UNIVERSITY OF CALIFORNIA SAN DIEGO

Learning Accurate and Interpretable Decision Rule Sets from Neural Networks

A thesis submitted in partial satisfaction of the requirements
for the degree Master of Science

in

Computer Science

by

Litao Qiao

Committee in charge:

Professor Bill Lin, Chair
Professor Sanjoy Dasgupta
Professor Yoav Freund

2020

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to acknowledge Professor Bill Lin for his support as the chair of my committee. Through multiple drafts and many long nights, his guidance has proved to be invaluable. I would also like to acknowledge Weijia Wang, without whom my research would have no doubt taken five times as long. He has helped me in immeasurable ways.

Chapters 1 to 6, in full are currently being prepared for submission for publication of the material. Qiao, Litao; Wang, Weijia. The thesis author was the primary investigator and author of this material.

Chapters 1 to 6 are also coauthored with Wang, Weijia. The thesis author was the primary author of these chapters.

ABSTRACT OF THE THESIS

Learning Accurate and Interpretable Decision Rule Sets from Neural Networks

By

Litao Qiao

Master of Science in Computer Science

University of California San Diego, 2020

Professor Bill Lin, Chair

This thesis proposes a novel way to learn a set of the Boolean rules in disjunctive normal form as an interpretable model for binary classifications. We consider the problem of learning an interpretable decision rule set as training a neural network in a specific architecture and converting each neuron in the network into a set of minimal decision rules. This approach can easily find a set of interpretable decision rules that has similarly high predictive performance as a full-precision fully-connected deep neural network, and the method can balance between accuracy and complexity. Moreover, we prove that the set of decision rules derived from each neuron is minimum, unique, and irredundant with respect to the original neuron. Our method is competitive with several other state-of-the-art rule learning algorithms, even with fewer rules and simpler rule conditions

Chapter 1

Introduction

Machine learning has made tremendous advances in recent years, particularly in the area of deep learning based on neural networks. However, the black box nature of neural networks makes it hard for humans to understand the reasoning behind their decision making. This creates a huge barrier for their widespread adoption in the mainstream, especially as machine learning is being considered in many aspects of our society, including healthcare, legal assistance, financial services, and even criminal justice. In highly-consequential applications like medical diagnosis or recidivism analysis, the lack of interpretability in the models or explainability in their decisions makes it difficult to gain public trust for their use [1]. While there have been some effort in developing explainable machine learning methods, these explanation methods are often just simplified (but inaccurate) post-hoc approximations of the original models [2, 3, 4].

On the other hand, a considerable amount of work around interpretable machine learning methods has been based on logical models, like various forms of rule sets or decision trees [5, 6, 7, 8, 9]. They are interpretable by design in the sense that decisions are made by activating some subset of logical rules or some path through a decision tree, where each logical rule in a rule set or each decision condition in a decision tree has a human-understandable interpretation. However, existing methods for generating these interpretable models have lagged behind state-of-the-art neural network approaches in terms of accuracy and generalization.

In this thesis, we pose the following question: can we get the best of both worlds? To answer this question, we propose a novel approach for automatically converting a restricted form of neural network into a set of interpretable decision rules that implements the same input/output behavior. Our approach is applicable to binary classification problems with categorical inputs. In particular, Boolean decision rules are learned by training a restricted two-layer neural network structure with state-of-the-art stochastic gradient descent (SGD) training algorithms. Our proposed two-layer neural network structure, called a OR-of-Neuron (OON) architecture, is designed so that it can be readily mapped to a set of decision rules once trained. Further, we propose to employ L_0 regularization [10] in our training algorithm to achieve rule simplifications through neural net sparsity. Different tradeoffs between accuracy and complexity can be explored via the degree of L_0 regularization. In contrast to some previous methods for generating decision rules, our approach does not require the premining of frequent association rules.

Experimental results show that our method is very competitive with other rule learners in terms of the predictive accuracy on the unseen instances of 5 datasets. When compared with RIPPER [11], a state-of-the-art rule learner, more detailed analysis shows that our method can achieve much better accuracy when comparing models with similar complexities.

Chapter 1, in full is currently being prepared for submission for publication of the material. Qiao, Litao; Wang, Weijia. The thesis author was the primary investigator and author of this material. Chapter 1 is also coauthored with Wang, Weijia. The thesis author was the primary author of this chapter.

Chapter 2

Related Work

As already mentioned, there has been increasing interest in the field of explainable machine learning in recent years. Similar to our work, some of these papers derive explanations as Boolean logic rules in the disjunctive normal form (DNF) or the conjunctive normal form (CNF). In particular, [9, 12] propose to formulate their models as integer programming problems, where the objective loss function is defined as the Hamming loss that measures the training accuracy and they bound the maximum model complexity in the constraint. [9] approximately solves the problem by relaxing it into a linear programming problem and applying the column generation algorithm, whereas the later utilizes various optimization approaches, including linear programming relaxation, block coordinate descent, alternating minimization algorithm, and redundancy aware binarization. In addition, [13] presents a Bayesian framework for learning decision rule sets in which they approximately construct the maximum a posteriori (MAP) estimation using a combination of a couple of cutting-plane methods.

There works are related to our work in the sense that we both derive Boolean rule set. However, these works are concentrating on the training algorithm of the model, whereas our main contribution is to propose a neural network architecture that can precisely mapped to sets of logical rules. Since neural networks are commonly considered to have a remarkable generalization capability, our approach also inherits this generalization capability in the rule sets that we generate. Further, as training methods for neural networks continue to advance, our approach will also benefit from these advances.

Chapter 2, in full is currently being prepared for submission for publication of the material. Qiao, Litao; Wang, Weijia. The thesis author was the primary investigator and author of this material. Chapter 2 is also coauthored with Wang, Weijia. The thesis author was the primary author of this chapter.

Chapter 3

Problem Formulation

In this work, given a binary classification dataset with binarized input features, our goal is to train a classifier in the form of a Boolean logic function in disjunctive normal form (OR-of-ANDs). In particular, in the first level of the function (the logical ANDs), each clause consists of a subset of input features or their negations, which serves as a decision rule. A clause sufficiently gives a positive prediction if all its features are presented in the input vector. In the upper level of the function (the logical OR), a positive final prediction is produced if at least one of the conjunctive clauses is satisfied; otherwise, a negative final prediction is produced.

Mathematically, the training set contains N data samples $(\mathbf{x}_n, y_n)$, where $\mathbf{x}_n$ comprises D binarized features $x_{i,n} \in \{0,1\}$ and $y_n \in \{0,1\}$. We define a literal to be either an input feature x_i or its negation $\bar{x}_i$ and a clause c is a conjunction of k literals where $1 \leq k \leq D$. If a feature of the feature space is not present in the clause c, then we say that feature is a “don't care” in clause c. A clause is constructed to be intuitive by itself due to fact that it can be directly translated to a rule: an input instance matches a clause (rule) if all literals of the clause are present in the input vector, and we denote this as $\mathbf{x} \rightarrow c$. The output rule set C learned from the neural network is thus defined as a set of conjunction clauses $C = \{c_1, c_2, \dots, c_m\}$.

3.1 Threshold Functions

A neuron with binary inputs and full-precision weights can be viewed as a threshold function with a full-precision bias term added to the output side. We start from a simple linear function f of the form:

$$z = f(\mathbf{x}) = \sum_{i=1}^{D} w_i x_i + b_i \tag{1}$$

where w_i is the weight of each feature and b is the bias term. This equation represents the dot product operation used in a single neuron in a neural network. The threshold function, in essence, is a single neuron with a step activation function:

$$A(z) = \begin{cases} 1, \text{if } z \geq 0 \\ 0, \text{otherwise} \end{cases} \tag{2}$$

3.2 Rule Generation from Threshold Functions

We first define some terminologies that describe the properties of clauses.

Definition 1. (Value of Clause) The value of a clause c with respect to a threshold function f with weights $\{w_1, w_2, \dots, w_D\}$ and bias b is equal to the minimum value of $f(\mathbf{x})$ where $\mathbf{x}$ matches c.

$$f(c) = \min\big(f(\mathbf{x})\big) \quad \text{s.t. } \mathbf{x} \to c \tag{3}$$

Since the rules that we want to generate should give a positive label, we are more interested in the clauses that turn on the threshold function. Thus, we name the clauses of function f to be ``positive clauses" if their values are greater or equal to 0.

Definition 2. (Prime Clause) A clause c is a prime clause of a linear function f if and only if c is a positive clause of f and is not a superset of any other positive clause of f.

The set of the prime clauses are the only clauses that we desire because each threshold function has only one unique and irredundant set of prime clauses. The definition of prime clause enables us to represent an arbitrary threshold function as an equivalent set of minimal rules, thus giving a way to convert the function to its most succinct form.

Definition 3. (Max Span) The max span $z_{\max}$ of a linear function f is the maximum output value that f can attain.

$$z_{\max} = \max\big(f(x)\big) \tag{4}$$

Definition 4. (Max Clause) The max clause $c_{\max}$ of a linear function f is the minimum clause whose value is the linear function's max span.

$$c_{\max} = \min(|c|) \quad \text{s.t. } f(c) = z_{\max} \tag{5}$$

By definition, there is only one pair of max span and max clause for a given threshold function. We can use the one-to-one relationship between a threshold function's weights and its max clause to find its max clause and max span pair. Given a threshold function with the set of weights $\{w_1, w_2, \dots, w_D\}$, its max clause is computed by applying a step function to its non-zero weights and its max span is obtained by plugging the max clause into the threshold function.

Next we present the primary theorem in our prime clause generation procedure and several properties that can be derived from the theorem.

Lemma 1. *Given a threshold logic function, all prime clauses contain the max clause.*

Proof: We prove this by contradiction. We assume that a prime clause p exists such that it does not contain the max clause. If $f(p) < 0$, then by definition it is not a prime. Next we consider the case when $f(p) \geq 0$. Since p does not contain the max clause, then there exists at least a literal ℓ in p such that the max clause has its negation version $\bar{\ell}$. Because of the definition of a max clause, replacing ℓ with $\bar{\ell}$ gives a new clause $\tilde{p}$ whose value $f(\tilde{p})$ is larger than $f(p)$ and is thus also ≥ 0. Since both $f(p)$ and $f(\tilde{p})$ are ≥ 0, it follows that the clause $p \setminus \{\ell\}$ is also ≥ 0 and by definition is a subset of p. Thus, p by definition is not a prime. In both cases, we have a contradiction.

Theorem 1. *All primes for a given threshold function correspond to the maximal removal of some subset* S_{max} *of literals from* $z_{\max}$ *such that*

$$\sum_{i \in S_{\max}} |w_i| \leq z_{\max} \tag{6}$$

Proof: We already proved in Lemma 1 that all prime clauses must contain $c_{\max}$, which means a prime clause p corresponds to some removal of literals from $c_{\max}$. According to the definition of prime clause, if a literal can be removed from the clause c without making the c 's value to be negative, then c is not a prime clause. The removal of a literal for x_i from c corresponds to consider both x_i is a feature and x_i as the negation of a feature. When w_i is positive, the maximum decrease to $f(\mathbf{x})$ occurs when we change x_i from 1 to 0, which would reduce $f(\mathbf{x})$ by w_I. Similarly, when w_i is negative, the maximum decrease to $f(\mathbf{x})$ occurs when we change x_i from 0 to 1, which would also reduce $f(\mathbf{x})$ by w_i. For the value of the corresponding clause to remain to be positive, it must be the case that the sum of the absolute weights corresponding to the removed literals is less or equal to the max span. The maximal removal of a subset $S_{\max}$ of literals such that $\sum_{i \in S_{\max}} |w_i| \leq z_{\max}$

means that no more literals can be removed from $c_{\max}$, and therefore the resulting clause is a prime clause.

Corollary 1. *The set of prime clauses for a threshold function is unique.*

Proof: We already proved in Lemma 1 that all prime clauses must contain $c_{\max}$, and we already proved in Theorem 1 that all prime clauses correspond to the maximal removal of some subset $S_{\max}$ of literals from $c_{\max}$. It follows that the set of prime clauses for a threshold function is unique because the set of maximal subsets that satisfies the equation 6 in Theorem 1 is unique.

Corollary 2. *The set of primes for a threshold function is irredundant.*

Proof: We prove this corollary by contradiction. Given a prime clause p for a threshold function, denote by P and $P \setminus \{p\}$ the set of prime clauses for that threshold function and P excluding p, respectively. Assume every point matching p also matches $P \setminus \{p\}$. Then, consider the point $\mathbf{x}$ that matches p and has:

$$\mathbf{x} = \underset{x}{\operatorname{argmin}}\big(f(x)\big) \quad \text{s.t. } \mathbf{x} \rightarrow c \tag{7}$$

In other words, for all features that are don't cares in p, they are present in $\mathbf{x}$ such that $f(\mathbf{x})$ is minimized. According to the assumption, there must exist another prime clause $\tilde{p} \in P \setminus \{p\}$ such that $\mathbf{x}$ matches $\tilde{p}$. Since flipping any feature in $\mathbf{x}$ that is a don't care in p only monotonically increases the weighted sum, all don't-care features of p can also be removed from $\tilde{p}$, and hence p is covered by $\tilde{p}$ ($\tilde{p} \subset p$), which is contradictory to the premise that p is a prime clause.

Theorem 1 leads to the recursive algorithm described in the Algorithm 1, which computes the maximal subsets of weights that could be removed to generate the prime clauses. The algorithm we proposed here is a simple yet effective depth-first tree search over the entire input space. In the worst case, however, our algorithm admittedly will require exponentially large processing time as the input space grows, and we will address the problem of reducing search space in the next section. After we get all the maximal subsets of weights whose absolute sum value is less than the max span, the prime clauses can be easily computed by converting the remaining weights of each found subset to max clause by the one-to-one relationship mentioned above.

Chapter 3, in full is currently being prepared for submission for publication of the material. Qiao, Litao; Wang, Weijia. The thesis author was the primary investigator and author of this material. Chapter 3 is also coauthored with Wang, Weijia. The thesis author was the primary author of this chapter.

Algorithm 1 Generate prime clauses from a threshold function

```
Input: a set of weights $\mathcal{W}$ and its corresponding max span $s$
Output: a set of weight subsets that can be removed to form the prime clauses $\mathcal{R}$

$\mathcal{W}_{abs} \leftarrow$ take absolute value of each $w_i$ in $W$
$\mathcal{W}_{start} \leftarrow$ Sort $\mathcal{W}_{abs}$ in decreasing order
$\mathcal{W}_c \leftarrow \emptyset$
$\mathcal{R} \leftarrow \emptyset$
getPrime($\mathcal{W}_c$, $s$, $\mathcal{W}_{start}$, $R$)
return $\mathcal{R}$

function GETPRIME($\mathcal{W}_c$, $s$, $\mathcal{W}_r$, $R$)
    Input: the current subset of the weights $\mathcal{W}_c$
           current max span $s$
           remaining possible weights to consider $\mathcal{W}_r$
           current set of maximal weight subsets to be removed $R$

    for $w_i$ in $\mathcal{W}_r$ do
        delete $w_i$ from $\mathcal{W}_r$
        if $w_i \leq s$ then
            $\mathcal{W}_c \leftarrow \mathcal{W}_c \cup w_i$
            if $\mathcal{W}_r$ is empty then
                if $\mathcal{W}_c$ is not a subset of any set in $\mathcal{R}$ then
                    $\mathcal{R} \leftarrow \mathcal{R} \cup W_c$
                end if
            else
                getPrime($\mathcal{W}_c$, $s - w_i$, $\mathcal{W}_r$, $R$)
            end if
        else if $\mathcal{W}_r$ is empty then
            if $\mathcal{W}_c$ is not a subset of any set in $\mathcal{R}$ then
                $\mathcal{R} \leftarrow \mathcal{R} \cup W_c$
            end if
        end if
    end for
end function
```

Figure 1: Recursive algorithm to generate primes from threshold function

Chapter 4

Neurons as Threshold Functions

As previously discussed, explainable rules can be effectively extracted from threshold functions whose formulation perfectly matches the arithmetic performed by a single neuron (a.k.a. a single-layer perceptron). In particular, with the set of weights w_i, the bias term b, and sigmoid as the activation function, a perceptron classifier exactly encodes the threshold function described in Equations 1 and 2 since the threshold in Equation 2 is mapped by sigmoid to the classification boundary of the neuron.

Therefore, we propose to utilize stochastic gradient descent to train our model as a single neuron with the sigmoid activation function and binary cross-entropy loss. Note that this essentially composes a logistic regression model. However, classic logistic regression barely has any zero weights, while as discussed above, zero and non-zero weights of a threshold function translate into literals and don't-cares, respectively. As a result, the max clause computed from a classic logistic regression model generally has the identical length with an input vector, which is not commonly considered interpretable. To improve the sparsity of weights, we employ the L_0 norm regularization technique proposed in [10], which penalizes the non-zero weights in a differentiable manner and allows to adjust the sparsity of weights by tuning the L_0 coefficient. We will show in the experiment section that with such a regularization approach, the trained threshold function translates well into concise rules that are human-interpretable.

4.1 OR-of-Neurons

Since a single-layer perceptron is generally considered as a linear model, it has a limited classification capability. Thus, we further propose a two-layer neural network structure, called *OR-of-Neurons* (OON) architecture, that is capable of learning more complicated decision boundaries.

The first layer of our OON model, called the *Rules Layer*, consists of a number of neurons with L_0 norm regularization applied, each of which performs the linear function f described in Equation 1. The outputs $\mathbf{z}$ are then binarized according to the activation function in Equation 2. In other words, every neuron in the first layer individually learns a threshold function and it outputs 1 when the threshold function is activated, or 0 otherwise.

We call the second layer the *OR Layer*, which is constructed with a single neuron whose weights all equal to 1 and the bias is -0.5. We do not train the parameters in this layer so it, with the sigmoid activation function and binary cross entropy loss, behaves as an OR gate: it by default makes a negative prediction when all its inputs being 0, whereas every activated threshold function sufficiently turns on this OR-gate and produces a positive output. Furthermore, since a threshold function also can be decomposed into the disjunction of a set of clauses (rules), this DNF pattern is propagated to the final output side according to the associative and idempotent laws, which results in a final prediction composed by a set of rules OR-ed together. For example, suppose the first layer encodes two threshold functions that can be converted into rules as follows: $F_1 = r_1 \vee r_2$ and $F_2 = r_2 \vee r_3$. The overall network can be then represented as follows:

$$F_{out} = F_1 \vee F_2 = (r_1 \vee r_2) \vee (r_2 \vee r_3) = r_1 \vee r_2 \vee r_3 \tag{8}$$

Since it has no impact on the rule-level complexity (number of literals in the rule), this OON model introduces the support of non-linear decision boundaries without any degradation of interpretability.

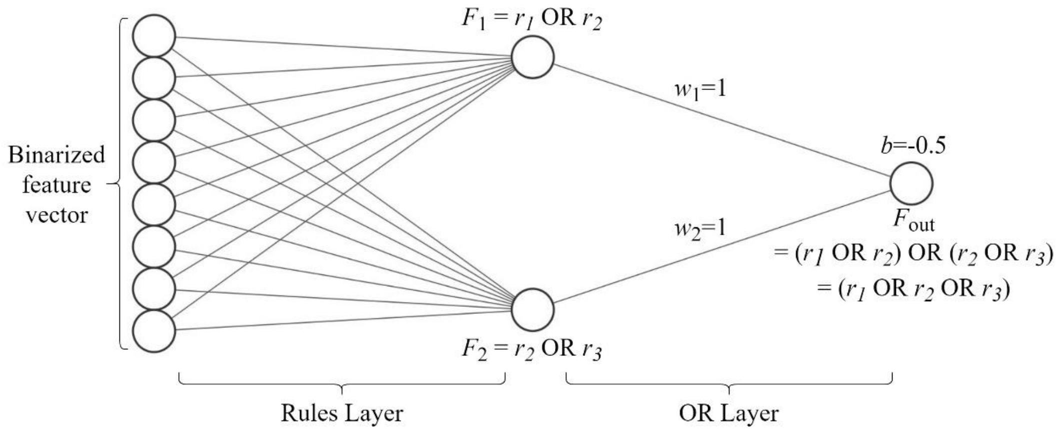

Figure 2: An example of the OON architecture with two neurons (i.e., threshold functions denoted by F_i) in rules layer.

4.2 Back-Propagation through Discretized Activations

With L_0 norm regularization applied to the Rules Layer, the overall loss function we optimize for can be expressed as follows:

$$L = L_{BCE} + \lambda L_R$$

where L_{BCE} is the binary cross-entropy loss, L_R stands for the regularization penalty, and λ is the regularization coefficient that balances the classification accuracy and rule simplicity. However, as can be observed, the activations of the first layer are discretized into binary integers that are not naturally differentiable and the classic gradient computation approach does not apply. Therefore, we utilize the straight-through estimator discussed in [14] with the gradient clipping technique. Denoted by $\hat{z}_i$ as the binarized activation based on z_i, we compute the gradient as follows:

$$g_{\hat{z}_i} = \begin{cases} 0, & \text{if } z_i < 0 \text{ or } z_i > 1, \dfrac{\delta L}{\delta z_i} < 0 \\ g_{z_i}, & \text{otherwise} \end{cases} \tag{10}$$

where $g_{\hat{z}_i}$ and g_{z_i} are the gradients of classification loss w.r.t. $g_{\hat{z}_i}$ and z_i, respectively. The condition $z_i < 0$ simulates the backward computation of the ReLU function, which introduces non-linearity into the training process and empirically improves the performance, whereas our motivation of the second condition is to address the saturation effect: we suppress the update of the full-precision activations that are greater than 1 and are still driven by the gradient to increase, since further raising activations does not produce any difference after binarization.

Based on empirical observations, we practically train the model by first training a full-precision version of the network with L_0 norm regularization applied, in which the binarization activation function is replaced, during the forward pass, with a clipped ReLU function whose ceiling value equals 1, and we clip its gradient using the same approach described in Equation 10 so that the training is rather consistent before and after introducing binarization. After the convergence of the full-precision model, we suppress the weight regularization and train the binarized network by only updating the non-zero weights. We will show in the experiment section this procedure leads to decent performance.

Chapter 4, in full is currently being prepared for submission for publication of the material. Qiao, Litao; Wang, Weijia. The thesis author was the primary investigator and author of this material. Chapter 4 is also coauthored with Wang, Weijia. The thesis author was the primary author of this chapter.

Chapter 5

Experiments

The numerical experiments are evaluated on 5 publicly available datasets with binary labels, which all have more than 10000 instances and more than 10 features before binarization. We only include the large datasets for evaluation because it is not our goal to generate optimized rules on small datasets. The first two selected datasets are from UCI Machine Learning Repository [15]: MAGIC gamma telescope (magic) and adult census (adult), which are also used in the recent works on rule set classifiers [9, 13, 16]. The magic is a dataset of pure numerical attributes while the adult dataset has mixed of both numerical attributes and categorical attributes. The recidivism predictions in North Carolina (reci) dataset has lots of binary attributes and is used to evaluate the models' performances on the datasets whose features are originally binary. The last one is a relatively recent dataset: FICO HELOC dataset (heloc), which has all continues features.

All the datasets except bank dataset are divided into 5 partitions using 5-fold stratified cross-validation method. To make sure the test results are comparable across the all the models, all models are trained and tested on the same training-test splits of the datasets that are preprocessed using the same method. For the binary attributes, we leave the value as it is while we applied the standard one-hot encoding to transform the value x to a vector of binary values for the categorical attributes. We used the same procedure recommended in the work [13] to binarize the numerical attributes: each value is compared with 9 quantile thresholds and encode 1 if it is less than the threshold and 0 otherwise. Note that one difference in the binarization procedure with other work is that we don't append negations in the one-hot encoding and comparison to simplify the training space.

Our goal is to learn a set of decision rules using our OON method and compare our result with other state-of-the-art rule learners and machine learning models. The results include models' test accuracy and interpretability. We define the model complexity the same way as in [9]: the number of rules plus the total number of conditions in the rule set and rule complexity is the average number of conditions in each rule of the model. We consider three other rule learner algorithms to directly compare with our work for both accuracy and interpretability: RIPPER algorithm , Bayesian Rule Sets (BRS) [13] and the column generation (CG) algorithm from [9]. The first one is an old rule set learning algorithm that is a variant of the Sequential Covering algorithm, while the other two are the representatives of the recent works on rule learning classifier. We used the open-source implementations we found on Github for all three algorithms. Other interpretable models used for comparison are the scikit-learn [17] implementations of decision tree of CART algorithm (CART) [18] and a fully connected layer of full-precision weights with ReLU activation function and 1 hidden layer of 10 neurons (FP-NN). The full-precision neural network is included in comparison as the representative of the best performance that black-box models can achieve on these datasets.

5.1 Classification Performance

We evaluate the predictive performance of our methods by comparing both test accuracy and interpretability with other state-of-the-art machine learning models. Nested cross validation was employed to select the parameters for other rule learners that explicitly trade-off between accuracy and interpretability to maximize the training set accuracies. To ensure that the final rule learner models are interpretable, we constrain the possible parameters for nested cross validation to a range which results in a low model complexity. Although there are many parameters in BRS to control the rule complexity, we only varied the multiplier κ in prior hyperparameter to save running time, which is also used in [9]. For

RIPPER, we varied the maximum number of conditions and maximum number of rules, which are directly related to the complexity of the model. The implementation of the CG algorithm [19] is a different variation from the one discussed in [9]. The CG implementation in [19] doesn't have the complexity bound C parameter but instead provides two parameters to specify the costs of each clause and of each condition, which are used in our experiment to control the rule set complexity. We leave all other parameters for these three algorithms as default. Since the range of our rule set complexity can vary a lot, which naturally corresponds to a wide accuracy range, we manually tune λ to generate the rule sets whose complexities are comparable to those of other rule learners, and test those rule sets on each test set. For CART, we constrained the maximum depth of trees to be 100 for all datasets to achieve better generalization. The test accuracy results of all models on all 5 datasets are shown in the table 1 and the corresponding complexities are shown in the table 2. We omitted the complexity results of CART and FP-NN because they have a different notion of model complexity and rule complexity.

Table 1: Test accuracy based on the 5-fold cross-validation (%, standard error in parentheses).

dataset	adult	magic	reci	heloc
OON	**82.36** (0.55)	82.23 (0.50)	64.14 (0.56)	**69.81** (3.83)
CG	81.33 (0.31)	72.17 (0.32)	53.41 (0.21)	65.96 (2.72)
BRS	81.05 (0.56)	82.38 (0.71)	**65.43** (0.58)	68.68 (0.31)
RIPPER	82.00 (0.60)	**83.01** (0.77)	64.89 (0.38)	69.76 (1.14)
CART	78.87 (0.12)	80.56 (0.86)	58.12 (0.52)	60.61 (2.83)
FP-NN	84.35 (0.33)	86.19 (0.61)	66.18 (0.59)	70.83 (3.42)

Table 2: Model complexity and rule complexity based on the 5-fold cross-validation.

dataset	adult	magic	reci	heloc
OON	23.80 | 3.10	135.8 | 4.43	**12.82 | 3.27**	33.20 | 4.35
CG	**14.50 | 1.90**	**41.80| 2.35**	19.61 | 7.76	**5.80 | 1.90**
BRS	50.20 | 3.28	141.0 | 2.98	14.29 | 2.95	23.20 | 3.00
RIPPER	73.00 | 4.54	241.0 | 5.96	50.20 | 5.41	109.4 | 6.92

While we constrained model complexity of OON to be similar to that of other rule learner algorithms, which were already tuned to have the best performance on each dataset, our method still generally beat all other rule learners on adult, bank and heloc datasets and were on par with FP-NN on bank, reci and heloc datasets. For those of the datasets where our method failed to be comparable with other rule learners, the accuracy differences are less than 2% and our models consistently have low complexity. Compared with RIPPER, which greedily mines good rules in each iteration to maximize the training accuracy, OON is very competitive in the sense that it has similar or equal test performance while consistently maintaining a lower model complexity and a rule complexity. The advantage of the CG algorithm over OON is that it consistently generate very sparse models despite that we varied the cost of each clause and the cost of each condition in a very wide range (10^{-2} to 10^{-20}). We suspect that this is due to that the version we found implements a heuristic beam search instead of the integer programming described in the paper. The CART decision tree algorithm turns out to be the worst performing models in the experiment, which might result from over-fitting.

5.2 Accuracy-Complexity trade-off

In this experiment, we compared the accuracy-complexity trade-off of our OON method with RIPPER. Both CG algorithm and BRS are ruled out in this experiment because we noticed that they didn't display expected trade-off capabilities in the first experiment. We varied λ and number of neurons in the hidden layer for OON method and maximum number of conditions and maximum number of rules for RIPPER to control the complexity of the rule set, resulting in two different sets of accuracy-complexity pairs. We run the experiment on adult dataset and the result is shown in the figure 3. The points that are connected by the lines are Pareto efficient, which are the accuracy-complexity pairs with less complexity and higher accuracy. From the figure 3, OON displays a better accuracy-complexity trade-off capability in two aspects. First, OON's Pareto frontier is strictly higher than RIPPER method, which is especially true in the lower complexity range. This confirms with the observation that we had in the first experiment: OON can get higher accuracy with lower complexity. Second, while the points of RIPPER method are clustered in a two narrow ranges despite that we evenly selected the parameters, OON has a much wider and smoother distribution of the trade-off.

Chapter 5, in full is currently being prepared for submission for publication of the material. Qiao, Litao; Wang, Weijia. The thesis author was the primary investigator and author of this material. Chapter 5 is also coauthored with Wang, Weijia. The thesis author was the primary author of this chapter.

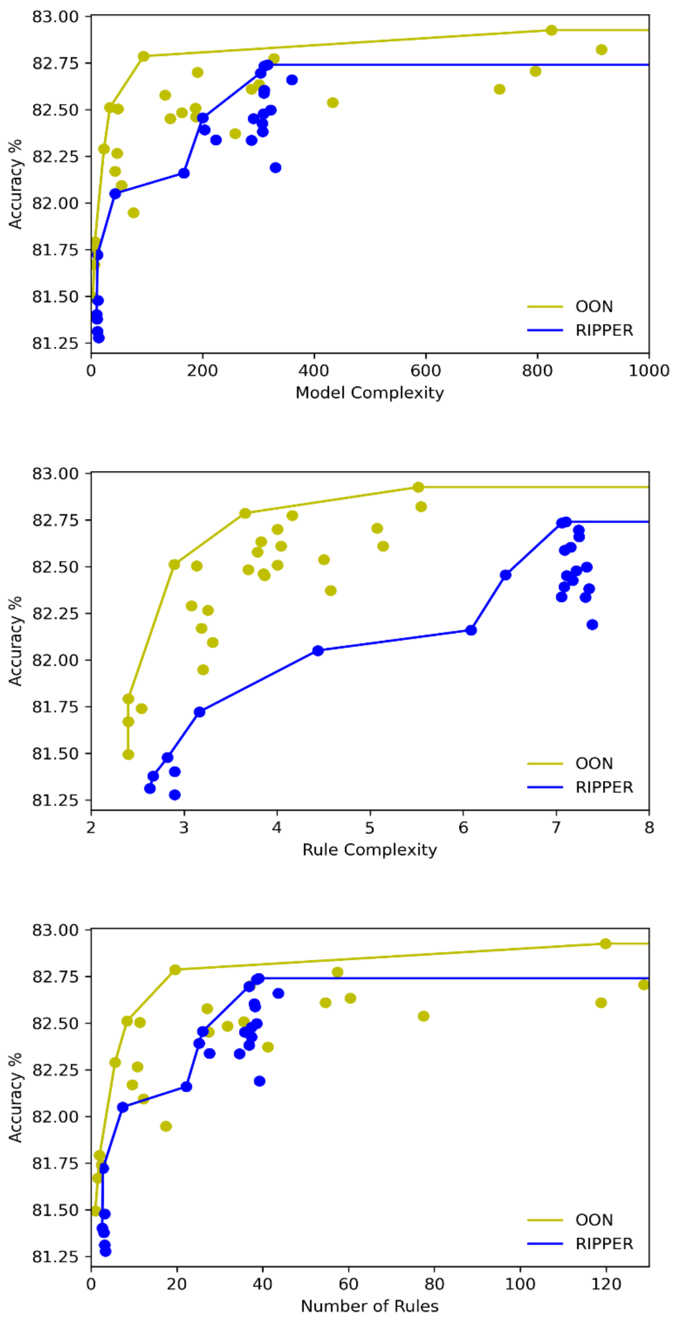

Figure 3: Accuracy-Complexity trade-offs on the adult dataset. Pareto efficient points are connected by line segments

Chapter 6

Conclusion

We have developed a neural network training method to learn accurate and interpretable decision rule sets for binary classification. A simple algorithm is proposed to convert threshold functions to decision rules with optimality guarantees. The rule set converted from the neural network inherits its good generalization capability and the experiments have shown that the our method achieved a superior predictive performance.

Although our method demonstrated competitive experimental results in Table 1 and 2, it comes with the sacrifice of the training time. Due to the specialty of neural network structure, the training of our method with L_0 regularization typically takes longer time to converge than other rule learners or machine learning models. Another challenge of putting our method for practical use is the best hyperparameter selection, which varies for different datasets. We believe those problems can be solved by integrating our method with some of the state-of-the-art neural training techniques and we leave these possible improvements for future work.

Chapter 6, in full is currently being prepared for submission for publication of the material. Qiao, Litao; Wang, Weijia. The thesis author was the primary investigator and author of this material. Chapter 6 is also coauthored with

Wang, Weijia. The thesis author was the primary author of this chapter.

References

[1] A. Flores, K. Bechtel and C. Lowenkamp, "False Positives, False Negatives, and False Analyses: A Rejoinder to “Machine Bias: There’s Software Used Across the Country to Predict Future Criminals. And it’s Biased Against Blacks.”," *Federal probation,* vol. 80, 9 2016.

[2] M. T. Ribeiro, S. Singh and C. Guestrin, "Anchors: High-Precision Model-Agnostic Explanations," in *AAAI*, 2018.

[3] M. T. Ribeiro, S. Singh and C. Guestrin, *"Why Should I Trust You?": Explaining the Predictions of Any Classifier,* 2016.

[4] H. Lakkaraju, E. Kamar, R. Caruana and J. Leskovec, "Faithful and Customizable Explanations of Black Box Models," in *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, New York, NY, USA, 2019.

[5] 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*, New York, NY, USA, 2016.

[6] D. Nauck and R. Kruse, "Obtaining interpretable fuzzy classification rules from medical data," *Artificial Intelligence in Medicine,* vol. 16, pp. 149-169, 1999.

[7] J. Zeng, B. Ustun and C. Rudin, "Interpretable classification models for recidivism prediction," *Journal of the Royal Statistical Society: Series A (Statistics in Society),* vol. 180, p. 689–722, 9 2016.

[8] R. Caruana, Y. Lou, J. Gehrke, P. Koch, M. Sturm and N. Elhadad, "Intelligible Models for HealthCare: Predicting Pneumonia Risk and Hospital 30-Day Readmission," in *Proceedings of the 21th ACM SIGKDD*

International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, 2015.

[9] S. Dash, O. Günlük and D. Wei, "Boolean Decision Rules via Column Generation," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, Red Hook, NY, USA, 2018.

[10] C. Louizos, M. Welling and D. P. Kingma, *Learning Sparse Neural Networks through L_0 Regularization,* 2017.

[11] W. W. Cohen, "Fast Effective Rule Induction," in *In Proceedings of the Twelfth International Conference on Machine Learning*, 1995.

[12] G. Su, D. Wei, K. R. Varshney and D. M. Malioutov, *Interpretable Two-level Boolean Rule Learning for Classification,* 2015.

[13] T. Wang, C. Rudin, F. Doshi-Velez, Y. Liu, E. Klampfl and P. MacNeille, "A Bayesian Framework for Learning Rule Sets for Interpretable Classification," *Journal of Machine Learning Research,* vol. 18, pp. 1-37, 2017.

[14] Y. Bengio, N. Léonard and A. Courville, *Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation,* 2013.

[15] D. Dua and C. Graff, *UCI Machine Learning Repository,* 2017.

[16] S. Dash, D. M. Malioutov and K. R. Varshney, "Screening for learning classification rules via Boolean compressed sensing," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014.

[17] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot and E. Duchesnay, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research,* vol. 12, p. 2825–2830, 2011.

[18] L. Breiman, J. Friedman, C. J. Stone and R. A. Olshen, Classification and Regression Trees, Taylor & Francis, 1984.

[19] V. Arya, R. K. E. Bellamy, P.-Y. Chen, A. Dhurandhar, M. Hind, S. C. Hoffman, S. Houde, Q. V. Liao, R. Luss, A. Mojsilović, S. Mourad, P. Pedemonte, R. Raghavendra, J. Richards, P. Sattigeri, K. Shanmugam, M. Singh, K. R. Varshney, D. Wei and Y. Zhang, *One Explanation Does Not Fit All: A Toolkit and Taxonomy of AI Explainability Techniques,* 2019.