} w_{ij} + w_{il} \frac{1+\eta_i}{\eta_i} . \tag{33}$$

Using this lower bound, we can upper bound the relaxed conjunction as

$$\hat{h}_i(\mathbf{x}; \theta_i) \leq \frac{\sum_{j=1}^{d} w_{ij}}{\sum_{j \neq l} w_{ij} + w_{il} \frac{1+\eta_i}{\eta_i}} \tag{34}$$

$$= \frac{\eta_i \sum_{j=1}^{d} w_{ij}}{\eta_i \sum_{j \neq l} w_{ij} + w_{il}(1+\eta_i)} \tag{35}$$

$$= \frac{\eta_i \sum_{j=1}^{d} w_{ij}}{\eta_i \sum_{i=1}^{d} w_{ij} + w_{il}} \tag{36}$$

$$= \frac{\frac{\epsilon}{\sum_{j=1}^{d} w_{ij}} \sum_{j=1}^{d} w_{ij}}{\left(\frac{\epsilon}{\sum_{j=1}^{d} w_{ij}} \sum_{i=1}^{d} w_{ij} \right) + w_{il}} \tag{37}$$

$$= \frac{\epsilon}{\epsilon + w_{il}} . \tag{38}$$

□

A.4 Proof of Proposition 1

Proposition 1 *Given binary rule heads $h_i(\mathbf{x};\theta_i) \in \{0,1\}$ and unique priorities $p_i \in \mathbb{R}_{>0}$, i.e. $\forall i \neq j : p_i \neq p_j$, the rule list $rl(\mathbf{x};\Theta,\mathbf{p})$ from Eq. 11 is equivalent to weighted sum of*

$$rl(\mathbf{x};\Theta,\mathbf{p}) = \sum_{i=1}^{k} c_i \cdot \mathbb{1}\left[i = \arg\max_j (h_j(\mathbf{x};\theta_j)\cdot p_j)\right] \ .$$

Proof: *The rule list $rl(\mathbf{x};\Theta,\mathbf{p})$ uses the rule active rule $h_i(\mathbf{x};\theta_i)=1$ with the highest priority p_i to determine the output, i.e.*

$$rl(\mathbf{x};\Theta,\mathbf{p}) = c_i \tag{39}$$

$$\text{s.t. } h_i(\mathbf{x};\theta_i) = 1 \ \wedge \ \forall j \neq i : h_j(\mathbf{x};\theta_j) = 0 \vee p_i > p_j \ . \tag{40}$$

Let i be the index of $\arg\max_j(h_j(\mathbf{x};\theta_j)\cdot p_j)$, and assume that there exist a further active rule l with $p_l > p_i$ and $h_l(\mathbf{x};\theta_l)=1$. Then $h_l(\mathbf{x};\theta_l)\cdot p_l > h_i(\mathbf{x};\theta_i)\cdot p_i$, which contradicts that i is the index of $\arg\max_j(h_j(\mathbf{x};\theta_j)\cdot p_j)$. □

B Gradient Shaping of $\hat{h}$

We analyze the proposed relaxed conjunction from Eq. 7 in terms of its gradient with respect to the predicate $\hat{\pi}_{il}$ and the weight w_{il}.

B.1 Derivative with respect to $\hat{\pi}_{ij}$

To compute its derivatives, we will use the quotient rule for differentiation, i.e. $\frac{d}{dx}\frac{f(x)}{g(x)} = \frac{f'(x)g(x)-f(x)g'(x)}{g(x)^2}$, where

$$f(\mathbf{x},\mathbf{w}) = \sum_{j=1}^{d} w_{ij} \ , \quad \frac{\partial f}{\partial w_{il}} = 1 \ , \quad \frac{\partial f}{\partial \hat{\pi}_{il}} = 0 \tag{41}$$

$$g(\mathbf{x},\mathbf{w}) = \sum_{j=1}^{d} w_{ij}\frac{1+\eta_i}{\hat{\pi}_{ij}+\eta_i} \ , \quad \frac{\partial g}{\partial w_{il}} = \frac{1+\eta_i}{\hat{\pi}_{il}+\eta_i} \ , \quad \frac{\partial g}{\partial \hat{\pi}_{il}} = -\frac{w_{il}(1+\eta_i)}{(\hat{\pi}_{il}+\eta_i)^2} \tag{42}$$

Then, the partial derivative of the relaxed conjunction with respect to the predicate $\hat{\pi}_{il}$ is

$$\frac{\partial \hat{h}_i}{\partial \hat{\pi}_{il}} = \frac{0(\sum_{j=1}^{d} w_{ij}\frac{1+\eta_i}{\hat{\pi}_{ij}+\eta_i}) + (\sum_{j=1}^{d} w_{ij})\frac{w_{il}(1+\eta_i)}{(\hat{\pi}_{il}+\eta_i)^2}}{\left(\sum_{j=1}^{d} w_{ij}\frac{1+\eta_i}{\hat{\pi}_{ij}+\eta_i}\right)^2} \tag{43}$$

$$= \frac{(\sum_{j=1}^{d} w_{ij})w_{il}(1+\eta_i)}{(\hat{\pi}_{il}+\eta_i)^2\left(\sum_{j=1}^{d} w_{ij}\frac{1+\eta_i}{\hat{\pi}_{ij}+\eta_i}\right)^2}. \tag{44}$$

Consider now the case where there exists a predicate k included rule i that is not active for sample $\mathbf{x}$, i.e. $w_{ik} > 0$ and $\hat{\pi}_{ik} = 0$. Then the derivative for any predicate $\hat{\pi}_{il}, l \in [d]$ is

$$\lim_{\hat{\pi}_{ik}\to 0} \frac{\partial \hat{h}_i}{\partial \hat{\pi}_{il}} = \frac{(\sum_{j=1}^{d} w_{ij})w_{il}(1+\eta_i)}{(\hat{\pi}_{il}+\eta_i)^2\left(w_{ik}\frac{1+\eta_i}{\eta_i} + \sum_{j\neq k} w_{ij}\frac{1+\eta_i}{\hat{\pi}_{ij}+\eta_i}\right)^2} \ . \tag{45}$$

The term $w_{ik}\frac{1+\eta_i}{\eta_i}$ is finite and positive so that the gradient does not vanish if a predicate is off. In the worst case that all predicates are off, i.e. $\hat{\pi}_{ij} = 0$ for all j, the derivative is

$$\frac{(\sum_{j=1}^{d} w_{ij})w_{il}(1+\eta_i)}{\eta_i^2\left(\sum_{j=1}^{d} w_{ij}\frac{1+\eta_i}{\eta_i}\right)^2} \tag{46}$$

We note that $1 + \eta_i \approx 1$ for small η_i, so that approximately the gradient is

$$\lim_{\forall j \in [d]: \hat{\pi}_{ij} \to 0} \frac{\partial \hat{h}_i}{\partial \hat{\pi}_{il}} \approx \frac{(\sum_{j=1}^{d} w_{ij}) w_{il}}{\eta_i^2 \left(\sum_{j=1}^{d} w_{il} \frac{1}{\eta_i}\right)^2} = \frac{w_{ij}}{\sum_{j=1}^{d} w_{ij}} \,. \tag{47}$$

That is, the gradient is only zero if the predicate is not active, i.e. $w_{il} = 0$, and otherwise finite and positive, scaled by the weight of the predicate.

B.2 Derivative with respect to w_{il}.

To compute the derivative with respect to the weight w_{il}, we additionally need to consider the dependence between η_i and w_{il}. Hence, we rewrite the expressions as

$$\hat{h}_i(\mathbf{x}) = \frac{f}{g}, \qquad f \;=\; \sum_{j=1}^{d} w_{ij}, \qquad g \;=\; \sum_{j=1}^{d} w_{ij} \, \frac{1+\eta_i}{\hat{\pi}_{ij} + \eta_i}, \qquad \eta_i = \frac{\varepsilon}{f}.$$

Step 1: the η_i-term introduced by the chain rule. Since η_i depends on every w_{ij},

$$\frac{\partial \eta_i}{\partial w_{il}} = -\frac{\varepsilon}{f^2} = -\frac{\eta_i}{f}.$$

For the inner fraction set

$$A_j := \frac{1+\eta_i}{\hat{\pi}_{ij} + \eta_i} \,, \quad \frac{\partial A_j}{\partial \eta_i} = \frac{1 \cdot (\hat{\pi}_{ij} + \eta_i) - (1+\eta_i) \cdot 1}{(\hat{\pi}_{ij} + \eta_i)^2} = \frac{\hat{\pi}_{ij} - 1}{(\hat{\pi}_{ij} + \eta_i)^2} \,.$$

Hence, the derivative of A_j with respect to w_{il} is

$$\frac{\partial A_j}{\partial w_{il}} = \frac{\partial A_j}{\partial \eta_i} \cdot \frac{\partial \eta_i}{\partial w_{il}} = -\frac{\eta_i}{f} \cdot \frac{1 - \hat{\pi}_{ij}}{(\hat{\pi}_{ij} + \eta_i)^2} \,.$$

Step 2 : $\partial g / \partial w_{il}$. We re-write g as $\sum_{j=1}^{d} w_{ij} A_j$. Then we have that $\frac{\partial g}{\partial w_{il}} = \sum_{j=1}^{d} \frac{\partial}{\partial w_{il}} (w_{ij} A_j)$. For each individual $w_{ij} A_j$, we use the product rule. For $j = l$ we obtain

$$\frac{\partial}{\partial w_{il}} (w_{il} A_l) = A_l + w_{il} \, \frac{\partial A_l}{\partial w_{il}}$$

while for $j \neq l$ the derivative is

$$\frac{\partial}{\partial w_{il}} (w_{ij} A_j) = w_{ij} \, \frac{\partial A_j}{\partial w_{il}}$$

Hence, summing up overall j gives

$$\frac{\partial g}{\partial w_{il}} = A_l + \sum_{j=1}^{d} w_{ij} \, \frac{\partial A_j}{\partial \eta_i} \, \frac{\partial \eta_i}{\partial w_{il}} = \frac{1+\eta_i}{\hat{\pi}_{il} + \eta_i} \;+\; \frac{\eta_i}{f} \left(\sum_{j=1}^{d} w_{ij} \, \frac{1 - \hat{\pi}_{ij}}{(\hat{\pi}_{ij} + \eta_i)^2} \right) \,.$$

Step 3: quotient rule for $\partial \hat{h}_i / \partial w_{il}$. Because $\partial f / \partial w_{il} = 1$,

$$\frac{\partial \hat{h}_i}{\partial w_{il}} = \frac{g - f \, A_l \;-\; \eta_i \left(\sum_{j=1}^{d} w_{ij} \, \frac{1 - \hat{\pi}_{ij}}{(\hat{\pi}_{ij} + \eta_i)^2} \right)}{g^{\,2}}.$$

Step 4: sign of the gradient. Write

$$g = \sum_j w_{ij} A_j = \sum_j w_{ij} (A_j - A_l) + \sum_j w_{ij} \, A_l = \sum_j w_{ij} (A_j - A_l) + f \, A_l$$

so the numerator splits naturally into

$$\underbrace{\sum_{j=1}^{d} w_{ij} (A_j - A_l)}_{T_1} \;+\; \underbrace{-\eta_i \left(\sum_{j=1}^{d} w_{ij} \, \frac{1 - \hat{\pi}_{ij}}{(\hat{\pi}_{ij} + \eta_i)^2} \right)}_{T_2 \; (< 0)} \,.$$

We make a case analysis (CA) on whether the predicate fires or not.

Case A — $\hat{\pi}_{il} = 0$ (predicate l is inactive). For $A_l = \dfrac{1+\eta_i}{\eta_i}$, it holds that $\forall j \in [d] : A_j \leq A_l$,

$$T_1 = \sum_j w_{ij}(A_j - A_l) \leq 0, \qquad T_2 < 0,$$

hence $N = T_1 + T_2 < 0$ for every inactive predicate:

$$\boxed{\hat{\pi}_{il} = 0 \implies \frac{\partial \hat{h}_i}{\partial w_{il}} < 0}$$

Case B — $\hat{\pi}_{il} = 1$ (predicate l is active). We re-write the partial derivative as

$$\frac{\partial \hat{h}_i}{\partial w_{il}} = \sum_{j=1}^{d} w_{ij} \left(A_j - A_l - \eta_i \frac{1-\hat{\pi}_{ij}}{(\hat{\pi}_{ij}+\eta_i)^2} \right) \tag{48}$$

$$= \sum_{j=1}^{d} w_{ij} \left(\frac{1+\eta_i}{\hat{\pi}_{ij}+\eta_i} - \frac{1+\eta_i}{\hat{\pi}_{il}+\eta_i} - \frac{\eta_i - \eta_i\hat{\pi}_{ij}}{(\hat{\pi}_{ij}+\eta_i)^2} \right) \tag{49}$$

$$= \sum_{j=1}^{d} w_{ij} \left(\frac{(1+\eta_i)(\hat{\pi}_{ij}+\eta_i)}{(\hat{\pi}_{ij}+\eta_i)^2} - \frac{1+\eta_i}{1+\eta_i} - \frac{\eta_i - \eta_i\hat{\pi}_{ij}}{(\hat{\pi}_{ij}+\eta_i)^2} \right) \tag{50}$$

$$= \sum_{j=1}^{d} w_{ij} \left(\frac{(\hat{\pi}_{ij} + \eta_i + \hat{\pi}_{ij}\eta_i + \eta_i^2) - (\eta_i - \eta_i\hat{\pi}_{ij})}{(\hat{\pi}_{ij}+\eta_i)^2} - 1 \right) \tag{51}$$

$$= \sum_{j=1}^{d} w_{ij} \left(\frac{\hat{\pi}_{ij} + 2\hat{\pi}_{ij}\eta_i + \eta_i^2}{\hat{\pi}_{ij}^2 + 2\hat{\pi}_{ij}\eta_i + \eta_i^2} - 1 \right) \tag{52}$$

$$\geq 0 \tag{53}$$

Since $\hat{\pi}_{ij} \in [0,1]$, $\hat{\pi}_{ij} \geq \hat{\pi}_{ij}^2$, such that each term $\frac{\hat{\pi}_{ij}+2\hat{\pi}_{ij}\eta_i+\eta_i^2}{\hat{\pi}_{ij}^2+2\hat{\pi}_{ij}\eta_i+\eta_i^2} - 1 \geq 0$. In particular, $\frac{\partial \hat{h}_i}{\partial w_{il}} = 0$ if and only if $\forall j \in [d] : w_{ij} = 0 \vee \hat{\pi}_{ij} = 1 \vee \hat{\pi}_{ij} = 0$, i.e. the predicates are strictly binary.

$$\boxed{\hat{\pi}_{il} = 1 \implies \frac{\partial \hat{h}_i}{\partial w_{il}} \geq 0}$$

Case C — $0 < \hat{\pi}_{il} < 1$ (predicate l is active).

1. How large can $A_j - A_l$ be? Because the mapping $z \mapsto A(z) = \dfrac{1+\eta_i}{z+\eta_i}$ is strictly decreasing on $z \in [0,1]$,

$$\operatorname{sgn}\big(A_j - A_l\big) = \operatorname{sgn}\big(\hat{\pi}_{ij} - \hat{\pi}_{il}\big).$$

Hence it is *positive* if $\hat{\pi}_{ij} < \hat{\pi}_{il}$, *zero* if $\hat{\pi}_{ij} = \hat{\pi}_{il}$, and *negative* if $\hat{\pi}_{ij} > \hat{\pi}_{il}$.

2. Splitting the index set Let

$$\mathcal{L} = \{j \mid \hat{\pi}_{ij} < \hat{\pi}_{il}\},\ \mathcal{G} = \{j \mid \hat{\pi}_{ij} > \hat{\pi}_{il}\}, \qquad f = \sum_j w_{ij}.$$

Then

$$\begin{aligned} T_1 &= \sum_{j\in\mathcal{L}} w_{ij}\big(A_j - A_l\big) + \sum_{j\in\mathcal{G}} w_{ij}\big(A_j - A_l\big) \\ &= (1+\eta_i) \sum_{j\in\mathcal{L}} w_{ij} \frac{\hat{\pi}_{il} - \hat{\pi}_{ij}}{(\hat{\pi}_{ij}+\eta_i)(\hat{\pi}_{il}+\eta_i)} - (1+\eta_i) \sum_{j\in\mathcal{G}} w_{ij} \frac{\hat{\pi}_{ij} - \hat{\pi}_{il}}{(\hat{\pi}_{ij}+\eta_i)(\hat{\pi}_{il}+\eta_i)}. \end{aligned}$$

3. Bounding the *negative* term $|T_2|$ Because $0 \le \hat{\pi}_{ij} \le 1$, $0 \le \frac{1-\hat{\pi}_{ij}}{(\hat{\pi}_{ij}+\eta_i)^2} \le 1/\eta_i^2$, so

$$0 \;\le\; |T_2| \;\le\; \frac{f}{\eta_i}.$$

4. Condition for $N > 0$ or $N < 0$

$$N \;=\; \underbrace{(1+\eta_i)\sum_{j\in\mathcal{L}} w_{ij}\,\frac{\hat{\pi}_{il}-\hat{\pi}_{ij}}{(\hat{\pi}_{ij}+\eta_i)(\hat{\pi}_{il}+\eta_i)}}_{\text{positive "support"}} \;-\; \underbrace{(1+\eta_i)\sum_{j\in\mathcal{G}} w_{ij}\,\frac{\hat{\pi}_{ij}-\hat{\pi}_{il}}{(\hat{\pi}_{ij}+\eta_i)(\hat{\pi}_{il}+\eta_i)}}_{\text{negative "drag"}} \;-\; |T_2|.$$

$$\boxed{N > 0 \Longleftrightarrow \text{support} \;>\; \text{drag} \;+\; |T_2|}$$

i.e. the positive help from less-active literals must exceed both the drag from more-active ones *and* the extra penalty $|T_2|$.

5. What the sign of N means in practice.

- $N > 0$ **(weight is increased).** This occurs exactly when

  $$\text{support} \;>\; \text{drag}+|T_2| \quad\Longleftrightarrow\quad \textstyle\sum_{j\in\mathcal{L}} w_{ij}(A_j - A_l) > \sum_{j\in\mathcal{G}} w_{ij}(A_l - A_j) + \eta_i \sum_j w_{ij}\,\frac{1-\hat{\pi}_{ij}}{(\hat{\pi}_{ij}+\eta_i)^2}.$$

  1. Small η_i ($\varepsilon \ll 1$) makes the penalty $|T_2|$ tiny, so the inequality is easier to satisfy.
  2. A literal that *often* fires (large set $\mathcal{L}$) quickly accumulates support and its weight is pushed up.
  3. If no other literal is *more* active ($\mathcal{G} = \varnothing$) the drag term vanishes and the weight always grows.
- $N < 0$ **(weight is decreased).** Happens when either
  1. one or more *more–active* predicates already carry substantial weight (large drag), *or*
  2. the literal fires only rarely, so the support term is too small to beat drag+$|T_2|$.

  In that case, the optimization gradually suppresses the weight, allowing "competing" literals to dominate the conjunction.

Interpretation (gradient shaping).

1. **Selective amplification.** For a predicate that *fires* ($\hat{\pi}_{il} > 0$), the gradient is positive *iff* the *support* from less-active literals outweighs both the *drag* from more-active ones *and* the small $|T_2|$-penalty:

   $$\text{support} \;>\; \text{drag} + |T_2|.$$

   This is easiest to satisfy when η_i is tiny or when the literal fires frequently, so those literals grow and become the dominant parts of the rule.
2. **Automatic pruning.** Predicates that stay *inactive* ($\hat{\pi}_{il} = 0$) always get a negative gradient ($N < 0$); their weights shrink on every update, driving irrelevant features to zero.
3. **Self-balancing dynamics.** Because both support and drag depend on the *current* weights, a literal can switch from negative to positive feedback (or vice versa) during training, letting the model continually re-allocate capacity toward the most informative predicates.

The optimisation therefore performs an *implicit feature-selection*: irrelevant literals are pruned while relevant ones are reinforced, producing sparse, interpretable rules without any explicit ℓ_1 penalty.

C Dataset Statistics

We retrieve the datasets from the UCI repository [Dheeru and Efi, 2017], the imodels-data repository [Singh et al., 2021], and the pmlb repository [Romano et al., 2016]. We give the base statistics for the 22 real-world binary classification datasets in Table 3 and those for the 6 real-world multi-class classification datasets in Table 4.

	Samples				Features			
Dataset	Total	Positive	Negative	Ratio	Total	Discrete	Continuous	Ratio
Adult	32561	7841	24720	0.24	14	5	9	0.64
Android Malware	29332	14700	14632	0.50	86	86	0	0.00
COMPAS	6172	2990	3182	0.48	20	16	4	0.20
Covid ICU	1494	809	685	0.54	16	15	1	0.06
Credit Card	30000	6636	23364	0.22	33	15	18	0.55
Credit Screening	690	307	383	0.44	15	8	7	0.47
Diabetes	768	268	500	0.35	8	0	8	1.00
Drug Response	597	148	449	0.25	196	104	92	0.47
Electricity	45312	19237	26075	0.42	8	1	7	0.88
FICO	10459	5459	5000	0.52	23	2	21	0.91
German Credit	1000	700	300	0.70	60	57	3	0.05
Heart Disease	270	120	150	0.44	15	10	5	0.33
Hepatitis	155	123	32	0.79	19	13	6	0.32
Juvenile	3640	487	3153	0.13	286	284	2	0.01
Magic	19020	6688	12332	0.35	10	0	10	1.00
Phishing	11055	6157	4898	0.56	30	30	0	0.00
Phoneme	5404	1586	3818	0.29	5	0	5	1.00
QSAR	1055	356	699	0.34	41	12	29	0.71
Ring	7400	3736	3664	0.50	20	0	20	1.00
Titanic	2099	681	1418	0.32	8	5	3	0.38
Tokyo	959	613	346	0.64	44	7	37	0.84
TuanDromd	4464	3565	899	0.80	241	241	0	0.00

Table 3: Dataset statistics for the 22 real-world datasets used in our experiments. We say a feature is numerical if it has more than 10 unique values.

			Features			
Dataset	Samples	Classes	Total	Discrete	Continuous	% Ratio
Car	1 728	4	6	6	0	0.00
Ecoli	327	5	7	2	5	0.71
Iris	150	3	4	0	4	1.00
Penguins	333	3	7	3	4	0.57
Sat. Image	6 435	6	36	0	36	1.00
Yeast	1 479	9	8	2	6	0.75

Table 4: Dataset statistics for the 6 real-world datasets used in our experiments. We say a feature is numerical if it has more than 10 unique values.

D Hardware

For the hyperparameter tuning and the comparison, we used machines equipped with 2x AMD Epyc 7773x 2.2GHz (base), 3.5GHz (max Boost) with 128 real cores and 2TB of memory. For RRL, we used an A100-40GB, since the code of the authors can only be run on GPU. For the ablation studies, we additionally used machines with Intel(R) Xeon(R) Gold 6244 CPU @ 3.60GHz with 32 real cores and 256GB of memory.

E Hyperparameters

We optimize the hyperparameters of each method using grid search, for each choosing that configuration that achieves the best average F_1 score on the validation datasets (`eeg_eye_state`, `horse_colic`, `ozone-level`, `pc1`, and `breast_cancer`). We extend the default setting from the respective papers for the hyperparameters we test. We allow a maximum runtime of 24 hours per run. For the combinatorial methods, we use equal height binning with 5 cutpoints. For the neural methods we use their inbuild discretization according to the paper.

For SBRL, we performed a grid search for the following hyperparameters: listlengthprior $\in [2, 3, 4]$; for listwidthprior $\in [1, 2, 3]$; for maxcardinality $\in [1, 2, 3]$ and for minsupport, we fixed the value at 0.05. The number of monte-carlo sampling chains is set to 5 or 10. The best configuration is listlengthprior $= 10$, listwidthprior $= 2$, maxcardinality $= 3$, minsupport $= 0.05$, and n_chains $= 10$.

For DRNET, we tested the following hyperparameters: lr $\in [0.001, 0.01, 0.1]$; and_lam $\in [0.0001, 0.001, 0.01]$; epochs $\in [1000, 2000, 3000]$; and or_lam $\in [0.0001, 0.001, 0.01]$. We obtained lr $= 0.01$, and_lam $= 0.0001$, epochs $= 3000$, and or_lam $= 0.0001$.

As GREEDY has only one hyperparameter, we optimized max_depth $\in [3, 5, 7, 10]$. The optimal setting is max_depth $= 5$.

For CLASSY, we tested the following hyperparameters: beam_width $\in [50, 100, 150, 200]$; n_cutpoints $\in [3, 5, 10]$; and max_depth $\in [3, 5, 10]$. The best parameters are beam_width $= 100$, n_cutpoints $= 5$, and max_depth $= 5$.

For RIPPER, we set -N to $n/(2k)$ and varied the -F parameter in $[1, 2, 3]$ and -O in $[1, 2, 3]$.

For CORELS, we performed a grid search with the following hyperparameters: c $\in [0.005, 0.01, 0.02]$; n_iter $\in [5000, 10000, 15000]$; max_card $\in [2, 3, 4, 5]$; and min_support $\in [0.01, 0.02, 0.05]$. Grid search selected c $= 0.01$, n_iter $= 10\,000$, max_card $= 5$, and min_support $= 0.01$.

For NEURULES, we performed a grid search with the following hyperparameters: n_epochs $\in [250, 500]$; min_support $\in [0.1, 0.2, 0.3]$; max_support $\in [0.8, 0.9]$; lambd $\in [0.4, 0.5, 0.6]$; bs $\in [25610242048]$ and lr $\in [0.002, 0.025, 0.05]$. We use n_epochs $= 500$, min_support $= 0.2$, max_support $= 0.9$, lambd $= 0.4$, bs $= 2048$, and lr $= 0.01$.

For RLNET, we conducted a grid search with the following hyperparameters: lr $\in [0.010.020.025]$; lambda_and $\in [0.0001, 0.001, 0.01]$; n_epochs $\in [1000, 2000, 3000]$; and l2_lambda $\in [0, 0.001, 0.01]$. The optimal configuration is lr $= 0.01$, lambda_and $= 0.001$, n_epochs $= 3000$, and l2_lambda $= 0.001$.

E.1 Hyperparameter Sensitivity

We re-run NEURULES on all tested datasets and vary one parameter at a time. We provide the results in Figure 5. The only noticeable impact on performance is observed when increasing the learning rate to 0.5, which adversely affects the performance of NEURULES. All other parameters, using our default values otherwise, do not significantly affect the performance of NEURULES on the tested datasets.

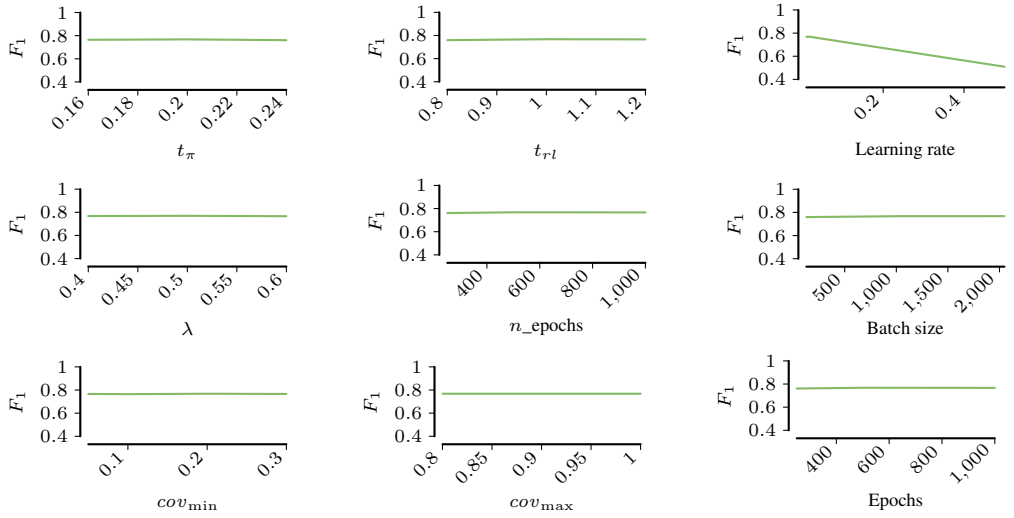


Figure 5: Average F_1 score over all datasets for different hyperparameters.

E.2 Temperature Schedules

Temperature schedules are crucial in optimization problems involving soft approximations of discrete functions. They help in gradually transitioning from a soft to a hard decision boundary, which can improve convergence and performance. By adjusting the temperature parameter, we control the

smoothness of the approximations, allowing the model to explore the solution space more effectively during the initial stages of training and then refine the solutions as training progresses.

We use a linear temperature decay during the second half of training for both temperatures. The temperature starts at 1.0 and linearly decreases to 0.1 for the rule priority temperature t_{rl} and ranges from 0.2 to 0.05 for the predicate temperature t_{π}. These values were determined through hyperparameter optimization and are unchanged across all experiments. The temperature is updated at each epoch as follows:

```
temp_start = 1.0
temp_end = 0.1
temp = temp_start
step_size = (temp_start - temp_end)/(total_epochs*2)
If epoch >= total_epochs/2:
    temp = temp - step_size
```

This schedule allows the model to maintain a high level of flexibility during the first half of training with multiple active rules and gradually focus on only a single rule per sample in the latter half.

We plot the impact of the relaxation with regard to different temperatures t_{rl} in Figure 6. We start training with a positive temperature $t_{rl} = 1.0$, where the rule with highest active priority has on average 0.75 of the weight, whilst the other rules contribute the remaining 0.25. We continuously decrease the temperature t_{rl} towards zero, so that in the end the indicator $\hat{I}(\mathbf{x}; \Theta, \mathbf{p}, t_{rl})$ of the actual rule dominates with a weight of 0.99. Using our appropriate annealing schedule NEURULES starts training using a relaxed rule list, which it can optimize, and continuously moves towards a strict rule list.

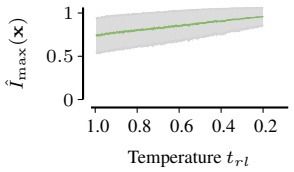
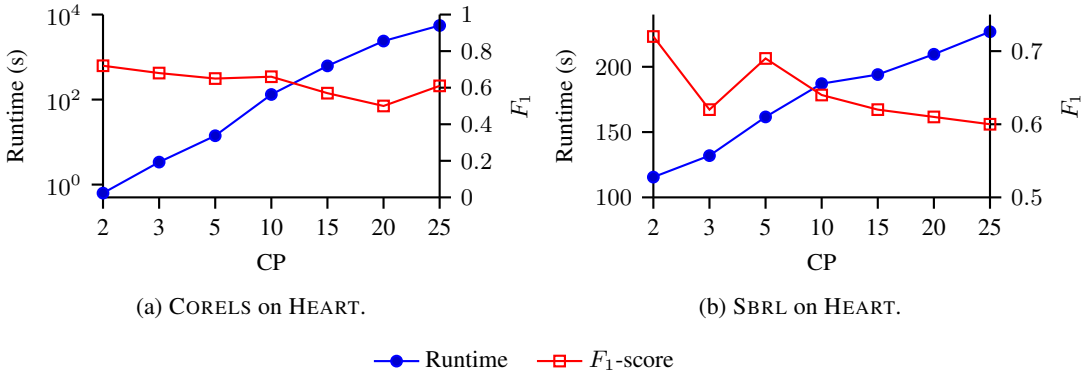


Figure 6: Weight of highest priority rule for decreasing t_{rl} (variance in grey) for a training run on the Heart Disease dataset.

F Real World: Accuracy

We report the accuracy of the methods on the real-world datasets in Table 5. The conclusions about the performance of the methods are consistent with the results obtained using the weighted F1 score. Nonetheless, we report the weighted F1 score as the main evaluation metric, as it is more informative about the performance of the methods in the presence of class imbalance.

G Real World: Runtime



(a) CORELS on HEART.

(b) SBRL on HEART.

Figure 7: Runtime (blue, left axis) and F_1 score (red, right axis) versus number of cut-points (CP) on the Heart dataset for CORELS (a) and SBRL (b).

Lastly, we examine the scalability of NEURULES in contrast to other rule lists. We provide the average runtime of each method across all benchmarks in Figure 4d. NEURULES on average takes 75s per

	NEURULES	RLNET	RRL	DRNET	GREEDY	CLASSY	RIPPER	CORELS	SBRL
Adult	0.81 ± 0.01	0.81 ± 0.0	0.77 ± 0.04	0.82 ± 0.0	0.80 ± 0.0	0.82 ± 0.0	0.82 ± 0.0	0.82 ± 0.0	**0.83** ± 0.0
Android Malware	0.93 ± 0.01	**0.95** ± 0.01	0.92 ± 0.03	**0.95** ± 0.0	0.87 ± 0.0	0.94 ± 0.0	0.87 ± 0.03	0.50 ± 0.0	*n/a* ± *n/a*
COMPAS	**0.68** ± 0.01	0.66 ± 0.01	0.60 ± 0.02	0.62 ± 0.02	0.66 ± 0.02	**0.68** ± 0.02	0.65 ± 0.01	0.65 ± 0.01	0.67 ± 0.01
Covid ICU	0.60 ± 0.05	0.61 ± 0.05	0.63 ± 0.03	0.49 ± 0.07	0.63 ± 0.02	0.61 ± 0.04	**0.64** ± 0.02	0.63 ± 0.03	**0.64** ± 0.04
Credit Card	0.81 ± 0.0	0.81 ± 0.01	0.75 ± 0.07	0.80 ± 0.01	**0.82** ± 0.0	0.81 ± 0.01	0.78 ± 0.01	*n/a* ± *n/a*	0.81 ± 0.01
German Credit	**0.73** ± 0.04	0.72 ± 0.04	0.72 ± 0.03	0.30 ± 0.02	0.72 ± 0.04	0.71 ± 0.04	0.72 ± 0.05	0.62 ± 0.16	0.70 ± 0.03
Credit Screening	**0.86** ± 0.02	0.84 ± 0.03	0.82 ± 0.03	0.50 ± 0.05	**0.86** ± 0.02	0.85 ± 0.02	**0.86** ± 0.02	0.73 ± 0.04	0.75 ± 0.05
Diabetes	0.74 ± 0.03	0.73 ± 0.03	**0.75** ± 0.05	0.45 ± 0.12	0.72 ± 0.03	0.73 ± 0.04	**0.75** ± 0.06	0.74 ± 0.05	0.71 ± 0.03
Drug Response	**0.77** ± 0.03	**0.77** ± 0.03	0.76 ± 0.01	0.75 ± 0.02	0.73 ± 0.04	0.76 ± 0.02	0.72 ± 0.03	*n/a* ± *n/a*	*n/a* ± *n/a*
Electricity	0.76 ± 0.01	0.71 ± 0.01	0.65 ± 0.08	0.67 ± 0.01	0.76 ± 0.0	0.66 ± 0.0	0.76 ± 0.01	0.73 ± 0.0	**0.78** ± 0.0
FICO	**0.71** ± 0.01	0.68 ± 0.01	0.65 ± 0.02	0.59 ± 0.02	0.70 ± 0.01	0.68 ± 0.02	0.70 ± 0.01	*n/a* ± *n/a*	0.70 ± 0.01
Heart Disease	**0.80** ± 0.03	0.75 ± 0.01	0.72 ± 0.04	0.52 ± 0.11	0.71 ± 0.05	0.78 ± 0.09	**0.80** ± 0.05	0.66 ± 0.05	0.66 ± 0.03
Hepatitis	**0.81** ± 0.05	0.79 ± 0.06	0.79 ± 0.07	0.24 ± 0.05	0.80 ± 0.05	0.78 ± 0.03	0.78 ± 0.07	0.77 ± 0.07	0.74 ± 0.1
Juvenile	**0.90** ± 0.02	0.89 ± 0.01	0.88 ± 0.01	0.89 ± 0.01	0.86 ± 0.01	0.89 ± 0.01	0.13 ± 0.01	*n/a* ± *n/a*	*n/a* ± *n/a*
Magic	0.82 ± 0.01	0.79 ± 0.01	0.74 ± 0.02	0.77 ± 0.03	0.74 ± 0.01	0.77 ± 0.0	0.78 ± 0.01	0.74 ± 0.0	**0.83** ± 0.0
Phishing	0.90 ± 0.03	**0.93** ± 0.01	0.83 ± 0.06	**0.93** ± 0.01	0.89 ± 0.0	0.92 ± 0.01	0.89 ± 0.0	0.44 ± 0.01	0.56 ± 0.01
Phoneme	0.79 ± 0.01	0.74 ± 0.01	0.74 ± 0.02	0.73 ± 0.01	0.76 ± 0.01	**0.81** ± 0.01	0.77 ± 0.02	0.76 ± 0.01	**0.81** ± 0.01
QSAR	0.82 ± 0.02	**0.84** ± 0.01	0.80 ± 0.02	0.64 ± 0.02	0.74 ± 0.03	0.82 ± 0.03	0.79 ± 0.03	*n/a* ± *n/a*	0.82 ± 0.01
Ring	**0.92** ± 0.01	0.81 ± 0.01	0.83 ± 0.04	0.50 ± 0.02	0.61 ± 0.02	0.68 ± 0.02	0.75 ± 0.04	*n/a* ± *n/a*	0.83 ± 0.01
Titanic	0.78 ± 0.02	0.77 ± 0.02	0.72 ± 0.05	0.45 ± 0.08	**0.79** ± 0.02	**0.79** ± 0.02	0.78 ± 0.02	0.71 ± 0.03	0.71 ± 0.03
Tokyo	**0.92** ± 0.02	0.91 ± 0.02	0.91 ± 0.01	0.37 ± 0.05	0.88 ± 0.01	**0.92** ± 0.02	**0.92** ± 0.03	*n/a* ± *n/a*	0.91 ± 0.02
TuanDromd	0.96 ± 0.01	0.98 ± 0.01	0.97 ± 0.01	**0.99** ± 0.0	0.93 ± 0.01	0.98 ± 0.0	0.93 ± 0.01	0.20 ± 0.01	*n/a* ± *n/a*
Avg. Acc	**0.8** ± 0.02	0.79 ± 0.02	0.76 ± 0.03	0.62 ± 0.03	0.76 ± 0.02	0.78 ± 0.02	0.75 ± 0.02	0.68 ± 0.03	0.75 ± 0.02
Rank	**2.59**	4.16	5.77	6.73	5.02	3.70	4.59	6.63	4.28

Table 5: We report the results on 22 real-world datasets stemming from domains such as medicine,finance, and criminal justice. We compare NEURULES against CORELS, SBRL, CLASSY, GREEDY, RLNET, RRL, DRNET, and XGBOOST. We report the accuracy averaged over 5-fold cross validation. The experiments were terminated after 24 hours, indicated by *n/a*. NEURULES performs the best with respect to the Acc score, indicated by the lowest rank.

dataset. This is faster than DRNET, RLNET, and SBRL, but significantly slower than the greedy approaches GREEDY, CLASSY and the neural RRL, which all take below 10s per dataset. In general, NEURULES incurs a computational overhead compared to the greedy methods but compensates for it in terms of classification accuracy. RRL optimizes only a rule set instead of a rule list and avoids the more costly rule list optimization, which explains its faster runtime.

H Rule Complexity

The sparsity of the discovered rules is interesting due to its correlation with the interpretability of a rule. We provide summary statistics for all methods in Table 6. NEURULES learns more succinct rules than other neural approaches (RLNET, DRNET), showing the efficacy of our gradient shaping in that regard. Compared to combinatorial methods that impose a limit on maximum cardinality, the rules we find are longer. We do not think this is necessarily a weakness; there may exist datasets where rules with longer clauses make sense. In general, we observe a power law-like trend of rule lengths learned by NEURULES, where the majority of rules have four or fewer clauses.

I Ablation Studies

We perform an ablation study to investigate the impact of the different components of our method. We provide the F_1 score when using uniform, k-means pre-processing of continuous features, a fixed rule priority $\mathbf{p}$ and with a strict conjunction respectively in Table 7. We re-run NEURULES on the same datasets as in the main experiments, using the same hyperparameters as in the main experiments, but with the respective components removed. We report the 5-fold cross-validated F1 score for each dataset in Table 7.

Method	Median	Mean	Std.
NEURULES	4.00	7.85	13.45
RLNET	6.00	9.65	15.16
DRNET	21.00	53.96	97.02
CORELS	2.00	1.83	0.86
CLASSY	3.00	2.89	0.90
RIPPER	2.00	1.94	1.37
SBRL	2.00	1.91	0.71

Table 6: Rule length statistics (number of predicates per rule) across datasets.

				Thresholding		
Dataset	Original	Strict Conjunction	Fixed Order	Uniform	k-means	Quantile
Adult	0.80	0.66	0.66	0.71	0.79	0.79
Android Malware	0.93	0.33	0.92	0.94	0.33	0.33
COMPAS	0.67	0.35	0.66	0.50	0.62	0.62
Covid ICU	0.57	0.29	0.63	0.56	0.58	0.58
Credit Card	0.79	0.68	0.68	0.71	0.68	0.68
German Credit	0.72	0.14	0.70	0.72	0.58	0.58
Credit Screening	0.86	0.76	0.86	0.86	0.68	0.68
Diabetes	0.71	0.51	0.69	0.54	0.63	0.63
Electricity	0.75	0.42	0.75	0.63	0.70	0.70
FICO	0.70	0.31	0.69	0.58	0.70	0.70
Heart Disease	0.80	0.40	0.80	0.78	0.66	0.66
Hepatitis	0.81	0.07	0.78	0.81	0.70	0.70
Juvenile	0.89	0.80	0.80	0.88	0.80	0.80
Magic	0.81	0.51	0.77	0.75	0.76	0.76
Phishing	0.90	0.90	0.91	0.90	0.40	0.40
Phoneme	0.80	0.59	0.59	0.76	0.65	0.65
QSAR	0.81	0.53	0.80	0.77	0.79	0.79
Ring	0.92	0.33	0.56	0.75	0.63	0.63
Titanic	0.75	0.59	0.77	0.77	0.54	0.54
Tokyo	0.92	0.19	0.92	0.92	0.88	0.88
Avg. Diff.	0.00	-0.33	-0.05	-0.05	-0.14	-0.14

Table 7: Ablation study comparing the obtained F_1 scores using a strict conjunction, fixed rule priority, uniform, kmeans and quantile based binning. NEURULES accuracy is negatively impacted by each's components removal.

Table 8: F_1 scores across CORELS regularization λ; NEURULES column shows our method. Best per row in **bold**.

		CORELS							
Dataset	NEURULES	λ=0	0.001	0.005	0.01	0.02	0.03	0.04	0.1
Android Malware	**0.93**	0.33	0.33	0.33	0.33	0.33	0.33	0.33	0.33
FICO	**0.70**	0.69	0.69	0.69	0.69	0.69	0.69	0.69	0.68
Heart Disease	**0.80**	0.68	0.68	0.68	0.68	0.68	0.68	0.68	0.07
Hepatitis	**0.81**	0.76	0.76	0.76	0.76	0.76	0.76	0.48	0.63
Rin	**0.92**	0.63	0.63	0.63	0.63	0.63	0.63	0.63	0.54
Titanic	**0.75**	0.68	0.66	0.66	0.66	0.62	0.54	0.54	0.54
Avg. F_1	**0.82**	0.63	0.62	0.62	0.62	0.62	0.61	0.56	0.47

Sensitivity of CORELS. Across a wide sweep of regularization values $\lambda \in \{0, 0.001, 0.005, 0.01, 0.02, 0.03, 0.04, 0.1\}$, CORELS shows largely stable performance: per–dataset F_1 varies little for $\lambda \leq 0.04$ (e.g., FICO stays at 0.69 across nearly all settings; Hepatitis remains 0.76; Android Malware stays 0.33). Only very strong regularization ($\lambda = 0.1$) causes notable degradation on some datasets (e.g., Heart Disease drops to 0.07). Despite this stability, CORELS consistently underperforms our method: NEURULES achieves higher F_1 on every dataset in the table (e.g., 0.93 vs. 0.33 on Android Malware, 0.70 vs. 0.69 on FICO, 0.81 vs. 0.76 on Hepatitis, 0.92 vs. 0.63 on Rin, 0.75 vs. ≤ 0.68 on Titanic). On average, NEURULES attains 0.82 F_1, whereas CORELS ranges from 0.47 ($\lambda = 0.1$) to 0.63 ($\lambda = 0$), never matching NEURULES.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: As stated, we introduce the first differentiable rule list learning algorithm that jointly learns the discretization of features, the rules, and their order.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss limitations in the conclusion section and state all assumptions in the main text.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: We prove all statements in the Appendix and provide all assumptions in the main text. To the best of our knowledge, the proofs are complete and correct.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We provide the source code with which the experimental results were obtained in the Supplemental Material. We use a fixed random seed and clearly describe how our architecture is implemented.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We provide the source code in the Supplemental Material. We provide references to the external datasets and their sources.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We describe our evaluation procedure in the main text and hyperparameter search in the Appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We use 5-fold cross validation in all our experiments and provide standard deviation in the main tables.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We describe the compute resources used in the Appendix and provide runtimes in the Experiment section.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have reviewed the NeurIPS Code of Ethics and our research conforms to it.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: We propose a new architecture for rule list learning. The model class itself is well established, so that there is no new societal impact to discuss that extends beyond the scope of the existing model class.

    Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper poses no such risk.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cite all respective authors for method code and datasets used in our experiments.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [No]

    Justification: We do not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.