Fast Provably Robust Decision Trees and Boosting

Jun-Qi Guo [1] **Ming-Zhuo Teng** [1] **Wei Gao** [1] **Zhi-Hua Zhou** [1]

Abstract

Learning with adversarial robustness has been a challenge in contemporary machine learning, and recent years have witnessed increasing attention on robust decision trees and ensembles, mostly working with high computational complexity or without guarantees of provable robustness. This work proposes the Fast Provably Robust Decision Tree (FPRDT) with the smallest computational complexity $O(n \log n)$, a tradeoff between global and local optimizations over the adversarial $0/1$ loss. We further develop the Provably Robust AdaBoost (PRAdaBoost) according to our robust decision trees, and present convergence analysis for training adversarial $0/1$ loss. We conduct extensive experiments to support our approaches; in particular, our approaches are superior to those unprovably robust methods, and achieve better or comparable performance to those provably robust methods yet with the smallest running time.

1. Introduction

A well-trained model may be heavily misled by examples of adversarial perturbations (Szegedy et al., 2014; Goodfellow et al., 2015), which has been a big challenge in machine learning. A large number of defensive algorithms have been developed to deal with adversarial examples (Goodfellow et al., 2015; Papernot et al., 2016; Madry et al., 2018; Liu et al., 2018; Akhtar et al., 2018; Liao et al., 2018), whereas most are heuristic without theoretical guarantees of robustness. *Provably robust training* has been an effective method to guarantee the robustness against adversarial perturbations, and the basic idea is to exactly optimize the adversarial loss or its upper bound (Mirman et al., 2018; Raghunathan et al., 2018a;b; Wong & Kolter, 2018; Wong et al., 2018; Cohen et al., 2019; Li et al., 2019; Balunović & Vechev, 2020).

[1]National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China. Correspondence to: Wei Gao <gaow@nju.edu.cn>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

Table 1. Comparisons of computational complexity and provable robustness for robust decision trees and boosting (n denotes the number of training data).

Methods	Comp. complexity	Prov. robustness
Our FPRDT	$O(n \log n)$	✓
Our PRAdaBoost	$O(n \log n)$	✓
ROCT (Vos & Verwer, 2021b)	$O(\exp(n))$	✓
TREANT (Calzavara et al., 2020)	$O(n^2)$	✓
PRB (Andriushchenko & Hein, 2019)	$O(n^2)$	✓
GROOT (Vos & Verwer, 2021a)	$O(n \log n)$	✗
RIGDT-heuristic (Chen et al., 2019a)	$O(n \log n)$	✗
RGBDT (Chen et al., 2019a)	$O(n \log n)$	✗
AdvBoost (Kantchelian et al., 2016)	$O(n \log n)$	✗

Recent years have witnessed the increasing attention on robust decision trees and ensembles, from the discoveries of adversarial examples in tree attacks (Kantchelian et al., 2016; Chen et al., 2019b; Cheng et al., 2019). Kantchelian et al. (2016) produced approximated adversarial examples, and trained decision trees iteratively based on the original data and new adversarial examples, and Andriushchenko & Hein (2019) proposed the provably robust boosting via upper bound of adversarial loss. Chen et al. (2019a) and Vos & Verwer (2021a) constructed robust decision trees and ensembles by adversarial Gini impurity or information gain. Calzavara et al. (2020) presented the provably robust decision tree with additional constraints over examples, and Vos & Verwer (2021b) suggested globally optimal robust decision trees. Table 1 summarizes the provable robustness and computational complexities for different methods.

This work presents the fast algorithms for provably robust decision trees and Boosting, and the main contributions can be summarized as follows:

- We propose the Fast Provably Robust Decision Tree (FPRDT), a greedy recursive approach on the direct minimization of the adversarial $0/1$ loss. Our approach makes optimization over all leaves and the splitting children, which is different from global optimization over all internal and leaf nodes, and local optimization over the single splitting node with its children. Our approach takes $O(n \log n)$ computational complexity.

- We further develop the Provably Robust AdaBoost (PRAdaBoost), a boosting algorithm to minimize an upper bound on the adversarial exponential loss, and the base learners are constructed with our FPRDT with slight modifications. We present the exponential convergence analysis of the training adversarial $0/1$ loss for our approach, and obtain the smallest $O(n \log n)$ computational complexity.
- We finally present extensive experiments to validate the effectiveness of our approaches. Specifically speaking, our approaches are clearly superior to those unprovably robust methods of decision trees and tree ensembles, and achieve better or comparable performance to provably robust methods yet of the smallest running time.

The rest of this work is organized as follows: Section 2 presents some preliminaries. Section 3 proposes the FPRDT approach. Section 4 develops the PRAdaBoost approach. Section 5 conducts extensive experiments, and Section 6 concludes with future work.

2. Preliminaries

Let $\mathcal{X} \subset \mathbb{R}^d$ be the instance/input space, and $\mathcal{Y} = \{-1, 1\}$ denotes the output space for binary classification. Suppose that $\mathcal{D}$ is an underlying distribution over the product space $\mathcal{X} \times \mathcal{Y}$. Distribution $\mathcal{D}$ is unknown in practice, and what we can observe is a training data

$$S_n = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), ..., (\boldsymbol{x}_n, y_n)\} ,$$

where each example is drawn i.i.d. from distribution $\mathcal{D}$. Let $\mathcal{N}_\epsilon(\boldsymbol{x})$ denote the ball with center $\boldsymbol{x}$ and radius ϵ under the L_∞ metric, that is,

$$\mathcal{N}_\epsilon(\boldsymbol{x}) = \{\boldsymbol{z} \colon \|\boldsymbol{z} - \boldsymbol{x}\|_\infty \leq \epsilon\} .$$

Let $\mathcal{H}$ be a hypothesis space, and a loss function $\ell(h(\boldsymbol{x}), y))$ is introduced to measure the prediction performance with hypothesis $h \in \mathcal{H}$ and example $(\boldsymbol{x}, y)$. Given example $\boldsymbol{x}$, an adversarial example $\boldsymbol{x}_{\text{adv}}$ is given by

$$\boldsymbol{x}_{\text{adv}} \in \arg\max_{\boldsymbol{x}' \in \mathcal{N}_\epsilon(\boldsymbol{x})} \{\ell(h(\boldsymbol{x}'), y)\} .$$

Given training sample S_n, the adversarial robust learning aims to look for a hypothesis $\hat{h}^* \in \mathcal{H}$ (Madry et al., 2018), such that

$$\hat{h}^* \in \arg\min_{h \in \mathcal{H}} \left\{ \sum_{i=1}^{n} \max_{\boldsymbol{x}' \in \mathcal{N}_\epsilon(\boldsymbol{x}_i)} \ell(h(\boldsymbol{x}'), y_i) \right\} . \quad (1)$$

One feasible method is to solve the inner maximization of Eqn. (1) approximately at each step, called *adversarial training* as mentioned by Goodfellow et al. (2015).

We introduce some notations in this work. Let $\mathbb{I}[\cdot]$ denote the indicator function, which returns 1 if the argument is true, and 0 otherwise. Let $\emptyset$ stand for the empty set, and denote by $[k] = \{1, 2, \cdots, k\}$ for integer $k > 0$.

3. Fast Provably Robust Decision Trees

A decision tree can be constructed with some internal and leaf nodes by partitioning training data recursively, and the prediction follows the path from root to leaf. For a decision tree of m leaf nodes, we could associate with m corresponding rectangle cells $\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_m$ as follows:

$$\mathcal{B}_j = (a_1^j, b_1^j] \times \cdots \times (a_d^j, b_d^j] \quad \text{for} \quad j \in [m] .$$

Given example $(\boldsymbol{x}_i, y_i) \in S_n$ and leaf node $\mathcal{B}_j$, we further introduce the set of adversarial examples of $\boldsymbol{x}_i$, falling into the j-th leaf node $\mathcal{B}_j$, as follows:

$$\mathcal{E}_{ij} = \{\boldsymbol{x} \in \mathcal{X} \colon \ \boldsymbol{x} \in \mathcal{N}_\epsilon(\boldsymbol{x}_i) \ \text{ and } \ \boldsymbol{x} \in \mathcal{B}_j\} .$$

Let $h(\cdot)$ denote the prediction function of a decision tree. We have $h(\boldsymbol{x}) = h(\boldsymbol{x}')$ for every $\boldsymbol{x}, \boldsymbol{x}' \in \mathcal{B}_j$, and this follows that $h(\boldsymbol{x}) = h(\boldsymbol{x}')$ for every $\boldsymbol{x}, \boldsymbol{x}' \in \mathcal{E}_{ij}$.

By traversing all leaf nodes of a decision tree, optimizing Eqn (1) is equivalent to minimizing the following objective

$$\sum_{i=1}^{n} \max_{j \in [m]} \{\ell(h(\boldsymbol{x}), y_i) \mathbb{I}[\boldsymbol{x} \in \mathcal{E}_{ij}]\} , \quad (2)$$

where $\ell(h(\boldsymbol{x}), y_i)\mathbb{I}[\boldsymbol{x} \in \mathcal{E}_{ij}] = 0$ when $\mathcal{E}_{ij} = \emptyset$ by default.

We directly use the 0/1 loss, rather than traditional surrogate losses such as square loss and softmax loss, as the splitting criterion during the construction of decision tree, that is,

$$\ell(h(\boldsymbol{x}), y_i) = \mathbb{I}[h(\boldsymbol{x}) \neq y_i] .$$

This is because the consistency between those surrogate losses and $0/1$ loss does not hold in the adversarial setting (Awasthi et al., 2021). Another reason is the efficiency on the robust optimization of splitting nodes with the $O(1)$ computational complexity, while those surrogate losses would require the $O(n)$ computational complexity via gradient descent. The details can be found in Appendix A.

Following the classic method (Rokach & Maimon, 2005), at one split with previous optimal loss $\hat{\mathcal{L}}^{\text{pre}}$ of Eqn. (2), we assume that the p-th leaf node will be split, and let t and η be the splitting feature and threshold, respectively. For its left and right child, we denote by v_l and v_r their respective values, and $\mathcal{B}_l$ and $\mathcal{B}_r$ represent their associated rectangle cells. Let v_j be the value of the j-th leaf node for $j \neq p$. We could write the objective loss of Eqn. (2) at one split as

$$\hat{\mathcal{L}}(t, \eta, v_l, v_r) = \sum_{i=1}^{n} \max \big\{ \mathbb{I}[v_l \neq y_i]\mathbb{I}[\mathcal{E}_{il} \neq \emptyset], \\ K_{ip}, \mathbb{I}[v_r \neq y_i]\mathbb{I}[\mathcal{E}_{ir} \neq \emptyset] \big\} ,$$

Algorithm 1 Fast Provably Robust Decision Tree (FPRDT)

Input: Dataset S_n
Output: A tree T
Initialize: Root of T with the majority class in S_n
$\hat{\mathcal{L}} = \min\{\sum_{i=1}^n \mathbb{I}[y_i = 1], \sum_{i=1}^n \mathbb{I}[y_i = -1]\}$
$E_i = 0$ for the majority class, and 1 otherwise
PRDT-recursive(S_n, T, ϵ, $\hat{\mathcal{L}}$)
Return: T

with $K_{ip} = \max_{j\in[m]\setminus\{p\}} \{\mathbb{I}[v_j \neq y_i]\mathbb{I}[\mathcal{E}_{ij} \neq \emptyset]\} \in \{0, 1\}$. Here, we denote by $\mathcal{E}_{il}$ and $\mathcal{E}_{ir}$ the sets of $\mathcal{N}_\epsilon(\boldsymbol{x}_i) \cap \mathcal{B}_l$ and $\mathcal{N}_\epsilon(\boldsymbol{x}_i) \cap \mathcal{B}_r$, respectively. The one-split goal for robust training is to solve

$$\{t^*, \eta^*, v_l^*, v_r^*\} \in \underset{t,\eta,v_l,v_r}{\arg\min} \left\{\hat{\mathcal{L}}(t, \eta, v_l, v_r)\right\} .$$

To calculate K_{ip} efficiently, let E_i $(i \in [n])$ be the number of leaves, including adversarial examples of $\boldsymbol{x}_i$. We have

$$E_i = \sum_{j\in[m]} \mathbb{I}[v_j \neq y_i]\mathbb{I}[\mathcal{E}_{ij} \neq \emptyset] ,$$

and this follows that, $K_{ip} = 1$ if and only if

$$E_i - \mathbb{I}[v_p \neq y_i]\mathbb{I}[\mathcal{E}_{ip} \neq \emptyset] \geq 1 , \tag{3}$$

where v_p denotes the value of the p-th leaf node.

Following the robust decision trees (Vos & Verwer, 2021a), we traverse all potential features $t \in [d]$ and thresholds

$$\eta \in W_t = \bigcup\nolimits_{i=1,\mathcal{E}_{ip}\neq\emptyset}^{n} \{x_{it} - \epsilon, x_{it}, x_{it} + \epsilon\}. \tag{4}$$

For any fixed splitting feature t' and threshold η', it is easy to observe that

$$\mathbb{I}[\mathcal{E}_{il} \neq \emptyset] = \mathbb{I}[\mathcal{E}_{ip} \neq \emptyset \text{ and } x_{it'} \leq \eta' + \epsilon] , \tag{5}$$
$$\mathbb{I}[\mathcal{E}_{ir} \neq \emptyset] = \mathbb{I}[\mathcal{E}_{ip} \neq \emptyset \text{ and } x_{it'} > \eta' - \epsilon] , \tag{6}$$

where $x_{it'}$ is the value of the t'-th feature of instance $\boldsymbol{x}_i$. We then solve the following problem

$$(v_l', v_r') = \underset{v_l,v_r\in\{0,1\}}{\arg\min} \left\{\hat{\mathcal{L}}(t', \eta', v_l, v_r)\right\} , \tag{7}$$

where it takes $O(1)$ computational complexity to solve the above problem based on loss decomposition when W_t is sorted. Note that it would take $O(n)$ computational complexity if we use other surrogate losses instead. The detailed solution and discussion can be found in Appendix A.

After traversing potential features and thresholds, and solving their respective minimizations of Eqn. (7), we can obtain the final optimal objective loss

$$\hat{\mathcal{L}}^* = \hat{\mathcal{L}}(t^*, \eta^*, v_l^*, v_r^*) = \min_{t'\in[d],\eta'\in W_t} \left\{\min_{v_l,v_r\in\{0,1\}} \hat{\mathcal{L}}(t', \eta', v_l, v_r)\right\}$$

Algorithm 2 FPRDT-recursive

Input: Dataset S_n, split leaf p, perturbation size ϵ, previous optimal loss $\hat{\mathcal{L}}^{\text{pre}}$
Output: None
Initialize $\hat{\mathcal{L}}^* \leftarrow +\infty$
Compute loss K_{ip} $(i \in [n])$ according to Eqn. (3)
for $t' = 1$ **to** d **do**
  Compute $W_{t'}$ according to Eqn. (4)
  for η' **in** sorted($W_{t'}$) **do**
    Compute $\mathbb{I}[\mathcal{E}_{il} \neq \emptyset]$ and $\mathbb{I}[\mathcal{E}_{ir} \neq \emptyset]$ by Eqns. (5)-(6)
    Solve v_l' and v_r' from Eqn. (7) with optimal loss $\hat{\mathcal{L}}'$
    if $\hat{\mathcal{L}}' < \hat{\mathcal{L}}^*$ **then**
      $t^* = t', \eta^* = \eta', v_l^* = v_l', v_r^* = v_r', \hat{\mathcal{L}}^* = \hat{\mathcal{L}}'$
    end if
  end for
end for
if $\hat{\mathcal{L}}^* < \hat{\mathcal{L}}^{\text{pre}}$ **then**
  Split leaf p via $\{t^*, \eta^*, v_l^*, v_r^*\}$, and obtain children l, r
  Update E_i $(i \in [n])$ according to Eqn. (8)
  FPRDT-recursive(S_n, l, ϵ, $\hat{\mathcal{L}}^*$)
  FPRDT-recursive(S_n, r, ϵ, $\hat{\mathcal{L}}^*$)
end if

with the optimal solution $\{t^*, \eta^*, v_l^*, v_r^*\}$.

We split the p-th leaf node only if $\hat{\mathcal{L}}^* < \hat{\mathcal{L}}^{\text{pre}}$, that is, the current optimal loss $\hat{\mathcal{L}}^*$ is smaller than the previous optimal loss $\hat{\mathcal{L}}^{\text{pre}}$. For a splitting node, we update

$$E_i = E_i + \mathbb{I}[v_l \neq y_i]\mathbb{I}[\mathcal{E}_{il} \neq \emptyset] + \mathbb{I}[v_r \neq y_i]\mathbb{I}[\mathcal{E}_{ir} \neq \emptyset] - \mathbb{I}[v_p \neq y_i]\mathbb{I}[\mathcal{E}_{ip} \neq \emptyset] . \tag{8}$$

Algorithms 1 and 2 present the detailed description of our Fast Provably Robust Decision Trees (FPRDT) approach, where $E_i (i \in [n])$ are defined as global variables.

We now present computational complexity analysis for our FPRDT approach at one split. It takes $O(n)$ computational complexity on the update of E_i from Eqn. (8) for $i \in [n]$, as well as $O(n)$ computational complexity for K_{ip} from Eqn. (3). We have at most $3n$ splits for each feature from Eqn. (4), and take $O(n \log n)$ computational complexity to sort W_t. In summary, our proposed FPRDT method takes the $O(n \log n)$ computational complexity.

Comparisons with previous work

Chen et al. (2019a) developed the robust decision trees based on adversarial Gini impurity, and Vos & Verwer (2021a) further introduced robust decision trees with better training efficiency in terms of their closed-form solutions. Yang et al. (2019) proved the equivalence between Gini impurity and square loss on the construction of regular decision trees, whereas we prove that such equivalence does not hold in the adversarial setting as follows:

Theorem 3.1. *Optimizing the adversarial Gini impurity of robust decision trees is equivalent to minimizing a lower bound of* $\sum_{i=1}^{n} \max_{j\in[m]}\{(1-h(\boldsymbol{x})y_i)^2\mathbb{I}[\boldsymbol{x}\in\mathcal{E}_{ij}]\}$.

From this theorem, we can see that the equivalence does not hold between adversarial Gini impurity and adversarial square loss, which proves the unprovable robustness of algorithms via optimizing the adversarial Gini impurity. The detailed proof of Theorem 3.1 is presented in Appendix B.

Calzavara et al. (2020) introduced provably robust decision trees of $O(n^2)$ computational complexity with additional constraints over examples, and Vos & Verwer (2021b) gave the robust optimal decision tree based on mixed-integer linear program, which is an NP-hard problem with exponential computational complexity. In comparison, our FPRDT approach takes the smallest $O(n\log n)$ computational complexity in all provably robust methods.

In our FPRDT approach, we introduce the maximal loss K_{ip} over all leaves except for the splitting leaf. This is different from the robust optimal decision tree (Vos & Verwer, 2021b), which takes the global optimization over all internal and leaf nodes. We also distinguish from local optimization (Chen et al., 2019a; Vos & Verwer, 2021a), which only focuses on the splitting leaf with its children, regardless of other leaves. Our method can be viewed as a trade-off between global and local optimizations.

4. Provably Robust AdaBoost

The AdaBoost algorithm (Freund et al., 1996) has been one of the most influential algorithms in machine learning, and the basic idea is to construct a "strong" classifier from a sequence of "weak" learners. Essentially, AdaBoost can be viewed as a procedure on the optimization of exponential loss (Schapire, 2013). This section focuses on provably robust guarantees for AdaBoost via an upper bound over the adversarial exponential loss.

Let $H_t(\boldsymbol{x}) = \sum_{i=1}^{t}\alpha_i h_i(\boldsymbol{x})$ be the output of AdaBoost with base learners $h_i \in \mathcal{H}$ and corresponding weights α_i. We have the empirical adversarial exponential loss over training sample S_n as follows:

$$\hat{\mathcal{L}}(h, S_n) = \frac{1}{n}\sum_{i=1}^{n}\max_{\boldsymbol{x}_i'\in\mathcal{N}_\epsilon(\boldsymbol{x}_i)}\left\{\exp(-H_t(\boldsymbol{x}_i')y_i)\right\}.$$

It is an NP-hard problem to directly optimize the above empirical adversarial loss for tree ensembles (Kantchelian et al., 2016), and we resort to an upper bound as follows:

$$\begin{aligned}\hat{\mathcal{L}}(h, S_n) &= \frac{1}{n}\sum_{i=1}^{n}\max_{\boldsymbol{x}_i'\in\mathcal{N}_\epsilon(\boldsymbol{x}_i)}\left\{\prod_{j=1}^{t}\exp(-y_i\alpha_j h_j(\boldsymbol{x}_i'))\right\}\\ &\le \frac{1}{n}\sum_{i=1}^{n}\prod_{j=1}^{t}\max_{\boldsymbol{x}_i'\in\mathcal{N}_\epsilon(\boldsymbol{x}_i)}\left\{\exp(-y_i\alpha_j h_j(\boldsymbol{x}_i'))\right\}.\end{aligned}$$

Algorithm 3 Provably Robust AdaBoost (PRAdaBoost)

Input: Dataset S_n, iteration num. T and perturb. size ϵ
Output: Classifier H
Initialize: Instance weights $w_{1,i} = 1$ $(i \in [n])$
for $t = 1$ **to** T **do**
  Solve h_t from Eqn. (9) by our modified FPRDT
  Calculate ϵ_t according to Eqn. (11)
  if $\epsilon_t > 1/2$ **then**
    break
  end if
  Calculate classifier weight α_t from Eqn. (10)
  Calculate instance weights $w_{t+1,i}$ from Eqn. (12)
end for
Return $H(\boldsymbol{x}) = \text{sign}(\sum_{t=1}^{T}\alpha_t h_t(\boldsymbol{x}))$

Such relaxation has also been used by Andriushchenko & Hein (2019). Let $w_{t,i}$ denote the weight of instance $(\boldsymbol{x}_i, y_i)$ after the $t-1$-th iteration, and motivated from the original AdaBoost, we have

$$w_{t,i} = \prod_{j=1}^{t-1}\max_{\boldsymbol{x}_i'\in\mathcal{N}_\epsilon(x_i)}\left\{\exp(-y_i\alpha_j h_j(\boldsymbol{x}_i'))\right\}.$$

In the t-th iteration, we try to solve the following problem

$$\{\alpha_t, h_t\} \in \arg\min_{\alpha,h}\left\{\sum_{i=1}^{n} w_{t,i}\max_{\boldsymbol{x}_i'\in\mathcal{N}_\epsilon(\boldsymbol{x}_i)}\exp(-y_i\alpha h(\boldsymbol{x}_i'))\right\}.$$

We first minimize the following weighted adversarial $0/1$ loss to solve h_t, for any given $\alpha_t > 0$,

$$h_t = \arg\min_{h}\sum_{i=1}^{n} w_{t,i}\max_{\boldsymbol{x}_i'\in\mathcal{N}_\epsilon(\boldsymbol{x}_i)}\mathbb{I}[h(\boldsymbol{x}_i')\neq y_i]\,, \quad (9)$$

and this can be efficiently solved based on our FPRDT approach with modification of objective loss in Eqn. (2) to weighted ones. The details can be found in Appendix C.

We then solve the weight α_t from its derivative as follows:

$$\alpha_t = \frac{1}{2}\times\ln(\frac{1-\epsilon_t}{\epsilon_t})\,, \quad (10)$$

where

$$\epsilon_t = \sum_{i=1}^{n}\frac{w_{t,i}}{\sum_{i=1}^{n} w_{t,i}}\max_{\boldsymbol{x}_i'\in\mathcal{N}_\epsilon(\boldsymbol{x}_i)}\{\mathbb{I}(h_t(\boldsymbol{x}_i')\neq y_i)\}\,. \quad (11)$$

We finally calculate instance weights $w_{t,i}$ according to the following recursive relationship

$$w_{t+1,i} = w_{t,i}\max_{\boldsymbol{x}_i'\in\mathcal{N}_\epsilon(\boldsymbol{x}_i)}\left\{\exp(-y_i\alpha_t h_t(\boldsymbol{x}_i'))\right\}. \quad (12)$$

Algorithm 3 presents the detailed description of Provably Robust AdaBoost (PRAdaBoost), and takes the $O(n\log n)$

computational complexity in each iteration since it requires $O(n)$ and $O(n \log n)$ computational complexities to compute instance weights and base learners, respectively.

We present the convergence analysis on the adversarial $0/1$ loss over training dataset for our PRAdaBoost as follows:

Theorem 4.1. *Let $H(\boldsymbol{x}) = sign(\sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x}))$ be the final classifier of PRAdaBoost with $\gamma_t = 1/2 - \epsilon_t > 0$ for each iteration $t \in [T]$. We have the empirical adversarial $0/1$ loss over training sample S_n as follows:*

$$\frac{1}{n}\sum_{i=1}^{n}\left[\max_{\boldsymbol{x}_i' \in \mathcal{N}_\epsilon(\boldsymbol{x}_i)} \mathbb{I}[H(\boldsymbol{x}_i') \neq y_i]\right] \leq \exp\left(-2\sum_{t=1}^{T}\gamma_t^2\right).$$

This theorem shows the exponential convergence rate for the PRAdaBoost approach when every base learner has slightly better performance than random guess in the adversarial setting, that is, $\gamma_t = 1/2 - \epsilon_t \geq \gamma$ for some small $\gamma > 0$. The detailed proof is presented in Appendix D.

Comparisons with previous work

Kantchelian et al. (2016) proposed to train a sequence of robust decision trees iteratively and recursively, under the L_0 metric, based on the original data and new approximated adversarial examples. Vos & Verwer (2021a) presented the robust random forests following the original idea of random forests (Breiman, 2001).

Andriushchenko & Hein (2019) introduced another robust boosting algorithm via an upper bound over adversarial exponential loss with $O(n^2)$ computational complexity, whereas we consider the modified FPRDT to construct the base learners based on the direct optimization of adversarial $0/1$ loss with $O(n \log n)$ computational complexity.

Our PRAdaBoost can be viewed as a general learning approach which can be used in any place where AdaBoost and robust boostings can be applied. An interesting future work is to exploit the generalization and consistency of AdaBoost and the convergence rate of random forests in adversarial settings, by analogy with the traditional studies (Bartlett & Traskin, 2006; Gao & Zhou, 2013; 2020).

5. Experiments

We conduct our experiments on 18 datasets[1,2], and most datasets have been used in previous robust decision trees and ensembles. Table 2 presents the detailed statistical information, and we take the same perturbation size ϵ as previous robust studies on decision trees or tree ensembles. The features have been scaled to $[-1, 1]$ for all datasets.

We take the adversarial accuracy as predictive measure in robust classifications (Goodfellow et al., 2015; Andriushchenko & Hein, 2019; Vos & Verwer, 2021a;b). For test data $T = \{(\boldsymbol{x}_1, y_1), \cdots, (\boldsymbol{x}_m, y_m)\}$, we define the test adversarial accuracy as

$$1 - \frac{1}{m}\sum_{i=1}^{m}\max_{\|\boldsymbol{z}-\boldsymbol{x}_i\|_\infty \leq \epsilon} \mathbb{I}(h(\boldsymbol{z}) \neq y_i),$$

following the implementation of (Vos & Verwer, 2021a).

Table 2. Datasets

Dataset (Perb.)	# Inst.	# Feat.	Dataset (Perb.)	# Inst.	# Feat.
ionos (.2)	351	34	cifar10:0v5 (.1)	12,000	3,072
breast (.3)	683	9	cifar10:0v6 (.1)	12,000	3,072
diabet (.05)	768	8	cifar10:4v8 (.1)	12,000	3,072
bank (.1)	1,372	4	mnist2v6 (.4)	13,866	784
Japan3v4 (.1)	3,087	14	mnist3v8 (.4)	13,966	784
har1v2 (.1)	3,266	561	F-mnist2v5 (.2)	14,000	784
spam (.05)	4,601	57	F-mnist3v4 (.2)	14,000	784
GesDvP (.01)	4,838	32	F-mnist7v9 (.2)	14,000	784
wine (.05)	6,497	11	mnist1v7 (.4)	15,170	784

For fair comparisons, we run all experiments on a single core without parallel optimizations, and experiments are performed with Python on nodes of a computational cluster with 20 CPUs (Intel Core i9-10900X 3.7GHz), running Ubuntu with 128GB main memory.

5.1. Comparisons on Robust Decision Trees

Besides regular decision trees (Safavian & Landgrebe, 1991), we compare with state-of-the-art robust decision trees as follows:

- RIGBT-h[3]: Robust decision trees based on adversarial information gain (Chen et al., 2019a);
- TREANT[4]: Robust decision trees with constraints over training examples (Calzavara et al., 2020);
- GROOT[5]: Robust decision trees based on adversarial Gini impurity (Vos & Verwer, 2021a);
- ROCT[5]: Robust decision trees with global optimization over all nodes (Vos & Verwer, 2021b);
- PRB tree[6]: Robust decision trees based on upper bound of exponential loss (Andriushchenko & Hein, 2019).

We take the maximum depth 4 for TREANT and ROCT due to high computational complexity, and do not restrict the

[1] https://www.openml.org/
[2] https://www.cs.toronto.edu/~kriz/cifar.html
[3] The code is taken from https://github.com/chenhongge.
[4] The code is taken from https://github.com//gtolomei/treant.
[5] The code is taken from https://github.com/tudelft-cda-lab.
[6] The code is taken from https://github.com/max-andr.

Table 3. Comparisons of adversarial accuracies (mean±std). ●/○ indicates that our FPRDT is significantly better/worse than the corresponding methods (pairwise t-tests at 95% significance level). 'N/A' means that no results were obtained after running out 12 hours.

Dataset	FPRDT	Decision tree	RIGDT-h	GROOT	TREANT	ROCT	PRB tree
ionos	.7954 ± .0302	.3100 ± .0549●	.7015 ± .0874●	.7829 ± .0325●	.7232 ± .0434●	.7897 ± .0330●	.7601 ± .0346●
breast	.8765 ± .0290	.2501 ± .0457●	.8381 ± .0317●	.8744 ± .0273	.8334 ± .0318●	.8735 ± .0256	.8706 ± .0258●
diabet	.6674 ± .0258	.6333 ± .0346●	.5695 ± .0640●	.6481 ± .0352●	.6695 ± .0228	.6552 ± .0244●	.6633 ± .0333
bank	.6577 ± .0311	.6333 ± .0346●	.4685 ± .0713●	.5410 ± .0440●	.6092 ± .0259●	.6539 ± .0167	.6299 ± .0328●
Japan3v4	.6673 ± .0129	.5751 ± .0377●	.5638 ± .0337●	.5829 ± .0468●	N/A	.6671 ± .0170	.5954 ± .0112●
har1v2	.8044 ± .0249	.2316 ± .0434●	.7074 ± .0257●	.8058 ± .0198	N/A	.7786 ± .0165●	.7383 ± .0148●
spam	.7404 ± .0124	.0006 ± .0012●	.4669 ± .1019●	.7231 ± .0188●	N/A	.4813 ± .0469●	.6968 ± .0110●
GesDvP	.7301 ± .0174	.4783 ± .1390●	.5483 ± .1035●	.7164 ± .0180●	N/A	.7071 ± .0126●	.7014 ± .0190●
wine	.6364 ± .0062	.3515 ± .1430●	.4032 ± .0375●	.6373 ± .0073	.6388 ± .0079○	.6117 ± .0343●	.6299 ± .0080●
cifar10:0v5	.6878 ± .0177	.2960 ± .0598●	.3469 ± .0457●	.4847 ± .0608●	N/A	.6639 ± .0113●	.6501 ± .0106●
cifar10:0v6	.6883 ± .0102	.5878 ± .0568●	.4771 ± .0168●	.5555 ± .0495●	N/A	.6684 ± .0077●	.6833 ± .0051●
cifar10:4v8	.6613 ± .0117	.2561 ± .0784●	.4882 ± .0468●	.4727 ± .0195●	N/A	.6319 ± .0114●	.6698 ± .0093○
mnist2v6	.8954 ± .0025	.0178 ± .0320●	.8850 ± .0079●	.8725 ± .0110●	N/A	.7842 ± .0222●	.8385 ± .0673●
mnist3v8	.8527 ± .0058	.0028 ± .0071●	.8102 ± .0102●	.7575 ± .0185●	N/A	.7565 ± .0280●	.8030 ± .0235●
F-mnist2v5	.9780 ± .0027	.3214 ± .2143●	.9446 ± .0080●	.9714 ± .0054●	N/A	.9394 ± .0128●	.9761 ± .0025●
F-mnist3v4	.8652 ± .0056	.0166 ± .0521●	.7928 ± .0129●	.8193 ± .0104●	N/A	.8271 ± .0168●	.8407 ± .0052●
F-mnist7v9	.8760 ± .0058	.2930 ± .1554●	.8100 ± .0108●	.8291 ± .0115●	N/A	.8376 ± .0188●	.8399 ± .0106●
mnist1v7	.9633 ± .0034	.0036 ± .0081●	.9325 ± .0076●	.9457 ± .0052●	N/A	.8937 ± .0095●	.9196 ± .0264●
Average	.7802 ± .1090	.2922 ± .2172	.6530 ± .1877	.7233 ± .1494	—	.7342 ± .1134	.7503 ± .1075
Win/Tie/Loss		**18/0/0**	**18/0/0**	**15/3/0**	**16/1/1**	**15/3/0**	**16/1/1**

depth for other methods. We set 10 as the minimum number of instances for a splitting leaf node, and each leaf node has at least 5 instances. We modify the original TREANT by adding perturbation ϵ for each feature once time as in the work of (Vos & Verwer, 2021a). ROCT is initialized with the return of GROOT, and outputs the final result by optimizing objective loss within 30 minutes as done by Vos & Verwer (2021b). More experimental settings and results can be found in Appendix E.

The performances of the compared methods are evaluated by five trials of 5-fold cross validation, where test adversarial accuracies are obtained by averaging over these 25 runs, as summarized in Table 3. These empirical results clearly show the effectiveness of our FPRDT. It is obvious that regular decision tree takes the worst performance without the consideration of adversarial examples.

Our FPRDT is better than unprovable robust algorithms RIGDT-h and GROOT experimentally, particularly for large datasets. The win/tie/loss counts show that FPRDT is clearly superior to these unprovable robust algorithms, as it wins for most times and never loses. An intuitive explanation is that RIGDT-h and GROOT make local optimization over splitting nodes without guarantee of provable robustness.

In comparison with those provably robust algorithms, our FPRDT approach achieves better performance in most cases when they return results. One possible reason is owing to the high computational complexity without reaching the optimal solutions, for example, ROCT takes exponential computational complexity yet returns result in 30 minutes as done in (Vos & Verwer, 2021b), and TREANT does not obtain results within 12 hours when datasets' cardinalities of features and instances exceed 20 and 1000, respectively; because it requires some extra computations on constraints

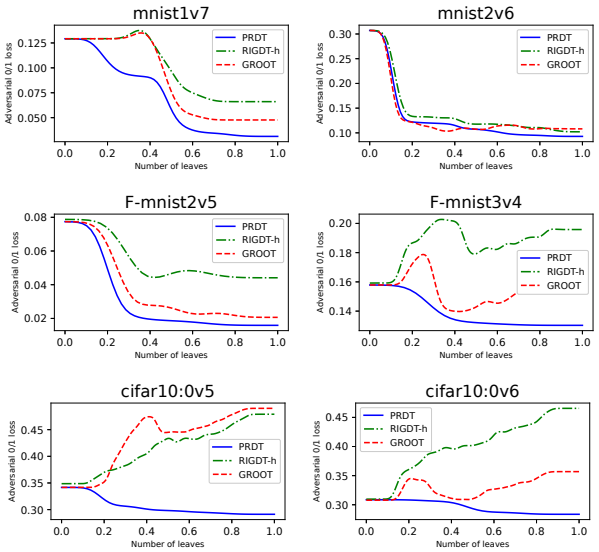

Figure 1. Comparisons of training adversarial errors in learning process, where we scale the number of leaf nodes to [0,1].

in implementation (Calzavara et al., 2020). Another possible reason is the local optimization, such as the PRB tree, which only focuses on the splitting node with its children. Our FPRDT approach keeps good balance between local and global optimizations with more information over leaf nodes.

We further exploit the convergence of adversarial $0/1$ errors during the training process between our FPRDT and those unprovably robust methods, as shown in Figure 1. It is clearly observable that our FPRDT can keep adversarial errors decrease during the whole training process, whereas those unprovably robust methods, such as RIGDT-h and GROOT, even increase the training adversarial errors on

Table 4. Comparisons of adversarial accuracies (mean±std). •/◦ indicates that our PRAdaBoost is significantly better/worse than the corresponding methods (pairwise t-tests at 95% significance level). 'N/A' means that no results were obtained after running out 12 hours.

Dataset	PRAdaBoost	AdaBoost	Random forests	RGBDT	RIGDT forests	GROOT forests	PRBoosting
ionos	.7960 ± .0329	.0321 ± .0137•	.1122 ± .0353•	.5276 ± .0472•	.6565 ± .0414•	.7869 ± .0346	.7755 ± .0370
breast	.8793 ± .0263	.0732 ± .0156•	.2167 ± .0193•	.7034 ± .0648•	.8437 ± .0280•	.8838 ± .0251	.8694 ± .0282•
diabet	.6635 ± .0281	.1352 ± .0156•	.4523 ± .0411•	.4560 ± .0371•	.5999 ± .0371•	.6578 ± .0210•	.6276 ± .0325•
bank	.6680 ± .0361	.4019 ± .0619•	.5087 ± .0292•	.4427 ± .0333•	.5087 ± .0292•	.6407 ± .0308•	.6195 ± .0403•
Japan3v4	.6816 ± .0165	.4913 ± .0231•	.5187 ± .0184•	.5885 ± .0099•	.6039 ± .0171•	.6580 ± .0160•	.6874 ± .0167
har1v2	.8601 ± .0162	.0092 ± .0065•	.8326 ± .0137•	.6417 ± .0165•	.4998 ± .0212•	.7917 ± .0169•	.8653 ± .0130
spam	.7540 ± .0129	.0000 ± .0000•	.0000 ± .0000•	.4888 ± .0349•	.6212 ± .0202•	.7495 ± .0130•	.7312 ± .0101•
GestDvP	.7315 ± .0165	.1031 ± .0079•	.1887 ± .0086•	.2420 ± .0090•	.6459 ± .0118•	.7314 ± .0127	.7318 ± .0179
wine	.6397 ± .0069	.0002 ± .0004•	.0910 ± .0112•	.1068 ± .0118•	.4161 ± .0129•	.6329 ± .0008•	.6356 ± .0066
cifar10:0v5	.6906 ± .0159	.0083 ± .0035•	.3015 ± .0058•	.3137 ± .0095•	.4413 ± .0094•	.5262 ± .0123•	N/A
cifar10:0v6	.6958 ± .0092	.0556 ± .0041•	.3678 ± .0103•	.3446 ± .0091•	.5199 ± .0082•	.5604 ± .0098•	N/A
cifar10:4v8	.6710 ± .0096	.0019 ± .0014•	.2956 ± .0074•	.2707 ± .0103•	.4614 ± .0074•	.4983 ± .0068•	N/A
mnist2v6	.9437 ± .0046	.0000 ± .0000•	.0000 ± .0000•	.8442 ± .0266•	.8999 ± .0062•	.9249 ± .0045•	.9365 ± .0044•
mnist3v8	.8829 ± .0106	.0000 ± .0000•	.0000 ± .0000•	.7155 ± .0147•	.7756 ± .0081•	.8228 ± .0057•	.8680 ± .0082•
F-mnist2v5	.9823 ± .0028	.3852 ± .0552•	.4561 ± .0041•	.9645 ± .0050•	.9654 ± .0040•	.9791 ± .0029•	.9843 ± .0018◦
F-mnist3v4	.8674 ± .0055	.0000 ± .0000•	.0441 ± .0311•	.7172 ± .0131•	.8063 ± .0064•	.8392 ± .0063•	.8640 ± .0054
F-mnist7v9	.8691 ± .0066	.0000 ± .0000•	.1357 ± .0358•	.7401 ± .0164•	.8282 ± .0067•	.8359 ± .0068•	.8779 ± .0069◦
mnist1v7	.9752 ± .0031	.0000 ± .0000•	.0000 ± .0000•	.7897 ± .1170•	.9600 ± .0032•	.9668 ± .0031•	.9781 ± .0028
Average	.7918 ± .1134	.0943 ± .1545	.2512 ± .2279	.5499 ± .2276	.6697 ± .1763	.7492 ± .1422	—
Win/Tie/Loss		**18/0/0**	**18/0/0**	**18/0/0**	**18/0/0**	**15/3/0**	**9/7/2**

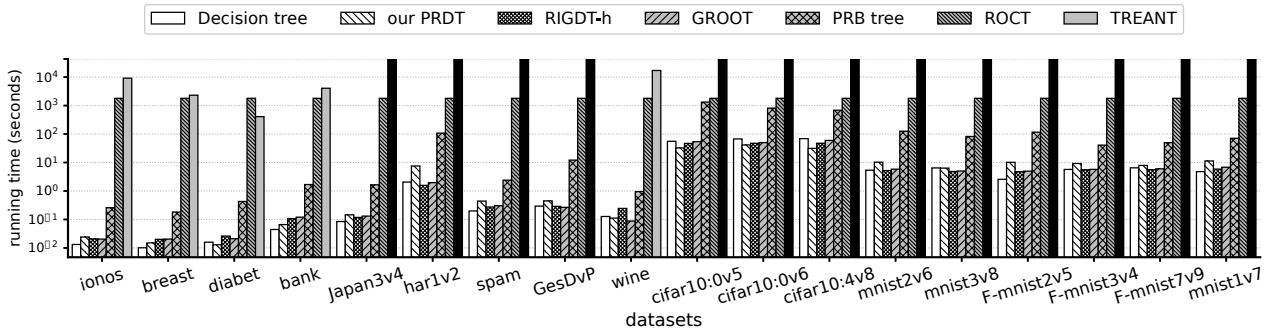

Figure 2. Comparisons of running time (in seconds) for different methods. Notice that the y-axis is in log-scale and full black columns imply that no result was obtained after running out 12 hours.

datasets F-mnist3v4 and cifar10:0v5, etc. This is because those methods optimize some lower bounds of adversarial loss without guarantee of provable robustness.

We also present the comparisons of average CPU time (in seconds) for our FPRDT and others, as shown in Figure 2. We can easily observe that our FPRDT takes comparable running time with local and unprovably robust methods RIGDT-h and GROOT, and takes the smallest running time among provably robust methods. This is nicely in agreement with the analysis of computational complexity in Table 1.

5.2. Experimental Results for Tree ensembles

In addition to original random forests (Breiman, 2001) and AdaBoost (Freund et al., 1996), we also compare with the following robust methods:

- RGBDT[7]: Robust tree ensembles based on the approximated worst loss of GBDT (Chen et al., 2019a);
- RIGDT forests[8]: Random forests with RIGDT-h as the base learner (Vos & Verwer, 2021a);
- GROOT forests[8]: Random forests with GROOT as the base learner (Vos & Verwer, 2021a);
- PRBoosting[9]: Robust tree ensembles via upper bound of exponential loss (Andriushchenko & Hein, 2019).

We do not compare with AdvBoost (Kantchelian et al., 2016) because it considers the attack under L_0 metric. We take at most 100 decision trees for forests ensemble, and select $\lfloor\sqrt{d}\rfloor$ candidate features randomly for splitting in random forests. We take the maximum depth 8 for PRBoosting due to high computational complexity, and do not restrict the depth for other methods. We also set 10 as the minimum number of instances for a splitting leaf node, and each leaf node has at least 5 instances.

[7]The code is taken from https://github.com/chenhongge.

[8]The code is taken from https://github.com/tudelft-cda-lab.

[9]The code is taken from https://github.com/max-andr.

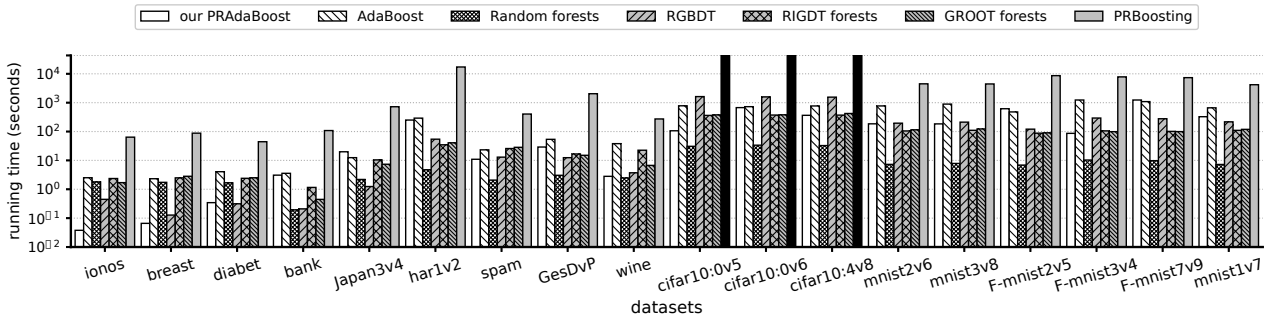

Figure 3. Comparisons of running time (in seconds) between different methods. Notice that the y-axis is in log-scale and full black columns imply that no result was obtained after running out 12 hours.

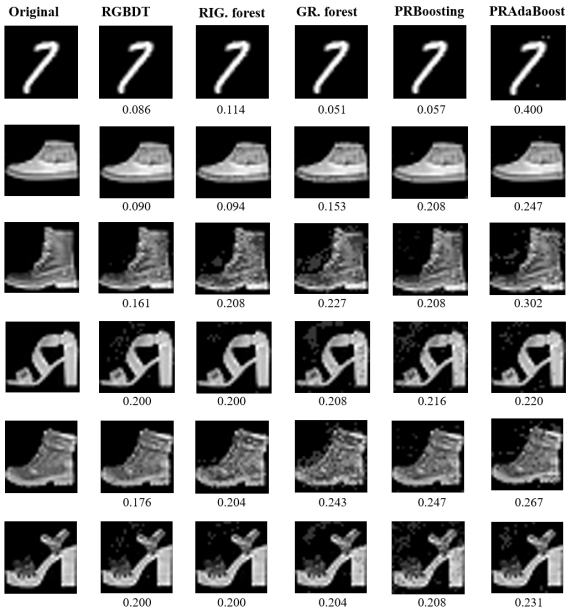

Figure 4. Visualization of the necessary minimum perturbations to change models' predictions. The larger the minimum perturbation, the better the robust method.

Table 4 presents comparisons of test adversarial accuracies for our PRAdaBoost and other ensemble methods. Our PRAdaBoost obviously outperforms original AdaBoost and random forests, which do not consider adversarial examples in training process. It is evident that our approach performs better than those unprovable robust methods RGBDT, RIGDT forests and GROOT forests experimentally, since the win/tie/loss counts show that our PRAdaBoost wins in most times and never loses.

From Table 4, it is also observable that our PRAdaBoost takes comparable performance with provably robust method PRBoosting, and an intuitive reason is that two methods optimize the same upper bound over adversarial exponential loss despite of different base learners. It is noteworthy that our PRAdaBoost takes smaller computational complexity; for example, there is no results returned by PRBoosting after running out 12 hours for dataset cifar10 of 3,072 features.

We also compare the running time of PRAdaBoost with other methods as shown in Figure 3. It is observable that our PRAdaBoost approach takes comparable running time with those unprovably robust methods RGBDT, RIGDT forests and GROOT forests, whereas our PRAdaBoost approach is about 10 times faster than the provably robust PRBoosting. This is in accordance with the analysis of computational complexity from Table 1.

We finally present the visualization of necessary minimum perturbations to change the prediction for robust methods, as shown in Figure 4. Here, perturbation denotes the distance between adversarial and original example under the L_∞ metric, and the larger the minimum perturbation, the better the robust method. As can be seen, our PRAdaBoost method takes the largest necessary minimum perturbations, which implies the strongest robustness to the adversarial defense under the L_∞ metric.

6. Conclusion

Recent years have attracted increasing attention on robust decision trees and ensembles. This work presents a fast provably robust decision tree, a tradeoff between global and local optimizations over the adversarial $0/1$ loss. We also provide provably robust boosting method based on our decision trees with theoretical guarantees. Extensive experiments are presented to verify the effectiveness and efficiency of our approaches. An interesting future work is to exploit other adversarial losses for provably robust decision trees, or look for practical and tractable upper bound over adversarial loss for robust tree ensembles methods.

Acknowledgements

The authors want to thank Dr. Daniël Vos and Sicco Verwer for their codes, and thank the reviewers for their helpful comments and suggestions. This research was supported by National Key R&D Program of China (2021ZD0112802) and NSFC (61921006, 61876078).

References

Akhtar, N., Liu, J., and Mian, A. Defense against universal adversarial perturbations. In *Proceedings of the 31th IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3389–3398, Salt Lake City, UT, 2018.

Andriushchenko, M. and Hein, M. Provably robust boosted decision stumps and trees against adversarial attacks. In *Advances in Neural Information Processing Systems 32*, pp. 12997–13008. MIT Press, Cambridge, MA, 2019.

Awasthi, P., Frank, N., Mao, A., Mohri, M., and Zhong, Y. Calibration and consistency of adversarial surrogate losses. In *Advances in Neural Information Processing Systems 34*, pp. 9804–9815. MIT Press, Cambridge, MA, 2021.

Balunović, M. and Vechev, M. Adversarial training and provable defenses: Bridging the gap. In *Proceedings of the 8th International Conference on Learning Representations*, Addis Ababa, Ethiopia, 2020.

Bartlett, P. and Traskin, M. Adaboost is consistent. In *Advances in Neural Information Processing Systems 19*, pp. 2347–2368. MIT Press, Cambridge, MA, 2006.

Breiman, L. Random forests. *Machine Learning*, 45(1): 5–32, 2001.

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

Chen, H., Zhang, H., Boning, D., and Hsieh, C.-J. Robust decision trees against adversarial examples. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 1122–1131, Long Beach, CA, 2019a.

Chen, H., Zhang, H., Si, S., Li, Y., Boning, D., and Hsieh, C.-J. Robustness verification of tree-based models. In *Advances in Neural Information Processing Systems 32*, pp. 12317–12328. MIT Press, Cambridge, MA, 2019b.

Cheng, M., Le, T., Chen, P.-Y., Zhang, H., Yi, J., and Hsieh, C.-J. Query-efficient hard-label black-box attack: An optimization-based approach. In *Proceedings of the 7th International Conference on Learning Representations*, New Orleans, LA, 2019.

Cohen, J., Rosenfeld, E., and Kolter, Z. Certified adversarial robustness via randomized smoothing. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 1310–1320, Long Beach, CA, 2019.

Freund, Y., Schapire, R. E., et al. Experiments with a new boosting algorithm. In *Proceedings of the 13th International Conference on Machine Learning*, pp. 148–156, Bari, Italy, 1996.

Gao, W. and Zhou, Z.-H. On the doubt about margin explanation of boosting. *Artificial Intelligence*, 203:1–18, 2013.

Gao, W. and Zhou, Z.-H. Towards convergence rate analysis of random forests for classification. In *Advances in Neural Information Processing Systems 33*, pp. 9300–9311. MIT Press, Cambridge, MA, 2020.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. In *Proceedings of the 3rd International Conference on Learning Representations*, San Diego, CA, 2015.

Kantchelian, A., Tygar, J. D., and Joseph, A. Evasion and hardening of tree ensemble classifiers. In *Proceedings of the 33rd International Conference on Machine Learning*, pp. 2387–2396, New York, NY, 2016.

Li, B., Chen, C., Wang, W., and Carin, L. Certified adversarial robustness with additive noise. In *Advances in Neural Information Processing Systems 32*, pp. 9459–9469. MIT Press, Cambridge, MA, 2019.

Liao, F., Liang, M., Dong, Y., Pang, T., Hu, X., and Zhu, J. Defense against adversarial attacks using high-level representation guided denoiser. In *Proceedings of the 31th IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1778–1787, Salt Lake City, UT, 2018.

Liu, X., Li, Y., Wu, C., and Hsieh, C.-J. Adv-bnn: Improved adversarial defense through robust bayesian neural network. In *Proceedings of the 6th International Conference on Learning Representations*, Vancouver, Canada, 2018.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. In *Proceedings of the 6th International Conference on Learning Representations*, Vancouver, Canada, 2018.

Mirman, M., Gehr, T., and Vechev, M. Differentiable abstract interpretation for provably robust neural networks. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 3578–3586, Stockholm, Sweden, 2018.

Papernot, N., McDaniel, P., Wu, X., Jha, S., and Swami, A. Distillation as a defense to adversarial perturbations against deep neural networks. In *Proceedings of the 37th IEEE Symposium on Security and Privacy*, pp. 582–597, San Jose, CA, 2016.

Raghunathan, A., Steinhardt, J., and Liang, P. Certified defenses against adversarial examples. In *Proceedings of the 6th International Conference on Learning Representations*, Vancouver, Canada, 2018a.

Raghunathan, A., Steinhardt, J., and Liang, P. S. Semidefinite relaxations for certifying robustness to adversarial examples. In *Advances in Neural Information Processing Systems 31*, pp. 10900–10910. MIT Press, Cambridge, MA, 2018b.

Rokach, L. and Maimon, O. Top-down induction of decision trees classifiers-a survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 35(4):476–487, 2005.

Safavian, S. R. and Landgrebe, D. A survey of decision tree classifier methodology. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(3):660–674, 1991.

Schapire, R. E. Explaining adaboost. In *Empirical Inference*, pp. 37–52. Springer, Berlin, Germany, 2013.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. In *Proceedings of the 2nd International Conference on Learning Representations*, Banff, Canada, 2014.

Vos, D. and Verwer, S. Efficient training of robust decision trees against adversarial examples. In *Proceedings of the 38th International Conference on Machine Learning*, pp. 10586–10595, 2021a.

Vos, D. and Verwer, S. Robust optimal classification trees against adversarial examples. *arXiv preprint arXiv:2109.03857*, 2021b.

Wong, E. and Kolter, Z. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 5286–5295, Stockholm, Sweden, 2018.

Wong, E., Schmidt, F. R., Metzen, J. H., and Kolter, J. Z. Scaling provable adversarial defenses. In *Advances in Neural Information Processing Systems 31*, pp. 8410–8419. MIT Press, Cambridge, MA, 2018.

Yang, B.-B., Gao, W., and Li, M. On the robust splitting criterion of random forest. In *Proceedings of the 19th IEEE International Conference on Data Mining*, pp. 1420–1425, Beijing, China, 2019.

A. The Detailed Solution of Eqn. (7) and Computational Complexity of Surrogate Losses

The detailed solution of Eqn. (7)

We will show how to determine the optimal (v_l', v_r') in Eqn. (7) with $O(1)$ computational complexity when W_t is sorted. We introduce $\hat{\mathcal{L}}_1(t,\eta), \hat{\mathcal{L}}_2(t,\eta), \hat{\mathcal{L}}_3(t,\eta), \hat{\mathcal{L}}_4(t,\eta)$, which are defined as

$$\hat{\mathcal{L}}_1(t,\eta) = F(t,\eta,0,0), \quad \hat{\mathcal{L}}_2(t,\eta) = F(t,\eta,0,1)$$
$$\hat{\mathcal{L}}_3(t,\eta) = F(t,\eta,1,0), \quad \hat{\mathcal{L}}_4(t,\eta) = F(t,\eta,1,1)$$

Once we know the values of $\hat{\mathcal{L}}_1, \hat{\mathcal{L}}_2, \hat{\mathcal{L}}_3, \hat{\mathcal{L}}_4$, Eqn. (7) can be solved in $O(1)$ computational complexity. We next show how to obtain the values of $\hat{\mathcal{L}}_1, \hat{\mathcal{L}}_2, \hat{\mathcal{L}}_3, \hat{\mathcal{L}}_4$ in in $O(1)$ computational complexity.

We introduce $k_1(t,\eta), k_2(t,\eta), k_3(t,\eta), k_4(t,\eta), k_5(t,\eta), k_6(t,\eta)$, which are defined as

$$k_1(t,\eta) = \sum_{i=1}^{n} \mathbb{I}[K_{ip} \neq 1]\mathbb{I}[\mathcal{E}_{il} \neq \emptyset]\mathbb{I}[\mathcal{E}_{ir} = \emptyset]\mathbb{I}[y_i = 0],$$
$$k_2(t,\eta) = \sum_{i=1}^{n} \mathbb{I}[K_{ip} \neq 1]\mathbb{I}[\mathcal{E}_{il} \neq \emptyset]\mathbb{I}[\mathcal{E}_{ir} = \emptyset]\mathbb{I}[y_i = 1],$$
$$k_3(t,\eta) = \sum_{i=1}^{n} \mathbb{I}[K_{ip} \neq 1]\mathbb{I}[\mathcal{E}_{il} = \emptyset]\mathbb{I}[\mathcal{E}_{ir} \neq \emptyset]\mathbb{I}[y_i = 0],$$
$$k_4(t,\eta) = \sum_{i=1}^{n} \mathbb{I}[K_{ip} \neq 1]\mathbb{I}[\mathcal{E}_{il} = \emptyset]\mathbb{I}[\mathcal{E}_{ir} \neq \emptyset]\mathbb{I}[y_i = 1],$$
$$k_5(t,\eta) = \sum_{i=1}^{n} \mathbb{I}[K_{ip} \neq 1]\mathbb{I}[\mathcal{E}_{il} \neq \emptyset]\mathbb{I}[\mathcal{E}_{ir} \neq \emptyset]\mathbb{I}[y_i = 0],$$
$$k_6(t,\eta) = \sum_{i=1}^{n} \mathbb{I}[K_{ip} \neq 1]\mathbb{I}[\mathcal{E}_{il} \neq \emptyset]\mathbb{I}[\mathcal{E}_{ir} \neq \emptyset]\mathbb{I}[y_i = 1].$$

Then we have

$$\hat{\mathcal{L}}_1 = k_2 + k_4 + k_6, \quad \hat{\mathcal{L}}_2 = k_2 + k_3 + k_5 + k_6,$$
$$\hat{\mathcal{L}}_3 = k_1 + k_4 + k_5 + k_6, \quad \hat{\mathcal{L}}_4 = k_1 + k_3 + k_5,$$

which can be verified by the definition.

Given a splitting feature t', for a sorted $W_{t'} = \{\eta_1, \eta_2, ..., \eta_q\}$ in ascending order, we can determine the values of $k_i(t', \eta_1)$ for $i = 1, 2, ..., 6$ by the definition. It takes $O(n)$ computational complexity. We will show that $k_i(t', \eta_j)$ for each $j = 2, ..., q$ can be determined by iteration in $O(1)$ computational complexity. Notice that $q = O(n)$. Thus it takes average $O(1)$ computational complexity to determine the values of $k_i(t', \eta_j)$ for each $i = 1, 2, ..., 6$ and each $j = 1, ..., q$

It can be seen that $W_{t'}$ is composed of $W_{t'}^1, W_{t'}^2$ and $W_{t'}^3$, where

$$W_{t'}^1 = \bigcup_{i=1,\mathcal{E}_{ip}\neq\emptyset}^{n} \{x_{it'} - \epsilon\}, \;\; W_{t'}^2 = \bigcup_{i=1,\mathcal{E}_{ip}\neq\emptyset}^{n} \{x_{it'}\}, \;\; W_{t'}^3 = \bigcup_{i=1,\mathcal{E}_{ip}\neq\emptyset}^{n} \{x_{it'} + \epsilon\}$$

If $\eta_j = x_{it'} - \epsilon \in W_{t'}^1$, for $y_i = 0$, we have

$$k_3(t', \eta_j) \leftarrow k_3(t', \eta_{j-1}) - 1, \quad k_5(t', \eta_j) \leftarrow k_5(t', \eta_{j-1}) + 1 \; .$$

And for $y_i = 1$, we have

$$k_4(t', \eta_j) \leftarrow k_4(t', \eta_{j-1}) - 1, \quad k_6(t', \eta_j) \leftarrow k_6(t', \eta_{j-1}) + 1 \; .$$

If $\eta_j = x_{it'} + \epsilon \in W_{t'}^3$, for $y_i = 0$, we have

$$k_5(t', \eta_j) \leftarrow k_5(t', \eta_{j-1}) - 1, \quad k_1(t', \eta_j) \leftarrow k_1(t', \eta_{j-1}) + 1 \ .$$

And for $y_i = 1$, we have

$$k_6(t', \eta_j) \leftarrow k_6(t', \eta_{j-1}) - 1, \quad k_2(t', \eta_j) \leftarrow k_2(t', \eta_{j-1}) + 1 \ .$$

Therefore, $k_i(t', \eta_j)$ for each $j = 2, ..., q$ can be determined by iteration in $O(1)$ computational complexity. And we can solve Eqn. (7) in $O(1)$ computational complexity.

The computational complexity of surrogate losses

However, once some convex surrogate losses are used in Eqn. (7), we have to solve it by iterative optimization algorithms, e.g., gradient decent. It would take $O(n)$ computational complexity to calculate the gradient at each iteration. Thus the total computational complexity is $O(Tn)$ for T iterations, much higher than $O(1)$ computational complexity.

B. Proof of Theorem. 3.1

Lemma B.1. *For a leaf p, let n_1 and n_{-1} represent the number of positive and negative samples respectively, the value of the leaf is v.*

The Gini impurity of the leaf p is defined as

$$Gini(p) = 1 - \left(\frac{n_{-1}}{n_{-1}+n_1}\right)^2 - \left(\frac{n_1}{n_{-1}+n_1}\right)^2 = \frac{2n_1 n_{-1}}{(n_{-1}+n_1)^2} \ .$$

The Mean Square loss of the leaf p is

$$\min_v \ell_{MSE}(p) = \min_v \sum_{i=1}^{n} (y_i - v)^2 \ .$$

Then the Gini impurity is essentially minimizing the Mean Square loss.

Proof. We have that

$$\begin{aligned}
\min_v \ell_{MSE}(p) &= \min_v \sum_{i=1}^{n} (y_i - v)^2 \\
&= \sum_{i=1}^{n} (y_i - \frac{n_1 - n_{-1}}{n_{-1}+n_1})^2 \\
&= \left(n_{-1}(-1 - \frac{n_1 - n_{-1}}{n_{-1}+n_1})^2 + n_1(1 - \frac{n_1 - n_{-1}}{n_{-1}+n_1})^2 \right) \\
&= 2n \cdot \frac{2n_1 n_{-1}}{(n_{-1}+n_1)^2} \\
&= 2n \cdot Gini(p)
\end{aligned}$$

Therefore, the Gini impurity is essentially minimizing the Mean Square loss.

□

Theorem B.2. *The sum of adversarial Gini impurity over leaves for robust decision trees is essentially minimizing a lower bound of*

$$\sum_{i=1}^{n} \max_{j \in [m]} \{(h(\boldsymbol{x}) - y_i)^2 \mathbb{I}[\boldsymbol{x} \in \mathcal{E}_{ij}]\} \ . \tag{13}$$

Proof. Let v_i represents the i-th leaf's value, Eqn. (13) can be written as

$$\sum_{i=1}^{n} \max_{j\in[m]}\{(v_j - y_i)^2 \mathbb{I}[\mathcal{E}_{ij} \neq \emptyset]\} .$$

To optimize the adversarial Gini impurity, at each split of the decision trees, the movement of the examples should make the Gini impurity worst (Chen et al., 2019a). Let $g(\boldsymbol{x}_i, p)$ indicate whether $(\boldsymbol{x}_i, y_i)$ falls into the p-th leaf after such movement, i.e., $g(\boldsymbol{x}_i, p) = 1$ if $(\boldsymbol{x}_i, y_i)$ falls into the p-th leaf after such movement, otherwise $g(\boldsymbol{x}_i, p) = 0$. We have that

$$g(\boldsymbol{x}_i, p) \leq \mathbb{I}[\mathcal{E}_{ip} \neq \emptyset] .$$

The adversarial Gini impurity for a leaf p can be defined as

$$AdvGini(p) = \frac{2n_1 n_{-1}}{(n_{-1} + n_1)^2} ,$$

where

$$n_1 = \sum_{i=1}^{n} g(x_i, p)\mathbb{I}[y_i = 1], n_{-1} = \sum_{i=1}^{n} g(x_i, p)\mathbb{I}[y_i = -1] .$$

Because the adversarial loss is defined on a whole tree, thus we consider the sum of the adversarial Gini impurity on a whole tree, i.e., the sum of the adversarial Gini impurity over all leaf nodes.

We have that

$$\begin{aligned}
&\min_{v_1,\ldots,v_m} \sum_{i=1}^{n} \max_{j\in[m]}\{(v_j - y_i)^2 \mathbb{I}[\mathcal{E}_{ij} \neq \emptyset]\} \\
&\geq \min_{v_1,\ldots,v_m} \sum_{i=1}^{n} \frac{1}{m} \sum_{j=1}^{m} (v_j - y_i)^2 \mathbb{I}[\mathcal{E}_{ij} \neq \emptyset] \\
&= \frac{1}{m} \sum_{j=1}^{m} \min_{v_j} \sum_{i=1}^{n} (v_j - y_i)^2 \mathbb{I}[\mathcal{E}_{ij} \neq \emptyset] \\
&\geq \frac{1}{m} \sum_{j=1}^{m} \min_{v_j} \sum_{i=1}^{n} (v_j - y_i)^2 g(\boldsymbol{x}_i, j)
\end{aligned}$$

Let $D_j = \{i\colon (\boldsymbol{x}_i, y_i) \in S_n \text{ and } g(\boldsymbol{x}_i, j) = 1\}$, we have that

$$\min_{v_j} \sum_{i=1}^{n} (v_j - y_i)^2 g(\boldsymbol{x}_i, j) = \min_{v_j} \sum_{i\in D_j} (v_j - y_i)^2 .$$

And $AdvGini(j) = \frac{2n_1 n_{-1}}{(n_{-1}+n_1)^2}$, where

$$n_1 = \sum_{i\in D_j} \mathbb{I}[y_i = 1], \; n_{-1} = \sum_{i\in D_j} \mathbb{I}[y_i = -1] .$$

According to Lemma B.1, the adversarial Gini impurity $AdvGini(j)$ is essentially minimizing $\sum_{i\in D_j}(v_j - y_i)^2$. Take the sum over $j \in [m]$ and we can see that the sum of adversarial Gini impurity is essentially minimizing a lower bound of the adversarial square loss. □

C. Modified FPRDT

In the r-th iteration of PRAdaBoost, the objective loss is a weighted objective loss :

$$\sum_{i=1}^{n} w_{r,i} \max_{j\in[m]} \left\{\ell(h(\boldsymbol{x}), y_i)\mathbb{I}[\boldsymbol{x} \in \mathcal{E}_{ij}]\right\} ,$$

We could write the weighted objective loss at one split:

$$\hat{\mathcal{L}}(t,\eta,v_l,v_r)=\sum_{i=1}^{n} w_{r,i}\max\left\{\mathbb{I}[v_l\neq y_i]\mathbb{I}[\mathcal{E}_{il}\neq\emptyset],\mathbb{I}[v_r\neq y_i]\mathbb{I}[\mathcal{E}_{ir}\neq\emptyset],K_{ip}\right\},$$

The one-split goal for robust training is to solve

$$\{t^*,\eta^*,v_l^*,v_r^*\}\in\underset{t,\eta,v_l,v_r}{\arg\min}\left\{\hat{\mathcal{L}}(t,\eta,v_l,v_r)\right\}.$$

The term K_{ip} is calculated as FPRDT. We traverse all possible values of t,η. For any fixed t' and η', we solve

$$(v_l',v_r')=\underset{v_l,v_r\in\{0,1\}}{\arg\min}\left\{\hat{\mathcal{L}}(t',\eta',v_l,v_r)\right\},$$

The algorithm to solve it is similar to FPRDT.

Following Appendix A, we just need to change the definition of k_1,k_2,k_3,k_4,k_5 and k_6 as

$$\begin{aligned}
k_1(t,\eta)&=\sum_{i=1}^{n} w_{r,i}\mathbb{I}[K_{ip}\neq 1]\mathbb{I}[\mathcal{E}_{il}\neq\emptyset]\mathbb{I}[\mathcal{E}_{ir}=\emptyset]\mathbb{I}[y_i=0],\\
k_2(t,\eta)&=\sum_{i=1}^{n} w_{r,i}\mathbb{I}[K_{ip}\neq 1]\mathbb{I}[\mathcal{E}_{il}\neq\emptyset]\mathbb{I}[\mathcal{E}_{ir}=\emptyset]\mathbb{I}[y_i=1],\\
k_3(t,\eta)&=\sum_{i=1}^{n} w_{r,i}\mathbb{I}[K_{ip}\neq 1]\mathbb{I}[\mathcal{E}_{il}=\emptyset]\mathbb{I}[\mathcal{E}_{ir}\neq\emptyset]\mathbb{I}[y_i=0],\\
k_4(t,\eta)&=\sum_{i=1}^{n} w_{r,i}\mathbb{I}[K_{ip}\neq 1]\mathbb{I}[\mathcal{E}_{il}=\emptyset]\mathbb{I}[\mathcal{E}_{ir}\neq\emptyset]\mathbb{I}[y_i=1],\\
k_5(t,\eta)&=\sum_{i=1}^{n} w_{r,i}\mathbb{I}[K_{ip}\neq 1]\mathbb{I}[\mathcal{E}_{il}\neq\emptyset]\mathbb{I}[\mathcal{E}_{ir}\neq\emptyset]\mathbb{I}[y_i=0],\\
k_6(t,\eta)&=\sum_{i=1}^{n} w_{r,i}\mathbb{I}[K_{ip}\neq 1]\mathbb{I}[\mathcal{E}_{il}\neq\emptyset]\mathbb{I}[\mathcal{E}_{ir}\neq\emptyset]\mathbb{I}[y_i=1].
\end{aligned}$$

And the updated rules are changed as

If $\eta_j=x_{it'}-\epsilon\in W_{t'}^1$, for $y_i=0$, we have

$$k_3(t',\eta_j)\leftarrow k_3(t',\eta_{j-1})-w_{r,i},\quad k_5(t',\eta_j)\leftarrow k_5(t',\eta_{j-1})+w_{r,i}.$$

And for $y_i=1$, we have

$$k_4(t',\eta_j)\leftarrow k_4(t',\eta_{j-1})-w_{r,i},\quad k_6(t',\eta_j)\leftarrow k_6(t',\eta_{j-1})+w_{r,i}.$$

If $\eta_j=x_{it'}+\epsilon\in W_{t'}^3$, for $y_i=0$, we have

$$k_5(t',\eta_j)\leftarrow k_5(t',\eta_{j-1})-w_{r,i},\quad k_1(t',\eta_j)\leftarrow k_1(t',\eta_{j-1})+w_{r,i}.$$

And for $y_i=1$, we have

$$k_6(t',\eta_j)\leftarrow k_6(t',\eta_{j-1})-w_{r,i},\quad k_2(t',\eta_j)\leftarrow k_2(t',\eta_{j-1})+w_{r,i}.$$

Therefore, $k_i(t',\eta_j)$ for each $j=2,...,q$ can be determined by iteration in $O(1)$ computational complexity. And we can solve Eqn. (7) in $O(1)$ computational complexity.

D. Proof of Theorem. 4.1

Lemma D.1. *For functions $f_i, i = 1, ..., m$ with S as their domain. If $f_i(x) > 0$ for all $x \in S$ and $i \in [m]$, we have that*

$$\max_{x \in S} \prod_{i=1}^{m} f_i(x) \leq \prod_{i=1}^{m} \max_{x \in S} f_i(x) \; .$$

Proof. Suppose $x^* \in \arg\max_{x \in S} \prod_{i=1}^{m} f_i(x)$, and $x_i^* \in \max_{x \in S} f_i(x)$. Thus we have that

$$f_i(x^*) \leq f_i(x_i^*)$$

Therefore, we have that

$$\max_{x \in S} \prod_{i=1}^{m} f_i(x) = \prod_{i=1}^{m} f_i(x^*) \leq \prod_{i=1}^{m} f_i(x_i^*) = \prod_{i=1}^{m} \max_{x \in S} f_i(x) \; .$$

□

Theorem D.2. *Let $H(\boldsymbol{x}) = sign(\sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x}))$ be the final classifier of PRAdaBoost, with $\gamma_t = 1/2 - \epsilon_t > 0$ for each iteration $t \in [T]$. We have the empirical adversarial $0/1$ loss over training sample S_n as follows:*

$$\frac{1}{n} \sum_{i=1}^{n} \left[\max_{\boldsymbol{x}_i' \in \mathcal{N}_\epsilon(\boldsymbol{x}_i)} \mathbb{I}[H(\boldsymbol{x}_i') \neq y_i] \right] \leq \exp\left(-2 \sum_{t=1}^{T} \gamma_t^2 \right) \; .$$

Proof. Let $F(\boldsymbol{x}) = \sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x})$ and define $\mathcal{D}_t((\boldsymbol{x}_i, y_i)) = w_{t,i} / \sum_{i=1}^{n} w_{t,i}$. Therefore, $\mathcal{D}_t$ is a distribution over the training set in the t-th iteration. Notice that $w_{1,i} = 1$ for $i \in [n]$.

We define

$$Z_t = \sum_{i=1}^{n} \mathcal{D}_t((\boldsymbol{x}_i, y_i)) \max_{\boldsymbol{x}_i' \in \mathcal{N}_\epsilon(\boldsymbol{x}_i)} \left\{ \exp(-y \alpha_t h_t(\boldsymbol{x}_i')) \right\} . \tag{14}$$

And we have that

$$\begin{aligned} Z_t &= \sum_{i=1}^{n} \mathcal{D}_t((\boldsymbol{x}_i, y_i)) \max_{\boldsymbol{x}_i' \in \mathcal{N}_\epsilon(\boldsymbol{x}_i)} \left\{ \exp(-y \alpha_t h_t(\boldsymbol{x}_i')) \right\} \\ &= \sum_{i=1}^{n} \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}} \max_{\boldsymbol{x}_i' \in \mathcal{N}_\epsilon(\boldsymbol{x}_i)} \left\{ \exp(-y \alpha_t h_t(\boldsymbol{x}_i')) \right\} \qquad (15) \\ &= \frac{\sum_{i=1}^{n} w_{t+1,i}}{\sum_{i=1}^{n} w_{t,i}} \; . \qquad (16) \end{aligned}$$

Eqn. (15) holds by the definition of $\mathcal{D}_t$. Eqn. (16) use the update rule Eqn. (12).

From Eqn. (16), we have $\sum_{i=1}^{n} w_{t+1,i} = Z_t \sum_{i=1}^{n} w_{t,i}$, and it follows that

$$\mathcal{D}_{T+1}((\boldsymbol{x}, y)) = \frac{w_{T+1,i}}{\sum_{i=1}^{n} w_{T+1,i}} = \frac{w_{T+1,i}}{Z_T \sum_{i=1}^{n} w_{T,i}} = \cdots = \frac{w_{T+1,i}}{(\prod_{t=1}^{T} Z_t) \sum_{i=1}^{n} w_{1,i}} = \frac{1}{n} \frac{w_{T+1,i}}{(\prod_{t=1}^{T} Z_t)} \; . \tag{17}$$

Unraveling the recurrence of Eqn. (12) gives

$$w_{T+1,i} = \max_{\boldsymbol{x}' \in \mathcal{N}_\epsilon(\boldsymbol{x})} \exp(-y \alpha_1 h_1(\boldsymbol{x}')) \times \cdots \times \max_{\boldsymbol{x}' \in \mathcal{N}_\epsilon(\boldsymbol{x})} \exp(-y \alpha_T h_T(\boldsymbol{x}')) \; . \tag{18}$$

Combining Equ. (17) and Equ. (18), we obtain that

$$\begin{aligned} \mathcal{D}_{T+1}((\boldsymbol{x}, y)) &= \frac{1}{n} \frac{\max_{\boldsymbol{x}' \in \mathcal{N}_\epsilon(\boldsymbol{x})} \exp(-y \alpha_1 h_1(\boldsymbol{x}')) \times \cdots \times \max_{\boldsymbol{x}' \in \mathcal{N}_\epsilon(\boldsymbol{x})} \exp(-y \alpha_T h_T(\boldsymbol{x}'))}{\prod_{t=1}^{T} Z_t} \\ &= \frac{\prod_{t=1}^{T} \max_{\boldsymbol{x}_i' \in \mathcal{N}_\epsilon(\boldsymbol{x}_i)} \left\{ \exp(-y_i \alpha_t h_t(\boldsymbol{x}_i')) \right\}}{n \prod_{t=1}^{T} Z_t} \; . \qquad (19) \end{aligned}$$

Since $H(\boldsymbol{x}) = \text{sign}(F(\boldsymbol{x}))$, if there exists $\boldsymbol{x}' \in \mathcal{N}_\epsilon(\boldsymbol{x})$ satisfying $H(\boldsymbol{x}') \neq y$, then $\min_{\boldsymbol{x}' \in \mathcal{N}_\epsilon(\boldsymbol{x})} yF(\boldsymbol{x}') \leq 0$, which implies that $\max_{\boldsymbol{x}' \in \mathcal{N}_\epsilon(\boldsymbol{x})} \exp(-yF(\boldsymbol{x}')) \geq 1$. That is

$$\max_{\boldsymbol{x}' \in \mathcal{N}_\epsilon(\boldsymbol{x})} \mathbb{I}[H(\boldsymbol{x}') \neq y] \leq \max_{\boldsymbol{x}' \in \mathcal{N}_\epsilon(\boldsymbol{x})} \exp(-yF(\boldsymbol{x}')) \,. \tag{20}$$

For adversarial training loss, we have that

$$\begin{aligned}
&\frac{1}{n}\sum_{i=1}^{n}\left[\max_{\boldsymbol{x}_i' \in \mathcal{N}_\epsilon(\boldsymbol{x}_i)} \mathbb{I}[H(\boldsymbol{x}_i') \neq y_i]\right] \\
&\leq \frac{1}{n}\sum_{i=1}^{n} \max_{\boldsymbol{x}_i' \in \mathcal{N}_\epsilon(\boldsymbol{x}_i)} \left\{\exp(-y_i F(\boldsymbol{x}_i'))\right\} &(21)\\
&= \frac{1}{n}\sum_{i=1}^{n} \max_{\boldsymbol{x}_i' \in \mathcal{N}_\epsilon(\boldsymbol{x}_i)} \left\{\prod_{t=1}^{T}\exp(-y_i\alpha_t h_t(\boldsymbol{x}_i'))\right\} \\
&\leq \frac{1}{n}\sum_{i=1}^{n}\prod_{t=1}^{T} \max_{\boldsymbol{x}_i' \in \mathcal{N}_\epsilon(\boldsymbol{x}_i)} \left\{\exp(-y_i\alpha_t h_t(\boldsymbol{x}_i'))\right\} &(22)\\
&= \sum_{i=1}^{n} \mathcal{D}_{T+1}((\boldsymbol{x}_i, y_i))\prod_{t=1}^{T} Z_t &(23)\\
&= \prod_{t=1}^{T} Z_t \,. &(24)
\end{aligned}$$

Eqn. (21), Eqn. (22) and Eqn. (23) holds by Eqn. (20), Lemma D.1 and Eqn. (19), respectively. Eqn. (24) uses the fact that $\mathcal{D}_{T+1}$ is a distribution, which sums to 1.

Finally, by the definition Eqn. (14) of Z_t , we have that

$$\begin{aligned}
Z_t &= \sum_{i=1}^{n} \mathcal{D}_t((\boldsymbol{x}_i, y_i)) \max_{\boldsymbol{x}_i' \in \mathcal{N}_\epsilon(\boldsymbol{x}_i)} \left\{\exp(-\alpha_t y_i h_t(\boldsymbol{x}_i'))\right\} \\
&= \sum_{i \in S_1} \mathcal{D}_t((\boldsymbol{x}_i, y_i))\exp(-\alpha_t) + \sum_{i \in S_2} \mathcal{D}_t((\boldsymbol{x}_i, y_i))\exp(\alpha_t) &(25)\\
&= \exp(-\alpha_t)(1-\epsilon_t) + \exp(\alpha_t)\epsilon_t &(26)\\
&= \exp(-\alpha_t)(\frac{1}{2}+\gamma_t) + \exp(\alpha_t)(\frac{1}{2}-\gamma_t) &(27)\\
&= \sqrt{1-4\gamma_t^2} &(28)\\
&\leq \exp(-2\gamma_t^2) \,. &(29)
\end{aligned}$$

where S_1 and S_2 is defined as

$$S_2 = \{i \in [n] \colon \min_{\boldsymbol{x}_i' \in \mathcal{N}_\epsilon(\boldsymbol{x}_i)} y_i h_t(\boldsymbol{x}_i') = -1\} \,, \quad S_1 = [n] \setminus S_2 \,.$$

Eqn. (25) uses the fact that both y_i and $h(\boldsymbol{x}_i)$ are $\{-1, +1\}$-valued. Eqn. (26) and Eqn. (27) follows from the definition of ϵ_t and γ_t, respectively. Eqn. (28) follows from the definition of α_t. For Eqn. (29), we simply apply the approximation $1 + x \leq e^x$ for all real x. Thus, we have that

$$\frac{1}{n}\sum_{i=1}^{n}\left[\max_{\boldsymbol{x}' \in \mathcal{N}_\epsilon(\boldsymbol{x}_i)} \mathbb{I}[H(\boldsymbol{x}') \neq y_i]\right] \leq \exp\left(-2\sum_{t=1}^{T}\gamma_t^2\right) \,.$$

□

E. Experiments

Experimental settings

The used hyperparameters are summarized in Table 5 and Table 6.

Table 5. Hyperparameters of all tree models used in our experiments. Parameters that were not applicable were left blank.

Parameter	FPRDT	Decision tree	RIGDT-h	GROOT	TREANT	ROCT	PRB tree
max depth	None	None	None	None	4	4	None
min_samples_split	10	10	10	10	10	10	10
min_samples_leaf	5	5	5	5	5	5	5
affine	-	-	-	-	-	-	False

Table 6. Hyperparameters of all tree ensemble models used in our experiments. Parameters that were not applicable were left blank.

Parameter	PRAdaBoost	AdaBoost	Random forests	RGBDT	RIGDT forests	GROOT forests	PRB
max depth	None	None	None	None	None	None	4
min_samples_split	10	10	10	-	10	10	10
min_samples_leaf	5	5	5	-	5	5	5
n_estimators	100	100	100	100	100	100	100
η	-	-	-	0.2	-	-	0.2
γ	-	-	-	1.0	-	-	-
min_child_weight	-	-	-	1	-	-	-

Except for n_estimators and max depth, the values were copied from their original works (Chen et al., 2019a; Andriushchenko & Hein, 2019; Calzavara et al., 2020; Vos & Verwer, 2021a;b). Note that all methods don't prune the tree except the PRB method. We do not prune our trees because our FPRDT generally prefers to generate a shallow tree, which avoids overfitting.

The curve of adversarial loss in the learning process

We present more comparisons of the convergence of adversarial errors during the training process between FPRDT and other unprovable methods, as shown in Figure 5.

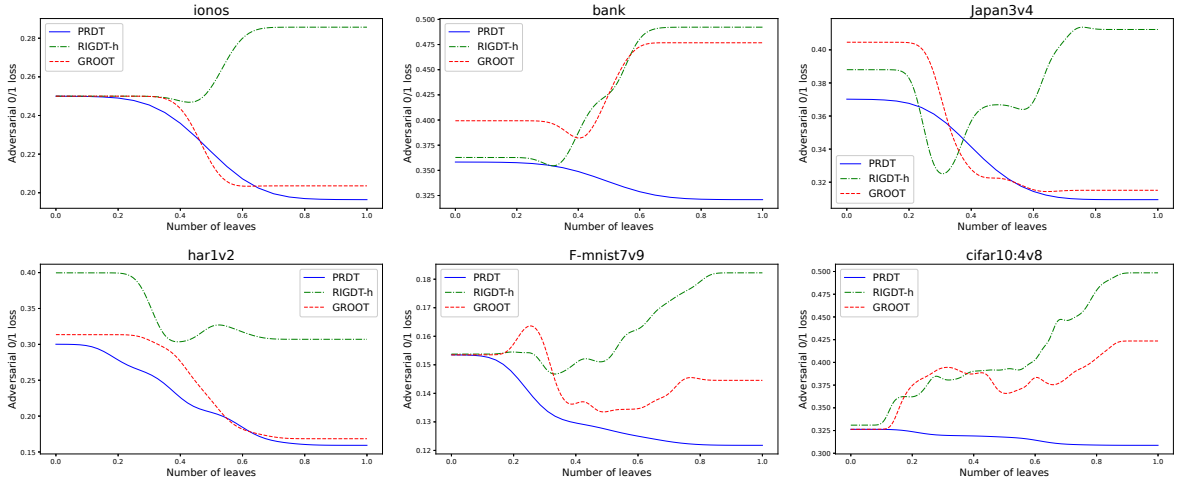

Figure 5. Comparisons of training adversarial errors between FPRDT and other unprovable methods in learning process, where we scale the number of leaf nodes to [0,1].

As can be seen, our FPRDT can still keep adversarial errors decrease during the whole training process, whereas those unprovably robust methods, such as RIGDT-h and GROOT, may increase the training adversarial errors in some datasets.

The depth of the robust trees

We summarize the average depth of the robust tree methods in Table 7.

Table 7. Average depth over 18 datasets. We use FPRDT as the baseline here.

Methods	FPRDT	Decision tree	RIGDT-h	GROOT	ROCT	PRB tree
Average depth	1x	0.88x	1.84x	1.81x	1.84x	0.89x

We omit TREANT method because it can not be trained successfully on most datasets within 12 hours. Note that the depth of FPRDT is similar to PRB tree, which is the only method that considers pruning.