Machine Learning with Applications 10 (2022) 100429

Contents lists available at ScienceDirect

Machine Learning with Applications

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

Tabular machine learning using conjunctive threshold neural networks

Weijia Wang [*], Litao Qiao, Bill Lin

Electrical and Computer Engineering, University of California San Diego, 9500 Gilman Dr., La Jolla, 92093, CA, United States

ARTICLE INFO

Keywords:
Tabular machine learning
Neural networks
Explainable machine learning

ABSTRACT

We propose a novel three-layer neural network architecture with threshold activations for tabular data classification problems. The hidden layer units correspond to trainable neurons with arbitrary weights and biases and a step activation. These neurons are logically equivalent to threshold logic functions. The output layer neuron is also a threshold function that implements a conjunction of the hidden layer threshold functions. This neural network architecture can leverage state-of-the-art network training methods to achieve high prediction accuracy, and the network is designed so that minimal human understandable explanations can be readily derived from the model. Further, we employ a sparsity-promoting regularization approach to sparsify the threshold functions to simplify them, and to sparsify the output neuron so that it only depends on a small subset of hidden layer threshold functions. Experimental results show that our approach outperforms other state-of-the-art interpretable decision models in prediction accuracy.

1. Introduction

In machine learning applications like healthcare and criminal justice where human lives may be deeply impacted, creating decision models that can provide human understandable explanations is critically important (Rudin, 2018). In these applications, the datasets are often provided as tabular data with samples as rows in a table and a common set of naturally meaningful features as columns. A toy example of a tabular dataset is shown in Table 1.

Due to their inherent explainability, decision rule sets (Cohen, 1995; Dash, Günlük, & Wei, 2018; Lakkaraju, Bach, & Leskovec, 2016; Wang et al., 2017) are often a popular model class in these tabular learning problems. Decision rule sets not only provide accurate predictions, but the corresponding matching rules also provide explanations that humans can easily understand. In particular, the explanations are expressed directly in terms of meaningful categorical (e.g., Cholesterol equal to Normal or High) or numerical (e.g., age) input attributes, where the binary encoding of categorical and numerical attributes is well-studied (Dash et al., 2018; Wang et al., 2017). However, they are not the winning model class in these domains in terms of prediction accuracy. For example, gradient boosted and ensemble decision trees (Breiman, 2001; Chen & Guestrin, 2016; Ke et al., 2017) and neural network models (Abutbul, Elidan, Katzir, & El-Yaniv, 2020; Arik & Pfister, 2019) are generally superior in prediction performance. However, these models are generally considered to be black-box models (Rudin, 2018) where the predictions are difficult or impossible to interpret. This is in sharp contrast to the interpretability of rule-based sentences that decision rule sets provide, which can be easily understood by humans.

In this paper, we propose a new neural network architecture for tabular data called a Conjunctive Threshold Neural Network, or CT-Net for short. The proposed structure comprises a hidden layer of threshold logic functions, which are just conventional neurons with a step activation function that are trainable with arbitrary (positive or negative) full-precision weights and biases. This neural network architecture can be trained to achieve high prediction accuracy, but unlike conventional neural network models, gradient boosting decision trees, and random decision forests, human understandable explanations in terms of meaningful input features similar to decision rules can be easily derived from CT-Net. Also, unlike existing interpretable rule-learning methods (Dash et al., 2018; Qiao, Wang, & Lin, 2021; Wang et al., 2017) that provide human understandable explanations, we do not ever explicitly generate a decision rule set from CT-Net. This means that our network of threshold functions can implicitly encode potentially complicated rules to achieve high prediction accuracy, but yet the explanations generated can nonetheless be understandable. In particular, the explanation derived is provably *minimal* in the number of features in the conjunction. Therefore, we believe CT-Net can be widely used as an replacement for tree ensemble methods (random forest and gradient boosting trees) in the area where both accuracy and interpretability are required.

The outline of the paper is as follows: Section 2 summarizes related work. Section 3 describes our proposed CT-Net architecture. Section 4 describes how provably minimal explanations can be easily derived from a CT-Net inference. Section 5 describes a sparsity-promoting regularization approach for training CT-Nets. Section 6 provides extensive evaluation of our proposed approach. Section 7 concludes the paper.

* Corresponding author.
E-mail addresses: wweijia@eng.ucsd.edu (W. Wang), l1qiao@eng.ucsd.edu (L. Qiao), billlin@eng.ucsd.edu (B. Lin).

https://doi.org/10.1016/j.mlwa.2022.100429
Received 21 July 2022; Received in revised form 20 October 2022; Accepted 21 October 2022
Available online 29 October 2022

Table 1
A toy example of a tabular dataset. The first seven columns are input features, and the last column is the classification.

Gender	Age	BP	Cholesterol	Glucose	Smoker	Drinker	Disease
Male	34	Normal	Normal	Normal	No	Yes	No
Female	62	High	Normal	High	Yes	No	Yes
Male	55	High	High	Normal	No	No	Yes
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Female	25	High	Normal	Normal	Yes	No	No

2. Related work

Besides decision rule sets (Cohen, 1995; Dash et al., 2018; Lakkaraju et al., 2016; Wang et al., 2017), decision lists (Letham, Rudin, McCormick, & Madigan, 2015; Rivest, 1987) and decision trees (Breiman, Friedman, Stone, & Olshen, 1984) are also interpretable rule-based models. Decision lists are ordered rules in an if-then-else sequence, and decision trees have paths that can be interpreted as rules. In addition to providing prediction, these methods also provide human understandable explanations that can be derived from the matching rule.

Gradient boosting decision trees (Chen & Guestrin, 2016; Ke et al., 2017) and random forests (Breiman, 2001) have also been used to provide better predictions in tabular data classification problems. Although these methods provide superior prediction performance in comparison to rule-based methods, they are generally not interpretable. In certain application areas, their lack of interpretability may make it difficult to gain public trust for their use, which may hinder their widespread adoption in these domains.

Building on the notable success that deep neural networks have shown on perceptual learning tasks, like image classification (He, Zhang, Ren, & Sun, 2015), researchers have recently turned to neural network models for tabular data learning as well (Abutbul et al., 2020; Arik & Pfister, 2019; Qiao et al., 2021). The work in Abutbul et al. (2020) and Arik and Pfister (2019) aim to capture aspects of gradient boosting decision trees and random forests that have made these models successful, and they are able to achieve comparable performance as these approaches with neural models. However, like gradient boosting decision trees and random decision forests, these models remain uninterpretable in the sense that they do not provide explanations that are easily understandable by humans.

In contrast, Qiao et al. (2021) recently proposed a neural network architecture that directly maps to a decision rule set in disjunctive normal form. In this architecture, the neurons in the hidden layer are restricted in a way so that they map directly to a conjunction (AND) of input features that correspond to interpretable decision rules, followed by an output neuron that maps to a disjunction (OR) operation that aggregates a collection of decision rules into a set. This approach achieves better performance than traditional rule-based and decision tree methods (Breiman et al., 1984; Cohen, 1995; Dash et al., 2018; Lakkaraju et al., 2016; Wang et al., 2017) while retaining the ability to provide human understandable explanations. However, the restrictions imposed on the hidden layer neurons to have direct one-to-one mappings to conjunctive rules unnecessarily limits the search space during neural net training.

Another body of work aims to develop post-hoc explainers that can explain predictions from black-box models. Heuristic algorithms are employed in Ribeiro, Singh, and Guestrin (2016, 2018) to generate explanations without the knowledge of the entire model. Although a primary objective of these algorithms is to achieve high fidelity of the explanations to the original model, the derived explanations cannot be guaranteed to be completely consistent with the underlying model distribution.

Finally, the works in Audemard, Koriche, and Marquis (2020), Choi, Shi, Shih, and Darwiche (2017), Ignatiev, Narodytska, and Marques-Silva (2018), Izza and Marques-Silva (2021), Shi, Shih, Darwiche, and Choi (2020) and Shih, Choi, and Darwiche (2018) are on the compilation of models into tractable forms. In these approaches, explanations consistent with the original model can be queried from the model's equivalent tractable form. Our work is complementary in that we aim for an approach in which a simple and fast algorithm can be applied to directly derive human understandable explanations from our proposed model.

3. Conjunctive Threshold Neural Networks

In this section, we introduce the architecture of the proposed Conjunctive Threshold Neural Network, or CT-Net for short. The network is designed for tabular classification problems where besides making accurate predictions, the *explanation* of decisions is also essential. In particular, CT-Net is a simple three layer neural network architecture, comprising an input layer of n input units, a hidden layer of k units, and an output layer with a single output unit. A toy example is shown in Fig. 1 for predicting heart disease risk, which we use to illustrate several key ideas in this section.

Input layer: The input layer consists of n input units, each passing its corresponding assigned binarized value to each neuron in the hidden layer. Tabular datasets often comprises binarized, categorical and numerical attributes. To handle categorical and numerical attributes, well-studied pre-processing procedures in the machine learning literature can be used to encode them into binarized input vectors[1]: standard one-hot encoding can be used for categorical attributes, and standard quantile discretization can be used for numerical attributes.

Hidden layer of threshold functions: The hidden layer comprises k neurons that are trainable with arbitrary (positive or negative) full-precision weights and biases. They implement threshold functions by means of a binary step activation function. In Fig. 1, the blue dashed lines at the inputs of a hidden neuron indicate that the corresponding features have zero weights, which means they do not appear in the corresponding threshold function. As discussed in the next section, each threshold function implicitly implements an underlying Boolean logic function that encodes logical conditions on the inputs that will lead to a positive prediction.

Output conjunction layer: The output unit implements a *conjunction* of a subset of the k threshold functions in the hidden layer, which is also implemented as a single threshold function with trainable binarized weights (more details are discussed in Section 5), where a weight of 1 or 0 indicates if the corresponding threshold function in the hidden layer is included in or excluded from the conjunction, respectively (a 0 weight is shown as a blue dashed line at the input of the output neuron in Fig. 1). Further, a dynamic bias based on the weights is employed as follows:

$$b = -\sum_{i=1}^{k} w_i + \epsilon. \qquad (1)$$

where ϵ is a small number between 0 and 1 (e.g., $\epsilon = 0.5$), and w_i is the binary weight (i.e., 1 or 0) as just discussed. This output threshold unit implements a logical-AND operation since the output unit can make a positive prediction (with the logits of ϵ) only if all threshold functions (that are not excluded with a zero weight) in the hidden layer are activated, whereas by default, it makes a negative prediction due to the negative summation of the weights. Since each threshold function implicitly implements an underlying Boolean logic function, the logical-AND of these threshold functions also implicitly implements a Boolean logic function for the network. This conjunction of threshold functions can be trained to implement any Boolean logic function at the output,

[1] Interpretable rule-learning methods (Dash et al., 2018; Qiao et al., 2021; Wang et al., 2017) widely studied to model tabular classification problems also commonly assume this input binarization pre-processing step.

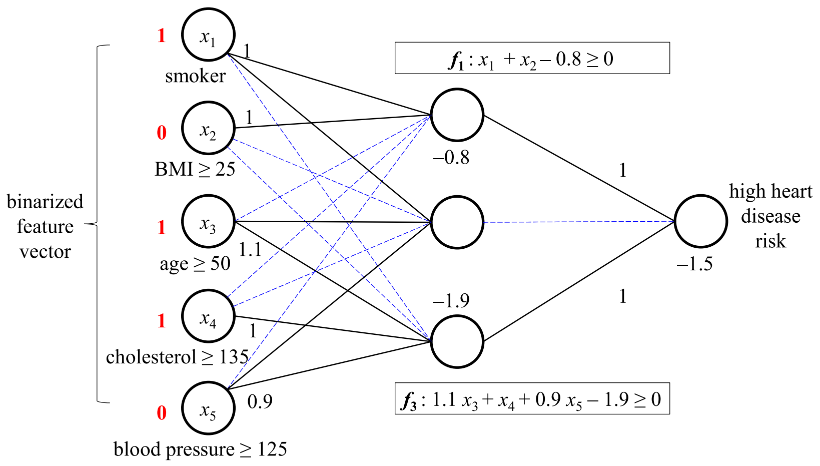

Fig. 1. A toy example of the CT-Net architecture with hidden layer neurons as threshold logic functions and the output neuron implementing a conjunction.

which can model any prediction problem that can be encoded into a binary classifier.

Straight-through estimator: As discussed earlier, the proposed neural network incorporates the binary step activation function, which is almost non-differentiable everywhere. Therefore, we adopt the straight-through estimator (Bengio, Léonard, & Courville, 2013) to address this issue. In particular, it was originally proposed in Bengio et al. (2013) to use the identity function as the derivative of a step function, while (Courbariaux & Bengio, 2016) further incorporates the gradient clipping technique to cancel the gradients when the activation is too large, i.e., the backward function is similar to ReLU, which also introduces non-linearity into the network. In our approach, we empirically found it is better to clip the gradients when the full-precision activation (pre-step activation) is either too large or too small, which is analogous to the backward function of the clipped ReLU activation. This approach in practice effectively prevents the weights from getting infinitely large or small. Mathematically, the straight-through estimator with the gradient clipping technique that we use to address this problem is as follows:

$$g_{\hat{z}_i} = \begin{cases} 0, & \text{if } z_i < -1 \text{ or } z_i > 0 \\ g_{z_i}, & \text{otherwise} \end{cases} \tag{2}$$

where z and $\hat{z}_i$ are the full-precision activation and the binarized activation after our step function, respectively. L is the classification loss. $g_{\hat{z}_i} = \frac{\partial L}{\partial \hat{z}_i}$ and $g_{z_i} = \frac{\partial L}{\partial z_i}$, which are the gradients of classification loss w.r.t. $\hat{z}_i$ and z_i, respectively.

Similar to the clipped ReLU activation function with a clipping boundary of 1, the outputs produced by the step function always fall into the range between 0 and 1. Therefore, our step activation function can be essentially viewed as a low-precision clipped ReLU function. However, we note that a floating-point 0 will be directly binarized to 1 in our approach, which corresponds to a shift introduced by the binarization, which should be addressed in backward propagation. As a result, we propose to clip the gradient at −1 and 0 rather than 0 and 1 as in the clipped ReLU activation, and experimental results show this straight-through estimator work very well in practice.

An example: Consider again the toy example shown in Fig. 1 for predicting heart disease risk. The input variables $x_1, x_2, \ldots, x_5$ correspond to whether or not the individual is a smoker, overweight, or older than 50, or has high cholesterol or blood pressure, respectively. The instance shown (in red) is the input assignment $\langle x_1, x_2, x_3, x_4, x_5 \rangle = [10110]$. With this instance, threshold functions f_1 and f_3 evaluate to true (i.e., evaluate to 1), whereas threshold function f_2 is directly turned off by the output layer. For f_1, the threshold function evaluates to true if either the individual is a smoker or overweight as the weights of x_1 and x_2 are both individually greater than 0.8. For f_3, the threshold function evaluates to true if any two out of the three conditions (older than 50, high cholesterol, or high blood pressure) are true, since the sum of the weights of any two conditions will be sufficient to exceed 1.9. Indeed, the classifier will make a positive prediction that this individual is at a high risk of heart diseases since both f_1 and f_3 are activated.

To guarantee CT-Net makes a positive prediction, both f_1 and f_3 need to evaluate to true at the same time (because the output neuron implements a conjunction of inputs with 1 weights), where the individual being a *smoker* suffices the first threshold function, and what sufficiently activates f_3 is that the individual is *older than 50* and has *high cholesterol*. Therefore, the *explanation* for *why* this individual is predicted to have a high heart disease risk is *smoker*, *older than 50* and *high cholesterol*. While the explanation for this prediction is unique, it is possible that there exist multiple explanations for certain positive predictions in some scenarios. As detailed later in the paper, provably minimal explanations can be readily derived for any given positive prediction.

Unlike existing rule-learning methods (Dash et al., 2018; Qiao et al., 2021; Wang et al., 2017), we do not ever explicitly generate a decision rule set from the conjunction threshold network. This means that our network of threshold functions can implicitly encode potentially complicated rules to achieve high prediction accuracy. State-of-the-art stochastic gradient descent training methods can be used to achieve high prediction accuracy. Also, well-developed sparsity-promoting techniques can be invoked to simplify the network in a way that leads to succinct threshold functions, as discussed later in the paper. In the next section, we describe more formally how provably minimal explanations can be readily derived for positive predictions made using CT-Net.

4. Deriving explanations

In this section, we describe how provably minimal human understandable explanations can be readily derived from a CT-Net prediction.

4.1. Threshold functions and slack

It should be clear from the previous section that a neuron in the hidden layer of the CT-Net corresponds to a *threshold function* of the form

$$z(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b, \tag{3}$$

$$f(\mathbf{x}) = \begin{cases} 1 & \text{if } z(\mathbf{x}) \geq 0 \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

When the n inputs are binary features, a threshold function f implements an underlying Boolean logic function $f : \{0,1\}^n \to \{0,1\}$.

As discussed, a CT-Net is a conjunction (logical-AND) of k threshold functions, $F = \{f_1, f_2, \dots, f_k\}$. As such, a CT-Net also implements an underlying Boolean logic function $F : \{0,1\}^n \rightarrow \{0,1\}$. Therefore, Boolean algebra terminologies and properties are applicable to both individual threshold functions as well as the overall CT-Net, which we summarize here.

An *instance* $\alpha \in \{0,1\}^n$ is a specific assignment to the input features. With respect to the CT-Net F, a positive instance is one such that $F(\alpha) = 1$, and a negative instance is one such that $F(\alpha) = 0$. A *literal* ℓ_i is a feature (positive literal) or its negation (negative literal), denoted as $\ell_i = x_i$ and $\ell_i = \bar{x}_i$, respectively. A *term* π is a consistent conjunction of literals, e.g., $x_1 \wedge \bar{x}_2 \wedge x_3$, or simply $x_1\bar{x}_2x_3$.[2] The *length* of π, denoted as $|\pi|$, is the number of literals that it includes. We say that a term π_i *covers* or *contains* another term π_j, written as $\pi_j \Rightarrow \pi_i$, if and only if π_j includes all the literals in π_i (e.g., $x_1\bar{x}_2$ covers $x_1\bar{x}_2x_3$).

An *implicant* π of a Boolean function F is a term that *satisfies* F, written as $\pi \Rightarrow F$, meaning all instances covered by π are positive instances. A *prime implicant* (or simply a *prime*) is an implicant that is not covered by any other implicant.[3]

We next describe several concepts that we will use in the algorithm for generating explanations for positive predictions of the CT-Net.

Definition 1 (*Slack*)**.** For an instance α, the *slack* of α with respect to a threshold function f corresponds to $z(\alpha)$ in Eq. (3). Therefore, $f(\alpha) = 1$ if the slack is non-negative, and 0 otherwise. For a term π, the *slack* of π is defined as the minimum slack achieved by the instances that π covers:

$$z(\pi) = \min_{\alpha} z(\alpha), \quad \text{s.t. } \alpha \Rightarrow \pi. \tag{5}$$

The slack of π can be directly computed by setting every feature x_i to its worst-case value if it does not appear in the term π: i.e., set $x_i = 0$ if $w_i > 0$ and $x_i = 1$ otherwise. As such, $z(\pi)$ is minimized.

Definition 2 (*Group Slack*)**.** Let F be the Boolean logic function defined by the *conjunction* of k threshold logic functions, $\{f_1, f_2, \dots, f_k\}$. For an instance α, let $z_i(\alpha)$ be the slack of α with respect to the corresponding threshold function f_i. Then the *group slack* of α with respect to F corresponds to the minimum among the slacks of the threshold functions:

$$z_F(\alpha) = \min_{i} z_i(\alpha). \tag{6}$$

Therefore, $F(\alpha) = 1$ if the group slack is non-negative, which implies the slacks of *all* individual threshold functions to be non-negative, and 0 otherwise. For a term π, the *group slack* of π is defined as the minimum slack achieved by the instances that π covers:

$$z_F(\pi) = \min_{\alpha} z_F(\alpha), \quad \text{s.t. } \alpha \Rightarrow \pi. \tag{7}$$

Here also, the group slack of π can be directly computed by setting every feature x_i to its worst-case value if it does not appear in the term π: i.e., set $x_i = 0$ if $w_i > 0$ and $x_i = 1$ otherwise. As such, $z_F(\pi)$ is minimized. Note that for $z_F(\pi)$ to be non-negative, the slacks of *all* individual $z_i(\pi)$ must also be non-negative.

4.2. Generating explanations

As discussed, a CT-Net F is equivalent to an underlying Boolean logic function, which can be viewed as a *binary classifier*, where $F(\alpha) = 1$ means the *decision* is positive, and negative otherwise. Intuitively, an explanation for a positive instance is some subset of its literals.

Referring to the example depicted in Fig. 1, an explanation for the overall CT-Net requires it to be an explanation for *both* f_1 and f_3, as the CT-Net output is a conjunction of both threshold functions. In this example, being a *smoker* suffices to activate f_1, and being *older than 50* with *high cholesterol* suffices to activate f_3. Therefore, an *explanation* as to *why* this individual is predicted to have a high heart disease risk is because the individual is a *smoker* and *older than 50,* and has *high cholesterol.* We formalize below what explanations are and how they can be readily derived from a CT-Net prediction.

Definition 3 (*Explanation*)**.** An *explanation* for a positive decision on an instance α is an implicant that contains the instance.

Definition 4 (*Minimal Explanation*)**.** A *minimal explanation* is a prime that contains the instance.

There can be different explanations that are consistent with the prediction that a CT-Net makes for a particular instance. From a user's perspective, simpler explanations are better, i.e. its length should be short, so that it can be easily comprehended. Also, shorter explanations usually cover more feature space, and thus provide users with more insights into the behavior of the CT-Net.

Algorithm 1 Derive Minimal Explanation

Input: A set of threshold functions $F = \{f_1, f_2, \dots, f_k\}$, positive instance α

Output: A minimal explanation π

```
1: $\pi \leftarrow \alpha$
2: while $\pi \neq \emptyset$ do
3:     $\mathcal{L} = \emptyset$
4:     for $\ell_i \in \pi$ do
5:         if $z_F(\pi \setminus \{\ell_i\}) \geq 0$ then
6:             $\mathcal{L} = \mathcal{L} \cup \{\ell_i\}$
7:         end if
8:     end for
9:     if $\mathcal{L} = \emptyset$ then
10:        return $\pi$
11:    else
12:        $\ell_i \leftarrow$ select_candidate($\mathcal{L}$)
13:        $\pi \leftarrow \pi \setminus \{\ell_i\}$
14:    end if
15: end while
16: return $\pi$
```

We now describe our algorithm for finding a minimal explanation for a positive prediction of a CT-Net. The pseudo code is outlined in Algorithm 1. In this algorithm, we start by treating the instance α itself as the current explanation π, and we then iteratively remove one feature at a time from the current explanation as long as a candidate feature can be identified such that the slack $z_j(\pi)$ remains non-negative for all threshold functions or until there are no more features.

There are two key parts to Algorithm 1. The first key part is in Lines 3–8, in which a list $\mathcal{L}$ of features are identified as candidates for removal. A feature ℓ_i is a candidate for removal if its removal does not cause the slack of any threshold function to become non-negative. That is

$$z_j(\pi \setminus \{\ell_i\}) \geq 0, \forall j.$$

If no such candidate exists or if there are no more features, then we have arrived at a minimal explanation.

The second key part is in Line 12, in which the **select_candidate** function is called to select a candidate feature from the set $\mathcal{L}$. One approach is to select the feature ℓ_i from $\mathcal{L}$ that maximizes the *average slack* among the threshold functions. That is, let $z_j(\pi \setminus \{\ell_i\})$ be the slack

[2] We will use [101] as a shorthand for the term $x_1\bar{x}_2x_3$. As another example, we will use [10−] as a shorthand for the term $x_1\bar{x}_2$, with "−" to mean that a literal for the corresponding feature is not included in the term.

[3] The terminologies term, implicant, and primes apply to any Boolean logic function, including both the individual threshold functions f_i and the overall logic function F induced by the CT-Net.

for threshold function f_j by removing ℓ_i, and let

$$z_{avg} = \text{average}\{z_j(\pi \setminus \{\ell_i\})\}. \tag{8}$$

We then select the ℓ_i that maximizes z_{avg}.

Alternatively, we can select the feature ℓ_i from $\mathcal{L}$ that maximizes the *minimum slack* among the threshold functions. That is, let

$$z_{min} = z_F(\pi \setminus \{\ell_i\}) = \min\{z_j(\pi \setminus \{\ell_i\})\}. \tag{9}$$

In this alternative approach, the ℓ_i that maximizes z_{min} would be selected.

Experimentally, we found that maximizing the average slack (i.e., Eq. (8)) to be more effective, and thus this is the approach that we adopted in our evaluation section (*cf.* Section 6).

Theorem 1. *The explanation derived using Algorithm 1 is minimal.*

Proof. We prove this theorem by contradiction. Assume the explanation π_1 generated with Algorithm 1 is not minimal. Then there must exists another prime π_2 such that $\pi_1 \Rightarrow \pi_2$ (π_2 covers π_1). This implies there exists a literal ℓ_i such that $\ell_i \in \pi_1$ and $\ell_i \notin \pi_2$. Since π_2 is a prime, it must guarantee that all threshold functions in the hidden layer evaluate to true, i.e., $z_F(\pi_2) \geq 0$. This would mean further removing ℓ_i from π_1 would still produce a positive group slack. However, this is contradictory to Algorithm 1 since removing any additional feature from π_1 would lead to a negative group slack (i.e., $\mathcal{L} = \emptyset$). □

5. Sparsity-Promoting training of CT-Net

We leverage the stochastic gradient descent (SGD) algorithm to efficiently train CT-Net. In particular, a binary cross-entropy function is used as the loss function to measure the error between the predicted output and the real labels. We recognize that the step functions have zero-gradients everywhere except 0, and we tackle this problem by employing a straight-through estimator approach (Bengio et al., 2013) with a gradient clipping technique to back-propagate gradient updates through the activations of threshold functions in the hidden layer.

It should be clear from the previous section that zero weights in a threshold function mean that the corresponding inputs will not have any effect on the logic of that threshold function, and the corresponding threshold formula becomes simpler. Intuitively, maximizing the sparsity of the threshold functions in the hidden layer encourages simpler explanations. Further, our algorithms for deriving explanations can also benefit from having weights that have small absolute values. This is because they will less impact on the available slack of the corresponding threshold function when removed.

To incorporate the idea proposed above in the training process, we propose to add a sparsity-promoting regularizer to encourage both zero weights and weights with small absolute values. In particular, we employ an improved version of L_1 regularization called reweighted L_1 regularization (Candes, Wakin, & Boyd, 2007), which drives the weights with smaller absolute values down to zero faster by giving those weights relative larger gradients. Mathematically, the reweighted L_1 minimization can be achieved by employing a log-sum penalty term as the regularization loss

$$\mathcal{L}_R(W) = \log(\|W\|_1 + \epsilon), \tag{10}$$

where $\epsilon > 0$ is a small value added to ensure numerical stability (e.g., $\epsilon = 0.1$). To even encourage more zero weights in the threshold functions, we further prune the weights with absolute values below a certain threshold by setting them directly to zero (Han, Pool, Tran, & Dally, 2015).

As discussed in the previous section, zero weights in the output layer are also helpful in excluding the unnecessary threshold functions of the hidden layer from the conjunction. Therefore, the reweighted L_1 minimization is applied again to the output layer to promote sparsity. However, since the binarized weights are required for the output layer, along with pruning the weights below a certain threshold as in the hidden layer, during the feed-forward phase, we also set the weights above the threshold directly to one, while we maintain and keep updating the full-precision values of the weights through back-propagation. Note that negative weights are not expected in the output layer, so we initialize the full-precision weights to be all one and always prune the weights below the positive threshold (whereas weights are also compared with a negative threshold in the hidden layer).

With the regularizer applied to the hidden layer and output layer, the overall loss function we optimize for becomes as follows: we add a regularization term into the loss function of the form

$$\mathcal{L} = \mathcal{L}_{BCE} + \lambda_1 \mathcal{L}_{R_1} + \lambda_2 \mathcal{L}_{R_2}, \tag{11}$$

where $\mathcal{L}_{BCE}$ is the binary cross-entropy loss, $\mathcal{L}_{R_1}$ and $\mathcal{L}_{R_2}$ are the regularization loss as explained in Eq. (10) for the hidden layer and output layer, respectively, and λ_1 and λ_2 are their corresponding regularization coefficients.

6. Experimental evaluation

Table 2
The details of the datasets used in the experiment. The imbalance ratio is calculated as the number of negative instances divided by the number of positive instance.

Dataset	# Instances	# Features	Imbalance ratio
Adult	30 162	14	3.02
Magic	19 020	10	1.84
House	22 784	16	0.42
Recidivism	8680	16	1.72
Chess	28 056	6	1.48
Retention	10 000	8	1.96
Churn	7032	19	2.76
Airline	25 893	22	1.28
Heloc	10 459	23	1.09
Churn2	10 000	10	3.91
Surgical	14 635	24	2.97

Datasets. In this section, we evaluated the proposed CT-Net along with a set of baseline approaches on 11 publicly available tabular classification datasets. Three datasets are from UCI Machine Learning Repository (Dua & Graff, 2017): Adult Census (adult), MAGIC Gamma Telescope (magic), and Chess: King–Rook vs. King (chess). Four datasets are from Kaggle: Telco Customer Churn (churn), Churn Modeling (churn2), Dataset Surgical binary classification (surgical), and Airline Passenger Satisfaction (airline). The other four datasets are: House_16H (house) (Vanschoren, van Rijn, Bischl, & Torgo, 2013), TED Dataset (retention) (Arya et al., 2019), Predicting Recidivism in North Carolina, 1978 and 1980 (recidivism) (Schmidt & Witte, 1988) and Home Equity Line of Credit Dataset (heloc) (Chen et al., 2018). Most of these datasets consist of more than 10,000 instances that originally include binary, categorical, and numerical attributes. More details of the datasets are shown in Table 2 As we can see from the "imbalance ratio" column of Table 2, the datasets selected vary from nearly balanced (1.09) to considerably imbalanced (3.91). Although other evaluation metrics such as F1-score might be better suited for imbalanced datasets, we choose to use the test accuracy that is much easier to interpret and widely used in the experiments of papers (Dash et al., 2018; Wang et al., 2017) on the similar topic.

Baselines and Pre-processing. The baseline approaches evaluated as comparisons consist of four rule learners, including Decision Rule Net (DR-Net) (Qiao et al., 2021), the Column-Generation-Based algorithm (CG) (Dash et al., 2018), RIPPER (Cohen, 1995), and Bayesian Rule Sets (BRS) (Wang et al., 2017); and three traditional machine learning classifiers, including decision trees (CART), random forests (RF), and gradient boosting trees (XGB). In particular, RIPPER is an old variant of the Sequential Covering algorithm for greedily mining rule set from

Table 3
Average test accuracy based on the nested 5-fold cross-validation. Standard deviations are in parentheses.

Dataset	CT-Net	DR-Net	CG	RIPPER	BRS	CART	RF	XGB
Adult	**84.08** (0.46)	82.55 (0.61)	82.60 (0.62)	82.25 (0.85)	78.78 (0.58)	82.44 (0.35)	84.03 (0.55)	84.41 (0.19)
Magic	**84.91** (0.56)	83.91 (0.53)	83.33 (0.59)	82.86 (0.52)	81.37 (0.73)	83.18 (0.44)	86.71 (0.48)	87.16 (0.36)
House	**88.47** (0.40)	86.07 (0.41)	83.80 (0.78)	81.43 (3.19)	83.26 (0.55)	85.10 (0.60)	88.49 (0.19)	88.92 (0.35)
Recidivism	**66.24** (1.00)	64.09 (0.46)	64.57 (0.67)	64.84 (0.36)	61.98 (0.75)	62.85 (0.87)	66.77 (0.66)	64.33 (1.25)
Chess	**91.47** (0.42)	84.47 (0.51)	81.93 (0.50)	85.46 (1.04)	74.66 (2.15)	85.36 (0.40)	92.63 (0.42)	94.98 (0.36)
Retention	**93.85** (0.46)	87.78 (0.37)	90.77 (0.57)	88.92 (0.58)	89.37 (1.60)	90.11 (0.65)	93.43 (0.42)	94.29 (0.28)
Churn	**80.36** (1.30)	78.85 (0.61)	79.21 (1.07)	78.27 (0.39)	76.74 (1.28)	79.00 (0.57)	80.35 (0.93)	77.45 (0.96)
Airline	**95.03** (0.41)	93.32 (0.30)	90.10 (0.31)	93.08 (1.27)	90.71 (0.46)	90.21 (0.43)	94.79 (0.42)	95.94 (0.23)
Heloc	**71.69** (0.80)	71.36 (0.75)	70.05 (0.43)	68.85 (1.19)	70.82 (0.74)	70.00 (1.19)	71.95 (0.67)	70.39 (0.56)
Churn2	85.35 (0.34)	**85.97** (0.07)	85.68 (0.59)	85.07 (0.39)	85.89 (0.55)	84.33 (0.08)	86.05 (0.54)	85.26 (0.68)
Surgical	83.64 (0.75)	**84.78** (0.58)	80.38 (0.58)	83.35 (1.05)	80.25 (0.49)	79.40 (0.59)	82.90 (0.31)	85.26 (0.33)

the dataset, whereas DR-Net, BRS and CG are more recent rule-set-generation classifiers that optimize interpretability and accuracy at the same time. We use the CART (Breiman et al., 1984) algorithm for learning decision trees, whereas random forest (RF) (Breiman, 2001) and XGBoost (XGB) (Chen & Guestrin, 2016) serve as uninterpretable baselines to illustrate the typical performances that black-box models can achieve on the evaluated datasets. We used scikit-learn (Pedregosa et al., 2011) implementations for CART and RF. The implementations of all baseline models are publicly available on GitHub.[4] For all datasets, we encoded the categorical and numerical attributes into binarized features according to the scheme described in Qiao et al. (2021). Moreover, for BRS and CG, negations of the binarized features are appended along with their positive counterparts, e.g., non-smoker vs. smoker, according to the steps described in their experimental sections so that negative literals can be considered in their rule sets, which is required in their papers.

CT-Net Configurations and Parameter Tuning. We used the Adam optimizer with a fixed learning rate of 10^{-3} when evaluating CT-Net. In addition, we incorporated the sparsity-promoting regularization discussed before, so we did not further apply L_2 regularization (weight decay) to the experiments. The neural networks were constructed with 100 neurons in the hidden layer and we let our regularization technique prune the unnecessary neurons. For simplicity, we used a mini-batch size of 2000 and all networks were trained for 2000 epochs to guarantee complete convergence. We employed the nested 5-fold cross-validation to select the parameters that maximize the training accuracy. In particular, each dataset was shuffled (with a fixed seed to ensure the consistency for all approaches) and split into 5 training–testing pairs, for each of which we derived a set of parameters that maximize the accuracy on the training subset. The parameters were then adopted to evaluate the corresponding training–testing pair and the final results were produced by averaging the performance over the 5 pairs. To be specific, we tune the regularization coefficients λ_1 and λ_2 for CT-Net, the minimum number of samples per leaf for CART and RF, the regularization term for XGBoost, and the same parameters for DR-Net, CG, RIPPER, and BRS as discussed in Qiao et al. (2021).

[4] DR-Net (https://github.com/Joeyonng/decision-rules-network); CG (https://github.com/Trusted-AI/AIX360); RIPPER (https://github.com/imoscovitz/wittgenstein); BRS (https://github.com/wangtongada/BOA); scikit-learn (https://github.com/scikit-learn/scikit-learn); xgboost (https://github.com/dmlc/xgboost).

Table 4
Statistics of CT-Net after pruning, where # pruned is the average number of neurons pruned in the hidden layer and sparsity represents the average percentage of the zero weights in the remaining neurons.

Dataset	# Pruned	Sparsity
Adult	97.0	63.28%
Magic	96.8	47.00%
House	97.2	54.21%
Recidivism	79.2	44.75%
Chess	45.4	72.57%
Retention	72.8	71.83%
Churn	97.2	71.20%
Airline	89.6	65.30%
Heloc	98.2	43.05%
Churn2	77.2	79.59%
Surgical	96.8	67.93%

Classification Performance. The classification performances of CT-Net and other baseline approaches were evaluated based on accuracy. In particular, the accuracy is the test accuracy computed based on the nested 5-fold cross-validation as previously explained. The accuracies for all models are summarized in Table 3. As can be observed in Table 3, CT-Net achieves the best test accuracy among all interpretable models across all datasets except for churn2 and surgical, which is close to or even higher than the black-box classifiers on some of the datasets (e.g., adult, recidivism, and retention). This reflects the significant advantage of CT-Net on its generalization capability.

We further analyzed the results in Table 3 using a two-step procedure recommended in Demšar (2006), which consists of a Friedman test to check whether all classifiers perform similarly and a follow-up Nemenyi test to compare pairs of the classifiers. Applying the Friedman test, we derived Friedman statistic to be 40.39, which is larger than the critical value of the chi-squared distribution with 7 degrees of freedom $\chi^2_7 = 14.07$ for $\alpha = 0.05$. Thus we can reject the null hypothesis that all classifiers tested have the equal performance. Then we continue to use Nemenyi test to compare whether there is significant difference between all pairs of the classifiers and the results are shown in Fig. 2. As we can see from the figure, CT-Net has the second highest average rank that is positioned between Random Forest and XGBoost, proving that CT-Net can achieve the state-of-the-art predictive performances similar to other black-box models. Compared with the interpretable models, CT-Net has shown to be statistically significantly better than RIPPER, CART, and BRS in terms of the testing accuracy, which demonstrates that CT-Net can be considered as a great alternative to other interpretable models.

Effects of Sparsity-Promoting Regularization. In our evaluation, the number of threshold functions, i.e., the number of effective neurons in the hidden layer, can be up to 100 depending on the parameter tuning, which at the same time guarantees enough generalization capacity and increases the complexity. As discussed in Section 5, we employed the sparsity-promoting regularization technique in our training procedure to attenuate overfitting and simplify the threshold functions. In particular, our regularization approach can not only sparsify the hidden layer, but also directly turn off the entire threshold function by pruning its corresponding weight in the output layer. We evaluated the performance of our regularization technique in both ways and the results are summarized in Table 4. As can be observed, the number of pruned neurons varies significantly from 45% (chess) to 98% (heloc), which validates the effectiveness of our regularization approach in removing redundant capacity while preserving the necessary neurons based on the complexity of the dataset. Further, the average sparsity of the remaining neurons is generally greater than

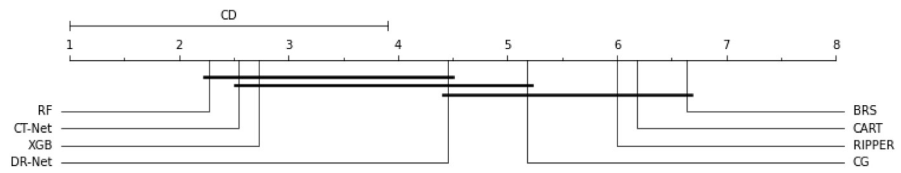

Fig. 2. Results of Nemenyi test for all classifiers. Groups of classifiers that are not significantly different at $\alpha = 0.1$ are connected.

50% except for the magic, recidivism, and heloc datasets, indicating the training procedure effectively excludes any literals that have little contribution to the prediction. In particular, it can be seen that while a relatively small number of neurons are pruned for chess, the remaining sparsity is higher than other datasets. This suggests that regularization can capture the underlying logic of the datasets and find a correct direction to simplify the neural network without severely hurting the generalization.

7. Conclusion

In this paper, we proposed a three-layer neural network architecture called CT-Net that can be trained for classifying tabular data to achieve high prediction accuracy. In particular, the trainable hidden layer neurons with step activation function logically correspond to a set of threshold logic functions, while the output layer further constructs a conjunction of these threshold functions. In addition, once the network is trained, for any positive prediction, a provably minimal explanation can be readily derived from the model. We further adopt a sparsity-promoting regularization technique to sparsify the network and simplify the threshold functions. Experimental results demonstrate that our approach has significant advantages on producing accuracy predictions over other state-of-the-art interpretable decision models.

Several potential improvements can be developed in the future work. First, while our proposed output layer essentially performs a logical conjunction (AND), other logic operations can also be used in place of the conjunctive operation, including OR, XOR, or simply a standard fully-connected layer. Second, multiple proposed layers that encode a logical operation can be stacked together to provide higher network capacities. Third, the current work is focused on binary classification with a single output neuron. Future work may extend the network to be a multi-class classifier by increasing the number of output neurons with additional modifications.

CRediT authorship contribution statement

Weijia Wang: Methodology, Formal analysis, Writing – original draft. **Litao Qiao:** Software, Validation, Data curation, Writing – original draft. **Bill Lin:** Conceptualization, Writing – review & editing Supervision, Funding acquisition.

Declaration of competing interest

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

Data availability

Data will be made available on request.

Acknowledgments

This research has been supported in part by the National Science Foundation, United States (NSF IIS-1956339).

References

Abutbul, A., Elidan, G., Katzir, L., & El-Yaniv, R. (2020). DNF-net: A neural architecture for tabular data. CoRR, abs/2006.06465, Retrieved from URL https://arxiv.org/abs/2006.06465.

Arik, S. Ö., & Pfister, T. (2019). TabNet: Attentive interpretable tabular learning. CoRR, abs/1908.07442, Retrieved from URL http://arxiv.org/abs/1908.07442.

Arya, V., Bellamy, R. K. E., Chen, P.-Y., Dhurandhar, A., Hind, M., Hoffman, S. C., Houde, S., Liao, Q. V., Luss, R., Mojsilović, A., Mourad, S., Pedemonte, P., Raghavendra, R., Richards, J., Sattigeri, P., Shanmugam, K., Singh, M., Varshney, K. R., Wei, D., & Zhang, Y. (2019). One explanation does not fit all: A toolkit and taxonomy of ai explainability techniques. Retrieved from https://arxiv.org/abs/1909.03012.

Audemard, G., Koriche, F., & Marquis, P. (2020). On Tractable XAI Queries based on Compiled Representations. In *Proceedings of the 17th international conference on principles of knowledge representation and reasoning* (pp. 838–849). http://dx.doi.org/10.24963/kr.2020/86, Retrieved from https://doi.org/10.24963/kr.2020/86.

Bengio, Y., Léonard, N., & Courville, A. C. (2013). Estimating or propagating gradients through stochastic neurons for conditional computation. CoRR, abs/1308.3432, Retrieved from http://arxiv.org/abs/1308.3432.

Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5–32. http://dx.doi.org/10.1023/A:1010933404324, Retrieved from.

Breiman, L., Friedman, J., Stone, C., & Olshen, R. (1984). *Classification and regression trees*. Taylor & Francis, Retrieved from https://books.google.com/books?id=JwQx-WOmSyQC.

Candes, E. J., Wakin, M. B., & Boyd, S. P. (2007). Enhancing sparsity by reweighted L1 minimization. http://dx.doi.org/10.48550/ARXIV.0711.1612, arXiv, Retrieved from https://arxiv.org/abs/0711.1612.

Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. CoRR, abs/1603.02754, Retrieved from http://arxiv.org/abs/1603.02754.

Chen, C., Lin, K., Rudin, C., Shaposhnik, Y., Wang, S., & Wang, T. (2018). An interpretable model with globally consistent explanations for credit risk. CoRR, abs/1811.12615, Retrieved from http://arxiv.org/abs/1811.12615.

Choi, A., Shi, W., Shih, A., & Darwiche, A. (2017). Compiling neural networks into tractable boolean circuits. *Intelligence*.

Cohen, W. W. (1995). Fast effective rule induction. In *Proceedings of the twelfth international conference on machine learning* (pp. 115–123). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc..

Courbariaux, M., & Bengio, Y. (2016). BinaryNet: Training deep neural networks with weights and activations constrained to +1 or -1. CoRR, abs/1602.02830, Retrieved from http://arxiv.org/abs/1602.02830.

Dash, S., Günlük, O., & Wei, D. (2018). Boolean decision rules via column generation. CoRR, abs/1805.09901, Retrieved from http://arxiv.org/abs/1805.09901.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, *7*, 1–30.

Dua, D., & Graff, C. (2017). UCI machine learning repository. Retrieved from http://archive.ics.uci.edu/ml.

Han, S., Pool, J., Tran, J., & Dally, W. J. (2015). Learning both weights and connections for efficient neural networks. CoRR, abs/1506.02626, Retrieved from http://arxiv.org/abs/1506.02626.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep residual learning for image recognition. CoRR, abs/1512.03385, Retrieved from http://arxiv.org/abs/1512.03385.

Ignatiev, A., Narodytska, N., & Marques-Silva, J. (2018). Abduction-based explanations for machine learning models. CoRR, abs/1811.10656, Retrieved from http://arxiv.org/abs/1811.10656.

Izza, Y., & Marques-Silva, J. (2021). On explaining random forests with SAT. CoRR, abs/2105.10278, Retrieved from https://arxiv.org/abs/2105.10278.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T.-Y. (2017). LightGBM: A highly efficient gradient boosting decision tree. In *Proceedings of the 31st international conference on neural information processing systems* (pp. 3149–3157). Red Hook, NY, USA: Curran Associates Inc..

Lakkaraju, H., Bach, S. H., & Leskovec, J. (2016). 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* (pp. 1675–1684). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/2939672.2939874, Retrieved from.

Letham, B., Rudin, C., McCormick, T. H., & Madigan, D. (2015). Interpretable classifiers using rules and Bayesian analysis: Building a better stroke prediction model. *The Annals of Applied Statistics*, *9*(3), http://dx.doi.org/10.1214/15-aoas848, Retrieved from.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, *12*(null), 2825–2830.

Qiao, L., Wang, W., & Lin, B. (2021). Learning accurate and interpretable decision rule sets from neural networks. CoRR, abs/2103.02826, Retrieved from https://arxiv.org/abs/2103.02826.

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "Why should I trust you?": Explaining the predictions of any classifier. http://dx.doi.org/10.48550/ARXIV.1602.04938, arXiv. Retrieved from https://arxiv.org/abs/1602.04938.

Ribeiro, M. T., Singh, S., & Guestrin, C. (2018). Anchors: High-precision model-agnostic explanations. In *AAAI*. Retrieved from https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16982.

Rivest, R. L. (1987). Learning decision lists. *Machine Learning*, *2*(3), 229–246.

Rudin, C. (2018). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. http://dx.doi.org/10.48550/ARXIV.1811.10154, arXiv. Retrieved from https://arxiv.org/abs/1811.10154.

Schmidt, P., & Witte, A. D. (1988). *Predicting recidivism in north Carolina, 1978 and 1980*. Inter-university Consortium for Political and Social Research.

Shi, W., Shih, A., Darwiche, A., & Choi, A. (2020). On tractable representations of binary neural networks.

Shih, A., Choi, A., & Darwiche, A. (2018). A symbolic approach to explaining Bayesian network classifiers. http://dx.doi.org/10.48550/ARXIV.1805.03364, arXiv. Retrieved from https://arxiv.org/abs/1805.03364.

Vanschoren, J., van Rijn, J. N., Bischl, B., & Torgo, L. (2013). Openml: Networked science in machine learning. *SIGKDD Explorations*, *15*(2), 49–60. http://dx.doi.org/10.1145/2641190.2641198, Retrieved from.

Wang, T., Rudin, C., Doshi-Velez, F., Liu, Y., Klampfl, E., & MacNeille, P. (2017). A Bayesian framework for learning rule sets for interpretable classification. *Journal of Machine Learning Research*, *18*(70), 1–37, Retrieved from http://jmlr.org/papers/v18/16-003.html.