Robust Decision Trees Against Adversarial Examples

Hongge Chen\textsuperscript{1}, Huan Zhang\textsuperscript{2}, Duane Boning\textsuperscript{1}, Cho-Jui Hsieh\textsuperscript{2}
\textsuperscript{1}MIT, Cambridge, MA 02139
\textsuperscript{2}UCLA, Los Angeles, CA 90095
chenhg@mit.edu, huan@huan-zhang.com, boning@mtl.mit.edu, chohsieh@cs.ucla.edu

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 \cite{szegedy2013intriguing, kos2018deep, cheng2018fooling, chen2018targeted, carlini2018audio, huang2017adversarial} 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 \cite{papernot2016transferability}.

In our paper, we shed light on the adversarial robustness of another class of important 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 \cite{zhou2017interpretable}. In this paper, we study the robustness of tree-based models under adversarial examples, and more importantly, we propose a novel robust training framework for tree-based models. Below we highlight our major contributions:

\begin{itemize}
    \item 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. These well-crafted examples are very close to natural examples often with changes imperceptible to humans, but that can easily fool existing tree-based models at the testing or deployment stages.
    \item 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.
    \item 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. 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 the fact 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 heuristic methods such as 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. Random forests, for example, are bootstrap aggregating tree ensembles. 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 which utilizes 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). Cost efficient tree boosting approaches that consider test-time budget during training have also been proposed recently by Peter et al. (2017) and Xu et al. (2019).

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 \( x_{adv} \),

\[
x_{adv} \leftarrow \text{clip}[x_0 + \xi \text{sgn}(\nabla J(x_0))],
\]

where \( \xi \) is the step size, \( \text{sgn}(\nabla J(x_0)) \) is the sign of the gradient of the training loss with respect to \( x_0 \), and \( \text{clip}(\cdot) \) ensures that \( x_{adv} \) stays in the valid input range. This method is strengthened as Iterative-FGSM (or I-FGSM) (Kurakin et al., 2017), which applies FGSM multiple times for a higher success rate and smaller distortion. C&W attack (Carlini & Wagner, 2017) formulates the attack as an optimization problem

\[
\min_x \lambda f(x) + \|x - x_0\|_2^2, \quad \text{s.t. } x \in [0, 1]^d,
\]

where \( f(x) \) is a loss function to encourage the model to output an incorrect label. 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 state-of-the-art robustness performances on MNIST and CIFAR. 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. Therefore, 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 as 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. \cite{Kantchelian2016} 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 trained model.

In our paper, in addition to Kantchelian attacks we also use a general attack method proposed in \cite{Cheng2019} 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, \cite{Cheng2019} 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 \cite{Kantchelian2016}, 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.

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 possible perturbations in \( \ell_\infty \) balls. At a high level, the robust splitting feature and threshold take the distances between data points into account (which is often ignored in most decision tree learning algorithms) and tries to optimize the worst case performance under adversarial perturbations.

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 \( D = \{(x_i, y_i)\} \) \((1 \leq i \leq N, y_i \in \mathbb{R}, x_i = [x_i^{(1)}, x_i^{(2)}, \ldots, x_i^{(j)}, \ldots, x_i^{(d)}] \in \mathbb{R}^d\)\), we first normalize the feature values to \([0, 1]\) such that \( x_i \in [0, 1]^d \) (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 \( I \subseteq D \) as the set of points at that node. For a split on the \( j \)-th
Figure 2: (Best viewed in color) A simple example illustrating how robust splitting works. Upper: A set of 10 points that can be easily separated with a horizontal split on feature $x^{(2)}$. The accuracy of this split is 0.8. Middle: The high accuracy horizontal split cannot separate the $\ell_\infty$ balls around the data points and thus an adversary can perturb any example $x_i$ within the indicated $\ell_\infty$ ball to mislead the model. The worst case accuracy under adversarial perturbations is 0 if all points are perturbed within the square boxes ($\ell_\infty$ norm bounded noise). Lower: a more robust split would be a split on feature $x^{(1)}$. The accuracy of this split is 0.7 under all possible perturbations within the same size $\ell_\infty$ norm bounded noise (square boxes).

feature with a threshold $\eta$, the sets which will be mentioned in Sections 4.2, 4.3 and 4.4 are summarized in Table 1.

| Notation | Definition |
|----------|------------|
| $\mathcal{I}$ | set of examples on the current node |
| $\mathcal{I}_0$ | $\mathcal{I} \cap \{(x_i, y_i) | y_i = 0\}$ (for classification) |
| $\mathcal{I}_1$ | $\mathcal{I} \cap \{(x_i, y_i) | y_i = 1\}$ (for classification) |
| $\mathcal{I}_L$ | $\mathcal{I} \cap \{(x_i, y_i) | x^{(j)} < \eta\}$ |
| $\mathcal{I}_R$ | $\mathcal{I} \cap \{(x_i, y_i) | x^{(j)} \geq \eta\}$ |
| $\Delta \mathcal{I}_L$ | $\mathcal{I} \cap \{(x_i, y_i) | \eta - \epsilon \leq x^{(j)} \leq \eta + \epsilon\}$ |
| $\Delta \mathcal{I}_R$ | $\mathcal{I} \cap \mathcal{I}_L \cap \mathcal{I}_R$ |
| $\mathcal{I}_L'$ | $\mathcal{I}_L \setminus \Delta \mathcal{I}$ |
| $\mathcal{I}_R'$ | $\mathcal{I}_R \setminus \Delta \mathcal{I}$ |

Table 1: Notations of different sets in Section 4. We assume a split is made on the $j$-th feature with a threshold $\eta$, and this feature can be perturbed by $\pm \epsilon$.

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: a function of the splits on left and right child nodes ($\mathcal{I}_L$ and $\mathcal{I}_R$), or equivalently the chosen feature $j$ to split and a corresponding threshold value $\eta$.

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:

$$j^*, \eta^* = \arg \max_{j, \eta} \text{score}(\mathcal{I}_L, \mathcal{I}_R) = \arg \max_{j, \eta} \text{score}(j, \eta, \mathcal{I})$$ (1)

In our setting, we consider the case where features of examples in $\mathcal{I}_L$ and $\mathcal{I}_R$ 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 a $\ell_\infty$ ball of radius $\epsilon$ around each example $x_i$:

$$B^\infty_\epsilon(x_i) := [x_i^{(1)} - \epsilon, x_i^{(1)} + \epsilon] \times \cdots \times [x_i^{(d)} - \epsilon, x_i^{(d)} + \epsilon].$$
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 metric value obtained by all possible perturbations in an $\ell_\infty$ ball with radius $\epsilon$,

$$j^*, \eta^* = \arg\max_{j, \eta} \text{robust_score}(j, \eta, \mathcal{I}), \tag{2}$$

where the robust_score is defined as

$$\text{robust_score}(j, \eta, \mathcal{I}) := \min_{\mathcal{I}' \supseteq \{ \mathbf{x}_i \}} \text{score}(j, \eta, \mathcal{I}')$$

s.t. $\mathbf{x}_i' \in B_\infty^\epsilon(\mathbf{x}_i)$, for all $\mathbf{x}_i' \in \mathcal{I}'$.

In other words, each $\mathbf{x}_i \in \mathcal{I}$ can be perturbed individually under a $\ell_\infty$ norm bounded perturbation to form a new set of training examples $\mathcal{I}'$. We consider the worst case perturbation, such that the set $\mathcal{I}'$ 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. We only need to consider perturbations within $\pm \epsilon$ along the $j$-th feature. We define $\Delta \mathcal{I}$ as the ambiguity set, containing examples with feature $j$ inside the $[\eta - \epsilon, \eta + \epsilon]$ region (see Table 1). Only examples in $\Delta \mathcal{I}$ may be perturbed from $\mathcal{I}_L$ to $\mathcal{I}_R$ or from $\mathcal{I}_R$ to $\mathcal{I}_L$ to reduce the score. Perturbing points in $\mathcal{I} \setminus \Delta \mathcal{I}$ will not change the score or the leaves they are assigned to. We denote $\mathcal{I}_L'$ and $\mathcal{I}_R'$ 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 $\Delta \mathcal{I}$ to be assigned to $\mathcal{I}_L$ and $\mathcal{I}_R$, respectively. The robust_score can be formulated as a 0-1 integer optimization problem with $|\Delta \mathcal{I}|$ variables,

$$\text{robust_score}(j, \eta, \mathcal{I}) = \min_{s_i \in \{0,1\}} \text{score}(\mathcal{I}_L', \mathcal{I}_R')$$

s.t. $\mathcal{I}_L' = \mathcal{I}_L \cup \{ (\mathbf{x}_i, y_i) \in \Delta \mathcal{I} | s_i = 0 \}$,

and $\mathcal{I}_R' = \mathcal{I}_R \cup \{ (\mathbf{x}_i, y_i) \in \Delta \mathcal{I} | s_i = 1 \}$.

0-1 integer optimization problems are NP-hard in general. Additionally, we need to scan through all $d$ features of all examples and solve $O(|\mathcal{I}|d)$ 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 robust_score($j$, $\eta$, $\mathcal{I}$). 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 $\epsilon_1$, $\epsilon_2$, ..., $\epsilon_d$ for each feature and then the adversary is allowed to perturb $\mathbf{x}_i$ 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

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

where $H(y)$ and $H(y|x^{(j)} < \eta)$ are given by

$$H(y) = -\frac{|\mathcal{I}_0|}{|\mathcal{I}|} \log\left(\frac{|\mathcal{I}_0|}{|\mathcal{I}|}\right) - \frac{|\mathcal{I}_1|}{|\mathcal{I}|} \log\left(\frac{|\mathcal{I}_1|}{|\mathcal{I}|}\right),$$

$$H(y|x^{(j)} < \eta) = -\frac{|\mathcal{I}_0|}{|\mathcal{I}|} \log\left(\frac{|\mathcal{I}_0|}{|\mathcal{I}|}\right) - \frac{|\mathcal{I}_1|}{|\mathcal{I}|} \log\left(\frac{|\mathcal{I}_1|}{|\mathcal{I}|}\right),$$

and $\mathcal{I}_0$ and $\mathcal{I}_1$ are the sets of examples in $\mathcal{I}$ that satisfy $x^{(j)} \leq x^{(j)}$ and $x^{(j)} > x^{(j)}$ respectively.
\[ H(y|x^{(j)} < \eta) = -\frac{|I_L|}{|I|} \left[ \frac{|I_L \cap I_0|}{|I_L|} \log\left(\frac{|I_L \cap I_0|}{|I_L|}\right) + \frac{|I_L \cap I_1|}{|I_L|} \log\left(\frac{|I_L \cap I_1|}{|I_L|}\right) \right] \\
- \frac{|I_R|}{|I|} \left[ \frac{|I_R \cap I_0|}{|I_R|} \log\left(\frac{|I_R \cap I_0|}{|I_R|}\right) + \frac{|I_R \cap I_1|}{|I_R|} \log\left(\frac{|I_R \cap I_1|}{|I_R|}\right) \right]. \]

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 4.1.** 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.

**Proof.** The information gain of this split can be written as a function of \( n_0 \) and \( n_1 \):

\[
IG = C \cdot n_0 \log\left(\frac{n_0}{N_0(n_1 + n_0)}\right) + n_1 \log\left(\frac{n_1}{N_1(n_1 + n_0)}\right) + (N_0 - n_0) \log\left(\frac{N_0 - n_0}{N_0(N_1 + N_0 - n_1 - n_0)}\right) + (N_1 - n_1) \log\left(\frac{N_1 - n_1}{N_1(N_1 + N_0 - n_1 - n_0)}\right) + C_1,
\]

where \( C > 0 \) and \( C_1 \) are constants with respect to \( n_0 \). Taking \( n_0 \) as a continuous variable, we have

\[
\frac{\partial IG}{\partial n_0} = C \cdot \log\left(\frac{n_0(N_1 + N_0 - n_1 - n_0)}{(N_0 - n_0)(n_1 + n_0)}\right) = C \cdot \log\left(1 + \frac{n_0 N_1 - N_0 n_1}{(N_0 - n_0)(n_1 + n_0)}\right) \tag{6}
\]

When \( \frac{\partial IG}{\partial n_0} < 0 \), perturbing one example in \( \Delta I_R \) with label 0 to \( I_L \) will increase \( n_0 \) and decrease the information gain. It is easy to see that \( \frac{\partial IG}{\partial n_0} < 0 \) if and only if \( \frac{n_0}{N_0} < \frac{n_1}{N_1} \). This indicates that when \( \frac{n_0}{N_0} < \frac{n_1}{N_1} \) and \( \frac{n_0 + 1}{N_0} \leq \frac{n_1}{N_1} \), perturbing one example with label 0 to \( I_L \) will always decrease the information gain. \( \square \)

Similarly we have \( \frac{\partial IG}{\partial n_1} < 0 \) if and only if \( \frac{n_0}{N_0} > \frac{n_1}{N_1} \) (similarly, \( n_1 \) is taken as continuous). 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 \( \frac{n_0}{N_0} - \frac{n_1}{N_1} \) as an approximation and upper bound to the optimal solution. Algorithm 2 shows an \( O(N) \) procedure to find such perturbation to approximately minimize the information gain. Since the algorithm scans through \( \{x_1^{(j)}, \ldots, x_d^{(j)}\} \) in the sorted order, the sets \( \Delta I, I^*_L, I^*_R \) can be maintained in amortized \( O(1) \) time in the inner loop. Therefore, the computational complexity of the robust training algorithm is \( O(dN^2) \).

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 any loss function in the next sub-section.

### 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 models and has been widely used in industry. GBDT is an additive tree ensemble model \( \phi(\cdot) \) combining outputs of \( K \) trees

\[
\hat{y}_i = \phi_K(x_i) = \sum_{k=1}^K f_k(x_i)
\]

where each \( f_k \) is a decision tree and \( \hat{y}_i \) is the final output for \( x_i \). Here we only focus on regression trees where \( \hat{y}_i \in \mathbb{R} \). Note that even for a classification problem, the modern treatment in GBDT is to consider the data with logistic loss, and use a regression tree to minimize this loss.
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.

\( I_0 \leftarrow \{(x_i, y_i)|y_i = 0\}, I_1 \leftarrow \{(x_i, y_i)|y_i = 1\}; \)

\( N_0 \leftarrow |I \cap I_0|, N_1 \leftarrow |I \cap I_1|; \)

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

\( \eta \leftarrow \frac{1}{2}(x_{m_0}^j + x_{m+1}^j), \Delta I \leftarrow I \cap \{(x_i, y_i)|\eta - \epsilon \leq x_i \leq \eta + \epsilon\}; \)

\( I_{p_0}^j \leftarrow \{(x_i, y_i)|x_i < \eta - \epsilon\}, I_{p_1}^j \leftarrow \{(x_i, y_i)|x_i > \eta + \epsilon\}; \)

\( n_{0j}^\star \leftarrow |I_{p_0}^j \cap I_0|, n_{1j}^\star \leftarrow |I_{p_1}^j \cap I_1|; \)

Find \( \Delta n_{0j}^\star, \Delta n_{1j}^\star \) to minimize \( |\frac{\Delta n_{0j}^\star + n_{0j}^\star}{N_0} - \frac{\Delta n_{1j}^\star + n_{1j}^\star}{N_1}| \) using Algorithm 2.

From \( \Delta I \), add \( \Delta n_{0j}^\star \) points with \( y = 0 \) and \( \Delta n_{1j}^\star \) points with \( y = 1 \) to \( I_0^j \) and obtain \( I_L^j \);

Add remaining points in \( \Delta I \) to \( I_R^j \) and obtain \( I_R^j \);

\( \text{robust-score}(j, \eta) \leftarrow IG(I_L^j, I_R^j); \)

\( j^\star, \eta^\star \leftarrow \arg \max_{j, \eta} \text{robust-score}(j, \eta); \)

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

Algorithm 2: Minimizing the Information Gain

Input: \( N_0 \) and \( N_1 \), number of instances with label 0 and 1.

Input: \( n_{0j}^\star \) and \( n_{1j}^\star \), number of instances with label 0 and 1 that are certainly on the left.

Input: \( |\Delta I \cap I_0| \) and \( |\Delta I \cap I_1| \), number of instances with label 0 and 1 that can be perturbed.

Output: \( \Delta n_{0j}^\star, \Delta n_{1j}^\star \), optimal number of points with label 0 and 1 in \( \Delta I \) to be placed on the left.

\( \Delta n_{0j}^\star \leftarrow 0, \Delta n_{1j}^\star \leftarrow 0, \min \_\text{diff} \leftarrow |\frac{n_{0j}^\star}{N_0} - \frac{n_{1j}^\star}{N_1}|; \)

for \( \Delta n_{0j} \leftarrow 0 \) to \( |\Delta I \cap I_0| \) do

\( \text{ceil} \leftarrow \left\lceil \frac{N_j(n_{0j}^\star + \Delta n_{0j})}{N_0} - n_{1j}^\star \right\rceil; \)

\( \text{floor} \leftarrow \left\lfloor \frac{N_j(n_{0j}^\star + \Delta n_{0j})}{N_0} - n_{1j}^\star \right\rfloor; \)

for \( \Delta n_{1j} \in \{\text{ceil}, \text{floor}\} \) do

\( \Delta n_{1j} \leftarrow \max\{\min\{\Delta n_{1j}^\star, |\Delta I \cap I_1|\}, 0\}; \)

if \( \min \_\text{diff} > |\frac{\Delta n_{0j} + n_{0j}^\star}{N_0} - \frac{\Delta n_{1j} + n_{1j}^\star}{N_1}| \) then

\( \Delta n_{0j}^\star \leftarrow \Delta n_{0j}, \Delta n_{1j}^\star \leftarrow \Delta n_{1j}, \min \_\text{diff} \leftarrow |\frac{\Delta n_{0j} + n_{0j}^\star}{N_0} - \frac{\Delta n_{1j} + n_{1j}^\star}{N_1}|; \)

Return \( \Delta n_{0j}^\star \) and \( \Delta n_{1j}^\star \);

During GBDT training, the trees \( f_k \) are added in an additive manner: when we consider the tree \( f_K \), all previous trees \( f_k, k \in \{1, \cdots, K-1\} \) are kept unchanged. For a general convex loss function \( l \) (such as MSE or logistic loss), we desire to minimize the following objective

\[
\mathcal{L}(\phi, D) = \sum_{i=1}^{N} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k) = \sum_{i=1}^{N} l(y_i, \phi_{K-1}(x_i)) + f_K(x_i),) + \sum_{k=1}^{K-1} \Omega(f_k) + \Omega(f_K)
\]

where \( \Omega(f) \) is a regularization term to penalize complex trees; for example, in XGBoost, \( \Omega(f) = \gamma T + \frac{1}{2} \lambda \|\omega\|_2^2 \), where \( T \) is the number of leaves, \( \omega \) is a vector of all leaf predictions and \( \lambda, \gamma \geq 0 \) are regularization constants. Importantly, when we consider \( f_K, \phi_{K-1} \) is a constant. The impact of \( f_K(x_i) \) on \( l(y_i, \hat{y}_i) \) can be approximated using a second order Taylor expansion:

\[
l(y_i, \phi_K(x_i)) \approx \hat{l}(y_i, \phi_K(x_i)) := l(y_i, \phi_{K-1}(x_i)) + g_i f_K(x_i) + \frac{1}{2} h_i f_K(x_i)^2
\]

where \( g_i = \frac{\partial l(y_i, \phi_K(x_i))}{\partial f_K(x_i)} \) and \( h_i = \frac{\partial^2 l(y_i, \phi_K(x_i))}{\partial f_K(x_i)^2} \) being 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
We focus on untargeted adversarial attacks. For each dataset we test on 100 examples, and we only attack
we present the average

Algorithm 3 Robust Split for Boosted Tree

Input: training set \( \{(x_i, y_i)\}^n_{i=1}, x_i \in \{0,1\}^d, y_i \in \mathbb{R} \).
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.

for \( j \leftarrow 1 \) to \( d \) do
for \( m \) in sorted(\( I \), ascending order by \( x^j_m \)) do
\( \eta \leftarrow \frac{1}{2}(x^j_m + x^j_{m+1}); \)
\( I^L_\eta \leftarrow \{(x_i, y_i)|x^j_i < \eta - \epsilon\}; \Delta I^L \leftarrow I \cap \{(x_i, y_i)|\eta - \epsilon \leq x^j_i < \eta\}; \)
\( I^R_\eta \leftarrow \{(x_i, y_i)|x^j_i > \eta + \epsilon\}; \Delta I^R \leftarrow I \cap \{(x_i, y_i)|\eta \leq x^j_i \leq \eta + \epsilon\}; \)
\( S_1 = S(I^L, I^R), S_2 = S(I^L, I^R \cup \Delta I^L), S_3 = S(I^L \cup \Delta I, I^R), S_4 = S(I^L \cup \Delta I^L, I^R \cup \Delta I^L); \)
\( \text{robust}\_\text{score}(j, \eta) \leftarrow \min\{\text{max}(S_1, S_2, S_3, S_4)\}; \)
Split on feature \( j^* \) with a threshold \( \eta^* \);

where \( \omega_L, \omega_R \) and \( \omega_P \) 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]:

\[
\text{score}(j, \eta, I) = S(I^L, I^R) := \frac{1}{2} \left[ \left( \sum_{i \in I^L} g_i \right)^2 \frac{1}{h_i + \lambda} + \left( \sum_{i \in I^R} g_i \right)^2 \frac{1}{h_i + \lambda} - \left( \sum_{i \in I} g_i \right)^2 \frac{1}{h_i + \lambda} \right] - \gamma, \tag{8}
\]

5 Experiments

5.1 Robust Information Gain Decision Trees

We present results on three small datasets with robust information gain based decision trees using Algorithm 3. We focus on untargeted adversarial attacks. For each dataset we test on 100 examples, 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 Cantchelian’s attacks. The \( \ell_1 \) and \( \ell_2 \) distortion found by Cantchelian’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 being not scalable to large datasets, Kantchelian’s attack can find the minimum adversarial examples, reflecting the true robustness of a tree-based model.

Figure 3 shows the decision boundaries and test accuracy of natural trees as well as robust trees with different $\epsilon$ values on two dimensional synthetic datasets. All trees have depth 5 and we plot training examples in the figure. The results show that the decision boundaries of our robust decision trees are simpler than the decision boundaries in natural decision trees, agreeing with the regularization argument above.

| Dataset | training set size | test set size | # of features | # of classes | depth | depth | test acc. | avg. $\ell_\infty$ dist. by Cheng’s $\ell_\infty$ attack | avg. $\ell_\infty$ dist. by Papernot’s attack | avg. $\ell_\infty$ dist. by Kantchelian’s $\ell_\infty$ attack |
|---------|------------------|---------------|---------------|-------------|-------|-------|-----------|---------------------------------|---------------------------------|---------------------------------|
| breast-cancer | 546              | 137           | 10            | 2           | 0.3   | 5     | 5         | 548                             | 542                             | 533                             |
| diabetes  | 614              | 154           | 8             | 2           | 0.2   | 5     | 5         | 688                             | 737                             | 206                             |
| ionosphere| 281              | 70            | 34            | 2           | 0.2   | 4     | 4         | 986                             | 929                             | 388                             |

Table 2: Test accuracy and robustness of information gain based single decision tree model. The robustness is evaluated by the average $\ell_\infty$ distortion of adversarial examples found by Cheng’s, Papernot’s and Kantchelian’s attacks. Average $\ell_\infty$ distortion of robust decision tree models found by three attack methods are consistently larger than that of the naturally trained ones.

### 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 3 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.

We next present experimental results to demonstrate the effectiveness of robust GBDT using Algorithm 3. Again, we focus on untargeted adversarial attacks. We consider nine real world large or medium sized datasets and two small datasets, spanning a variety of data types (not just images). 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)](https://jmlr.org/papers/v17/15-480.html). We use the same number of trees, depth and step size shrinkage as in [Kantchelian et al. (2016)](https://jmlr.org/papers/v17/15-480.html) to train our robust and natural models. Same as [Kantchelian et al. (2016)](https://jmlr.org/papers/v17/15-480.html), 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.

Table 3: GBDT model test accuracy and robustness. Average $\ell_\infty$ distortion of our robust GBDT models are consistently larger than those of the naturally trained models. The robustness is evaluated by the average $\ell_\infty$ distortion of adversarial examples found by Cheng’s and Kantchelian’s attacks. Only small or medium sized binary classification models can be evaluated by Kantchelian’s attack, but it finds the minimum adversarial example with smallest possible distortion.

Figure 3: (Best viewed in color) The decision boundaries and test accuracy of natural decision trees and robust decision trees with depth 5 on synthetic datasets with two features.
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 but our method achieves significant robustness improvement with the same training parameters and evaluation metrics, as shown in Tables 3 and 5.

Another interesting finding is that for MNIST and Fashion-MNIST datasets, models with more trees are generally more robust. This may not be true in other datasets; for example, results from Table 6 in the Appendix shows that on some other datasets, the natural GBDT models lose robustness when more trees are added.

One might hope that one can simply reduce the depth of trees to improve robustness since shallower trees provides stronger regularization effects. Unfortunately, this is not true. As demonstrated in Figure 5, the robustness of naturally trained GBDT models are much worse when compared to robust models, no matter how shallow they are or how many trees are in the ensemble. Also, when the number of trees in the ensemble model is limited, reducing tree depth will significantly lower the model accuracy.

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 6 in the Appendix.

![Figure 4: (Best viewed in color) $\ell_\infty$ distortion vs. classification accuracy of GBDT models on MNIST and Fashion-MNIST datasets with different numbers of trees. The adversarial examples are found by Cheng’s $\ell_\infty$ attack. The robust training parameter $\epsilon = 0.3$ for MNIST and $\epsilon = 0.1$ for Fashion-MNIST. With robust training the distortion needed to fool a model increases dramatically with less than 1% accuracy loss.](image)

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.
Figure 5: (Best viewed in color) Robustness vs. classification accuracy plot of GBDT models on MNIST dataset with different depth and different numbers of trees. The adversarial examples are found by Cheng’s $\ell_\infty$ attack. The robust training parameter $\epsilon = 0.3$. Reducing the model depth cannot improve robustness effectively compared to our proposed robust training procedure.
Acknowledgements

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

References

Athalye, A., Carlini, N., and Wagner, D. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In International Conference on Machine Learning, 2018.

Breiman, L. Classification and regression trees. Routledge, 1984.

Buckman, J., Roy, A., Raffel, C., and Goodfellow, I. Thermometer encoding: One hot way to resist adversarial examples. International Conference on Learning Representations, 2018.

Carlini, N. and Wagner, D. Towards evaluating the robustness of neural networks. In 2017 IEEE Symposium on Security and Privacy (SP), pp. 39–57. IEEE, 2017.

Carlini, N. and Wagner, D. Audio adversarial examples: Targeted attacks on speech-to-text. arXiv preprint arXiv:1801.01944, 2018.

Chen, H., Zhang, H., Chen, P.-Y., Yi, J., and Hsieh, C.-J. Attacking visual language grounding with adversarial examples: A case study on neural image captioning. In Annual Meeting of the Association for Computational Linguistics, 2018a.

Chen, P.-Y., Zhang, H., Sharma, Y., Yi, J., and Hsieh, C.-J. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security, pp. 15–26. ACM, 2017.

Chen, P.-Y., Sharma, Y., Zhang, H., Yi, J., and Hsieh, C.-J. Ead: elastic-net attacks to deep neural networks via adversarial examples. In Thirty-Second AAAI Conference on Artificial Intelligence, 2018b.

Chen, T. and Guestrin, C. XGBoost: A scalable tree boosting system. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 785–794. ACM, 2016.

Cheng, M., Yi, J., Zhang, H., Chen, P.-Y., and Hsieh, C.-J. Seq2sick: Evaluating the robustness of sequence-to-sequence models with adversarial examples. arXiv preprint arXiv:1803.01128, 2018.

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 International Conference on Learning Representations, 2019.

Dorogush, A. V., Ershov, V., and Gulin, A. Catboost: gradient boosting with categorical features support. arXiv preprint arXiv:1810.11363, 2018.

Feng, J., Yu, Y., and Zhou, Z.-H. Multi-layered gradient boosting decision trees. In Neural Information Processing Systems, 2018.

Friedman, J., Hastie, T., Tibshirani, R., et al. Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). The annals of statistics, 28(2):337–407, 2000.

Friedman, J. H. Greedy function approximation: a gradient boosting machine. Annals of statistics, pp. 1189–1232, 2001.

Friedman, J. H. Stochastic gradient boosting. Computational Statistics & Data Analysis, 38(4):367–378, 2002.

Goodfellow, I., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. In International Conference on Learning Representations, 2015.
Guo, C., Rana, M., Cisse, M., and van der Maaten, L. Countering adversarial images using input transformations. *arXiv preprint arXiv:1711.00117*, 2017.

Huang, S., Papernot, N., Goodfellow, I., Duan, Y., and Abbeel, P. Adversarial attacks on neural network policies. *arXiv preprint arXiv:1702.02284*, 2017.

Ilyas, A., Engstrom, L., Athalye, A., and Lin, J. Query-efficient black-box adversarial examples. *arXiv preprint arXiv:1712.07113*, 2017.

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

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., and Liu, T.-Y. LightGBM: A highly efficient gradient boosting decision tree. In *Neural Information Processing Systems*, pp. 3146–3154, 2017.

Kos, J., Fischer, I., and Song, D. Adversarial examples for generative models. In *2018 IEEE Security and Privacy Workshops (SPW)*, pp. 36–42. IEEE, 2018.

Kurakin, A., Goodfellow, I., and Bengio, S. Adversarial machine learning at scale. In *International Conference on Learning Representations*, 2017.

Ma, X., Li, B., Wang, Y., Erfani, S. M., Wijewickrema, S., Houle, M. E., Schoenebeck, G., Song, D., and Bailey, J. Characterizing adversarial subspaces using local intrinsic dimensionality. *arXiv preprint arXiv:1801.02613*, 2018.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. 2018.

Papernot, N., McDaniel, P., and Goodfellow, I. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*, 2016a.

Papernot, N., McDaniel, P., Wu, X., Jha, S., and Swami, A. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE Symposium on Security and Privacy (SP)*, pp. 582–597. IEEE, 2016b.

Peter, S., Diego, F., Hamprecht, F. A., and Nadler, B. Cost efficient gradient boosting. In *Neural Information Processing Systems*, 2017.

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

Raghunathan, A., Steinhardt, J., and Liang, P. Certified defenses against adversarial examples. *arXiv preprint arXiv:1801.09344*, 2018.

Samangouei, P., Kabkab, M., and Chellappa, R. Defense-GAN: Protecting classifiers against adversarial attacks using generative models. *arXiv preprint arXiv:1805.06605*, 2018.

Sinha, A., Namkoong, H., and Duchi, J. Certifying some distributional robustness with principled adversarial training. In *International Conference on Learning Representations*, 2018.

Song, Y., Kim, T., Nowozin, S., Ermon, S., and Kushman, N. Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. *arXiv preprint arXiv:1710.10766*, 2017.

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

Tyree, S., Weinberger, K. Q., Agrawal, K., and Paykin, J. Parallel boosted regression trees for web search ranking. In *Proceedings of the 20th international conference on World wide web*, pp. 387–396. ACM, 2011.

Wang, S., Chen, Y., Abdou, A., and Jana, S. Mixtrain: Scalable training of formally robust neural networks. *arXiv preprint arXiv:1811.02625*, 2018.
Wong, E. and Kolter, J. Z. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference on Machine Learning*, 2018.

Wong, E., Schmidt, F., Metzen, J. H., and Kolter, J. Z. Scaling provable adversarial defenses. In *Neural Information Processing Systems*, 2018.

Xu, W., Evans, D., and Qi, Y. Feature squeezing: Detecting adversarial examples in deep neural networks. *arXiv preprint arXiv:1704.01155*, 2017.

Xu, Z. E., Kusner, M. J., Weinberger, K. Q., and Zheng, A. X. Gradient regularized budgeted boosting. 2019.

Zhang, H., Si, S., and Hsieh, C.-J. GPU-acceleration for large-scale tree boosting. *SysML Conference*, 2018.

Zhou, Z.-H. and Feng, J. Deep forest: Towards an alternative to deep neural networks. In *International Joint Conferences on Artificial Intelligence*, 2017.
Appendix

6.1 Omitted Results on $\ell_1$ and $\ell_2$ distortion

In Tables 4 and 5 we present the $\ell_1$ and $\ell_2$ distortions of vanilla (information gain based) decision trees and GBDT models obtained by Kantchelian’s $\ell_1$ and $\ell_2$ attacks. Again, only small or medium sized binary classification models can be evaluated by Kantchelian’s attack. From the results we can see that although our robust decision tree training algorithm is designed for $\ell_\infty$ perturbations, it can also improve models’ $\ell_1$ and $\ell_2$ robustness significantly.

| Dataset      | # of features | # of classes | robust $\epsilon$ | depth | test acc. | avg. $\ell_1$ dist. by Kantchelian’s $\ell_1$ attack | avg. $\ell_2$ dist. by Kantchelian’s $\ell_2$ attack |
|--------------|---------------|--------------|-------------------|-------|-----------|---------------------------------------------------|---------------------------------------------------|
| breast-cancer| 10            | 2            | 0.3               | 5     | 5         | 948                                               | 942                                               |
| diabetes     | 8             | 2            | 0.2               | 5     | 5         | 688                                               | 747                                               |
| ionosphere   | 34            | 2            | 0.2               | 4     | 4         | 986                                               | 929                                               |

Table 4: The test accuracy and robustness of information gain based single decision tree models. The robustness is evaluated by the average $\ell_1$ and $\ell_2$ distortions of adversarial examples found by Kantchelian’s $\ell_1$ and $\ell_2$ attacks. Average $\ell_\infty$ distortions of robust decision tree models found by the two attack methods are consistently larger than those of the naturally trained ones.

| Dataset      | # of features | # of features | # of classes | # of trees | robust $\epsilon$ | avg. $\ell_1$ dist. by Kantchelian’s $\ell_1$ attack | avg. $\ell_2$ dist. by Kantchelian’s $\ell_2$ attack |
|--------------|---------------|---------------|--------------|------------|-------------------|---------------------------------------------------|---------------------------------------------------|
| breast-cancer| 10            | 10            | 2            | 4          | 0.3               | 978                                               | 964                                               |
| diabetes     | 8             | 2             | 0.2           | 5          | 4                 | 880                                               | 965                                               |
| ionosphere   | 34            | 2             | 0.2           | 5          | 4                 | 986                                               | 929                                               |

Table 5: The test accuracy and robustness of GBDT models. Average $\ell_1$ and $\ell_2$ distortions of robust GBDT models are consistently larger than those of the naturally trained models. The robustness is evaluated by the average $\ell_1$ and $\ell_2$ distortions of adversarial examples found by Kantchelian’s $\ell_1$ and $\ell_2$ attacks. Only small or medium sized binary classification models can be evaluated by Kantchelian’s attack.

6.2 Omitted Results on Models with Different Number of Trees

In Table 6 we present the test accuracy and $\ell_\infty$ distortion of models with different number of trees obtained by Cheng’s $\ell_\infty$ attack. For each dataset, models are generated during a single boosting run. We can see that the robustness of robustly trained models consistently outperforms that of natural models with the same number of trees.

6.3 More MNIST and Fashion-MNIST Adversarial Examples

In Figure 6 we present more adversarial examples for MNIST and Fashion-MNIST datasets.
| Dataset     | \( \epsilon = 0.3 \) | \( \epsilon = 0.2 \) | \( \epsilon = 0.1 \) | \( \epsilon = 0.05 \) | \( \epsilon = 0.01 \) | \( \epsilon = 0.001 \) |
|-------------|---------------------|---------------------|---------------------|---------------------|---------------------|---------------------|
| breast-cancer (2) | 546, 137, 10 | test | train | test | train | test | train | test | train | test | train | test |
| depth, \( d = 8 \), depth, \( d = 6 \) | 10 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| covtype (7) | 400,000, 54 | train | test | test | test | test | test | test | test | test | test | test |
| depth, \( d = 8 \), depth, \( d = 4 \) | 8 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| cod-rma (2) | 59,535, 271,617 | train | test | test | test | test | test | test | test | test | test | test |
| depth, \( d = 5 \), depth, \( d = 4 \) | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | -1 | -2 | -3 |
| diabetes (2) | 614, 154 | train | test | test | test | test | test | test | test | test | test | test |
| depth, \( d = 8 \), depth, \( d = 5 \) | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | -1 | -2 | -3 |
| Fashion-MNIST (10) | 60,000, 10,000, 784 | train | test | test | test | test | test | test | test | test | test | test |
| depth, \( d = 8 \), depth, \( d = 5 \) | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | -1 | -2 | -3 |
| HIGGS (2) | 10,500,000, 500,000, 28 | train | test | test | test | test | test | test | test | test | test | test |
| depth, \( d = 8 \), depth, \( d = 5 \) | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | -1 | -2 | -3 |
| ijcnn1 (2) | 49,990, 91,701 | train | test | test | test | test | test | test | test | test | test | test |
| depth, \( d = 8 \), depth, \( d = 8 \) | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 |
| MNIST (10) | 60,000, 10,000, 784 | train | test | test | test | test | test | test | test | test | test | test |
| depth, \( d = 10 \), depth, \( d = 5 \) | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | -1 | -2 | -3 |
| Sensorless (11) | 48,509, 10,000 | train | test | test | test | test | test | test | test | test | test | test |
| depth, \( d = 6 \), depth, \( d = 5 \) | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 |
| welspsamp (2) | 300,000, 50,000, 254 | train | test | test | test | test | test | test | test | test | test | test |
| depth, \( d = 8 \), depth, \( d = 5 \) | 254 | 253 | 252 | 251 | 250 | 249 | 248 | 247 | 246 | 245 | 244 | 243 |

Table 6: The test accuracy and robustness of GBDT models. Here depth, \( n \) is the depth of natural trees and depth, \( r \) is the depth of robust trees. Robustness is evaluated by the average \( \ell_{\infty} \) distortion of adversarial examples found by Cheng's attack [Cheng et al., 2019]. The number in the parentheses after each dataset name is the number of classes. Models are generated during a single boosting run. We can see that the robustness of our robust models consistently outperforms that of natural models with the same number of trees.
Figure 6: MNIST and Fashion-MNIST examples and their adversarial examples found using the untargeted Cheng’s $\ell_\infty$ attack (Cheng et al., 2019) on 200-tree gradient boosted decision tree (GBDT) models trained using XGBoost with depth=8. For both MNIST and Fashion-MNIST robust models, we use $\epsilon = 0.3$. 