Rule-Based Machine Learning and Abductive Reasoning for Explainable AI

(説明可能 AI のためのルールベース機械学習とアブダクティブ推論)

Yoichi Sasaki

August 2025

Graduate School of Information Science and Technology

Hokkaido University

Abstract

Recently, explainable artificial intelligence (XAI) has attracted research interest because, in practical applications, machine learning models are expected not only to predict accurately but also to be interpretable so that humans can understand the reasoning behind model predictions, especially when applied to important decision-making tasks such as automated credit approvals and medical diagnosis.

Traditionally, studies on AI started from symbolic AI, which is also called logic-based AI. Abduction, or abductive reasoning, has been known as a form of reasoning that seeks the best explanation for given observations. Recently, clearly in the context of XAI, abduction has been gaining attention because the knowledge base used for explanations is interpretable by users and the outputs are based on strict logical reasoning. On the other hand, its weaknesses have been recognized, including computational intractability, the knowledge acquisition bottleneck, and a lack of flexibility.

Concurrently with the above pure logical reasoning approaches, data-driven AI, also known as (statistical) machine learning, has been widely studied. This approach has the advantage of automatically learning input-output mappings from a given dataset. In the context of XAI, there are two main streams: (i) using simple interpretable models, and (ii) generating post-hoc explanations for black-box models.

The first approach aims to improve the predictive performance of simple interpretable

models such as (generalized) linear models, decision trees, or rule lists. These models and their explanations for predictions are inherently interpretable, although their prediction accuracy is often lower than that of black-box models. The second approach aims to provide local post-hoc explanations for each prediction made by any black-box model. One drawback of these approaches is the high cost of generating explanations for each input, making it impractical for humans to verify every prediction in real-time.

Although most XAI studies for generating explanations from both data-driven AI and symbolic AI have been conducted separately in different contexts, in order for users to truly be convinced, it is necessary to connect both novel insights derived from data or black-box models and logical interpretations based on users' existing knowledge.

The aim of this thesis is to develop a framework that can generate explanations connecting the knowledge possessed by users with novel insights inherent in the data or black-box models, thereby enabling users to understand and verify the explanations more effectively. To achieve this, we connect symbolic AI and data-driven AI by employing rules as a common representation framework, and construct an explainable rule-based AI that enables users to be convinced by the results. This approach allows us to obtain rule-based explanations both inductively and deductively from data, and further enables us to verify whether the rules do not contain logical contradictions and to explore whether there are alternative or better hypotheses when combined with the users' existing knowledge. Furthermore, we formulate these problems as an appropriate mathematical optimization problem, which allows us to utilize recently advancing discrete optimization solvers and achieve the computational efficiency needed for practical applications.

In this thesis, to achieve our research goal, we make three main technical contributions to rule-based XAI. These contributions are presented from both the data-driven AI perspective (Chapters 4 and 5) and the symbolic AI perspective (Chapter 6).

In Chapter 4, we introduce a new machine learning model, called the Alternative Rule Model (ARM), which selects a rule from a ruleset that provides a prediction close to that of any black-box model based on the input instance. This model outputs the prediction from the selected rule, allowing humans to verify the correctness of the rule rather than the black-box prediction itself. We also propose an algorithm to find a small, high-coverage ruleset for the ARM. The ruleset should have enough coverage to explain many inputs while remaining small enough for human verification. We formalize this problem as a weighted MaxSAT problem. Our approach enables efficient identification of a small ruleset that can support black-box predictions and be checked by humans. Unlike purely rule-based models like decision trees, the ARM has the potential for better predictive performance by selecting rules that are close to the black-box model's predictions.

In Chapter 5, we present a pure rule-based model called the top-k rule list, an ordered list of rules similar to traditional rule lists, but differing in that it selects not just the first rule but the first k rules whose conditions are satisfied by the input, and then makes a prediction by taking an ensemble of these k rules. The top-k rule list can be viewed as a generalization of the traditional rule list, which is a special case with $k = 1$, and this generalization enables the user to obtain a preferable trade-off between accuracy and interpretability by selecting different values of k. Learning the traditional rule list is known as a computationally hard combinatorial optimization problem, and learning the top-k rule list can be even harder because multiple rules must be considered for each prediction. To enable efficient learning, we propose an Integer Linear Programming (ILP) formulation for learning the top-k rule list, and effectively reduce the number of variables and constraints by introducing an assumption that the rules are ordered by certainty score. The experimental results show that the top-k rule list learned with the ILP formulation for $k > 1$ outperformed the traditional rule list in accuracy for most datasets.

In Chapter 6, we consider weighted abduction, which is one of the commonly used mathematical models for abduction. In general, the abduction problem is NP-hard, and existing approaches such as ILP and Answer Set Programming (ASP) struggle to scale when dealing with large amounts of background knowledge and observations. To address this challenge, we propose an efficient formulation of the depth-limited weighted abduction problem as a weighted MaxSAT problem. Our key insight is that the majority of the resulting hard clauses are Horn, resulting in a formulation that closely resembles the Max Horn SAT problem. This structure enables us to leverage recent advances in MaxSAT solvers. Empirical evaluations on large-scale, real-world instances demonstrate that our approach is up to 100 times faster than existing ILP-based methods, while maintaining equivalent solution quality. These results suggest that SAT-based formulations offer a scalable and effective alternative to ILP for solving weighted abduction problems.

Finally, in Chapter 7, we conclude this thesis and discuss future work. In this thesis, we introduce rule-based explanation methods using optimization solvers, through the framework of logical reasoning based on background knowledge and the framework of data-driven rule-based models. Through this thesis, we contribute to the first step of developing a framework that can generate rule-based explanations linking users' existing knowledge with novel insights derived from the data or black-box models, thereby enabling users to better understand and trust the results. For future work, it is of interest to apply our method to real-world problems and to identify the domains in which it is most suitable.

Acknowledgement

First, I would like to express my deep gratitude to Professor Hiroki Arimura, who has supervised me from the very beginning of my research activities. I also wish to thank Professor Masaharu Yoshioka and Professor Takashi Horiyama for their patient guidance and invaluable comments. I am grateful to everyone in the Information Knowledge Network Laboratory. I would like to express my appreciation to all my current and former colleagues at NEC Corporation, especially Mr. Yuzuru Okajima, Dr. Kazeto Yamamoto, Mr. Takashi Onishi, Mr. Kunihiko Sadamasa, Mr. Kenji Soejima, and Dr. Satoshi Morinaga. I am also grateful to Dr. Takanori Maehara and Dr. Takumi Akazaki. I could not have accomplished this work without their understanding and support. Finally, I would like to voice my heartfelt thanks to my family. My wife, Shiori, has been extremely supportive of me throughout my research activities and has always cared about my health. My son, Ryusuke, has continually provided the motivation to complete my degree.

Contents

1 Introduction **11**

1.1 Background 11

1.2 Research Goal 13

1.3 Contributions 13

2 Preliminaries **17**

2.1 Basic definition 17

2.2 Mathematical optimization 17

2.2.1 Maximum satisfiability problems 18

2.2.2 Integer Linear Programming 19

3 Overview of Rule-Based XAI **21**

3.1 Background 21

3.2 Rule-based machine learning 22

3.2.1 Rule-based model 22

3.2.2 Rule-based post-hoc explanation 23

3.3 Abductive reasoning 24

4 Ruleset Discovery to Support Black-box Model Predictions **27**

4.1 Introduction 28
4.2 Alternative Rule Model 31
4.3 Ruleset Optimization 33
4.3.1 Problem definition 33
4.3.2 Weighted Max Horn SAT formulation 34
4.4 Experiments 36
4.4.1 Experimental Setup 36
4.4.2 Results 40
4.5 Discussion 43
4.5.1 Interpretability of obtained rules 43
4.5.2 Structured Data vs Unstructured Data 44
4.5.3 Comparison with CORELS and defragTrees 45
4.5.4 Limitation 46
4.5.5 Use Cases 46
4.5.6 Future Work 47
4.6 Related Work 48
4.7 Conclusion 49

5 Top-k Rule List Learning **57**
5.1 Introduction 58
5.2 Model Description 60
5.2.1 Traditional Rule List (Top-1 Rule List) 60
5.2.2 Top-k Rule List 61
5.2.3 Calculating Predictions of Rules 62
5.3 Learning Top-k Rule Lists 63
5.4 Experiments 67

5.4.1 Results 71
5.5 Related Work 74
5.6 Conclusion and Future Work 75

6 Solving Weighted Abduction via Max-SAT **77**
6.1 Introduction 78
6.2 Preliminaries 80
6.2.1 Weighted Abduction 80
6.3 Weighted Abduction via Max-SAT 81
6.3.1 Backchaining Graph 82
6.3.2 Max-SAT Formulation 83
6.4 Experiments 87
6.4.1 Datasets 87
6.4.2 Evaluation 88
6.4.3 Results 89
6.5 Conclusion 90

7 Conclusion and Future Work **93**

Chapter 1

Introduction

1.1 Background

Recently, explainable artificial intelligence (XAI) has attracted research interest. This is because it is important to provide not only highly accurate predictions but also sufficient interpretability so that humans can understand the reasons behind them. This requirement becomes especially significant in critical applications such as automated loan approvals [62] or medical decision making [71].

Traditionally, studies on AI started from symbolic AI, which is also called logic-based AI [57]. Its key techniques include logical reasoning systems [64], automatic programming [26], knowledge representation [11,56], and question answering systems [27]. Abduction, or abductive reasoning, has been known as a form of reasoning that seeks the best explanation for given observations [33, 64]. Recently, clearly in the context of XAI, abduction has been gaining attention because the knowledge base used for explanations is interpretable by users and the outputs are based on strict logical reasoning [9, 37]. On the other hand, its weaknesses have been recognized, including computational intractability, the knowledge

acquisition bottleneck, and a lack of flexibility.

Concurrently with the above pure logical reasoning approaches, data-driven AI, also known as (statistical) machine learning, has been widely studied. This approach has the advantage of automatically learning input-output mappings from a given dataset. In the context of XAI, there are two main streams: (i) using simple interpretable models, and (ii) generating post-hoc explanations for black-box models.

The first approach aims to improve the predictive performance of simple interpretable models such as (generalized) linear models [31,54], decision trees [13,31,67], and rule lists, also known as decision lists, [4, 48, 59, 83, 86]. In these interpretable models, rule-based models including decision trees and rule lists are suitable in terms of explainability because the prediction processes of the models are considered as applying If-Then rules, which are intuitive for humans. These rule-based models and their explanations for predictions are inherently interpretable, although their prediction accuracy is often lower than that of black-box models.

The second approach aims to provide local post-hoc explanations for each prediction made by any black-box model [44,50,68,69]. One drawback of these approaches is the high cost of generating explanations for each input, making it impractical for humans to verify every prediction in real-time. To overcome these difficulty, a few works aims to incorporate interpretable models as part of the decision making of black-box models by using the interpretable part as an explanation [2, 60]. The rule constrained network (RCN) [60] chooses one rule from a limited rule set and makes its decision accordingly, whereas an earlier study created a separate graphical model for each decision [2].

Although most XAI studies for generating explanations from both data-driven AI and symbolic AI have been conducted separately in different contexts, in order to truly be convinced for users, it is necessary to connect both novel insights derived from data or black-box

models and logical interpretations based on users' existing knowledge.

1.2 Research Goal

The aim of this thesis is to develop a framework that can generate explanations connecting the knowledge possessed by users with novel insights inherent in the data or black-box models, thereby enabling users to understand and verify the explanations more effectively. To achieve this, we connect symbolic AI and data-driven AI by *employing rules as a common representation framework*, and construct an explainable rule-based AI that enables users to be convinced by the results. This approach allows us to obtain rule-based explanations both inductively and deductively from data, and further enables us to verify whether the rules do not contain logical contradictions and to explore whether there are alternative or better hypotheses when combined with users' existing knowledge. Furthermore, we formulate these problems as appropriate mathematical optimization problems, which allows us to utilize recently advancing discrete optimization solvers and achieve the computational efficiency needed for practical applications.

1.3 Contributions

In this thesis, to achieve our research goal, we make three main technical contributions to rule-based XAI. These contributions are presented from both the data-driven AI perspective (Chapters 4 and 5) and the symbolic AI perspective (Chapter 6).

In Chapter 4, we introduce a new machine learning model, called the Alternative Rule Model (ARM), which selects a rule from a ruleset that provides a prediction close to that of any black-box model based on the input instance. Unlike the RCN, which is not designed

to work with arbitrary black-box models, ARM can be applied to work with any black-box model. This model outputs the prediction from the selected rule, allowing humans to verify the correctness of the rule rather than the black-box prediction itself. We also propose an algorithm to find a small, high-coverage ruleset for the ARM. The ruleset should have enough coverage to explain many inputs while remaining small enough for human verification. We formalize this problem as a bilevel optimization problem and solve it by reducing it to a weighted MaxSAT problem. Our approach enables efficient identification of a small ruleset that can support black-box predictions and be checked by humans. Unlike purely rule-based models like decision trees, the ARM has the potential for better predictive performance by selecting rules that are close to the black-box model's predictions.

In Chapter 5, we present a pure rule-based model called the top-k rule list, an ordered list of rules similar to traditional rule lists, but differing in that it selects not just the first rule but the first k rules whose conditions are satisfied by the input, and then makes a prediction by taking an ensemble of these k rules. The top-k rule list extends the traditional rule list, which corresponds to the special case where $k = 1$. This extension allows users to adjust the trade-off between accuracy and interpretability by varying the value of k. While learning a conventional rule list is already known to be a computationally challenging combinatorial optimization problem, constructing a top-k rule list is even more complex, as it requires considering multiple rules for each prediction. To address this, we present an Integer Linear Programming (ILP) approach for learning the top-k rule list, where we substantially reduce the number of variables and constraints by assuming that the rules can be ordered according to their certainty scores. Experimental results demonstrate that, for most datasets, the top-k rule list generated by our ILP formulation with $k > 1$ achieves higher accuracy than the standard rule list.

In Chapter 6, we consider weighted abduction, which is one of the commonly used math-

ematical models for abduction. In general, the abduction problem is NP-hard, and existing approaches such as ILP and Answer Set Programming (ASP) struggle to scale when dealing with large amounts of background knowledge and observations. To address this challenge, we propose an efficient formulation of the depth-limited weighted abduction problem as a weighted MaxSAT problem. Our key insight is that the majority of the resulting hard clauses are Horn, resulting in a formulation that closely resembles the Max Horn SAT problem. This structure enables us to leverage recent advances in MaxSAT solvers. Empirical evaluations on large-scale, real-world instances demonstrate that our approach is up to 100 times faster than existing ILP-based methods, while maintaining equivalent solution quality. These results suggest that SAT-based formulations offer a scalable and effective alternative to ILP for solving weighted abduction problems.

Finally, in Chapter 7, we conclude this thesis and discuss future work. In this thesis, we introduce rule-based explanation methods using optimization solvers, through the framework of logical reasoning based on background knowledge and the framework of data-driven rule-based models. Through this thesis, we contribute to the first step of developing a framework that can generate rule-based explanations linking users' existing knowledge with novel insights derived from the data or black-box models, thereby enabling users to better understand and trust the results. For future work, it is of interest to apply our method to real-world problems and to identify the domains in which it is most suitable.

This thesis is based on the following peer-review articles:

- Y. Sasaki, T. Maehara, T. Akazaki, K. Yamamoto, and K. Sadamasa. Solving weighted abduction via max-sat solvers. Proceedings of the Thirty-Third International Florida Artificial Intelligence Research Society Conference, pp. 142–147. AAAI Press, 2020[1].

- Y. Sasaki and Y. Okajima. Alternative ruleset discovery to support black-box model

[1] ©2020 AAAI. Reprinted, with permission, from [73]

predictions. In IEEE International Conference on Data Mining, pp. 1312–1317, 2021[2].

- Y. Sasaki and Y. Okajima. Alternative ruleset discovery to support black-box model predictions. IEICE Trans. Inf. Syst., 106(6):1130–1141, 2023[3].

- Y. Sasaki and Y. Okajima. Top-k rule list learning via integer linear programming. PRICAI 2024: Trends in Artificial Intelligence - 21st Pacific Rim International Conference on Artificial Intelligence, Proceedings, Part V, volume 15285 of Lecture Notes in Computer Science, pp. 16–28. Springer, 2024[4].

[2]©2021 IEEE. Reprinted, with permission, from [74]

[3]©2023 IEICE. Reprinted, with permission, from [75]

[4]©2024 Springer. Reprinted, with permission, from [76]

Chapter 2

Preliminaries

2.1 Basic definition

$\mathcal{X}$ and $\mathcal{Y}$ denote input domain and output domain, respectively. We also define $f : \mathcal{X} \to \mathcal{Y}$ as a function. $L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ denotes loss function. Given a set of function $\mathcal{F}$, a loss function L, and training instances $T = \{(\boldsymbol{x_i}, t_i)\}_{i=1}^{n}$, where $\boldsymbol{x}_i \in \mathcal{X}$ is an input and $t_i \in \mathcal{Y}$ is a output, the goal of supervised learning problem is to find the function $f \in \mathcal{F}$ such as $\arg\min_{f \in \mathcal{F}} \sum_{i=1}^{n} L(f(\boldsymbol{x_i}), t_i)$. If $\mathcal{Y}$ is a real-valued set $\mathbb{R}$, the problem is called regression. If the output domain $\mathcal{Y}$ is a discrete set $\mathcal{C}$, the problem is called classification. In Chapter 4 and 5 of this thesis, we mainly deal with the G-class classification problem, where $\mathcal{C} = \{C_1, \ldots, C_G\}$.

2.2 Mathematical optimization

In this thesis, we utilize two mathematical optimization approaches: the weighted maximum satisfiability problem and the integer linear programming problem.

2.2.1 Maximum satisfiability problems

The satisfiability (SAT) problem is the problem of determining whether there exists an assignment of truth values to Boolean variables such that a given formula is satisfied. A literal is a variable or its logical negation, which is called a positive literal or a negative literal, respectively. In the SAT problem, the formula is given in the form of a conjunction of clauses, which is called conjunctive normal form (CNF), and each clause is a disjunction of literals. A Horn clause is defined as a clause with at most one positive literal. If all the clauses of the given formula are Horn clauses, the SAT problem is referred to as the Horn SAT problem. Although the SAT problem is known to be NP-hard [15], the Horn SAT problem can be solved in linear time [23].

The maximum SAT (MaxSAT) problem is the problem of finding a truth assignment that maximizes the number of satisfied clauses. The weighted MaxSAT problem is a variant of the MaxSAT problem for considering a formula in which each clause is assigned a weight. A clause having infinite weight is called a hard clause and must be satisfied; otherwise, the clause is called a soft clause. The goal of the weighted MaxSAT problem is to find a truth assignment that maximizes the total weight of satisfied clauses, or equivalently, that minimizes the total weight of unsatisfied clauses. From the viewpoint of minimization, we call the weights costs and say that a clause pays its cost if it is unsatisfied.

Lastly, the Max Horn SAT problem is a MaxSAT problem in which all clauses of the given formula are Horn clauses. This problem is NP-hard [41] and has no constant-factor approximation algorithm unless P = NP [47], but in practice, it can be solved much faster than general MaxSAT problems by applying modern MaxSAT solvers [36, 52].

2.2.2 Integer Linear Programming

An integer linear programming (ILP) problem [46] is a mathematical optimization problem in which all variables are restricted to be integers and both the objective function and all constraints are linear, as follows:

$$\text{maximize}_{\boldsymbol{x} \in \mathbb{Z}^n} \boldsymbol{c}^{\mathrm{T}} \boldsymbol{x}$$

$$\text{subject to } \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}, \boldsymbol{x} \geq \boldsymbol{0}$$

, where $\boldsymbol{c} \in \mathbb{R}^n, \boldsymbol{b} \in \mathbb{R}^m$ are the vectors of size n and m, respectively, and $\boldsymbol{A} \in \mathbb{R}^{m \times n}$. ILP problem is widely recognized as a fundamental and powerful method for solving various combinatorial problems due to its flexibility [46]. Recently, several efficient solvers for the problem has been developed such as Gurobi [1] and CPLEX [2].

[1] http://www.gurobi.com/

[2] https://www.ibm.com/analytics/cplex-optimizer

Chapter 3

Overview of Rule-Based XAI

This chapter presents a brief overview of rule-based explainable AI studies from the perspective of both statistical AI and symbolic AI.

3.1 Background

Recently, the demand for explainability in AI has been rapidly increasing. This is because in practical applications machine learning models are expected not only to make accurate predictions but also to be interpretable. This allows humans to understand the reasoning behind their predictions. Explainability is especially important in high-stakes decision-making tasks such as automated credit approvals and medical diagnosis [71]. There exist several definitions of explainability [16, 22].

On the other hand, a rule is commonly accepted as an interpretable form by humans [49, 51, 69, 81]. In symbolic AI, especially in logical reasoning, rules are typically expressed using propositional or first-order logic. In data-driven AI, a (decision) rule typically consists of a condition over feature values and a corresponding prediction for instances that satisfy

that condition. Both can be interpreted as simple "If-Then" rules. In logical reasoning, it is assumed that the rules are provided as prior knowledge and the goal is to generate new hypotheses by applying rules in a multi-step manner, while in statistical machine learning, the goal is mainly to find the rules from the dataset.

In this chapter, we examine rule-based XAI from the perspective of both data-driven AI and symbolic AI. In this thesis, the former is referred to as *rule-based machine learning*, while the latter corresponds to *abduction* or *abductive reasoning*.

3.2 Rule-based machine learning

Rule-based (machine) learning is defined as an approach to constructing models that employ rules to represent the behavior and distribution of the training data [7]. We categorize rule-based machine learning into *rule-based model* and *rule-based post-hoc explanation*, and provide an explanation for each category as detailed in the following.

3.2.1 Rule-based model

A rule-based model is a type of machine learning model that makes decisions based on "If-Then" rules. In the following, we introduce several representative rule-based models.

First, we define a rule in the context of rule-based machine learning. A rule or a decision rule is represented as $r = (c, \hat{y})$, where c denotes the condition (i.e., If) and $\hat{y} \in \mathcal{Y}$ denotes the prediction (i.e., Then). The condition c is defined as a function $c : \mathbb{R}^d \to \{\mathbf{True}, \mathbf{False}\}$, which returns a Boolean value for a given input $\boldsymbol{x} \in \mathbb{R}^d$. If $c(\boldsymbol{x}) = \mathbf{True}$, we say that the input $\boldsymbol{x}$ satisfies the condition c.

A decision tree is a model composed of a set of mutually exclusive decision rules, represented in the form of a binary tree [13, 67]. Given an input, the decision tree makes a

prediction by following a path from the root to a leaf, where the leaf node represents the output value.

A rule set is a model composed of a set of decision rules [45]. Formally, a rule set of size H is defined as $R = \{r_h\}_{h=1}^{H}$. We also define a default rule as $r_{\text{def}} = (\mathbf{True}, \hat{y}_{def})$, which always outputs $\hat{y}_{\text{def}} \in \mathcal{Y}$ for any input. The ruleset R outputs y_r when $\boldsymbol{x}$ satisfies the condition of exactly one rule $r = (c_r, y_r) \in R$ and outputs $\hat{y}_{\text{def}}$ when $\boldsymbol{x}$ satisfies no rule. If $\boldsymbol{x}$ satisfies multiple rules, a user-defined tie-brake procedure is applied, such as majority voting for classification or averaging the outputs of the satisfied rules.

A rule list of length H is an ordered list of H rules $R = (r_h = (c_h, \hat{y}_h))_{h=1}^{H}$ [70]. Given an input $\boldsymbol{x}$, the rule list returns the prediction of the first rule whose condition is satisfied by $\boldsymbol{x}$. The last rule $r_H = r_{\text{def}}$ is a default rule.

3.2.2 Rule-based post-hoc explanation

A post-hoc explanation for a black-box model is referred to as local if it focuses on explaining a single prediction, and global if it aims to capture the overall behavior of the model.

Global rule-based post-hoc explanations are constructed by approximating the black-box model with a simpler, interpretable rule-based model such as a set of decision rules and a decision tree. For example, there are methods extracting interpretable models from tree ensembles, such as constructing surrogate trees from synthetic data [14, 18], assigning sparse weights to forest nodes [55], or selecting important rules from ensembles [21, 30].

Nevertheless, capturing the full behavior of a complex model using a single, interpretable rule-based model is fundamentally challenging and often leads to low fidelity between the surrogate and the original model's predictions. To overcome this limitation, recent research has shifted toward local rule-based post-hoc explanation methods. Anchor [69] is the first rule-based post-hoc explanations method and the improved approach is proposed in subse-

quent work [20].

One major drawback of these approaches is the high cost of generating explanations for each input. A new explanation must be computed for every prediction, which can be time-consuming and inefficient, especially in real-time applications. Moreover, expecting humans to verify each explanation individually is unrealistic, making these methods impractical in large-scale settings.

3.3 Abductive reasoning

Abduction, also known as abductive reasoning, is a form of inference that aims to find the most plausible explanation or hypothesis for a given observation [64]. It relies on prior background knowledge, which can be considered as a set of rules, to guide the reasoning process, which makes it particularly useful in tasks that require compelling explanations, such as legal analysis [3], plan recognition [33], and discourse analysis [58]. Abduction has long been a core problem in artificial intelligence and was extensively studied in its early development [42].

The abductive reasoning can be described as follows.

- **Given:** Background knowledge B and observations O, where both B and O consist of logical formulas, such as propositional or first-order logic.
- **Find:** a hypothesis H such that $H \cup B \models O$ and $H \cup B \not\models \bot$, where H is a set of logical formulas.

This means that the goal of abductive reasoning is to find a hypothesis that explains the observation when combined with the background knowledge.

Note that it is common for multiple hypotheses to explain the observation. Therefore, various abductive reasoning frameworks have been proposed to identify the most appropriate

hypothesis, such as probabilistic Horn abduction [65], weighted abduction [33], etcetera abduction [38], Markov logic network-based abduction [43], and Bayesian abductive logic programming [80].

Due to the maturity of the study of abduction and the rise of post-hoc explanation approaches, a new combined explanation method, called abductive explanation, has recently been proposed [9, 34, 37, 88]. Given a function f, for each input $\boldsymbol{x} \in \mathcal{X}$, abductive explanations provide a minimal subset of features that are sufficient to generate the outcome $f(\boldsymbol{x})$, based on the fundamental principles of abduction. This approach converts the structure of the given black-box model into background knowledge B and then solve the corresponding abduction problem.

It is important to note that to solve the abduction problem, abductive explanations utilize and depend on existing efficient methods such as [35]. Therefore, the development of efficient solvers for abduction problems remains a crucial task.

Chapter 4

Ruleset Discovery to Support Black-box Model Predictions

In this chapter, we study a method for ruleset discovery to support predictions made by a given black-box model. More specifically, we propose a model-agnostic explanation method to find a pre-verifiable finite ruleset from which a decision rule is selected to support every prediction made by a given black-box model. First, we define an explanation model that selects the rule, from a ruleset, that gives the closest prediction; this rule works as an alternative explanation or supportive evidence for the prediction of a black-box model. The ruleset should have high coverage to give close predictions for future inputs, but it should also be small enough to be checkable by humans in advance. However, minimizing the ruleset while keeping high coverage leads to a computationally hard combinatorial problem. Hence, we show that this problem can be reduced to a weighted MaxSAT problem composed only of Horn clauses, which can be efficiently solved with modern solvers. Experimental results showed that our method found small rulesets such that the rules selected from them can achieve higher accuracy for structured data as compared to the existing method using

rulesets of almost the same size.

The material and all the figures in this chapter have been reproduced from [74][1] [75][2] with the permission of IEEE and IEICE.

4.1 Introduction

The interpretability of machine learning models has been gaining increasing attention because complex, black-box models like deep neural networks and tree ensemble models achieve high accuracy but their complexity prevents humans from understanding the reasoning behind their decisions. Simple rule-based models like decision trees or decision lists have high interpretability, but approximating the whole behavior of a complex model with a simple model may significantly degrade the accuracy or the fidelity to the original model [68].

The conflicting demands of high interpretability and high accuracy have recently led to the development of *local post-hoc* explanation methods that generate a new explanation for every new input such that the explanation is only valid near the input because a complex decision function can be approximated within a small region by a simple model [68, 69]. However, producing a new explanation for every new input requires a huge cost for human checking; it is not practical for humans to check every output and its explanation of a machine learning model in operation.

Explanations should be finite and checkable by humans before a model is deployed in a real application. A recent study proposed a first response to this demand [60]. The rule-constrained network (RCN) is a neural network that selects a decision rule from a finite ruleset for every input and applies the rule to make a prediction. All its predictions are

[1] ©2021 IEEE

[2] ©2023 IEICE

explained by rules from the finite ruleset, which can be checked in advance. The finite ruleset from which the RCN selects rules can be constructed from a large initial ruleset using the ruleset optimization algorithm, which is proposed together with the RCN [60]. However, this model is highly dependent on the neural network structure and cannot be used for other models.

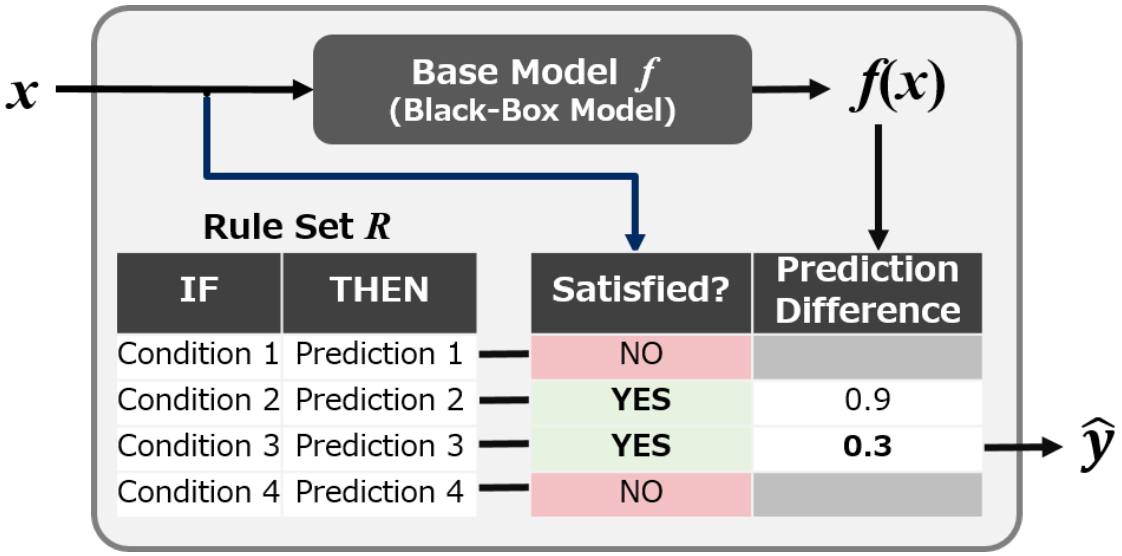


Figure 4.1: The alternative rule model (ARM). Given black-box model f and ruleset R, the ARM selects the rule that has the prediction closest to $f(x)$ from the rules whose conditions are satisfied by x. The prediction of the selected rule is used as the prediction of the ARM. In this example, the third rule is selected, and Prediction 3 is output as the prediction, $\hat{y}$.

In this chapter, we propose a model-agnostic explanation method to find a pre-verifiable finite ruleset from which a decision rule is selected as an explanation. First, we define the alternative rule model (ARM), an explanation model that we use in this chapter. From the ruleset, this model selects a rule as an explanation such that it gives the closest prediction to that of a black-box model while its condition is satisfied by an input instance. Then, the model outputs the prediction from the rule as its prediction for the input as shown in Fig.

4.1. If the ruleset includes rules that give close enough predictions, the selected rules achieve accuracy as high as that of the original black-box model; then, humans can simply verify the correctness of the selected rules instead of the black-box prediction itself.

Next, we propose an algorithm to find a small, high-coverage ruleset for the ARM. The ruleset should have high coverage, i.e. it should include so many rules that the ARM can give close explanations to as many inputs as possible; but it should also be small enough to be checkable by humans in advance. We show that the problem of minimizing the ruleset while maintaining high coverage can be formalized as bilevel optimization, a computationally hard combinatorial problem. We propose a practical solution to this problem by reducing it to a weighted MaxSAT problem composed only of Horn clauses, which can be efficiently solved with modern solvers.

The benefit of our approach is that it enables us to find a small ruleset that provides supportive evidence for the predictions of a given black-box model and can be checked by humans in advance. The rule selected for a prediction does *not* explain what happens in the black-box model. Instead, it provides humans an alternative way to check an interpretable rule that gives a close prediction. If the prediction is close enough and the rule is reliable, then humans can accept the prediction. Because the rules are selected from a small ruleset, they can also be used as a screening measure for further analysis with other methods by skipping instances that are assigned rules that humans have already verified.

In comparison with models that are purely based on rules such as decision trees, our approach has a potential advantage in terms of predictive performance, because the predictions of the ARM are selected to be close to those of black-box models and thus could provide better predictions than predictions that are purely based on rules.

Our experimental results show that the ARMs with the rulesets optimized by our method provided more accurate predictions for structure data using almost the same number of rules

Rule	Condition	Class probability
r_1	$x_1 = "A"$	$[0.2, 0.8]$
r_2	$x_1 = "B"$	$[0.8, 0.2]$
r_3	$x_2 > 2$	$[0.6, 0.4]$
r_4	$x_2 < 8$	$[0.3, 0.7]$
r_5	$x_1 = "B"$ AND $x_2 > 2$	$[0.9, 0.1]$

Table 4.1: An example of a decision-rule set. The conditions are defined for a feature representation $x = (x_1, x_2)^T$. The predictions of the rules are binary class probabilities.

as compared to the RCNs and purely rule-based models, while maintaining high fidelity to the predictions of the base models.

The remainder of this chapter is organized as follows. In Section 4.2, we introduce the ARM with the description of basic notations and definitions. In Section 4.3, we present the algorithm to find a small, high-coverage ruleset for the ARM. In Section 4.4, we show our experiments and the results. The characteristics of the ARM are discussed in Section 4.5, the related work is reviewed in Section 4.6, and the conclusion of the chapter is given in Section 4.7.

4.2 Alternative Rule Model

We consider a K-class classification problem with training instances $D = \{(\boldsymbol{x}_i, t_i)\}_{i=1}^n$, where $\boldsymbol{x}_i$ is a d-dimensional feature vector in $\mathbb{R}^d$ and t_i is its label chosen from a set of labels $\mathcal{G} = \{G_1, \ldots, G_K\}$. We assume that we are given a pre-trained black-box model or *base model*, $f : \mathbb{R}^d \to \Delta^{K-1}$, in which Δ^{K-1} is a K-dimensional probability simplex, where each point represents a class probability vector over $\mathcal{G}$.

Let R denote the ruleset from which the ARM selects rules. A decision rule $r \in R$ is a pair $(c_r, \boldsymbol{y}_r)$, where c_r is a condition over feature values and $\boldsymbol{y}_r$ is a prediction for instances that satisfy c_r. For classification, $\boldsymbol{y}_r$ is a vector of class probabilities over $\mathcal{G}$; Table 4.1 lists examples. The probability of belonging to a class is the normalized count of training instances of the class that satisfy the condition. We call a decision rule based on such simple counting *verifiable*, because the user can verify the prediction by recounting the training instances satisfying the condition. Such a property is often required by data analysts, because they may not understand the reasons for model decisions until they look into the data behind the model.

Verifiable rules can be extracted by using association rule mining or random forests trained with the training data by considering paths from the root to each leaf as decision rules [60]. The ARM can accept any Boolean-valued functions of the type $c_r : \mathbb{R}^d \to \mathbb{B}$ as conditions because the ARM only needs to access the truth values of conditions; thus the form of conditions only depends on the method used for the construction of the ruleset. The construction of the ruleset will be further explained in Section 4.3.

Given a test instance $\boldsymbol{x} \in \mathbb{R}^d$, the ARM extracts the rules in R whose conditions are satisfied by $\boldsymbol{x}$; from them, it selects the rule whose prediction is closest to $f(\boldsymbol{x})$, which we call the *alternative rule* for $\boldsymbol{x}$. Finally, the ARM outputs the prediction of the alternative rule. The alternative rule works as an alternative explanation of $f(\boldsymbol{x})$, because it explains how a similar prediction to $f(\boldsymbol{x})$ can be obtained by applying a decision rule to $\boldsymbol{x}$.

This selection of an alternative rule can be written in a functional form as

$$g(\boldsymbol{x}, R, f) := \underset{\substack{r \in R \\ \text{s.t. } \boldsymbol{x} \text{ satisfies } c_r}}{\arg\min} \; L(f(\boldsymbol{x}), \boldsymbol{y}_r), \tag{4.1}$$

where L is a loss function that measures the error or closeness between two predictions. In this study, we use the Kullback-Leibler (KL) divergence as L:

$$L_{KL}(\boldsymbol{y}, \boldsymbol{y'}) = \sum_{k=1}^{K} y_k \log \frac{y_k}{y'_k}, \tag{4.2}$$

which is defined between two class probability vectors $\boldsymbol{y} = (y_1, .., y_K)$ and $\boldsymbol{y'} = (y'_1, .., y'_K)$.

4.3 Ruleset Optimization

4.3.1 Problem definition

The ARM selects the rule from the ruleset that gives the closest prediction to that of the base model as an alternative explanation, which means that the quality of explanations depends on whether the ruleset covers possible predictions made by the base model. If the ruleset is too small, in most cases the ARM fails to find a rule that gives a close prediction. If the ruleset is too large, however, the cost of human checking may be unacceptably high: an analyst can hardly check all the rules, and previously unchecked rules may pop up for every new prediction. How can we build a ruleset that has enough rules for future predictions? One simple approach is to add decision rules to the ruleset until the ARM can return close enough predictions for all training instances. Unfortunately, this approach may lead to a huge ruleset in which each rule covers only a small fraction of the training data. To avoid a high checking cost, we need to find a small ruleset with high coverage.

To this end, we solve a ruleset optimization problem. We first generate an initial ruleset R_0, which is large enough to provide predictions close to the base model's predictions for all training instances. The goal is to extract a subset R from R_0 such that R contains a number of rules small enough for human checking and also covers the predictions of the base model as widely as possible. To achieve this goal, we minimize the sum of the cost of incorporating rules into the ruleset and the loss between the predictions of the base model and the ARM

for the training instances. The ruleset optimization problem is formally defined as follows:

Problem 1 (A ruleset optimization problem for the ARM)**.**
Input: *A set of training instances $D = \{(\boldsymbol{x_i}, t_i)\}_{i=1}^n$, an initial ruleset R_0, a base model f, and a rule cost λ.*
Output: *A ruleset R such that*

$$R \quad = \quad \underset{R \subset R_0}{\arg\min} \sum_{i=1}^{n} L(f(\boldsymbol{x}_i), \boldsymbol{y}_{r_i}) + \lambda |R|, \tag{4.3}$$

where

$$r_i \quad = \quad g(\boldsymbol{x}_i, R, f). \tag{4.4}$$

The rule cost, λ, works as a regularization hyperparameter. A higher cost leads to a smaller ruleset in which each rule needs to cover more training instances and is thus expected to be more general than in a larger ruleset.

The ruleset optimization problem is a bilevel optimization problem, which is known to be strongly NP-hard [29].

4.3.2 Weighted Max Horn SAT formulation

Here, we show that the ruleset optimization problem can be reduced to a weighted MaxSAT problem. This reduction aims to take advantage of modern MaxSAT solvers, which are known to be able to solve various problems efficiently when they are reduced to weighted MaxSAT problems [73, 89].

We introduce two types of Boolean variables. The first type, o_r, is introduced for all $r \in R_0$ and takes `True` if and only if r is included in R. The second type, $e_{i,r}$, is introduced for all $1 \leq i \leq n$ and $r \in R_0$ if and only if $\boldsymbol{x}_i$ satisfies c_r; $e_{i,r}$ takes `True` if and only if r is selected as the alternative rule for $\boldsymbol{x}_i$.

Soft clause (Objective function) The target formula of the weighted MaxSAT problem is

$$\bigwedge_{r \in R_0} (\neg o_r) \wedge \bigwedge_{\substack{i=1..n,\ r \in R_0, \\ \text{s.t. } \boldsymbol{x}_i \text{ satisfies } c_r}} (\neg e_{i,r}). \tag{4.5}$$

This formula is composed of multiple clauses each of which includes just one negative literal. The clauses are assigned weights as follows:

$$w(\neg o_r) = \lambda, w(\neg e_{i,r}) = L(f(\boldsymbol{x}_i), \boldsymbol{y}_r). \tag{4.6}$$

These weights mean that we pay cost λ if r is included in R or cost $L(f(\boldsymbol{x_i}), \boldsymbol{y_r})$ if r is the alternative rule for x_i.

Hard clause 1 If a rule is selected as the alternative rule for any input instance, it must be included in R. This requirement is met by adding the following hard constraints:

$$\bigwedge_{\substack{i=1..n,\ r \in R_0, \\ \text{s.t. } \boldsymbol{x}_i \text{ satisfies } c_r}} (e_{i,r} \Rightarrow o_r). \tag{4.7}$$

Hard clause 2 Each training instance must have at least one rule that can be its alternative rule. This requirement is met by adding the following hard constraints:

$$\bigwedge_{i=1..n} \bigvee_{\substack{r \in R_0 \\ \text{s.t. } \boldsymbol{x}_i \text{ satisfies } c_r}} e_{i,r}. \tag{4.8}$$

The weighted MaxSAT problem with these soft and hard clauses is equivalent to the ruleset optimization problem. We also show that this problem can be reduced to a weighted Max Horn SAT problem.

Theorem 1. *A ruleset optimization problem for an ARM can be reduced to a weighted Max Horn SAT problem.*

Proof. Each clause in (4.5), (4.7), and (4.8) consists of at most one negative literal. These logical formulas, which are called dual-Horn SAT formulas, can be converted to Horn SAT formulas by replacing each Boolean valuable with its negation (i.e., $o_r = \neg o'_r$, $e_{i,r} = \neg e'_{i,r}$) [78]. □

As mentioned in Sec. 2.2.1, Max Horn SAT problems are efficiently solved by modern MaxSAT solvers. Thus, these solvers can be expected to efficiently solve the ruleset optimization problem.

4.4 Experiments

4.4.1 Experimental Setup

Datasets We conducted experiments with the unstructured and structured datasets listed in Table 4.2. The unstructured datasets consisted of the top five largest datasets from the UCR time-series repository [17], which were the same ones used in the RCN paper [60], which were the mean, variance or slope of each interval extracted from the time series with a certain window size. Only the CNNs took raw unstructured data as input directly because their hidden layers learn feature representations of unstructured data automatically. Table 4.2 shows the number of features after the data preparation above. The structured datasets were tabular data from the UCI data repository [24]. The categorical features in the structured data were expanded using one-hot encoding.

Base Models We used neural networks (NNs) and random forests (RFs) as the base (black-box) models of our method. In general, RFs perform better than NNs for structured data, while NNs perform well for unstructured data. Our method is model-agnostic, and we expected that it could work well with both structured and unstructured data by changing the

Table 4.2: Details of datasets used in the experiments.

	Name	#Instances	#Features
Unstructured	Straw	983	208
	Prox	891	48
	FordB	4446	412
	FordA	4921	412
	Phar	2658	48
Structured	adult	32562	105
	diabetic	101766	252
	miniboone	130064	252
	phishing	11055	46
	shopper	12330	74
	susy	5000000	18
	truck	60000	170

base model; in contrast, the RCN is dependent on its neural network structure and was not expected to perform well with structured data. To evaluate this point, we used NNs and RFs as the base models for our method; hereafter we call them the *base NNs* and the *base RFs*, respectively. The base NNs for unstructured data were the same CNNs used in the RCN paper, while the base NNs for structured data were multi-layer perceptrons (MLPs). The hyper parameters of MLPs were tuned in the inner cross validation (CV) as follows: the number of middle layers was selected from $[1, 2, 3]$; the number of nodes in a middle layer was selected from $[16, 32, 64, 128, 256]$; $weight\_decay$ was selected from [1e-4, 1e-3, 1e-2, 1e-1, 1.0, 1e+1]. The architecture of the CNN that was used as the base NN for unstructured data is shown in Table 4.3. It is the same CNN used in the RCN paper [60], which is originally

proposed in [84]. For the CNNs, we uses the same hyper parameters as used in the RCN paper, which are shown in Table 4.3, without tuning. All the NNs were trained by the Adam optimizer with its default hyper parameters.

The base RFs were an ensemble of 100 decision trees. We used the scikit-learn implementation for random forests. The best value of $min\_samples\_leaf$ was selected from $[1, 2, 4, 8, 16, 32, 64]$ in the inner CV. The other hyper parameters were set to the default values.

For each dataset, the hyper parameters of the base NNs and RFs were tuned in a nested cross validation.

Initial Rulesets RCNs and ARMs both require initial rulesets as inputs. We constructed initial rulesets in the same way as in the RCN paper [60]. For each train/test split, we used the training data to build a shallow RF of 20 decision trees with the maximum depth set to 4, which we call the *rule source RF*. Then the initial ruleset was constructed from the rule source RF by considering each path from the root to a leaf as a rule. Note that we trained the rule source RFs for initial ruleset construction apart from the base RFs, because the base RFs were too deep to extract interpretable rules. Thus, the goal of both ARMs and RCNs was to construct a finite ruleset from the initial ruleset, which were derived from the rule source RFs, in order to explain the predictions of the base RFs or NNs, which were too complex to interpret directly.

RCNs For each train/test split, we constructed an RCN from the base NN in the same way explained in the original RCN paper [60]. The ruleset for the RCN was extracted from the initial ruleset using the ruleset optimization algorithm proposed for the RCNs, simultaneously with the training of the RCN on the training data. The performance of the RCN was then evaluated using the test data with the optimized ruleset.

ARMs For each train/test split, a ruleset for the ARM was extracted from the initial rule-

set by solving the ruleset optimization problem, described in Section 4.3, using open-wbo (Ver. 2.0)[3], a free and efficient MaxSAT solver, with a 200 seconds timeout.

Purely Rule-based Models We also evaluated three purely rule-based models as baselines: decision trees (DTs), defragTrees, and CORELS. We used the scikit-learn implementation for decision trees. The best value of $min\_samples\_leaf$ was selected from $[1, 2, 4, 8, 16, 32, 64]$ in the inner CV. The other hyper parameters were set to the default values.

defragTrees is one of the state-of-the-art simplification methods that extract simple rules from a tree ensemble and, in the original paper [30], experimentally obtained smaller rules than other methods [14,21,55] keeping the same accuracy. The original paper of defragTrees [30] proposes two algorithm to find a ruleset. The EM algorithm works with fixed ruleset size K, while the other algorithm, called factorized asymptotic Baysian (FAB) inference, can determine K automatically. We used the original authors' implementation of defragTrees[4] to extract rules from the base RFs.

CORELS [4] is a state-of-the-art optimal decision list learner. We used the authors' implementation[5] in the experiments. CORELS's regularization hyper parameter, which determines the trade off between the accuracy and the list size, was set to its smallest value (the inverse of the training data size), so that CORELS can achieve its best accuracy. We set the maximum node size to 10^8, which is used in the original paper [4]. The other hyper parameters were also set to the default values.

Cross Validation We conducted nested cross validation to evaluate models with hyper-parameter tuning. Each dataset was first divided into five training/test splits by five-fold CV (the outer CV). For each split, we built the initial ruleset, trained models and selected

[3]http://sat.inesc-id.pt/open-wbo/ Because the solver can only handle integer weights, we multiplied the weights by 100 and rounded them to integers.

[4]https://github.com/sato9hara/defragTrees

[5]https://github.com/corels/pycorels

the best hyper parameters of models in three-fold CV on the training data (the inner CV); the models with the selected hyper parameters were evaluated on the test data.

Data Sampling For each train/test split of the structured datasets, the size of the training data was reduced by random sampling to 1,000 if larger than 1,000, and the test data was reduced by random sampling to 100,000 if larger than 100,000, because several datasets were very large. In addition, for the ruleset optimization of the ARMs, we only used 500 instances randomly selected from the training data, if it was larger than 500, to obtain a ruleset in a reasonably short time.

Computing environment All experiments except for CORELS were conducted with an Intel Xeon E3-1279 V2 (4C/3.50 GHx/8M) processor and 32 GB of memory on CentOS 7.7. Since CORELS exhausted the memory in the above environment, the experiments for CORELS were conducted in a larger memory environment with two Intel Xeon E5-2667 v4 (8C/3.20GHz/25M) processers and 768 GB of memory.

4.4.2 Results

Our method and the RCN both have regularization hyperparameter λ, which penalizes the size of a ruleset. A smaller ruleset is obtained with a higher λ for our method, but with a smaller λ for the RCN. Because the accuracy depends on the size of the ruleset for both methods, we conducted two experiments. In the first experiment, we compared our method with the RCN in terms of accuracy with λ fixed. Then, in the second experiment, we varied λ and measured the effect of λ on accuracy.

In the first experiment, we fixed λ so that our method and the RCN constructed rulesets of almost the same size. Unfortunately, the size of rulesets cannot be directly specified for both methods. Thus, we forced them to use rulesets of almost the same size in the following way. First, we trained the RCN with λ set to 100, which means the ruleset size was limited

below 100. Then, for each dataset, we constructed three rulesets that were optimized by our method with λ set to 0.01, 0.1, and 1, respectively; and then we employed the largest ruleset that was smaller than that of the RCN for the same dataset. As for defragTrees with the EM algorithm, the ruleset size can be directly specified with parameter K; thus we set K to 100.

The results of the first experiment are shown in Table 4.4. As for the two uninterpretable base models, the RFs outperformed the NNs for the structured data by winning six out of seven datasets while the NNs only won three, although the NNs outperformed the RFs for most unstructured datasets. Because the RCNs were based on the NNs, the RCNs also performed worse than the RFs for the structured data. On the other hand, the ARM using the RFs as the base model, denoted by ARM(RF) in Table 4.4, outperformed the RCN for almost all the structured data, while the ARMs using the NNs as the base model, denoted by ARM(NN), performed as well as the RCNs for the unstructured data. These results showed that the ARM using the rulesets optimized by our method can achieve high accuracies for different data if suitable models are used as the base models, unlike the RCN that depends on neural networks.

Table 4.4 also shows that the ARMs outperformed all the purely rule-based models (the decision trees, defragTrees with the EM algorithm and FAB inference, denoted by defragTrees(EM) and defragTrees(FAB), respectively, and CORELS) for almost all datasets. The purely rule-based models showed lower accuracies because their predictions are only based on interpretable rules in the test phase, although defragTrees extracts rules from black-box models in the training phase. On the other hand, the predictions of the ARMs rely on the base models in the test phase. The base models first determine their predictions, and the ARMs select the closest rules according to the base models. Because of this mechanism, the ARMs can outperform the purely rule-based models as long as the base models outperform the purely rule-based models. For the same reason, however, the ARMs cannot outperform

the purely rule-based methods if the base models do not outperform the purely rule-based models. For example, for the diabetic dataset, CORELS and RFs had almost the same accuracy, and thus ARM(RF) performed as well as CORELS but did not significantly outperform it.

To check whether the ARMs provided predictions close to those of the base models, we define the *fidelity* of a ARM to the base model as the accuracy of the predictions of the ARM that is measured by considering those of the base model as true labels. The measured fidelities shown in Table 4.5 revealed that the ARMs could provided the same predictions for more than 95% of the predictions of the base models, except for one dataset (diabetic). They indicate that more than 95% predictions of the base models were supported by decision rules; and the ARMs changed the remaining few predictions of the base models because they were not supported by any decision rules in the rulesets.

In the second experiment, we checked the effect of λ on accuracy by varying λ. For the structured datasets, the ARMs using the RFs as the base models were expected to outperform the RCNs if the ARM can provide predictions close enough to those of the RFs. The goal of the second experiment was to check whether the ARM outperformed the NNs even when changing the values of λ. Besides, we measured the effect of λ on the fidelity of the ARMs to the RFs. We also measured the accuracies of defragTrees(EM) for different ruleset sizes for comparison.

The results are shown in Fig. 4.2. Even for different sizes of the ruleset, the ARMs outperformed the RCNs for four datasets (adult, miniboone, shopper, and susy). For the other three datasets (diabetic, phishing and truck), the RCNs performed comparable or slightly better than the ARMs, but the differences were not much larger than the standard errors.

We also observed the trade-off between the accuracy, or the fidelity, and the size of the ruleset; the accuracies and the fidelities were relatively low if the rulesets were small. This

observation highlights the importance of choosing λ to balance the smallness of the ruleset with the accuracy or the fidelity to the base model.

4.5 Discussion

In this section, we discuss the characteristics of the ARM from different viewpoints.

4.5.1 Interpretability of obtained rules

First, we evaluate the explainability of the ARMs by examining rules obtained for two real datasets, adult and strawberry, which are examples of structured and unstructured datasets, respectively.

Figure 4.3 shows a ruleset optimized for the ARM(RF) with $\lambda = 10$ for the adult dataset. The goal for the adult dataset is to predict whether income is high (> 50k). As seen from Figure 4.3, the seven rules in the ruleset can be classified into two groups: the four rules with condition 'Marital-status="Married-civ-spouse"' (Rules 1–4) and the three rules with its negation, 'Marital-status$\neq$"Married-civ-spouse"' (Rules 5–7). Rule 7 says that the income is low with high probability (0.94) if the Marital-status is not "Married-civ-spouse". Rules 5–6 work as the exceptions for Rule 7; they show additional conditions that increase the probability of high income. On the other hand, Rule 4 gives borderline probabilities, $[0.52, 0.48]$, which indicates the Marital-status being "Married-civ-spouse" is not enough to predict whether the income is high. Rules 1, 2, and 3 expand Rule 4 with additional conditions. Rules 1 and 2 say that high education and being in middle age lead to high income, while Rule 3 says low education and low capital status lead to low income.

Figure 4.4 shows the rule selected for a real test instance. This instance satisfies Rules 1, 2, and 4, and Rule 1 is selected as the explanation because it gives the closest prediction to the

base model's prediction, $[0.15, 0.85]$. In comparison with Rules 2 and 4, Rule 1 gives more additional conditions (high education and being in middle age), which provides supportive evidence for the base model's prediction that the income is high.

For the strawberry dataset, Figure 4.5 shows a ruleset and Figure 4.6 shows the rule selected for a real test instance. The goal for the strawberry dataset is to predict whether the spectrographs are generated from authentic strawberries. Features "mean(s/e)" and "var(s/e)" represent the mean and the variance of the absorbance in interval from s to e. As seen from Figure 4.5, the most frequently used feature is 'mean(11/47)'. Rule 2 says the input instance is authentic strawberry with high probability if 'mean(11/47)' is low. In contrast to Rule 2, Rule 6 says the input instance is not authentic strawberry with probability 0.66 if 'mean(11/47)' is high. The other rules give additional conditions based on features other than 'mean(11/47)'. For the test instance in Figure 4.6, Rule 5 is selected because it gives the closest prediction to the base model's prediction, $[0.995, 0.005]$.

Through these observations, we conclude that it is not difficult to interpret what the obtained rules mean and how the conditions in the rules affect the predictions of the rules, especially for the adult dataset. As for the strawberry dataset, even though the rules themselves are simple, the interpretation may require domain knowledge due to its highly technical nature.

4.5.2 Structured Data vs Unstructured Data

We discuss the difference between structured data and unstructured data. The first difference is the suitable base models. Because the ARM's accuracy is affected by the base model's accuracy, it is important to select the base models suitable for the data type. RFs perform better than NNs for structured data, while NNs perform better than RFs for unstructured data. Thus, the ARMs using the RFs as the base models perform well with structured data, while

the ARMs using the NNs as the base models perform well with unstructured data, as seen in the experiments.

The second difference is the necessity of feature engineering. The examination of rulesets in Section 4.5.1 indicates that rules for structured and unstructured data can be interpreted in almost the same way. However, the difference is that unstructured data require feature engineering to extract interpretable features from the raw data. We used the original features for the structured data (except for one-hot encoding for categorical features), but for unstructured data, we needed to design and extract features from the raw data. The interpretability of rules highly depends on the selection of features. We used means and variances as features in the experiments, but domain experts may be able to design more interpretable or more effective features. Careful consultation with domain experts should be needed to design features for unstructured data.

4.5.3 Comparison with CORELS and defragTrees

We discuss the differences between the ARM and the two purely rule-based models, CORELS and defragTrees, on the basis of the experimental results. The ARM outperformed the purely rule-based models in terms of accuracy. The ARM's high accuracy comes from the fact that its predictions depend on the black-box model's predictions. Compared with the ARM, CORELS showed lower accuracy with a smaller number of decision rules. CORELS tends to output small decision lists because its branch-and-bound algorithm starts with a decision list of length zero and exhaustively search all possible decision lists of short lengths, making it hard to reach large decision lists. Even though CORELS tends to make short decision lists, rules in a list cannot be interpreted independently. In a decision list made by CORELS, rules are ordered by priority, and the prediction of a rule is affected by all the preceding rules. Thus, each rule cannot be interpreted without considering other rules, which makes it diffi-

cult to interpret decision lists [45]. On the other hand, the ARM and defragTrees tend to use more rules than CORELS, but their rules can be interpreted independently, unlike CORELS, because the prediction of a rule is not affected by other rules. These differences among the models should be considered when they are used in practical applications.

4.5.4 Limitation

We discuss the limitation of the explainability of the ARMs. The main focus of the ARM is just to provide a rule that gives a prediction close to that of the black-box model, and, as described in Section 4.1, the rules selected by the ARM do *not* explain what happens in the black-box model. The black-box model and the selected rules are internally based on different inference mechanisms and may depend on different features, even if they eventually reach the same prediction. Thus, it is not appropriate to use the ARM to interpret how the black-box model actually works. For example, the ARM should not be used to check whether the black-box model depends on sensitive features such as sex or race in the context of bias detection because the black-box model's predictions may internally depend on sensitive features even if the explanations do not. This problem, called *fairwashing*, was already reported for other explanation techniques [1].

4.5.5 Use Cases

We discuss the use cases of the ARM in industrial applications. The ARM can be used for both structured and unstructured data, but the latter requires feature engineering to design interpretable features, as discussed in Section 4.5.2. Thus, it is straightforward to apply the ARM to the domains that already have structured data with interpretable features and experts who can interpret them. From this viewpoint, finance and healthcare would be the

most important application areas for the ARM.

In finance, loan examiners evaluate the financial data of loan applicants and decide whether to offer a loan to them. This process requires great effort, and AI support is in demand. High interpretability is required to support experts' decision-making, while high accuracy is also needed to make a profit on their loan business. The ARM is suitable for such cases. It can offer high accuracy thanks to the black-box model behind it and also provides a set of interpretable rules. A possible use case scenario is to classify loan applicants according to the rules assigned by the ARM. If the loan examiners think the assigned rules are reliable enough, the examination of the applicants can be simplified, and the efforts involved in the examination can be reduced. As a result, the loan examiners can concentrate their effort on the examination of the applicants who are assigned rules with borderline probabilities and thus need further investigation by human experts.

The same discussion holds for healthcare. Electric health records are structured data with interpretable features, and doctors spend great efforts interpreting them. High accuracy and high interpretability are both necessary to support doctors in diagnosing. The decision rules given by the ARM can be easy for doctors to interpret because doctors are accustomed to evaluate clinical test results with certain thresholds they empirically know.

4.5.6 Future Work

As discussed in Section 4.5.4, the ARM is not suitable for gaining insight into the internal mechanism of the black-box model, and the features used in selected rules may be different from the features used by the black-box models. In order to overcome this limitation, a possible line of future work is to make the ARM not only make the same predictions, but also use the same features as the black-box model uses. Another important future work is to evaluate the effectiveness of the ARM in real applications such as finance and healthcare, as

discussed in Section 4.5.5.

4.6 Related Work

Here, we categorize studies on interpretability into three approaches (for a survey, see [28]).

The first approach tries to improve the accuracy of simple, interpretable models like decision trees or decision lists. The accuracy of decision lists has recently been significantly improved by exploiting the high performance of modern computers and search algorithms [4, 48, 59, 83, 86], although decision lists still perform less accurately than black-box models.

The second approach produces post-hoc explanations when given a black-box model. An explanation is called *local* if it only explains a single prediction made by the model or *global* if it explains the model's whole behavior.

Global explanations can be made by approximating a black-box model with a simple model like a decision tree. TREPAN [18] and born again trees [14] construct a simple decision tree from synthetic examples labeled by a black-box model. Node harvest [55] simplifies a random forest by assigning sparse weights to nodes using quadratic programming with linear inequality constraints. inTrees [21] extracts frequent rules from a tree ensemble. defragTrees [30] finds a small set of rules that approximates a tree ensemble by modeling the tree ensemble in a probabilistic manner.

Explaining the whole behavior of a complex model by a single simple model is inherently difficult and leads to low fidelity to the original complex model. To overcome this problem, many recent studies have focused on producing local explanations by linear models (LIME) [68], decision rules (Anchor) [69], feature importance (SHAP) [50], and important instances [44].

The third approach incorporates interpretable models as part of the decision making of

black-box models by using the interpretable part as an explanation. The RCN selects a decision rule from a finite ruleset and makes a decision based on the selected rule [60], whereas a preceding study generated a graphical model for every decision [2]. The RCN is the most similar model to ours, because both select a decision rule from a finite ruleset and optimize the ruleset in terms of smallness in size while keeping predictive performance; but our method is model-agnostic unlike the RCN and outperformed the RCN for structured data in the experiments reported in this chapter.

4.7 Conclusion

We proposed a model-agnostic explanation method that finds a finite ruleset from which a decision rule is selected as an explanation for each prediction made by a given black-box model. The selected rules give predictions close to those of the black-box model, but the predictions are verifiable by humans, unlike those of the black-box model.

We showed that finding a small ruleset that covers the predictions of a black-box model is a computationally hard bilevel optimization problem but can be reduced to a weighted Max Horn SAT problem, which can be solved efficiently with modern MaxSAT solvers.

Experimental results showed that our method found small rulesets such that the rules selected from them can achieve higher accuracy for structured data as compared to existing methods using rulesets of almost the same size, while maintaining high fidelity to the predictions of the base models. We also experimentally compared the proposed method with CORELS and defragTrees.

Furthermore, we examined rulesets constructed for real datasets and discussed the characteristics of the proposed method from different viewpoints including interpretability, limitation and possible use cases.

Table 4.3: CNN: The base NN for the unstructured data.

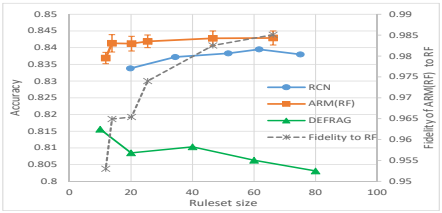
Conv2D	
# filters	128
kernel size	8×8
strides	1×1
padding	valid
Batch normalization	
ReLU	
Conv2D	
# filters	256
kernel size	5×5
strides	1×1
padding	valid
Batch normalization	
ReLU	
Conv2D	
# filters	128
kernel size	3×3
strides	1×1
padding	valid
Batch normalization	
ReLU	
GlobalAveragePooling2D	
Dense	
Softmax	

Table 4.4: Accuracies of the models averaged over the outer cross validations. ARM(RF) and ARM(NN) represent the ARMs using the base RFs and the base NNs as the base models. The numbers in parentheses and brackets represent the standard errors of the means and the sizes of the rulesets, respectively. For DTs, the number of leaves is considered as the size of the ruleset. The numbers in boldface indicate the models that achieved the highest accuracy among uninterpretable or interpretable models. The models whose accuracy fell within the range of one standard error from the highest accuracy were also written in boldface.

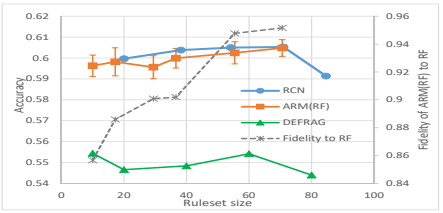
	Uninterpretable base models		Interpretable						
	RF	NN	DT	CORELS	defragTrees(EM)	defragTrees(FAB)	RCN	ARM(RF)	ARM(NN)
Straw	**.9644(.0091)**	.9390(.0273)	.9257(.0100)[21.6]	.9532(.0080)[6.0]	.9349(.0118)	.9278(.0130)[12.8]	**.9715(.0062)[55.0]**	.9624(.0092)[54.4]	.9451(.0273)[47.8]
Prox	.8530(.0099)	**.9114(.0087)**	.8160(.0130)[36.0]	.8104(.0212)[6.4]	.7834(.0050)	.7575(.0093)[8.8]	**.8911(.0126)[74.6]**	.8429(.0075)[74.0]	**.9024(.0115)[65.0]**
FordB	.7393(.0069)	**.8837(.0028)**	.6766(.0041)[46.4]	.6662(.0056)[4.6]	.6943(.0070)	.6907(.0071)[12.6]	**.8770(.0042)[99.6]**	.7409(.0054)[88.6]	**.8749(.0042)[84.8]**
FordA	.7604(.0049)	**.9130(.0068)**	.6818(.0040)[50.8]	.6804(.0084)[4.2]	.6743(.0123)	.6757(.0055)[13.6]	**.9023(.0050)[75.2]**	.7568(.0028)[36.0]	**.9079(.0061)[70.2]**
Phar	.8232(.0056)	**.8499(.0062)**	.7604(.0131)[137.6]	.7246(.0087)[4.6]	.7427(.0105)	.7137(.0096)[13.4]	.8273(.0088)[78.2]	.8074(.0072)[39.4]	**.8345(.0036)[72.2]**
adult	**.8470(.0019)**	.8351(.0026)	.8120(.0038)[12.8]	.8357(.0025)[5.2]	.8077(.0026)	.8004(.0061)[9.6]	.8380(.0013)[75.0]	**.8429(.0021)[66.2]**	.8321(.0035)[59.6]
diabetic	**.6060(.0046)**	.5834(.0024)	.5780(.0110)[35.6]	**.6087(.0054)[5.4]**	.5482(.0080)	.5549(.0036)[12.6]	.5914(.0141)[84.6]	**.6048(.0040)[70.6]**	.5929(.0025)[79.4]
miniboone	**.8987(.0009)**	.8887(.0014)	.8606(.0046)[29.8]	.8592(.0052)[7.2]	.8689(.0032)	.8643(.0037)[10.8]	.8879(.0029)[76.8]	**.8958(.0011)[53.0]**	.8898(.0012)[76.2]
phishing	**.9401(.0022)**	**.9412(.0026)**	.9173(.0057)[67.0]	.9118(.0029)[4.8]	.8644(.0066)	.9008(.0025)[9.0]	**.9345(.0038)[71.2]**	**.9325(.0029)[46.4]**	**.9326(.0049)[43.0]**
shopper	**.8880(.0029)**	.8811(.0043)	**.8904(.0025)[20.0]**	**.8898(.0013)[6.2]**	.8723(.0078)	.8492(.0011)[10.4]	.8886(.0037)[79.8]	**.8922(.0024)[69.0]**	.8835(.0036)[63.2]
susy	.7802(.0021)	**.7835(.0013)**	.7386(.0056)[30.2]	.7546(.0024)[6.0]	.7371(.0056)	.7225(.0012)[11.8]	.7718(.0041)[84.6]	**.7795(.0020)[61.4]**	.7771(.0028)[51.8]
truck	**.9841(.0004)**	**.9846(.0007)**	.9822(.0014)[3.8]	.9803(.0009)[3.4]	**.9837(.0006)**	.9833(.0006)[7.8]	**.9846(.0010)[47.4]**	**.9837(.0006)[15.6]**	**.9843(.0007)[12.8]**

Table 4.5: Fidelities of the ARMs to the base models. The numbers in parentheses represent the standard errors.

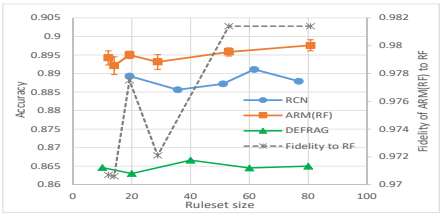
	ARM(RF)	ARM(NN)
Straw	.9939(.0030)	.9919(.0041)
Prox	.9652(.0107)	.9776(.0073)
FordB	.9678(.0079)	.9863(.0025)
FordA	.9602(.0047)	.9937(.0010)
Phar	.9601(.0098)	.9545(.0060)
adult	.9851(.0026)	.9674(.0045)
diabetic	.9516(.0063)	.9351(.0134)
miniboone	.9814(.0028)	.9839(.0023)
phishing	.9756(.0049)	.9782(.0026)
shopper	.9765(.0046)	.9886(.0033)
susy	.9823(.0026)	.9721(.0047)
truck	.9977(.0013)	.9960(.0007)



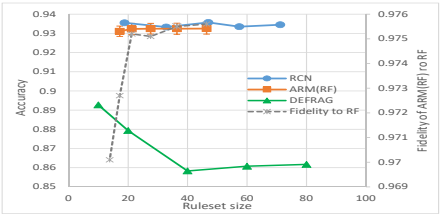
(a) adult



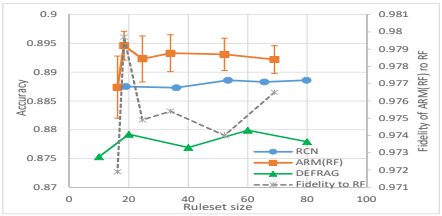
(b) diabetic



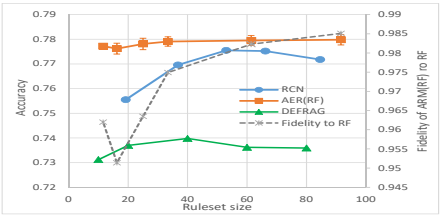
(c) miniboone



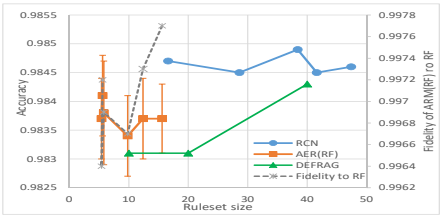
(d) phishing



(e) shopper



(f) susy



(g) truck

Figure 4.2: Mean accuracies of the RCNs and the ARMs based on the RFs with different values of λ. The blue and orange lines indicate the accuracies of the RCNs and the ARMs, respectively. The vertical bars indicate the range of one standard error. The gray dot lines are the fidelity of the ARMs to the RFs. The values of the RCN's λ were [100, 80, 60, 40, 20], and the values of λ of our method were [0.01, 0.1, 0.5, 1, 2, 3]. The green lines are the accuracies of defragTrees for different Ks in [80, 60, 40, 20, 10], which represent the exact ruleset size.

	IF	THEN
1	Marital-status="Married-civ-spouse" AND Education-num>9 AND 35<Age≤59	[0.18,0.82]
2	Marital-status="Married-civ-spouse" AND Education-num>10	[0.25,0.75]
3	Marital-status="Married-civ-spouse" AND Education-num≤11 AND Capital-loss≤1791.5 AND Capital-gain≤5095.5	[0.71,0.29]
4	Marital-status="Married-civ-spouse"	[0.52,0.48]
5	Marital-status≠"Married-civ-spouse" AND Education-num>12 AND Age>30	[0.69,0.31]
6	Marital-status≠"Married-civ-spouse" AND Capital-gain≤7731.5 AND Age>33 AND Sex="Male"	[0.82,0.18]
7	Marital-status≠"Married-civ-spouse"	[0.94,0.06]

Figure 4.3: A ruleset optimized for the ARM(RF) with $\lambda = 10$ for the adult dataset. Numbers in THEN column represent the probabilities of the income being low and high, respectively.

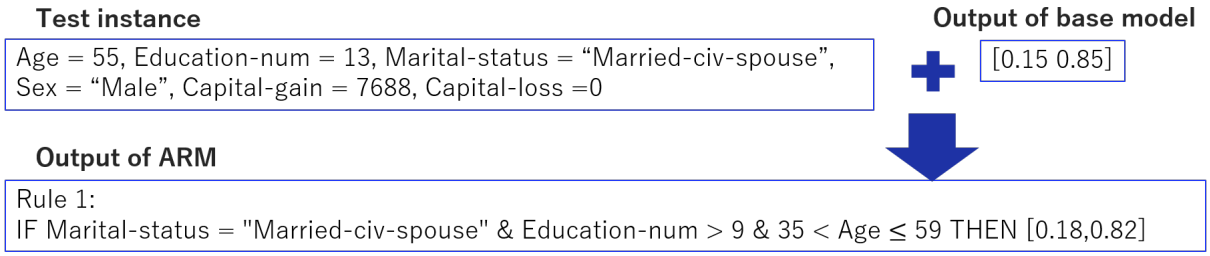


Figure 4.4: An example of a test instance from the adult dataset, the output of the base model, and the rule selected by the ARM.

	IF	THEN
1	mean(11/47)≤1.1238 AND var(15/47) ≤0.0001 AND var(3/23)>0.0676	[0.58,0.42]
2	mean(11/47) ≤1.1215	[0.08,0.92]
3	mean(11/47)>1.1077 AND mean(20/23)>-0.9229	[0.08,0.92]
4	mean(11/47)>1.1088 AND mean(20/23) ≤-0.9228 AND var(3/11) ≤0.0303	[0.36,0.64]
5	mean(11/47)>1.1088 AND mean(20/23) ≤-0.9228 AND var(3/11)>0.0303 AND var(21/47)>0.0008	[0.96,0.04]
6	mean(11/47)>1.1097	[0.66,0.34]

Figure 4.5: A ruleset optimized for the ARM(RF) with $\lambda = 10$ for the strawberry dataset. Numbers in THEN column represent the probabilities of the input instance is not authentic strawberry and is authentic strawberry, respectively.

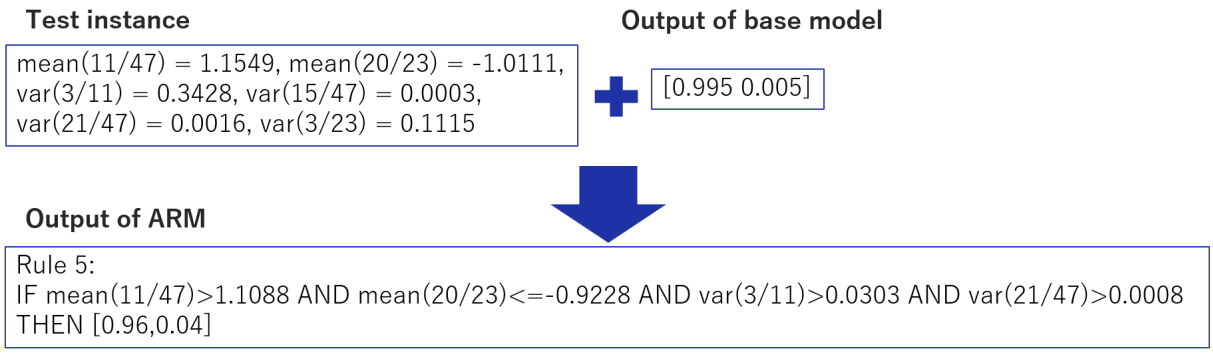


Figure 4.6: An example of a test instance from the strawberry dataset, the output of the base model, and the rule selected by the ARM.

Chapter 5

Top-k Rule List Learning

In this chapter, we study a new rule-based model called a top-k rule list in order to improve the accuracy of rule lists. In contrast to the traditional rule list, which makes a prediction by using the first rule in the list whose conditions are satisfied by the input, the top-k rule list utilizes an ensemble of the first k rules instead of just the first one. The top-k rule list can be viewed as a generalization of the traditional rule list, which is a special case with $k = 1$, and this generalization enables the user to obtain a preferable trade-off between accuracy and interpretability by selecting different values of k. Learning the traditional rule list is known as a computationally hard combinatorial optimization problem, and learning the top-k rule list can be even harder because multiple rules must be considered for each prediction. To enable efficient learning, we propose an integer linear programming formulation for learning the top-k rule list, and effectively reduce the number of variables and constraints by introducing an assumption that the rules are ordered by certainty score. The experimental results show that the top-k rule list learned with the ILP formulation for $k > 1$ outperformed the traditional rule list in accuracy for most datasets.

The material and all the figures in this chapter have been reproduced from [76][1] with the permission of Springer.

5.1 Introduction

A rule list [70] is one of the best-known interpretable models for meeting this interpretability demand. A rule list, also known as a decision list, is essentially an ordered list of rules. When given an input, the rules are checked from the top to the bottom of the list, and the prediction is made by applying the first rule whose condition is satisfied by the input. This simple inference algorithm makes it easy for users to understand how the predictions are made. Despite the simplicity of the inference algorithm, learning a rule list is computationally hard due to its combinatorial nature. To tackle this problem, the research community has developed a variety of learning algorithms, including ones based on a branch-and-bound method [5], mathematical programming [72], and continuous relaxation [59]. Although these studies have experimentally shown that rule lists achieve competitive and often better performances than other rule-based models such as decision trees, the problem is that rule lists still have considerably lower accuracy than complex black-box models such as tree ensembles. It is inherently difficult to fill the gap in accuracy between rule lists and tree ensembles because rule lists can only apply a single rule for each prediction whereas tree ensembles can make a more robust prediction by applying an ensemble of multiple rules.

In this chapter, in order to overcome this limitation of the traditional rule list and improve the accuracy, we propose a model called a top-k rule list. It is an ordered list of rules, the same as the traditional rule list, but differs in that it selects not just the first rule but rather the first k rules whose conditions are satisfied by the input and then makes a prediction by taking

[1] ©2024 Springer

an ensemble of the k rules. The top-k rule list is a generalization of the traditional rule list, which is a special case with $k = 1$. The advantage of the top-k rule list is that different k values provide different tradeoffs between accuracy and interpretability. The top-k rule list with a larger k can achieve higher accuracy because its prediction is made by an ensemble of more rules, but its interpretation may be more difficult because more rules need to be checked. The value k can thus be chosen on the basis of user preference about how many rules they can check to achieve high accuracy.

As none of the existing algorithms can be used to optimize a rule list for $k > 1$, we propose a learning algorithm designed for the top-k rule list. The rule list learning problem becomes even harder for $k > 1$ due to the complexity of identifying the first k rules that are used for each prediction during training. To reduce the complexity, we take a three-step approach consisting of rule generation, ordering, and selection. The selection step is conducted by solving an integer linear programming (ILP) problem in which the identification of the top-k rules are represented as ILP constraints. Although a large number of variables and constraints is needed to identify the top-k rules if all possible orders are considered in the ILP problem, we show that this number can be reduced by determining the order of rules in the ordering step under the common assumption of rule lists that the rules closer to the top are preferable.

In the experiments, we trained the top-k rule list with $k \in [1, 3, 5]$ using the ILP formulation. The results showed that the top-k rule list with $k = 3$ showed a good trade-off between accuracy and the list size; it outperformed the traditional rule list trained by the state-of-the-art learning algorithm for most datasets while having less than 25 rules.

5.2 Model Description

In this section, we review the traditional rule list and then propose the top-k rule list.

5.2.1 Traditional Rule List (Top-1 Rule List)

We consider a G-class classification with training instances $T = \{(\boldsymbol{x_i}, t_i)\}_{i=1}^{n}$, where $\boldsymbol{x}_i \in \mathbb{R}^d$ is an input and $t_i \in \mathcal{C} = \{C_1, \ldots, C_G\}$ is a label. A rule r is a pair $(c, \hat{\boldsymbol{y}})$ of a condition $c : \mathbb{R}^d \to \{\texttt{True}, \texttt{False}\}$ and a class probability vector $\hat{\boldsymbol{y}} \in \Delta^{G-1}$, where Δ^{G-1} denotes $G-1$ dimensional simplex and $\hat{\boldsymbol{y}}$ represents a prediction for any $\boldsymbol{x}$ that satisfies c (i.e., $c(\boldsymbol{x})$ is true). For example, a rule can be like ('$x_1 > 30$' $\wedge$ '$x_2 < 100$', $[0.8, 0.2]$) for the binary classification of inputs in the form of $\boldsymbol{x} = [x_1, x_2]$.

A rule list of length M is an ordered list of M rules $R = (r_m = (c_m, \hat{\boldsymbol{y}}_m))_{m=1}^{M}$. Given an input $\boldsymbol{x}$, the rule list returns the prediction of the first rule whose condition is satisfied by $\boldsymbol{x}$. The last rule $r_M = \tilde{r} = (\texttt{True}, \hat{\boldsymbol{y}}_{def})$ is a special rule called a *default rule* whose condition is true for any input so that the list always returns a prediction.

Following several previous studies [66, 86], we define the consequent of a rule to be class probabilities, instead of a single class. This is because, while a class probability vector can easily be converted into a single class by just taking the class with the maximum probability, probabilities have additional information about how *certain* the prediction is. We say a rule is certain if one of the classes is given a much higher probability than the other classes, and the certainty score of a rule can be derived from its class probabilities using existing measures such as the Gini index or the entropy. The certainty score will be used to order rules in Section 5.3.

5.2.2 Top-k Rule List

A top-k rule list returns the average of predictions of the first k rules whose conditions are satisfied by the input. As with a traditional rule list, a top-k rule list R of length M is an ordered list of M rules. However, to guarantee that any input satisfies at least k rules, we assume that the last k rules $(r_{M-k+1}, ..., r_M)$ are the same default rules whose condition is always true. We define the $covers$ function as follows:

Definition 1. *For the m-th rule $r_m = (c_m, \hat{\boldsymbol{y}}_m)$, an input $\boldsymbol{x}$, and $1 \leq h \leq M$, we define $covers(r_m, \boldsymbol{x}, h)$ as*

$$covers(r_m, \boldsymbol{x}, h) = \mathbb{1}(c_m(\boldsymbol{x}) \wedge \text{`}|C_{\boldsymbol{x},m}| = h\text{'}),$$

where $C_{\boldsymbol{x},m} = \{c_l | c_l(\boldsymbol{x}) = True; \forall 1 \leq l \leq m\}$ and $\mathbb{1}(\cdot)$ is the indicator function that takes 1 if the argument is true and 0 otherwise.

Note that $covers(r_m, \boldsymbol{x}, h)$ equals 1 if r_m is the h-th rule that is satisfied by $\boldsymbol{x}$ and equals 0 otherwise. The $covers$ function defined here is a generalization of the function of the same name introduced in [63], which is defined for the traditional rule list. Using the $covers$ function, we define the prediction of the top-k rule list as follows:

Definition 2 (Prediction of a top-k rule list)**.** *For input $\boldsymbol{x}$, $1 \leq k \leq M$, and the top-k rule list $R = (r_m = (c_m, \hat{\boldsymbol{y}}_m))_{m=1}^M$, the prediction $\hat{y} = h_R(\boldsymbol{x})$ is defined as*

$$h_R(\boldsymbol{x}) = \frac{1}{k} \sum_{m=1}^{M} \hat{\boldsymbol{y}}_m \sum_{h=1}^{k} covers(r_m, \boldsymbol{x}, h). \tag{5.1}$$

This formula means that the prediction of a top-k rule list is the average of the predictions of the first k rules whose conditions are satisfied by the input.

5.2.3 Calculating Predictions of Rules

The m-th rule r_m is a pair of condition c_m and the prediction $\hat{\boldsymbol{y}}_m$. To calculate the predictions of M rules $\hat{\boldsymbol{y}}_1, \ldots, \hat{\boldsymbol{y}}_M$, the previous studies on the traditional rule list take one of the following two approaches.

The first approach is simple: the training instances that satisfy c_m are counted for each class. The class probability vector $\hat{\boldsymbol{y}}_m$ is the vector of the counts that is normalized over the classes, i.e., $y_{m,g} = \frac{|\{\boldsymbol{x}_i | t_i = C_g \wedge c_m(\boldsymbol{x}_i)\}|}{|\{\boldsymbol{x}_i | c_m(\boldsymbol{x}_i)\}|}$, where $y_{m,g}$ is the g-th element of $\hat{\boldsymbol{y}}_m$. A rule is called *verifiable* if its prediction consists of the normalized counts of training instances, as users can easily verify the rule by counting the training examples that satisfy the condition [74]. This approach has been utilized in several studies [59, 72].

The second approach is a little more complicated. The class probability vector $\hat{\boldsymbol{y}}_m$ is also given as the normalized counts of training instances that satisfy c_m, but the training instances that satisfy the conditions of the $m-1$ preceding rules $(c_1, \ldots, c_{m-1})$ are excluded, i.e., $y_{m,g} = \frac{|\{\boldsymbol{x}_i | t_i = C_g \wedge c_m(\boldsymbol{x}_i) \wedge \neg(\vee_{m'=1}^{m'=m-1} c_{m'}(\boldsymbol{x}_i))\}|}{|\{\boldsymbol{x}_i | c_m(\boldsymbol{x}_i) \wedge \neg(\vee_{m'=1}^{m'=m-1} c_{m'}(\boldsymbol{x}_i))\}|}$. This approach has also been utilized in various studies [5, 48].

We call the first and second approaches *order-insensitive* and *order-sensitive*, respectively, because the prediction of a rule is affected by the order of rules in the second approach but not in the first. Both approaches have pros and cons. The order-sensitive approach is likely to learn a shorter list because a rule can be simpler if the training instances are reduced by its preceding rules. However, rules are rather hard to interpret or verify because the prediction of a rule is affected by all the preceding rules, and thus users have to consider all of these rules in order to interpret or verify one rule. The order-insensitive approach has the opposite characteristics: the list is likely to be longer, but the prediction of each rule does not depend on the preceding rules and thus is easier to interpret or verify.

In this chapter, we take the order-insensitive approach because it is suitable for the top-

k rule list. An ensemble of rules does not work properly in the order-sensitive approach because the prediction of a rule is affected by the preceding rules, and thus the mutual independence of rules is violated.

5.3 Learning Top-k Rule Lists

In this section, we propose the algorithm to learn the top-k rule list. Most of the current research on rule list learning [5,59,72,86] has taken a two-step approach. In Step 1 (rule generation), a set of rules is generated from training data using an existing rule mining method. In Step 2 (selection and ordering), a rule list is constructed by selecting rules from the rule set and ordering them appropriately. In this chapter, following the previous studies, we assume that a set of rules Z is generated from a set of training instances $T = \{(\boldsymbol{x}_i, t_i)\}_{i=1}^n$. One popular approach to generate a ruleset utilizes association rule mining [5, 72]. Another approach uses random forests [59], where a random forest is trained with the training instances and then rules are extracted from the decision trees in the random forest by considering each path from the root to a leaf as one rule.

Let $\mathcal{R}(Z)$ denote the set of all rule lists all of whose rules are chosen from Z. After the rule set Z is generated, we obtain the optimized rule list by solving the following optimization problem:

$$\min_{R \in \mathcal{R}(Z)} \sum_{i=1}^{n} L(h_R(\boldsymbol{x}_i), \boldsymbol{t}_i) + \lambda |R|, \tag{5.2}$$

where L is a loss function, λ is a regularization parameter, and $\boldsymbol{t}_i$ is a one-hot vector corresponding to label t_i. Almost the same objective has been used for the optimization of the traditional rule list [5, 72]. However, it is known to be a hard combinatorial optimization problem [70], and it is difficult to find the optimal rule list for a large dataset in a practical amount of time [59].

The recent studies based on the two-step approach try to deal with two things, rule selection and ordering, at the same time, and this is where the main difficulty arises. For example, a mathematical formulation of the traditional rule list proposed in [72] tries to determine the selection and order of rules simultaneously by introducing a $|Z| \times |Z|$ binary matrix to represent the permutation of $|Z|$ rules. The matrix will be huge if the ruleset size is large, and such a large number of variables deteriorates the efficiency. Their paper states that this method "takes longer to run and cannot handle a huge dataset". Furthermore, such formulation is hard to extend for the case with $k > 1$, which is considered in this chapter, because it requires a number of complicated constraints to identify the first k rules to be used for prediction if any permutation is possible.

To make the learning more efficient, we propose imposing restrictions on the order of the rules. The inference algorithms of the traditional and top-k rule lists are based on the common assumption that the rules closer to the top are preferable. The prediction of the (top-k) rule list is made by taking the first rule(s) that are satisfied by the input and disregarding any rules that follow. This algorithm requires that more certain rules be located before less certain rules. For example, in binary classification, rules with a certain prediction such as $[0.99, 0.01]$ are expected to be located before rules with less certain predictions such as $[0.5, 0.5]$. Thus, it is natural to assume that the rules in the top-k rule list are ordered in descending order of their certainties.

On the basis of this consideration, we take a three-step approach: 1) rule generation, 2) rule ordering, and 3) rule selection. A rule set is generated in Step 1. In Step 2, we evaluate the certainty scores of the rules in the rule set and order the rules in descending order of the scores. In Step 3, we solve the optimization problem (5.2) by selecting rules from the rule set with the order restriction. Our technical focus is the rule selection, and any existing measure that scores the certainty of rules, such as the Gini index and the entropy,

can be utilized for the ordering. In the experiments, we used the negative of the Gini index as certainty scores for ordering rules. We index the rules in the rule set as $Z = \{z_1, \ldots, z_{|Z|}\}$ in the descending order of the scores. To simplify the formulation, we assume that the last k rules $z_{|Z|-k+1}, \ldots, z_{|Z|}$ are the same default rule; its class probabilities equal the ratios of the classes in all training data.

After the order of rules is determined in Step 2, we solve the problem of rule selection in Step 3. To represent the selection of rules, we introduce a binary vector $\boldsymbol{\gamma}$ of size $|Z|$ where $\gamma_{m'} = 1$ if $z_{m'}$ is included in R and 0 otherwise. The rules are ordered in R in the same order as in Z. Thus, the rule list R is uniquely specified if $\boldsymbol{\gamma}$ is determined.

The problem is now reduced to the problem of assigning 1 or 0 to the elements of $\boldsymbol{\gamma}$. However, it is still a hard combinatorial problem because there are $2^{|Z|}$ possible solutions. We solve this assignment problem by formulating it as an ILP problem. To define the ILP problem, we introduce several notions and variables. Let s_i be the number of rules in Z whose conditions are satisfied by $\boldsymbol{x}_i$. Let $\boldsymbol{b}_i = (b_{ij})_{j=1}^{s_i}$ be the indexes of the s_i rules in Z that are satisfied by $\boldsymbol{x}_i$. We say $z_{b_{ij}}$ is a *prediction rule* for $\boldsymbol{x}_i$ if $z_{b_{ij}}$ is selected in R (i.e., $\gamma_{b_{ij}} = 1$) and $z_{b_{ij}}$ is one of the first k rules whose conditions are satisfied by $\boldsymbol{x}_i$ (i.e., $z_{b_{ij}}$ is used to make the prediction for $\boldsymbol{x}_i$). We introduce a vector of binary variables $\boldsymbol{D}_i = (D_{ij})_{j=1}^{s_i}$ for every $1 \leq i \leq n$, where D_{ij} is 1 if $z_{b_{ij}}$ is a prediction rule for $\boldsymbol{x}_i$ and 0 otherwise. Additionally, we introduce a vector of integer variables $\boldsymbol{\theta} = (\theta_i)_{i=1}^{n}$, where θ_i is in $[1, s_i]$. We use θ_i to indicate the position on $\boldsymbol{b}_i$ such that $z_{b_{ij}}$ is a prediction rule if both $j \leq \theta_i$ and $\gamma_{b_{ij}} = 1$ hold and not a prediction rule otherwise. Note that such a position θ_i exists because rules in R are in the same order as those in Z. With these variables for all $1 \leq i \leq n$ and $1 \leq j \leq s_i$, the constraints on the prediction rules can be expressed as follows:

$$\sum_{j=1}^{s_i} D_{ij} = k; \ \forall i, \tag{5.3}$$

$$\begin{cases} D_{ij} > 0 \text{ if } j \leq \theta_i \text{ and } \gamma_{b_{ij}} > 0, \\ D_{ij} < 1 \text{ if } j \leq \theta_i \text{ and } \gamma_{b_{ij}} < 1, \\ D_{ij} < 1 \text{ if } j > \theta_i. \end{cases} \tag{5.4}$$

Constraint (5.3) guarantees that the number of prediction rules for each input is just k. Constraint (5.4) guarantees that the prediction rules are the first rules in R whose conditions are satisfied by the input. (5.4) can be rewritten without if-clauses as follows:

$$\begin{cases} |Z|(D_{ij} + 1 - \gamma_{b_{ij}}) + (j - \theta_i) > 0, \\ |Z|(1 - D_{ij} + \gamma_{b_{ij}}) + (j - \theta_i) > 0, \\ |Z|(1 - D_{ij}) + (\theta_i - j + 1) > 0, \end{cases} \tag{5.5}$$

for all $1 \leq i \leq n$ and $1 \leq j \leq s_i$. If D_{ij} satisfies Constraint (5.5), it also satisfies (5.4). This can be verified by checking all possible four cases, i.e., the combinations of $\gamma_{b_{ij}}$ is 1 or 0 and $j > \theta_i$ or otherwise. Finally, the following constraints are required to guarantee that the k default rules are included in the list:

$$\gamma_{|Z|-k+1} = \cdots = \gamma_{|Z|} = 1. \tag{5.6}$$

Under constraints (5.3), (5.5), and (5.6), the prediction of the top-k rule list (5.1) can be rewritten as $h_R(\boldsymbol{x}_i) = \frac{1}{k} \sum_{j=1}^{s_i} \hat{\boldsymbol{y}}_{b_{ij}} D_{ij}$, and thus the objective function (5.2) can be rewritten as $\underset{\boldsymbol{\gamma},\boldsymbol{\theta},\boldsymbol{D}}{\arg\min} \sum_{i=1}^{n} L((\frac{1}{k} \sum_{j=1}^{s_i} \hat{\boldsymbol{y}}_{b_{ij}} D_{ij}), \boldsymbol{t}_i) + \lambda \sum_{m'=1}^{|Z|} \gamma_{m'}$. If the loss function L is affine with respect to the prediction, we have $L(\frac{1}{k} \sum_{i=1}^{k} \boldsymbol{y}_i, \boldsymbol{t}) = \frac{1}{k} \sum_{i=1}^{k} L(\boldsymbol{y}_i, \boldsymbol{t})$, and thus the objective function can be rewritten as

$$\underset{\boldsymbol{\gamma},\boldsymbol{\theta},\boldsymbol{D}}{\arg\min} \sum_{i=1}^{n} \sum_{j=1}^{s_i} \frac{D_{ij}}{k} L(\hat{\boldsymbol{y}}_{b_{ij}}, \boldsymbol{t}_i) + \lambda \sum_{m'=1}^{|Z|} \gamma_{m'}. \tag{5.7}$$

Objective (5.7) is linear with respect to $\boldsymbol{\gamma}$, $\boldsymbol{\theta}$, and $\boldsymbol{D}$ because $L(\hat{\boldsymbol{y}}_{b_{ij}}, \boldsymbol{t}_i)$ is constant. Thus, we can minimize objective (5.7) as an ILP problem under constraints (5.3), (5.5), and (5.6)

Table 5.1: Details of datasets. Categorical features were expanded using one-hot encoding.

Name	adult	diabetic	higgs	miniboone	shopper	susy	magic	hepmass
#Instances	32562	101766	11000000	130064	12330	5000000	19020	10500000
#Features	105	252	28	252	74	18	10	28

and obtain the optimized $\boldsymbol{\gamma}$. Note that only $\boldsymbol{\gamma}$, $\boldsymbol{\theta}$, and $\boldsymbol{D}$ are variables in the ILP problem, and all others are treated as constants. Finally, the top-k rule list R is constructed from $\boldsymbol{\gamma}$. In the experiments, we utilized the L_1 loss function, $L_1(\boldsymbol{y}, \boldsymbol{t}) = \sum_{g=1}^{G} |y_g - t_g|$, because t_g only takes the values of zero or one and thus L_1 is affine.

We conclude this section by reviewing how the number of variables and constraints is reduced in our formulation. Because our ILP formulation only deals with the rule selection, only the binary vector γ of length $|Z|$ is needed for the representation, whereas a $|Z| \times |Z|$ binary matrix is required if the formulation needs to deal with the permutation of rules. Furthermore, because the order of rules is determined in advance, the k prediction rules for $\boldsymbol{x}_i$ can be identified using just one position variable θ_i, whereas more variables will be needed if the permutation of rules is allowed in the formulation. As the result of the reduction of variables, the number of variables in our ILP formulation is $O(n|Z|)$ because the largest group of variables is D_{ij} for $1 \leq i \leq n$ and $1 \leq j \leq s_i (\leq |Z|)$. The number of constraints is also $O(n|Z|)$ because most of the constraints are constraints (5.4) and (5.5) for the same combinations of i and j as for D_{ij}.

5.4 Experiments

Datasets We conducted experiments with the tabular datasets from the UCI data repository [24] listed in Table 5.1. Large datasets with over $10,000$ instances were selected to reduce

Table 5.2: Accuracies of models averaged over the outer cross-validations. Numbers in brackets represent the averaged number of the rules. For decision trees, the number of leaves is shown instead of the number of the rules. Numbers in bold indicate the models with the highest accuracy or the accuracies that fell within the range of one standard error of the highest. For top-k rule lists, underscored numbers represent the cases where the optimal solution was reached in Step 3 before the time limit. The last row shows the averaged ranks among the models.

	Decision tree	CORELS	ORL ($n = 100$)	Top-k rule list (λ=3) $k = 1$	$k = 3$	$k = 5$	Ensemble of 3 DLs
adult	.809[13.2]	**.835**[6.0]	.808[13.2]	**.837**[11.4]	.832[12.6]	.827[14.2]	**.835**[35.0]
diabetic	.555[34.6]	.586[6.0]	.562[32.2]	.583[18.6]	**.600**[13.0]	**.597**[8.8]	.581[65.0]
higgs	.596[27.4]	.590[8.0]	.577[35.0]	.615[19.4]	**.629**[21.2]	.613[17.8]	.615[61.2]
miniboone	.834[22.2]	.855[7.4]	.849[14.0]	.864[11.0]	**.871**[15.6]	**.871**[20.6]	.869[33.8]
shopper	.883[13.6]	.885[5.2]	.872[13.2]	.883[8.4]	**.892**[9.2]	**.891**[12.2]	**.889**[30.6]
susy	.741[18.4]	.755[5.6]	.733[24.0]	.755[12.6]	**.761**[19.2]	.757[21.2]	.757[50.2]
magic	.781[15.2]	.794[5.8]	.774[17.6]	.800[12.6]	**.811**[16.6]	.787[15.8]	**.812**[40.2]
hepmass	.800[9.6]	.808[5.6]	.781[22.0]	.811[12.2]	**.813**[17.0]	.807[18.2]	**.816**[42.2]
Ave. rank	6.063	4.125	6.750	3.438	1.688	3.375	2.563

the standard error in the evaluation.

Cross-validation (CV) We conducted nested CV. Each dataset was first divided into five training/test splits by five-fold CV (the outer CV). For each split, we selected the best hyper parameters for models in three-fold CV on the training data (the inner CV); the models with the selected hyper parameters were evaluated on the test data.

Data Sampling The datasets were collectively very large, so we reduced the sizes by random sampling. For each train/test split, the size of the training and test data were reduced by random sampling to 500 and 10^7 if larger than these values, respectively.

Table 5.3: The accuracies of top-k rule lists with $k \in [3, 5]$ and $\lambda \in [1, 2, 3]$. Numbers in brackets represent the averaged number of the rules. For each λ, we compared the accuracy of $k = 3$ with that of $k = 5$ and highlighted the higher accuracy in bold; but we also highlighted the lower accuracy that fell within the range of one standard error of the higher.

	$k = 3$			$k = 5$		
	$\lambda = 1$	$\lambda = 2$	$\lambda = 3$	$\lambda = 1$	$\lambda = 2$	$\lambda = 3$
adult	**.841**[29.4]	**.835**[17.6]	**.832**[12.6]	**.840**[31.6]	**.834**[20.0]	**.827**[14.2]
diabetic	.596[40.4]	**.600**[21.6]	**.600**[13.0]	**.605**[38.8]	**.604**[17.0]	**.597**[8.8]
higgs	**.633**[40.6]	**.629**[28.2]	**.629**[21.2]	**.633**[41.6]	**.630**[27.0]	.613[17.8]
miniboone	**.875**[27.8]	**.872**[19.8]	**.871**[15.6]	**.873**[32.4]	**.873**[24.2]	**.871**[20.6]
shopper	**.887**[21.4]	**.891**[11.6]	**.892**[9.2]	**.888**[20.8]	**.890**[15.0]	**.891**[12.2]
susy	**.762**[32.2]	**.763**[24.8]	**.761**[19.2]	**.763**[37.2]	**.762**[27.6]	.757[21.2]
magic	**.815**[32.8]	**.813**[23.2]	**.811**[16.6]	.809[39.0]	.805[24.2]	.787[15.8]
hepmass	**.819**[31.2]	**.818**[21.6]	**.813**[17.0]	**.819**[36.6]	**.814**[24.0]	.807[18.2]

Models We compared top-k rule lists with traditional rule lists trained by CORELS [5] and ORL [72]. ORL takes the order-insensitive approach and is based on mathematical programming, while CORELS takes the order-sensitive approach and utilizes a branch-and-bound solver customized to learn rule lists. We used the authors' implementation[2] for CORELS and our own implementation for ORL. For CORELS and ORL, their regularization hyper parameter, which determines the trade-off between accuracy and list size, was set to its smallest value (the inverse of the training data size for CORELS and the inverse of the size of a given ruleset plus 1 for ORL) so that they can achieve their best accuracy with the least limitation on the list size. For CORELS, we set the maximum node size to 10^8, which is what was used

[2]https://github.com/corels/pycorels

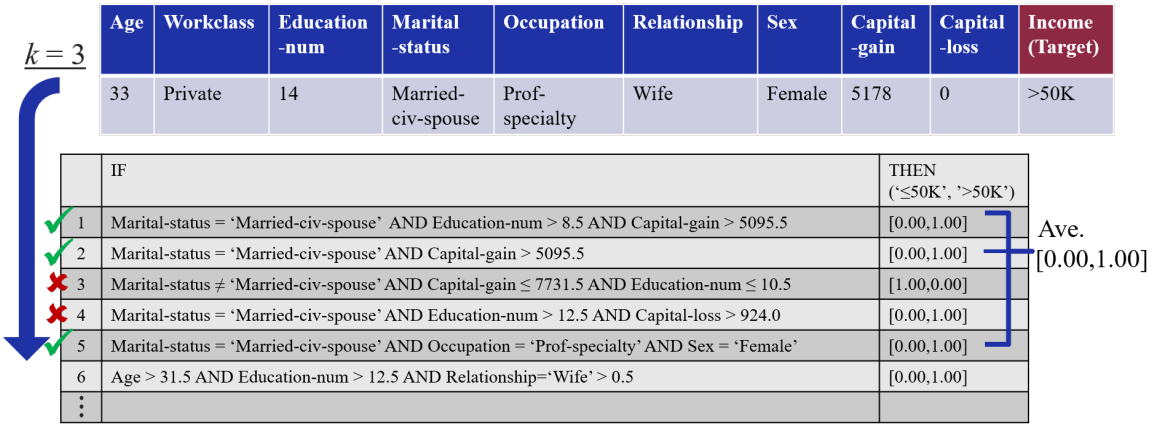


Figure 5.1: An example of a top-k rule list trained with $k = 3$ for the adult dataset. Given the test instance shown at the top, the prediction of the top-k rule list is the average of the predictions of Rules 1, 2, and 5, the first three rules whose conditions are satisfied.

in the original paper [5]. The other hyper parameters were set to the default values. The top-k rule list was evaluated for $k \in [1, 3, 5]$ with the regularization parameter λ set to 1. To solve the ILP problem for ORL and the top-k decision list, we used the CPLEX optimizer[3] with the default parameters except for the time limit set to 5000s and the MIP emphasis parameter set to 4, which emphasizes finding hidden feasible solutions. We also evaluated decision trees as baselines using their scikit-learn implementations. The best value of $min\_samples\_leaf$ was selected from $[1, 2, 4, 8, 16, 32, 64]$ in the inner CV. The other hyper parameters were set to the default values. We also evaluated the bagging ensemble of three top-k rule lists with $k = 1$, denoted as *Ensemble of 3DLs* in Table 5.2, to see how the top-k rule list perform differently from an ensemble of k rule lists.

Ruleset Generation The top-k rule list, CORELS, and ORL require a ruleset to be given as input. For fair comparison, we provided them with the same ruleset. For each train/test split,

[3]https://www.ibm.com/analytics/cplex-optimizer

we used the training data to train a random forest of 20 decision trees with the maximum depth set to 3. The ruleset was then obtained from the random forest by considering each path from the root to a leaf as a rule. We set the maximum depth to 3 because rule lists are hard to interpret if the conditions of rules are too complicated.

Computing Environments All experiments except for CORELS were conducted with an Intel Xeon E3-1279 V2 (4C/3.50 GHx/8M) processor and 32 GB of memory, which we call the *default environment*. CORELS exhausted the memory in the default environment, so the experiments for CORELS were conducted again in a larger memory environment with two Intel Xeon E5-2667 v4 (8C/3.20GHz/25M) processers and 768 GB of memory, which we call the *large-memory environment*.

5.4.1 Results

The accuracies of the models are shown in Table 5.2. As we can see, CORELS achieved competitive or better performances with fewer rules compared to the decision trees and ORL. As stated above, CORELS's regularization hyper parameter was set to its smallest value, which means it can achieve its best accuracy with the least limitation on the list size. Additionally, only CORELS was evaluated in the large-memory environment because it had exhausted the memory in the default environment, where other models were evaluated. Thanks to the larger memory, CORELS successfully found the rule lists optimal with the given ruleset. From these observations, we conclude that the results shown in Table 5.2 are the best possible performances of CORELS in our settings and environments. For the evaluation of ORL, we downsampled the training instances, because ORL performed much worse when trained with 500 instances; ORL needs to solve the ordering of rules, and so it was so slow that it did not reach optimal solutions when many instances were given. We evaluated ORL with downsampled training data sizes (50, 100, 200, and 500). ORL with 100 training instances showed

its best performance, which we show in Table 5.2, but it was still worse than CORELS.

Next, we examine the results of the top-k rule lists. In the evaluation shown in Table 5.2, λ was set to 3 so that most lists would include less than 25 rules because too long lists are hard to interpret. For $k = 1$, the top-k rule list outperformed ORL even though both used the same ILP solver, because, unlike ORL, the top-k rule list does not need to solve the ordering in the optimization and thus reached the better accuracy within the same time limit. There are no considerable differences in accuracy between the top-k rule list with $k = 1$ and CORELS, but the top-k rule lists with $k = 3$ outperformed CORELS for all datasets except for the adult dataset and won the best averaged rank among the interpretable models. Interestingly, top-k rule lists with $k = 5$ performed worse than with $k = 3$. This is because setting k to large value requires large number of rules, but the list size was not able to be large due to the regulation of $\lambda = 3$. To clarify this point and see the effect of λ, Table 5.3 shows the comparison between accuracies with $k = 3$ and those with $k = 5$ for varying λ. For $\lambda = 3$, the accuracies with $k = 5$ were worse than those with $k = 3$ for many datasets, but for $\lambda = 1$, the accuracies with $k = 5$ were almost the same as those with $k = 3$ for most datasets. This result means that large k needs small λ. However, Table 5.3 also shows small λ leads to large number of rules, and for $\lambda = 1$, learned lists had much more than 25 rules. The value of λ can be smaller than 1, but too many rules would harm the interpretability. Considering the trade-off between accuracy and intepretability, we recommend setting $k = 3$ and $\lambda = 3$ so that the rule lists do not have too many rules, like more than 25 rules.

The top-k rule list with $k = 1$ had more rules than CORELS even though both had almost the same accuracy. However, the cost for verifying a single rule differs depending on the models. CORELS is based on the order-sensitive approach, and thus the list is likely to be shorter, but all the preceding rules need to be considered if human users verify a rule. The top-k rule list and ORL are based on the order-insensitive approach, and thus the list is likely

to be longer than CORELS, but the rules can be verified independently from other rules.

In Table 5.2, the bagging ensemble of three top-k rule lists with $k = 1$ had more than twice as many rules but performed slightly worse than the top-k rule list with $k = 3$. This result clarifies how the top-k rule list performs differently from the ensemble of k rule lists. An ensemble of k rule lists may include many redundant rules because lists are trained independently and may share almost the same rules. On the other hand, the top-3 rule list achieved better performance with much fewer rules than bagging because the list of rules is optimized as a whole and efficient in terms of the number of rules.

For top-k rule lists, Steps 1 and 2 took less than a few seconds, and thus the total runtimes were determined by Step 3. The underscored numbers in Table 5.2 represent the cases where the optimal solution was reached in Step 3 before the time limit of 5000s. The optimal solutions could not be reached for several datasets, but top-k rule lists still performed well in terms of accuracy.

Figure 5.1 gives an example of a top-k rule list trained with $k = 3$ for the adult dataset. The goal is to predict whether income is high. When the test instance shown at the top of Figure 5.1 is given, the first three rules whose condition is satisfied by the instance are Rule 1, 2, and 5. If the list was the traditional rule list, only Rule 1 would be used for prediction, and the marital status, the education number, and the capital gain would be used as the reasons for the prediction. However, the top-k rule list with $k = 3$ selects Rule 2 and 5 in addition to Rule 1. Although the three rules all predict that the income is high, Rule 5 gives additional reasons—the occupation and the sex—to back up the prediction. These additional reasons may provide new insight, and if Rule 5 predicted that the income was low, the additional reasons would work as the reasons for objection against Rule 1 and 2. Note that Rule 2 is almost the same as Rule 1 because even an optimal top-k rule list may include redundant rules if the redundancy improves the averaged predictions. In comparison with the

traditional rule list, the top-k rule list may provide more reasons, which may be redundant or contradicting, but also may be insightful, and help improve the accuracy as shown in the experiments.

5.5 Related Work

Rule Lists In early studies, rule lists were optimized in a greedy manner because the exact optimization is computationally hard [70]. Recently, rule lists have regained attention due to the increasing demand for the interpretability of machine learning models, and the exact optimization has been pursued with the help of high-performance modern computers. The studies listed below take the two-step approach: assuming that a rule set is generated by rule mining, a rule list is constructed by selecting and ordering rules from the given rule set. The Bayesian rule list (BRL) [48] is based on the Bayesian Markov chain Monte Carlo approach and the rule list with the maximum probability is selected from sampled rule lists. The scalable Bayesian rule list [86] improved BRL in scalability by using bit-vector operations. A continuously relaxed version of the rule list [59] was also proposed to improve scalability. ORL [72] is based on mathematical programming. Its advantage is the flexibility for changing the objective function or adding specific constraints, but it is so slow that it cannot be applied to large datasets. In order to make the learning more efficient, CORELS [5] was proposed in a follow-up paper. Instead of using a general mathematical programming solver, CORELS utilizes a branch-and-bound solver customized for learning rule lists to minimize the same objective. Since all the rule list algorithms mentioned above are based on the common premise of the traditional rule list that only one rule is used for prediction, they cannot be directly applied to the case with $k > 1$.

Associative Classification An associative classification approach called CPAR [87] is based

on a set of rules, and the rules are ordered by confidence scores. In training, a set of rules are extracted for each class by association rule mining. When predicting the class, it chooses the top k rules with the highest confidence for each class and then chooses the class with the highest confidence averaged over the k rules. Unlike the top-k rule list, CPAR cannot be viewed as a generalization of the rule list because it chooses k rules for each class. Moreover, it is based on a greedy algorithm and does not provide means to optimize the accuracy or the rule set size, whereas our approach optimizes the trade-off between the accuracy and the rule list size with the adjustable parameter λ.

Rule Ensembles Although tree ensemble models such as random forests [12] have high accuracy, they are hard to interpret because of the large number of trees required. Rule ensemble models such as RuleFit [25] were proposed to improve the interpretability of tree ensemble models while retaining the accuracy. Given a trained tree ensemble, these methods extract rules from the decision trees in the ensemble and then learn a linear combination of the rules. A prediction is given as a weighted average of the predictions of applied rules. The weights are sparsified so that most rules can be disregarded in interpretation. However, these approaches require additional human effort to carefully interpret the non-zero weights because the sum of apparently small weights may affect the decision in total. In contrast, the top-k rule list proposed in this chapter does not involve such weights and can be interpreted as a direct generalization of the traditional rule list. Moreover, unlike top-k rule lists, rule ensemble models do not limit the maximum number of rules used for one prediction.

5.6 Conclusion and Future Work

We proposed a new model called a top-k rule list. When making a prediction, the top-k rule list takes the ensemble of the first k rules whose conditions are satisfied by the input. This is

a generalization of the traditional rule list, which is a special case with $k = 1$. The advantage of the top-k rule list is that more accurate predictions can be made with larger k at the expense of more rules to interpret. We also proposed an ILP formulation to learn the top-k rule list. The number of variables is efficiently reduced by considering the characteristics of the top-k rule list. Our experiments showed that the top-k rule list with $k > 1$ outperformed the traditional rule list trained by the state-of-the-art learning algorithm for most datasets.

To fully evaluate the interpretability and scalability of the top-k rule list, an interesting future direction is to conduct comprehensive experiments on more diverse and larger datasets, incorporating human evaluation. In particular, since the selection of k and λ can significantly impact interpretability, a strategy to determine these parameters should be developed through human evaluation. Furthermore, it would be valuable to perform a comparative analysis with other interpretable models beyond traditional rule lists, such as decision trees or other rule-based models.

Chapter 6

Solving Weighted Abduction via Max-SAT

In this chapter, we study weighted abduction, a variant of abductive reasoning in which hypotheses are evaluated based on associated costs or weights. In general, the abduction problem is NP-hard, and existing approaches such as Integer Linear Programming (ILP) and Answer Set Programming (ASP) struggle to scale when dealing with large amounts of background knowledge and observations. To address this challenge, we propose an efficient formulation of the depth-limited weighted abduction problem as a Maximum Satisfiability (Max-SAT) problem. Our key insight is that the majority of the resulting hard clauses are Horn, resulting in a formulation that closely resembles the Max Horn SAT problem. This structure enables us to leverage recent advances in Max-SAT solvers. Empirical evaluations on large-scale, real-world instances demonstrate that our approach is up to 100 times faster than existing ILP-based methods, while maintaining equivalent solution quality. These results suggest that SAT-based formulations offer a scalable and effective alternative to ILP for solving weighted abduction problems.

The material and all the figures in this chapter have been reproduced from [73][1] with the permission of AAAI.

6.1 Introduction

We consider *weighted abduction*, which is a cost-based abduction method that selects H having the lowest cost induced by backchaining and unification (see Sec. 6.2). The weighted abduction has been widely used in several tasks [6,10,61] since it is a basic model of abduction and easy to incorporate the uncertainty of the background knowledge and observation [32]. In particular, [61] observed that the weighted abduction has an advantage over alternative approaches when they are applied to the discourse interpretation tasks.

The main difficulty of the weighted abduction (and other models) is its computational cost. In general, the abduction problem is NP-hard [15]; hence, it will be impractical if we have to handle a large amount of background knowledge and observations. To overcome this difficulty, [39] introduced the *depth-limited weighted abduction problem* (see Sec. 6.3) and formulated the weighted abduction as an ILP problem. To accelerate the performance of the ILP approach to the weighted abduction, [40] observed that the most constraints represent the transitivity among equalities of the variables, and proposed a method called cutting-plane inference to improve the performance. [85] proposed a method to prune the redundant candidate to improve the performance. [79] proposed an *answer set programming (ASP)*-based approach for the weighted abduction. Similar to the ILP-based approach of [40], their approach also specifies a parameter that limits the depth of the backchaining, which are incomparable to the depth parameter of [40].

However, even using these methods, the weighted abduction still cannot be applied to

[1] ©2020 AAAI

large-scale instances, which typically contain more than hundreds of items in background knowledge and observations. Our purpose is to establish a more efficient algorithm for the weighted abduction.

In this chapter, we propose an efficient method for the weighted abduction. We make the following contributions.

- We formulate the depth-limited weighted abduction problem as a *maximum satisfiability (Max-SAT) problem* whose majority of the hard clauses are Horn clauses, and the number of soft clauses is small.

- We propose to solve the problem using Max-SAT solvers. Our experiments showed that this approach is 100 times faster than the existing method that uses ILP solver for large-scale real-world instances.

As discussed above, most of the existing methods formulated the problem as an ILP problem to use efficient ILP solvers. However, since abduction is a logic problem, we believe that it is more natural to formulate it as a SAT problem.

Our first contribution shows that our problem is "nearly" Max Horn SAT problem with a small number of soft clauses. For example, in our largest instance, the total number of hard clauses is about 2,000,000, but only 6,000 are non-Horn clauses. Also, the number of soft clauses is 6,000. Since the complexity of the Max Horn SAT is exponential in the number of soft clauses, i.e., not the number of hard clauses, it is beneficial to use our Max Horn-like SAT formulation.

Our second contribution verifies the above analytic contribution via experiments. The Max-SAT and ILP formulation had almost the same number of variables and constraints; however, the former solved the problem much faster than the latter. This verifies that our Max-SAT formulation is much better than the ILP formulation for the weighted abduction

problem. Here, we have to thank the recent progress of Max-SAT solvers [8]. The performance of SAT solvers improves every year, and many high-performance implementations (e.g., OpenWBO [53] and MaxHS [19]) are freely available. This might change the situation around solving abduction problems.

6.2 Preliminaries

We follow the standard notation of first-order logic. We denote by $\vec{x} = (x_1, \ldots, x_k)$ a set of variables with a suitable length, and $p(\vec{x})$ for an atomic formula with variable $\vec{x}$.

6.2.1 Weighted Abduction

Weighted abduction [33] is a method of abduction based on the costs. In weighted abduction, background knowledge B is a set of first-order Horn clauses, where all of the variables on the left-hand side are universally quantified with the widest possible scope, and the variables only on the right-hand side are existentially quantified. Each Horn clause in the background knowledge is referred to as a *rule*. Each atomic formula $p(\vec{x})$ on the left-hand side of a rule has a positive *weight* w; this is denoted by $p(\vec{x})^w$. Thus, in general, a rule is represented as

$$\forall \vec{x} \exists \vec{y}\, (p_1(\vec{x}_1)^{w_1} \wedge \cdots \wedge p_k(\vec{x}_k)^{w_k} \Rightarrow q(\vec{x}_0, \vec{y})), \tag{6.1}$$

where $\vec{x}$ and $\vec{y}$ denote (possibly empty) sets of variables and $\vec{x}_0, \ldots, \vec{x}_k$ are subsets of $\vec{x}$. Observation O is a conjunction of existentially quantified atomic formulas $p(\vec{x})$, each of which has a positive cost c; this is denoted by $p(\vec{x})^{\$c}$. Thus, in general, an observation is represented as

$$\exists \vec{x}\, (p_1(\vec{x}_1)^{\$c_1} \wedge \cdots \wedge p_k(\vec{x}_k)^{\$c_k}). \tag{6.2}$$

A candidate hypothesis H is also in the same format as the observation. The cost of H is the sum of all costs of atomic formulas in the hypothesis, i.e.,

$$c(H) = \sum_{p(\vec{x})^w \in H} w, \tag{6.3}$$

where $p(\vec{x})^w \in H$ means that $p(\vec{x})$ appears in H.

A candidate hypothesis H is generated as follows. Initially, we set $H := O$. Then, we iteratively apply *backchaining* and *unification* in arbitrary order.

Backchaining We select an atomic formula $q(\vec{x}_0, \vec{y})^{\$c} \in H$ and a rule $\forall \vec{x}' \exists \vec{y}' \, (p_1(\vec{x}'_1)^{w_1} \wedge \cdots \wedge p_k(\vec{x}'_k)^{w_k} \Rightarrow q(\vec{x}'_0, \vec{y}')) \in B$. Then, we replace H by removing $q(\vec{x}_0, \vec{y})^{\$c}$ and adding $p(\vec{x}_1)^{w_1 \$c} \wedge \cdots \wedge p(\vec{x}_1)^{w_k \$c}$. For example, if $H = \exists x_1, x_2 \, (p(x_1, x_2)^{\$2} \wedge q(x_2)^{\$1})$ and $\forall x'_1, x'_2, \exists y \, (r(x'_1)^{\$3} \wedge s(x'_2)^{\$4} \Rightarrow p(x'_1, y)) \in B$ then we obtain $H' = \exists x_1, x_2, y \, (r(x_1)^{\$6} \wedge s(x_2)^{\$8} \wedge q(y)^{\$1})$.

Unification We select sets $\vec{x}, \vec{y}$ of variables of the same size in H. Then, we replace H by letting $\vec{x} = \vec{y}$. This is allowed only if all of the atomic formula containing $\vec{x}_j$ and $\vec{y}_j$ are the same up to the variables. The cost of the resulting atomic formulas are the smallest costs of the original atomic formulas. For example, if $H = \exists x, y \, (p(x)^{\$2} \wedge p(y)^{\$3} \wedge q(x)^{\$5} \wedge q(y)^{\$4})$, we have $H' = \exists x \, (p(x)^{\$2} \wedge q(x)^{\$4})$.

A different inference process gives a different hypothesis. The goal of weighted abduction is to find a hypothesis that has the lowest cost.

6.3 Weighted Abduction via Max-SAT

Here, we formulate the problem as a Max-SAT problem whose majority of hard clauses are Horn clauses.

A naive approach is searching hypothesis using backchaining and unification from the observation. However, this approach is impractical because of the combinatorial explosion. To overcome this difficulty, [39] introduced the *depth-limited weighted abduction problem*, which restricts the depth of backchaining at most d, where d is a positive integer specified as a parameter. This formulation has advantages in both application and computation. In a practical application, we usually hope explanations with shallow backchaining because an explanation with deep backchaining may look like the "butterfly effect." Therefore, it is reasonable to restrict the depth of backchaining. In computation, this removes the combinatorial explosion due to backchaining.

6.3.1 Backchaining Graph

A *backchaining graph of depth* d is a node-weighted directed hypergraph[2] $G = (P, R) \equiv (P^{(i)}, R^{(i)})_{i=0,\ldots,d}$ such that each $\alpha \in P$ corresponds to an atomic formula $p(\vec{x})$ (with different variable names), and each $e \in R$ is associated with a rule in the background knowledge. It is constructed as follows. First, we define $P^{(0)} = \{\alpha_1, \ldots, \alpha_k\}$ for observation $O = \exists\vec{x}\,(p_1(\vec{x}_1)^{\$c_1} \wedge \cdots \wedge p_k(\vec{x}_k)^{\$c_k})$, where $\alpha_1, \ldots, \alpha_k$ are associated with $p_1(\vec{x}_1), \ldots, p_k(\vec{x}_k)$, respectively, and the weights are defined by $w(\alpha_1) = c_1$, ..., $w(\alpha_k) = c_k$. We define $R^{(0)} = \emptyset$. Then, we perform the following procedure for $i = 1, \ldots, d$. We initialize $P^{(i)} = P^{(i-1)}$ and $R^{(i)} = R^{(i-1)}$. For each rule $\forall\vec{x}'\exists\vec{y}'\,(p_1(\vec{x}'_1)^{w_1} \wedge \cdots \wedge p_k(\vec{x}'_k)^{w_k} \Rightarrow q(\vec{x}'_0, \vec{y}')) \in B$ and node $\beta \in P^{(i-1)}$ that is associated with $q(\vec{x}_0, \vec{y})$ (i.e., having the same predicate name), we create new nodes $\beta_1, \ldots, \beta_k$, which are associated with $\bar{p}_1(\vec{x}'_1), \ldots, \bar{p}_k(\vec{x}'_k)$, respectively, and the weights are defined by $w(\beta_1) = w_1 w(\beta)$, ..., $w(\beta_k) = w_k w(\beta)$. Here, we introduce new variables $\vec{x}'$ to distinguish them from existing variables. We add these new nodes to $P^{(i)}$ and

[2] A *directed hypergraph* (P, R) is a pair of finite sets P and $R \subseteq 2^P \times P$. Each hyperedge $e \in R$ is denoted by $e = (S, t)$, where $S \subseteq V$ and $t \in V$.

add new hyperedge $(\{\beta_1, \ldots, \beta_k\}, \beta)$ to $R^{(i)}$. We refer to the nodes in P as *latent atomic formula*. The backchaining graph is a hyperforest since each application of backchaining creates new nodes.

The above construction generates all atomic formulas using the applications of backchaining with depth at most d. [85] proposed an A^*-search-based heuristics to construct a smaller graph by pruning atomic formulas that do not constitute the optimal hypothesis.

6.3.2 Max-SAT Formulation

After constructing the backchaining graph, our remaining tasks are to determine (1) which atomic formulas are included in the best hypothesis, and (2) which variables are unified. We solve them by reducing to a Max-SAT problem.

Let V be the set of variables in the atomic formulas corresponding to P, and C be the set of constants in them. Our encoding to Max-SAT is based on [39]. We introduce four types of Boolean variables: h_α, r_α, $u_{\alpha,\beta}$ for all $\alpha \in P$, and $\beta \in P$ associated with the same atomic formula (i.e., with a different variable name) with that of α, and $s_{x,y}$ for all variables or constants $x, y \in V \cup C$. In the following, we design formulas to enforce these variables representing the following events.

- $h_\alpha = \texttt{True}$ if $p(\vec{x})$ corresponding to α can be deduced from the hypothesis (i.e., it is explained by others or included in hypothesis).
- $r_\alpha = \texttt{True}$ if $p(\vec{x})$ corresponding to α does not have to pay the cost even if $p(\vec{x})$ is explained (i.e., it is explained or unified with another atomic formula).
- $u_{\alpha,\beta} = \texttt{True}$ if $p(\vec{x})$ corresponding to α and $p(\vec{y})$ corresponding to β are unified.
- $s_{x,y} = \texttt{True}$ if the variables or constants x and y are equal. We define $s_{x,y} = \texttt{False}$ for all distinct constants.

Then, we can see that $h_\alpha \wedge \neg r_\alpha$ implies that α is included in the solution hypothesis. Note that we use this representation because most of the constraints use h_α instead of $h_\alpha \wedge \neg r_\alpha$; hence, it reduces the lengths of the formulas.

Objective Function (Soft Clauses) The objective of weighted abduction is to find a set of latent atomic formulas such that the total cost is the smallest. By the encoding, α pays a cost if $h_\alpha \wedge \neg r_\alpha$ holds. Thus, we set the negations of these formulas to the soft clauses as

$$h_\alpha \Rightarrow r_\alpha; \;\; \text{weight} = w(\alpha), \quad \alpha \in P. \tag{6.4}$$

By the construction, if an assignment falsifies this clause, the assignment contains α in the solution hypothesis; therefore, it pays the cost $w(\alpha)$. These are Horn clauses.

Constraint 1: Observation must be true We require that the latent observation must be true. Thus, we add

$$h_\alpha, \quad \alpha \in \bar{O} \tag{6.5}$$

to the formula. These are Horn clauses.

Constraint 2: Unification is only performed for explained atomic formulas Two latent atomic formulas can be unified only if both are explained or included in hypothesized. So, we add the following constraint, which are Horn clauses.

$$u_{\alpha,\beta} \Rightarrow h_\alpha, \quad u_{\alpha,\beta} \Rightarrow h_\beta, \quad \alpha, \beta \in P \tag{6.6}$$

Constraint 3: Unified atomic formulas do not have to pay the costs A latent atomic formula α does not have to pay the cost only if it is explained or unified. To represent this

constraint compactly, we first add the constraint:

$$(\bigwedge_{\beta\in S} h_\beta) \vee (\bigwedge_{\beta\in S} \neg h_\beta), \quad (S,t) \in R \text{ for } t \in P. \tag{6.7}$$

This means that all the atomic formulas that are children of $\beta \in P$ have the same value. Since the generated hypergraph is a hyperforest, we can safely add this constraint. (Note that two latent atomic formulas associated with the same atomic formula can have different values). Since these are not Horn clauses, we convert them using the Tseitin transformation [82]. For each S, we introduce a new variable X_S and add the clause $\left(\bigwedge_{\beta\in S} h_\beta \Rightarrow X_S\right) \wedge \left(X_S \Rightarrow \bigwedge_{\beta\in S} h_\beta\right)$ to the formula. Then, $X_S \equiv \bigwedge_{\beta\in S} h_\beta \Rightarrow X_S$. The first conjunction is a Horn clause, and the second conjunction is a conjunction of a Horn clause, i.e., $X_S \Rightarrow \bigwedge_{\beta\in S} h_\beta$ is equivalent to $X_S \Rightarrow h_\beta$ for all $\beta \in S$. For the above event, the original clause is equivalent to $\bigwedge_{\beta\in S} (h_\beta \Rightarrow X_S)$. Therefore, (6.7) is represented by Horn clauses.

Under this constraint, the cost condition is represented by

$$\neg r_\alpha \vee \left(\bigvee_{\beta\in I(\alpha)} h_\beta\right) \vee \left(\bigvee_{\beta\in \text{small}(\alpha)} u_{\alpha,\beta}\right) \tag{6.8}$$

where $I(\alpha) = \{\beta \in P : (S,\alpha) \in R, \beta \in S\}$ is the set of the in-neighborhood of α, which are the nodes that explain α, and $\text{small}(\alpha) = \{\beta \in P : c(\beta) \leq c(\alpha)\}$ is the nodes having a smaller cost than β. Here, we introduce arbitrary tie-breaking to make all of the costs have different values. This cannot be converted into a Horn clause if we use any transformation. So, (6.8) is not represented by Horn clauses.

Constraint 4: Variable transitivity If $x = y$ and $y = z$ then $x = z$. This is represented as follows:

$$\begin{aligned} &s_{x,y} \wedge s_{y,z} \Rightarrow s_{x,z}, \quad s_{x,z} \wedge s_{y,z} \Rightarrow s_{x,y}, \\ &s_{x,y} \wedge s_{x,z} \Rightarrow s_{y,z}, \quad x,y,z \in V \cup C. \end{aligned} \tag{6.9}$$

These are Horn clauses.

Constraint 5: Valid substitution If we unify two latent atomic formulas p and q, all corresponding variables must take the same value. This condition is represented by

$$u_{p,q} \Rightarrow s_{x,y}, \quad p, q \in P, (x, y) \in \text{usub}(p, q), \tag{6.10}$$

where usub(p, q) is the matching pairs of the variables of p and q that can be unified. For example, given atomic formula $p'(x_1, x_2, x_3)$ and $q'(y_1, y_2, y_3)$ corresponding nodes $p \in P$ and $q \in P$, respectively, we have usub$(p, q) = \{(x_1, y_1), (x_2, y_2), (x_3, y_3)\}$. These are Horn clauses.

Now all the constraints of the depth-limited weighted abduction problem are encoded as an instance of the Max-SAT problem. We analyze the instance. First, we see that most of the hard clauses come from Constraint 4 (variable transitivity), which are Horn clauses. The non-Horn clauses only come from Constraint 3, whose number is at most the number of the nodes in the backchaining graph, which are much smaller than the total number of the hard clauses. Next, we see that the number of soft clauses is equal to the number of vertices of the backchaining graph. Thanks to the A^*-search technique mentioned in the last of Section 6.3.1, the size of the backchaining graph is not too large. By summarizing the above discussion, we obtain the following.

Claim 2. *A depth-limited weighted abduction problem is reduced to a Max-SAT problem with a small number of soft clauses and a small number of non-Horn clauses.*

As mentioned in Section 6.2, modern Max-SAT solvers efficiently solve Max Horn-SAT problems. We expect these also solves our problem efficiently.

6.4 Experiments

We conducted the following experiments to evaluate our proposed method. We refer to our method as *SAT-WA*.

All experiments were conducted on a computer (Intel Xeon E3-1270 V2 (4C/3.50GHz/8M) with 32GB of memory). The code of our method was implemented in C++ (g++ [version4.8.5] with the -O2 option). We implemented our algorithm (*SAT-WA*) using Open-WBO[3] and MaxHS[4], which are particularly efficient Max-SAT solvers. For comparison, we employed David[5], which is a successor to Phillip [85], with the Gurobi optimizer[6] and ASP-based method (ASP-WA)[7] [79] using Gringo[8] for grounding and Wasp[9] for solving. Note that ASP-WA can be used for datasets that do not contain cycles because the method does not support depth-limitation. As a result, we compared ASP-WA with other method only for the ACCEL dataset without the depth limitation (see below).

6.4.1 Datasets

Fraud

We created a new dataset for a real-world event detection problem. The task is to seek the possibility of "fraud" from emails[10]. The background knowledge consists of rules such as $\mathit{competitor}(x, y) \wedge \mathit{provide}(\mathit{PRICE\_INFO}, x, y) \Rightarrow \mathit{cartel}(x, y)$, which means that "*if* x *and*

[3]http://sat.inesc-id.pt/open-wbo/

[4]http://www.maxhs.org/docs/papers.html

[5]https://github.com/aurtg/open-david

[6]http://www.gurobi.com/

[7]https://bitbucket.org/knowlp/asp-fo-abduction

[8]http://potassco.sourceforge.net/

[9]https://github.com/alviano/wasp/

[10]The emails are manually and artificially created based on the actual incidents. Thus, there is no issue in law, privacy, and ethics.

y are competitors and x provides y the price information, then x and y form a cartel." We created 215 rules by hand. The observation is a set of events that are extracted from emails. We selected 10 observations that are extracted using a lexical analysis. Each observation has 331.9 atomic formulas on average. By performing weighted abduction for this instance, we may obtain a possibility of fraud (e.g., cartel) that can explain the observation. We assign the uniform weights to the rules so that the total weight equals 1.2. We assign a cost of 10 for each observation. This dataset contains cycles; thus, ASP-WA method cannot be applied for this dataset.

ACCEL

The ACCEL dataset is the dataset used in [79]. It is extracted from the dataset originally developed for the abductive plan recognition system ACCEL [58] and one rule that makes the background knowledge cycle is removed from it.

The background knowledge and observations of this dataset were obtained by extracting 189 background axioms and 50 plan recognition problems, respectively. The plan recognition problems provide agents' partial actions as a conjunction of atomic formulas. The number of rules in the background knowledge is 189 and the average number of atomic formulas for 50 observations is 12.58. This contains no cycles. Thus, we can apply ASP-WA for this dataset.

6.4.2 Evaluation

We measured the number of solved problems in 1000 seconds and the average running times of David and SAT-WA on the Fraud and ACCEL datasets with the depth of 1, 3, and 5. We also measured the average running time of ASP-WA and the above methods on ACCEL with the depth of ∞. The average is taken over the instances in the datasets (i.e., 50 for ACCEL

and 10 for Fraud). The running time is separated as the following three parts: i) Graph: the running time for generating the backchaining graph, ii) Convert: the running time for converting to the ILP or Max Horn-SAT problem and iii) Solve: the running time for finding an optimal hypothesis by the solvers. SAT-WA and David performed the same procedure for Graph part and almost the same for Convert part. The difference will appear in Solve part.

6.4.3 Results

Tables 6.1a and 6.2a present the running times for the Fraud and ACCEL datasets that could be solved in 1000 seconds, respectively. For all cases, SAT-WA outperformed David significantly. For the Fraud dataset with $d = 5$, the all of the problems were solved by SAT-WAs with all SAT solvers, while the 30% (3/10) of the problems were not solved by David with ILP solver. For the ACCEL dataset with $d = 5$, all problems were solved by SAT-WAs with all SAT solvers, while the 48% (24/50) of the problem could not be solved by David with ILP solver. This implies that SAT-WAs were much efficient than David, and they efficiently worked for reasonable depths. Besides, for depth $= \infty$ SAT-WA with OpenWBO efficiently ran as well as ASP-WA.

To understand the reason why SAT-WA was so efficient, we looked into the details of the ILP and Max-SAT problems. Tables 6.1b and 6.2b show the size of these problems converted from the weighted abduction problem. As expected by Claim 2, the number of soft clauses, which equals to the number of nodes of the backchaining graph, is small compared with the other parameters. Hence, we can expect that the SAT solver can solve the problem efficiently.

6.5 Conclusion

We proposed a Max-SAT formulation of the weighted abduction problem and proposed to solve the problem using Max-SAT solvers. The Max-SAT problems obtained by the formulation have a small number of soft clauses compared with the hard clauses, and the hard clauses are mostly Horn clauses. This indicates that the problems can be solved efficiently using modern SAT solvers. Experimental results supports this observation. Our method outperformed the existing methods by a factor of 100 for real-world instances.

There are several possible directions. One promising direction is to develop a tailored solver, instead of general Max-SAT solvers. Our Max-SAT problems have many Horn clauses; thus, a progress on the Max Horn-SAT solvers can be incorporated. Another promising direction is to learn the parameters (i.e., the cost and weight) for the weighted abduction problem from dataset. A learning process may require to solve the weighted abduction problem several times; hence, our efficient method will help such the process.

(a) Running time [sec] varying depth value $d = 1, 3, 5$.

d		Graph	Convert	Solve	Total (ratio of solved)
1	David_gurobi	0.067	0.002	0.008	0.077 (10/10)
	SAT-WA_openwbo	0.065	0.002	0.006	0.074 (10/10)
	SAT-WA_maxhs	0.064	0.002	0.006	0.073 (10/10)
3	David_gurobi	5.13	0.053	362.75	367.93 (7/10)
	SAT-WA_openwbo	4.71	0.051	0.176	4.94 (10/10)
	SAT-WA_maxhs	4.51	0.051	77.02	81.59 (7/10)
5	David_gurobi	8.72	0.110	468.06	476.89 (7/10)
	SAT-WA_openwbo	8.25	0.107	0.305	8.66 (10/10)
	SAT-WA_maxhs	8.12	0.107	77.81	86.04 (10/10)

(b) The size of ILP and Max SAT problem

	$d = 1$	$d = 3$	$d = 5$
#nodes of graph	564.9	2983.3	4385.5
#valuables	797.9	15372.5	33167.5
#constraints	994.7	26500.2	63285.6

Table 6.1: Comparison of David and SAT-WA for Fraud. All values are the averages over the instances.

(a) Running time [sec] varying depth value $d = 1, 3, 5, \infty$.

d		Graph	Convert	Solve	Total (ratio of solved)
1	David_gurobi	0.006	0.01	6.167	6.185 (50/50)
	SAT-WA_openwbo	0.01	0.01	0.031	0.056 (50/50)
	SAT-WA_maxhs	0.008	0.01	0.115	0.137 (50/50)
3	David_gurobi	0.48	0.16	159	159.6 (27/50)
	SAT-WA_openwbo	2.78	1.81	6.303	10.89(50/50)
	SAT-WA_maxhs	2.68	1.81	53.60	58.10(50/50)
5	David_gurobi	0.74	0.23	262.8	263.8 (26/50)
	SAT-WA_openwbo	4.24	2.41	9.05	15.70(50/50)
	SAT-WA_maxhs	3.92	2.43	78.19	84.54(50/50)
∞	David_gurobi	0.68	0.23	263.9	264.9 (26/50)
	SAT-WA_openwbo	4.39	2.46	9.22	16.08(50/50)
	SAT-WA_maxhs	4.08	2.46	79.65	86.20(50/50)
	ASP-WA	—	—	—	14.71(50/50)

(b) The size of ILP and Max SAT problem

	$d = 1$	$d = 3$	$d = 5$	$d = \infty$
#nodes of graph	141.38	4853.42	5917.84	5917.84
#valuables	3575.66	540853	723639	723639
#constraints	7830.48	1248961	1672480	1672480

Table 6.2: Comparison of David and SAT-WA for ACCEL. All values are the averages over the instances.

Chapter 7

Conclusion and Future Work

In this thesis, we studied on rule-based XAI from the perspectives of both data-driven AI and symbolic AI.

In Chapter 4, we introduced a new machine learning model, called the Alternative Rule Model (ARM), which selects a rule from a ruleset that provides a prediction close to that of any black-box model based on the input instance. Unlike the RCN, which is not designed to work with arbitrary black-box models, ARM can be applied to work with any black-box model. This model outputs the prediction from the selected rule, allowing humans to verify the correctness of the rule rather than the black-box prediction itself. We also proposed an algorithm to find a small, high-coverage ruleset for the ARM. The ruleset should have enough coverage to explain many inputs while remaining small enough for human verification. We formalized this problem as a bilevel optimization and solved it by reducing it to a weighted MaxSAT problem. Our approach enables efficient identification of a small ruleset that can support black-box predictions and be checked by humans. Unlike purely rule-based models like decision trees, the ARM has the potential for better predictive performance by selecting rules that are close to the black-box model's predictions.

In Chapter 5, we presented a pure rule-based model called the top-k rule list, an ordered list of rules similar to traditional rule lists, but differing in that it selects not just the first rule but the first k rules whose conditions are satisfied by the input, and then makes a prediction by taking an ensemble of these k rules. The top-k rule list generalizes the traditional rule list, which can be seen as a special case in which $k = 1$. The advantage of the top-k rule list is that varying k provides different trade-offs between accuracy and interpretability. Experimental results demonstrated that, for most datasets, the top-k rule list generated by our ILP formulation with $k > 1$ achieved higher accuracy than the standard rule list.

In Chapter 6, we considered weighted abduction, which is a commonly used mathematical model for abduction. The main difficulty associated with applying weighted abduction to real problems is its computational complexity. To tackle this difficulty, we proposed an efficient method for weighted abduction by formulating the depth-limited weighted abduction problem as a weighted MaxSAT problem, where most of the hard clauses are Horn clauses and the number of soft clauses is small. Empirical evaluations on large-scale, real-world instances demonstrated that our approach was up to 100 times faster than existing ILP-based methods, while maintaining equivalent solution quality. These results suggest that SAT-based formulations offer a scalable and effective alternative to ILP for solving weighted abduction problems.

Through this thesis, we contributed to the first step of developing a framework that can generate rule-based explanations linking users' existing knowledge with novel insights derived from the data or black-box models, thereby enabling users to understand and trust the results more effectively. For future work, it is of interest to apply our method to real-world problems and to identify the domains in which it is most suitable. Another direction is to revise rule-based explanations in the context of long-term machine learning operation. Especially in the context of XAI, understanding how the rules constituting a rule-based model

have changed before and after updates is crucial for users who wish to understand the model. As a first step toward tackling this problem, we have been considering a general framework for updating predictive models while taking into account the compatibility of model representations, and, as an example of such a model, an update algorithm for simple rule lists [77]. It is worth extending this approach to the other rule-based models presented in this thesis, ARM and top-k rule lists.

Bibliography

[1] U. Aïvodji, H. Arai, S. Gambs, and S. Hara. Characterizing the risk of fairwashing. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021*, pages 14822–14834, 2021.

[2] M. Al-Shedivat, A. Dubey, and E. P. Xing. Contextual explanation networks. *CoRR*, abs/1705.10301, 2017.

[3] T. Anderson, D. Schum, and W. Twining. *Analysis of evidence*. Cambridge University Press, 2005.

[4] E. Angelino, N. Larus-Stone, D. Alabi, M. Seltzer, and C. Rudin. Learning certifiably optimal rule lists. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 35–44. ACM, 2017.

[5] E. Angelino, N. Larus-Stone, D. Alabi, M. I. Seltzer, and C. Rudin. Learning certifiably optimal rule lists for categorical data. *J. Mach. Learn. Res.*, 18:234:1–234:78, 2017.

[6] D. E. Appelt and M. E. Pollack. Weighted abduction for plan ascription. *User modeling and user-adapted interaction*, 2(1-2):1–25, 1992.

[7] A. B. Arrieta, N. D. Rodríguez, J. D. Ser, A. Bennetot, S. Tabik, A. Barbado, S. García, S. Gil-Lopez, D. Molina, R. Benjamins, R. Chatila, and F. Herrera. Explainable ar-

tificial intelligence (XAI): concepts, taxonomies, opportunities and challenges toward responsible AI. *Inf. Fusion*, 58:82–115, 2020.

[8] T. Balyo, M. Heule, and M. Järvisalo. *Proceedings of SAT Competition 2017: Solver and Benchmark Descriptions*. University of Helsinki, Department of Computer Science, 2017.

[9] G. Biradar, Y. Izza, E. A. Lobo, V. Viswanathan, and Y. Zick. Axiomatic aggregations of abductive explanations. In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024*, pages 11096–11104. AAAI Press, 2024.

[10] J. Blythe, J. R. Hobbs, P. Domingos, R. J. Kate, and R. J. Mooney. Implementing weighted abduction in markov logic. In *Proceedings of the 9th International Conference on Computational Semantics*, pages 55–64, 2011.

[11] R. J. Brachman and H. J. Levesque. Competence in knowledge representation. In *Proceedings of the National Conference on Artificial Intelligence*, pages 189–192. AAAI Press, 1982.

[12] L. Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, 2001.

[13] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth, 1984.

[14] L. Breiman and N. Shang. Born again trees. Technical report, University of California, Berkeley, 1996.

[15] T. Bylander, D. Allemang, M. C. Tanner, and J. R. Josephson. The computational complexity of abduction. *Artificial intelligence*, 49(1-3):25–60, 1991.

[16] L. Chazette, W. Brunotte, and T. Speith. Exploring explainability: a definition, a model, and a knowledge catalogue. In *2021 IEEE 29th international requirements engineering conference (RE)*, pages 197–208. IEEE, 2021.

[17] Y. Chen, E. Keogh, B. Hu, N. Begum, A. Bagnall, A. Mueen, and G. Batista. The ucr time series classification archive, July 2015. www.cs.ucr.edu/~eamonn/time_series_data/.

[18] M. W. Craven and J. W. Shavlik. Extracting tree-structured representations of trained networks. In *Proceedings of the 8th International Conference on Neural Information Processing Systems*, page 24–30, 1995.

[19] J. Davies and F. Bacchus. Solving MAXSAT by solving a sequence of simpler SAT instances. In J. H. Lee, editor, *Principles and Practice of Constraint Programming - CP 2011 - 17th International Conference*, volume 6876 of *Lecture Notes in Computer Science*, pages 225–239. Springer, 2011.

[20] J. Delaunay, L. Galárraga, and C. Largouët. Improving anchor-based explanations. In *CIKM '20: The 29th ACM International Conference on Information and Knowledge Management*, pages 3269–3272. ACM, 2020.

[21] H. Deng. Interpreting tree ensembles with intrees. *Int. J. Data Sci. Anal.*, 7(4):277–287, 2019.

[22] F. Doshi-Velez and B. Kim. Towards a rigorous science of interpretable machine learning, 2017.

[23] W. F. Dowling and J. H. Gallier. Linear-time algorithms for testing the satisfiability of propositional horn formulae. *J. Log. Program.*, 1(3):267–284, 1984.

[24] D. Dua and C. Graff. UCI machine learning repository, 2017.

[25] J. H. Friedman and B. E. Popescu. Predictive learning via rule ensembles. *The Annals of Applied Statistics*, 2(3):916 – 954, 2008.

[26] C. C. Green. Application of theorem proving to problem solving. In *Proceedings of the 1st International Joint Conference on Artificial Intelligence*, pages 219–240. William Kaufmann, 1969.

[27] C. C. Green. *The Application of Theorem Proving to Question-Answering Systems*. Outstanding Dissertations in the Computer Sciences. Garland Publishing, New York, 1969.

[28] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi. A survey of methods for explaining black box models. *ACM Comput. Surv.*, 51(5):93:1–93:42, 2019.

[29] P. Hansen, B. Jaumard, and G. Savard. New branch-and-bound rules for linear bilevel programming. *SIAM J. Sci. Comput.*, 13(5):1194–1217, 1992.

[30] S. Hara and K. Hayashi. Making tree ensembles interpretable: A bayesian model selection approach. In *International Conference on Artificial Intelligence and Statistics, AISTATS 2018*.

[31] T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, 2nd Edition*. Springer Series in Statistics. Springer, 2009.

[32] J. R. Hobbs. Abduction in natural language understanding. *Handbook of pragmatics*, pages 724–741, 2004.

[33] J. R. Hobbs, M. E. Stickel, D. E. Appelt, and P. Martin. Interpretation as abduction. *Artificial intelligence*, 63(1-2):69–142, 1993.

[34] A. Ignatiev, Y. Izza, P. J. Stuckey, and J. Marques-Silva. Using maxsat for efficient explanations of tree ensembles. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022*, pages 3776–3785. AAAI Press, 2022.

[35] A. Ignatiev, A. Morgado, and J. Marques-Silva. Propositional abduction with implicit hitting sets. In *ECAI 2016 - 22nd European Conference on Artificial Intelligence - Including Prestigious Applications of Artificial Intelligence (PAIS 2016)*, volume 285 of *Frontiers in Artificial Intelligence and Applications*, pages 1327–1335. IOS Press, 2016.

[36] A. Ignatiev, A. Morgado, and J. Marques-Silva. On tackling the limits of resolution in sat solving. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 164–183, 2017.

[37] A. Ignatiev, N. Narodytska, and J. Marques-Silva. Abduction-based explanations for machine learning models. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019*, pages 1511–1519. AAAI Press, 2019.

[38] N. Inoue and A. S. Gordon. A scalable weighted Max-SAT implementation of propositional Etcetera Abduction. In *Proceedings of the International Conference of the Florida AI Society*, pages 62–67, 2017.

[39] N. Inoue and K. Inui. Ilp-based reasoning for weighted abduction. In *Plan, Activity, and Intent Recognition, Papers from the 2011 AAAI Workshop*, 2011.

[40] N. Inoue and K. Inui. Large-scale cost-based abduction in full-fledged first-order predicate logic with cutting plane inference. In *Logics in Artificial Intelligence - 13th European Conference, JELIA 2012, Proceedings*, pages 281–293, 2012.

[41] B. Jaumard and B. Simeone. On the complexity of the maximum satisfiability problem for horn formulas. *Information Processing Letters*, 26(1):1–4, 1987.

[42] J. R. Josephson and S. G. Josephson. *Abductive inference: Computation, philosophy, technology*. Cambridge University Press, 1996.

[43] R. J. Kate and R. J. Mooney. Probabilistic abduction using markov logic networks. In *Proceedings of the IJCAI-09 Workshop on Plan, Activity, and Intent Recognition*, 2009.

[44] P. W. Koh and P. Liang. Understanding black-box predictions via influence functions. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1885–1894. PMLR, 2017.

[45] H. Lakkaraju, S. H. Bach, and J. Leskovec. Interpretable decision sets: A joint framework for description and prediction. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1675–1684. ACM, 2016.

[46] G. Lancia and P. Serafini. *Integer Linear Programming*, pages 43–66. Springer International Publishing, Cham, 2018.

[47] E. Lee. *Optimal Approximabilities beyond CSPs*. PhD thesis, Carnegie Mellon University Pittsburgh, PA, 2017.

[48] B. Letham, C. Rudin, T. H. McCormick, and D. Madigan. Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model. *CoRR*, abs/1511.01644, 2015.

[49] B. Y. Lim, A. K. Dey, and D. Avrahami. *Why and why not* explanations improve the intelligibility of context-aware intelligent systems. In *Proceedings of the 27th International Conference on Human Factors in Computing Systems, CHI 2009*, pages 2119–2128. ACM, 2009.

[50] S. M. Lundberg and S. Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems*, pages 4765–4774, 2017.

[51] D. Macha, M. Kozielski, L. Wróbel, and M. Sikora. Rulexai - A package for rule-based explanations of machine learning model. *SoftwareX*, 20:101209, 2022.

[52] J. Marques-Silva, A. Ignatiev, and A. Morgado. Horn maximum satisfiability: Reductions, algorithms and applications. In *Portuguese Conference on Artificial Intelligence*, pages 681–694. Springer, 2017.

[53] R. Martins, V. M. Manquinho, and I. Lynce. Open-wbo: A modular maxsat solver,. In *Theory and Applications of Satisfiability Testing - SAT 2014 - 17th International Conference*, volume 8561 of *Lecture Notes in Computer Science*, pages 438–445. Springer, 2014.

[54] P. McCullagh and J. A. Nelder. *Generalized Linear Models*. Springer, 1989.

[55] N. Meinshausen. Node harvest. *Ann. Appl. Stat.*, 4(4):2049–2072, Dec. 2010.

[56] J. Mylopoulos and H. J. Levesque. An overview of knowledge representation. In B. Neumann, editor, *GWAI-83, 7th German Workshop on Artificial Intelligence*, volume 76 of *Informatik-Fachberichte*, pages 143–157. Springer, 1983.

[57] A. Newell and H. A. Simon. Computer science as empirical inquiry: Symbols and search. *Commun. ACM*, 19(3):113–126, 1976.

[58] H. T. Ng and R. J. Mooney. Abductive plan recognition and diagnosis: A comprehensive empirical evaluation. In *Proceedings of the 3rd International Conference on Principles of Knowledge Representation and Reasoning*, pages 499–508, 1992.

[59] Y. Okajima and K. Sadamasa. Decision list optimization based on continuous relaxation. In *Proceedings of the 2019 SIAM International Conference on Data Mining*, pages 315–323, 2019.

[60] Y. Okajima and K. Sadamasa. Deep neural networks constrained by decision rules. In *The Thirty-Third AAAI Conference on Artificial Intelligence*, pages 2496–2505, 2019.

[61] E. Ovchinnikova, N. Montazeri, T. Alexandrov, J. R. Hobbs, M. C. McCord, and R. Mulkar-Mehta. Abductive reasoning with a large knowledge base for discourse processing. In *Computing Meaning*, pages 107–127, 2014.

[62] A. M. Özbayoglu, M. U. Gudelek, and O. B. Sezer. Deep learning for financial applications : A survey. *Appl. Soft Comput.*, 93:106384, 2020.

[63] D. Pan, T. Wang, and S. Hara. Interpretable companions for black-box models. In *The 23rd International Conference on Artificial Intelligence and Statistics*, volume 108, pages 2444–2454, 2020.

[64] C. S. Peirce. A theory of probable inference. *The Johns Hopkins Studies in Logic*, pages 126–181, 1883.

[65] D. Poole. Probabilistic horn abduction and bayesian networks. *Artificial intelligence*, 64(1):81–129, 1993.

[66] H. M. Proença and M. van Leeuwen. Interpretable multiclass classification by mdl-based rule lists. *Inf. Sci.*, 512:1372–1393, 2020.

[67] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[68] M. T. Ribeiro, S. Singh, and C. Guestrin. "why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144, 2016.

[69] M. T. Ribeiro, S. Singh, and C. Guestrin. Anchors: High-precision model-agnostic explanations. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pages 1527–1535, 2018.

[70] R. L. Rivest. Learning decision lists. *Mach. Learn.*, 2(3):229–246, 1987.

[71] C. Rudin. Please stop explaining black box models for high stakes decisions. *CoRR*, abs/1811.10154, 2018.

[72] C. Rudin and S. Ertekin. Learning customized and optimized lists of rules with mathematical programming. *Math. Program. Comput.*, 10(4):659–702, 2018.

[73] Y. Sasaki, T. Maehara, T. Akazaki, K. Yamamoto, and K. Sadamasa. Solving weighted abduction via max-sat solvers. In *Proceedings of the Thirty-Third International Florida Artificial Intelligence Research Society Conference*, pages 142–147. AAAI Press, 2020.

[74] Y. Sasaki and Y. Okajima. Alternative ruleset discovery to support black-box model predictions. In *IEEE International Conference on Data Mining*, pages 1312–1317, 2021.

[75] Y. Sasaki and Y. Okajima. Alternative ruleset discovery to support black-box model predictions. *IEICE Trans. Inf. Syst.*, 106(6):1130–1141, 2023.

[76] Y. Sasaki and Y. Okajima. Top-k rule list learning via integer linear programming. In *PRICAI 2024: Trends in Artificial Intelligence - 21st Pacific Rim International Conference on Artificial Intelligence, PRICAI 2024, Proceedings, Part V*, volume 15285 of *Lecture Notes in Computer Science*, pages 16–28. Springer, 2024.

[77] Y. Sasaki, Y. Okajima, and H. Arimura. Revising rulelists via integer linear programming. Technical report, IEICE, vol. 124, no. 321, IBISML2024-45, December 2024. In Japanese.

[78] T. J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the Tenth Annual ACM Symposium on Theory of Computing*, page 216–226, 1978.

[79] P. Schüller. Modeling variations of first-order horn abduction in answer set programming. *Fundamenta Informaticae*, 149(1-2):159–207, 2016.

[80] P. Singla and R. J. Mooney. Abductive markov logic for plan recognition. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence*, 2011.

[81] S. Stumpf, V. Rajaram, L. Li, M. M. Burnett, T. G. Dietterich, E. Sullivan, R. Drummond, and J. L. Herlocker. Toward harnessing user feedback for machine learning. In *Proceedings of the 12th International Conference on Intelligent User Interfaces, IUI 2007*, pages 82–91. ACM, 2007.

[82] G. Tseitin. On the complexity of derivation in propositional calculus. *Studies in constructive mathematics and mathematical logic*, pages 115–125, 1968.

[83] F. Wang and C. Rudin. Falling rule lists. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, 2015.

[84] Z. Wang, W. Yan, and T. Oates. Time series classification from scratch with deep neural networks: A strong baseline. *CoRR*, abs/1611.06455, 2016.

[85] K. Yamamoto, N. Inoue, K. Inui, Y. Arase, and J. Tsujii. Boosting the efficiency of first-order abductive reasoning using pre-estimated relatedness between predicates. *International Journal of Machine Learning and Computing*, 5(2):114–120, 2015.

[86] H. Yang, C. Rudin, and M. Seltzer. Scalable bayesian rule lists. In *Proceedings of the 34th International Conference on Machine Learning*.

[87] X. Yin and J. Han. CPAR: classification based on predictive association rules. In *Proceedings of the Third SIAM International Conference on Data Mining*, pages 331–335. SIAM, 2003.

[88] J. Yu, A. Ignatiev, and P. J. Stuckey. On formal feature attribution and its approximation. *CoRR*, abs/2307.03380, 2023.

[89] J. Yu, A. Ignatiev, P. J. Stuckey, and P. L. Bodic. Computing optimal decision sets with SAT. In *Principles and Practice of Constraint Programming - 26th International Conference*, volume 12333 of *Lecture Notes in Computer Science*, pages 952–970. Springer, 2020.