Robust Optimal Classification Trees against Adversarial Examples

Daniël Vos, Sicco Verwer

Delft University of Technology
d.a.vos@tudelft.nl, s.e.verwer@tudelft.nl

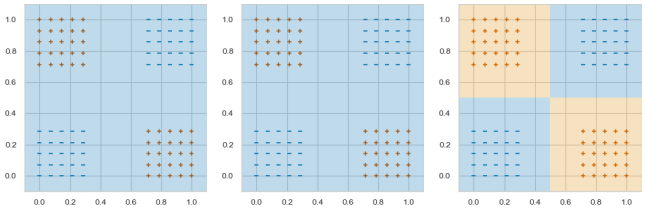
Abstract

Decision trees are a popular choice of explainable model, but just like neural networks, they suffer from adversarial examples. Existing algorithms for fitting decision trees robustly against adversarial examples are greedy heuristics and lack approximation guarantees. In this paper we propose ROCT, a collection of methods to train decision trees that are optimally robust against user-specified attack models. We show that the min-max optimization problem that arises in adversarial learning can be solved using a single minimization formulation for decision trees with 0-1 loss. We propose such formulations in Mixed-Integer Linear Programming and Maximum Satisfiability, which widely available solvers can optimize. We also present a method that determines the upper bound on adversarial accuracy for any model using bipartite matching. Our experimental results demonstrate that the existing heuristics achieve close to optimal scores while ROCT achieves state-of-the-art scores.

Introduction

While breakthroughs in machine learning research have enabled training of powerful predictive models, most models are still vulnerable to adversarial examples, samples with tiny perturbations that cause them to be misclassified. Since the discovery of adversarial examples in neural networks (Szegedy et al. 2013) much work has gone into training models that are robust to these attacks and recently, the first efforts were made to train robust decision trees against adversarial examples (Chen et al. 2019; Calzavara et al. 2020; Vos and Verwer 2020). However, the current methods are greedy and offer no performance guarantees. They can fail on arbitrary datasets and give results no better than random guessing (Figure 1).

In decision tree learning, there has been an increased interest in optimal learning algorithms (Carrizosa, Molero-Río, and Morales 2021). Although the problem of learning decision trees is NP-complete (Laurent and Rivest 1976), these methods can produce optimally accurate decision trees for many (typically small) datasets. Most methods translate the problem to well-known frameworks such as Mixed-Integer Linear Programming (Bertsimas and Dunn 2017; Verwer and Zhang 2017), Boolean Satisfiability (Narodytska et al. 2018; Avellaneda 2020), and Constraint Programming (Verhaeghe et al. 2020).



(a) GROOT (b) TREANT (c) ROCT (ours)

Figure 1: Existing methods (a)(b) greedily optimize one split at a time and cannot find a good tree to fit the XOR-shaped data. ROCT optimizes the entire tree at once and finds the optimal tree that exactly fits the dataset.

In this work, we combine these lines of research and propose Robust Optimal Classification Trees (ROCT), a method to train decision trees that are optimally robust against user-specified adversarial attack models. This model is robust in the sense that it predicts the correct ground-truth label in a box of specified size surrounding each sample, this optimizes robustness against corrupted instances (Diochnos, Mahloujifar, and Mahmoody 2018). Like existing robust decision tree learning algorithms (Calzavara et al. 2020; Vos and Verwer 2020), ROCT allows users to specify a box-shaped attack model that encodes an attacker's capability to modify feature values with the aim of maximizing loss. Existing robust decision tree learning methods use a greedy node splitting approach. Other robust learning algorithms such as adversarial training (Madry et al. 2017) solve the inner maximization (adversarial attacks) and the outer minimization problems (minimize expected loss) separately. In this work we prove that this separation is not needed in the case of decision trees. We provide a formulation that solves the problem of fitting robust decision trees exactly in a single minimization step for trees up to a given depth.

ROCT[1] uses a novel translation of the problem of fitting robust decision trees into Mixed-Integer Linear Programming (MILP) or Maximum Satisfiability (MaxSAT)

[1] https://github.com/tudelft-cda-lab/ROCT

formulations. We also propose a new upper-bound calculation for the adversarial accuracy of any machine learning model based on bipartite matching, which can be used to choose appropriate attack models for experimentation. Our results show that ROCT trees optimized with a warm-started MILP solver achieve state-of-the-art adversarial accuracy scores compared to existing methods on 8 datasets. Moreover, given sufficient solver time, ROCT provably finds an optimally robust decision tree. In our experiments, ROCT was able to fit and prove optimality of depth 2 decision trees on six datasets. Where there are no known approximation bounds on the performance of existing heuristic methods for fitting robust decision trees, our results demonstrate that they are empirically close to optimal.

Background and Related Work

Mixed-Integer Linear Programming

Mixed-Integer Linear Programming (MILP) is a variation of Linear Programming in which some variables are integer or binary. The goal of these formulations is to optimize a linear function under linear constraints. While the problem is generally NP-hard there exist many fast MILP solvers. In this paper, we translate the robust decision tree learning problem into a MILP formulation and solve it using GUROBI[2] 9. Since MILP solvers usually become less efficient with more integer variables, we introduce two formulations that differ in the number of such variables.

Maximum Satisfiability Solving

Maximum Satisfiability (MaxSAT) is an optimization version of the classical boolean satisfiability problem (SAT). In MaxSAT, problems are modeled as a set of hard clauses that have to be satisfied and a set of soft clauses of which the solver tries to satisfy as many as possible. One advantage of MaxSAT solvers is their availability, with many state-of-the-art solvers available as open source programs. In this work we use the PySat implementation (Alexey, Antonio, and Joao 2018) of the Linear Sat-Unsat (LSU) algorithm (Morgado et al. 2013) improved with incremental cardinality constraints (Martins et al. 2014), and the RC2 algorithm (Ignatiev, Morgado, and Marques-Silva 2019). These algorithms differ in the direction in which they optimize, LSU starts with a poor solution and creates increasingly optimal solutions over time while RC2 starts by attempting to satisfy all soft clauses then relaxes this constraint until it finds a solution. Both algorithms use the Glucose[3] 4.1 SAT solver.

Optimal Decision Trees

Most popular decision tree learning algorithms such as CART (Breiman et al. 1984), ID3 (Quinlan 1986) and C4.5 (Quinlan 1993) are greedy and can return arbitrarily bad trees (Kearns 1996). In recent years, there has been extensive effort to train optimal trees. One of the earliest works (Bertsimas and Shioda 2007) proposes a MILP formulation for finding a tree of a given maximum depth that uses clustering to reduce the dataset size. Independently, (Nijssen and Fromont 2010) maps this problem for the restricted case of Boolean decision nodes to itemset mining. Several years later the first MILP formulations were proposed for the full problem (Bertsimas and Dunn 2017) and (Verwer and Zhang 2017). The latest methods improve these works using non-crisp decision boundaries (Rhuggenaath et al. 2018), a binary encoding (Verwer and Zhang 2019), new analytical bounds and an improved tree representation translation (Hu, Rudin, and Seltzer 2019), by translating to CP (Verhaeghe et al. 2020), using dynamic programming with search (Demirović et al. 2020), by caching branch-and-bound (Aglin, Nijssen, and Schaus 2020), and optimized randomization (Blanquero et al. 2021). In this work, we build on these works to create the first formulation for optimal learning of robust decision trees.

Robust Decision Trees

Previous works have already put effort into fitting decision trees that are more robust to adversarial perturbations than the trees created by regular decision tree algorithms. (Kantchelian, Tygar, and Joseph 2016) defines a MILP formulation for finding adversarial examples in decision tree ensembles and used these samples to fit an ensemble of more robust decision trees. (Chen et al. 2019) adapts greedy decision tree learning algorithms by using the worst case score functions under attacker influence to fit more robust trees against L_∞ norm bounded attackers. Later, TREANT (Calzavara et al. 2020) uses a more flexible greedy algorithm that could optimize arbitrary convex score functions under attacker influence and allowed users to describe attacker capabilities using axis-aligned rules. This flexibility comes at a cost in run-time, as it uses an iterative solver to optimize this score for each split it learns. GROOT (Vos and Verwer 2020) improve the greedy procedure by efficiently computing the worst-case Gini impurity and allowing users to specify box-shaped attacker perturbation limits. In this paper, we compare against GROOT and TREANT as these greedy methods achieve state-of-the-art scores.

ROCT: Robust Optimal Classification Trees

When training robust classifiers we find ourselves in a competition with the adversary. Madry et al. (Madry et al. 2017) present the robust learning problem as the following min-max optimization problem:

$$\min_{\theta} \mathbb{E}_{(x,y)\sim D} \left(\max_{\delta \in S} L(\theta, x+\delta, y) \right) \tag{1}$$

Like in traditional machine learning, the goal is to find model parameters θ that minimize the expected loss $L(\theta, x, y)$ over feature x and class y variables from distribution D (outer minimization). This minimization takes into account that an attacker aims to maximize this loss by changing samples (x, y) from D with perturbation $\delta \in S$ (inner maximization), where S is a predefined set of allowed perturbations. Intuitively, the min-max nature of training robust models makes it a much more challenging optimization

[2]https://www.gurobi.com/

[3]https://www.labri.fr/perso/lsimon/glucose/

problem than regular learning. For example in adversarial training (Madry et al. 2017) one approximately optimizes this function by incorporating expensive adversarial attacks into the training procedure. In this work, we demonstrate that this intuition is wrong when learning decision trees.

Let $\mathcal{T}$ denote a decision tree that maps any data point x to a leaf node $t = \mathcal{T}(x)$ and assigns c_t as its prediction. $\mathcal{T}_L$ denotes the set of leaf nodes. A leaf node t represents a box in feature space $t_f = \{x' \in \mathbb{R}^p \mid t = \mathcal{T}(x)\}$. When this set intersects with the space of possible perturbations $S(x) = \{x + \delta \mid \delta \in S\}$, we say t is reachable and denote the set of reachable leafs using $\mathcal{T}_L^{S(x)} = \{t \in \mathcal{T}_L \mid t_f \cap S(x) \neq \emptyset\}$. We now present Robust Optimal Classification Trees (ROCT), which turns Equation 1 into a single minimization problem that can be solved using combinatorial optimization:

Theorem 1. *Robust learning (Equation 1) with 0-1 loss in the case of binary classification trees is equivalent to:*

$$\min_{\theta} \sum_{(x,y)\sim D} \left[\bigvee_{t \in \mathcal{T}_L^{S(x)}} c_t \neq y \right]$$

Proof. For 0-1 loss $L_{0\text{-}1}$, Equation 1 is equivalent to:

$$\min_{\theta} \sum_{(x,y)\sim D} \left(\max_{\delta \in S} L_{0\text{-}1}(\theta, x+\delta, y) \right)$$

Any perturbation in the inner maximization $\max_{\delta \in S}$ such that $\mathcal{T}(x) = \mathcal{T}(x+\delta)$ gives the same classification outcome for 0-1 loss. The maximization over all $\delta \in S$ can therefore be replaced by a maximization over all reachable leaf nodes $t \in \mathcal{T}_L^{S(x)}$. By definition, the 0-1 loss term is equivalent to the absolute difference $|c_t - y|$ of prediction c_t and label y, which gives:

$$\min_{\theta} \sum_{(x,y)\sim D} \left(\max_{t \in \mathcal{T}_L^{S(x)}} |c_t - y| \right)$$

The term $|c_t - y|$ takes value 1 when $c_t \neq y$ and 0 otherwise. When any of the reachable leaves $t \in \mathcal{T}_L^{S(x)}$ predict $c_t \neq y$, the inner maximization becomes 1. This is equivalent to the disjunction over $c_t \neq y$ for all reachable leaves:

$$\min_{\theta} \sum_{(x,y)\sim D} \left[\bigvee_{t \in \mathcal{T}_L^{S(x)}} c_t \neq y \right]$$

□

ROCT solves this formulation in one shot using discrete optimization solvers. We present 6 versions that vary in the kind of solver (MILP or MaxSAT) and type of variables used to represent splitting thresholds, see Table 2.

Attack Model

We assume the existence of a white-box adversary that can move all samples within a box-shaped region around each sample. This box-shaped region is defined by two vectors Δ^l and Δ^r from $\mathbb{R}^n$ specifying for each feature $i \in [1, n]$ how much i can be decreased and increased respectively, i.e., $S = \{\delta \in \mathbb{R}^n : \forall_{1 \leq i \leq n} \Delta_i^l \leq \delta_i \leq \Delta_i^r\}$. For the ease of our formulation we scale all feature values to be in the range $[0, 1]$ which means that the values in Δ^l and Δ^r encode distance as a fraction of the feature range. While our encoding is more flexible, we only test on attack models where $\Delta^l = \Delta^r = (\epsilon, \ldots, \epsilon)$, encoding an L_∞ norm with ϵ perturbation radius. This allows us to easily evaluate performance against a variety of attacker strengths.

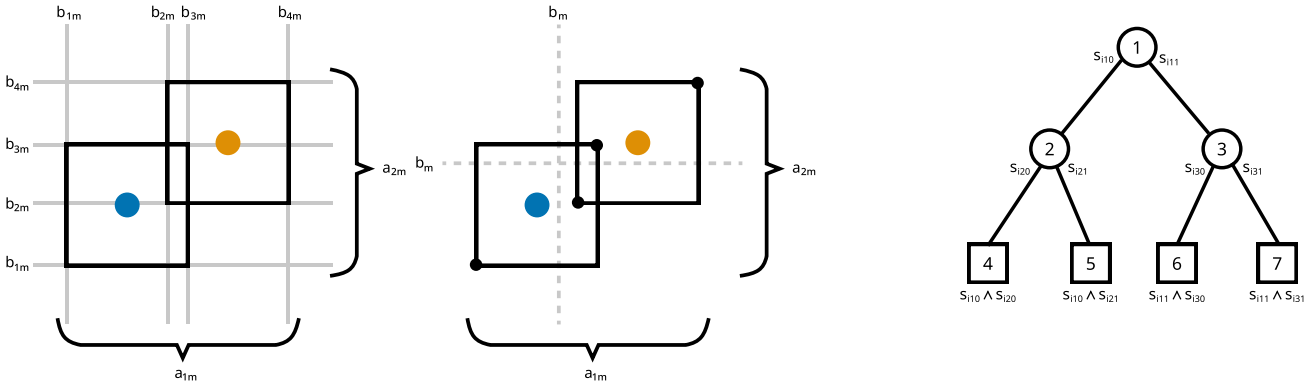
Symbol	Type	Definition
a_{jm}	variable	node m splits on feature j
b_{vm}	variable	node m's threshold is left/right of v
b'_m	variable	node m's continuous threshold value
c_t	variable	leaf node t predicts class 0 or 1
s_{im0}	variable	sample i can move left of node m
s_{im1}	variable	sample i can move right of node m
e_i	variable	sample i can be misclassified
X_{ij}	constant	value of data row i in feature j
y_i	constant	class label of data row i
Δ_j^l	constant	left perturbation range for feature j
Δ_j^r	constant	right perturbation range for feature j
n	constant	number of samples
p	constant	number of features
$A(t)$	set	ancestors of node t
$A_l(t)$	set	... with left branch on the path to t
$A_r(t)$	set	... with right branch on the path to t
S	set	all possible perturbations
$S(x)$	set	... applied to sample x
$\mathcal{T}_B$	set	all decision nodes
$\mathcal{T}_L$	set	all leaf nodes
$\mathcal{T}_L^{S(x)}$	set	... that intersect with $S(x)$
V_j	set	unique values in feature j

Table 1: Summary of the notation used throughout the paper.

Intuition

We borrow much of the notation from OCT (Bertsimas and Dunn 2017), summarized in Table 1. Figure 2 visualizes the variables in ROCT and Figure 3 shows an example of the constraints for a single sample and a tree of depth 1. In the regular learning setting where samples cannot be perturbed by an adversary, samples can only propagate to the left or right child of decision node. In the adversarial setting, samples can permute and are able to reach both the left and right sides, i.e. s_{im0} and s_{im1} can be true at the same time.

Given the attacker capabilities Δ^l and Δ^r, we create the constraints to set the variables s. To determine whether sample X_i can move left of the chosen split we can decrease its feature values as far as the attacker capabilities allow ($X_i - \Delta^l$) and see if it reaches the left side. Similarly to see if it reaches the right side we increase the feature values maximally ($X_i + \Delta^r$). We give two kinds of constraints for determining these s variables that differ in whether decision



(a) Decision node m (binary) (b) Decision node m (continuous) (c) Path variables for sample i

Figure 2: Example of ROCT's formulation. For each decision node the a variables select a splitting feature and b select the threshold value. b can be defined as multiple binary (a) or a single continuous (b) variable. Using the s variables (c) ROCT traces all sample paths through the tree to the leaves and counts an error if any reachable leaf predicts the wrong class.

thresholds are represented by binary or continuous variables.

Continuous Decision Thresholds To select a threshold value an intuitive method is to create a continuous variable b_m for every decision node. We can then use this variable to determine the values s_{im0} and s_{im1} by checking whether $X_i - \Delta^l$ and $X_i + \Delta^r$ can reach the left and right side of the threshold respectively. We create the following constraints:

$$(\mathbf{X_i} - \mathbf{\Delta^l}) \cdot \mathbf{a_m} \leq b'_m \implies s_{im0}$$

$$(\mathbf{X_i} + \mathbf{\Delta^r}) \cdot \mathbf{a_m} > b'_m \implies s_{im1}$$

Since these constraints use a dot product with continuous variables it is not possible to implement this in MaxSAT. Another challenge comes with the second constraint being a strict inequality which is not directly supported in MILP. Like (Bertsimas and Dunn 2017), we add a small value to the right hand side to turn it into a regular inequality.

Binary Decision Thresholds We create a set of variables b_{vm} for each unique decision threshold value v, with v in ascending order. Instead of forcing one of them to `true`, we create an ordering in the variables such that if one threshold variable is `true`, the larger variables also become `true`:

$$b_{vm} \implies b_{(v+1)m}$$

Intuitively if b_{vm} is set to `true` a sample with feature value v will be sent to the right of the split and when b_{vm} is `false` it will be sent to the left. A useful property of this constraint is that we only have to encode the local influence of a threshold variable b_{vm} on close-by data points, the rest is forced by the chain of constraints. For each feature j we determine what threshold values v^l and v^r correspond to $X_{ij} - \Delta^l_j$ and $X_{ij} + \Delta^r_j$ and check whether their b_{vm} values indicate that the sample can reach the left / right side:

$$a_{jm} \wedge \neg b_{v^l m} \implies s_{im0}$$

$$a_{jm} \wedge b_{v^r m} \implies s_{im1}$$

Method	Threshold formulation	Solver	Init. with GROOT
LSU-MaxSAT	binary	LSU (glucose 4.1)	
RC2-MaxSAT	binary	RC2 (glucose 4.1)	
Binary-MILP	binary	GUROBI 9	
Binary-MILP-warm	binary	GUROBI 9	✓
MILP	continuous	GUROBI 9	
MILP-warm	continuous	GUROBI 9	✓

Table 2: Summary of introduced methods, they differ in solver type and whether thresholds are formulated with binary or continuous variables. The 'warm' methods are initialized with the GROOT heuristic.

Selecting Features Consider a single decision node m, such a decision node needs to decide a feature to split on. We create a binary variable a_{jm} for each feature j and force that exactly one of these variables can be equal to 1:

$$\sum_{j=1}^{p} a_{jm} = 1$$

This constraint can be relaxed to $\sum_{j=1}^{p} a_j \geq 1$ as selecting more than one feature can only make more s variables `true` and thus can only increase the number of errors.

Counting errors We create a variable e_i for each sample i which is true when any reachable leaf $t \in \mathcal{T}_L^{S(x)}$ (see Theorem 1) predicts the other class. These leaves are found by following all paths a sample can take through the tree using the s_{im0} and s_{im1} variables. This is visualized in Figure 2c. Sample i can reach leaf t when the val-

ues $s_{im\ldots}$ are `true` for all nodes m on the path to t, i.e. $\bigwedge_{m\in A_l(t)} s_{im0} \bigwedge_{m\in A_r(t)} s_{im1}$. Here $A_l(t)$ refers to the set of ancestors of leaf t of which we follow the path through its left child and $A_r(t)$ for child nodes on the right. When sample i can reach leaf t and its label does not match t's prediction ($y_i \neq c_t$), force e_i to `true`:

$$\bigwedge_{m\in A_l(t)} s_{im0} \bigwedge_{m\in A_r(t)} s_{im1} \wedge (c_t \neq y_i) \implies e_i$$

With one constraint per decision leaf and sample combination this determines the e values. To then turn all possible paths into predictions we need to assign a prediction label to each decision leaf. Each leaf t gets a variable c_t where `false` means class 0 and `true` means class 1.

Objective Function Our goal is to minimize the equation from Theorem 1. This is equivalent to minimizing the sum of errors e_i ($i = 1...n$). We convert this MILP objective to MaxSAT by adding a soft constraint $\neg e_i$ for each sample and maximizing the number of correctly predicted samples:

$$\text{maximize} \quad \sum_{i=1}^{n} \neg e_i \quad \text{or} \quad \text{minimize} \quad \sum_{i=1}^{n} e_i$$

Complete Formulation

Below we give the full formulation for ROCT, in Table 1 we summarize the notation used. The equations can easily be formulated as MILP or MaxSAT instances, for MILP this was done with big-M constraints.

$$\text{min.} \sum_{i=1}^{n} e_i$$

subject to:

$$\sum_{j=1}^{p} a_{jm} = 1, \qquad \forall m \in \mathcal{T}_B$$

$$b_{vm} \Rightarrow b_{(v+1)m}, \qquad \forall m \in \mathcal{T}_B, v{=}1..|V_j|-1$$

$$\bigwedge_{m\in A_l(t)} s_{im0} \bigwedge_{m\in A_r(t)} s_{im1} \wedge [c_t{\neq}y_i] \Rightarrow e_i, \qquad \forall t \in \mathcal{T}_L, i{=}1..n$$

continuous threshold variables:

$$(\mathbf{X_i} - \mathbf{\Delta^l}) \cdot \mathbf{a_m} \leq b'_m \Rightarrow s_{im0} \qquad \forall m \in \mathcal{T}_B, i{=}1..n$$

$$(\mathbf{X_i} + \mathbf{\Delta^r}) \cdot \mathbf{a_m} > b'_m \Rightarrow s_{im1} \qquad \forall m \in \mathcal{T}_B, i{=}1..n$$

binary threshold variables:

$$a_{jm} \wedge \neg b_{v^l m} \Rightarrow s_{im0}, \qquad \forall m \in \mathcal{T}_B, i{=}1..n, j{=}1..p$$

$$a_{jm} \wedge b_{v^r m} \Rightarrow s_{im1}, \qquad \forall m \in \mathcal{T}_B, i{=}1..n, j{=}1..p$$

In both the continuous and binary threshold formulations the size of the instances is dominated by the constraints setting the s variables. In the continuous case this size is of complexity $\mathcal{O}(2^d n)$ where d is the depth of the tree and n the number of samples. For the binary threshold case the size complexity is $\mathcal{O}(2^d np)$ where p is the number of features. The solvers run in (worst-case) exponential time.

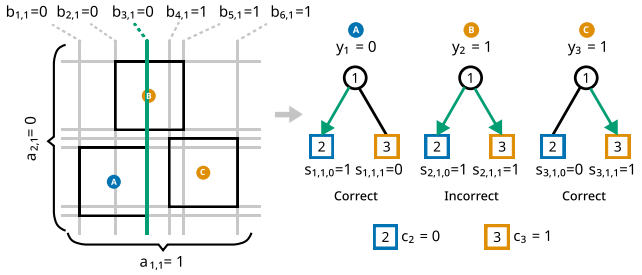


Figure 3: Example of a decision tree of depth 1 with the binary threshold formulation. Sample A and C get correctly classified since all their reachable leaves predict the correct label. Sample B reaches both leaves, since the left leaf predicts the wrong label, B gets misclassified.

Example For clarity we give a small example of a decision tree of depth 1 that we fit on 3 samples. In Figure 3, we show three data points $A{=}(0.2, 0.2)$, $B{=}(0.5, 0.8)$, $C{=}(0.8, 0.3)$ and all their feature values can be perturbed within a L_∞ norm radius 0.2. This results in feature 1 taking one of the 6 possible threshold values: $\{0.0, 0.3, 0.4, 0.6, 0.7, 1.0\}$ (due to bounding boxes). Suppose the solver selects feature 1 for decision node 1: $a_{1,1} = 1$ and $a_{2,1} = 0$. Suppose the solver selects the third threshold value: $b_{1,1} = b_{1,2} = b_{1,3} = 0$ and $b_{1,4} = b_{1,5} = b_{1,6} = 1$ (note this is a unary encoding). Due to the binary threshold constraints we then obtain:

$$a_{1,1} \wedge \neg b_{1,1} \Rightarrow s_{1,1,0}, \quad 1 \wedge \neg 0 \Rightarrow s_{1,1,0}{=}1$$

Thus, the first data point (A) can move to the left of decision node 1, since $b_{1,1} = 0$. From Figure 3, we see that $b_{1,1} = 0$ implies the decision threshold is to the right of the lower bound of the bounding box for point A. Hence indeed, it should be able to move left. Similarly for the upper bound:

$$a_{1,1} \wedge b_{3,1} \Rightarrow s_{1,1,1}, \quad 1 \wedge 0 \Rightarrow s_{1,1,1}{\in}\{0, 1\}$$

Thus, since $b_{3,1} = 0$, the constraints pose no restriction on whether point A can move to right of decision node 1. The correct behavior (A cannot move to the right) is forced by the objective function, which can only become worse by setting $s_{1,1,1} = 1$. The remaining s variables become:

$$a_{1,1} \wedge \neg b_{2,1} \Rightarrow s_{2,1,0}, \quad 1 \wedge \neg 0 \Rightarrow s_{2,1,0}{=}1$$

$$a_{1,1} \wedge b_{5,1} \Rightarrow s_{2,1,1}, \quad 1 \wedge 1 \Rightarrow s_{2,1,1}{=}1$$

$$a_{1,1} \wedge b_{6,1} \Rightarrow s_{3,1,1}, \quad 1 \wedge 1 \Rightarrow s_{3,1,1}{=}1$$

$s_{3,1,0}$ remains unconstrained and can therefore be set to 0 by the solver. Since $c_1 = y_1$ and $c_3 = y_3$, e_1 and e_3 are unconstrained and minimized to 0 by the solver, and since $s_{2,0} = s_{2,1} = 1$ the constraints force $e_2 = 1$. The second sample is hence misclassified (it reaches at least one leaf with a prediction value different than its label). Note that, although the thresholds in Figure 3 are always exactly on the perturbation ranges of a sample, we post-process these to maximize the margin.

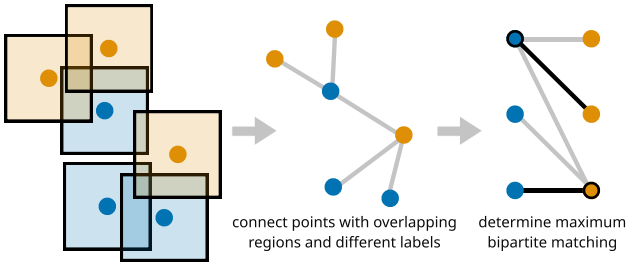


Figure 4: Computing a bound on adversarial accuracy by maximum matching. The maximum matching and minimum vertex cover are shown in black. Since the matching has a cardinality of 2 it is impossible to misclassify fewer than 2 samples when accounting for perturbations.

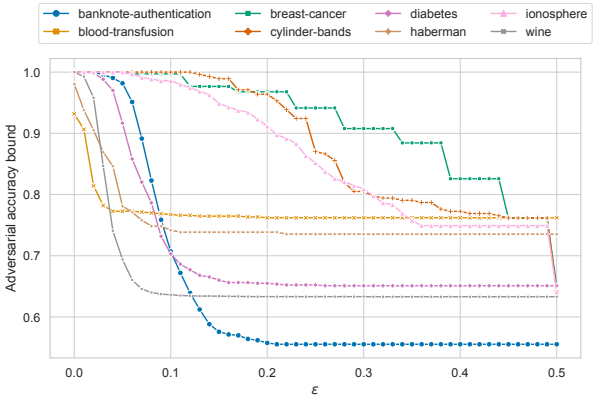


Figure 5: Varying the L_∞ perturbation radius ϵ and computing the adversarial accuracy bound. Datasets are affected differently, e.g. ϵ=0.1 has no effect on cylinder-bands while the bound for blood-transfusion shows that it is not possible to score better than constantly predicting its majority class.

Upper Bound on Adversarial Accuracy

In a regular learning setting with stationary samples one strives for a predictive accuracy of 100%. As long as there are no data points with different labels but same coordinates achieving this score is theoretically possible. However, we realize that in the adversarial setting a perfect classifier cannot always score 100% accuracy as samples can be perturbed. We present a method to compute the upper bound on adversarial accuracy using a bipartite matching that can be computed regardless of what model is used. We use this bound to choose better ϵ values for our experiments. It also lets us compare the scores of optimal decision trees to a score that is theoretically achievable by perfect classifiers. Such a matching approach was also used in (Wang, Jha, and Chaudhuri 2018) to train robust kNN classifiers.

Theorem 2. *The maximum cardinality bipartite matching between samples with overlapping perturbation range and different labels* $\{(i,j) : S_i \cap S_j \neq \emptyset \wedge y_i \neq y_j\}$ *gives an upper bound to the adversarial accuracy achievable by any model for binary decision problems.*

Proof. The reduction to maximum bipartite matching is based on the realization that when the perturbation ranges of two samples with different labels overlap it is not possible to predict both of these samples correctly. A visual explanation is given in Figure 4. Formally, given a classifier C that maps samples to a class 0 or 1, a sample i can only be correctly predicted against an adversary if its entire perturbation range S_i is correctly predicted:

$$\forall x \in S_i : C(x) = y_i \tag{2}$$

Now given a sample j of a different class (e.g. $y_i = 0$ and $y_1 = 1$) that has an overlapping perturbation range such that $S_i \cap S_j \neq \emptyset$, it is clear that Equation 2 cannot simultaneously hold for both samples. We create a bipartite graph $G = (V_0, V_1, E)$ with $V_0 = \{i : y_i = 0\}$ and $V_1 = \{i : y_i = 1\}$, i.e., vertices representing samples of class 0 on one side and class 1 on the other. We then connect two vertices with an edge if their perturbation ranges overlap and their labels are different: $E = \{(i,j) : S_i \cap S_j \neq \emptyset \wedge y_i \neq y_j\}$.

To obtain the upper bound, we consider the minimum vertex cover V' from G. By removing all vertices / samples in V', none of the remaining samples can be transformed to have identical feature values with a sample from the opposite class. A perfect classifier $C'(x)$ would therefore assign these rows their correct class values and an attacker will not be able to influence the score of this classifier. It is not possible to misclassify fewer samples than the cardinality of the minimum vertex cover V' since removing any vertex from it will add at least one edge $e \in \{(i,j) : S_i \cap S_j \neq \emptyset \wedge y_i \neq y_j\}$ which will cause an additional misclassification. By König's theorem such a minimum cover in a bipartite graph is equivalent to a maximum matching. Therefore we can use a maximum matching solver to compute an upper bound on the adversarial accuracy. □

Improving Experiment Design

In previous works (Calzavara et al. 2020; Vos and Verwer 2020) attacker capabilities were arbitrarily chosen but this limits the value of algorithm comparisons, shown in Figure 5. In this figure we vary the L_∞ radius ϵ by which an adversary can perturb samples. Particularly, if this value is chosen too large, the best possible model is a trivial one that constantly predict the majority class. If ϵ is chosen too small, the adversary has no effect on the learning problem.

To improve the design of our experiments we propose to choose values for ϵ along these curves that cause the adversarial accuracy bound to be non-trivial. In our experiments we choose three ϵ values for each dataset such that their values corresponds to an adversarial accuracy bound that is at 25%-50%-75% of the range. When choosing ϵ at 100% of the range, the bound is equal to the ratio of the majority class samples, i.e. predicting only that class.

Results

To demonstrate the effectiveness of ROCT we compare it to the state-of-the-art robust tree learning algorithms TREANT and GROOT, and to the regular decision trees from

Dataset (OpenML)	n	p	Maj.
haberman (1)	306	3	.735
blood-transfusion-service-center (1)	748	4	.762
cylinder-bands (2)	277	37	.643
diabetes (1)	768	8	.651
ionosphere (1)	351	34	.641
banknote-authentication (1)	1372	4	.555
breast-w (1)	683	9	.650
wine_quality (1)	6497	11	.633

Table 3: Overview of datasets used in the experiments. Number of samples, features and ratio of majority class samples.
.

Algorithm	Mean adv. accuracy	Mean rank	Wins
Decision Tree	.388 $\pm$ .055	8.917 $\pm$ .083	0
TREANT	.692 $\pm$ .013	5.167 $\pm$ .604	7
Binary-MILP	.714 $\pm$ .013	3.958 $\pm$ .576	10
MILP	.720 $\pm$ .015	2.917 $\pm$ .454	12
RC2-MaxSAT	.724 $\pm$ .014	2.667 $\pm$ .393	10
GROOT	.726 $\pm$ .015	2.375 $\pm$ .450	16
Binary-MILP-warm	.726 $\pm$ .015	2.083 $\pm$ .399	16
LSU-MaxSAT	.729 $\pm$ .014	2.125 $\pm$ .303	13
MILP-warm	**.735** $\pm$ .015	**1.583** $\pm$ .225	**17**

Table 4: Aggregate test scores over 8 datasets, means are shown with standard error. All methods trained for 30 minutes and selected their depth using 3-fold cross validation.

scikit-learn (Pedregosa et al. 2011). First we run the algorithms on an artificial XOR dataset to show that the heuristics can theoretically learn arbitrarily bad trees, see Figure 1. Then to compare the practical performance we run the algorithms on eight popular datasets (Chen et al. 2019; Vos and Verwer 2020) and varying perturbation radii (ϵ). All of our experiments ran on 15 Intel Xeon CPU cores and 72 GB of RAM total, where each algorithm ran on a single core. These datasets are used in many of the existing works to compare robust tree learning algorithms. The datasets are summarized in Table 3 and are available on OpenML[4].

Predictive Performance on Real Data

To demonstrate the practical performance of ROCT we compared the scores of ROCT, GROOT and TREANT on eight datasets. For each dataset we used an 80%-20% train-test split. To limit overfitting it is typical to constrain the maximum depth of the decision tree. To this end we select the best value for the maximum depth hyperparameter using 3-fold stratified cross validation on the training set. In each run, every algorithm gets 30 minutes to fit. For MILP, binary-MILP and LSU-MaxSAT this means that we stop the solver and retrieve its best solution at that time. The methods GROOT, TREANT and RC2-MaxSAT cannot return a solution when interrupted. Therefore when these algorithms exceed the timeout we use a dummy classifier that predicts a constant value. As the dual of the MILP-based formulations is hard to solve, we focus the solver on the primal problem. The final adversarial accuracy scores were determined by testing for each sample whether a sample with a different label intersects its perturbation range.

[4]http://www.openml.org

Dataset	ϵ	Dec. Tree	MILP	GROOT	LSU	MILP warm
banknote-authentication	.07	.665	.742	.775	.796	**.822**
	.09	.589	.669	.684	**.724**	.720
	.11	.491	.625	.640	**.644**	.629
blood-transfusion-service-center	.01	.687	.747	.720	**.760**	.747
	.02	.647	.727	.727	**.767**	**.767**
	.03	.627	**.767**	**.767**	.760	**.767**
breast-cancer	.28	.095	**.869**	**.869**	**.869**	**.869**
	.39	.073	**.818**	**.818**	**.818**	**.818**
	.45	.073	**.774**	**.774**	**.774**	**.774**
cylinder-bands	.23	.000	**.732**	**.732**	.714	.714
	.28	.000	.679	.643	.679	**.750**
	.45	.000	.643	.643	**.679**	.643
diabetes	.05	.455	**.649**	**.649**	**.649**	**.649**
	.07	.364	**.649**	**.649**	**.649**	**.649**
	.09	.286	**.649**	**.649**	**.649**	**.649**
haberman	.02	.726	**.742**	.726	**.742**	**.742**
	.03	.726	**.742**	**.742**	**.742**	**.742**
	.05	.677	**.742**	**.742**	**.742**	**.742**
ionosphere	.2	.310	.817	**.845**	.817	**.845**
	.28	.169	.817	**.845**	.817	**.845**
	.36	.042	**.775**	**.775**	**.775**	**.775**
wine	.02	.602	.638	**.680**	.661	.674
	.03	.541	.633	**.662**	.639	**.662**
	.04	.472	.633	**.659**	.635	**.659**

Table 5: Individual test scores for each dataset and ϵ combination. Best scores are marked in bold.

Table 4 shows the aggregated results over these 8 datasets, 5 contains the individial test scores for a selection of the compared methods. The overall best scores were achieved with the MILP-warm method which is the MILP formulation with continous variables for thresholds and is warm started with the tree produced by GROOT. The LSU-MaxSAT method also performed well and runs without reliance on trees trained with GROOT. TREANT's scores were lower than expected which can be attributed to the number of time outs.

Runtime

An advantage of using optimization solvers for training robust decision trees is that most solvers can be early stopped to output a valid tree. In figure 6 we plotted the mean training scores over all datasets for trees of depth 3 of the solvers that can be stopped. We see that all algorithms converge to nearly the same value given enough time. Moreover we find

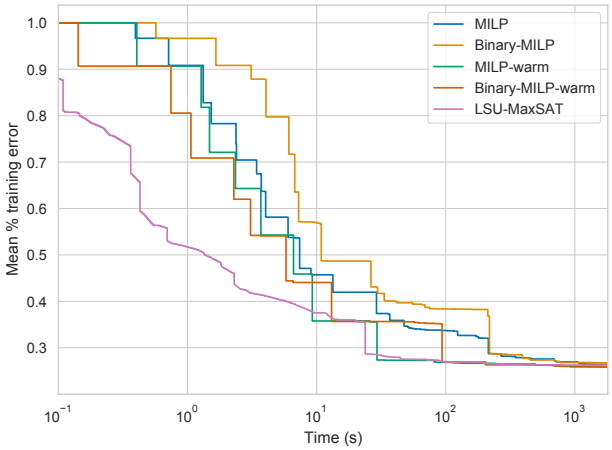


Figure 6: Mean percentage of misclassified training samples of all 8 datasets over time for trees of depth 3. The ranges represent one standard error. LSU-MaxSAT is faster at first but after 30 minutes the other methods catch up.

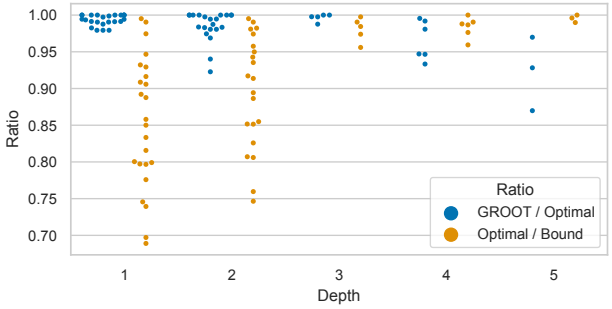


Figure 7: Ratios of training adversarial accuracy scores between GROOT vs LSU-MaxSAT's optimal trees and optimal trees vs our bound (Theorem 2). In most cases GROOT performs within 5% of optimal. In some cases trees of depth 1 or 2 already score as well as the upper bound.

that LSU-MaxSAT quickly achieves good scores where it takes MILP-warm and Binary-MILP-warm approximately 10 and 100 seconds to catch up. The MILP-based methods that were not warm started with GROOT took approximately 1000 seconds to catch up with LSU-MaxSAT.

Optimality

Existing robust decision tree learning algorithms such as TREANT and GROOT have no performance guarantees. Using the LSU-MaxSAT solver we can find trees and prove their optimality on the training set which allows us to compare the scores of the heuristics with these optimal scores. In Figure 7 we plot the approximation ratios of GROOT trees after 2 hours of training. Although LSU-MaxSAT was not able to prove optimality for many datasets after a depth of 2 we can still see that GROOT scores close to optimal. All but one tree scores within a ratio of 0.92 with only one case having a ratio of approximately 0.87. We also plot the ratio between our upper bound and optimal trees. Interestingly, optimal trees of depths 1 and 2 already score close to the upper bounds in some cases.

Conclusions

In this work we propose ROCT, a new solver based method for fitting robust decision trees against adversarial examples. Where existing methods for fitting robust decision trees can perform arbitrarily poorly in theory, ROCT fits the optimal tree given enough time. Important for the computational efficiency of ROCT is the insight and proof that the min-max adversarial training procedure can be computed in one shot for decision trees (Theorem 1). We compared ROCT to existing methods on 8 datasets and found that given 30 minutes of runtime ROCT improved upon the state-of-the-art. Moreover, although greedy methods have been compared to each other in earlier works, we demonstrate for the first time that the state-of-the-art actually performs close to optimal. We also presented a new upper bound for adversarial accuracy that can be computed efficiently using maximum bipartite matching (Theorem 2).

Although ROCT was frequently able to find an optimal solution and shows competitive testing performance, the choice of tree depth strongly influences runtime. Optimality could only be proven for most datasets up to a depth of 2 and for some until depth 4. Additionally, the size of ROCT's formulation grows linearly in terms of the number of unique feature values of the training dataset which results in an exponential increase in runtime. For small datasets of up to a few 1000 samples and tens of features ROCT is likely to improve performance over state-of-the-art greedy methods. Overall, ROCT can increase the performance of state-of-the-art heuristic methods and, due to its optimal nature and new upper bound, provide insight into the difficulty of robust learning.

In the future, we will investigate realistic use cases of adversarial learning in security such as fraud / intrusion / malware detection. We expect our upper-bound method to be a useful tool in determining the sensibility of adversarial learning problems and for robust feature selection.

References

Aglin, G.; Nijssen, S.; and Schaus, P. 2020. Learning optimal decision trees using caching branch-and-bound search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 3146–3153.

Alexey, I.; Antonio, M.; and Joao, M. 2018. PySAT: A Python Toolkit for Prototyping with SAT Oracles. In *SAT*, 428–437.

Avellaneda, F. 2020. Efficient inference of optimal decision trees. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 3195–3202.

Bertsimas, D.; and Dunn, J. 2017. Optimal classification trees. *Machine Learning*, 106(7): 1039–1082.

Bertsimas, D.; and Shioda, R. 2007. Classification and regression via integer optimization. *Operations Research*, 55(2): 252–271.

Blanquero, R.; Carrizosa, E.; Molero-Río, C.; and Morales, D. R. 2021. Optimal randomized classification trees. *Computers & Operations Research*, 132: 105281.

Breiman, L.; Friedman, J.; Stone, C. J.; and Olshen, R. A. 1984. *Classification and regression trees*. CRC press.

Calzavara, S.; Lucchese, C.; Tolomei, G.; Abebe, S. A.; and Orlando, S. 2020. Treant: training evasion-aware decision trees. *Data Mining and Knowledge Discovery*, 34(5): 1390–1420.

Carrizosa, E.; Molero-Río, C.; and Morales, D. R. 2021. Mathematical optimization in classification and regression trees. *TOP*, 1–29.

Chen, H.; Zhang, H.; Boning, D.; and Hsieh, C.-J. 2019. Robust decision trees against adversarial examples. *arXiv preprint arXiv:1902.10660*.

Demirović, E.; Lukina, A.; Hebrard, E.; Chan, J.; Bailey, J.; Leckie, C.; Ramamohanarao, K.; and Stuckey, P. J. 2020. MurTree: Optimal Classification Trees via Dynamic Programming and Search. *arXiv preprint arXiv:2007.12652*.

Diochnos, D. I.; Mahloujifar, S.; and Mahmoody, M. 2018. Adversarial risk and robustness: General definitions and implications for the uniform distribution. *arXiv preprint arXiv:1810.12272*.

Hu, X.; Rudin, C.; and Seltzer, M. 2019. Optimal sparse decision trees. *Advances in Neural Information Processing Systems (NeurIPS)*.

Ignatiev, A.; Morgado, A.; and Marques-Silva, J. 2019. RC2: An efficient MaxSAT solver. *Journal on Satisfiability, Boolean Modeling and Computation*, 11(1): 53–64.

Kantchelian, A.; Tygar, J. D.; and Joseph, A. 2016. Evasion and hardening of tree ensemble classifiers. In *International Conference on Machine Learning*, 2387–2396.

Kearns, M. 1996. Boosting theory towards practice: Recent developments in decision tree induction and the weak learning framework. In *Proceedings of the national conference on artificial intelligence*, 1337–1339.

Laurent, H.; and Rivest, R. L. 1976. Constructing optimal binary decision trees is NP-complete. *Information processing letters*, 5(1): 15–17.

Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; and Vladu, A. 2017. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*.

Martins, R.; Joshi, S.; Manquinho, V.; and Lynce, I. 2014. Incremental cardinality constraints for MaxSAT. In *International Conference on Principles and Practice of Constraint Programming*, 531–548. Springer.

Morgado, A.; Heras, F.; Liffiton, M.; Planes, J.; and Marques-Silva, J. 2013. Iterative and core-guided MaxSAT solving: A survey and assessment. *Constraints*, 18(4): 478–534.

Narodytska, N.; Ignatiev, A.; Pereira, F.; Marques-Silva, J.; and RAS, I. 2018. Learning Optimal Decision Trees with SAT. In *IJCAI*, 1362–1368.

Nijssen, S.; and Fromont, E. 2010. Optimal constraint-based decision tree induction from itemset lattices. *Data Mining and Knowledge Discovery*, 21(1): 9–51.

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.; and Duchesnay, E. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12: 2825–2830.

Quinlan, J. 1993. C4. 5: Programs for machine learning Morgan Kaufmann San Francisco. *CA, USA*.

Quinlan, J. R. 1986. Induction of decision trees. *Machine learning*, 1(1): 81–106.

Rhuggenaath, J.; Zhang, Y.; Akcay, A.; Kaymak, U.; and Verwer, S. 2018. Learning fuzzy decision trees using integer programming. In *2018 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, 1–8. IEEE.

Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I.; and Fergus, R. 2013. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*.

Verhaeghe, H.; Nijssen, S.; Pesant, G.; Quimper, C.-G.; and Schaus, P. 2020. Learning optimal decision trees using constraint programming. *Constraints*, 25(3): 226–250.

Verwer, S.; and Zhang, Y. 2017. Learning decision trees with flexible constraints and objectives using integer optimization. In *International Conference on AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, 94–103. Springer.

Verwer, S.; and Zhang, Y. 2019. Learning optimal classification trees using a binary linear program formulation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 1625–1632.

Vos, D.; and Verwer, S. 2020. Efficient Training of Robust Decision Trees Against Adversarial Examples. *arXiv preprint arXiv:2012.10438*.

Wang, Y.; Jha, S.; and Chaudhuri, K. 2018. Analyzing the robustness of nearest neighbors to adversarial examples. In *International Conference on Machine Learning*, 5133–5142. PMLR.