Robust Decision Trees Against Adversarial Examples

Hongge Chen 1  Huan Zhang 2  Duane Boning 1  Cho-Jui Hsieh 2

Abstract
Although adversarial examples and model robustness have been extensively studied in the context of linear models and neural networks, research on this issue in tree-based models and how to make tree-based models robust against adversarial examples is still limited. In this paper, we show that tree based models are also vulnerable to adversarial examples and develop a novel algorithm to learn robust trees. At its core, our method aims to optimize the performance under the worst-case perturbation of input features, which leads to a max-min saddle point problem. Incorporating this saddle point objective into the decision tree building procedure is non-trivial due to the discrete nature of trees—a naive approach to finding the best split according to this saddle point objective will take exponential time. To make our approach practical and scalable, we propose efficient tree building algorithms by approximating the inner minimizer in this saddle point problem, and present efficient implementations for classical information gain based trees as well as state-of-the-art tree boosting models such as XGBoost. Experimental results on real world datasets demonstrate that the proposed algorithms can substantially improve the robustness of tree-based models against adversarial examples.

1. Introduction
The discovery of adversarial examples in various deep learning models (Szegedy et al., 2013; Kos et al., 2018; Cheng et al., 2018; Chen et al., 2018a; Carlini & Wagner, 2018; Huang et al., 2017) has led to extensive studies of deep neural network (DNN) robustness under such maliciously crafted subtle perturbations. Although deep learning-based model robustness has been well-studied in the recent literature from both attack and defense perspectives, studies on the robustness of tree-based models are quite limited (Papernot et al., 2016a).

In our paper, we shed light on the adversarial robustness of an important class of machine learning models — decision trees. Among machine learning models used in practice, tree-based methods stand out in many applications, with state-of-the-art performance. Tree-based methods have achieved widespread success due to their simplicity, efficiency, interpretability, and scalability on large datasets. They have been suggested as an advantageous alternative to deep learning in some cases (Zhou & Feng, 2017). In this paper, we study the robustness of tree-based models under adversarial attacks, and more importantly, we propose a novel robust training framework for tree-based models. Below we highlight our major contributions:

• We study the robustness of decision tree-based machine learning algorithms through the lens of adversarial examples. We study both classical decision trees and state-of-the-art ensemble boosting methods such as XGBoost. We show that, similar to neural networks, tree-based models are also vulnerable to adversarial examples.
• We propose a novel robust decision tree training framework to improve robustness against adversarial examples. This method seeks to optimize the worst case condition by solving a max-min problem. This framework is quite general and can be applied to tree-based models with any score function used to choose splitting thresholds. To the best of our knowledge, this is the first work contributing a general robust decision tree training framework against adversarial examples.
• We implement our framework in both classical information gain based classification trees and state-of-the-art large-scale tree boosting systems. To scale up our framework, we make necessary and efficient approximations to handle complex models and real world data sets. Our experimental results show consistent and substantial improvements on adversarial robustness.

2. Related Works

2.1. Decision Tree and Gradient Boosted Decision Tree
Decision tree learning methods are widely used in machine learning and data mining. As considered here, the goal is to
create a tree structure with each interior node corresponding to one of the input features. Each interior node has two children, and edges to child nodes represent the split condition for that feature. Each leaf provides a prediction value of the model, given that the input features satisfy the conditions represented by the path from the root to that leaf. In practice, decision tree learning algorithms are based on greedy search, which builds a tree starting from its root by making locally optimal decisions at each node. Classical decision tree training recursively chooses features, sets thresholds and splits the examples on a node by maximizing a pre-defined score, such as information gain or Gini impurity.

Decision trees are often used within ensemble methods. A well-known gradient tree boosting method has been developed by Friedman et al. (2000); Friedman (2001) and Friedman (2002) to allow optimization of an arbitrary differentiable loss function. Later scalable tree boosting systems have been built to handle large datasets. For example, pGBRT (Tyree et al., 2011) parallelizes the training procedure by data partitioning for faster and distributed training. XGBoost (Chen & Guestrin, 2016) is a prominent tree boosting software framework; in data mining contests, 17 out of 29 published winning solutions at Kaggle’s blog in 2015 used XGBoost in their models. LightGBM (Ke et al., 2017; Zhang et al., 2018) is another highly efficient boosting framework that uses histograms on data features to significantly speed up training. mGBDT (Feng et al., 2018) learns hierarchical representations by stacking multiple layers of gradient boosted decision trees (GBDTs). Other variants such as extreme multi-label GBDT (Si et al., 2017) and cost efficient tree boosting approaches (Peter et al., 2017; Xu et al., 2019) have also been proposed recently.

2.2. Adversarial Attack for Deep Neural Networks

An adversarial attack is a subtle modification of a benign example. In a successful attack, the classifier will misclassify this modified example, while the original example is correctly classified. Such attacks can be roughly divided into two categories, white-box attacks and black-box attacks. White-box attacks assume that the model is fully exposed to the attacker, including parameters and structures, while in black-box attacks, the attacker can query the model but has no (direct) access to any internal information inside the model. FGSM (Goodfellow et al., 2015) is one of the first methods in the white-box attack category. It computes the gradient only once to generate an adversarial example. This method is strengthened as Iterative-FGSM (or I-FGSM) (Kurakin et al., 2017), which applies FGSM multiple times for a higher attack success rate and smaller distortion. C&W attack (Carlini & Wagner, 2017) formulates the attack as an optimization problem with an $\ell_2$ penalty. EAD-L1 attack (Chen et al., 2018b) uses a more general formulation than C&W attack with elastic-net regularization. To bypass some defenses with obfuscated gradients, the BPDA attack introduced by Athalye et al. (2018) is shown to successfully circumvent many defenses.

The white-box setting is often argued as being unrealistic in the literature. In contrast, several recent works have studied ways to fool the model given only model output scores or probabilities. Methods in Chen et al. (2017) and Ilyas et al. (2017) are able to craft adversarial examples by making queries to obtain the corresponding probability outputs of the model. A stronger and more general attack has been developed recently by Cheng et al. (2019), which does not rely on the gradient nor the smoothness of model output. This enables attackers to successfully attack models that only output hard labels.

2.3. Defenses for Deep Neural Networks

It is difficult to defend against adversarial examples, especially under strong and adaptive attacks. Some early methods, including feature squeezing (Xu et al., 2017) and defensive distillation (Papernot et al., 2016b) have been proven ineffective against stronger attacks like C&W. Many recently proposed defense methods are based on obfuscated gradients (Guo et al., 2017; Song et al., 2017; Buckman et al., 2018; Ma et al., 2018; Samangouei et al., 2018) and are already overcome by the aforementioned BPDA attack.

Adversarial training, first introduced in Kurakin et al. (2017), is effective on DNNs against various attacks. In adversarial training, adversarial examples are generated during the training process and are used as training data to increase model robustness. This technique has been formally posed as a min-max robust optimization problem in Madry et al. (2018) and has achieved very good performance under adversarial attacks. Several recent work have tried to improve over the original adversarial training formulation (Liu & Hsieh, 2019; Liu et al., 2019; Zhang et al., 2019). There are some other methods in the literature seeking to give provable guarantees on the robustness performance, such as distributional robust optimization (Sinha et al., 2018), convex relaxations (Wong & Kolter, 2018; Wong et al., 2018; Wang et al., 2018) and semidefinite relaxations (Raghunathan et al., 2018). Some of these methods can be deployed in medium-sized networks and achieve satisfactory robustness.

However, all of the current defense methods assume the model to be differentiable and use gradient based optimizers, so none of them can be directly applied to decision tree based models, which are discrete and non-differentiable.

3. Adversarial Examples of Decision Tree Based Models

Recent developments in machine learning have resulted in the deployment of large-scale tree boosting systems in
critical applications such as fraud and malware detection. Unlike deep neural networks (DNNs), tree based models are non-smooth, non-differentiable and sometimes interpretable, which might lead to the belief that they are more robust than DNNs. However, the experiments in our paper show that similar to DNNs, tree-based models can also be easily compromised by adversarial examples. In this paper, we focus on untargeted attacks, which are considered to be successful as long as the model misclassifies the adversarial examples.

Unlike DNNs, algorithms for crafting adversarial examples for tree-based models are poorly studied. The main reason is that tree-based models are discrete and non-differentiable, thus we cannot use common gradient descent based methods for white-box attack. An early attack algorithm designed for single decision trees has been proposed by Papernot et al. (2016a), based on greedy search. To find an adversarial example, this method searches the neighborhood of the leaf which produces the original prediction, and finds another leaf labeled as a different class by considering the path from the original leaf to the target leaf, and changing the feature values accordingly to result in misclassification.

A white-box attack against binary classification tree ensembles has been proposed by Kantchelian et al. (2016). This method finds the exact smallest distortion (measured by some $\ell_p$ norm) necessary to mislead the model. However, the algorithm relies on Mixed Integer Linear Programming (MILP) and thus can be very time-consuming when attacking large scale tree models as arise in XGBoost. In this paper, we use the $\ell_\infty$ version of Kantchelian’s attack as one of our methods to evaluate small and mid-size binary classification model robustness. Kantchelian et al. (2016) also introduce a faster approximation to generate adversarial examples using symbolic prediction with $\ell_0$ norm minimization and combine this method into an adversarial training approach. Unfortunately, the demonstrated adversarial training is not very effective; despite increasing model robustness for $\ell_0$ norm perturbations, robustness for $\ell_1$, $\ell_2$ and $\ell_\infty$ norm perturbations are noticeably reduced compared to the naturally (non-robustly) trained model.

In our paper, in addition to Kantchelian attacks we also use a general attack method proposed in Cheng et al. (2019) which does not rely on the gradient nor the smoothness of output of a machine learning model. Cheng’s attack method has been used to efficiently evaluate the robustness of complex models on large datasets, even under black-box settings. To deal with non-smoothness of model output, this method focuses on the distance between the benign example and the decision boundary, and reformulates the adversarial attack as a minimization problem of this distance. Despite the non-smoothness of model prediction, the distance to decision boundary is usually smooth within a local region, and can be found by binary search on vector length given a direction vector. To minimize this distance without gradient, Cheng et al. (2019) used a zeroth order optimization algorithm with a randomized gradient-free method. In our paper, we use the $\ell_\infty$ version of Cheng’s attack.

Some adversarial examples obtained by this method are shown in Figure 1, where we display results on both MNIST and Fashion-MNIST datasets. The models we test are natural GBDT models trained using XGBoost and our robust GBDT models, each with 200 trees and a tree depth of 8. Cheng’s attack is able to craft adversarial examples with very small distortions on natural models; for human eyes, the adversarial distortion added to the natural model’s adversarial examples appear as imperceptible noise. We also conduct white-box attacks using the MILP formulation (Kantchelian et al., 2016), which takes much longer time to solve but the $\ell_\infty$ distortion found by MILP is comparable to Cheng’s method; see Section 5 for more details. In contrast, for our robust GBDT model, the required adversarial example distortions are so large that we can even vaguely see a number 8 in subfigure (c). The substantial increase in the $\ell_\infty$ distortion required to misclassify as well as the increased visual impact of such distortions shows the effectiveness of our robust decision tree training, which we will introduce in detail next. In the main text, we use the $\ell_\infty$ version of Kantchelian’s attack; we present results of $\ell_1$ and $\ell_2$ Kantchelian attacks in the appendix.

<table>
<thead>
<tr>
<th></th>
<th>Original</th>
<th>Adversarial of nat. GBDT</th>
<th>Adversarial of rob. GBDT</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td><img src="mnist" alt="2" /></td>
<td><img src="mnist_adv_nat0" alt="2" /></td>
<td><img src="mnist_adv_rob0" alt="2" /></td>
</tr>
<tr>
<td>pred. = 2</td>
<td></td>
<td>$\ell_\infty$ dist. = 0.069</td>
<td>$\ell_\infty$ dist. = 0.344</td>
</tr>
<tr>
<td>(d)</td>
<td><img src="fashion" alt="Ankle Boot" /></td>
<td><img src="fashion_adv_nat0" alt="Ankle Boot" /></td>
<td><img src="fashion_adv_rob0" alt="Ankle Boot" /></td>
</tr>
<tr>
<td>pred. = “Ankle Boot”</td>
<td></td>
<td>$\ell_\infty$ dist. = 0.074</td>
<td>$\ell_\infty$ dist. = 0.394</td>
</tr>
</tbody>
</table>

Figure 1. MNIST and Fashion-MNIST examples and their adversarial examples found using the untargeted attack proposed by Cheng et al. (2019) on 200-tree gradient boosted decision tree (GBDT) models trained using XGBoost with depth=8. Natural GBDT models (nat.) are fooled by small $\ell_\infty$ perturbations (b, e), while our robust (rob.) GBDT models require much larger perturbations (c, f) for successful attacks. For both MNIST and Fashion-MNIST robust models, we use $\epsilon = 0.3$ (a robust training hyper-parameter which will be introduced in Section 4). More examples are shown in the appendix.
Robust Decision Trees Against Adversarial Examples

4. Robust Decision Trees

4.1. Intuition

As shown in Section 3, tree-based models are vulnerable to adversarial examples. Thus it is necessary to augment the classical natural tree training procedure in order to obtain reliable models robust against adversarial attacks. Our method formulates the process of optimally finding best split threshold in decision tree training as a robust optimization problem. As a conceptual illustration, Figure 2 presents a special case where the traditional greedy optimal splitting may yield non-robust models. A horizontal split achieving high accuracy or score on original points may be easily compromised by adversarial perturbations. On the other hand, we are able to select a better vertical split considering positions of the two different splits in Figure 2 using a dichotomy on the robustness is straightforward, as seen in the two different splits in Figure 2 using \( x^{(1)} \) (a robust feature) and \( x^{(2)} \) (a non-robust feature).

4.2. General Robust Decision Tree Framework

In this section we formally introduce our robust decision tree training framework. For a training set with \( N \) examples and \( d \) real valued features \( \mathcal{D} = \{(\mathbf{x}_i, y_i)\} \ (1 \leq i \leq N, \ y_i \in \mathbb{R}, \ \mathbf{x}_i = [x_i^{(1)}, x_i^{(2)}, \ldots, x_i^{(j)}, \ldots, x_i^{(d)}] \in \mathbb{R}^d) \), we first normalize the feature values to \([0,1]d\) (the best feature value for split will also be scaled accordingly, but it is irrelevant to model performance). For a general decision tree based learning model, at a given node, we denote \( \mathcal{I} \subseteq \mathcal{D} \) as the set of points at that node. For a split on the \( j \)-th feature with a threshold \( \eta \), the sets that will be mentioned in Sections 4.2, 4.3 and 4.4 are summarized in Table 1.

In classical tree based learning algorithms (which we refer to as “natural” trees in this paper), the quality of a split on a node can be gauged by a score function \( S(.): \) a function of the splits on left and right child nodes (\( I_L \) and \( I_R \)), or equivalently on the chosen feature \( j \) to split and a corresponding threshold value \( \eta \). Since \( I_L \) and \( I_R \) are determined by \( j, \eta \) and \( \mathcal{I} \), we abuse the notation and define \( S(j, \eta, \mathcal{I}) := S(I_L, I_R) \).

Traditionally, people consider different scores for choosing the “best” split, such as information gain used by ID3 (Quinlan, 1986) and C4.5 (Quinlan, 1986), or Gini impurity in CART (Breiman, 1984). Modern software packages (Chen & Guestrin, 2016; Ke et al., 2017; Dorogush et al., 2018) typically find the best split that minimize a loss function directly, allowing decision trees to be used in a large class of problems (i.e., mean square error loss for regression, logistic loss for classification, and ranking loss for ranking problems). A regular (“natural”) decision tree training process will either exactly or approximately evaluate the score function, for all possible features and split thresholds on the leaf to be split, and select the best \( j, \eta \) pair:

\[
\begin{align*}
\hat{j}, \hat{\eta} & = \arg \max_{j, \eta} S(I_L, I_R) = \arg \max_{j, \eta} S(j, \eta, \mathcal{I}).
\end{align*}
\]

In our setting, we consider the case where features of examples in \( I_L \) and \( I_R \) can be perturbed by an adversary. Since a typical decision tree can only split on a single feature at one time, it is natural to consider adversarial perturbations within an \( \ell_\infty \) ball of radius \( \epsilon \) around each example \( \mathbf{x}_i \):

\[
B^\infty_\epsilon(\mathbf{x}_i) := [x_i^{(1)} - \epsilon, x_i^{(1)} + \epsilon] \times \cdots \times [x_i^{(d)} - \epsilon, x_i^{(d)} + \epsilon].
\]
Such perturbations enable the adversary to minimize the score obtained by our split. So instead of finding a split with highest score, an intuitive approach for robust training is to maximize the minimum score value obtained by all possible perturbations in an $\ell_\infty$ ball with radius $\epsilon$.

$$j^*, \eta^* = \arg \max_{j, \eta} RS(j, \eta, I),$$

where $RS(\cdot)$ is a robust score function defined as

$$RS(j, \eta, I) := \min_{I'} S(j, \eta, I')$$

s.t. $x_i' \in B_\epsilon^\infty(x_i)$, for all $x_i' \in I'$.

In other words, each $x_i \in I$ can be perturbed individually under an $\ell_\infty$ norm bounded perturbation to form a new set of training examples $I'$. We consider the worst case perturbation, such that the set $I'$ triggers the worst case score after split with feature $j$ and threshold $\eta$. The training objective (2) becomes a max-min optimization problem.

Note that there is an intrinsic consistency between boundaries of the $\ell_\infty$ balls and the decision boundary of a decision tree. For the split on the $j$-th feature, perturbations along features other than $j$ do not affect the split. So we only need to consider perturbations within $\pm \epsilon$ along the $j$-th feature. We define $\Delta I$ as the ambiguity set, containing examples with feature $j$ inside the $[|\eta - \epsilon, \eta + \epsilon|]$ region (see Table 1).

Only examples in $\Delta I$ may be perturbed from $I_L$ to $I_R$ or from $I_R$ to $I_L$ to reduce the score. Perturbing points in $I \setminus \Delta I$ will not change the score or the leaves they are assigned to. We denote $I_L^{\epsilon}$ and $I_R^{\epsilon}$ as the set of examples that are certainly on the left and right child leaves under perturbations (see Table 1 for definitions). Then we introduce 0-1 variables $s_i = \{0, 1\}$ denoting an example in the ambiguity set $\Delta I$ to be assigned to $I_L$ and $I_R$, respectively. Then the $RS$ can be formulated as a 0-1 integer optimization problem with $|\Delta I|$ variables, which is NP-hard in general. Additionally, we need to scan through all $d$ features of all examples and solve $O(|\Delta I|d)$ minimization problems for a single split at a single node. This large number of problems to solve makes this computation intractable. Therefore, we need to find an approximation for the $RS(j, \eta, I)$. In Sections 4.3 and 4.4, we present two different approximations and corresponding implementations of our robust decision tree framework, first for classical decision trees with information gain score, and then for modern tree boosting systems which can minimize any loss function.

It is worth mentioning that we normalize features to $[0, 1]^d$ for the sake of simplicity in this paper. One can also define $\epsilon_1, \epsilon_2, \ldots, \epsilon_d$ for each feature and then the adversary is allowed to perturb $x_i$ within $[x_i^{(1)} - \epsilon_1, x_i^{(1)} + \epsilon_1] \times \cdots \times [x_i^{(d)} - \epsilon_d, x_i^{(d)} + \epsilon_d]$. In this case, we would not need to normalize the features. Also, $\epsilon$ is a hyper-parameter in our robust model. Models trained with larger $\epsilon$ are expected to be more robust and when $\epsilon = 0$, the robust model is the same as a natural model.

### 4.3. Robust Splitting for Decision Trees with Information Gain Score

Here we consider a decision tree for binary classification, $y_i \in \{0, 1\}$, with information gain as the metric for node splitting. The information gain score is

$$S(j, \eta, I) := IG(j, \eta) = H(y) - H(y|x^{(j)} < \eta),$$

where $H(\cdot)$ and $H(\cdot|\cdot)$ are entropy and conditional entropy on the empirical distribution. For simplicity, we denote $N_0 := |I_0|$, $N_1 := |I_1|$, $n_0 := |I_L \cap I_0|$ and $n_1 := |I_L \cap I_1|$. The following theorem shows adversary’s perturbation direction to minimize the information gain.

**Theorem 1.** If $\frac{n_0}{N_0} < \frac{n_1}{N_1}$ and $\frac{n_0 + 1}{N_0} \leq \frac{n_1}{N_1}$, perturbing one example in $\Delta I_R$ with label 0 to $I_L$ will decrease the information gain.

Similarly, if $\frac{n_0}{N_0} < \frac{n_1}{N_1}$ and $\frac{n_0 + 1}{N_0} \leq \frac{n_1}{N_1}$, perturbing one example in $\Delta I_L$ with label 1 to $I_L$ will decrease the information gain. The proof of this theorem will be presented in Section A in the appendix. Note that we also have a similar conclusion for Gini impurity score, which will be shown in Section B in the appendix. Therefore, to decrease the information gain score, the adversary needs to perturb examples in $\Delta I$ such that $\frac{n_0}{N_0}$ and $\frac{n_1}{N_1}$ are close to each other (the ideal case $\frac{n_0}{N_0} = \frac{n_1}{N_1}$ may not be achieved because $n_0, n_1, N_0$ and $N_1$ are integers). The robust split finding algorithm is shown in Algorithm 1. In this algorithm we find a perturbation that minimizes $\left| \frac{n_0}{N_0} - \frac{n_1}{N_1} \right|$ as an approximation and upper bound to the optimal solution. Algorithm 3 in Section A in the appendix shows an $O(|I|)$ procedure to find such perturbation to approximately minimize the information gain. Since the algorithm scans through $\{x_i^{(1)}, \ldots, x_i^{(d)}\}$ in the sorted order, the sets $\Delta I$, $I_L^{\epsilon}$, $I_R^{\epsilon}$ can be maintained in amortized $O(1)$ time in the inner loop. Therefore, the computational complexity of the robust training algorithm is $O(d|I|^2)$ per split.

Although it is possible to extend our conclusion to other traditional scores of classification trees, we will focus on the modern scenario where we use a regression tree to fit traditional scores of classification trees, respectively. In this case, we would not need to normalize the features. Also, $\epsilon$ is a hyper-parameter in our robust model. Models trained with larger $\epsilon$ are expected to be more robust and when $\epsilon = 0$, the robust model is the same as a natural model.

### 4.4. Robust Splitting for GBDT models

We now introduce the regression tree training process used in many modern tree boosting packages including XGBoost (Chen & Guestrin, 2016), LightGBM (Ke et al., 2017) and CatBoost (Dorogush et al., 2018). Specifically, we focus on the formulation of gradient boosted decision tree (GBDT), which is one of the most successful ensemble
where \( \Omega(\cdot) \) we desire to minimize the following objective function.

Algorithm 1 Robust Split with Information Gain

**Input:** Training set \( \{(x_i, y_i)\}_{i=1}^N, x_i \in [0, 1]^d, y_i \in \{0, 1\} \).

**Input:** The instance set of the current node \( I \).

**Input:** \( \epsilon \), the radius of the \( \ell_\infty \) ball.

**Output:** Optimal split of the current node \( I \).

\[
I_0 \leftarrow \{(x_i, y_i) | y_i = 0\}, I_1 \leftarrow \{(x_i, y_i) | y_i = 1\}; \\
N_0 \leftarrow |I \cap I_0|, N_1 \leftarrow |I \cap I_1|.
\]

for \( j \leftarrow 1 \) to \( d \) do

for \( m \) in sorted(\( I \), ascending order by \( x^2_{m,j} \)) do

\[ \eta \leftarrow \frac{1}{2}(x_{j}^{d_i} + x_{j}^{d_m+1}), \Delta I \leftarrow I \cap \{(x_i, y_i) | y_i \leq \eta \} \]

\[ I^\circ_L \leftarrow \{(x_i, y_i) | x^{(j)} < \eta \}, I^\circ_R \leftarrow \{(x_i, y_i) | x^{(j)} > \eta \} \]

\[ n^\circ_L \leftarrow |I^\circ_L \cap I_0|, n^\circ_R \leftarrow |I^\circ_R \cap I_1|; \]

Find \( \Delta n^\circ_L, \Delta n^\circ_R \) to minimize \( |\Delta n^\circ_L + n^\circ_R - \Delta n^\circ_R + n^\circ_L| \)

using Algorithm 3 in Section A in the appendix.

From \( \Delta I \), add \( \Delta n^\circ_R \) points with \( y = 0 \) and \( \Delta n^\circ_L \) points with \( y = 1 \) to \( I^\circ_L \) and obtain \( I_L \).

Add remaining points in \( \Delta I \) to \( I^\circ_R \) and obtain \( I_R \).

\[ RS(j, \eta) \leftarrow IG(I_L, I_R); \]

end for

end for

\[ j^*, \eta^* \leftarrow \arg \max_j, \eta RS(j, \eta); \]

Split on feature \( j^* \) with a threshold \( \eta^* \);

end for

Robust Decision Trees Against Adversarial Examples

Algorithm 2 Robust Split for Boosted Tree

**Input:** training set \( \{(x_i, y_i)\}_{i=1}^N, x_i \in [0, 1]^d, y_i \in \mathbb{R} \).

**Input:** \( \epsilon \), the radius of the \( \ell_\infty \) ball.

**Output:** Optimal split of the current node \( I \).

for \( j \leftarrow 1 \) to \( d \) do

for \( m \) in sorted(\( I \), ascending order by \( x^2_{m,j} \)) do

\[ \eta \leftarrow \frac{1}{2}(x_{j}^{d_i} + x_{j}^{d_m+1}); \]

\[ I^\circ_L \leftarrow \{(x_i, y_i) | x^{(j)} < \eta \}, I^\circ_R \leftarrow \{(x_i, y_i) | x^{(j)} \geq \eta \}; \]

\[ S_I = S(I_L, I_R), S_2 = S(I^\circ_L, I^\circ_R \cup I_R), S_3 = S(I^\circ_L \cup I_R, I^\circ_R \cup \Delta I); \]

\[ RS(j, \eta) \leftarrow \min\{S_1, S_2, S_3, S_4\}; \]

end for

end for

impact of \( f_K(x_i) \) on \( l(y_i, \hat{y}_i) \) can be approximated using a second order Taylor expansion:

\[ l(y_i, \phi_K(x_i)) \approx l(y_i, \phi_K(x_i)) + g_i f_K(x_i) + \frac{1}{2} h_i (f_K(x_i))^2 \]

where \( g_i = \frac{\partial l(y_i, \phi_K(x_i))}{\partial f_K(x_i)} \) and \( h_i = \frac{\partial^2 l(y_i, \phi_K(x_i))}{\partial f_K(x_i)^2} \) are the first and second order derivatives on the loss function with respect to the prediction of decision tree \( f_K \) on point \( x_i \).

Conceptually, ignoring the regularization terms, the score function can be given as:

\[ S(I_L, I_R) = \sum_{i \in I_L} l(y_i, \phi_K(x_i)) | \phi_K(x_i) = \omega_L \]

\[ + \sum_{i \in I_R} l(y_i, \phi_K(x_i)) | \phi_K(x_i) = \omega_R \]

where \( \omega_L, \omega_R \) are the prediction values of the left, right and parent nodes. The score represents the improvements on reducing the loss function \( L \) for all data examples in \( I \). The exact form of score used in XGBoost with regularization terms is given in (Chen & Guestrin, 2016):

\[ S(j, \eta, I) = S(I_L, I_R) \]

\[ := \frac{1}{2} \left[ \sum_{i \in I_L} g_i^2 + \sum_{i \in I_R} g_i^2 + \sum_{i \in I_L} h_i + \lambda + \sum_{i \in I_R} h_i + \lambda - (\sum_{i \in I} g_i)^2 \right] - \gamma, \]

where \( \gamma \) is a regularization constant. Again, to minimize the score by perturbing points in \( \Delta I \), the adversary needs to solve an intractable 0-1 integer optimization at each possible splitting position. Since GBDT is often deployed in large-scale data mining tasks with a large amount of training data to scan through at each node, and we need to solve \( RS \)
Robust Decision Trees Against Adversarial Examples

5. Experiments

Our code is at https://github.com/chenhongge/RobustTrees.

5.1. Robust Information Gain Decision Trees

We present results on three small datasets with robust information gain based decision trees using Algorithm 1. We focus on untargeted adversarial attacks. For each dataset we test on 100 examples (or the whole test set), and we only attack correctly classified images. Attacks proceed until the attack success rate is 100%; the differences in robustness are reflected in the distortion of the adversarial examples required to achieve a successful attack. In Table 2, we present the average $\ell_\infty$ distortion of the adversarial examples of both classical natural decision trees and our robust decision trees trained on different datasets. We use Papernot’s attack as well as $\ell_\infty$ versions of Cheng’s and Kantchelian’s attacks. The $\ell_1$ and $\ell_2$ distortion found by Kantchelian’s $\ell_1$ and $\ell_2$ attacks are presented in Table 4 in the appendix. The adversarial examples found by Cheng’s, Papernot’s and Kantchelian’s attacks have much larger $\ell_\infty$ norm for our robust trees compared to those for the natural trees, demonstrating that our robust training algorithm improves the decision tree robustness substantially. In some cases our robust decision trees also have higher test accuracy than the natural trees. This may be due to the fact that the robust score tends to encourage the tree to split at thresholds where fewer examples are in the ambiguity set, and thus the split is also robust against random noise in the training set. Another possible reason is the implicit regularization in the robust splitting. The robust score is always lower than the regular score and thus our splitting is more conservative. Also, from results in Table 2 we see that most of the adversarial examples found by Papernot’s attack have larger $\ell_\infty$ norm than those found by Cheng’s $\ell_\infty$ attack. This suggests that the straight-forward greedy search attack is not as good as a sophisticated general attack for attacking decision trees. Cheng’s attack is able to achieve similar $\ell_\infty$ distortion as Kantchelian’s attack, without solving expensive MILPs. While not scalable to large datasets, Kantchelian’s attack can find the minimum adversarial examples, reflecting the true robustness of a tree-based model.

5.2. Robust GBDT Models

In this subsection, we evaluate our algorithm in the tree boosting setting, where multiple robust decision trees are created in an ensemble to improve model accuracy. We implement Algorithm 2 by slightly modifying the node splitting procedure in XGBoost. Our modification is only relevant to computing the scores for selecting the best split, and is compatible with other existing features of XGBoost. We also use XGBoost to train natural (undefended) GBDT models. Again, we focus on untargeted adversarial attacks. We consider nine real world large or medium sized datasets and two small datasets (Chang & Lin, 2011), spanning a variety of data types (including both tabular and image data). For small datasets we use 100 examples and for large or medium sized datasets, we use 5000 examples for robustness evaluation, except for MNIST 2 vs. 6, where we use 100 examples. MNIST 2 vs. 6 is a subset of MNIST to only distinguish between 2 and 6. This is the dataset tested in Kantchelian et al. (2016). We use the same number of trees, depth and step size shrinkage as in Kantchelian et al. (2016) to train our robust and natural models. Same as Kantchelian et al. (2016), we only test 100 examples for MNIST 2 vs. 6 since the model is relatively large. In Table 3, we present the average $\ell_\infty$ distortion of adversarial examples found by Cheng’s $\ell_\infty$ attack for both natural GBDT and robust GBDT models trained on those datasets. For small and medium binary classification models, we also present results of Kantchelian’s $\ell_\infty$ attack, which finds the minimum adversarial example in $\ell_\infty$ norm. The $\ell_1$ and $\ell_2$ distortion found by Kantchelian’s $\ell_1$ and $\ell_2$ attacks are presented in Table 5 in the appendix. Kantchelian’s attack can only handle binary classification problems and small scale models due to its time-consuming MILP formulation. Papernot’s attack is inapplicable here because it is for attacking a single tree only. The natural and robust models have the same number of trees for comparison. We only attack correctly classified images and all examples are successfully attacked. We see that our robust GBDT models consistently outperform the natural GBDT models in terms of $\ell_\infty$ robustness.

For some datasets, we need to increase tree depth in robust GBDT models in order to obtain accuracy comparable to the natural GBDT models. The requirement of larger model capacity is common in the adversarial training literature: in the state-of-the-art defense for DNNs, Madry et al. (2018) argues that increasing the model capacity is essential for adversarial training to obtain good accuracy.

Figure 3 and Figure 5 in the appendix show the distortion.
and accuracy of MNIST and Fashion-MNIST models with different number of trees. The adversarial examples are found by Cheng’s $\ell_\infty$ attack. Models with $k$ trees are the first $k$ trees during a single boosting run of $K$ ($K \geq k$) trees. The $\ell_\infty$ distortion of robust models are consistently much larger than those of the natural models. For MNIST dataset, our robust GBDT model loses accuracy slightly when the model has only 20 trees. This loss is gradually compensated as more trees are added to the model; regardless of the number of trees in the model, the robustness improvement is consistently observed, as our robust training is embedded in each tree’s building process and we create robust trees beginning from the very first step of boosting. Adversarial training in Kantchelian et al. (2016), in contrast, adds adversarial examples with respect to the current model at each boosting round so adversarial examples produced in the later stages of boosting are only learned by part of the model. The non-robust trees in the first few rounds of boosting still exist in the final model and they may be the weakness of the ensemble. Similar problems are not present in DNN adversarial training since the whole model is exposed to new adversarial examples throughout the training process. This may explain why adversarial training in Kantchelian et al. (2016) failed to improve $\ell_1, \ell_2$, or $\ell_\infty$ robustness on the MNIST 2 vs. 6 model, while our method achieves significant robustness improvement with the same training parameters and evaluation metrics, as shown in Tables 3 and 5. Additionally, we also evaluate the robustness of natural and robust models with different number of trees on a variety of datasets using Cheng’s $\ell_\infty$ attack, presented in Table 7 in the appendix.

Figure 3. (Best viewed in color) $\ell_\infty$ distortion vs. classification accuracy of GBDT models on MNIST dataset with different numbers of trees (circle size). The adversarial examples are found by Cheng’s $\ell_\infty$ attack. The robust training parameter $\epsilon = 0.3$ for MNIST. With robust training (purple) the distortion needed to fool a model increases dramatically with less than 1% accuracy loss.

We also test our framework on random forest models and the results are shown in Section G in the appendix.

6. Conclusion

In this paper, we study the robustness of tree-based machine learning models under adversarial attacks. Our experiments show that just as in DNNs, tree-based models are also vulnerable to adversarial attacks. To address this issue, we propose a novel robust decision tree training framework. We make necessary approximations to ensure scalability and implement our framework in both classical decision tree and tree boosting settings. Extensive experiments on a variety of datasets show that our method substantially improves model robustness. Our framework can be extended to other tree-based models such as Gini impurity based classification trees, random forest, and CART.
Acknowledgements

The authors thank Aleksander Madry for fruitful discussions. The authors also acknowledge the support of NSF via IIS-1719097, Intel, Google Cloud, Nvidia, SenseTime and IBM.

References


Robust Decision Trees Against Adversarial Examples


