Robustness for Non-Parametric Classification: A Generic Attack and Defense

Yao-Yuan Yang* **Cyrus Rashtchian*** **Yizhen Wang** **Kamalika Chaudhuri**
{yay005, crashtchian, yiw248, kamalika}@eng.ucsd.edu
University of California, San Diego, Computer Science & Engineering

Abstract

Adversarially robust machine learning has received much recent attention. However, prior attacks and defenses for non-parametric classifiers have been developed in an ad-hoc or classifier-specific basis. In this work, we take a holistic look at adversarial examples for non-parametric classifiers, including nearest neighbors, decision trees, and random forests. We provide a general defense method, adversarial pruning, that works by preprocessing the dataset to become well-separated. To test our defense, we provide a novel attack that applies to a wide range of non-parametric classifiers. Theoretically, we derive an optimally robust classifier, which is analogous to the Bayes Optimal. We show that adversarial pruning can be viewed as a finite sample approximation to this optimal classifier. We empirically show that our defense and attack are either better than or competitive with prior work on non-parametric classifiers. Overall, our results provide a strong and broadly-applicable baseline for future work on robust non-parametrics.

1 Introduction

State-of-the-art classifiers have been shown to suffer from substantial drops in accuracy when faced with adversarially modified inputs even if the modifications are imperceptibly slight. Due to the security concerns that this raises, a body of recent research has investigated the construction and prevention of adversarial examples – small perturbations of valid inputs that cause misclassification (Carlini, 2018; Szegedy et al., 2014).

Most previous work has looked at parametric methods, i.e., neural networks and linear classifiers (Biggio et al., 2013; Lowd and Meek, 2005; Madry et al., 2018; Papernot et al., 2016b), and there is a mature understanding of what properties can be exploited to design adversarial attacks and defenses for any parametric model. For example, parametric classifiers are based on continuous functions with gradients, which has been used to design gradient-based attacks (Athalye et al., 2018; Carlini and Wagner, 2017). Likewise, parametric models are mostly trained by minimizing a training loss, which has been exploited to build an effective and generic defense – adversarial training, retraining after data augmentation with adversarial examples (Carlini et al., 2019; Madry et al., 2018; Song et al., 2019).

An alternative statistical paradigm is that of non-parametric methods, such as nearest neighbor, decision tree, and random forest classifiers, which typically apply to dense data in lower dimensional spaces. These are local predictors, whose output depends on labeled points close to an input. Surprisingly, these methods behave very differently from parametrics when it comes to adversarial examples. In many cases, they have no gradients, and adversarial examples for parametric models fail to transfer (Papernot et al., 2016a). Generic defenses, such as adversarial training, appear to be ineffective as well (Dubey et al., 2019; Papernot and McDaniel, 2018; Wang et al., 2018).

While prior work has constructed attacks and defenses for some specific classifiers (Chen et al., 2019; Dubey et al., 2019; Kantchelian et al., 2016; Sitawarin and Wagner, 2019; Wang et al., 2018), there appear to be no generic approaches, and no generic principles that can be used to guide the design of attacks and defenses for variety of non-parametric methods.

In this work, we identify two key general principles, and use them to design a generic defense and an attack that apply to a variety of non-parametric methods.

To design defenses, we ask: when do non-parametric methods work well? Figure 1 depicts two variants of

*Equal Contribution.

Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS) 2020, Palermo, Italy. PMLR: Volume 108. Copyright 2020 by the author(s).

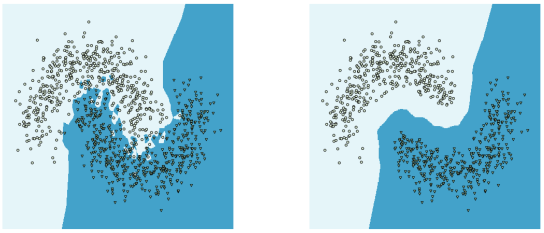


Figure 1: Normal vs. Defended 1-Nearest Neighbor.

random forests. In the left figure, we observe that datasets with nearby oppositely-labeled points may lead to classifiers with convoluted decision boundaries. In the right figure, we see that well-separated data lead to classification regions that are more robust to small perturbations. We will use this low-dimensional intuition as a starting point for generic defense methods.

Figure 1 suggests that since these methods make local predictions, they might work well when data from different classes are well-separated in space. We clearly cannot hope for such separation in most real datasets. Therefore, we propose to preprocess the training data by removing a subset so that different classes are well-separated. To ensure classification accuracy, we propose removing the minimal subset of points that ensure this property. We call our method *Adversarial Pruning*, which can be used as a pre-processing step before training any generic non-parametric classifier.

To evaluate our defense, we propose a new attack that is based on our next key observation: many non-parametric methods divide the instance space into convex polyhedra, and predict in a piecewise constant manner in each. For example, for 1-nearest neighbor, these polyhendra are the Voronoi cells. This suggests the following attack: find the closest polyhedron to an input where the classifier predicts a different label and output the closest point in this region. We implement this strategy by solving a collection of convex programs, and in cases where solution is computationally expensive, we provide a heuristic method for finding an approximate solution. We refer to these attacks as the exact and approximate *region-based attack*.

We next provide some theoretical justification for our methods. For our defense, we show that adversarial pruning can be interpreted as a finite-sample version of a robust analogue to the Bayes Optimal. We formally introduce this robust classifier, that we call the r-optimal, and show that it maximizes *astuteness* (accuracy where it is robust with radius r). For our attack, we show that the exact region-based attack is optimal, in the sense that it yields the closest adversarial example to a test input.

We empirically evaluate the adversarial pruning defense using the region based attack and prior attacks. We provide a general and thorough evaluation, for k-nearest neighbors (k-NN), decision trees, and random forests. We see that adversarial pruning consistently improves robustness, outperforming adversarial training on several datasets and is competitive with classifier-specific defenses. For our attacks, we see that even without any classifier-specific optimization, our new attacks either outperform or are competitive with prior attacks (in terms of perturbation amount). This suggests that both the adversarial pruning defense as well as the region based attack are good generic baselines for evaluating the robustness of non-parametric methods.

2 Preliminaries

We begin with a brief introduction to non-parametric methods that are local classifiers whose output depends on training data close to the test instance. These methods are typically used with dense lower-dimensional data, such as those in Figure 1. Examples are k-nearest neighbor (k-NN) and tree-based classifiers. The *k-NN classifier* outputs the plurality label among the k training examples closest to $\mathbf{x}$ in an ℓ_p metric. A *tree ensemble* contains T decision trees whose leaves are labeled with vectors in $\mathbb{R}^C$. Each input $\mathbf{x}$ determines T root-to-leaf paths, corresponding to vectors $\mathbf{u}^1, \ldots, \mathbf{u}^T$. The output is the largest coordinate in $\mathbf{u}^1 + \cdots + \mathbf{u}^T$. Random forests are a subclass of tree ensembles.

In what follows, $f : \mathbb{R}^d \to [C]$ denotes a classifier with C classes, where $[C] := \{1, 2, \ldots, C\}$. The training data for f is a dataset $\mathcal{S} = \{(\mathbf{x}^j, y^j)\}_{j=1}^n$ of n labeled examples, with $\mathbf{x}^j \in \mathbb{R}^d$ and $y^j \in [C]$.

Robustness. We study robustness in an adversarial model. The adversary's goal is to modify a true input by a small amount and cause the classifier to output the wrong label. Two main threat models have been proposed. The *black-box* setting restricts the adversary to only querying a classifier f on various inputs. In the *white-box* setting, the adversary has full access to f, including the model structure and parameters.

Fix a classifier f and a norm $\|\cdot\|$ on $\mathbb{R}^d$. An *adversarial example* for f at $\mathbf{x}$ is any other input $\widetilde{\mathbf{x}}$ such that $f(\mathbf{x}) \neq f(\widetilde{\mathbf{x}})$. An *optimal adversarial example* for f at $\mathbf{x}$ is an input $\widetilde{\mathbf{x}}$ that minimizes $\|\mathbf{x} - \widetilde{\mathbf{x}}\|$ subject to $f(\mathbf{x}) \neq f(\widetilde{\mathbf{x}})$. In other words, an optimal adversarial example $\widetilde{\mathbf{x}}$ is a closest vector to $\mathbf{x}$ that receives a different label. In practice it is not always possible to find the optimal adversarial example, and hence the goal is to find $\widetilde{\mathbf{x}}$ that is as close to $\mathbf{x}$ as possible. We also define the robustness radius, the minimum perturbation needed to change the classifier label.

Definition 1. Let $\mathcal{X} \times [C]$ be a labeled space with norm $\|\cdot\|$. The *robustness radius* of f at $\mathbf{x} \in \mathcal{X}$ is

$$\rho(f, \mathbf{x}) := \min_{\widetilde{\mathbf{x}} \in \mathcal{X}} \{\|\mathbf{x} - \widetilde{\mathbf{x}}\| : f(\mathbf{x}) \neq f(\widetilde{\mathbf{x}})\}$$

3 Adversarial Pruning Defense

When are non-parametric methods robust? Since these are local classifiers, Figure 1 suggests that they may be robust when training data from different classes is well-separated, and may fail when they overlap.

The training data may not be separated, so we will preprocess the data. We remove a subset of the training set, so that the remaining data are well-separated. Then, we train a non-parametric classifier on the rest. A remaining question is which subset of points to remove. For high classification accuracy, we remove the minimum subset whose removal ensures this property.

This process of removing examples from training set so that certain properties hold is called *pruning*. In this section, we first introduce the method used to prune the dataset. In Section 5, we justify our method by interpreting it in light of classical results in statistical learning theory (Chaudhuri and Dasgupta, 2014; Cover and Hart, 1967; Devroye et al., 1994).

Formally, given a robustness radius r and training set $\mathcal{S}$, we propose the following generic way to preprocess the training set and improve the robustness of classifiers:

Adversarial Pruning. Given r and a set $\mathcal{S}$, compute a maximum subset $\mathcal{S}^{\mathsf{AP}} \subseteq \mathcal{S}$ such that differently-labeled points have distance at least $2r$. Then, train any nonparametric classifier on $\mathcal{S}^{\mathsf{AP}}$.

After computing $\mathcal{S}^{\mathsf{AP}}$ once for a dataset, then we may train any classifier on the pruned training set. Our main hypothesis is that this will lead to more robust classifiers when using non-parametric methods. We will demonstrate empirically that this works well, and we will argue that this defense method is a finite-sample approximation to the optimal robust classifier.

Observe that while adversarial pruning is similar to the defense in Wang et al. (2018), they actually retain additional points with confident labels, which ensures that their method converges to being robust where the *Bayes Optimal* is robust. Their work builds on previous results of Gottlieb et al. (2014a) and Kontorovich and Weiss (2015) that sharpen the risk analysis of 1-NN by using pruning. As we explain in Section 5, our method instead can be interpreted as a finite sample version of a different and more appropriate limit.

One drawback of this approach is that the metric must be fine-grained enough to distinguish between close and far pairs. For most datasets and norms (e.g, Euclidean distance) for which non-parametrics are used, this will be the case. However, for binary features and the ℓ_∞ distance, we have the problem that every pair of different points has distance exactly one, and therefore, the similarity structure is meaningless. To circumvent this, we preprocess the binary feature vectors using standard feature-extraction methods (e.g., PCA), and then operate on the resulting space.

Computing the Robust Dataset. We use known graph algorithms to efficiently compute $\mathcal{S}^{\mathsf{AP}}$. Each training example is a vertex in the graph. Edges connect pairs of differently-labeled examples $\mathbf{x}$ and $\mathbf{x}'$ whenever $\|\mathbf{x} - \mathbf{x}'\| \leq 2r$. We remove as few examples as possible so that no more edges remain. This is equivalent to computing the minimum vertex cover. For binary labels, this graph is bipartite, and a minimum vertex cover can be derived from a maximum matching. The fastest method to solve maximum matching is the Hopcroft-Karp algorithm (Hopcroft and Karp, 1973). For a graph with n vertices and m edges, it takes time $O(m\sqrt{n})$. Fortunately, in practice, the graph of close pairs is quite sparse (for small r and high dimensional feature spaces, with relatively separated classes). For example, if $m = \widetilde{O}(n)$ edges, then computing $\mathcal{S}^{\mathsf{AP}}$ takes time $\widetilde{O}(n^{3/2})$. For large datasets, we note that *linear time* approximation algorithms are known (Duan and Pettie, 2014).

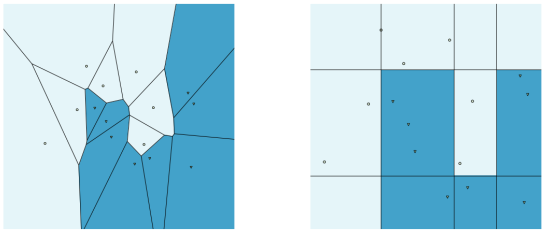
When there are more than two labels, that is $C \geq 3$, it is NP-Hard to compute the optimal pruned subset, but approximation algorithms are known (Gottlieb et al., 2014a; Kontorovich and Weiss, 2015). The greedy algorithm provably generates a 2-approximation. A suboptimal solution still ensures that different classes are separated, and hence, the robustness of the classifier does not require finding the optimal pruned dataset.

4 Region-Based Attack

In this section, we develop a way to evaluate robustness of non-parametric methods. For parametric algorithms, generic gradient-based attacks exist. Our goal is to develop an analogous general attack method, which works well for multiple non-parametrics. Moreover, we aim to develop a white-box attack that will serve as a better baseline than black-box attacks.

The main challenge of finding adversarial examples is that these classifiers have complicated decision regions. The central idea behind our attack is that for many classifiers, such as k-NN or random forests, we can decompose the decision regions into convex sets.

Definition 2. An (s, m)*-decomposition* is a partition of $\mathbb{R}^d$ into convex polyhedra $P_1, \ldots, P_s$ such that each P_i can be described by up to m linear constraints, and f is



(a) 1-NN

(b) Decision tree

Figure 2: (s, m)-decompositions of two non-parametrics.

(s, m)*-decomposable* if there is an (s, m)-decomposition such that f is constant on P_i for each $i \in [s]$.

Figure 2 demonstrates the decomposition for two examples. Figure 2(a) shows how 1-NN is decomposed. In particular, a Voronoi diagram for n points is an $(n, n-1)$-decomposition ($P_1, \ldots, P_n$ are Voronoi cells). If $k \geq 1$, then a k-NN classifier is $\left(\binom{n}{k}, k(n-k)\right)$-decomposable; every k points correspond to polyhedra defined by $k(n-k)$ hyperplanes separating the k points from the other $n-k$ points (Aurenhammer, 1991).

Tree-based classifiers also fit into our framework, and Figure 2(b) shows how a decision tree is decomposed. Any decision tree of depth D with L leaves is (L, D)-decomposable; each root-to-leaf path corresponds to a polyhedron P_i defined by D hyperplanes. Generally, if f is an ensemble of T trees, each with depth D and L leaves, then f is (L^T, DT)-decomposable (proofs in Appendix A). An exponential dependence on T is expected, since the adversarial example problem for tree ensembles is NP-Hard (Kantchelian et al., 2016).

The existence of (s, m)-decompositions suggests the following attack. Given a classifier f and an input $\mathbf{x}$, suppose we could find the closest polyhedron P_i in the decomposition where f predicts a different label than $f(\mathbf{x})$. Then, the closest point in P_i would be the optimal adversarial example. Our attack implements this strategy by searching over all polyhedra.

Region-Based Attack. Let f be an (s, m)-decomposable classifier with decomposition $P_1, \ldots, P_s$, where $f(\mathbf{z}) = y_i$ when $\mathbf{z} \in P_i$, for labels $y_i \in [C]$. To find an adversarial example for $\mathbf{x}$, consider all polyhedra P_i such that $f(\mathbf{x}) \neq y_i$. Then, output $\widetilde{\mathbf{x}}$ minimizing

$$\min_{i: f(\mathbf{x}) \neq y_i} \; \min_{\mathbf{z} \in P_i} \; \|\mathbf{x} - \mathbf{z}\|. \qquad (1)$$

Each P_i is described by $\leq m$ linear constraints, and the norm objective is convex (Boyd and Vandenberghe, 2004). Thus, we can solve each inner minimization problem in (1) separately by solving a convex program with $O(m)$ constraints. This results in candidates $\mathbf{z}^i \in P_i$. Taking the outer minimum over i with $f(\mathbf{x}) \neq y_i$ leads to the optimal adversarial example $\widetilde{\mathbf{x}} = \operatorname{argmin}_{\mathbf{z}^i} \|\mathbf{x} - \mathbf{z}^i\|$.

Efficiency. The running of the exact attack algorithm depends on two things: (i) the number of regions, which is based on the complexity of the classifier, and (ii) the number of constraints and dimensionality of the polyhedra. Due to advances in linear/quadratic program solvers, finding the adversarial example in a single region is quite efficient, i.e., the inner minimization problem in (1) is easy. We find that the number of regions s dominates the running time, i.e., the outer minimization problem in (1) is hard. For k-NN, the number of convex polyhedra scales with $O(n^k)$. When $k = 1$, this is efficiently solvable, because polyhedra have at most n constraints, and the adversarial examples can be found quickly using a linear program for ℓ_∞ perturbations. Unfortunately, for $k > 1$, this attack does not scale well, and we will develop an approximation algorithm for larger values of k.

For a single decision tree, again the exact attack is very efficient, depending only on the number of nodes in the tree. But for larger tree ensembles (e.g., large random forests), the optimal attack is very slow, as expected.

Speeding Up the Search. The exact attack is computationally intensive when s is large; hence, finding optimal solutions is infeasible for random forests (with many trees) or k-NN (when k is large). We next provide a computationally-efficient algorithm, which searches a constant number of regions.

The region-based attack for an (s, m)-decomposable f requires solving up to s convex programs, one for each polyhedron P_i with a different label. If the number of polyhedra is large, then this may be computationally infeasible. Fortunately, (1) has an obvious subdivision, based on the outer minimum over convex polyhedra. We use a relaxation that considers only a subset of polyhedra. We observe that each training point corresponds to a polyhedron—the one that f uses to predict the label. When finding adversarial examples for $\mathbf{x}$, the natural choice is to utilize training data close to $\mathbf{x}$.

Approximate Region-Based Attack. Let $\mathcal{S}$ be the training data. To find an adversarial example under ℓ_p for $\mathbf{x}$, we first compute the subset $\mathcal{S}' \subseteq \mathcal{S}$ of s' points closest in ℓ_p distance to $\mathbf{x}$, while having different training labels than $f(\mathbf{x})$. Next, we determine at most s' polyhedra $P_{i_1}, \ldots, P_{i_{s'}}$ containing points in $\mathcal{S}'$ (as the polyhedra partition $\mathbb{R}^d$). We solve the inner optimization problem in (1) for each P_{i_j} to find candidates $\mathbf{z}^i$ for $i \in [s']$. Finally, we output $\widetilde{\mathbf{x}} = \operatorname{argmin}_{\mathbf{z}^i} \|\mathbf{x} - \mathbf{z}^i\|$, where the minimum is over these s' candidates.

As we only solve $s' \ll s$ convex programs, the running time is greatly reduced compared to the optimal region-based attack. Empirically, this approximation finds adversarial examples with low perturbation.

5 Theoretical Justification

We provide some theoretical results to support our methods. To understand the robustness of non-parametric methods, we first derive a theoretically optimal classifier that takes into account robustness as a core objective. Then, we show that adversarial pruning can be interpreted as a finite sample approximation to the optimally robust classifier. Finally, we analyze the exact and approximate region-based attacks.

5.1 Adversarial Pruning vs. Optimal

Under certain conditions, many non-parametric methods converge in the infinite sample limit to the *Bayes Optimal classifier*, the most accurate classifier for a data distribution. In this way, non-parametric classifiers may be viewed as finite-sample approximations to the Bayes Optimal. However, the Bayes Optimal may not be robust to adversarial examples.

We next introduce a novel robust analogue to the Bayes Optimal. For a perturbation amount r, we call it the r*-Optimal classifier*. Surprisingly, to the best of our knowledge, such an analogue seems to be new in the context of adversarial examples.

Let μ denote a distribution on labeled examples $\mathcal{X} \times [C]$ and fix a distance on $\mathcal{X}$. What is the true objective of a robust classifier? Prior work measures astuteness under μ, which is the probability that the classifier is both r-robust and accurate for a new sample $(\mathbf{x}, y)$ (Madry et al., 2018; Wang et al., 2018).

Definition 3. For distribution μ on $\mathcal{X} \times [C]$, the *astuteness* of a classifier f at radius r is

$$\mathsf{ast}_\mu(f, r) := \Pr_{(\mathbf{x},y)\sim\mu}[\rho(f, \mathbf{x}) \geq r \text{ and } f(\mathbf{x}) = y].$$

Robust Analogue to Bayes Optimal. We exhibit a classifier, the r-Optimal classifier, that achieves optimal astuteness. It is convenient to rewrite astuteness in terms of certain robust subsets of the input space. Then, we define the r-Optimal classifier using these subsets. Formally, for a classifier f and label j, let $S_j(f, r) := \{\mathbf{x} \in \mathcal{X} \mid f(\mathbf{x}) = j \text{ and } \rho(f, \mathbf{x}) \geq r\}$. The following lemma expresses astuteness under μ using these subsets (proofs in Appendix B).

Lemma 1. $\mathsf{ast}_\mu(f, r) = \displaystyle\sum_{j=1}^{C} \int_{\mathbf{x}\in S_j(f,r)} p(y = j \mid \mathbf{x}) d\mu.$

How should we define the classifier that maximizes astuteness? Lemma 1 implies that, to calculate astuteness, it suffices to consider the robust regions $S_j(f, r)$ for a classifier. As a consequence, we claim that in order to determine the optimal classifier, it suffices to find the optimal robust regions under μ. We first formalize this intermediate goal using the following maximization problem.

$$\max_{S_1,\ldots,S_C} \sum_{j=1}^{C} \int_{\mathbf{x}\in S_j} p(y = j \mid \mathbf{x}) d\mu \quad (2)$$
$$\text{s.t. } d(S_j, S_{j'}) \geq 2r \text{ for all } j \neq j'$$

where $d(S_j, S_{j'}) := \min_{u\in S_j, v\in S_{j'}} \|u - v\|$. Notice that for any classifier f, the sets $S_j(f, r)$ for $j \in [C]$ have pairwise distance at least $2r$, implying that they are feasible solutions for (2).

Besides being distance $2r$ apart, an optimal solution $S_1^*, \ldots, S_C^*$ to (2) maximizes accuracy in the following sense. The integral measures the probability that $(\mathbf{x}, y) \sim \mu$ has $y = j$ and $\mathbf{x} \in S_j^*$. In other words, S_j^* has the highest frequency of points with label j under μ, subject to the distance constraint. The sets S_j^* form the basis for the optimal classifier's decision regions. To ensure the separation, we consider the distance r ball around these sets. Formally, we have the following.

Definition 4. Fix r and μ. Let $S_1^*, \ldots, S_C^*$ be optimizers of (2). The r*-Optimal classifier* f_{ropt} is any classifier such that $f_{\mathsf{ropt}}(\mathbf{x}) = j$ whenever $d(\mathbf{x}, S_j^*) \leq r$.

We remark that when $r = 0$, the 0-Optimal classifier is the standard Bayes Optimal classifier. Finally, because $S_j(f_{\mathsf{ropt}}, r) = S_j^*$, Lemma 1 then implies that r-Optimal classifier maximizes astuteness:

Theorem 1. $f_{\mathsf{ropt}} = \operatorname{argmax}_f \mathsf{ast}_\mu(f, r)$.

Finite Sample Approximation. Prior work shows that 1-NN applied to a variant of adversarial pruning leads to provably robust classifiers (Wang et al., 2018). The main difference with our work is their method also selects a subset of confident training examples to keep in the pruned subset - which ensures that the classifier converges to being robust in regions where the Bayes Optimal is robust. In contrast, our aim is to develop generic techniques, for multiple classifiers, and we show that our method can be interpreted as a finite sample approximation to the r-Optimal classifier – the optimally astute classifier.

Adversarial pruning works by removing certain training points so that no oppositely labeled pairs of examples remain. We can view this process in the light of the r-optimal classifier as follows. To prune the dataset $\mathcal{S}$, we solve the maximization problem:

$$\max_{S_1,\ldots,S_C\subseteq\mathcal{S}} \sum_{j=1}^{C} \sum_{\mathbf{x}^i\in S_j} \mathbf{1}_{\{y^i=j\}} \quad (3)$$
$$\text{s.t. } d(S_j, S_{j'}) \geq 2r \text{ for all } j \neq j'.$$

The solution to (3) will be maximum subsets of training data with pairwise distance $2r$. As long as the training set $\mathcal{S}$ is representative of the underlying distribution μ, these subsets will approximate the optimal S_j^* sets. Hence, we posit that a non-parametric method trained on $\mathcal{S}^{\mathsf{AP}}$ should approximate the r-Optimal classifier.

5.2 Attack Algorithm Analysis

The run time of the region-based attack depends on the norm. We focus on ℓ_p with $p \in \{1, 2, \infty\}$ as these are the most relevant for adversarial examples. We prove the following theorem in Appendix A.

Theorem 2. *If f is (s, m)-decomposable, then the region-based attack outputs optimal adversarial examples in time $s \cdot \mathrm{poly}(m, d)$, for ℓ_p distance, $p \in \{1, 2, \infty\}$.*

As k-NN and tree ensembles are (s, m)-decomposable, the region-based attack produces an optimal adversarial example for these. Note that an optimal attack *certifies* the robustness radius. Indeed, if on input $\mathbf{x}$ the region-based attack outputs $\widetilde{\mathbf{x}}$, then $\|\mathbf{x} - \widetilde{\mathbf{x}}\| = \rho(f, \mathbf{x})$.

Approximate Attack Guarantees. We claim that the approximate region-based attack outputs a valid adversarial example when f is (s, m)-decomposable. Each region is defined by m constraints, and f is constant on each region. We search in s' regions, finding the best candidate $\mathbf{z}^i$ from each. Each considered region contains a training example with a different label than $f(\mathbf{x})$. Therefore, the best adversarial example $\widetilde{\mathbf{x}}$ in that region receives a different label $f(\widetilde{\mathbf{x}}) \neq f(\mathbf{x})$. The analysis of the time complexity for finding candidates is $\mathrm{poly}(m, d)$ for each region P_i. Compared to the exact attack (Theorem 2) we only consider s' regions, so the total time is only $s' \cdot \mathrm{poly}(m, d)$. We find in practice that $s' = 50$ regions suffices for a good attack, and the time only scales with m and d.

6 Experiments

We investigate the effectiveness of our methods by evaluating multiple classifiers on nine datasets. We address the following questions:

1. Does adversarial pruning increase robustness across multiple non-parametric classifiers?
2. How well does the region-based attack perform compared with prior work?

Classifiers and Datasets. We evaluate three non-parametric classifiers: k-nearest neighbor (k-NN), decision tree (DT) and random forest (RF) (Breiman, 2001, 2017; Cover and Hart, 1967). We use nine standard binary classification datasets. All features are scaled to be in [0,1]. We evaluate in ℓ_∞ to be consistent with prior work. We reduce the feature dimension of the image datasets (f-mnist and mnist) with PCA to 25 dimensions for two reasons: (i) non-parametrics are normally used for low dimensional spaces, (ii) adversarial pruning requires non-binary features for ℓ_∞. Details are in Appendix C; code in a public repository.[1]

Performance Measures. Besides measuring accuracy, we evaluate attacks using empirical robustness, following prior work (Chen et al., 2019; Kantchelian et al., 2016). Intuitively, we want to measure the perturbation distance to the nearest adversarial example (as opposed to fixing r and evaluating error). Formally, the *empirical robustness* for attack A on f at input $\mathbf{x}$ is $\mathsf{ER}(A, f, \mathbf{x}) := \|\mathbf{x} - \widetilde{\mathbf{x}}_A\|_\infty$, where A outputs $\widetilde{\mathbf{x}}_A$ as the adversarial example for f at $\mathbf{x}$. Observe that larger empirical robustness means worse attacks, and the minimal empirical robustness of f at $\mathbf{x}$ is the robustness radius $\rho(f, \mathbf{x})$. To fairly compare classifiers having different accuracies, we actually compute $\mathsf{ER}(A, f, S, t)$ over t test inputs. To do so, we draw t random samples S_t from S that are classified correctly by f, and we report the average of $\mathsf{ER}(A, f, \mathbf{x})$ over $\mathbf{x} \in S_t$. We set $t = 100$ to balance efficiency and thoroughness.

Again, for defenses, we use perturbation distance to evaluate robustness. Each defense method D produces a classifier f_D. We evaluate a defense D by assigning it a score, the defscore. The *defscore* with respect to an attack A, a test set S and test size t is the ratio

$$\mathrm{defscore}(D, A, f, S, t) = \frac{\mathsf{ER}(A, f_D, S, t)}{\mathsf{ER}(A, f, S, t)},$$

where f is the undefended classifier. A larger defscore implies a better defense.

Attack Algorithms. For 1-NN and DT, we apply the exact region-based attack (RBA-Exact). For 3-NN and RF, the RBA-Exact attack is computationally intensive, and we use the approximate region-based attack (RBA-Approx). For 3-NN, it uses $s' = 50$ polyhedra, and for RF, it uses $s' = 100$ polyhedra. We compare RBA-Exact and RBA-Approx against several baselines. A general attack that applies to all methods is the black-box attack (BBox) (Cheng et al., 2019); this attack seems to be the state-of-the-art for non-parametrics. For k-NN, we compare against two white-box attacks, the direct attack (Direct) and kernel substitution attack (Kernel) (Papernot et al., 2016a). The direct attack perturbs the test instance towards the center of the k nearest oppositely-labeled training examples. The kernel substitution attack uses a soft nearest neighbor to build a substitution model and applies the projected gradient descent attack (Kurakin et al., 2016). For DT, the RBA-Exact attack is optimal, and so is the attack

[1] https://github.com/yangarbiter/adversarial-nonparametrics/

	1-NN					3-NN				DT			RF	
	Direct	BBox	Kernel	RBA Exact	RBA Approx	Direct	BBox	Kernel	RBA Approx	Papernot's	BBox	RBA Exact	BBox	RBA Approx
austr.	.442	.336	.379	**.151**	**.151**	.719	.391	.464	**.278**	.140	.139	**.070**	**.364**	.446
cancer	.223	.364	.358	**.137**	**.137**	.329	.376	.394	**.204**	.459	.334	**.255**	.451	**.383**
covtype	.130	.199	.246	**.066**	.067	.200	.259	.280	**.108**	.254	.083	**.051**	.233	**.214**
diabetes	.074	.112	.165	**.035**	**.035**	.130	.143	.191	**.078**	.237	.133	**.085**	**.181**	.184
f-mnist06	.080	.140	.187	**.029**	.030	.129	.169	.202	**.051**	.189	.134	**.079**	.206	**.188**
f-mnist35	.187	.244	.259	**.075**	.077	.234	.238	.266	**.094**	.262	.185	**.115**	**.188**	.246
fourclass	.109	.124	.137	**.090**	**.090**	.101	.113	.134	**.096**	.288	.197	**.137**	.159	**.133**
halfmoon	.070	.129	.102	**.058**	**.058**	.105	.132	.115	**.096**	.098	.148	**.085**	.182	**.149**
mnist17	.161	.251	.262	**.070**	.073	.221	.261	.269	**.097**	.219	.171	**.123**	**.250**	**.250**

Table 1: The Empirical Robustness for different attacks on four classifiers **(lower is better; best is in bold)**.

	1-NN			3-NN		DT			RF		
	AT	WJC	AP	AT	AP	AT	RS	AP	AT	RS	AP
aus.	0.64	**1.65**	**1.65**	0.68	**1.20**	2.36	**5.86**	2.37	1.07	**1.12**	1.04
can.	0.82	1.05	**1.41**	1.06	**1.39**	0.85	1.09	**1.19**	0.87	**1.54**	1.26
cov.	0.61	**4.38**	**4.38**	0.88	**3.31**	1.47	2.73	**4.51**	1.02	1.01	**2.13**
dia.	0.83	**4.69**	**4.69**	0.87	**2.97**	0.93	1.53	**2.22**	1.19	1.25	**2.22**
f06	0.90	1.93	**2.59**	0.88	**1.75**	1.33	2.33	**2.57**	1.04	1.10	**1.77**
f35	0.83	1.05	**1.19**	0.83	**1.15**	0.97	**3.03**	2.06	0.99	1.23	**1.41**
fou.	0.93	**3.09**	**3.09**	0.89	**3.09**	1.06	1.23	**3.04**	1.03	1.92	**3.59**
hal.	1.05	2.00	**2.78**	0.93	**1.92**	1.54	1.98	**2.58**	1.04	1.01	**1.82**
m17	0.88	1.06	**1.39**	0.80	**1.13**	1.11	**3.97**	1.32	0.88	0.92	**1.26**

Table 2: defscore using different defenses **(higher is better; best is in bold)**. The defscore for undefended classifiers is 1.00 (greater than 1.00 is more robust). We use RBA-Exact for 1-NN and DT, and RBA-Approx for 3-NN and RF. We use RBA-Approx for AT on large datasets.

by Kantchelian et al. (2016); we only report RBA-Exact because these achieve the same results. We also evaluate the heuristic DT attack by Papernot et al. (2016a). For RF, both optimal attacks are infeasible, and we only evaluate BBox and RBA-Approx.

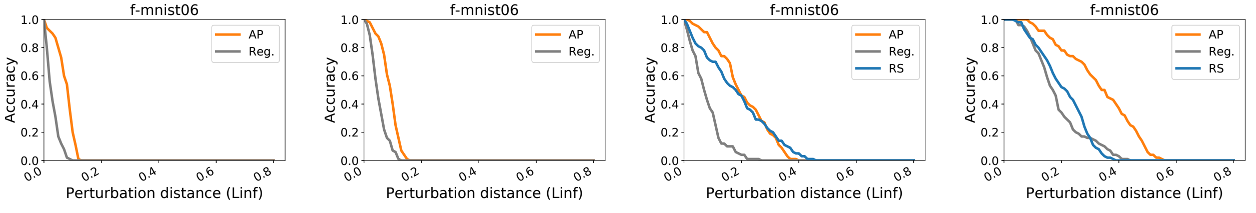
Defense Methods. For our defense, we train each classifier on the dataset pre-processed with adversarial pruning (AP); we use ℓ_∞ to determine examples to prune. For the separation r of AP, we found that $r = 0.3$ balances robustness vs. accuracy. We set $r = 0.3$ for all datasets (Appendix C.4 has other r settings). A generic baseline is adversarial training (AT), where the training data is augmented with examples generated by the corresponding attack algorithm. AT has been reported to be ineffective for 1-NN and boosted decision tree (Wang et al., 2018; Chen et al., 2019), but we include it for completeness. For AT, we retrain the classifier after attacking each training point once; we augment the training data with adversarial examples that are distance at most 0.3 from the original input. The parameter 0.3 matches the parameter r for AP. For 1-NN, an available baseline defense is Wang et al. (2018), but for general k-NN, we are not aware of other defenses. For DT and RF, we compare against the best known defense algorithm, Robust Splitting (RS) (Chen et al., 2019). We set the RS parameter to 0.3 as well.

Results. We separately evaluate attacks and defenses, in Tables 1 and 2, respectively. We provide an accuracy vs. perturbation distance experiment in Figure 3.

Effectiveness of Attacks. Table 1 exhibits empirical robustness across four undefended classifiers and nine datasets. Recall that a smaller empirical robustness implies a more effective attack. For 1-NN, we see that RBA-Exact works as expected, achieving the smallest empirical robustness. For 3-NN, our RBA-Approx attack is more effective than prior attacks, with a much lower empirical robustness. This indicates that RBA-Approx can be a strong attack for $k > 1$, where previously no consistently effective baseline is known. For DT, RBA-Exact again has the best performance. The improvement in many cases shows that the optimal attack for 1-NN and DT can be significantly better than heuristics, which will lead to a more informative defense evaluation. For RF, RBA-Approx wins on five of the nine datasets, and BBox wins on four. Overall, our RBA-Approx attack is competitive with the state-of-the-art attack for RF, and better for 3-NN.

Effectiveness of Defenses. Table 2 shows defscore across four classifiers and several defense methods. For each dataset, the AP defense trains all four classifiers on the same pruned version of the dataset. For all classifiers, we see that AP results in a greater than one defscore, indicating that classifiers trained with AP are more robust. In contrast, AT usually achieves defscore less than one, worse than the undefended classifier; this corroborates previous results (Wang et al., 2018). For 1-NN, observe that AP is slightly better than the defense of Wang et al. (2018). We believe that this is because their method converges to Bayes Optimal, while AP approximates the r-Optimal classifier. For the DT and RF experiments, we see that RS and AP perform competitively, each winning out on some datasets. Overall, AP performs slightly better than RS. We remark that we have evaluated 1-NN and DT against the optimal attack. This provides concrete evidence that AP leads to a more robust classifier.

Discussion. From the results, we see that our generic attack and defense either outperform or perform competitively with prior work on many datasets. We note that there can be a big difference in the perturbation distance depending on the attack algorithms. We also see that our adversarial pruning achieves more robustness compared both to undefended variants and to the classifiers trained using adversarial training. Surprisingly, the pruned subset is computed ahead of time, yet



(a) 1-NN **(b)** 3-NN **(c)** Decision tree **(d)** Random forest

Figure 3: Accuracy (y-axis) vs. perturbation distance (x-axis) for four classifiers on Fashion MNIST classes 0 vs. 6 for the ℓ_∞ distance after applying PCA to 25 dimensions **(larger accuracy is better)**. Other datasets appear in Appendix C.4.1. In the legend, Reg. = regular (undefended) classifier, AP = adversarial pruning, and RS = robust splitting.

it improves the robustness of many different classifiers.

The main conclusion from the experiments is that our work provides a new and suitable baseline for many methods. This is analogous to how AT and PGD are generic baselines for parametrics. In particular, if a new non-parametric algorithm is developed, then AP and RBA may be used to evaluate robustness. Our work also opens to the door to combine AP with classifier-specific defenses, e.g. robust boosting (Chen et al., 2019). We note that our methods can sometimes be slow, but we expect that classier-specific optimizations and techniques will readily improve the running time.

7 Related Work

The bulk of research on robust classifiers has focused on parametric models, with many generic attacks (Carlini and Wagner, 2017; Liu et al., 2017; Papernot et al., 2017b, 2016b; Szegedy et al., 2014), as well as defenses (Hein and Andriushchenko, 2017; Katz et al., 2017; Madry et al., 2018; Papernot et al., 2015; Raghunathan et al., 2018; Sinha et al., 2018). In contrast, adversarial examples for non-parametrics have been studied in a more case-by-case basis.

For tree ensembles, Kantchelian et al. (2016) formulate an optimal attack as a Mixed Integer Linear Program (superseding an earlier attack (Papernot et al., 2016a)) and prove NP-Hardness for many trees. Chen et al. (2019) increase the robustness of *boosted* ensembles. Concurrent work also studies the robustness of decision stumps, and we leave it as future work to compare our methods to theirs (Andriushchenko and Hein, 2019).

For k-NN, prior work on adversarial examples only considers suboptimal attacks, such the direct attack and variants thereof (Amsaleg et al., 2017; Sitawarin and Wagner, 2019; Wang et al., 2018). Concurrent work (Khoury and Hadfield-Menell, 2019) on Voronoi-based adversarial training for neural networks also introduces the optimal attack for 1-NN (i.e., Region-Based attack restricted to 1-NN). In terms of defenses, Wang et al. (2018) increase 1-NN robustness by strategically removing training points. Besides only testing 1-NN against suboptimal attacks, they do not consider other non-parametrics; additionally, their defense is shown to be robust in the large sample limit only where the Bayes Optimal is robust. Our methods are thus more general, and our defense can be interpreted as a finite sample approximation to the r-optimal classifier.

Outside the realm of adversarial examples, pruning has been used to improve the accuracy and generalization (but not robustness) of 1-NN (Gates, 1972; Gottlieb et al., 2014b; Hart, 1968; Kontorovich et al., 2017). Related attacks and defenses have been developed for ReLU networks (Croce et al., 2019; Jordan et al., 2019; Tjeng et al., 2019; Xiao et al., 2019). These results do not directly pertain to non-parametrics, as ReLUs are fundamentally different. The geometric attacks and defenses are similar in spirit to ours. Optimizations based on the dual formulation may improve the efficiency of our methods (Tjeng et al., 2019; Xiao et al., 2019). It would be interesting to explore the relationship between our defense method (adversarial pruning) and the ReLU defense methods and robustness certificates. For example, do robust ReLU networks approximate or converge to the r-Optimal classifier?

8 Conclusion

We consider adversarial examples for non-parametric methods, with a focus on *generic* attacks and defenses. We provide a new attack, the region-based attack, which often outperforms previous attacks. We also provide a new method of defense, adversarial pruning, which should serve as a strong baseline for evaluating the robustness of many classifiers. On the theory side, we prove that the region-based attack outputs the optimal adversarial example. We also introduce and analyze a novel robust analogue to the Bayes Optimal. We prove that the r-Optimal classifier maximizes astuteness. On the experimental side, we demonstrate that our methods are better than or competitive with prior work, while being considerably more general.

Acknowledgments. We thank Somesh Jha, Ruslan Salakhutdinov and Michal Moshkovitz for helpful discussions. Part of this research is supported by ONR under N00014-16-1-261, UC Lab Fees under LFR 18-548554 and NSF under 1804829 and 1617157.

References

Amsaleg, L., Bailey, J., Barbe, D., Erfani, S., Houle, M. E., Nguyen, V., and Radovanović, M. (2017). The vulnerability of learning to adversarial perturbation increases with intrinsic dimensionality. In *WIFS*, pages 1–6.

Andriushchenko, M. and Hein, M. (2019). Provably robust boosted decision stumps and trees against adversarial attacks. *arXiv preprint arXiv:1906.03526*.

Athalye, A., Carlini, N., and Wagner, D. (2018). Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples. In *ICML*, pages 274–283.

Aurenhammer, F. (1991). Voronoi diagrams—a survey of a fundamental geometric data structure. *ACM Computing Surveys*, 23(3):345–405.

Biggio, B., Corona, I., Maiorca, D., Nelson, B., Šrndić, N., Laskov, P., Giacinto, G., and Roli, F. (2013). Evasion attacks against machine learning at test time. In *ECML-PKDD*, pages 387–402.

Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge Univ. Press.

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

Breiman, L. (2017). *Classification and regression trees*. Routledge.

Carlini, N. (2018). *Evaluation and Design of Robust Neural Network Defenses*. PhD thesis, EECS Department, University of California, Berkeley.

Carlini, N., Athalye, A., Papernot, N., Brendel, W., Rauber, J., Tsipras, D., Goodfellow, I. J., Madry, A., and Kurakin, A. (2019). On Evaluating Adversarial Robustness. *CoRR*, abs/1902.06705.

Carlini, N. and Wagner, D. (2017). Towards evaluating the robustness of neural networks. *IEEE Symposium on Security and Privacy*.

Chaudhuri, K. and Dasgupta, S. (2014). Rates of convergence for nearest neighbor classification. In *NeurIPS*, pages 3437–3445.

Chen, H., Zhang, H., Boning, D., and Hsieh, C.-J. (2019). Robust Decision Trees Against Adversarial Examples. In *ICML*.

Cheng, M., Le, T., Chen, P.-Y., Yi, J., Zhang, H., and Hsieh, C.-J. (2019). Query-efficient Hard-label Blackbox Attack: An Optimization-based Approach. In *ICLR*.

Cover, T. M. and Hart, P. E. (1967). Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27.

Croce, F., Andriushchenko, M., and Hein, M. (2019). Provable robustness of relu networks via maximization of linear regions. In *AIStats*.

Devroye, L., Gyorfi, L., Krzyzak, A., and Lugosi, G. (1994). On the strong universal consistency of nearest neighbor regression function estimates. *The Annals of Statistics*, pages 1371–1385.

Duan, R. and Pettie, S. (2014). Linear-time approximation for maximum weight matching. *Journal of the ACM (JACM)*, 61(1):1.

Dubey, A., van der Maaten, L., Yalniz, Z., Li, Y., and Mahajan, D. (2019). Defense against adversarial images using web-scale nearest-neighbor search. *arXiv preprint arXiv:1903.01612*.

Gates, G. (1972). The Reduced Nearest Neighbor Rule. *IEEE transactions on information theory*, 18(3):431–433.

Gottlieb, L.-A., Kontorovich, A., and Krauthgamer, R. (2014a). Efficient classification for metric data. *IEEE Transactions on Information Theory*, 60(9):5750–5759.

Gottlieb, L.-A., Kontorovich, A., and Nisnevitch, P. (2014b). Near-Optimal Sample Compression for Nearest Neighbors. In *NeurIPS*, pages 370–378.

Gurobi Optimization, L. (2018). Gurobi optimizer reference manual.

Hart, P. (1968). The Condensed Nearest Neighbor Rule. *IEEE transactions on information theory*, 14(3):515–516.

Hein, M. and Andriushchenko, M. (2017). Formal guarantees on the robustness of a classifier against adversarial manipulation. In *NeurIPS*, pages 2263–2273.

Hopcroft, J. E. and Karp, R. M. (1973). An nˆ5/2 algorithm for maximum matchings in bipartite graphs. *SIAM Journal on computing*, 2(4):225–231.

Jordan, M., Lewis, J., and Dimakis, A. G. (2019). Provable Certificates for Adversarial Examples: Fitting a Ball in the Union of Polytopes. *arXiv preprint arXiv:1903.08778*.

Kantchelian, A., Tygar, J., and Joseph, A. (2016). Evasion and Hardening of Tree Ensemble Classifiers. In *ICML*, pages 2387–2396.

Katz, G., Barrett, C., Dill, D. L., Julian, K., and Kochenderfer, M. J. (2017). Towards proving the adversarial robustness of deep neural networks. *arXiv preprint arXiv:1709.02802*.

Khoury, M. and Hadfield-Menell, D. (2019). Adversarial Training with Voronoi Constraints. *Safe Machine Learning workshop at ICLR.*

Kontorovich, A., Sabato, S., and Weiss, R. (2017). Nearest-neighbor Sample Compression: Efficiency, Consistency, Infinite Dimensions. In *NeurIPS*, pages 1573–1583.

Kontorovich, A. and Weiss, R. (2015). A Bayes Consistent 1-NN classifier. In *AIStats*.

Kurakin, A., Goodfellow, I. J., and Bengio, S. (2016). Adversarial examples in the physical world.

Liu, Y., Chen, X., Liu, C., and Song, D. (2017). Delving into transferable adversarial examples and black-box attacks. *ICLR*.

Lowd, D. and Meek, C. (2005). Adversarial learning. In *SIGKDD*, pages 641–647.

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

Manning, C., Raghavan, P., and Schütze, H. (2010). Introduction to Information Retrieval. *Natural Language Engineering*, 16(1):100–103.

Mulmuley, K. (1991). On levels in arrangements and voronoi diagrams. *Discrete & Computational Geometry*, 6(3):307–338.

Papernot, N., Carlini, N., Goodfellow, I., Feinman, R., Faghri, F., Matyasko, A., Hambardzumyan, K., Juang, Y.-L., Kurakin, A., Sheatsley, R., Garg, A., and Lin, Y.-C. (2017a). cleverhans v2.0.0: an adversarial machine learning library. *arXiv preprint arXiv:1610.00768*.

Papernot, N. and McDaniel, P. (2018). Deep k-nearest neighbors: Towards confident, interpretable and robust deep learning. *arXiv preprint arXiv:1803.04765*.

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

Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, B., and Swami, A. (2017b). Practical black-box attacks against deep learning systems using adversarial examples. In *ASIACCS*.

Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z. B., and Swami, A. (2016b). The limitations of deep learning in adversarial settings. In *EuroS&P*.

Papernot, N., McDaniel, P., Wu, X., Jha, S., and Swami, A. (2015). Distillation as a defense to adversarial perturbations against deep neural networks. *arXiv preprint arXiv:1511.04508*.

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

Raghunathan, A., Steinhardt, J., and Liang, P. (2018). Certified defenses against adversarial examples. In *ICLR*.

Sinha, A., Namkoong, H., and Duchi, J. (2018). Certifiable Distributional Robustness with Principled Adversarial Training. In *ICLR*.

Sitawarin, C. and Wagner, D. (2019). On the Robustness of Deep K-Nearest Neighbors. *arXiv preprint arXiv:1903.08333*.

Song, C., He, K., Wang, L., and Hopcroft, J. E. (2019). Improving the Generalization of Adversarial Training with Domain Adaptation. In *ICLR*.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. (2014). Intriguing properties of neural networks. In *ICLR*.

Tjeng, V., Xiao, K., and Tedrake, R. (2019). Evaluating Robustness of Neural Networks with Mixed Integer Programming. In *ICLR*.

Wang, Y., Jha, S., and Chaudhuri, K. (2018). Analyzing the Robustness of Nearest Neighbors to Adversarial Examples. In *ICML*, pages 5120–5129.

Xiao, K. Y., Tjeng, V., Shafiullah, N. M., and Madry, A. (2019). Training for faster adversarial robustness verification via inducing relu stability. In *ICLR*.

A Attack Algorithm: Theoretical Results and Omitted Proofs

In this section, we analyze the exact and approximate region-based attacks. To do so, we provide details about the decompositions for k-NN and tree ensemble classifiers. We also prove Theorem 2 in general, and we give a corollary for the classifiers that we consider. Finally, we discuss our approximate attack, providing more details and an analysis.

Before getting into these details, we observe that our attack actually holds for the more general class of linear decision trees, which we now define.

Defining Linear Decision Trees

A *linear decision tree* is a binary tree consisting of (i) internal nodes associated with affine functions and (ii) leaf nodes associated with labels in $[C]$. The value $f(\mathbf{x})$ is determined by following the root to a leaf, going left or right depending on whether $\mathbf{x}$ satisfies or violates the linear constraint in the current node; then, $f(\mathbf{x})$ is the label of the leaf. Such trees generalize (standard) decision trees, which restrict each constraint to depend on a single variable.

An *ensemble of linear decision trees* is collection of trees with the modification that the leaves are labeled with vectors in $\mathbb{R}^C$. The value $f(\mathbf{x})$ is determined by a two-stage process. First, find the root-to-leaf path associated with each tree separately, resulting in a collection of vectors $\mathbf{u}^1, \dots, \mathbf{u}^T \in \mathbb{R}^C$, where T is the number of trees. Then, letting $\mathbf{u} = \mathbf{u}^1 + \cdots + \mathbf{u}^T$, the output $f(\mathbf{x})$ equals the index of the largest coordinate $i \in [C]$ in the vector $\mathbf{u}$. Note that for binary labels, this is equivalent to the definition of having scalar leaf labels and outputting the sign of the sum.

A.1 Decompositions for Specific Classifiers

We now describe the decompositions for tree ensembles and k-NN. Parameters for the decompositions will directly determine the running time of the optimal attack algorithm.

Decomposition for Tree Ensembles

Lemma 2. *If f is an ensemble of T linear decision trees, each with depth at most D and with at most L leaves, then f is (L^T, TD)-decomposable.*

Proof. We first describe the decomposition for a single tree, then generalize to an ensemble of trees. Let $\mathcal{T}$ be a linear decision tree with depth D leaves $(\ell_1, \ell_2, \dots, \ell_m)$. The polyhedron P_i will be the set of $\mathbf{z}$ that reach leaf ℓ_i in $\mathcal{T}$. The hyperplane description for P_i can be computed as follows. Each internal node v from the root of $\mathcal{T}$ to the leaf ℓ_i contains a linear constraint $a_v(\mathbf{z}) \le b_v$. On the path to ℓ_i, group all the violated (resp. satisfied) constraints a_v, b_v as rows of the matrix A^- and entries of the vector $\mathbf{b}^-$ (resp. A^+ and $\mathbf{b}^+$). Then, all $\mathbf{z}$ that reach ℓ_i are exactly the vectors that satisfy $A^-\mathbf{z} > \mathbf{b}^-$ and $A^+\mathbf{z} \le \mathbf{b}^+$. Therefore, these at most D constraints determine P_i precisely.

Now, consider ensembles of T trees with depth at most D and at most L leaves. The polyhedra correspond to combinations of one leaf from each tree. Each leaf contributes at most D constraints, for at most TD total constraints. There are at most L^T choices for one leaf from each of T trees. □

Decomposition for k-NN

The decomposition for k-NN is a standard fact, known as the k^{th} order Voronoi diagram, and it is a classical result in machine learning and computational geometry (see for example Chapter 12 in the book (Manning et al., 2010), or the survey (Aurenhammer, 1991), or the paper (Mulmuley, 1991)). We sketch a proof for completeness.

Lemma 3. *If f is a k-NN classifier for a dataset of size n, then f is $(\binom{n}{k}, k(n-k))$-decomposable.*

Proof. (Sketch). Let $\mathcal{S}$ be the training dataset on n points. We define $\binom{n}{k}$ convex polyhedra, one for each subset $U \subseteq \mathcal{S}$ containing $|U| = k$ points. The polyhedron P_U is the subset of $\mathbb{R}^d$ such that if $\mathbf{z} \in P_U$, then the k nearest neighbors to $\mathbf{z}$ from the dataset $\mathcal{S}$ in the ℓ_2 distance are the k points in U. By definition, the k-NN classifier will be constant on each polyhedron P_U, as the output label is completely determined by the k nearest neighbors for $\mathbf{z}$, which is the set U.

We show that P_U can be defined by $k(n-k)$ hyperplanes as follows. For each of the k points $\mathbf{x} \in U$, we use the $(n-k)$ bisecting hyperplanes separating $\mathbf{x}$ from each of the $n-k$ points not in U (that is, separating $\mathbf{x}$ from the points $\mathcal{S} \setminus U$). This is a total of $k(n-k)$ linear constraints, and we define P_U as the intersection of these $k(n-k)$ halfspaces. Clearly, P_U is a convex polyhedron.

To see the nearest neighbor property, consider any $\mathbf{z} \in P_U$. For every $\mathbf{x} \in U$, the constraints defining P_U include the $(n-k)$ bisecting hyperplanes that separate $\mathbf{x}$ from the $n-k$ points outside of U. In particular, $\mathbf{z}$ is closer to $\mathbf{x}$ than to these $n-k$ other points. To put this another way, $\mathbf{z}$ is in the Voronoi cell for $\mathbf{x}$ in the reduced dataset consisting only of $\mathbf{x}$ and the other $n-k$ points (that is, $\mathbf{x} \cup (\mathcal{S} \setminus U)$). As this is true for each of the k points in U, we have that $\mathbf{z}$ is closer to each of the k points in U than to the other $n-k$ points.

Therefore, we conclude that U consists of the k nearest neighbors to $\mathbf{z}$. □

A.2 Analyzing the Region-Based Attack

We have just shown that f is decomposable when it is the classifier determined by k-NN or a linear decision tree (or, more generally, an ensemble of linear decision trees). The consequence of this is that Theorem 2 implies an efficient and optimal algorithm for a wide-range of non-parametric classifiers. We first discuss the specific convex programs, then finish the proof of the theorem.

Norms as Convex Objectives

Recall that if a classifier is (s, m)-decomposable, then there exists s polyhedra $P_1, \ldots, P_s$ such that each P_i is the intersection of at most m halfspaces. Moreover, the classifier is constant on each of these convex regions, predicting label y_i at all points in P_i.

For an input $\mathbf{x}$, let $\mathcal{I}_{\mathbf{x}}$ be the indices of polyhedra P_i such that $f(\mathbf{x}) \neq y_i$. Then, the region-based attack optimizes over all polyhedra P_i for $i \in \mathcal{I}_{\mathbf{x}}$ by solving the inner minimization of Equation (1), namely

$$\min_{\mathbf{z} \in P_i} \|\mathbf{x} - \mathbf{z}\|_p. \tag{4}$$

Given that P_i is a polyhedron, the constraint $\mathbf{z} \in P_i$ can be expressed using the m linear constraints that define P_i. Then, the norm minimization can be expressed as a convex objective. In particular, the problem (4) can be solved with a linear program for $p \in \{1, \infty\}$ or a quadratic program for $p = 2$ using standard techniques (Boyd and Vandenberghe, 2004). The following are the specific LP formulations for $p \in \{1, \infty\}$.

ℓ_∞ **norm.** Let $t \in \mathbb{R}$ be single variable. When $p = \infty$, the problem in (4) can be solved in $\mathbb{R}^d$ using the following linear program with $d+1$ variables and $m+2d$ linear constraints.

$$\begin{aligned} \underset{\mathbf{z}, t}{\text{minimize}} \quad & t \\ \text{subject to} \quad & \mathbf{z} \in P_i \\ & (\mathbf{z} - \mathbf{x})_j \leq t \qquad \forall\, j \in [d] \\ & (\mathbf{z} - \mathbf{x})_j \geq -t \quad \forall\, j \in [d] \end{aligned} \tag{5}$$

ℓ_1 **norm.** Let $\mathbf{t} \in \mathbb{R}^d$ be vector. When $p = 1$, the problem in (4) can be solved in $\mathbb{R}^d$ using the following linear program with $2d$ variables and $m + 2d$ linear constraints.

$$\begin{aligned} \underset{\mathbf{z}, \mathbf{t}}{\text{minimize}} \quad & \mathbf{1}^T \mathbf{t} \\ \text{subject to} \quad & \mathbf{z} \in P_i \\ & (\mathbf{z} - \mathbf{x})_j \leq \mathbf{t}_j \qquad \forall\, j \in [d] \\ & (\mathbf{z} - \mathbf{x})_j \geq -\mathbf{t}_j \quad \forall\, j \in [d] \end{aligned} \tag{6}$$

Finishing the Analysis of the Exact Region-Based Attack

Proof of Theorem 2. We first claim that the attack produces the optimal adversarial example when f is any (s, m)-decomposable classifier. By assumption, there is a partition of $\mathbb{R}^d$ into polyhedra $P_1, \ldots, P_s$ such that f is constant on each P_i region. Let y_i be the label that f gives to all points in P_i for each $i \in [s]$. On input $\mathbf{x}$, the algorithm considers $i \in \mathcal{I}_{\mathbf{x}}$, where $\mathcal{I}_{\mathbf{x}} \subseteq [s]$ are the indices such that $f(\mathbf{x}) \neq y_i$. Thus, the point $\mathbf{z}^i \in P_i$ closest to $\mathbf{x}$ will have

$$f(\mathbf{z}^i) = y_i \neq f(\mathbf{x}).$$

Finally, the algorithm's output is

$$\underset{\{\mathbf{z}^i \mid i \in \mathcal{I}_{\mathbf{x}}\}}{\text{argmin}} \|\mathbf{z}^i - \mathbf{x}\|.$$

As the regions P_i partition $\mathbb{R}^d$, this is the closest point to $\mathbf{x}$ that receives a different label under f.

We now analyze the running time. For the ℓ_p distance, $p \in \{1, 2, \infty\}$, finding each candidate point $\mathbf{z}^i$ requires solving a convex program with $O(m)$ constraints and $O(d)$ variables. This can be done in $\text{poly}(d, m)$ time using standard optimization techniques (e.g., the interior point method). The number of convex programs is $|\mathcal{I}_{\mathbf{x}}| \leq s$. Therefore, the total running time is at most $s \cdot \text{poly}(d, m)$. □

Remark 3 (Targeted Attack). So far, we have considered untargeted attacks, allowing adversarial examples to have any label other than $f(\mathbf{x})$. An important variation is a *targeted attack*, which specifies a label $\ell \in [C]$, and the goal is to output a close point $\widetilde{\mathbf{x}}$ such that $f(\widetilde{\mathbf{x}}) = \ell$. We note that the region-based attack can be easily modified for this by only searching over $\mathcal{I}_{\mathbf{x}}^{\ell} = \{i \in [s] \mid y_i = \ell\}$. This may significantly reduce the running time in practice, as $|\mathcal{I}_{\mathbf{x}}^{\ell}|$ may be much smaller than $|\mathcal{I}_{\mathbf{x}}|$.

We specialize the above theorem to ensembles of linear decision trees and the k-NN classifier.

Corollary 1. *Let n be the size of the training set. If $f : \mathbb{R}^d \to [C]$ is a classifier determined by k-NN with $k = O(1)$ or an ensemble of $O(1)$ linear decision trees with depth* poly(n) *and* poly(n) *total leaves, then the region-based attack outputs the optimal adversarial example in time* poly(d, n).

Proof. When f is an ensemble of T linear decision trees, each with depth D and L leaves, Lemma 2 implies that f is (L^T, TD)-decomposable. Assuming that T is a constant and L and D are polynomial means that f is $(\mathrm{poly}(n), \mathrm{poly}(n))$-decomposable. Applying Theorem 2, the running time of the exact region-based attack is thus $\mathrm{poly}(d, n)$.

When f is the k-NN classifier, Lemma 3 implies that f is $(\binom{n}{k}, k(n-k))$-decomposable. Assuming that k is a constant means that f is $(\mathrm{poly}(n), O(n))$-decomposable. Applying Theorem 2, the running time of the exact region-based attack is thus $\mathrm{poly}(d, n)$. □

We leave it as an interesting open question to develop provably optimal algorithms with better running time. For example, in the case of large tree ensembles, the attack searches over all combinations of one leaf from each tree. This seems wasteful, as many of these polyhedra may be empty (in fact, we find that most potential regions are infeasible for random forests trained on real datasets).

B r-Optimal Classifier: Theoretical Results and Omitted Proofs

In this section, we prove that the r-Optimal classifier maximizes astuteness. This result hinges on Lemma 1, which we also prove. Let μ be a distribution on labeled examples $\mathcal{X} \times [C]$.

Proof of Lemma 1. Recall the definition of the robust regions of a classifier,

$$S_j(f, r) = \{\mathbf{x} \in \mathcal{X} \mid f(\mathbf{x}) = j \text{ and } \rho(f, \mathbf{x}) \geq r\}.$$

Starting with the definition of astuteness, we compute the following.

$$\begin{aligned}
&\mathsf{ast}_\mu(f, r) \\
&\quad = \Pr_{(\mathbf{x},y)\sim\mu}[\rho(f, \mathbf{x}) \geq r \text{ and } f(\mathbf{x}) = y] \\
&\quad = \int_{\mathbf{x}} p(y \mid \mathbf{x}) \cdot \mathbf{1}_{\{\rho(f,\mathbf{x})\geq r\}} \cdot \mathbf{1}_{\{f(\mathbf{x})=y\}} \; d\mu(\mathbf{x}) \\
&\quad = \sum_{j=1}^{C} \int_{\mathbf{x}} p(y = j \mid \mathbf{x}) \mathbf{1}_{\{\rho(f,\mathbf{x})\geq r\}} \mathbf{1}_{\{f(\mathbf{x})=j\}} d\mu(\mathbf{x}) \\
&\quad = \sum_{j=1}^{C} \int_{\mathbf{x}\in S_j(f,r)} p(y = j \mid \mathbf{x}) \; d\mu(\mathbf{x}).
\end{aligned}$$

□

We now use the above lemma to prove that the r-Optimal classifier maximizes astuteness.

Proof of Theorem 1. Recall that the r-Optimal classifier f_{ropt} is defined in terms of an optimal solution $S_1^*, \ldots, S_C^*$ to the maximization problem

$$\max_{S_1,\ldots,S_C} \sum_{j=1}^{C} \int_{\mathbf{x}\in S_j} p(y = j|\mathbf{x}) d\mu(\mathbf{x})$$

subject to

$d(S_j, S_{j'}) \geq 2r$ for all $j \neq j'$.

By definition, $f_{\mathsf{ropt}}(\mathbf{x}) = j$ whenever $d(S_j^*, \mathbf{x}) \leq r$. In other words, $S_j^* = S_j(f_{\mathsf{ropt}}, r)$.

We will need the fact that for any classifier f, the sets $S_j(f, r)$ are a feasible solution to the above maximization problem. That is, for $j \neq j'$, the distance between $S_j(f, r)$ and $S_{j'}(f, r)$ is at least $2r$. To see this, consider any two points $u \in S_j(f, r)$ and $v \in S_{j'}(f, r)$. Then, consider the line segment between them $w = \lambda u + (1 - \lambda)v$, for $\lambda \in [0, 1]$. By definition of the robustness radius, we know that $f(w) = f(u) = j$ whenever $d(w, u) \leq r$. Similarly, $f(w) = f(v) = j'$ whenever $d(w, v) \leq r$. Therefore, we must have that $d(u, v) \geq 2r$. As u and v were an arbitrary pair of points in $S_j(f, r)$ and $S_{j'}(f, r)$, we conclude that these subsets have distance at least $2r$, and this holds for all $j \neq j'$.

Using Lemma 1, we now compute the following.

$$\begin{aligned}
\mathsf{ast}_\mu(f, r) &= \sum_{j=1}^{C} \int_{\mathbf{x}\in S_j(f,r)} p(y = j \mid \mathbf{x}) \; d\mu(\mathbf{x}) \\
&\leq \sum_{j=1}^{C} \int_{\mathbf{x}\in S_j^*} p(y = j \mid \mathbf{x}) \; d\mu(\mathbf{x}) \\
&= \sum_{j=1}^{C} \int_{\mathbf{x}\in S_j(f_{\mathsf{ropt}},r)} p(y = j \mid \mathbf{x}) \; d\mu(\mathbf{x}) \\
&= \mathsf{ast}_\mu(f_{\mathsf{ropt}}, r).
\end{aligned}$$

The inequality uses that the sets $S_j(f, r)$ have pairwise distance at least $2r$, and therefore, they are feasible for the above maximization problem, which has optimal solution $S_j^* = S_j(f_{\mathsf{ropt}}, r)$. □

C More Experimental Details

The experiment is run on desktop with Intel - Core i7-9700K 3.6 GHz 8-Core Processor and 32 GB of RAM.

C.1 Classifier Implementation Details

The implementation for DT, RF and k-NN are based on `scikit-learn` (Pedregosa et al., 2011). For DT and RF, the splitting criterion is set to "entropy".

For computational efficiency, we fixed the maximum depth of DT and RF to be five. For reproducibility, all other hyper-parameters are set to the default parameter settings of the specific implementation.

C.2 Attack and Defense Implementation Details

For kernel substitution attack, we set the approximation parameter $c = 1.0$ and attack the substitution model with Projected Gradient Descent (PGD) (Madry et al., 2018). For both Region-Based Attacks (RBA-Exact and RBA-Approx), the underlying LP solver that we use is Gurobi (Gurobi Optimization, 2018). For kernel substitution attack, we use PGD implemented in `Cleverhans` (Papernot et al., 2017a). The implementation of the black-box attack by (Cheng et al., 2019) (BBox) is provided by authors in their public repository.[2]

For k-NN, we do not compare with the gradient-based extension (Sitawarin and Wagner, 2019) attack directly in Section 6 since it is under a different setting. Their algorithm only works if k-NN uses the cosine distance instead of ℓ_2 distance.

C.3 Dataset Details

For each dataset, we reserve 200 examples for testing. We evaluate the testing accuracy on these 200 examples. To compute empirical robustness $\mathsf{ER}(A, f_D, S, t)$ and defscore(D, A, f, S, t), we randomly select 100 correctly predicted examples for each classifier. For efficiency purposes, the feature dimension for fashion-mnist (f-mnist), mnist is reduced to 25 using principle component analysis (PCA). The original covtype is subsampled to 2200 examples. mnist17 represents a subset of mnist dataset for the binary classification problem distinguishing between 1 and 7. Similarly, f-mnist35 is the task of distinguishing between 3rd and 5th class, and f-mnist06 is the task of distinguishing between 0th and 6th class. The features are scaled to $[0, 1]$ so the solver will avoid numerical rounding errors.

C.4 Additional Experiment Results

Table 3, 4, 5, 6 show additional experiment results with adversarial pruning (AP) as defense. In these tables, for AP with separation parameter $r = 0.5$, we have some invalid values. These values are caused by setting a too large value of r which results int that the adversarial pruned datasets to be highly unbalanced in label or even making the dataset have a single label left. If the training accuracy goes below 0.5 or the prediction of the classifier outputs only one label, we will put the value being ”-” in the table. For diabetes with 3-NN, its caused by 3-NN only predicts one label.

Testing accuracy is a sanity check that we are not giving away all accuracy for robustness. The higher the empirical robustness is means the classifier is more robust to the given attack. When considering the strength of the attack, empirical robustness is lower the better. When considering the strength of the defense, defscore is higher the better. For defscore higher mean that after defense (AP), the classifier become more robust, thus higher the better.

C.4.1 Defense figures

Figures 4 and 5 show the complete experiment results for the experiment in Figure 3. The accuracy (y-axis) is measured on the 100 correctly predicted testing examples sampled initially.

[2] https://github.com/cmhcbb/attackbox

	1-NN			AP (separation parameter r=.1)				AP (separation parameter r=.3)				AP (separation parameter r=.5)			
	ER	test accuracy	# train	ER	test accuracy	# train	defscore	ER	test accuracy	# train	defscore	ER	test accuracy	# train	defscore
austr.	.151	.805	490	.162	.800	484	1.073	.249	.820	458	1.649	.311	.825	427	2.060
cancer	.137	.950	483	.137	.950	483	1.000	.193	.950	473	1.409	.261	.965	458	1.905
covtype	.066	.725	2000	.072	.700	1904	1.091	.289	.685	1417	4.379	.346	.675	1384	5.242
diabetes	.035	.695	568	.035	.700	535	1.000	.164	.660	379	4.686	.375	.660	370	10.714
f-mnist06	.029	.800	12000	.031	.820	11509	1.069	.075	.765	7348	2.586	-	.495	6000	-
f-mnist35	.075	1.000	12000	.075	1.000	11999	1.000	.089	.980	10477	1.187	.104	.945	8139	1.387
fourclass	.090	1.000	662	.107	.960	559	1.189	.278	.750	453	3.089	-	.565	442	-
halfmoon	.058	.920	2000	.151	.915	1702	2.603	.161	.840	1144	2.776	-	.480	1004	-
mnist17	.070	.975	13007	.072	.975	13004	1.029	.097	.965	11128	1.386	.118	.810	6783	1.686

Table 3: The number of training data left after adversarial pruning (AP), testing accuracy, empirical robustness, and defscore with different separation parameter of AP for 1-NN.

	3-NN			AP (separation parameter r=.1)				AP (separation parameter r=.3)				AP (separation parameter r=.5)			
	ER	test accuracy	# train	ER	test accuracy	# train	defscore	ER	test accuracy	# train	defscore	ER	test accuracy	# train	defscore
austr.	.278	.805	490	.317	.810	484	1.140	.333	.815	458	1.198	.371	.825	427	1.335
cancer	.204	.975	483	.204	.975	483	1.000	.283	.960	473	1.387	.350	.970	458	1.716
covtype	.108	.750	2000	.117	.735	1904	1.083	.357	.685	1417	3.306	.394	.680	1384	3.648
diabetes	.078	.755	568	.078	.750	535	1.000	.232	.655	379	2.974	-	.660	370	-
f-mnist06	.051	.795	12000	.050	.825	11509	.980	.089	.750	7348	1.745	-	.495	6000	-
f-mnist35	.094	1.000	12000	.093	1.000	11999	.989	.108	.985	10477	1.149	.121	.950	8139	1.287
fourclass	.096	.995	662	.127	.960	559	1.323	.297	.750	453	3.094	-	.565	442	-
halfmoon	.096	.940	2000	.159	.920	1702	1.656	.184	.845	1144	1.917	-	.480	1004	-
mnist17	.097	.985	13007	.094	.985	13004	.969	.110	.960	11128	1.134	.141	.795	6783	1.454

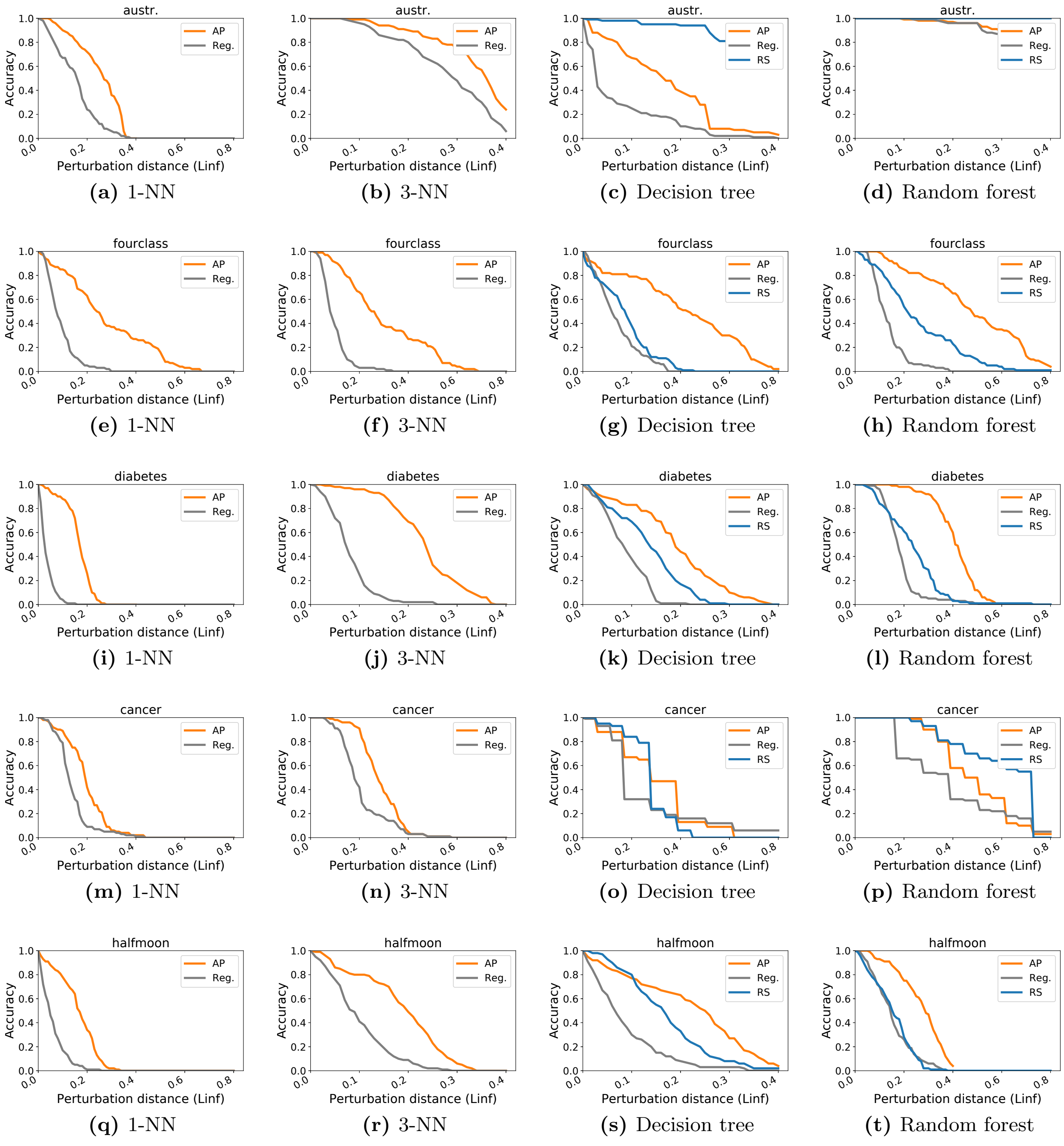
Table 4: The number of training data left after adversarial pruning (AP), testing accuracy, empirical robustness, and defscore with different separation parameter of AP for 3-NN.

	DT			AP (separation parameter r=.1)				AP (separation parameter r=.3)				AP (separation parameter r=.5)			
	ER	test accuracy	# train	ER	test accuracy	# train	defscore	ER	test accuracy	# train	defscore	ER	test accuracy	# train	defscore
austr.	.070	.855	490	.194	.835	484	2.771	.166	.835	458	2.371	.450	.835	427	6.429
cancer	.255	.930	483	.255	.930	483	1.000	.303	.965	473	1.188	.358	.960	458	1.404
covtype	.051	.715	2000	.051	.740	1904	1.000	.230	.680	1417	4.510	.221	.665	1384	4.333
diabetes	.085	.715	568	.085	.720	535	1.000	.189	.670	379	2.224	.378	.670	370	4.447
f-mnist06	.079	.805	12000	.092	.825	11509	1.165	.203	.770	7348	2.570	-	.495	6000	-
f-mnist35	.115	.995	12000	.110	.995	11999	.957	.237	.940	10477	2.061	.281	.925	8139	2.443
fourclass	.137	.900	662	.138	.910	559	1.007	.416	.680	453	3.036	-	.565	442	-
halfmoon	.085	.950	2000	.167	.895	1702	1.965	.219	.670	1144	2.576	-	.480	1004	-
mnist17	.123	.975	13007	.126	.970	13004	1.024	.162	.955	11128	1.317	.316	.830	6783	2.569

Table 5: The number of training data left after adversarial pruning (AP), testing accuracy, empirical robustness, and defscore with different separation parameter of AP for DT.

	RF			AP (separation parameter r=.1)				AP (separation parameter r=.3)				AP (separation parameter r=.5)			
	ER	test accuracy	# train	ER	test accuracy	# train	defscore	ER	test accuracy	# train	defscore	ER	test accuracy	# train	defscore
austr.	.446	.845	490	.426	.855	484	.955	.465	.840	458	1.043	.496	.835	427	1.112
cancer	.383	.970	483	.383	.970	483	1.000	.481	.965	473	1.256	.496	.955	458	1.295
covtype	.214	.750	2000	.226	.700	1904	1.056	.456	.680	1417	2.131	.481	.695	1384	2.248
diabetes	.184	.755	568	.175	.740	535	.951	.409	.660	379	2.223	.710	.660	370	3.859
f-mnist06	.188	.790	12000	.215	.785	11509	1.144	.333	.755	7348	1.771	-	.495	6000	-
f-mnist35	.246	1.000	12000	.236	.995	11999	.959	.346	.925	10477	1.407	.289	.925	8139	1.175
fourclass	.133	.980	662	.181	.865	559	1.361	.478	.665	453	3.594	-	.565	442	-
halfmoon	.149	.930	2000	.198	.900	1702	1.329	.271	.755	1144	1.819	-	.480	1004	-
mnist17	.250	.970	13007	.230	.965	13004	.920	.314	.945	11128	1.256	.359	.800	6783	1.436

Table 6: The number of training data left after adversarial pruning (AP), testing accuracy, empirical robustness, and defscore with different separation parameter of AP for RF.



(a) 1-NN (b) 3-NN (c) Decision tree (d) Random forest

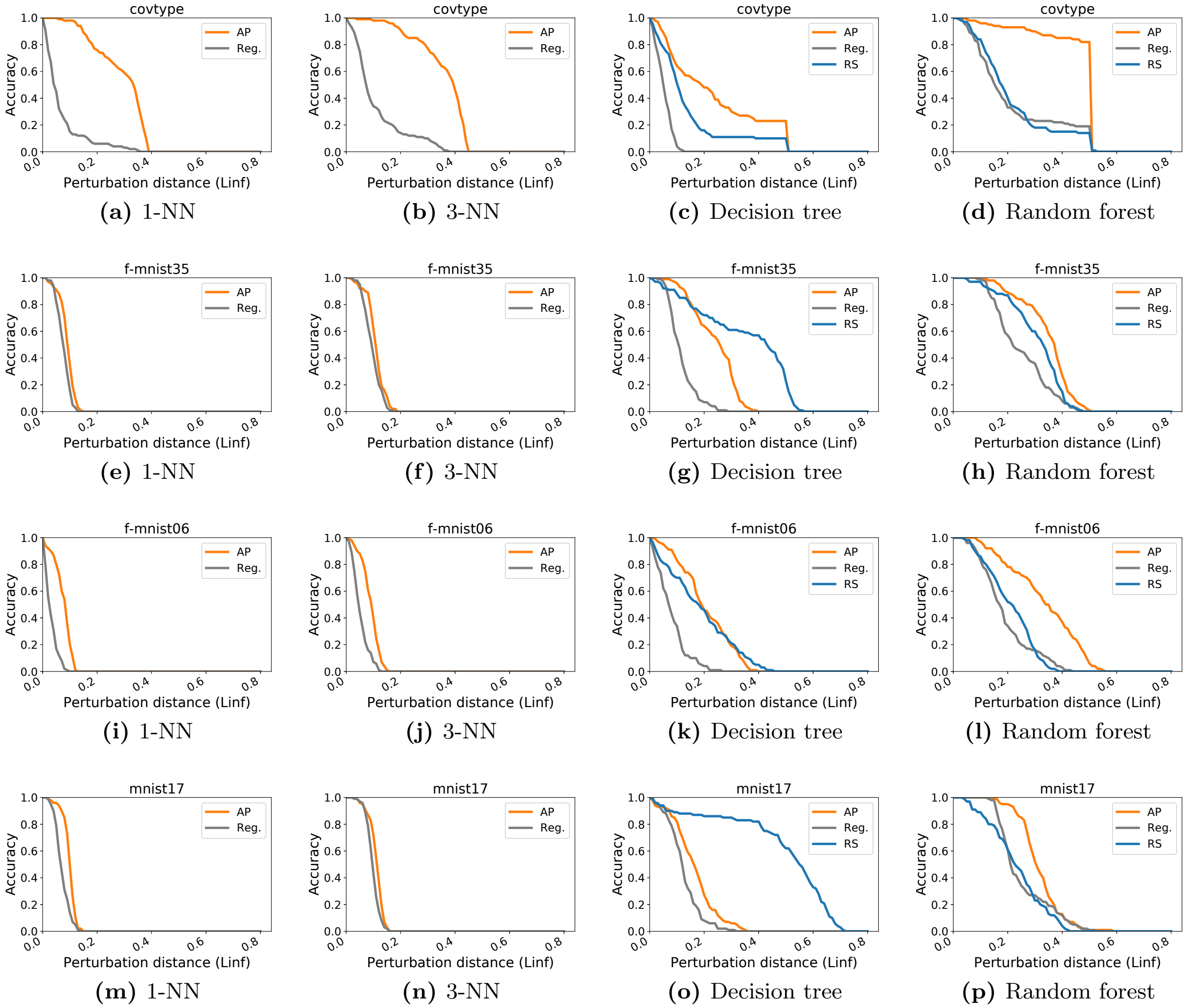
(e) 1-NN (f) 3-NN (g) Decision tree (h) Random forest

(i) 1-NN (j) 3-NN (k) Decision tree (l) Random forest

(m) 1-NN (n) 3-NN (o) Decision tree (p) Random forest

(q) 1-NN (r) 3-NN (s) Decision tree (t) Random forest

Figure 4: The maximum perturbation distance allowed versus the accuracy on the 100 correctly predicted test examples (see Section C.3 for details).



(a) 1-NN **(b)** 3-NN **(c)** Decision tree **(d)** Random forest

(e) 1-NN **(f)** 3-NN **(g)** Decision tree **(h)** Random forest

(i) 1-NN **(j)** 3-NN **(k)** Decision tree **(l)** Random forest

(m) 1-NN **(n)** 3-NN **(o)** Decision tree **(p)** Random forest

Figure 5: The maximum perturbation distance allowed versus the accuracy on the 100 correctly predicted test examples (see Section C.3 for details).