UNIVERSITY OF CALIFORNIA SAN DIEGO

The Primacy of Applied Privacy

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy

in

Computer Science

by

Casey Meehan

Committee in charge:

Professor Kamalika Chaudhuri, Chair
Professor Taylor Berg-Kirkpatrick
Professor Sanjoy Dasgupta
Professor Alon Orlitsky

2023

The Dissertation of Casey Meehan is approved, and it is acceptable in quality and form for publication on microfilm and electronically.

University of California San Diego

2023

DEDICATION

The fact that I have made something I can write a dedication for is owed all to my parents. I cannot imagine following my heart these past few years without their unrelenting support and encouragement.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

VITA

2015	Bachelor of Science, Brown University
2018	Master of Science, Harvard University
2023	Doctor of Philosophy, University of California, San Diego

ABSTRACT OF THE DISSERTATION

The Primacy of Applied Privacy

by

Casey Meehan

Doctor of Philosophy in Computer Science

University of California San Diego, 2023

Professor Kamalika Chaudhuri, Chair

As data collection for machine learning (ML) tasks has become more pervasive, it has also become more heterogeneous: we share our writing, images, voices, and location online every day. Naturally, the associated privacy risks are just as complex and variable. My research advances practical data privacy through two avenues: 1) drafting provable privacy definitions and mechanisms for safely sharing data in different ML domains, and 2) empirically quantifying how ML models memorize their sensitive training data and thereby risk disclosing it. This dissertation details the various data domains/tasks considered, and the corresponding privacy methods proposed.

Chapter 1

When are Non-Parametric Methods Robust?

1.1 Introduction

Recent work has shown that many classifiers tend to be highly non-robust and that small strategic modifications to regular test inputs can cause them to misclassify [1, 2, 3]. Motivated by the use of machine learning in safety-critical applications, this phenomenon has recently received considerable interest; however, what exactly causes this phenomenon – known in the literature as *adversarial examples* – still remains a mystery.

Prior work has looked at three plausible reasons why adversarial examples might exist. The first, of course, is the possibility that in real data distributions, different classes are very close together in space – which does not seem plausible in practice. Another possibility is that classification algorithms may require more data to be robust than to be merely accurate; some prior work [4, 5, 6] suggests that this might be true for certain classifiers or algorithms. Finally, others [7, 8, 5] have suggested that better training algorithms may give rise to more robust classifiers – and that in some cases, finding robust classifiers may even be computationally challenging.

In this work, we consider this problem in the context of general non-parametric classifiers. Contrary to parametrics, non-parametric methods are a form of local classifiers, and include a large number of pattern recognition methods such as nearest neighbors, decision trees, random

forests and kernel classifiers. There is a richly developed statistical theory of non-parametric methods [9], which focuses on accuracy, and provides very general conditions under which these methods converge to the Bayes optimal with growing number of samples. We, in contrast, analyze robustness properties of these methods, and ask instead when they converge to the classifier with the highest astuteness at a desired radius r. Recall that the astuteness of a classifier at radius r is the fraction of points from the distribution on which it is accurate and has the same prediction up to a distance r [5, 4].

We begin by looking at the very simple case when data from different classes is well-separated – by at least a distance $2r$. Although achieving astuteness in this case may appear trivial, we show that even in this highly favorable case, not all non-parametric methods provide robust classifiers – and this even holds for methods that converge to the Bayes optimal in the large sample limit.

This raises the natural question – when do non-parametric methods produce astute classifiers? We next provide conditions under which a non-parametric method converges to the most astute classifier in the large sample limit under well-separated data. Our conditions are analogous to the classical conditions for convergence to the Bayes optimal [9, 10], but a little stronger. We show that nearest neighbors and kernel classifiers whose kernel functions decay fast enough, satisfy these conditions, and hence converge to astute classifiers in the large sample limit. In constrast, histogram classifiers, which do converge to the Bayes optimal in the large sample limit, may not converge to the most astute classifier. This indicates that there may be some non-parametric methods, such as nearest neighbors and kernel classifiers, that are more naturally robust when trained on well-separated data, and some that are not.

What happens when different classes in the data are not as well-separated? For this case, [11] proposes a method called Adversarial Pruning that preprocesses the training data by retaining the maximal set of points such that different classes are distance $\geq 2r$ apart, and then trains a non-parametric method on the pruned data. We next prove that if a non-parametric method has certain properties, then the classifier produced by Adversarial Pruning followed by

the method does converges to the most astute classifier in the large sample limit. We show that again nearest neighbors and kernel classifiers whose kernel functions decay faster than inverse polynomials satisfy these properties. Our results thus complement and build upon the empirical results of [11] by providing a performance guarantee.

What can we conclude about the cause for adversarial examples? Our results seem to indicate that at least for non-parametrics, it is mostly the training algorithms that are responsible. With a few exceptions, decades of prior work in machine learning and pattern recognition has largely focussed on designing training methods that provide increasingly accurate classifiers – perhaps to the detriment of other aspects such as robustness. In this context, our results serve to (a) provide a set of guidelines that can be used for designing non-parametric methods that are robust and accurate on well-separated data and (b) demonstrate that when data is not well-separated, preprocessing through adversarial pruning [11] may be used to ensure convergence to optimally astute solutions in the large sample limit.

1.1.1 Related Work

There is a large body of work on adversarial attacks [12, 13, 14, 15, 1] and defenses [16, 17, 18, 19, 20, 21] in the parametric setting, specifically focusing on neural networks. On the other hand, adversarial examples for nonparametric classifiers have mostly been studied in a much more ad-hoc manner, and to our knowledge, there has been no theoretical investigation into general properties of algorithms that promote robustness in non-parametric classifiers.

For nearest neighbors, there has been some prior work on adversarial attacks [22, 23, 5, 11] as well as defenses. Wang et. al. [5] proposes a defense for 1-NN by pruning the input sample. However, their defense learns a classifier whose robustness regions converge towards those of the Bayes optimal classifier, which itself may potentially have poor robustness properties. Yang et. al. [11] accounts for this problem by proposing the notion of the r-optimal classifier, and propose an algorithm called Adversarial Pruning which can be interpreted as a finite sample approximation to the r-optimal. However, they do not provide formal performance guarantees

for Adversarial Pruning, which we do.

For Kernel methods, Hein and Andriushchenko [16] study lower bounds on the norm of the adversarial manipulation that is required for changing a classifiers output. They specifically study bounds for Kernel Classifiers, and propose an empirically based regularization idea that improves robustness. In this work, we improve the robustness properties of kernel classification through adversarial pruning, and show formal guarantees regarding convergence towards the r-optimal classifier.

For decision trees and random forests, attacks and defenses have been provided by [24, 25, 26]. Again, most of the work here is empirical in nature, and convergence guarantees are not provided.

Pruning has a long history of being applied for improving nearest neighbors [27, 28, 29, 30, 31, 32], but this has been entirely done in the context of generalization, without accounting for robustness. In their work, Yang et. al. empirically show that adversarial pruning can improve robustness for nearest neighbor classifiers. However, they do not provide any formal guarantees for their algorithms. In this work, we prove formal guarantees for *adversarial pruning* in the large sample limit, both for nearest neighbors as well as for more general *weight functions.*

There is a long history of literature for understanding the consistency of Kernel classifiers [33, 10], but this has only been done for accuracy and generalization. In this work, we find different conditions are needed to ensure that a Kernel classifier converges in robustness in addition to accuracy.

1.2 Preliminaries

1.2.1 Setting

We consider binary classification where instances are drawn from a totally bounded metric space $\mathscr{X}$ that is equipped with distance metric denoted by d, and the label space is $\{\pm 1\} = \{-1, +1\}$. The classical goal of classification is to build a highly *accurate* classifier,

which we define as follows.

Definition 1. *(Accuracy) Let $\mathscr{D}$ be a distribution over $\mathscr{X} \times \{\pm 1\}$, and let $f \in \{\pm 1\}^{\mathscr{X}}$ be a classifier. Then the* ***accuracy*** *of f over $\mathscr{D}$, denoted $A(f, \mathscr{D})$, is the fraction of examples $(x, y) \sim \mathscr{D}$ for which $f(x) = y$. Thus*

$$A(f, \mathscr{D}) = P_{(x,y)\sim\mathscr{D}}[f(x) = y].$$

In this work, we consider *robustness* in addition to accuracy. Let $B(x, r)$ denoted the closed ball of radius r centered at x.

Definition 2. *(Robustness) A classifier $f \in \{\pm 1\}^{\mathscr{X}}$ is said to be* ***robust*** *at x with radius r if $f(x) = f(x')$ for all $x' \in B(x, r)$.*

Our goal is to find non-parametric algorithms that output classifiers that are robust, in addition to being accurate. To account for both criteria, we combine them into a notion of *astuteness* [5, 4].

Definition 3. *(Astuteness) A classifier $f \in \{\pm 1\}^{\mathscr{X}}$ is said to be* ***astute*** *at (x, y) with radius r if f is robust at x with radius r and $f(x) = y$. The* ***astuteness*** *of f over $\mathscr{D}$, denoted $A_r(f, \mathscr{D})$, is the fraction of examples $(x, y) \sim \mathscr{D}$ for which f is astute at (x, y) with radius r. Thus*

$$A_r(f, \mathscr{D}) = P_{(x,y)\sim\mathscr{D}}[f(x') = y, \forall x' \in B(x, r)].$$

It is worth noting that $A_0(f, \mathscr{D}) = A(f, \mathscr{D})$, since astuteness with radius 0 is simply the accuracy. For this reason, we will use $A_0(f, \mathscr{D})$ to denote accuracy from this point forwards.

1.2.2 Notions of Consistency

Traditionally, a classification algorithm is said to be consistent if as the sample size grows to infinity, the accuracy of the classifier it learns converges towards the best possible accuracy on the underlying data distribution. We next introduce and formalize an alternative form of consistency, called r-consistency, that applies to robust classifiers.

We begin with a formal definition of the Bayes Optimal Classifier – the most accurate classifier on a distribution – and consistency.

Definition 4. *(Bayes Optimal Classifier) The* ***Bayes Optimal Classifier*** *on a distribution $\mathscr{D}$, denoted by g^*, is defined as follows. Let $\eta(x) = p_{\mathscr{D}}(+1|x)$. Then*

$$g^*(x) = \begin{cases} +1 & \eta(x) \geq 0.5 \\ -1 & \eta(x) < 0.5 \end{cases}$$

It can be shown that g^* achieves the highest accuracy over $\mathscr{D}$ over all classifiers.

Definition 5. *(Consistency) Let M be a classification algorithm over $\mathscr{X} \times \{\pm 1\}$. M is said to be* ***consistent*** *if for any $\mathscr{D}$ over $\mathscr{X} \times \{\pm 1\}$, and any ε, δ over $(0,1)$, there exists N such that for $n \geq N$, with probability $1-\delta$ over $S \sim \mathscr{D}^n$, we have:*

$$A(M(S), \mathscr{D}) \geq A(g^*, \mathscr{D}) - \varepsilon,$$

where g^* is the Bayes optimal classifier for $\mathscr{D}$.

How can we incorporate robustness in addition to accuracy in this notion? A plausible way, as used in [5], is that the classifier should converge towards being astute where the Bayes Optimal classifier is astute. However, the Bayes Optimal classifier is not necessarily the most astute classifier and may even have poor astuteness. To see this, consider the following example.

Example 1

Consider $\mathscr{D}$ over $\mathscr{X} = [0,1]$ such that $\mathscr{D}_{\mathscr{X}}$ is the uniform distribution and

$$p(y=1|x) = \frac{1}{2} + \sin\frac{4\pi x}{r}.$$

For any point x, there exists $x_1, x_2 \in ([x-r, x+r] \cap [0,1])$ such that $p(y=1|x_1) > \frac{1}{2}$ and $p(y=1|x_2) < \frac{1}{2}$. $A_r(g^*, r) = 0$. However, the classifier that always predicts $f(x) = +1$ does

better. It is robust everywhere, and since $P_{(x,y)\sim\mathscr{D}}[y=+1]=\frac{1}{2}$, it follows that $A_r(f,\mathscr{D})=\frac{1}{2}$.

This motivates the notion of the r-optimal classifier, introduced by [11], which is the classifier with maximum astuteness.

Definition 6. *(r-optimal classifier) The* ***r-optimal classifier*** *of a distribution G denoted by g_r^* is the classifier with maximum astuteness. Thus*

$$g_r^* = \underset{f\in\{\pm1\}^{\mathscr{X}}}{\arg\max}\, A_r(f,\mathscr{D}).$$

We let $A_r^*(\mathscr{D})$ denote $A_r(g_r^*,\mathscr{D})$.

Observe that g_r^* is not necessarily unique. To account for this, we use $A_r^*(\mathscr{D})$ in our definition for r-consistency.

Definition 7. *(r-consistent) Let M be a classification algorithm over $\mathscr{X}\times\{\pm1\}$. M is said to be* ***r-consistent*** *if for any $\mathscr{D}$, any $\varepsilon,\delta\in(0,1)$, and $0<\gamma<r$, there exists N such that for $n\geq N$, with probability $1-\delta$ over $S\sim\mathscr{D}^n$,*

$$A_{r-\gamma}(M(S),\mathscr{D})\geq A_r^*(\mathscr{D})-\varepsilon.$$

if the above conditions hold for a specific distribution $\mathscr{D}$, we say that M is r-consistent with respect to $\mathscr{D}$.

Observe that in addition to the usual ε and δ, there is an extra parameter γ which measures the gap in the robustness radius. We may need this parameter as when classes are exactly $2r$ apart, we may not be able to find the exact robust boundary with only finite samples.

Our analysis will be centered around understanding what kinds of algorithms M provide highly astute classifiers for a given radius r. We begin by first considering the special case of

r-separated distributions.

Definition 8. *(r-separated distributions) A distribution $\mathscr{D}$ is said to be* ***r-separated*** *if there exist subsets $T^+, T^- \subset \mathscr{X}$ such that*

1. $\mathbb{P}_{(x,y)\sim\mathscr{D}}[x \in T^y] = 1$.
2. $\forall x_1 \in T^+, \forall x_2 \in T^-$, $d(x_1, x_2) > 2r$.

Observe that if $\mathscr{D}$ is r-separated, $A_r(g_r^*, \mathscr{D}) = 1$.

1.2.3 Non-parametric Classifiers

Many non-parametric algorithms classify points by averaging labels over a local neighborhood from their training data. A very general form of this idea is encapsulated in *weight functions* – which is the general form we will use.

Definition 9. *[9] A* ***weight function*** *W is a non-parametric classifier with the following properties.*

1. *Given input $S = \{(x_1, y_1), (x_2, y_2,), \ldots, (x_n, y_n)\} \sim \mathscr{D}^n$, W constructs functions $w_1^S, \ldots, w_n^S : \mathscr{X} \to [0,1]$ such that for all $x \in \mathscr{X}$, $\sum_1^n w_i^S(x) = 1$. The functions w_i^S are allowed to depend on $x_1, x_2, \ldots x_n$ but must be independent of $y_1, y_2, \ldots, y_n$.*
2. *W has output W_S defined as*

$$W_S(x) = \begin{cases} +1 & \sum_1^n w_i^S(x) y_i > 0 \\ -1 & \sum_1^n w_i^S(x) y_i \leq 0 \end{cases}$$

As a result, $w_i^S(x)$ can be thought of as the weight that (x_i, y_i) has in classifying x.

Weight functions encompass a fairly extensive set of common non-parametric classifiers, which is the motivation for considering them. We now define several common non-parametric algorithms that can be construed as weight functions.

Definition 10. *A **histogram classifier**, H, is a non-parametric classification algorithm over $\mathbb{R}^d \times \{\pm 1\}$ that works as follows. For a distribution $\mathscr{D}$ over $\mathbb{R} \times \{\pm 1\}$, H takes $S = \{(x_i, y_i) : 1 \leq i \leq n\} \sim \mathscr{D}^n$ as input. Let k_i be a sequence with $\lim_{i\to\infty} k_i = \infty$ and $\lim_{i\to\infty} \frac{k_i}{i} = 0$. H constructs a set of hypercubes $C = \{c_1, c_2, \ldots, c_m\}$ as follows:*

1. *Initially $C = \{c\}$, where $S \subset c$.*
2. *For $c \in C$, if c contains more than k_n points of S, then partition c into 2^d equally sized hypercubes, and insert them into C.*
3. *Repeat step 2 until all cubes in C have at most k_n points.*

For $x \in \mathbb{R}$ let $c(x)$ denote the unique cell in C containing x. If $c(x)$ doesn't exist, then $H_S(x) = -1$ by default. Otherwise,

$$H_S(x) = \begin{cases} +1 & \sum_{x_i \in c(x)} y_i > 0 \\ -1 & \sum_{x_i \in c(x)} y_i \leq 0 \end{cases}.$$

Histogram classifiers are weight functions in which all x_i contained within the same cell as x are given the same weight $w_i^S(x)$ in predicting x, while all other x_i are given weight 0.

Definition 11. *A **kernel classifier** is a weight function W over $\mathscr{X} \times \{\pm 1\}$ constructed from function $K : \mathbb{R}^+ \cup \{0\} \to \mathbb{R}^+$ and some sequence $\{h_n\} \subset \mathbb{R}^+$ in the following manner. Given $S = \{(x_i, y_i)\} \sim \mathscr{D}^n$, we have*

$$w_i^S(x) = \frac{K(\frac{d(x,x_i)}{h_n})}{\sum_{j=1}^n K(\frac{d(x,x_j)}{h_n})}.$$

Then, as above, W has output

$$W_S(x) = \begin{cases} +1 & \sum_1^n w_i^S(x) y_i > 0 \\ -1 & \sum_1^n w_i^S(x) y_i \leq 0 \end{cases}$$

Finally, we note that k_n-nearest neighbors is also a weight function; $w_i^S(x) = \frac{1}{k_n}$ if x_i is one of the k_n closest neighbors of x and 0 otherwise.

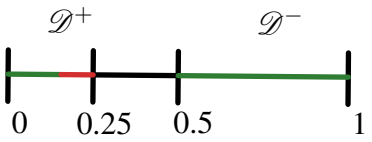

Figure 1.1. H_S is astute in the green region, but not robust in the red region.

1.3 Warm Up: r-separated distributions

We begin by considering the case when the data distribution is r-separated; the more general case is considered in Section 1.4. While classifying r-separated distributions robustly may appear almost trivial, learning an arbitrary classifier does not necessarily produce an astute result. To see this, consider the following example of a histogram classifier – which is known to be consistent.

We let H denote the histogram classifier over $\mathbb{R}$.

Example 2

Consider the data distribution $\mathscr{D} = \mathscr{D}^+ \cup \mathscr{D}^-$ where D^+ is the uniform distribution over $[0, \frac{1}{4})$ and D^- is the uniform distribution over $(\frac{1}{2}, 1]$, $p(+1|x) = 1$ for $x \in \mathscr{D}^+$, and $p(-1|x) = 1$ for $x \in \mathscr{D}^-$.

We make the following observations (refer to Figure 1.1).

1. $\mathscr{D}$ is 0.1-separated, since the supports of $\mathscr{D}^+$ and $\mathscr{D}^-$ have distance $0.25 > 0.2$.

2. If n is sufficiently large, H will construct the cell $[0.25, 0.5)$, which will not be split because it will never contain any points.

3. $H_S(x) = -1$ for $x \in [0.25, 0.5)$.

4. H_S is not astute at $(x, 1)$ for $x \in (0.15, 0.25)$. Thus $A_{0.1}(H_S, \mathscr{D}) = 0.8$.

Example 2 shows that histogram classifiers do not always learn astute classifiers even

when run on r-separated distributions. This motivates the question: which non-parametric classifiers do?

We answer this question in the following theorem, which gives sufficient conditions for a weight function (definition 9) to be r-consistent over an r-separated distribution.

Theorem 12. *Let $\mathscr{D}$ be a distribution over $\mathscr{X} \times \{\pm 1\}$, and let W be a weight function. Let X be a random variable with distribution $\mathscr{D}_{\mathscr{X}}$, and $S = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\} \sim \mathscr{D}^n$. Suppose that for any $0 < a < b$,*

$$\lim_{n\to\infty} \mathbb{E}_{X,S}\big[\sup_{x' \in B(X,a)} \sum_{1}^{n} w_i^S(x') I_{||x_i - x'|| > b}\big] = 0.$$

Then if $\mathscr{D}$ is r-separated, W is r-consistent with respect to $\mathscr{D}$.

First, we compare Theorem 12 to Stone's theorem [10], which gives sufficient conditions for a weight function to be consistent (i.e. converge in accuracy towards the Bayes optimal). For convenience, we include a statement of Stone's theorem.

Theorem 13. *[10] Let W be weight function over $\mathscr{X} \times \{\pm 1\}$. Suppose the following conditions hold for any distribution $\mathscr{D}$ over $\mathscr{X} \times \{\pm 1\}$. Let X be a random variable with distribution $\mathscr{D}_{\mathscr{X}}$, and $S = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\} \sim \mathscr{D}^n$. All expectations are taken over X and S.*

1. *There is a constant c such that, for every nonnegative measurable function f satisfying $\mathbb{E}[f(X)] < \infty$,*

$$\mathbb{E}[\sum_{1}^{n} w_i^S(X) f(x_i)] \leq c\mathbb{E}[f(x)].$$

2. *For all $a > 0$,*

$$\lim_{n\to\infty} \mathbb{E}[\sum_{1}^{n} w_i^S(x) I_{||x_i - X|| > a}] = 0,$$

   where $I_{||x_i - X|| > a}$ is an indicator variable.

3.

$$\lim_{n\to\infty} \mathbb{E}[\max_{1 \leq i \leq n} w_i^S(X)] = 0.$$

Then W is consistent.

There are two main differences between Theorem 12 and Stone's theorem.

1. Conditions 1. and 3. of Stone's theorem are no longer necessary. This is because r-separated distributions are well-separated and thus have simpler conditions for consistency. In fact, a slight modification of the arguments of [10] shows that for r-separated distributions, condition 2. alone is sufficient for consistency.

2. Condition 2. is strengthened. Instead of requiring the weight of x_i's outside of a given radius to go to 0 for $X \sim \mathscr{D}$, we require the same to *uniformly* hold over a ball centered at X.

Theorem 12 provides a general condition that allows us to verify the r-consistency of non-parametric methods. We now show below that two common non-parametric algorithms – k_n-nearest neighbors and kernel classifiers with rapidly decaying kernel functions – satisfy the conditions of Theorem 12.

Corollary 14. *Let $\mathscr{D}$ be any r-separated distribution. Let k_n be any sequence such that $\lim_{n\to\infty} \frac{k_n}{n} = 0$, and let M be the k_n-nearest neighbors classifier on a sample $S \sim \mathscr{D}^n$. Then M is r-consistent with respect to $\mathscr{D}$.*

Remarks:

1. Because the data distribution is r-separated, $k_n = 1$ will be r-consistent. Also observe that for r-separated distributions, $k_n = 1$ will converge towards the Bayes Optimal classifier.

2. In general, M converges towards the Bayes Optimal classifier provided that $k_n \to \infty$ in addition to $k_n/n \to 0$. This condition is not necessary for r-consistency– because the distribution is r-separated.

We next show that kernel classifiers are also r-consistent on r-separated data distributions, provided the kernel function decreases rapidly enough.

Corollary 15. *Let W be a kernel classifier over $\mathscr{X} \times \{\pm 1\}$ constructed from K and h_n. Suppose the following properties hold for K and h_n.*

1. *For any $c > 1$, $\lim_{x\to\infty} \frac{K(cx)}{K(x)} = 0$.*
2. *$\lim_{n\to\infty} h_n = 0$.*

If $\mathscr{D}$ is an r-separated distribution over $\mathscr{X} \times \{\pm 1\}$, then W is r-consistent with respect to $\mathscr{D}$.

Observe that Condition 1. is satisfied for any $K(x)$ that decreases more rapidly than an inverse polynomial – and is hence satisfied by most popular kernels like the Gaussian kernel. Is the condition on K in Corollary 15 necessary? The following example illustrates that a kernel classifier with any arbitrary K is not necessarily r-consistent. This indicates that some sort of condition needs to be imposed on K to ensure r-consistency; finding a tight necessary condition however is left for future work.

Example 3

Let $\mathscr{X} = [-1,1]$ and let $\mathscr{D}$ be a distribution with $p_{\mathscr{D}}(-1,-1) = 0.1$ and $p_{\mathscr{D}}(1,1) = 0.9$. Clearly, $\mathscr{D}$ is 0.3-separated. Let $K(x) = e^{-\min(|x|,0.2)^2}$. Let h_n be any sequence with $\lim_{n\to\infty} h_n = 0$ and $\lim_{n\to\infty} nh_n = \infty$. Let W be the weight classifier with input $S = \{(x_1,y_1),(x_2,y_2),\ldots,(x_n,y_n)\}$ such that

$$w_i^S(x) = \frac{K(\frac{|x-x_i|}{h_n})}{\sum_{j=1}^n K(\frac{|x-x_j|}{h_n})}.$$

W can be shown to satisfy all the conditions of Theorem 13 (the proof is analogous to the case for a Gaussian Classifier), and is therefore consistent. However, W does not learn a robust classifier on $\mathscr{D}$ for $r = 0.3$.

Consider $x = -0.7$. For any $\{(x_1,y_1),(x_2,y_2),\ldots,(x_n,y_n)\} \sim \mathscr{D}^n$, all x_i will either be -1 or 1. Therefore, since $K(|x-(-1)|) = K(|x-1|)$, it follows that $w_i^S(x) = \frac{1}{n}$ for all $1 \le i \le n$. Since $x_i = 1$ with probability 0.9, it follows that with high probability x will be classified as 1 which means that f, the output of W, is not robust at $x = -1$. Thus f has astuteness at most 0.9 which means that W is *not* r-consistent for $r = 0.3$.

1.4 General Distributions

We next consider more general data distributions, where data from different classes may be close together in space, and may even overlap. Observe that unlike the r-separated case, here there may be no classifier with astuteness one. Thus, a natural question is: what does the optimally astute classifier look like, and how can we build non-parametric classifiers to this limit?

1.4.1 The r-Optimal Classifier and Adversarial Pruning

[11] propose a large-sample limit – called the r-optimal – and show that it is analogous to the Bayes Optimal classifier for robustness. More specifically, given a data distribution D, to find the r-optimal classifier, we solve the following optimization problem.

$$\begin{aligned} \max_{S_{+1},S_{-1}} & \int_{x\in S_{+1}} p(y=+1|x)d\mu_{\mathscr{D}}(x)+ \\ & \int_{x\in S_{-1}} p(y=-1|x)d\mu_{\mathscr{D}}(x) \\ & \text{subject to } d(S_{+1},S_{-1})>2r \end{aligned} \tag{1.1}$$

Then, the r-optimal classifier is defined as follows.

Definition 16. *[11] Fix $r,\mathscr{D}$. Let S^*_{+1} and S^*_{-1} be any optimizers of (1.1). Then the r-optimal classifier, g^*_r is any classifier such that $g^*_r(x)=j$ whenever $d(S^*_j,x)\leq r$.*

[11] show that the r-optimal classifier achieves the optimal astuteness – out of all classifiers on the data distribution $\mathscr{D}$; hence, it is a robustness analogue to the Bayes Optimal Classifier. Therefore, for general distributions, the goal in robust classification is to find non-parametric algorithms that output classifiers that converge towards g^*_r.

To find robust classifiers, [11] propose Adversarial Pruning – a defense method that preprocesses the training data by making it better separated. More specifically, Adversarial

Pruning takes as input a training dataset S and a radius r, and finds the largest subset of the training set where differently labeled points are at least distance $2r$ apart.

Definition 17. *A set $S_r \subset \mathscr{X} \times \{\pm 1\}$ is said to be **r-separated** if for all $(x_1, y_1), (x_2, y_2) \in S_r$, if $y_1 \neq y_2$, then $d(x_1, x_2) > 2r$. To **adversarially prune** a set S is to return its largest r-separated subset. We let $AdvPrun(S, r)$ denote the result of adversarially pruning S.*

Once an r-separated subset S_r of the training set is found, a standard non-parametric method is trained on S_r. While [11] show good empirical performance of such algorithms, no formal guarantees are provided. We next formally characterize when adversarial pruning followed by a non-parametric method results in a classifier that is provably r-consistent.

Specifically, we consider analyzing the general algorithm provided in Algorithm 1.

Algorithm 1: RobustNonPar

Input: $S \sim \mathscr{D}^n$, weight function W, robustness radius r;

$S_r \leftarrow AdvPrun(S, r)$;

Output: W_{S_r};

1.4.2 Convergence Guarantees

We begin with some notation. For any weight function W and radius $r > 0$, we let $RobustNonPar(W, r)$ represent the weight function that outputs weights for $S \sim \mathscr{D}^n$ according to $RobustNonPar(S, W, r)$. In particular, this can be used to convert any weight function algorithm into a new weight function which takes robustness into account. A natural question is, for which weight functions W is $RobustNonPar(W, r)$ r-consistent? Our next theorem provides sufficient conditions for this.

Theorem 18. *Let W be a weight function over $\mathscr{X} \times \{\pm 1\}$, and let $\mathscr{D}$ be a distribution over $\mathscr{X} \times \{\pm 1\}$. Fix $r > 0$. Let $S_r = AdvPrun(S, r)$. For convenience, relabel x_i, y_i so that $S_r =$*

$\{(x_1,y_1),(x_2,y_2),\ldots,(x_m,y_m)\}$. *Suppose that for any* $0<a<b$,

$$\lim_{n\to\infty}\mathbb{E}_{S\sim\mathscr{D}^n}\Big[\frac{1}{m}\sum_{i=1}^{m}\sup_{x\in B(x_i,a)}\sum_{j=1}^{m}w_j^{S_r}(x)I_{||x_j-x||>b}\Big]=0.$$

Then RobustNonPar(W,r) *is r-consistent with respect to* $\mathscr{D}$.

Remark:

There are two important differences between the conditions in Theorem 18 and Theorem 12.

1. We replace S with S_r.

2. The expectation over $X\sim\mathscr{D}_{\mathscr{X}}$ is replaced with an average over $\{x_1,x_2,\ldots,x_m\}$. The intuition here is that we are replacing $\mathscr{D}$ with a uniform distribution over S_r. While $\mathscr{D}$ may not be r-separated, the uniform distribution over S_r is, and represents the region of points where our classifier is astute.

A natural question is what satisfies the conditions in Theorem 18. We next show that k_n-nearest neighbors and kernel classifiers with rapidly decaying kernel functions continue to satisfy the conditions in Theorem 18; this means that these classifiers, when combined with Adversarial Pruning, will converge to r-optimal classifiers in the large sample limit.

Corollary 19. *Let* k_n *be a sequence with* $\lim_{n\to\infty}\frac{k_n}{n}=0$*, and let M denote the* k_n*-nearest neighbor algorithm. Then for any* $r>0$*, RobustNonPar*(M,r) *is r-consistent.*

Remark:

Corollary 19 gives a formal guarantee in the large sample limit for the modified nearest-neighbor algorithm proposed by [11].

Corollary 20. *Let W be a kernel classifier over* $\mathscr{X}\times\{\pm1\}$ *constructed from K and* h_n*. Suppose the following properties hold for K and* h_n*.*

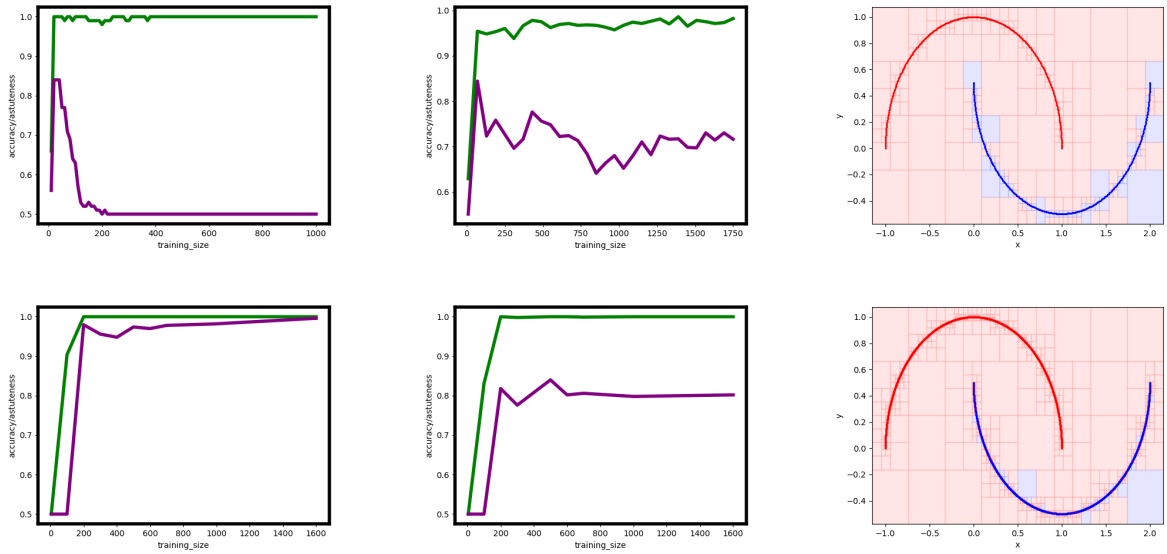

Figure 1.2. Empirical accuracy/astuteness of different classifiers as a function of training sample size. Accuracy is shown in green, astuteness in purple. Left : Noiseless Setting. Right: Noisy Setting. Top Row: Histogram Classifier, Bottom Row: 1-Nearest Neighbor

1. *For any* $c > 1$, $\lim_{x\to\infty} \frac{K(cx)}{K(x)} = 0$.

2. $\lim_{n\to\infty} h_n = 0$.

Then for any $r > 0$, $RobustNonPar(W, r)$ *is r-consistent.*

Observe again that Condition 1. is satisfied by any K that decreases more rapidly than an inverse polynomial kernel; it is thus satisfied by most popular kernels, such as the Gaussian kernel.

1.5 Validation

Our theoretical results are, by nature, large sample; we next validate how well they apply to the finite sample case by trying them out on a simple example. In particular, we ask the following question:

> How does the robustness of non-parametric classifiers change with increasing sample size?

This question is considered in the context of two simple non-parametric classifiers – one nearest neighbor (which is guaranteed to be r-consistent) and histograms (which is not). To be able to measure performance with increasing data size, we look at a simple synthetic dataset – the Half Moons.

1.5.1 Experimental Setup

Classifiers and Dataset.

We consider two different classification algorithms – one nearest neighbor (NN) and a Histogram Classifier (HC). We use the Halfmoon dataset with two settings of the gaussian noise parameter σ, $\sigma = 0$ (Noiseless) and $\sigma = 0.08$ (Noisy). For the Noiseless setting, observe that the data is already 0.1-separated; for the Noisy setting, we use Adversarial Pruning (Algorithm 1) with parameter $r = 0.1$ for both classification methods.

Performance Measure.

We evaluate robustness with respect to the ℓ_∞ metric, that is commonly used in the adversarial examples literature. Specifically, for each classifier, we calculate the *empirical astuteness*, which is the fraction of test examples on which it is astute.

Observe that computing the empirical astuteness of a classifier around an input x amounts to finding the adversarial example that is *closest to* x according to the ℓ_∞ norm. For the 1-nearest neighbor, we do this using the optimal attack algorithm proposed by Yang et. al. [11]. For the histogram classifier, we use the optimal attack framework proposed by [11], and show that the structure of the classifier can be exploited to solve the convex program efficiently. Details are in Appendix C.

We use an attack radius of $r = 0.1$ for the Noiseless setting, and $r = 0.09$ for the Noisy setting. For all classification algorithms, we plot the empirical astuteness as a function of the training set size. As a baseline, we also plot their standard accuracy on the test set.

1.5.2 Results

The results are presented in Figure 1.2; the left two panels are for the Noiseless setting while the two center ones are for the Noisy setting.

The results show that as predicted by our theory, for the Noiseless setting, the empirical astuteness of nearest neighbors converges to 1 as the training set grows. For Histogram Classifiers, the astuteness converges to 0.5 – indicating that the classifier may grow less and less astute with higher sample size even for well-separated data. This is plausibly because the cell size induced by the histogram grows smaller with growing training data; thus, the classifier that outputs the default label -1 in empty cells is incorrect on adversarial examples that are close to a point with $+1$ label, but belongs to a different, empty cell. The rightmost panels in Figure 1.2 provide a visual illustration of this process.

For the Noisy setting, the empirical astuteness of adversarial pruning followed by nearest neighbors converges to 0.8. For histograms with adversarial pruning, the astuteness converges to 0.7, which is higher than the noiseless case but still clearly sub-optimal.

1.5.3 Discussion

Our results show that even though our theory is asymptotic, our predictions continue to be relevant in finite sample regimes. In particular, on well-separated data, nearest neighbors that we theoretically predict to be intrinsically robust is robust; histogram classifiers, which do not satisfy the conditions in Theorem 12 are not. Our predictions continue to hold for data that is not well-separated. Nearest neighbors coupled with Adversarial Pruning continues to be robust with growing sample size, while histograms continue to be non-robust. Thus our theory is confirmed by practice.

1.6 Conclusion

In conclusion, we rigorously analyze when non-parametric methods provide classifiers that are robust in the large sample limit. We provide a general condition that characterizes when non-parametric methods are robust on well-separated data, and show that Adversarial Pruning of [11] works on data that is not well-separated.

Our results serve to provide a set of guidelines that can be used for designing non-parametric methods that are robust and accurate on well-separated data; additionally, we demonstrate that when data is not well-separated, preprocessing by adversarial pruning [11] does lead to optimally astute solutions in the large sample limit.

Chapter 2

Data-Copying in Generative Models: A Formal Framework

2.1 Introduction

Deep generative models have shown impressive performance. However, given how large, diverse, and uncurated their training sets are, a big question is whether, how often, and how closely they are memorizing their training data. This question has been of considerable interest in generative modeling [34, 35] as well as supervised learning [36, 37]. However, a clean and formal definition of memorization that captures the numerous complex aspects of the problem, particularly in the context of continuous data such as images, has largely been elusive.

For generative models, [38] proposed a formal definition of memorization called "data-copying", and showed that it was orthogonal to various prior notions of overfitting such as mode collapse [39], mode dropping [40], and precision-recall [41]. Specifically, their definition looks at three datasets – a training set, a set of generated example, and an independent test set. Data-copying happens when the training points are considerably closer on average to the generated data points than to an independently drawn test sample. Otherwise, if the training points are further on average to the generated points than test, then there is underfitting. They proposed a three sample test to detect this kind of data-copying, and empirically showed that their test had good performance.

However, despite its practical success, this method may not capture even blatant cases of

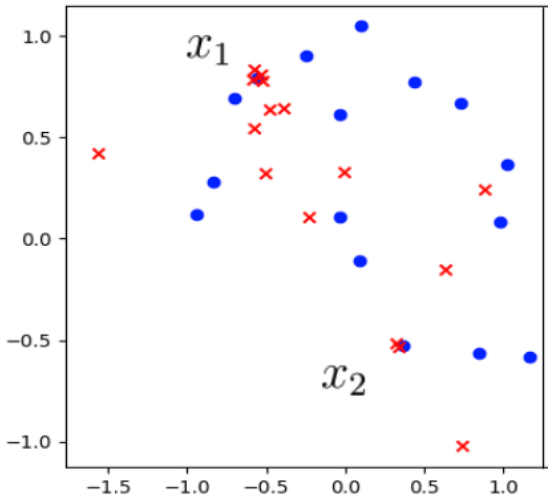

Figure 2.1. In this figure, the blue points are sampled from the halfmoons dataset (with Gaussian noise). The red points are sampled from a generated distribution that is a mixture of (40 %) blatant data copier (that outputs a random subset of the training set), and (60 %) a noisy underfit version of halfmoons. Although the generated distribution is clearly doing some form of copying at points x_1 and x_2, detecting this is challenging because of the canceling effect of the underfit points.

memorization. To see this, consider the example illustrated in Figure 2.1, in which a generated model for the halfmoons dataset outputs one of its training points with probability 0.4, and otherwise outputs a random point from an underfit distribution. When the test of [38] is applied to this distribution, it is unable to detect any form of data copying; the generated samples drawn from the underfit distribution are sufficient to cancel out the effect of the memorized examples. Nevertheless, this generative model is clearly an egregious memorizer as shown in points x_1 and x_2 of Figure 2.1.

This example suggests a notion of *point-wise* data copying, where a model q can be thought of as copying a given training point x. Such a notion would be able to detect q's behavior nearby x_1 and x_2 regardless of the confounding samples that appear at a global level. This stands in contrast to the more global distance based approach taken in Meehan et. al. which is unable to detect such instances. Motivated by this, we propose an alternative point-by-point approach to defining data-copying.

We say that a generative model q data-copies an individual training point, x, if it has an

unusually high concentration in a small area centered at x. Intuitively, this implies q is highly likely to output examples that are very similar to x. In the example above, this definition would flag q as copying x_1 and x_2.

To parlay this definition into a global measure of data-copying, we define the overall *data-copying rate* as the total fraction of examples from q that are copied from some training example. In the example above, this rate is 40%, as this is the fraction of examples that are blatant copies of the training data.

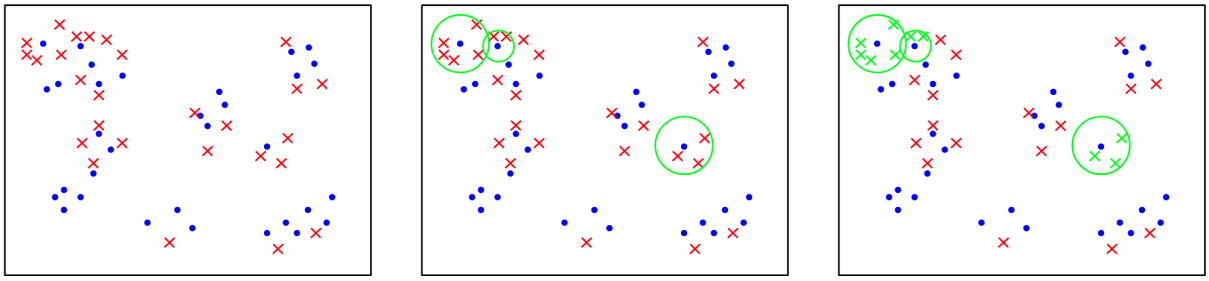

Figure 2.2. In the three panels above, the blue points are a training sample from p, and the red points are generated examples from q. In the middle panel, we highlight in green regions that are defined to be *data-copying regions*, as q overrepresents them with comparison to p. In the third panel, we then color all points from q that are considered to be copied green.

Next, we consider how to detect data-copying according to this definition. To this end, we provide an algorithm, Data_Copy_Detect, that outputs an estimate for the overall data-copying rate. We then show that under a natural smoothness assumption on the data distribution, which we call *regularity*, Data_Copy_Detect is able to guarantee an accurate estimate of the total data-copying rate. We then give an upper bound on the amount of data needed for doing so.

We complement our algorithm with a lower bound on the minimum amount of a data needed for data-copying detection. Our lower bound also implies that some sort of smoothness condition (such as regularity) is necessary for guaranteed data-copying detection; otherwise, the required amount of data can be driven arbitrarily high.

2.1.1 Related Work

Recently, understanding failure modes for generative models has been an important growing body of work e.g. [42, 43, 41]. However, much of this work has been focused on other forms of overfitting, such as mode dropping or mode collapse.

A more related notion of overfitting is *memorization* [34, 35, 44], in which a model outputs exact copies of its training data. This has been studied in both supervised [36, 37] and unsupervised [45, 46] contexts. Memorization has also been considered in language generation models [47].

The first work to explicitly consider the more general notion of *data-copying* is [38], which gives a three sample test for data-copy detection. We include an empirical comparison between our methods in Section 2.5.2, where we demonstrate that ours is able to capture certain forms of data-copying that theirs is not.

Finally, we note that this work focuses on detecting natural forms of memorization or data-copying, that likely arises out of poor generalization, and is not concerned with detecting *adversarial* memorization or prompting, such as in [48], that are designed to obtain sensitive information about the training set. This is reflected in our definition and detection algorithm which look at the specific generative model, and not the algorithm that trains it. Perhaps the best approach to prevent adversarial memorization is training the model with differential privacy [49], which ensures that the model does not change much when one training sample changes. However such solutions come at an utility cost.

2.2 A Formal Definition of Data-Copying

We begin with the following question: what does it mean for a generated distribution q to copy a single training example x? Intuitively, this means that q is guilty of overfitting x in some way, and consequently produces examples that are very similar to it.

However, determining what constitutes a 'very similar' generated example must be done

contextually. Otherwise the original data distribution, p, may itself be considered a copier, as it will output points nearby x with some frequency depending on its density at x. Thus, we posit that q data copies training point x if it has a significantly higher concentration nearby x than p does. We express this in the following definition.

Definition 2.2.1. *Let p be a data distribution, $S \sim p^n$ a training sample, and q be a generated distribution trained on S. Let $x \in S$ be a training point, and let $\lambda > 1$ and $0 < \gamma < 1$ be constants. A generated example $x' \sim q$ is said to be a (λ, γ)-**copy** of x if there exists a ball B centered at x (i.e. $\{x' : ||x' - x|| \leq r\}$) such that following hold:*

- $x' \in B$.
- $q(B) \geq \lambda p(B)$
- $p(B) \leq \gamma$

Here $q(B)$ and $p(B)$ denote the probability mass assigned to B by p and q respectively.

The parameters λ and γ are user chosen parameters that characterize data-copying. λ represents the rate at which q must overrepresent points close to x, with higher values of λ corresponding to more egregious examples of data-copying. γ represents the maximum size (by probability mass) of a region that is considered to be data-copying – the ball B represents all points that are "copies" of x. Together, λ and γ serve as practitioner controlled knobs that characterize data-copying about x.

Our definition is illustrated in Figure 2.2 – the training data is shown in blue, and generated samples are shown in red. For each training point, we highlight a region (in green) about that point in which the red density is much higher than the blue density, thus constituting data-copying. The intuition for this is that the red points within any ball can be thought of as "copies" of the blue point centered in the ball.

Having defined data-copying with respect to a single training example, we can naturally extend this notion to the entire training dataset. We say that $x' \sim q$ is copied from training set S

if x' is a (λ, γ)-copy of some training example $x \in S$. We then define the *data-copy rate* of q as the fraction of examples it generates that are copied from S. Formally, we have the following:

Definition 2.2.2. *Let p, S, q, λ, and γ be as defined in Definition 2.2.1. Then the **data-copy rate**, $cr(q, \lambda, \gamma)$ of q (with respect to p, S) is the fraction of examples from q that are (λ, γ)-copied. That is,*

$$cr(q, \lambda, \gamma) = \Pr_{x' \sim q}[q \ (\lambda, \gamma)\text{-copies } x'].$$

In cases where λ, γ are fixed, we use $cr_q = cr(q, \lambda, \gamma)$ to denote the data-copy rate.

Despite its seeming global nature, cr_q is simply an aggregation of the point by point data-copying done by q over its entire training set. As we will later see, estimating cr_q is often reduced to determining which subset of the training data q copies.

2.2.1 Examples of data-copying

We now give several examples illustrating our definitions. In all cases, we let p be a data distribution, S, a training sample from p, and q, a generated distribution that is trained over S.

The uniform distribution over S:

In this example, q is an egregious data copier that memorizes its training set and randomly outputs a training point. This can be considered as the canonical *worst* data copier. This is reflected in the value of cr_q – if p is a continuous distribution with finite probability density, then for any $x \in S$, there exists a ball B centered at x for which $q(B) >> p(B)$. It follows that q (λ, γ)-copies x for all $x \in S$ which implies that $cr_q = 1$.

The perfect generative model: $q = p$:

In this case, $q(B) = p(B)$ for all balls, B, which implies that q does not perform any data-copying (Definition 2.2.1). It follows that $cr_q = 0$, matching the intuition that q does not data-copy at all.

Kernel Density Estimators:

Finally, we consider a more general situation, where q is trained by a *kernel density estimator* (KDE) over $S \sim p^n$. Recall that a kernel density estimator outputs a generated distribution, q, with pdf defined by

$$q(x) = \frac{1}{n\sigma_n} \sum_{x_i \in S} K\left(\frac{x - x_i}{\sigma_n}\right).$$

Here, K is a kernel similarity function, and σ_n is the bandwidth parameter. It is known that for $\sigma_n = O(n^{-1/5})$, q converges towards p for sufficiently well behaved probability distributions.

Despite this guarantee, KDEs intuitively appear to perform some form of data-copying – after all they implicitly include each training point in memory as it forms a portion of their outputted pdf. However, recall that our main focus is in understanding *overfitting* due to data-copying. That is, we view data-copying as a function of the outputted pdf, q, and not of the training algorithm used.

To this end, for KDEs the question of data-copying reduces to the question of whether q overrepresents areas around its training points. As one would expect, this occurs *before* we reach the large sample limit. This is expressed in the following theorem.

Theorem 2.2.3. *Let $1 < \lambda$ and $\gamma > 0$. Let σ_n be a sequence of bandwidths and K be any regular kernel function. For any $n > 0$ there exists a probability distribution π with full support over $\mathbb{R}^d$ such that with probability at least $\frac{1}{3}$ over $S \sim \pi^n$, a KDE trained with bandwidth σ_n and kernel function K has data-copy rate $cr_q \geq \frac{1}{10}$.*

This theorem completes the picture for KDEs with regards to data-copying – when n is too low, it is possible for the KDE to have a significant amount of data-copying, but as n continues to grow, this is eventually smoothed out.

The Halfmoons dataset

Returning to the example given in Figure 2.1, observe that our definition exactly captures the notion of data-copying that occurs at points x_1 and x_2. For even strict choices of λ and γ, Definition 2.2.1 indicates that the red distribution copies both x_1 and x_2. Furthermore, the data-copy rate, cr_q, is 40% by construction, as this is the proportion of points that are outputted nearby x_1 and x_2.

2.2.2 Limitations of our definition

Definition 2.2.1 implicitly assumes that the goal of the generator is to output a distribution q that approaches p in a mathematical sense; a perfect generator would output q so that $q(M) = p(M)$ for all measurable sets. In particular, instances where q outputs examples that are far away from the training data are considered completely irrelevant in our definition.

This restriction prevents our definition from capturing instances in which q memorizes its training data and then applies some sort of transformation to it. For example, consider an image generator that applies a color filter to its training data. This would not be considered a data-copier as its output would be quite far from the training data in pixel space. Nevertheless, such a generated distribution can be very reasonably considered as an egregious data copier, and a cursory investigation between its training data and its outputs would reveal as much.

The key difference in this example is that the generative algorithm is no longer trying to closely approximate p with q – it is rather trying to do so in some kind of transformed space. Capturing such interactions is beyond the scope of our paper, and we firmly restrict ourselves to the case where a generator is evaluated based on how close q is to p with respect to their measures over the input space.

2.3 Detecting data-copying

Having defined cr_q, we now turn our attention towards *estimating it.* To formalize this problem, we will require a few definitions. We begin by defining a generative algorithm.

Definition 2.3.1. *A **generative algorithm**, A, is a potentially randomized algorithm that outputs a distribution q over $\mathbb{R}^d$ given an input of training points, $S \subset \mathbb{R}^d$. We denote this relationship by $q \sim A(S)$.*

This paradigm captures most typical generative algorithms including both non-parametric methods such as KDEs and parametric methods such as variational autoencoders.

As an important distinction, in this work we define data-copying as a property of the generated distribution, q, rather than the generative algorithm, A. This is reflected in our definition which is given solely with respect to q, S, and p. For the purposes of this paper, A can be considered an arbitrary process that takes S and outputs a distribution q. We include it in our definitions to emphasize that while S is an i.i.d sample from p, it is *not* independent from q.

Next, we define a *data-copying detector* as an algorithm that estimates cr_q based on access to the training sample, S, along with the ability to draw any number of samples from q. The latter assumption is quite typical as sampling from q is a purely computational operation. We do not assume any access to p beyond the training sample S. Formally, we have the following definition.

Definition 2.3.2. *A **data-copying detector** is an algorithm D that takes as input a training sample, $S \sim p^n$, and access to a sampling oracle for $q \sim A(S)$ (where A is an arbitrary generative algorithm). D then outputs an estimate, $D(S,q) = \hat{cr}_q$, for the data-copy rate of q.*

Naturally, we assume D has access to $\lambda, \gamma > 0$ (as these are practitioner chosen values), and by convention don't include λ, γ as formal inputs into D.

The goal of a data-copying detector is to provide accurate estimates for cr_q. However, the precise definition of cr_q poses an issue: data-copy rates for varying values of λ and γ can vastly differ. This is because λ, γ act as thresholds with everything above the threshold being counted, and everything below it being discarded. Since λ, γ cannot be perfectly accounted for, we will require some tolerance in dealing with them. This motivates the following.

Definition 2.3.3. *Let $0 < \varepsilon$ be a tolerance parameter. Then the **approximate data-copy rates**, $cr_q^{-\varepsilon}$ and cr_q^{ε}, are defined as the values of cr_q when the parameters (λ, γ) are shifted by a factor of $(1+\varepsilon)$ to respectively decrease and increase the copy rate. That is,*

$$cr_q^{-\varepsilon} = cr\left(q, \lambda(1+\varepsilon), \gamma(1+\varepsilon)^{-1}\right),$$

$$cr_q^{\varepsilon} = cr\left(q, \lambda(1+\varepsilon)^{-1}, \gamma(1+\varepsilon)\right).$$

The shifts in λ and γ are chosen as above because increasing λ and decreasing γ both reduce cr_q seeing as both result in more restrictive conditions for what qualifies as data-copying. Conversely, decreasing λ and increasing γ has the opposite effect. It follows that

$$cr_q^{-\varepsilon} \leq cr_q \leq cr_q^{\varepsilon},$$

meaning that $cr_q^{-\varepsilon}$ and cr_q^{ε} are lower and upper bounds on cr_q.

In the context of data-copying detection, the goal is now to estimate cr_q in comparison to $cr_q^{\pm\varepsilon}$. We formalize this by defining *sample complexity* of a data-copying detector as the amount of data needed for accurate estimation of cr_q.

Definition 2.3.4. *Let D be a data-copying detector and p be a data distribution. Let $\varepsilon, \delta > 0$ be standard tolerance parameters. Then D has **sample complexity**, $m_p(\varepsilon, \delta)$, with respect to p if for all $n \geq m_p(\varepsilon, \delta)$, $\lambda > 1$, $0 < \gamma < 1$, and generative algorithms A, with probability at least $1 - \delta$ over $S \sim p^n$ and $q \sim A(S)$,*

$$cr_q^{-\varepsilon} - \varepsilon \leq D(S, q) \leq cr_q^{\varepsilon} + \varepsilon.$$

Here the parameter ε takes on a somewhat expanded as it is both used to additively bound our estimation of cr_q and to multiplicatively bound λ and γ.

Observe that there is no mention of the number of calls that D makes to its sampling

oracle for q. This is because samples from q are viewed as *purely computational*, as they don't require any natural data source. In most cases, q is simply some type of generative model (such as a VAE or a GAN), and thus sampling from q is a matter of running the corresponding neural network.

2.4 Regular Distributions

Our definition of data-copying (Definition 2.2.1) motivates a straightforward point by point method for data-copying detection, in which for every training point, x_i, we compute the largest ball B_i centered at x_i for which $q(B_i) \geq \lambda\, p(B_i)$ and $p(B_i) \leq \gamma$. Assuming we compute these balls accurately, we can then query samples from q to estimate the total rate at which q outputs within those balls, giving us our estimate of cr_q.

The key ingredient necessary for this idea to work is to be able to reliably estimate the masses, $q(B)$ and $p(B)$ for any ball in $\mathbb{R}^d$. The standard approach to doing this is through *uniform convergence*, in which large samples of points are drawn from p and q (in p's case we use S), and then the mass of a ball is estimated by counting the proportion of sampled points within it. For balls with a sufficient number of points (typically $O(d \log n)$), standard uniform convergence arguments show that these estimates are reliable.

However, this method has a major pitfall for our purpose – in most cases the balls B_i will be very small because data-copying intrinsically deals with points that are very close to a given training point. While one might hope that we can simply ignore all balls below a certain threshold, this does not work either, as the sheer number of balls being considered means that their union could be highly non-trivial.

To circumvent this issue, we will introduce an interpolation technique that estimates the probability mass of a small ball by scaling down the mass of a sufficiently large ball with the same center. While obtaining a general guarantee is impossible – there exist pathological distributions that drastically change their behavior at small scales – it turns out there is a relatively natural

condition under which such interpolation will work. We refer to this condition as *regularity,* which is defined as follows.

Definition 2.4.1. *Let $k > 0$ be an integer. A probability distribution p is k-**regular** the following holds. For all $\varepsilon > 0$, there exists a constant $0 < p_\varepsilon \leq 1$ such that for all x in the support of p, if $0 < s < r$ satisfies that $p(B(x,r)) \leq p_\varepsilon$, then*

$$\left(1+\frac{\varepsilon}{3}\right)^{-1} \frac{r^k}{s^k} \leq \frac{p(B(x,r))}{p(B(x,s))} \leq \left(1+\frac{\varepsilon}{3}\right) \frac{r^k}{s^k}.$$

*Finally, a distribution is **regular** if it is k-regular for some integer $k > 0$.*

Here we let $B(x,r) = \{x' : ||x - x'|| \leq r\}$ denote the closed ℓ_2 ball centered at x with radius r.

The main intuition for a k-regular distribution is that at a sufficiently small scale, its probability mass scales with distance according to a power law, determined by k. The parameter k dictates how the probability density behaves with respect to the distance scale. In most common examples, k will equal the *intrinsic dimension* of p.

As a technical note, we use an error factor of $\frac{\varepsilon}{3}$ instead of ε for technical details that enable cleaner statements and proofs in our results (presented later).

2.4.1 Distributions with Manifold Support

We now give an important class of k-regular distributions.

Proposition 2.4.2. *Let p be a probability distribution with support precisely equal to a compact k dimensional sub-manifold (with or without boundary) of $\mathbb{R}^d$, M. Additionally, suppose that p has a continuous density function over M. Then it follows that p is k-regular.*

Proposition 2.4.2 implies that most data distributions that adhere to some sort of manifold-hypothesis will also exhibit regularity, with the regularity constant, k, being the intrinsic dimension of the manifold.

2.4.2 Estimation over regular distributions

We now turn our attention towards designing estimation algorithms over regular distributions, with our main goal being to estimate the probability mass of arbitrarily small balls. We begin by first addressing a slight technical detail – although the data distribution p may be regular, this does not necessarily mean that the regularity constant, k, is known. Knowledge of k is crucial because it determines how to properly interpolate probability masses from large radius balls to smaller ones.

Luckily, estimating k turns out to be an extremely well studied task, as for most probability distributions, k is a measure of the *intrinsic dimension*. Because there is a wide body of literature in this topic, we will assume from this point that k has been correctly estimated from S using any known algorithm for doing so (for example [50]). Nevertheless, for completeness, we provide an algorithm with provable guarantees for estimating k (along with a corresponding bound on the amount of needed data) in Appendix **??**.

We now return to the problem of $p(B(x,r))$ for a small value of r, and present an algorithm, $Est(x,r,S)$ (Algorithm 2), that estimates $p(B(x,r))$ from an i.i.d sample $S \sim p^n$.

Algorithm 2: $Est(x,r,S)$

$n \leftarrow |S|$
$b \leftarrow O\left(\frac{d \ln \frac{n}{\delta}}{\varepsilon^2}\right)$
$r_* = \min\{s > 0, |S \cap B(x,s)| = b\}$.
if $r_* > r$ **then**
    Return $\frac{br^k}{nr_*^k}$
else
    Return $\frac{|T \cap B(x,r)|}{n}$

Est uses two ideas: first, it leverages standard uniform convergence results to estimate the probability mass of all balls that contain a sufficient number of training examples. This is what leads to the specific value of b that is chosen. Second, it estimates the mass of smaller balls by interpolating from its estimates from larger balls. The k-regularity assumption is crucial for

this second step as it is the basis on which such interpolation is done.

Est has the following performance guarantee, which follows from standard uniform convergence bounds and the definition of k-regularity.

Proposition 2.4.3. *Let p be a regular distribution, and let $\varepsilon > 0$ be arbitrary. Then if $n = O\left(\frac{d \ln \frac{d}{\delta \varepsilon p_\varepsilon}}{\varepsilon^2 p_\varepsilon}\right)$ with probability at least $1-\delta$ over $S \sim p^n$, for all $x \in \mathbb{R}^d$ and $r > 0$,*

$$\left(1+\frac{\varepsilon}{2}\right)^{-1} \leq \frac{Est(x,r,S)}{p(B(x,r))} \leq \left(1+\frac{\varepsilon}{2}\right).$$

2.5 A Data-copy detecting algorithm

Algorithm 3: $DataCopyDetect(S,q,m)$

$m \leftarrow O\left(\frac{dn^2 \ln \frac{nd}{\delta\varepsilon}}{\varepsilon^4}\right)$
Sample $T \sim q^m$
$\{x_1, x_2, \ldots, x_n\} \leftarrow S$
$\{z_1, z_2, \ldots, z_m\} \leftarrow T$
for $i = 1, \ldots, n$ **do**
    Let $p_i(r)$ denote $Est(x_i, r, S)$
    Let $q_i(r)$ denote $\frac{|B(x_i,r) \cap T|}{m}$
    $radii \leftarrow \{||z - x_i|| : z \in T\} \cup \{0\}$
    $radii \leftarrow \{r : p_i(r) \leq \gamma, r \in radii\}$
    $r_i^* \leftarrow \max\{r : q_i(r) \geq \lambda p_i(r), r \in radii\}$
end for
Sample $U \sim q^{20/\varepsilon^2}$
$V \leftarrow U \cap (\bigcup_{i=1}^n B(x_i, r_i^*))$
Return $\frac{|V|}{|U|}$.

We now now leverage our subroutine, *Est*, to construct a data-copying detector, *Data_Copy_Detect* (Algorithm 3), that has bounded sample complexity when p is a regular distribution. Like all data-copying detectors (Definition 2.3.2), *Data_Copy_Detect* takes as input the training sample S, along with the ability to sample from a generated distribution q that is trained from S. It then

performs the following steps:

1. (line 1) Draw an i.i.d sample of $m = O\left(\frac{dn^2 \ln \frac{nd}{\delta\varepsilon}}{\varepsilon^4}\right)$ points from q.

2. (lines 6 - 10) For each training point, x_i, determine the largest radius r_i for which

$$\frac{|B(x_i, r_i) \cap T|}{m} \geq \lambda Est(x_i, r_i, S),$$
$$Est(x_i, r_i, S) \leq \gamma.$$

3. (lines 12 - 13) Draw a fresh sample of points from $U \sim q^{O(1/\varepsilon^2)}$, and use it to estimate the probability mass under q of $\cup_{i=1}^{n} B(x_i, r_i)$.

In the first step, we draw a *large* sample from q. While this is considerably larger than the amount of training data we have, we note that samples from q are considered free, and thus do not affect the sample complexity. The reason we need this many samples is simple – unlike p, q is not necessarily regular, and consequently we need enough points to properly estimate q around every training point in S.

The core technical details of *Data_Copy_Detect* are contained within step 2, in which data-copying regions surrounding each training point, x_i, are found. We use $Est(x, r, S)$ and $\frac{|B(x,r)\cap T|}{m}$ as proxies for p and q in Definition 2.2.1, and then search for the maximal radius r_i over which the desired criteria of data-copying are met for these proxies.

The only difficulty in doing this is that this could potentially require checking an infinite number of radii, r_i. Fortunately, this turns out not to be needed because of the following observation – we only need to check radii at which a new point from T is included in the estimation $q_i(r)$. This is because these our estimation for $q_i(r)$ does not change between them meaning that our estimate of the ratio between q and p is maximal nearby these points.

Once we have computed r_i, all that is left is to estimate the data-copy rate by sampling q once more to find the total mass of data-copying region, $\cup_{i=1}^{n} B(x_i, r_i)$.

2.5.1 Performance of Algorithm 3

We now show that given enough data, *Data_Copy_Detect* provides a close approximation of cr_q.

Theorem 2.5.1. *Data_Copy_Detect is a data-copying detector (Definition 2.3.2) with sample complexity at most*

$$m_p(\varepsilon,\delta) = O\left(\frac{d\ln\frac{d}{\delta\varepsilon p_\varepsilon}}{\varepsilon^2 p_\varepsilon}\right),$$

for all regular distributions, p.

Theorem 3 shows that our algorithm's sample complexity has standard relationships with the tolerance parameters, ε and δ, along with the input space dimension d. However, it includes an additional factor of $\frac{1}{p_\varepsilon}$, which is a distribution specific factor measuring the regularity of the probability distribution. Thus, our bound cannot be used to give a bound on the amount of data needed without having a bound on p_ε.

We consequently view our upper bound as more akin to a convergence result, as it implies that our algorithm is guaranteed to converge as the amount of data goes towards infinity.

2.5.2 Applying Algorithm 3 to Halfmoons

We now return to the example presented in Figure **??** and empirically investigate the following question: is our algorithm able to outperform the one given in [38] over this example?

To investigate this, we test both algorithms over a series of distributions by varying the parameter ρ, which is the proportion of points that are "copied." Figure **??** demonstrates a case in which $\rho = 0.4$. Additionally, we include a parameter, c, for [38]'s algorithm which represents the number of clusters the data is partitioned into (with c-means clustering) prior to running their test. Intuitively, a larger number of clusters means a better chance of detecting more localized data-copying.

The results are summarized in the following table where we indicate whether the algorithm determined a statistically significant amount of data-copying over the given generated

distribution and corresponding training dataset. Full experimental details can be found in Sections **??** and **??** of the appendix.

Table 2.1. Statistical Significance of data-copying Rates over Halfmoons

Algo	$\mathbf{q = p}$	$\rho = \mathbf{0.1}$	**0.2**	**0.3**	**0.4**
Ours	no	yes	yes	yes	yes
$\mathbf{c = 1}$	no	no	no	no	no
$\mathbf{c = 5}$	no	no	no	no	yes
$\mathbf{c = 10}$	no	no	no	no	yes
$\mathbf{c = 20}$	no	no	no	yes	yes

As the table indicates, our algorithm is able to detect statistically significant data-copying rates in all cases it exists. By contrast, [38]'s test is only capable of doing so when there is a large data-copy rate and when the number of clusters, c, is quite large.

2.6 Is smoothness necessary for data copying detection?

Algorithm 3's performance guarantee requires that the input distribution, p, be regular (Definition 2.4.1). This condition is essential for the algorithm to successfully estimate the probability mass of arbitrarily small balls. Additionally, the parameter, p_ε, plays a key role as it serves as a measure of how "smooth" p is with larger values implying a higher degree of smoothness.

This motivates a natural question – can data copying detection be done over unsmooth data distributions? Unfortunately, the answer turns out to be no. In the following result, we show that if the parameter, p_ε is allowed to be arbitrarily small, then this implies that for any data-copy detector, there exists p for which the sample complexity is arbitrarily large.

Theorem 2.6.1. *Let B be a data-copying detector. Let $\varepsilon = \delta = \frac{1}{3}$. Then, for all integers $\kappa > 0$,*

there exists a probability distribution p such that $\frac{1}{9\kappa} \leq p_\varepsilon \leq \frac{1}{\kappa}$, and $m_p(\varepsilon,\delta) \geq \kappa$, implying that

$$m_p(\varepsilon,\delta) \geq \Omega\left(\frac{1}{p_\varepsilon}\right).$$

Although Theorem 2.6.1 is restricted to regular distributions, it nevertheless demonstrates that a bound on smoothness is essential for data copying detection. In particular, non-regular distributions (with no bound on smoothness) can be thought of as a degenerate case in which $p_\varepsilon = 0$.

Additionally, Theorem 2.6.1 provides a lower bound that complements the Algorithm 3's performance guarantee (Theorem 2.5.1). Both bounds have the same dependence on p_ε implying that our algorithm is optimal at least in regards to p_ε. However, our upper bound is significantly larger in its dependence on d, the ambient dimension, and ε, the tolerance parameter itself.

While closing this gap remains an interesting direction for future work, we note that the existence of a gap isn't too surprising for our algorithm, *Data_Copy_Detect*. This is because *Data_Copy_Detect* essentially relies on manually finding the entire region in which data-copying occurs, and doing this requires precise estimates of p at all points in the training sample.

Conversely, detecting data-copying only requires an *overall* estimate for the data-copying rate, and doesn't necessarily require finding all of the corresponding regions. It is plausible that more sophisticated techniques might able to estimate the data-copy rate *without* directly finding these regions.

2.7 Conclusion

In conclusion, we provide a new modified definition of "data-copying" or generating memorized training samples for generative models that addresses some of the failure modes of previous definitions [38]. We provide an algorithm for detecting data-copying according to our definition, establish performance guarantees, and show that at least some smoothness conditions are needed on the data distribution for successful detection.

With regards to future work, one important direction is in addressing the limitations discussed in section 2.2.2. Our definition and algorithm are centered around the assumption that the goal of a generative model is to output q that is close to p in a mathematical sense. As a result, we are unable to handle cases where the generator tries to generate *transformed* examples that lie outside the support of the training distribution. For example, a generator restricted to outputting black and white images (when trained on color images) would remain completely undetected by our algorithm regardless of the degree with which it copies its training data. To this end, we are very interested in finding generalizations of our framework that are able to capture such broader forms of data-copying.

Bibliography

[1] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[2] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples, March 20 2014. URL http://arxiv.org/abs/1412.6572.

[3] Daniel Lowd and Christopher Meek. Adversarial learning. In Robert Grossman, Roberto J. Bayardo, and Kristin P. Bennett, editors, *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Chicago, Illinois, USA, August 21-24, 2005*, pages 641–647. ACM, 2005. ISBN 1-59593-135-X.

[4] Ludwig Schmidt, Shibani Santurkar, Dimitris Tsipras, Kunal Talwar, and Aleksander Madry. Adversarially robust generalization requires more data. In *Advances in Neural Information Processing Systems 31*, pages 5014–5026. Curran Associates, Inc., 2018.

[5] Yizhen Wang, Somesh Jha, and Kamalika Chaudhuri. Analyzing the robustness of nearest neighbors to adversarial examples. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pages 5120–5129, 2018.

[6] Omar Montasser, Steve Hanneke, and Nathan Srebro. VC classes are adversarially robustly learnable, but only improperly. In *Conference on Learning Theory, COLT 2019, 25-28 June 2019, Phoenix, AZ, USA*, pages 2512–2530, 2019.

[7] Hadi Salman, Greg Yang, Jerry Li, Pengchuan Zhang, Huan Zhang, Ilya P. Razenshteyn, and Sébastien Bubeck. Provably robust deep learning via adversarially trained smoothed classifiers. *CoRR*, abs/1906.04584, 2019. URL http://arxiv.org/abs/1906.04584.

[8] Dmitrii Avdiukhin, Slobodan Mitrovic, Grigory Yaroslavtsev, and Samson Zhou. Adversarially robust submodular maximization under knapsack constraints. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019.*, pages 148–156, 2019. doi:

10.1145/3292500.3330911.

[9] Luc Devroye, László Györfi, and Gábor Lugosi. *A Probabilistic Theory of Pattern Recognition*, volume 31 of *Stochastic Modelling and Applied Probability*. Springer, 1996.

[10] Charles Stone. Consistent nonparametric regression. *The Annals of Statistics*, 5(4):595–645, 1977.

[11] Yao-Yuan Yang, Cyrus Rashtchian, Yizhen Wang, and Kamalika Chaudhuri. Adversarial examples for non-parametric methods: Attacks, defenses and large sample limits. *CoRR*, abs/1906.03310, 2019. URL http://arxiv.org/abs/1906.03310.

[12] Nicholas Carlini and David A. Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017*, pages 39–57, 2017.

[13] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into transferable adversarial examples and black-box attacks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

[14] Nicolas Papernot, Patrick D. McDaniel, Ian J. Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. Practical black-box attacks against deep learning systems using adversarial examples. *ASIACCS*, 2017.

[15] Nicolas Papernot, Patrick D. McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *IEEE European Symposium on Security and Privacy, EuroS&P 2016, Saarbrücken, Germany, March 21-24, 2016*, pages 372–387, 2016.

[16] Matthias Hein and Maksym Andriushchenko. Formal guarantees on the robustness of a classifier against adversarial manipulation. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 2266–2276. Curran Associates, Inc., 2017.

[17] Guy Katz, Clark W. Barrett, David L. Dill, Kyle Julian, and Mykel J. Kochenderfer. Towards proving the adversarial robustness of deep neural networks. In *Proceedings First Workshop on Formal Verification of Autonomous Vehicles, FVAV@iFM 2017, Turin, Italy, 19th September 2017.*, pages 19–26, 2017.

[18] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.

[19] Nicolas Papernot, Patrick D. McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *IEEE Symposium on Security and Privacy, SP 2016, San Jose, CA, USA, May 22-26, 2016*, pages 582–597, 2016.

[20] Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Certified defenses against adversarial examples. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.

[21] Aman Sinha, Hongseok Namkoong, and John C. Duchi. Certifying some distributional robustness with principled adversarial training. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.

[22] Laurent Amsaleg, James Bailey, Dominique Barbe, Sarah M. Erfani, Michael E. Houle, Vinh Nguyen, and Milos Radovanovic. The vulnerability of learning to adversarial perturbation increases with intrinsic dimensionality. In *2017 IEEE Workshop on Information Forensics and Security, WIFS 2017, Rennes, France, December 4-7, 2017*, pages 1–6, 2017.

[23] Chawin Sitawarin and David A. Wagner. On the robustness of deep k-nearest neighbors. In *2019 IEEE Security and Privacy Workshops, SP Workshops 2019, San Francisco, CA, USA, May 19-23, 2019*, pages 1–7, 2019.

[24] Maksym Andriushchenko and Matthias Hein. Provably robust boosted decision stumps and trees against adversarial attacks. In H. Wallach, H. Larochelle, A. Beygelzimer, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 12997–13008. Curran Associates, Inc., 2019.

[25] Alex Kantchelian, J. D. Tygar, and Anthony D. Joseph. Evasion and hardening of tree ensemble classifiers. *CoRR*, abs/1509.07892, 2015. URL http://arxiv.org/abs/1509.07892.

[26] Hongge Chen, Huan Zhang, Duane S. Boning, and Cho-Jui Hsieh. Robust decision trees against adversarial examples. *CoRR*, abs/1902.10660, 2019.

[27] Geoffrey W. Gates. The reduced nearest neighbor rule (corresp.). *IEEE Trans. Information Theory*, 18(3):431–433, 1972.

[28] Lee-Ad Gottlieb, Aryeh Kontorovich, and Pinhas Nisnevitch. Near-optimal sample compression for nearest neighbors. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 370–378, 2014.

[29] Peter E. Hart. The condensed nearest neighbor rule (corresp.). *IEEE Trans. Information Theory*, 14(3):515–516, 1968.

[30] Aryeh Kontorovich, Sivan Sabato, and Roi Weiss. Nearest-neighbor sample compression: Efficiency, consistency, infinite dimensions. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 1573–1583, 2017.

[31] Aryeh Kontorovich and Roi Weiss. A bayes consistent 1-nn classifier. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2015, San Diego, California, USA, May 9-12, 2015*, 2015.

[32] Steve Hanneke, Aryeh Kontorovich, Sivan Sabato, and Roi Weiss. Universal bayes consistency in metric spaces. *CoRR*, abs/1906.09855, 2019.

[33] Ingo Steinwart. Consistency of support vector machines and other regularized kernel classifiers. *IEEE Trans. Information Theory*, 51(1):128–142, 2005.

[34] David Lopez-Paz and Maxime Oquab. Revisiting classifier two-sample tests. *arXiv preprint arXiv:1610.06545*, 2016.

[35] Qiantong Xu, Gao Huang, Yang Yuan, Chuan Guo, Yu Sun, Felix Wu, and Kilian Q. Weinberger. An empirical study on evaluation metrics of generative adversarial networks. *CoRR*, abs/1806.07755, 2018. URL http://arxiv.org/abs/1806.07755.

[36] Gavin Brown, Mark Bun, Vitaly Feldman, Adam Smith, and Kunal Talwar. When is memorization of irrelevant training data necessary for high-accuracy learning? In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*. ACM, jun 2021. doi: 10.1145/3406325.3451131. URL https://doi.org/10.1145%2F3406325.3451131.

[37] Vitaly Feldman. Does learning require memorization? a short tale about a long tail. In Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 954–959. ACM, 2020. doi: 10.1145/3357713.3384290. URL https://doi.org/10.1145/3357713.3384290.

[38] Casey Meehan, Kamalika Chaudhuri, and Sanjoy Dasgupta. A three sample hypothesis test for evaluating generative models. In Silvia Chiappa and Roberto Calandra, editors, *The 23rd International Conference on Artificial Intelligence and Statistics, AISTATS 2020, 26-28 August 2020, Online [Palermo, Sicily, Italy]*, volume 108 of *Proceedings of Machine Learning Research*, pages 3546–3556. PMLR, 2020.

[39] Hoang Thanh-Tung and Truyen Tran. Catastrophic forgetting and mode collapse in gans. In *2020 International Joint Conference on Neural Networks, IJCNN 2020, Glasgow, United Kingdom, July 19-24, 2020*, pages 1–10. IEEE, 2020. doi: 10.1109/IJCNN48605.2020.9207181. URL https://doi.org/10.1109/IJCNN48605.2020.9207181.

[40] Yasin Yazici, Chuan-Sheng Foo, Stefan Winkler, Kim-Hui Yap, and Vijay Chandrasekhar. Empirical analysis of overfitting and mode drop in gan training. In *IEEE International Conference on Image Processing, ICIP 2020, Abu Dhabi, United Arab Emirates, October 25-28, 2020*, pages 1651–1655. IEEE, 2020. doi: 10.1109/ICIP40778.2020.9191083. URL https://doi.org/10.1109/ICIP40778.2020.9191083.

[41] Mehdi S. M. Sajjadi, Olivier Bachem, Mario Lucic, Olivier Bousquet, and Sylvain Gelly. Assessing generative models via precision and recall. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 5234–5243, 2018.

[42] Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *CoRR*, abs/1606.03498, 2016. URL http://arxiv.org/abs/1606.03498.

[43] Eitan Richardson and Yair Weiss. On gans and gmms. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 5852–5863, 2018.

[44] Satrajit Chatterjee. Learning and memorization. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 755–763. PMLR, 10–15 Jul 2018.

[45] Gerrit van den Burg and Chris Williams. On memorization in probabilistic deep generative models. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 27916–27928. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/file/eae15aabaa768ae4a5993a8a4f4fa6e4-Paper.pdf.

[46] Ching-Yuan Bai, Hsuan-Tien Lin, Colin Raffel, and Wendy Chi-wen Kan. On training sample memorization: Lessons from benchmarking generative modeling with a large-scale competition. In Feida Zhu, Beng Chin Ooi, and Chunyan Miao, editors, *KDD '21: The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, Singapore, August 14-18, 2021*, pages 2534–2542. ACM, 2021. doi: 10.1145/3447548.3467198. URL https://doi.org/10.1145/3447548.3467198.

[47] Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramèr, and Chiyuan Zhang. Quantifying memorization across neural language models. *CoRR*, abs/2202.07646, 2022. URL https://arxiv.org/abs/2202.07646.

[48] Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. The secret sharer: Evaluating and testing unintended memorization in neural networks. In Nadia Heninger and Patrick Traynor, editors, *28th USENIX Security Symposium, USENIX Security 2019, Santa Clara, CA, USA, August 14-16, 2019*, pages 267–284. USENIX Association, 2019.

[49] Cynthia Dwork. Differential privacy. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *Automata, Languages and Programming, 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part II*, volume 4052 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2006.

[50] Adam Block, Zeyu Jia, Yury Polyanskiy, and Alexander Rakhlin. Intrinsic dimension estimation using wasserstein distances. *Journal of machine learning research*, 1533-7928, 2022.