A Formal Approach to Adversarial Machine Learning

A

Dissertation

Presented to

the faculty of the School of Engineering and Applied Science

University of Virginia

in partial fulfillment

of the requirements for the degree

Doctor of Philosophy

by

Saeed Mahloujifar

August 2020

APPROVAL SHEET

This

Dissertation

is submitted in partial fulfillment of the requirements
for the degree of

Doctor of Philosophy

Author: Saeed Mahloujifar

This Dissertation has been read and approved by the examing committee:

Advisor: Mohammad Mahmoody

Advisor:

Committee Member: David Evans

Committee Member: Somesh Jha

Committee Member: David Wu

Committee Member: Denis Nekipelov

Committee Member:

Committee Member:

Accepted for the School of Engineering and Applied Science:

Craig H. Benson, School of Engineering and Applied Science

August 2020

Abstract

With the ever increasing applications of machine learning algorithms many new challenges, beyond accuracy, have been raised. Among them, and one of the most important ones, is robustness against adversarial attacks. The persistent impact of these attacks on the security of otherwise successful machine learning algorithms begs a fundamental investigation. This dissertation aims at building a foundation to systematically investigate robustness of machine learning algorithms in the presence of different adversaries.

Two special cases of security threats, which have been the focus of many studies in the recent years, are *evasion attacks* and *poisoning attacks*. Evasion attacks occur during the inference phase and refer to adversaries who perturb the input to a classifier to get their desired output. Poisoning attacks occur in the training phase where an adversary perturbs the training data, with the goal of leading the learning algorithm to choose an *insecure* hypothesis. This dissertation studies provable evasion and then poisoning attacks that could be applied to any learning algorithm and classification model. The dissertation also studies algorithmic aspects of such attacks and study the possibility of using hardness assumptions to prevent these general purpose attacks. Most of the attacks discussed in this dissertation are inspired by (and have implications for) coin tossing attacks in cryptography.

Acknowledgements

Throughout my graduate studies I had the opportunity to meet incredible friends and colleagues. I would like to take the chance thank them all for making this experience gratifying for me.

First, I want to thank my fantastic advisor, Professor Mohammad Mahmoody, who was my companion in this journey. I had such an enjoyable research experience with him. I am grateful for all the hundreds of hours of discussions that he had with me. His way of looking into research problems were very inspiring. I learned many lessons from him, most important of which was how to enjoy doing research. I believe he is the best advisor and college that I could have, and I am really grateful that I could have his advice during past 5 years.

I also want to thank my Ph.D. committee members for their participation in my committee. I was honored to have Professors David Evans, Somesh Jha, Denis Nekipelov, and David Wu in my dissertation committee and received a lot of great comments from them which made this dissertation possible.

I also must thank all my collaborators and mentors. I would like to thank Professor David Evans whose comments helped me from the early days of my Ph.D.. He was like a second adviser for me and I really appreciate my collaboration with him. I also want to thank Professor Somesh Jha, who I had the privilege of working with during the past year. He has been very supportive of me and my research. I also want to thank my mentors during my two internships at Microsoft Research, Esha Ghosh and Melissa Chase. My internships had a great influence on my research orientation. I would like to thank Professor Dimitrios Diochnos who was a great collaborator for me and helped me to get a better understanding of machine learning. I really appreciate my joyful collaboration with professor who was a great inspiration for me from the early days of Ph.D.. My collaboration with Omid Etesami was also very inspiring and I learned a lot from him. I also greatly enjoyed my work with Professor Abhradeep Thakurta and Professor Yuan Tian. Although I enjoyed my collaboration with senior researchers above, but my favorite collaborations are those that included another student. I want to thank Ameer Mohammed, Xiao Zhang and Fnu Suya for being such awesome friends and collaborators.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In recent years machine learning has been shown to be very successful in many domains. The progress of machine learning in practical applications has gone far beyond what theory can explain. On the other hand, as machine learning algorithms find new applications, there are emerging concerns about their deployment for critical applications. Trying to address these new concerns, machine learning developers have tried to come up with new algorithms. However, many of these concerns are still unresolved.

One important new aspect that has recently been studied is machine learning in adversarial settings. Many modern learning algorithms are shown to be vulnerable to adversaries who try to interfere with the training phase or testing phase. It has been shown for many learning algorithms that there exist adversarial algorithms who change small fraction of training data and cause the output model to make mistakes in favor of the adversary. These types of attacks are called poisoning attacks and there is huge line of work exploring them. There is also another type of adversarial attacks called evasion attacks that has got a lot of attention recently. The goal of an evasion adversary is to fool the learning model by adding carefully crafted and small perturbations to the test instances which will produce an "adversarial example".

The line of research on the power and limitations of poisoning and evasion attacks contains numerous attacks and many defenses designed (usually specifically) against them (e.g., see Yuan et al. [2019] and references therein). The state of affairs in attacks and defenses with regard to the robustness of learning systems in both the evasion and poisoning contexts shapes the main questions in my research. The first fundamental question in my research is about the limitations of robustness in adversarial settings.

What are the inherent limitations of defense mechanisms for evasion and poisoning attacks?

Equivalently, what are the inherent power of such attacks?

Understanding the answer to the above question is fundamental for finding the right bounds that robust learning systems can indeed achieve, and achieving such bounds would be the next natural goal. The second

question in my research is about possibility of designing machine learning algorithms that are *provably* secure against such adversaries.

> *Can we design provably secure machine learning algorithms, relying on the fact that adversary is computationally limited?*

In fact, this is the main theme that enables modern cryptography to prove security by leveraging the limitations of the adversary, e.g. computational power. We follow the same approach here and try to see what will change if we limit our adversary to be computationally bounded.

This thesis has three main parts:

1. Part 1: Training-time Attacks
2. Part 2: Inference-time Attacks
3. Part 3: Computational Complexity of Attacks

The first part studies different adversarial perturbation in the training phase which are known as data poisoning attacks. The second part is devoted to the adversarial attacks that happen during the inference phase and are called evasion attacks (the product of these attacks are perturbed instances that are known as adversarial examples). The third part studies the computational complexity of both inference and training-time attacks. In the rest of this section we will give a short overview of each part and give an bird's eye view of the contributions of this thesis. But before getting to the overview, we define several important notations that are used throughout the document.

1.1 Notations

The following introduces several notation that is used in this section and across other sections of this document. Note that this is not a complete list of notations used in this document and later we will provide notations that are specific to each chapter.

General notation. We use calligraphic letters (e.g., $\mathcal{X}$) for sets. By $u \leftarrow \mathbf{u}$ we denote sampling u from the probability distribution $\mathbf{u}$. For a randomized algorithm $R(\cdot)$, by $y \leftarrow R(x)$ we denote the randomized execution of R on input x outputting y. For a joint distribution $(\mathbf{u}, \mathbf{v})$, by $(\mathbf{u} \mid v)$ we denote the conditional distribution $(\mathbf{u} \mid \mathbf{v} = v)$. By $\text{Supp}(\mathbf{u}) = \{u \mid \Pr[\mathbf{u} = u] > 0\}$ we denote the support set of $\mathbf{u}$. By $T^{\mathbf{u}}(\cdot)$ we denote an algorithm $T(\cdot)$ with oracle access to a sampler for distribution $\mathbf{u}$ that upon every query returns a fresh sample from $\mathbf{u}$. By $\mathbf{u} \times \mathbf{v}$ we refer to the product distribution in which $\mathbf{u}$ and $\mathbf{v}$ are sampled independently. By $\mathbf{u}^n$ we denote the n-fold product $\mathbf{u}$ with itself returning n iid samples. Multiple instances of a random

variable D in the same statement (e.g., $(D, c(D))$ refer to the same sample. By PPT we denote "probabilistic polynomial time".

Notation for classification problems. A classification problem $(\mathcal{X}, \mathcal{Y}, D, \mathcal{C}, \mathcal{H})$ is specified by the following components. The set $\mathcal{X}$ is the set of possible *instances*, $\mathcal{Y}$ is the set of possible *labels*, D is a distribution over $\mathcal{X}$, $\mathcal{C}$ is a class of *concept* functions where $c \in \mathcal{C}$ is always a mapping from $\mathcal{X}$ to $\mathcal{Y}$. Even though in a learning problem, we usually work with a *family* of distributions (e.g., all distributions over $\mathcal{X}$) here we work with only one distribution D. The reason is that our results are *impossibility* results, and proving limits of learning under a known distribution D are indeed stronger results. We did not state the loss function explicitly, as we work with classification problems. In the cases that we work with a special loss function we use the notation $(\mathcal{X}, \mathcal{Y}, D, \mathcal{C}, \mathcal{H}, \ell oss)$. Also, sometimes where the concept class and the hypothesis class are the same, we use the notation $(\mathcal{X}, \mathcal{Y}, D, \mathcal{H})$. For $x \in \mathcal{X}, c \in \mathcal{C}$, the *risk* or *error* of a hypothesis $h \in \mathcal{H}$ is equal to $\mathsf{Risk}(h, c) = \Pr_{x \leftarrow D}[h(x) \neq c(x)]$. We are usually interested in learning problems $(\mathcal{X}, \mathcal{Y}, D, \mathcal{C}, \mathcal{H})$ with a specific metric d defined over $\mathcal{X}$ for the purpose of defining risk and robustness under instance perturbations controlled by metric d. Then, we simply write $(\mathcal{X}, \mathcal{Y}, D, \mathcal{C}, \mathcal{H}, \mathsf{d})$ to include d. For a concept function c, and a hypothesis h, we denote the error region of h by $\mathcal{E}(c, h) = \{x \in \mathcal{X}; h(x) \neq c(x)\}$.

1.2 Training-time (Poisoning) Attacks

A very important notion of robustness for a learning algorithm deals with adversaries that could make perturbations to the data that is used in the *training* phase. Here, we would like to know how much the risk of the produced hypothesis h might increase, if an adversary A tampers with the training data $\mathcal{T}$ with the goal of increasing the "error" (or any "bad" event in general) during the test phase. Such attacks are referred to as *poisoning* attacks Biggio et al. [2012]. The following definition captures poisoning adversaries who try to increase the probability of a bad event and decrease the "adversarial confidence".

Definition 1.2.1 (Adversarial confidence)**.** Let L be a learning algorithm for a classification problem $\mathcal{P} = (\mathcal{X}, \mathcal{Y}, D, \mathcal{C}, \mathcal{H})$, m be the sample complexity of L, and $c \in \mathcal{C}$ be any concept. Also let $B \colon \mathcal{H} \to 0, 1$ be an arbitrary bad property defined over the hypothesis class. We define the (adversarial) confidence as follows

$$\mathsf{Conf}_{\mathsf{A}}(m, c, B) = 1 - \Pr_{\substack{\mathcal{S} \leftarrow (D, c(D))^m \\ h \leftarrow L(\mathsf{A}(\mathcal{S}))}}[B(h) = 1].$$

By $\mathsf{Conf}(\cdot)$ we denote the confidence without any attack; namely, $\mathsf{Conf}(\cdot) = \mathsf{Conf}_I(\cdot)$ for the trivial (identity function) adversary I that does not change the training data.

The line of research on the power and limitations of poisoning attacks contains numerous attacks and many defenses designed against them. The goal of these attacks is usually hurting the accuracy of the trained model either on the whole distribution or on a specific instance (We can define the bad property B according with respect to each of these situations). The first part of this document is devoted to these type of attacks. Bellow is a quick summary of the result of part 2 which are based on 4 published papers.

1. Saeed Mahloujifar, Dimitrios I Diochnos, and Mohammad Mahmoody. The curse of concentration in robust learning: Evasion and poisoning attacks from concentration of measure. *Conference on Artificial Intelligence (AAAI)*, 2019a

2. Saeed Mahloujifar, Dimitrios I Diochnos, and Mohammad Mahmoody. Learning under p-tampering poisoning attacks. *Annals of Mathematics and Artificial Intelligence*, pages 1–34, 2019b

3. Saeed Mahloujifar, Mohammad Mahmoody, and Ameer Mohammed. Universal multi-party poisoning attacks. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 4274–4283. PMLR, 2019c. URL http://proceedings.mlr.press/v97/mahloujifar19a.html

4. Melissa Chase, Esha Ghosh, and Saeed Mahloujifar. Poisoning attacks against privacy of collaborative learning. *Under Submission*, 2020

One aspect of poisoning adversaries that must be defined is their tampering pattern. For instance, one can imagine a poisoning adversary who looks at the training data and changes a fraction of the training examples arbitrarily. Alternatively, a weaker adversary may only add some poisoned data to the original dataset, without knowing the other examples in training set[1]. The first three chapters of part 1, studies different poisoning adversaries based on the tampering pattern

1.2.1 Random Access Poisoning Attacks

In Chapter 1 of part 1 we introduce poisoning adversaries who could substitute a random p fraction of a training examples and replace them with other examples[2]. We showed that in this model there are adversaries who increase the probability of an arbitrary bad property by $\Omega(p)$ if the probability of getting the bad property is originally constant. The adversaries in this attack model, called p-tampering model, are very powerful in the sense that they can achieve these bounds with many restrictions such as (only) black-box access to the training algorithm, working online, and using the correct labels for the training examples. We

[1] These two models are known as strong contamination model and Huber's contamination model. For more details see Diakonikolas and Kane [2019].

[2] This adversarial model is tightly related to valiant's malicious noise model. For more details see Kearns and Li [1993a].

also prove that our attacks could be implemented in polynomial time, given oracle access to the training algorithm and enough samples from the distribution of instances.

1.2.2 Byzantine Adversaries in Multi-party Learning

Multi-party learning enables distinct parties to combine their data and train a shared model. With the recent advances in collaborative machine learning, it has become very important to study the effect of malicious parties who provide corrupted data. In Chapter 2 of part 1, we introduce a new model of (k, p) poisoning adversaries, in multi-party learning setting, where there are m parties who provide the training data. Among those, k are partially corrupted meaning that for each training example provided (by the partially corrupted parties) there is a probability p that the example is generated by the adversary. For $k = m$, this model becomes the notion of p-tampering poisoning, and for $p = 1$ it coincides with the standard notion of static corruption in multi-party computation. we showed, in this setting, for any m-party learning protocol there exist a computationally bounded (k, p) poisoning adversary that increases the probability of the bad property by $\Omega(p \cdot k/m)$. Our (k, p) poisoning attacks are online and only use correct labels for the corrupted training data. Moreover, we showed that our attack can be implemented in polynomial time as long as it has access to sampling oracle for distributions of all the parties as well as oracle access to the training algorithm.

1.2.3 Strong Adaptive Adversaries

One can define a stronger poisoning adversary that has control over the tampering locations. In particular, in this strong adversarial model, the adversary can inspect the training data first and then change a small fraction of the training data. In Chapter 3 of part 1, we studied these type of adversaries and proved that there are adversaries who select $\tilde{O}(\sqrt{m})$ (m is the sample complexity of the learning algorithm) number of training examples, replace them with other correctly labeled training examples, and increase the probability of the bad property to almost 1, if the original probability is $1/\text{poly}(m)$. The significance of this attack is that it uses sub-linear number of tampering and yet is able to increase the probability of the bad property to almost 1. This attack is based on the concentration of measure property of product distributions. Later, in par 3 of the document, we will see how we made this information theoretic attacks to run in polynomial.

1.2.4 Poisoning Attacks against Privacy

A good thing about defining poisoning attacks in an abstract way , and based on a bad property (See Definition 1.2.1), is that it can capture different adversarial goals. The current body of work on poisoning attacks usually considers adversarial goals that relate to the error of the final mode (e.g. increasing the population error

or error on a specific instance). To show that poisoning adversaries can attack other qualities of machine learning, we studied the poisoning attacks that aim at causing the output model to leak information about the underlying training set. In particular, we consider aggregate property leakage, where there is a property P over the training examples and the goal of adversary is to find out the fraction of examples in the training set that satisfy the property P. It is important to not that these type of leakage are not protected by Differential Privacy and hence it is important to study them. In Section 4 of Part 1 we show how poisoning attacks could significantly increase these property leakage. In particular, we first show a theoretical attack that uses the generalization power of machine learning algorithms to infer any property leakage, as long as it can add sufficient number of poisoning data to the training set. Then we implement these attacks for simple machine learning algorithms and show it can significantly boost the property leakage. Note that poisoning attacks against privacy do not have knowledge of the training data (otherwise the property leakage would be obsolete) and that makes it harder to find poisoning points. However, our attack uses a very simple poisoning strategy, enabling the adversary to infer the expected value of any property on the training set.

1.3 Inference-time (Evasion) Attacks

Learning how to classify instances based on labeled examples is a fundamental task in machine learning. The goal is to find, with high probability, the correct label $c(x)$ of a given test instance x coming from a distribution D. Thus, we would like to find a good-on-average "hypothesis" h (also called the trained model) that minimizes the error probability $\Pr_{x \leftarrow D}[h(x) \neq c(x)]$, which is referred to as the risk of h with respect to the ground truth c. Due to the explosive use of learning algorithms in real-world systems (e.g., using neural networks for image classification) a more modern approach to the classification problem aims at making the learning process more *robust*. Namely, even if the instance x is perturbed in a limited way into x' by an adversary A, we would like to have the hypothesis h still predict the right label for x'.

One major motivation behind this problem comes from scenarios such as image classification, in which the adversarially perturbed instance x' would still "look similar" to the original x, at least in humans' eyes, even though the classifier h might now misclassify x' Goodfellow et al. [2018]. In fact, starting with the work of Szegedy et al. Szegedy et al. [2014] an active line of research investigated various attacks and possible defenses to resist such attacks. The race between attacks and defenses in this area motivates a study of whether or not such robust classifiers could ever be found.

The current literature contains multiple definitions of risk and robustness in the presence of evasion adversaries. In a work with Diochnos and Mahmoody Mahloujifar et al. [2018d], we formalized these definitions

based on their goals. The definition that we work with in this document is the *error region* definition of Mahloujifar et al. [2018d]. In Chapter 1 of Part 2 we will discuss other definitions as well.

Definition 1.3.1 (Error region adversarial risk)**.** Let $(\mathcal{X}, \mathcal{Y}, D, \mathcal{C}, \mathcal{H}, \mathsf{d})$ be a classification problem. For $h \in \mathcal{H}$ and $c \in \mathcal{C}$, let $\mathcal{E} = \{x \in \mathcal{X} \mid h(x) \neq c(x)\}$ be the error region of h with respect to c. Then, Adversarial risk is defined as follows. For $b \in \mathbb{R}_+$, the (error-region) *adversarial risk* under b-perturbation is

$$\mathsf{Risk}_b(h, c) = \Pr_{x \leftarrow D}[\exists\, x' \in \mathcal{B}all_b(x) \cap \mathcal{E}] = D(\mathcal{E}_b).$$

We might call b the "budget" of an imaginary "adversary" who perturbs x into x'. Using $b = 0$, we recover the standard notion of risk: $\mathsf{Risk}(h, c) = \mathsf{Risk}_0(h, c) = D(\mathcal{E})$.

Throughout Part 2, we try to understand why adversarial examples persist despite all the research that is devoted to eliminate them. Bellow is a summary of all chapters except for Chapter 1 which is devoted to definitions. These chapters are based on 4 papers as follows:

1. Dimitrios Diochnos, Saeed Mahloujifar, and Mohammad Mahmoody. Adversarial risk and robustness: General definitions and implications for the uniform distribution. In *Advances in Neural Information Processing Systems*, pages 10359–10368, 2018a

2. Saeed Mahloujifar, Dimitrios I Diochnos, and Mohammad Mahmoody. The curse of concentration in robust learning: Evasion and poisoning attacks from concentration of measure. *Conference on Artificial Intelligence (AAAI)*, 2019a

3. Saeed Mahloujifar, Xiao Zhang, Mohammad Mahmoody, and David Evans. Empirically measuring concentration: Fundamental limits on intrinsic robustness. *Advances in Neural Information Processing Systems*, 2019d

4. Dimitrios I Diochnos, Saeed Mahloujifar, and Mohammad Mahmoody. Lower bounds for adversarially robust pac learning. *arXiv preprint arXiv:1906.05815*, 2019

1.3.1 Adversarial Examples from Concentration of Measure

Persistence of adversarial examples has raised a serious concern regarding possibility of implementing a robust machine learning algorithm. To investigate this important issue we posed a research question. In particular, we asked, is there an upper bound on the robustness of machine learning algorithms against evasion adversaries? Alternatively, is there a lower bound on the power of evasion adversaries? We attend to this questions in a Section 2 of Part 2. We show an inherent upper bound on achievable robustness in the presence

of evasion adversaries. Explicitly, we draw a connection between the robustness of learning algorithms and a well-studied mathematical phenomena known as *concentration of measure*. We showed that if the metric probability space of the underlying input distribution is well concentrated and the trained hypothesis has some non-negligible error, for most of the instances, there exist perturbations with sub-linear magnitude which when applied to that instance, cause the classifier to output a wrong label. The concentration of measure phenomenon, which is tightly related to isoperimetric and optimal transport inequalities, states that for many natural metric probability spaces (e.g. all so called Lévy families) and for any subset with a constant measure, almost every point sampled from the measure has sub-linear distance from that subset. There are many mathematical results on the concentration of natural metric probability spaces, such as product distributions under hamming distance, Gaussian distribution under euclidean distance, product of unit spheres under the euclidean or geodesic distance and more. We showed that any such concentration inequality for a metric probability space will give an upper bound on achievable robustness of any classification problem where instances are coming from that probability space.

1.3.2 Black-box Estimation of Concentration

Although the result of Section 2 of Part 2 shows the connection between concentration of measure and adversarial robustness, it was still unclear whether real-world data distributions are concentrated enough to justify the existence of adversarial examples. The next immediate question then was whether we can translate the theoretical upper bounds to the real world applications. In Section 3 of Part 2, we introduce a new method to estimate concentration of measure for an arbitrary metric probability space, using i.i.d. samples. We design an empirical concentration problem and proved that the solution of this empirical problem converges to the solution of the actual concentration problem asymptotically. In our published paper Mahloujifar et al. [2019d], we also provided a heuristic algorithm to solve the empirical concentration for estimating concentration of measure on image datasets such as MNIST and CIFAR10[3]. Our results showed that even though there are cases where the robustness of the algorithm in practice is very close to what our estimation of concentration suggests, for some cases, the gap between the two is large. This finding suggested that concentration of measure alone cannot fully explain the existence of adversarial examples in some of the practical scenarios.

[3]The description of the heuristic and experimental results are not present in this dissertation , please see the full version of our paper for more details on the experiments.

1.3.3 Lower Bounds on Adversarially Robust PAC Learning

Our result in Section 2 of Part 2 leveraged the fact that the target classifier have some initial error. A natural question that follows is whether we can decrease the error rate of the classifier so that the effect of the adversaries are not as severe. In other words, can we mitigate these attacks by training better classifiers with smaller error rate? To answer this question, in Chapter 3 of Part 2, we studied the effect of evasion adversaries on the required sample complexity of learning algorithms. We showed that there exist a learning problem which requires exponential sample complexity to achieve small adversarial risk. However, in the absence of adversaries, the same problem can be learned with polynomial number of samples. This shows a barrier against learning robust classifier by simply using more training data.

1.4 Computational Complexity of Attacks

An important open question about security of machine learning is whether one can rely on the fact that adversaries are computationally bounded and design secure schemes. This technique is what enables many constructions in cryptography which are provably secure as long as the adversary can only compute bounded number of operations. Inspired by the success of the field of cryptography in exploiting computational limitation of adversaries, we discuss the power and limitation of computationally bounded training-time and inference-time adversaries in Part 3 of this dissertation. Part 3 consists of 3 parts and is based on the following three papers.

1. Saeed Mahloujifar and Mohammad Mahmoody. Can adversarially robust learning leverage computational hardness? *arXiv preprint arXiv:1810.01407*, 2018a

2. Omid Etesami, Saeed Mahloujifar, and Mohammad Mahmoody. Computational concentration of measure: Optimal bounds, reductions, and more. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 345–363. SIAM, 2020

3. Sanjam Garg, Somesh Jha, Saeed Mahloujifar, and Mahmoody Mohammad. Adversarially robust learning could leverage computational hardness. In *Algorithmic Learning Theory*, pages 364–385, 2020

Computational hardness can potentially be used for resisting both training and inference time attacks. We show some provable attacks for both training and in Parts 1 and 2 of the documents, some of the strongest attacks only work if the adversary has infinite computational power. In particular, the strong adaptive poisoning attacks that use sublinear number of perturbations and also the impossibility result for inference-time robustness only hold for computationally unbounded adversaries. We tried to attend the

question of possibility of using computational hardness to resist these attacks. Bellow is a summary of the progress we have made so far in this direction.

1.4.1 Can adversarially robust learning leverage computational hardness?

In Section 1 of Part 3, we discuss the general idea of using computational hardness for robustness of machine learning. In particular, we provide "computational" definition similar to definition of cryptographic primitives. In that section, we also provide some negative result. Specifically, we show that

we showed that if instances are coming from a product distribution, it is computationally feasible to find adversarial examples with $O(\sqrt{n})$ perturbations (under Hamming distance) as long as the adversary has black-box access to the hypothesis. In this section, we introduced a new notion called *Computational Concentration of Measure* and showed that it is sufficient for getting polynomial time inference-time attacks. Remember that our strong adaptive poisoning attacks in Part 1, Section 4 were also based on concentration of measure for product distributions. Our computational concentration of measure result also implies that our strong adaptive poisoning attacks could be implemented in polynomial-time as well. This rules out the possibility of using computational hardness for general purpose poisoning attacks. Although these two results sound disappointing, there is still some gap between computational and information theoretic adversaries. We will discuss this gap in the Sections 2 and 3 In particular for poisoning attacks, as we our result on computational concentration does not have the exact same power of our information theoritic attack.

1.4.2 Optimal Bounds for Computational Concentration of Measure

Although the result of Section 1 shows a barrier against leveraging on hardness assumptions to design learning algorithms that are robust against polynomial time adversaries. Yet, there was a gap between the power of algorithmic attacks of Section 1 of Part 3 and the existential attacks of Parts 1 and 2. In this section, we show that how we can improve our computational concentration of measure bounds to match the information theoretic version of concentration of measure. Using this, we could improve our poisoning attack and close the gap between power of computationally bounded and unbounded attack for poisoning. As for evasion attacks, we extend the computational concentration beyond Hamming distance and prove our result for another theoretically studied distribution. Specifically, we show that isotropic Gaussian distribution, equipped with ℓ_1 distance is computationally concentrated. This shows that for such distributions, there are computationally bounded adversaries who can find adversarial examples in polynomial time.

1.4.3 Separating Computational Robustness from Information-theoretic Robustness

Although the result of Sections 1 and 2 might sounds disappointing, it does not imply that computational hardness assumptions cannot be helpful. Considering that there is a gap between existing algorithm's robustness and theoretical upper bounds, computational hardness might help closing this gap. In particular, improving machine learning algorithms to match the existing theoretical upper bounds is still an open problem. One might be able to use computational hardness assumptions to close this gap by assuming adversaries that are computationally bounded. In Section 3 of Part 3, we explore this problem. We constructed a learning problem which its computational robustness was much higher than its information theoretic robustness. To achieve this, we use some cryptographic primitive to construct learning problems that has high computational robustness and low information-theoretic robustness. This is a first step in showing that machine learning can enjoy computational assumptions that is used in cryptography.

Part I

Training-time Attacks

Chapter 1

Random Access Poisoning Attacks

1.1 Introduction

In his seminal work, Valiant [Valiant, 1984] introduced the Probably Approximately Correct (PAC) model of learning that triggered a significant amount of work in the theory of machine learning.[1] An important characteristic of learning algorithms is their ability to cope with noise. Valiant also initiated a study of adversarial noise [Valiant, 1985] in which each incoming training example is chosen, with independent probability p, by an adversary who knows the learning algorithm. Since no assumptions are made on such adversarial examples, this type of noise is called *malicious*. Subsequently, Kearns and Li [Kearns and Li, 1993b] and the follow-up work of Bshouty et. al [Bshouty et al., 2002] essentially proved the impossibility of PAC learning under such malicious noise by heavily relying on the existence of *mistakes* (i.e., wrong labels) in adversarial examples given to the learner under a carefully chosen specific distribution. In its simplest form, the main idea of their approach was to make it impossible for the learner to distinguish between two different target concepts, and this was achieved by generating wrong labels at an appropriate rate under a carefully chosen pathological distribution. This approach for obtaining a negative result is a consequence of Valiant's model of distribution-free PAC learning, since in general, the learning algorithms have to be able to deal well with all distributions.[2]

The method of induced distributions gained popularity and was seen as a tool that was used in order to prove negative results within various noise models. Sloan in [Sloan, 1995] used this method in order to determine an upper bound on the error rate that can be tolerated in a noise model where the labels can be

[1] The original model studies learnability in a distribution-free sense, but it also make sense for classes of distributions; [Benedek and Itai, 1991].

[2] In fact, determining properties of *distribution-free* learning algorithms by looking at their behavior *under specific distributions* makes sense in the noise-free setting as well; for example, [Blumer et al., 1989, Ehrenfeucht et al., 1989] obtain lower bounds on the number of examples needed for learning by looking at specific distributions.

mislabeled maliciously. Bshouty, Eiron, and Kushilevitz [Bshouty et al., 2002] studied a noise model closely related to Valiant's malicious noise, in which the adversary is allowed to make its choices based on the full knowledge of the original training examples; in their work they used the method of induced distributions in order to give an upper bound on the maximum amount of noise that can be tolerated by any learning algorithm.

In contrast to the works of [Kearns and Li, 1993b, Sloan, 1995, Bshouty et al., 2002] who used the method of (pathological) induced distributions from where the malicious samples were drawn, in this Section we are interested in attackers who do *not* have any control over the the original distributions, but they can choose and inject malicious examples in certain (restricted) ways. This is indeed a subtle point, as adversary might still shift the distribution of the instances through the attack, but we emphasize that it is not the adversary (nor our proof of the negative result) that chooses the *original* untampered distribution. On the other hand, it is also worth noting that near the end of our work in this chapter we also provide a construction for a negative result within PAC learning. Interestingly, our idea for the behavior of the adversary that yields this negative learning result in our framework is the same as the key idea underlying the method of induced distributions where one tries to make it impossible for the learner to disambiguate between competing target concepts; however, in our context no wrong labels are used.

Poisoning attacks. Impossibility results against learning under adversarial noise could be seen as attacks against learners in which the attacker injects some malicious training examples to the training set and tries to prevent the learner from finding a hypothesis with low risk. Such attackers, in general, are studied in the context of *poisoning* (a.k.a causative) attacks[3] [Barreno et al., 2006, Biggio et al., 2012, Papernot et al., 2016a]. This type of attack has recently gained a lot of attention in machine learning community as a security threat in machine learning systems. There has been a lot of empirical studies on power of poisoning attacks that show learning algorithms are vulnerable to small adversarial changes in the training set (For example see [Biggio et al., 2012, Shafahi et al., 2018a, Wang and Chaudhuri, 2018]). In this section, we focus on attacks that can achieve provable bounds against any learning algorithm with the hope of getting a better theoretical understanding of this phenomenon.

Poisoning attacks could happen naturally when a learning process happens over time [Rubinstein et al., 2009b,a] and the adversary has some noticeable chance of injecting or substituting malicious training data in an online manner. A stronger form of poisoning attacks are the so called *targeted* (poisoning) attacks [Barreno et al., 2006, Shen et al., 2016], where the adversary performs the poisoning attack while she has a

[3] At a technical level, the malicious noise model also allows the adversary to know the *full* state (and thus the randomness) of the learner, while this knowledge is not given to the adversary of the poisoning attacks, who might be limited in other ways as well.

particular test example in mind, and her goal is to make the final generated hypothesis fail on that particular test example. While poisoning attacks against *specific* learners were studied before [Awasthi et al., 2014, Xiao et al., 2015, Shen et al., 2016], the recent work of Mahloujifar and Mahmoody [Mahloujifar and Mahmoody, 2017b] presented a generic targeted poisoning attack that could be adapted to apply to *any learner*, so long as there is an initial non-negligible error over the target point.

p-tampering (random access) attacks. The work of [Mahloujifar and Mahmoody, 2017b] proved their result using a special case of Valiant's malicious noise, called p-tampering, in which the attacker can only use *mistake-free* (i.e., correct label) *malicious noise*. Namely, similar to Valiant's model, any incoming training example might be chosen adversarially with independent probability p (see Definition 1.2.5 for a formalization). However, the difference between p-tampering noise and Valiant's malicious noise (and even from all of its special cases studied before [Sloan, 1995]) is that a p-tampering adversary is only allowed to choose *valid* tampered examples with *correct* labels[4] to substitute the original examples. As such, although the attributes can change pretty much arbitrarily in the tampered examples, the label of the tampered examples shall still reflect the correct label. For example, the adversary can repeatedly present the same example to the learner, thus reducing the effective sample size, or it can be the case that the adversary returns correct examples that are somehow chosen against the learner's algorithm and based on the whole history of the examples so far. Therefore, as opposed to the general model of Valiant's malicious noise, p-tampering noise/attacks are 'defensible' as the adversary can always claim that a malicious training example is indeed generated from the same original distribution from which the rest of the training examples are generated. Similar notions of defensible attacks are previously explored in the context of cryptographic attacks [Haitner et al., 2010, Aumann and Lindell, 2007]. Therefore, learning under p-tampering can be seen as a generalization of "robustness" [Xu and Mannor, 2012, Yamazaki et al., 2007, González and Abu-Mostafa, 2015] in which the training distribution can *adaptively* and *adversarially* deviate from the testing distribution without using wrong labels.

Targeted Poisoning Attacks through Biasing Bounded Functions. At the heart of the poisoning attacks of [Mahloujifar and Mahmoody, 2017b] against learners was a p-tampering attack for the more basic task of *biasing* the expected value of bounded real-valued functions. In particular, [Mahloujifar and Mahmoody, 2017b] proved that for any (polynomial time computable) function f mapping inputs drawn from distributions like $S \equiv D^n$ (consisting of n independent identically distributed 'blocks') to $[0,1]$, there is always a *polynomial time* p-tampering attacker A who changes the input distribution S into $\widetilde{S}$ while increasing

[4] This is assuming that the original training distribution only contains correct labels.

the expected value by at least $\frac{p}{3+5p} \cdot \mathbb{V}[f(S)]$ where $\mathbb{V}[\cdot]$ is the variance.[5] (Note that the bias shall somehow depend on $\mathbb{V}[f(S)]$ since constant functions cannot be biased by changing their inputs.)

To see the relation of these biasing attacks to targeted poisoning, consider f to be a real valued function over training set S such that $f(S)$ is equal to the loss of the final model over a target instance x, where S is the training set that is used to train the model. By applying the biasing attack to this function, the attacker can increase the expected loss of the final model over the target instance x. Note that $\mathbb{V}[f(S)]$ is the variance of the loss function over x and one can obtain lower bounds for it using bias-variance trade-off arguments [Rao, 1992].

The work of [Mahloujifar and Mahmoody, 2017b] shows that for some functions even *computationally unbounded* p-tampering attackers (who can run in exponential time) cannot achieve better than $\frac{\ln(1/\mu)}{1-\mu} \cdot p \cdot \mathbb{V}[f(S)]$ for all $p, \mu \in (0,1)$, if $\mu = \mathbb{E}[f(S)]$, which because of $\lim_{\mu \to 1} \frac{\ln(1/\mu)}{1-\mu} = 1$, means the best possible universal constant c to achieve bias $c \cdot p \cdot \mathbb{V}[f(S)]$ through p-tampering is at most $c \leq 1$. For the special case of *Boolean* function $f(\cdot)$, or alternatively when the p-tampering attacker is allowed to run in *exponential time*, [Mahloujifar and Mahmoody, 2017b] achieved almost optimal bias of $\frac{p}{1+p\cdot\mu-p} \cdot \mathbb{V}[f(S)] > p \cdot \mathbb{V}[f(S)]$. Using their biasing attacks, [Mahloujifar and Mahmoody, 2017b] directly obtained p-tampering targeted poisoning attacks with related bounds. Therefore, a main question that remained open after [Mahloujifar and Mahmoody, 2017b] and is a subject of our study is the following. What is the maximum possible bias of real-valued functions through p-tampering attacks? Resolving this question, directly leads to improved p-tampering poisoning attacks against learners, when the loss function is real-valued.

Example 1.1.1. As an example, consider a regression problem where a regressor L tries to learn a bounded function $c\colon \mathcal{X} \to [0,1]$ through another function $h\colon \mathcal{X} \to [0,1]$. Let x be a target point and the adversary tries to increase the absolute error loss of h over x. Assume that the average of loss over x is 0.2 and the variance of loss over x is 0.1. According to the biasing attack of [Mahloujifar and Mahmoody, 2017b] if an adversary can control each training example with probability $p = 0.5$ then she can increase the the average loss over x to ≈ 0.21.

1.1.1 Summary of Results

Improved p-tampering biasing attacks. Our main technical result in this section is to improve the efficient (polynomial-time) p-tampering biasing attack of [Mahloujifar and Mahmoody, 2017b] to achieve the bias of $\frac{p}{1+p\cdot\mu-p} \cdot \mathbb{V}[f(S)] \geq p \cdot \mathbb{V}[f(S)]$ (where $\mu = \mathbb{E}[f(S)]$ for $S \equiv D^n$ and $\mathbb{V}[\cdot]$ is the variance) in *polynomial time* and for *real-valued* bounded functions with output in $[0,1]$ (see Theorem 1.3.1). This main result

[5] In the original version a slightly stronger bound of $\frac{2p}{3+4p} \cdot \mathbb{V}[f(S)]$ was claimed, though the full version [Mahloujifar and Mahmoody, 2017a] corrected this to the weaker bound $\frac{p}{3+5p} \cdot \mathbb{V}[f(S)]$

immediately allows us to get improved polynomial-time targeted p-tampering attacks against learners for scenarios where the loss function is not Boolean (see Corollary 1.3.2). As in [Mahloujifar and Mahmoody, 2017b], our attacks apply to any learning problem P and any learner L for P as long as L has a non-zero initial error over a specific test example d.

Example 1.1.2. To see the gap between our result and previous work of [Mahloujifar and Mahmoody, 2017b] we go back to the setting of Example 1.1.1. Namely, if the average loss over x is 0.2 and the variance of loss over x is 0.1, then the adversary by running the biasing attack of our work can increase the average loss to more than 0.28. Compare this with 0.21 that the attack of previous work could achieve in the same setting.

Special case of p-resetting attacks. The biasing attack of [Mahloujifar and Mahmoody, 2017b] has an extra property that for each input block (or training example) d_i, if the adversary gets to tamper with d_i, it either does not change d_i at all, or it simply 'resets' it by resampling it from the original (training) distribution D. In this section, we refer to such limited forms of p-tampering attacks as *p-resetting* attacks. Interestingly, p-resetting attacks were previously studied in the work of Bentov, Gabizon, and Zuckerman [Bentov et al., 2016] in the context of (ruling out) extracting uniform randomness from Bitcoin's blockchain [Nakamoto, 2008] when the adversary controls p fraction of the computing power.[6] Bentov, et al. [Bentov et al., 2016] showed how to achieve bias $p/12$ when the original (untampered) distribution D is uniform and the function f is Boolean and balanced.[7] As a special case of p-tampering attacks, p-resetting attacks have interesting properties that are not present in general p-tampering attacks. For example, if an attacker chooses its adversarial examples from a large pool by "skipping" some of them, then p-resetting attacks need a pool of about $\approx (1+p)\cdot n$, while p-tampering attackers might need much more. That is because, for each tampered example, the adversary simply needs to choose one out of two original correctly labeled examples, while a p-tampering attacker might need more samples. Motivated by special applications of p resetting attacks and the special properties of p-resetting attacks, in this section we also study such attacks over arbitrary block distributions D and achieve bias of at least $\frac{p}{1+p\cdot\mu}\cdot\mathbb{V}[f(S)]$, improving the bias of $\frac{p}{3+5p}\cdot\mathbb{V}[f(S)]$ proved in [Mahloujifar and Mahmoody, 2017b].

PAC learning under p-tampering. We also study the power of p-tampering (and p-resetting) attacks in the *non-targeted* setting where the adversary's goal is simply to increase the risk of the generated hypothesis.[8]

[6] To compare the terminologies, the work of [Bentov et al., 2016] studies *p-resettable* sources of randomness, while here we study p-resetting attackers that generate such sources.

[7] The running time of the p-resetting attacker of [Bentov et al., 2016] was $\mathrm{poly}(n, 2^{|D|})$ where $|D|$ is the length of the binary representation of any $d \leftarrow D$. In contrast, our p-resetting attacks run in time $\mathrm{poly}(n, |D|)$.

[8] In the targeted setting, the ε parameter of (ε, δ)-PAC learning goes away, due to the pre-selection of the target test.

In this setting, it is indeed meaningful to study the possibility (or impossibility) of PAC learning, as the test example is chosen at random. We show that in this model, p-tampering attacks cannot prevent PAC learnability for 'realizable' settings; that is when there is always a hypothesis consistent with the training data (see Theorem 1.4.5). We further go beyond p-tampering attacks and study PAC learning under more powerful adversaries who might *choose* the location of training examples that are tampered with but are still limited to choose $\leq p \cdot n$ such examples. We show that PAC learning under such adversaries depends on whether the adversary makes its tampering choices *before* or *after* getting to see the original sample d_i. We call these two class of attacks, respectively, weak and strong p-budget tampering attacks (see Definition 1.4.4).

Comparison with classical models of malicious and nasty noise. Our notion of strong p-budget tampering is inspired by notions of adaptive corruption [Canetti et al., 1996b] and particularly *strong* adaptive corruption [Goldwasser et al., 2015a] studied in cryptographic contexts. Furthermore, p-tampering and p-budget tampering can be seen as analogues of malicious noise and nasty noise respectively, where the adversary shall respect the correct label of the perturbed instance. One subtle difference is that, the nasty noise model of Bshouty et. al [Bshouty et al., 2002] allows the adversary to see the *whole* training set before tampering with a small fraction of it. However, we note that we *improve* their negative result (i.e., their lower bound for PAC learning in their nasty noise model) by showing the impossibility of PAC learning even if the adversary is limited to correct labels, and even if it is "online" the same way Valiant's malicious noise and p-tampering noise are modeled. On the other hand, our positive result about PAC learning under p-budget attacks is incomparable to the positive result of Kearns and Li [Kearns and Li, 1993b] and Bshouty et. al [Bshouty et al., 2002]. Our result is weaker in the sense that we require correct labels and we do not allow the adversary to see the next example before deciding to corrupt it (what we call the weak attack model), yet our result is stronger in the sense that we allow much higher noise rate, which is essentially close to one! Our impossibility result of PAC learnability under strong p-budget attacks (see Theorem 1.4.7) shows that PAC learning under 'mistake-free' adversarial noise is *not* always possible.

Finally, we would like to point out that our positive result about PAC learnability under p-tampering attacks (see Theorem 1.4.5) shows a stark contrast between the 'mistake-free' adversarial noise and general malicious noise for $p > 1/2$. Indeed, when the adversary can tamper with $p \approx 1/2$ fraction of the training data in an arbitrary way for a binary classification problem, it can make the training data completely useless by always picking the labels at random from $\{0, 1\}$. Such adversary will end up changing only $p \approx 1/2$ of the examples, but will make the labels independent of the features. However, as we prove in Theorem 1.4.5, PAC learning is possible under p-tampering for any constant $p < 1$.

Applications beyond attacking learners. Similar to how [Mahloujifar and Mahmoody, 2017b] used their biasing attacks in applications other than attacking learners, our new biasing attacks can also be used to obtain improved polynomial-time attacks for biasing the output bit of any (candidate) seedless randomness extractors [Von Neumann, 1951, Chor and Goldreich, Santha and Vazirani, 1986], hence ruling out their existence for bias $o(p)$. Moreover, similarly to [Mahloujifar and Mahmoody, 2017b] we also can obtain blockwise p-tampering (and p-resetting) attacks against security of indistinguishability-based cryptographic primitives (e.g., encryption, secure computation, etc.). Previous attacks of [Dodis et al., 2004] used information theoretic Santha-Vazirani [Santha and Vazirani, 1986] sources with high min-entropy while our attacks similar to [Mahloujifar and Mahmoody, 2017b] are algorithmic and run in polynomial time. We refer to [Mahloujifar and Mahmoody, 2017b] for the statement of the attacks and formal results, and note that our new attacks imply new algorithmic attacks on the same set of primitives. Furthermore, as in [Mahloujifar and Mahmoody, 2017b], our new improved biasing attacks apply to any *joint* distribution (e.g., martingales) when the tampered values affect the random process in an online way. In this section, however, we focus on the case of product distributions as they suffice for getting our attacks against learners and include all the main ideas even for the general case of random processes. We refer the reader to the work of [Mahloujifar and Mahmoody, 2017b] for the statement of these extra applications of p-tampering attacks. Finally, we note that p-tampering is an information theoretic framework (not focused on cryptography) even though it was initially studied in cryptographic contexts. In fact, by using p-tampering in our work in a learning context we confirm the generality of this information theoretic tampering framework. f

Recent positive results achieving algorithmic robustness. On the positive (algorithmic) side, the seminal works of Diakonikolas et al. [Diakonikolas et al., 2016] and Lai et al. [Lai et al., 2016] showed the surprising power of algorithmic robust inference over poisoned data with error that does not depend on the dimension of the distribution (but still depends on the fraction of poisoned data). These works led to an active line of work (e.g., see [Charikar et al., 2017, Diakonikolas et al., 2017, 2018b,a, Prasad et al., 2018, Diakonikolas et al., 2018c] and references therein) exploring the possibility of robust statistics over poisoned data with algorithmic guarantees. The works of [Charikar et al., 2017, Diakonikolas et al., 2018b] performed *list-decodable* learning, and [Diakonikolas et al., 2018a, Prasad et al., 2018] studied supervised learning. In our attacks, however, similarly to virtually all attacks in the literature (over specific learners and models) we demonstrate inherent power of poisoning attacks (that apply to *any* learner and hypothesis class) to *amplify* the error of classifiers starting from small and perhaps acceptable error rates, while after the attack the error probability is essentially one. Namely, our results show that in order to resist poisoning attacks, the same algorithms should do much better in the no-attack setting, as otherwise a poisoning attacker can increase the

targeted error probability significantly.

Implications to (impossibility of) computational robustness. Previously [Mahloujifar and Mahmoody, 2019b, Garg et al., 2019] it was asked whether security of learning algorithms can leverage computational hardness by making successful attacks run in exponential (or at least super-polynomial) time over the input length, making them infeasible in practice. Since our attacks run in polynomial time, as a result, we conclude a stronger negative result. In particular, our results show that there is no way to beat our impossibility results by modeling the adversary as an efficiently bounded entity and rely on computational intractably assumptions to prove the security.

Comparison with related work. In this section, our attacks work in the p-tampering poisoning model, in which an attacker get to tamper with each training example with *independent* probability p, and the adversary is also limited to use only correct labels. Here we show how to increase the probability of a bad predicate over the hypothesis by $\Omega(p)$. In [Mahloujifar and Mahmoody, 2019b, Etesami et al., 2019b] it was shown shown that if the adversary can *choose* the location of the tampered examples, then it would have much more power. Namely, in that case an adversary who changes only $\widetilde{O}(\sqrt{m})$ of the training examples, where m is the size of the training set, can increase the probability of any bad event from any non-negligible probability $\Omega(1/\operatorname{poly}(m)$ to ≈ 1. The Our results are incomparable to the ones above, as our adversary does not choose the location of the tampering, while the increase in the probability of bad event is more in [Mahloujifar and Mahmoody, 2019b, Etesami et al., 2019b].

Ideas behind our new biasing attacks and our approach

Our new biasing attacks build upon ideas developed in previous work [Reingold et al., 2004, Dodis et al., 2004, Beigi et al., 2017, Dodis and Yao, 2015, Bentov et al., 2016] in the context of attacking deterministic randomness extractors from the so called Santha-Vazirani sources [Santha and Vazirani, 1986]. In [Mahloujifar and Mahmoody, 2017b] the authors generalized the idea of 'half-space' sources (introduced in [Reingold et al., 2004, Dodis et al., 2004]) to real-valued functions, using which they showed how to find p-tampering biasing attacks with bias $\frac{p}{1+p\cdot\mu-p}\cdot\mathbb{V}[f(S)]$. However, their attacks need *inefficient* (i.e., super polynomial time) tampering algorithms. In particular, [Mahloujifar and Mahmoody, 2017b] directly defined a perturbed joint distribution $\widetilde{S}=(\widetilde{D}_1,\ldots,\widetilde{D}_n)$ of the original product distribution $S\equiv D$ such that has two properties hold: (1) $\mathbb{E}[f(\widetilde{S})]$ achieves the desired bias, and (2) $\Pr[\widetilde{S}=z]\leq c\cdot\Pr[S=z]$ for all points z and sufficiently small constant c, meaning that $\widetilde{S}$ does not increase the point-wise probabilities "too much". It was shown in [Mahloujifar and Mahmoody, 2017b] that the second property guarantees that the distribution $\widetilde{S}$ can

be obtained from S by *some* tampering algorithm, but their proof was existential, namely it said nothing about the computational complexity of such tampering algorithm. Achieving the same bias *efficiently* for *real-valued* functions is the main technical challenge in this section.

Our approach. At a very high level, we show how to achieve in *polynomial-time* the same bias achieved in [Mahloujifar and Mahmoody, 2017b] through the following two steps.

1. We first show how to obtain the same exact *final* distribution achieved in [Mahloujifar and Mahmoody, 2017b] through *local* p-tampering decisions that could be implemented in polynomial time using an idealized oracle $\hat{f}[\cdot]$ that provides certain information about function $f(\cdot)$.

2. We then, show that the idealized oracle $\hat{f}[\cdot]$ can be approximated in polynomial time, and more importantly, the p-tampering attack of the previous step (using idealized oracle $\hat{f}[\cdot]$) is robust to this approximation and still achieves almost the same bias.

Idealized oracle $\hat{f}[\cdot]$. Let $d_{\leq i} = (d_1, \ldots, d_i)$ be the first i blocks given as input to a function f.[9] Now, suppose the adversary gets the chance to determine the next block d_{i+1} based on its knowledge of the previously generated blocks $(d_1, \ldots, d_i)$. We achieve the goal of the first step depicted above, with the help of the following oracle provided for free to the p-tampering attacker.

$$\hat{f}[d_{\leq i}] = \mathop{\mathbb{E}}_{d_{i+1},\ldots,d_n \leftarrow D^{n-i}}[f(d_1, \ldots, d_n)].$$

In other words, $\hat{f}[d_{\leq i}]$ computes the expected value of f when each of the blocks (examples) $d_{i+1}, \ldots, d_n$ is drawn independent and identically distributed according to D, while the first i blocks $d_1, \ldots, d_i$ are fixed as dictated by $d_{\leq i}$.

Although the partial averages $\hat{f}[d_{\leq i}]$ are not *exactly* computable in polynomial time, they can indeed be efficiently approximated within arbitrary small additive error. As we show, our attacks are also robust to such approximations, and using the approximations of $\hat{f}[d_{\leq i}]$ (rather than their exact values) we can still bound the bias. See Sections 1.3.2 and Section 1.3.3 for the details.

The case of p-resetting attacks. When it comes to p-resetting attacks, we cannot achieve the same bias that we do achieve through general p-tampering attacks. However, we still use the same recipe as described above. Namely, we use the idealized oracle $\hat{f}[d_{\leq i}]$ to make careful local sampling to keep or reset a given

[9] Alternatively the first i training examples, when we attack learners. However, some of the blocks in $(d_1, \ldots, d_i)$ might be the result of previous tampering decisions.

block d_i, so that the final distribution has the desired bias. We then approximate the idealized oracle while arguing that the analysis is robust to this change.

Comparison with the polynomial-time attacker of [Mahloujifar and Mahmoody, 2017b]. As mentioned before, the work of [Mahloujifar and Mahmoody, 2017b] also provides polynomial p-tampering attacks with weaker bounds. At a high level, the attacks of [Mahloujifar and Mahmoody, 2017b] were simple to describe (without using the idealized oracle $\hat{f}$), while their analyses were extremely complicated and used the function $\hat{f}$ as well as a carefully chosen potential function based on ideas from [Austrin et al., 2014a] in which authors presented a p-tampering biasing attack for the special case of uniform Boolean blocks (i.e., $D \equiv U_1$). Our new (polynomial time) attacks takes a dual approach: the analysis of our attacks are conceptually simpler, as they directly achieve the desired bias, but the description of our attacks are more complicated as they also depend on the idealized oracle $\hat{f}$.

1.2 Preliminaries

Notation. We use calligraphic letters (e.g., $\mathcal{D}$) for sets and capital non-calligraphic letters (e.g., D) for distributions. By $d \leftarrow D$ we denote that d is sampled from D. Usually D denotes the joint distribution over pairs (x, y) in which x is an instance and y is its label. By $D \in \mathcal{S}$ we denote that D always outputs in $\mathcal{S}$, namely $\text{Supp}(D) \subseteq \mathcal{S}$. By $T^D(\cdot)$ we denote an algorithm $T(\cdot)$ with oracle access to a sampler for D. By D^n we denote n independent identically distributed samples from D. By $\varepsilon(n) \leq \frac{1}{\text{poly}(n)}$ we mean $\varepsilon(n) \leq \frac{1}{n^{\Omega(1)}}$ and by $t(n) \leq \text{poly}(n)$ we mean $t(n) \leq n^{O(1)}$.

An *example* s is a pair $s = (x, y)$ where $x \in \mathcal{X}$ and $y \in \mathcal{Y}$. An example is usually sampled from a distribution D. A *sample* set (or sequence) $\mathcal{S}$ of size n is a set (or sequence) of n examples. A hypothesis h is *consistent* with a sample set (or sequence) $\mathcal{S}$ if and only if $h(x) = y$ for all $(x, y) \in \mathcal{S}$. We assume that instances, labels, and hypotheses are encoded as strings over some alphabet such that given a hypothesis h and an instance x, $h(x)$ is computable in polynomial time.

Definition 1.2.1 (Realizability)**.** We say that the problem $\mathsf{P} = (\mathcal{X}, \mathcal{Y}, \mathcal{D}, \mathcal{H}, \ell oss)$ is realizable, if for all $D \in \mathcal{D}$, there exists an $h \in \mathcal{H}$ such that $\mathsf{Risk}_D(h) = 0$.

We can now define *Probably Approximately Correct (PAC)* learning. Our definition is with respect to a given set of distributions $\mathcal{D}$, and it can be instantiated with one distribution $\{D\} = \mathcal{D}$ to get the distribution-specific case. We can also recover the distribution-independent scenario, whenever the projection of $\mathcal{D}$ over $\mathcal{X}$ covers all distributions.

Definition 1.2.2 (PAC Learning)**.** A realizable problem $\mathsf{P} = (\mathcal{X}, \mathcal{Y}, \mathcal{D}, \mathcal{H}, \ell oss)$ is (ε, δ)-PAC learnable if there is a (possibly randomized) learning algorithm L such that for every n and every $D \in \mathcal{D}$, it holds that,

$$\Pr_{S \leftarrow D^n, h \leftarrow L(S)}[\mathsf{Risk}_D(h) \leq \varepsilon(n)] \geq 1 - \delta(n).$$

We call P simply PAC learnable if $\varepsilon(n), \delta(n) \leq 1/\operatorname{poly}(n)$, and we call it *efficiently* PAC learnable if, in addition, L is running in polynomial time.

Definition 1.2.3 (Average Error of a Test)**.** For a problem $\mathsf{P} = (\mathcal{X}, \mathcal{Y}, \mathcal{D}, \mathcal{H}, \ell oss)$, a (possibly randomized) learning algorithm L, a fixed test sample $(x, y) = d \leftarrow D$ for some distribution S over $\operatorname{Supp}(D)^n$ (e.g., $S \equiv D^n$) for some $n \in \mathbb{N}$, the *average error*[10] of the test example d (with respect to S, L) is defined as,

$$\mathsf{Err}_{S,L}(d) = \mathop{\mathbb{E}}_{S \leftarrow S, h \leftarrow L(S)}[\ell oss(h(x), y)].$$

We call $\mathsf{Err}_{S,L} = \mathbb{E}_{d \leftarrow D} \mathsf{Err}_{S,L}(d)$ simply the average error. When L is clear from the context, we simply write $\mathsf{Err}_S(d)$ (resp. Err_S) to denote $\mathsf{Err}_{S,L}(d)$ (resp. $\mathsf{Err}_{S,L}$).

It is easy to see that a realizable problem $\mathsf{P} = (\mathcal{X}, \mathcal{Y}, \mathcal{D}, \mathcal{H}, \ell oss)$ with bounded loss function ℓoss is PAC learnable if and only if there is a learner L (for P) such that its average error Err_S is bounded by a fixed $1/\operatorname{poly}(n)$ function for all $D \in \mathcal{D}$.[11]

Poisoning attacks. PAC learning under adversarial noise is already defined in the literature, however, poisoning attacks include broader classes of attacks. For example, a poisoning adversary might *add* adversarial examples to the training data (thus, increasing its size) or *remove* some of it adversarially. A more powerful form of poisoning attack is the so called *targeted* poisoning attack where the adversary gets to know the target test example before poisoning the training examples. More formally, suppose $\mathcal{S} = (d_1, \ldots, d_n)$ is the training examples sampled independently and identically distributed from $D \in \mathcal{D}$. For a poisoning attacker A, by $\widetilde{\mathcal{S}} \leftarrow \mathsf{A}(\mathcal{S})$ we denote the process through which A generates an adversarial training set $\widetilde{\mathcal{S}}$ based on $\mathcal{S}$. Note that, this notation does not specify the exact limitations of how A is allowed to tamper with $\mathcal{S}$, and that is part of the definition of A. In the targeted case, the adversary A is also given a test example $(x, y) = d \leftarrow D$. So, we would denote this by writing $\widetilde{\mathcal{S}} \leftarrow \mathsf{A}(d, \mathcal{S})$ to emphasize that d is the test example given as input to A. We use calligraphic $\mathcal{A}$ to denote a *class* of attacks. Note that a particular adversary $\mathsf{A} \in \mathcal{A}$ might try to poison a training set $\mathcal{S}$ *based* on the knowledge of a problem $\mathsf{P} = (\mathcal{X}, \mathcal{Y}, \mathcal{D}, \mathcal{H}, \ell oss)$. On the other hand,

[10] The work [Mahloujifar and Mahmoody, 2017b] called the same notion the 'cost' of d.

[11] Suppose $\ell oss(\cdot)$ is bounded (i.e., always in $[0, 1]$). If P is (ε, δ)-PAC learnable, then by a union bound, $\mathsf{Err}_S \leq \varepsilon + \delta$. Moreover, if L is *not* (ε, δ)-PAC learnable, then its average error is at least $\varepsilon \cdot \delta$. This means that if L has average error $\gamma = \mathsf{Err}_S$, then L is an $(\sqrt{\gamma}, \sqrt{\gamma})$-PAC learner as well.

because sometimes we would like to limit the adversary's power based on the specific distribution D (e.g., by always choosing tampered data to be in $\mathrm{Supp}(D)$), by $\mathcal{A}_D \subseteq \mathcal{A}$ we denote the adversary class for a particular distribution D.

Definition 1.2.4 (Learning under poisoning)**.** Suppose $\mathsf{P} = (\mathcal{X}, \mathcal{Y}, \mathcal{D}, \mathcal{H}, \ell oss)$ is a problem, $\mathcal{A} = \cup_{D\in\mathcal{D}}\mathcal{A}_D$ is an adversary class, and L is a (possibly randomized) learning algorithm for P.

- **PAC learning under poisoning.** For problem P, L is an (ε, δ)-PAC learning algorithm for P under poisoning attacks of $\mathcal{A}$, if for every $D \in \mathcal{D}, n \in \mathbb{N}$, and every adversary $\mathsf{A} \in \mathcal{A}_D$,

$$\Pr_{\mathcal{S}\leftarrow D^n, \widetilde{\mathcal{S}}\leftarrow \mathsf{A}(\mathcal{S}), h\leftarrow L(\widetilde{\mathcal{S}})}[\mathsf{Risk}_D(h) \le \varepsilon(n)] \ge 1 - \delta(n).$$

  PAC learnability and efficient PAC learnability are then defined similarly to Definition 1.2.2.

- **Average error under targeted poisoning.** If $\mathcal{A}$ contains *targeted* poisoning attackers, for a distribution $D \in \mathcal{D}$ and an attack $\mathsf{A} \in \mathcal{A}_D$, the *average error* $\mathsf{Err}^{\mathsf{A}}_{D^n}(d)$ for a test example $d = (x, y)$ under poisoning attacker A is equal to $\mathsf{Err}_{\widetilde{S}}(d)$ where $\widetilde{S} \equiv \mathsf{A}(d, S)$ for $S \equiv D^n$.

***p*-tampering attacks.** We now define the specific class of poisoning attacks studied in this section. Informally speaking, p-tampering attacks model attackers who will manipulate the training sequence $\mathcal{S} = (d_1, \ldots, d_n)$ in an *online* way, meaning while tampering with d_i, they do not rely on the knowledge of $d_j, j > i$. Moreover, such attacks get to tamper with d_i only with independent probability p, modeling scenarios where the tampering even is random and outside the adversary's choice. A crucial point about p-tampering attacks is that they always stay in $\mathrm{Supp}(D)$. The formal definition follows.

Definition 1.2.5 (p-tampering/resetting attacks)**.** The class of p-tampering attacks $\mathcal{A}^p_{\mathrm{tam}} = \cup_{D\in\mathcal{D}}\mathcal{A}_D$ is defined as follows. For a distribution $D \in \mathcal{D}$, any $\mathsf{A} \in \mathcal{A}_D$ has a (potentially randomized) tampering algorithm Tam such that (1) given oracle access to D, $\mathsf{Tam}^D(\cdot) \in \mathrm{Supp}(D)$, and (2) given any training sequence $\mathcal{S} = (d_1, \ldots, d_n)$, the tampered $\widetilde{\mathcal{S}} = (\widetilde{d}_1, \ldots, \widetilde{d}_n)$ is generated by A inductively (over $i \in [n]$) as follows.

- With probability $1 - p$, let $\widetilde{d}_i = d_i$.
- Otherwise, (with probability p), get $\widetilde{d}_i \leftarrow \mathsf{Tam}^D(1^n, \widetilde{d}_1, \ldots, \widetilde{d}_{i-1}, d_i)$.

The class of *p-resetting* attacks $\mathcal{A}^p_{\mathrm{res}} \subset \mathcal{A}^p_{\mathrm{tam}}$ include special cases of p-tampering attacks where the tampering algorithm Tam is restricted as follows. Either $\mathsf{Tam}(1^n, \widetilde{d}_1, \ldots, \widetilde{d}_{i-1}, d_i)$ outputs d_i, or otherwise, it will output a *fresh* sample $d'_i \leftarrow D$. In the *targeted* case, the adversary A_D and its tampering algorithm Tam are also given the final test example $d_0 \leftarrow D$ as extra input (that they can read but not tamper with). An attacker A_D is called *efficient*, if its oracle-aided tampering algorithm Tam^D runs in polynomial time.

Subtle aspects of the definition. Even though one can imagine a more general definition for tampering algorithms, in all the attacks of [Mahloujifar and Mahmoody, 2017b] and the attacks of this section, the tampering algorithms do *not* need to know the original un-tampered values $d_1, \ldots, d_{i-1}$. Since our goal here is to design p-tampering attacks, we use the simplified definition above, while all of our positive results still hold for the stronger version in which the tampering algorithm is given the full history of the training examples. Another subtle issue is about whether d_i is needed to be given to the tampering algorithm. As already noted in [Mahloujifar and Mahmoody, 2017b], when we care about p-tampering distributions of D^n, d_i is not necessary to be given to the tampering algorithm Tam, as Tam can itself sample a copy from D and treat it like d_i. Therefore the 'stronger' form of such attacks (where d_i is given) is equivalent to the 'weaker' form where d_i is not given. In fact, if D is samplable in polynomial time, then this equivalence holds with respect to efficient adversaries (with efficient Tam algorithm) as well. In this section, for both p-tampering and p-resetting attacks we choose to always give d_i to Tam. Interestingly, as we will see in Section 1.4, if the adversary can *choose* the $p \cdot n$ locations of tampering, the weak and strong attackers will have different powers!

1.2.1 Concentration Bounds

Lemma 1.2.6 (Hoeffding inequality [Hoeffding, 1963])**.** *Let $X_1, \ldots, X_n$ be n independent random variables where $\mathrm{Supp}(X_i) \subseteq [0,1]$ for all $i \in [n]$. Let $X = \frac{1}{n}\sum_{i=1}^{n} X_i$ and $\lambda = \mathbf{E}[X]$. Then, for any $\xi \geq 0$,*

$$\Pr[|X - \lambda| \geq \xi] \leq 2\mathrm{e}^{-2n\xi^2} .$$

Lemma 1.2.7 (Chernoff Bound [Chernoff, 1952])**.** *Let $X_1, \ldots, X_n$ be n independent Boolean random variables, $\mathrm{Supp}(X_i) \subseteq \{0,1\}$ for all $i \in [n]$. Let $X = \frac{1}{n}\sum_{i=1}^{n} X_i$ and $\lambda = \mathbf{E}[X]$. Then, for any $\gamma \in [0,1]$,*

$$\Pr[X \geq (1+\gamma)\cdot\lambda] \leq \mathrm{e}^{-n\cdot\lambda\cdot\gamma^2/3} ,$$

1.3 Improved p-Tampering and p-Resetting Poisoning Attacks

In this section we study the power of p-tampering attacks in the targeted setting and improve upon the p-tampering and p-resetting attacks of [Mahloujifar and Mahmoody, 2017b]. Our main tool is the following theorem giving new improved p-tampering and p-resetting attacks to bias the output of bounded real-valued functions.

1.3.1 The Statement of Results

Theorem 1.3.1 (Improved biasing attacks)**.** *Let D be any distribution, $S \equiv D^n$, and $f\colon \mathrm{Supp}(S) \to [0,1]$. Suppose $\mu = \mathbb{E}[f(S)]$ and $\nu = \mathbb{V}[f(S)]$ be the expected value and the variance of $f(S)$ respectively. For every constant $p \in (0,1)$, and a given parameter $\xi \in (0,1)$, the following holds.*

1. *There is a p-tampering attack $\mathsf{A}_{\mathrm{tam}}$ such that,*

$$\mathop{\mathbb{E}}_{\widetilde{\mathcal{S}} \leftarrow \mathsf{A}_{\mathrm{tam}}(S)} [f(\widetilde{\mathcal{S}})] \geq \mu + \frac{p \cdot \nu}{1 + p \cdot \mu - p} - \xi$$

   and given oracle access to f and sampling oracle for D, the tampering algorithm $\mathsf{Tam}_{\mathrm{tam}}^{D,f}$ of $\mathsf{A}_{\mathrm{tam}}$ could be implemented in time $\mathrm{poly}(|D| \cdot n/\xi)$ where $|D|$ is the bit length of $d \leftarrow D$.

2. *There is a p-resetting attack $\mathsf{A}_{\mathrm{res}}$ such that,*

$$\mathop{\mathbb{E}}_{\widetilde{\mathcal{S}} \leftarrow \mathsf{A}_{\mathrm{res}}(S)} [f(\widetilde{\mathcal{S}})] \geq \mu + \frac{p \cdot \nu}{1 + p \cdot \mu} - \xi$$

   and given oracle access to f and sampling oracle for D, the tampering algorithm $\mathsf{Tam}_{\mathrm{res}}^{D,f}$ of $\mathsf{A}_{\mathrm{res}}$ could be implemented in time $\mathrm{poly}(|D| \cdot n/\xi)$ where $|D|$ is the bit length of $d \leftarrow D$.

See Section 1.3.2 for the full proof of Theorem 1.3.1. In this section, we use Theorem 1.3.1 and obtain the following improved attacks in the targeted setting against any learner. In particular, for any fixed $(x,y) = d \leftarrow D$, the following corollary follows from Theorem 1.3.1 by letting $f(\mathcal{S}) = \mathbb{E}_{h \leftarrow L(\mathcal{S})}[\ell oss(h(x), y)]$.

Corollary 1.3.2 (Improved targeted p-tampering attacks)**.** *Given a problem $\mathsf{P} = (\mathcal{X}, \mathcal{Y}, \mathcal{D}, \mathcal{H}, \ell oss)$ with a bounded loss function ℓoss, for any distribution $D \in \mathcal{D}$, test example $(x,y) = d \leftarrow D$, learner L, and $n \in \mathbb{N}$, let $\mu = \mathsf{Err}_D(d)$ be the average error for d, and let,*

$$\nu = \mathop{\mathbb{V}}_{\mathcal{S} \leftarrow D^n} \Big[\mathop{\mathbb{E}}_{h \leftarrow L(\mathcal{S})} [\ell oss(h(x), y)] \Big].$$

Then, for any constant $0 < p < 1$, and any $0 < \xi < 1$ there is a p-tampering (resp. p-resetting) attack $\mathsf{A}_{\mathrm{tam}}$ (resp. $\mathsf{A}_{\mathrm{res}}$) that increases the average error by $\frac{p \cdot \nu}{1 + p \cdot \mu - p} - \xi$ (resp. $\frac{p \cdot \nu}{1 + p \cdot \mu} - \xi$). Moreover, if D is polynomial-time samplable and both functions $f, \ell oss$ are polynomial-time computable, then $\mathsf{A}_{\mathrm{tam}}, \mathsf{A}_{\mathrm{res}}$ could be implemented in $\mathrm{poly}(|D| \cdot n/\xi)$ time.

Remark 1.3.3. Even when the average error $\mu = \mathsf{Err}_D(d)$ is not too small, the variance ν (as defined in Corollary 1.3.2) could be negligible in general. However, for natural cases this cannot happen. For example,

if the loss function $\ell oss(\cdot)$ is Boolean (e.g., P is a classification problem) and if L is a deterministic learning algorithm, then $\nu = \mu \cdot (1-\mu)$.

We now demonstrate the power of p-tampering and p-resetting attacks on PAC learners by using them to increase the failure probability of deterministic PAC learners.

Corollary 1.3.4 (p-tampering attacks on PAC learners)**.** *Given a problem* $\mathsf{P} = (\mathcal{X}, \mathcal{Y}, \mathcal{D}, \mathcal{H}, \ell oss)$*,* $D \in \mathcal{D}, n \in \mathbb{N}$*, and deterministic learner* L*, suppose,*

$$\Pr_{\mathcal{S} \leftarrow D^n,\ h = L(\mathcal{S})}[\mathsf{Risk}_D(h) \geq \varepsilon] = \delta.$$

Then, there is a $\mathrm{poly}(|D| \cdot n/\varepsilon)$ *time* p*-tampering attack* $\mathsf{A}_{\mathrm{tam}}$ *and a* p*-resetting attack* $\mathsf{A}_{\mathrm{res}}$ *such that,*

$$\Pr_{\mathcal{S} \leftarrow D^n, \widetilde{\mathcal{S}} \leftarrow \mathsf{A}_{\mathrm{tam}}(\mathcal{S}), h = L(\widetilde{\mathcal{S}})}[\mathsf{Risk}_D(h) \geq 0.99 \cdot \varepsilon] \geq \delta + \frac{p \cdot (\delta - \delta^2)}{1 + p \cdot \delta - p} - \mathrm{e}^{-n}$$
$$\Pr_{\mathcal{S} \leftarrow D^n, \widetilde{\mathcal{S}} \leftarrow \mathsf{A}_{\mathrm{res}}(\mathcal{S}), h = L(\widetilde{\mathcal{S}})}[\mathsf{Risk}_D(h) \geq 0.99 \cdot \varepsilon] \geq \delta + \frac{p \cdot (\delta - \delta^2)}{1 + p \cdot \delta} - \mathrm{e}^{-n}.$$

Before proving this we prove a useful proposition.

Proposition 1.3.5. *The following functions are increasing for* $\delta \in [0,1]$ *and any constant* $p \in (0,1)$*.*

$$\gamma_{\mathrm{tam}}(\delta) = \delta + \frac{p \cdot (\delta - \delta^2)}{1 + p \cdot \delta - p}, \qquad \gamma_{\mathrm{res}}(\delta) = \delta + \frac{p \cdot (\delta - \delta^2)}{1 + p \cdot \delta}.$$

Proof. The lemma holds because we have,

$$\frac{\partial \gamma_{\mathrm{tam}}}{\partial \delta} = \frac{1-p}{(p(\delta-1)+1)^2} > 0 \quad \text{and} \quad \frac{\partial \gamma_{\mathrm{res}}}{\partial \delta} = \frac{1+p}{(p \cdot \delta + 1)^2} > 0\,.$$

□

Proof of Corollary 1.3.4. The inefficient versions of the attacks follow from Theorem 1.3.1 by letting $f(\mathcal{S}) = 1$ if $\mathsf{Risk}_D(h) \geq \varepsilon$ and $f(\mathcal{S}) = 0$ otherwise. When the attacks are supposed to run in polynomial time, we have to approximate $\mathsf{Risk}_D(h)$ using oracle access to D. Suppose we have access to some oracle $\tilde{f}(.)$ such that,

$$\tilde{f}(\mathcal{S}) = \begin{cases} 1 & \text{if } \mathsf{Risk}_D(L(\mathcal{S})) \geq \varepsilon, \\ 0 & \text{if } \mathsf{Risk}_D(L(\mathcal{S})) \leq 0.99 \cdot \varepsilon, \\ 0 \text{ or } 1 & \text{if } 0.99 \cdot \varepsilon \leq \mathsf{Risk}_D(L(\mathcal{S})) \leq \varepsilon. \end{cases}$$

We first show that by using the oracle $\tilde{f}(.)$ instead of $f(\cdot)$, we can achieve the desired bias, and then we will approximate $f(\cdot)$ using oracle access to a sampling oracle for D such that we obtain a simulated oracle for $\tilde{f}(.)$ with probability $1 - \mathrm{e}^{-n}$.

If $\tilde{\delta} = \mathbb{E}_{\mathcal{S} \leftarrow D^n}[\tilde{f}(\mathcal{S})]$, then Theorem 1.3.1 shows that given oracle access to $\tilde{f}(.)$, there is a p-tampering attack $\mathsf{A}_{\mathrm{tam}}$ and a p-resetting attack $\mathsf{A}_{\mathrm{res}}$ that can bias the average of $\tilde{f}$ as,

$$\begin{cases} \mathbb{E}_{\mathcal{S} \leftarrow D^n, \widetilde{\mathcal{S}} \leftarrow \mathsf{A}_{\mathrm{tam}}(\mathcal{S})}[\tilde{f}(\hat{\mathcal{S}})] \geq \tilde{\delta} + p \cdot \frac{p \cdot (\tilde{\delta} - \tilde{\delta}^2)}{1 + p \cdot \tilde{\delta} - p}, \text{or} \\ \mathbb{E}_{\mathcal{S} \leftarrow D^n, \widetilde{\mathcal{S}} \leftarrow \mathsf{A}_{\mathrm{res}}(\mathcal{S})}[\tilde{f}(\hat{\mathcal{S}})] \geq \tilde{\delta} + p \cdot \frac{p \cdot (\tilde{\delta} - \tilde{\delta}^2)}{1 + p \cdot \tilde{\delta}}. \end{cases}$$

On the other hand, we know that for all $\mathcal{S} \in \mathrm{Supp}(D^n)$, $f(\mathcal{S}) \leq \tilde{f}(\mathcal{S})$. Therefore,

$$\delta = \mathop{\mathbb{E}}_{\mathcal{S} \leftarrow D^n}[f(\mathcal{S})] \leq \tilde{\delta}.$$

We also know that $\tilde{f}(\mathcal{S}) = 1$ implies that $\mathsf{Risk}(L(\mathcal{S})) \geq 0.99 \cdot \varepsilon$, thus for any distribution Z defined on $\mathrm{Supp}(D^n)$ we have,

$$\mathop{\mathbb{E}}_{\widetilde{\mathcal{S}} \leftarrow Z}[\tilde{f}(\hat{\mathcal{S}})] \leq \Pr_{\widetilde{\mathcal{S}} \leftarrow Z}[\mathsf{Risk}(L(\hat{\mathcal{S}})) \geq 0.99 \cdot \varepsilon].$$

Combining the above inequalities for the p-tampering attack, we get,

$$\begin{aligned} \Pr_{\mathcal{S} \leftarrow D^n, \widetilde{\mathcal{S}} \leftarrow \mathsf{A}_{\mathrm{tam}}(\mathcal{S})}[\mathsf{Risk}(L(\hat{\mathcal{S}})) \geq 0.99 \cdot \varepsilon] &\geq \mathop{\mathbb{E}}_{\mathcal{S} \leftarrow D^n, \widetilde{\mathcal{S}} \leftarrow \mathsf{A}_{\mathrm{tam}}(\mathcal{S})}[\tilde{f}(\hat{\mathcal{S}})] \\ &\geq \hat{\delta} + p \cdot \frac{p \cdot (\tilde{\delta} - \tilde{\delta}^2)}{1 + p \cdot \tilde{\delta} - p} \\ \text{(By Proposition 1.3.5)} \quad &\geq \delta + p \cdot \frac{p \cdot (\delta - \delta^2)}{1 + p \cdot \delta - p}. \end{aligned}$$

Similarly, for the p-resetting attack we get,

$$\begin{aligned} \Pr_{\mathcal{S} \leftarrow D^n, \widetilde{\mathcal{S}} \leftarrow \mathsf{A}_{\mathrm{res}}(\mathcal{S})}[\mathsf{Risk}(L(\hat{\mathcal{S}})) \geq 0.99 \cdot \varepsilon] &\geq \mathop{\mathbb{E}}_{\mathcal{S} \leftarrow D^n, \widetilde{\mathcal{S}} \leftarrow \mathsf{A}_{\mathrm{res}}(\mathcal{S})}[\tilde{f}(\hat{\mathcal{S}})] \\ &\geq \hat{\delta} + p \cdot \frac{p \cdot (\tilde{\delta} - \tilde{\delta}^2)}{1 + p \cdot \tilde{\delta}} \\ \text{(By Proposition 1.3.5)} \quad &\geq \delta + p \cdot \frac{p \cdot (\delta - \delta^2)}{1 + p \cdot \delta}. \end{aligned}$$

Now, we show how to obtain an oracle $\tilde{f}(.)$ that provides the properties above with high probability by accessing sampling oracle for D. The simulated oracle $\tilde{f}(.)$ works as follows. Given a training set $\mathcal{S}$, it first performs L on $\mathcal{S}$ to get the hypothesis h. Then it samples m examples $d_1 = (x_1, y_1), \ldots, d_m = (x_m, y_m)$ from D^m, for

m to be chosen later, and it computes an "empirical risk" $r(h)$ as follows: $r(h) = \frac{1}{m}\sum_{i=1}^{m} \ell oss(h, x_i, y_i)$. If $r(h) \geq 0.995$, $\tilde{f}(\mathcal{S})$ outputs 1, otherwise it outputs 0. By Hoeffding's inequality, it holds that,

$$\Pr[|r(h) - \mathsf{Risk}_D(h)| \geq 0.005 \cdot \varepsilon] \leq 2 \cdot \mathrm{e}^{-\frac{m \cdot \varepsilon^2}{20000}}.$$

Therefore,

$$\Pr[((r(h) \leq 0.995 \cdot \varepsilon) \wedge (\mathsf{Risk}_D(h) \geq \varepsilon)) \vee ((r(h) \geq 0.995 \cdot \varepsilon) \wedge (\mathsf{Risk}_D(h) \leq 0.99 \cdot \varepsilon))]$$

is upper bounded by the quantity

$$2 \cdot \mathrm{e}^{-\frac{m \cdot \varepsilon^2}{20000}},$$

which means that the oracle $\tilde{f}(.)$ has the required properties with very high probability. Now, if the original attacker $\mathsf{A}_{\mathrm{tam}}$ or $\mathsf{A}_{\mathrm{res}}$ runs in time $t = \mathrm{poly}(|D| \cdot n/\varepsilon)$, we choose $m = \mathrm{poly}(|D| \cdot n/\varepsilon)$ large enough such that $t \cdot \mathrm{e}^{-\frac{m \cdot \varepsilon^2}{20000}} \leq \mathrm{e}^{-n}$. In particular, we choose $m \geq (n + \ln(2t)) \cdot 20000/\varepsilon^2$. Therefore, by a union bound, with probability $1 - \mathrm{e}^{-n}$, all the queries to $\tilde{f}(\cdot)$ would be within $\pm\varepsilon/200$ of the answer that the ideal oracle $f(\cdot)$ would provide. This concludes the proof of the corollary. □

1.3.2 New p-Tampering and p-Resetting Biasing Attacks

In this subsection and Subsection 1.3.3 we prove Theorem 1.3.1. Our focus is on describing the relevant tampering algorithms Tam; the general attacks will be defined accordingly. (Recall Definition 1.2.5 and that the p-tampering attacker has an internal 'tampering' algorithm Tam that is executed with independent probability p.) We first describe our tampering algorithms in an ideal model where certain parameters of the function f are given for free by an oracle. In Section 1.3.3, we get rid of this assumption by approximating these parameters in polynomial time.

Definition 1.3.6 (Function $\hat{f}$)**.** Let D be a distribution, $f\colon \mathrm{Supp}(S) \mapsto \mathbb{R}$ be defined over D^n for some $n \in \mathbb{N}$, and $d_{\leq i} \in \mathrm{Supp}(D)^i$ for some $i \in [n]$. We define the following functions.

- $f_{d_{\leq i}}(\cdot)$ is a defined as $f_{d_{\leq i}}(d_{\geq i+1}) = f(z)$ for $z = (d_{\leq i}, d_{\geq i+1}) = (d_1, \ldots, d_n)$.
- $\hat{f}[d_{\leq i}] = \mathbb{E}_{d_{\geq i+1} \leftarrow D^{n-i}}[f_{d_{\leq i}}(d_{\geq i+1})]$. We let $\mu = \hat{f}[\varnothing]$ denote $\hat{f}[d_{\leq 0}] = \mathbb{E}[f(S)]$.

The key idea in both of our attacks is to design them (to run in polynomial time) based on oracle access to $\hat{f}$. The point is that $\hat{f}$ could later be approximated within arbitrarily small $1/\mathrm{poly}(n)$ factors, thus leading to sufficiently close approximations of our attacks. After describing the 'ideal' version of the attacks, we will describe how to make them efficient by approximating oracle calls to $\hat{f}$.

Changing the range of $f(\cdot)$. In both of our attacks, we describe our attacks using functions with range $[-1,+1]$. To get the results of Theorem 1.3.1 we simply need to scale the parameters back appropriately.

New p-Tampering Biasing Attack (Ideal Version)

Our Ideal p-Tam attack below, might repeat a loop indefinitely, but as we will see in Section 1.3.3, we can cut this rejection sampling procedure after a large enough polynomial number of rejection trials.

Construction 1.3.7 (Ideal p-Tam tampering)**.** Let D be an arbitrary distribution and $S \equiv D^n$ for some $n \in N$. Also let $f\colon \text{Supp}(D)^n \mapsto [-1,+1]$ be an arbitrary function.[12] For any $i \in [n]$, given a prefix $d_{\le i-1} \in \text{Supp}(D)^{i-1}$,[13] *ideal* p-Tam is a p-tampering attack defined as follows.

1. Let $r[d_{\le i}] = \frac{1-\hat{f}[d_{\le i}]}{3-p-(1-p)\cdot\hat{f}[d_{\le i-1}]}$.
2. With probability $1 - r[d_{\le i}]$ return d_i. Otherwise, sample a fresh $d_i \leftarrow D$ and go to step 1.

Proposition 1.3.8. *Ideal p-Tam attack is well defined. Namely, $r[d_{\le i}] \in [0,1]$ for all $d_{\le i} \in \text{Supp}(D)^i$.*

Proof. Both $\hat{f}[d_{\le i}], \hat{f}[d_{\le i-1}]$ are in $[-1,1]$. Therefore $0 \le 1 - \hat{f}[d_{\le i}] \le 2$ and $3 - p - (1-p)\cdot\hat{f}[d_{\le i-1}] \ge 2$ which implies $0 \le r[d_{\le i}] \le 1$. □

In the following, let A_{tam} be the p-tampering adversary using tampering algorithm Ideal p-Tam.[14]

Claim 1.3.9. *Let $\widetilde{S} = (\widetilde{D}_1, \ldots, \widetilde{D}_n)$ be the joint distribution after A_{tam} attack is performed on $S \equiv D^n$ using ideal p-Tam tampering algorithm. For every prefix $d_{\le i} \in \text{Supp}(D)^i$ we have,*

$$\frac{\Pr[\widetilde{D}_i = d_i \mid d_{\le i-1}]}{\Pr[D = d_i]} = \frac{2 - p\cdot(1-\hat{f}[d_{\le i}])}{2 - p\cdot(1-\hat{f}[d_{\le i-1}])}.$$

Proof. During its execution, ideal p-Tam keeps sampling examples and rejecting them until a sample is accepted. For $\ell \in \mathbb{N}$ we define R_ℓ to be the event that is true if the ℓ'th sample in the tampering algorithm is rejected, conditioned on reaching the ℓth sample. We have,

$$\begin{aligned}\Pr[\mathsf{R}_\ell] &= \sum_{d_i} \Pr[D = d_i] \cdot \left(\frac{1-\hat{f}[d_{\le i}]}{3-p-(1-p)\cdot\hat{f}[d_{\le i-1}]}\right) \\ &= \frac{\sum_{d_i}\Pr[D=d_i]\cdot(1-\hat{f}[d_{\le i}])}{3-p-(1-p)\cdot\hat{f}[d_{\le i-1}]} = \frac{1-\hat{f}[d_{\le i-1}]}{3-p-(1-p)\cdot\hat{f}[d_{\le i-1}]}.\end{aligned}$$

[12] As mentioned before, we describe our attacks using range $[-1,+1]$, and then we will do the conversion back to $[0,1]$.
[13] Note that here d_i is the 'original' untampered value for block i, while $d_1, \ldots, d_{i-1}$ might be the result of tampering.
[14] Therefore, A_D, inductively runs p-Tam over the current sequence with probability p. See Definition 1.2.5.

Let $c[d_{\le i-1}] = \frac{1-\hat{f}[d_{\le i-1}]}{3-p-(1-p)\cdot\hat{f}[d_{\le i-1}]}$. Then we have,

$$\begin{aligned}\frac{\Pr[\widetilde{D}_i = d_i \mid d_{\le i-1}]}{\Pr[D = d_i]} &= 1-p+p\cdot\left(\sum_{j=0}^{\infty}(1-r[d_{\le i}])\cdot\prod_{\ell=1}^{j}\Pr[\mathsf{R}_\ell]\right)\\ &= 1-p+p\cdot\left(\sum_{j=0}^{\infty}(1-r[d_{\le i}])\cdot c[d_{\le i-1}]^j\right)\\ &= 1-p+p\cdot\left(\frac{1-r[d_{\le i}]}{1-c[d_{\le i-1}]}\right) = \frac{2-p+p\cdot\hat{f}[d_{\le i}]}{2-p+p\cdot\hat{f}[d_{\le i-1}]}.\end{aligned}$$

□

The next corollary follows from Claim 1.3.9 and induction. (Recall that $\mu = \hat{f}[\varnothing] = \hat{f}[d_{\le 0}] = \mathbb{E}[f(S)]$.)

Corollary 1.3.10. *By applying the attack $\mathsf{A}_{\mathrm{tam}}$ based on the ideal p-Tam tampering algorithm, the distribution after the attack would be as follows,*

$$\Pr[\widetilde{S} = z] = \frac{2-p+p\cdot f(z)}{2-p+p\cdot\mu}\cdot\Pr[S = z].$$

Corollary 1.3.11. *The p-tampering attack $\mathsf{A}_{\mathrm{tam}}$ (based on the ideal p-Tam tampering algorithm) biases $f(\cdot)$ by $\frac{p\cdot\nu}{2-p+p\cdot\mu}$ where $\mu = \mathbb{E}[f(S)], \nu = \mathbb{V}[f(S)]$.*

Proof. It holds that $\mathbb{E}[f(\widetilde{S})]$ is equal to

$$\begin{aligned}&\sum_{z\in\mathrm{Supp}(D)^n}\Pr[\widetilde{S} = z]\cdot f(z) = \sum_{z\in\mathrm{Supp}(D)^n}\frac{2-p+p\cdot f(z)}{2-p+p\cdot\mu}\cdot\Pr[S = z]\cdot f(z)\\ &= \frac{2-p}{2-p+p\cdot\mu}\cdot\left(\sum_{z\in\mathrm{Supp}(D)^n}\Pr[S = z]\cdot f(z)\right)\\ &+ \frac{p}{2-p+p\cdot\mu}\cdot\left(\sum_{z\in\mathrm{Supp}(D)^n}\Pr[S = z]\cdot f(z)^2\right)\\ &= \frac{(2-p)\cdot\mu}{2-p+p\cdot\mu} + \frac{p\cdot(\nu+\mu^2)}{2-p+p\cdot\mu} = \mu + \frac{p\cdot\nu}{2-p+p\cdot\mu}.\end{aligned}$$

□

Corollary 1.3.12. *For any $S \equiv D^n$ and any function $f\colon \mathrm{Supp}(D^n) \to [0,1]$, there is a p-tampering attack that given oracle access to $\hat{f}(\cdot)$ and a sampling oracle for D, it biases the expected value of f by $\frac{p\cdot\nu}{1-p+p\cdot\mu}$ where $\mu = \mathbb{E}[f(S)], \nu = \mathbb{V}[f(S)]$.*

Proof. Consider another function $f' = 2 \cdot f - 1$. The range of f' is now $[-1, +1]$ and we have $\mu' = \mathbb{E}[f'(S)] = 2 \cdot \mu - 1$ and $\nu' = \mathbb{V}[f'(S)] = 4 \cdot \nu$. By Corollary 1.3.11, the p-tampering attack A_{tam} biases f' by $\frac{p \cdot \nu'}{2-p+p \cdot \mu'}$. Let $\widetilde{S}$ be the tampered distribution after performing A_{tam} on function f' and S. We have,

$$\mathbb{E}[f'(\widetilde{S})] \geq \mu' + \frac{p \cdot \nu'}{2 - p + p \cdot \mu'}.$$

Therefore we have,

$$\mathbb{E}[f(\widetilde{S})] = \frac{\mathbb{E}[f'(\widetilde{S})] + 1}{2} \geq \frac{\mu' + 1}{2} + \frac{p \cdot \nu'}{2 \cdot (2 - p + p \cdot \mu')} = \mu + \frac{p \cdot \nu}{1 - p + p \cdot \mu}.$$

□

New p-Resetting Biasing Attack (Ideal Version)

Construction 1.3.13 (Ideal p-Res)**.** Let D be an arbitrary distribution and $S \equiv D^n$ for some $n \in N$. Also let $f \colon \text{Supp}(D)^n \mapsto [-1, +1]$ be an arbitrary function.[15] For any $i \in [n]$, and given a prefix $d_{\leq i-1} \in \text{Supp}(D)^{i-1}$, the p-Res tampering algorithm works as follows.

1. Let $r[d_{\leq i}] = \frac{1-\hat{f}[d_{\leq i}]}{2+p \cdot (1+\hat{f}[d_{\leq i-1}])}$.
2. With probability $1 - r[d_{\leq i}]$ output the given d_i.
3. Otherwise sample $d_i' \leftarrow D$ (i.e., 'reset' d_i) and return d_i'.

Proposition 1.3.14. *Ideal p-Res algorithm is well defined. Namely, $r[d_{\leq i}] \in [0, 1]$ for all $d_{\leq i} \in \text{Supp}(D)^i$.*

Proof. We have $\hat{f}[d_{\leq i}] \in [-1, +1]$ and $\hat{f}[d_{\leq i-1}] \in [-1, +1]$. Therefore $0 \leq 1 - \hat{f}[d_{\leq i}] \leq 2$ and $2 + p \cdot (1 + \hat{f}[d_{\leq i-1}]) \geq 2$ which implies $0 \leq r[d_{\leq i}] \leq 1$. □

In the following let A_{res} be the p-tampering adversary using ideal p-Res. (See Definition 1.2.5.)

Claim 1.3.15. *Let $\widetilde{S} = (\widetilde{D}_1, \ldots, \widetilde{D}_n)$ be the distribution after the attack A_{res} (using ideal p-Res tampering algorithm) is performed on $S \equiv D^n$. For all $d_{\leq i} \in \text{Supp}(D)^i$ it holds that,*

$$\frac{\Pr[\widetilde{D}_i = d_i \mid d_{\leq i-1}]}{\Pr[D = d_i]} = \frac{2 + p \cdot (1 + \hat{f}[d_{\leq i}])}{2 + p \cdot (1 + \hat{f}[d_{\leq i-1}])}.$$

[15] As mentioned before, we describe our attacks using range $[-1, +1]$, and then we will do the conversion back to $[0, 1]$.

Proof. If R_1 defines the event that is true if the given sample is rejected, then,

$$\begin{aligned}\Pr[\mathsf{R}_1] &= \sum_{d_i} \Pr[D = d_i] \cdot \left(\frac{1 - \hat{f}[d_{\le i}]}{2 + p \cdot (1 + \hat{f}[d_{\le i-1}])} \right) \\ &= \frac{\sum_{d_i} \Pr[D = d_i] \cdot (1 - \hat{f}[d_{\le i}])}{2 + p \cdot (1 + \hat{f}[d_{\le i-1}])} = \frac{1 - \hat{f}[d_{\le i-1}]}{2 + p \cdot (1 + \hat{f}[d_{\le i-1}])}.\end{aligned}$$

Therefore, we conclude that,

$$\begin{aligned}\frac{\Pr[\widetilde{D}_i = d_i \mid d_{\le i-1}]}{\Pr[D = d_i]} &= 1 - p + p \cdot (1 - r[d_{\le i}] + \Pr[\mathsf{R}_1]) \\ &= 1 - p + p \cdot \left(1 + \frac{\hat{f}[d_{\le i}] - \hat{f}[d_{\le i-1}]}{2 + p \cdot (1 + \hat{f}[d_{\le i-1}])} \right) \\ &= 1 + p \cdot \left(\frac{\hat{f}[d_{\le i}] - \hat{f}[d_{\le i-1}]}{2 + p \cdot (1 + \hat{f}[d_{\le i-1}])} \right) = \frac{2 + p \cdot (1 + \hat{f}[d_{\le i}])}{2 + p \cdot (1 + \hat{f}[d_{\le i-1}])}.\end{aligned}$$

□

The next corollary follows from Claim 1.3.15 and induction. (Recall that $\mu = \hat{f}[\varnothing] = \hat{f}[d_{\le 0}] = \mathbb{E}[f(S)]$.)

Corollary 1.3.16. *By applying attack $\mathsf{A}_{\mathrm{res}}$ (using ideal p-Res), the distribution after the attack is,*

$$\Pr[\widetilde{S} = z] = \frac{2 + p + p \cdot f(z)}{2 + p + p \cdot \mu} \cdot \Pr[S = z].$$

Corollary 1.3.17. *The p-resetting attack $\mathsf{A}_{\mathrm{res}}$ (using ideal p-Res) biases the function by $\frac{p \cdot \nu}{2 + p + p \cdot \mu}$ where $\mu = \mathbb{E}[f(S)], \nu = \mathbb{V}[f(S)]$.*

Proof. It holds that $\widetilde{\mu} = \mathbb{E}[f(\widetilde{S})]$ is equal to

$$\begin{aligned}&\sum_{z \in \mathrm{Supp}(D)^n} \Pr[\widetilde{S} = z] \cdot f(z) = \sum_{z \in \mathrm{Supp}(D)^n} \frac{2 + p + p \cdot f(z)}{2 + p + p \cdot \mu} \cdot \Pr[S = z] \cdot f(z) \\ &= \frac{2 + p}{2 + p + p \cdot \mu} \cdot \left(\sum_{z \in \mathrm{Supp}(D)^n} \Pr[S = z] \cdot f(z) \right) \\ &+ \frac{p}{2 + p + p \cdot \mu} \cdot \left(\sum_{z \in \mathrm{Supp}(D)^n} \Pr[S = z] \cdot f(z)^2 \right) \\ &= \frac{(2 + p) \cdot \mu}{2 + p + p \cdot \mu} + \frac{p \cdot (\nu + \mu^2)}{2 + p + p \cdot \mu} = \mu + \frac{p \cdot \nu}{2 + p + p \cdot \mu}.\end{aligned}$$

□

Corollary 1.3.18. *For $S \equiv D^n$ and any $f\colon \mathrm{Supp}(S) \to [0,1]$ there exist a p-resetting attack that, given oracle access to $\hat{f}$ and a sampling oracle for D, it biases f by $\frac{p\cdot\nu}{1+p\cdot\mu}$ where $\mu = \mathbb{E}[f(S)], \nu = \mathbb{V}[f(S)]$.*

Proof. Consider another function $f' = 2\cdot f - 1$. Now, the range of f' is $[-1,+1]$, and we have $\mu' = \mathbb{E}[f'(S)] = 2\cdot\mu - 1$ and $\nu' = \mathbb{V}[f'(S)] = 4\cdot\nu$. By Corollary 1.3.17, the p-resetting attack A_{res} biases f' by $\frac{p\cdot\nu'}{2-p+p\cdot\mu'}$. Let $\widetilde{S}$ be the tampered distribution after performing A_{tam} on function f' and S. We have,

$$\mathbb{E}[f'(\widetilde{S})] \geq \mu' + \frac{p\cdot\nu'}{2+p+p\cdot\mu'}.$$

Therefore we have,

$$\mathbb{E}[f(\widetilde{S})] = \frac{\mathbb{E}[f'(\widetilde{S})]+1}{2} \geq \frac{\mu'+1}{2} + \frac{p\cdot\nu'}{2\cdot(2+p+p\cdot\mu')} = \mu + \frac{p\cdot\nu}{1+p\cdot\mu}.$$

□

1.3.3 Approximating the Ideal Attacks in Polynomial Time

In this subsection, we describe the efficient version of the attacks of Theorem 1.3.1 and prove their properties. We first describe the efficient version of our p-resetting attack, where achieving efficiency is indeed simpler. We then go over the efficient variant of our p-tampering attack. In both cases, we describe the modifications needed for the *tampering algorithms* and it is assumed that such tampering algorithms are used by the main efficient attackers (see Definition 1.2.5).

We start by approximating in polynomial time our Ideal p-resetting attack, as it is simpler to argue about the polynomial-time version of this attack. We will then use lemmas and ideas that we develop along the way to also make our 1st Ideal p-tampering attacker also polynomial time.

Polynomial-time Variant of the Ideal p-Resetting Biasing Attack

The p-resetting attack of Construction 1.3.13 is not polynomial-time since it needs access to the idealized oracle providing partial averages. In general, we can not compute such averages exactly in polynomial time, however in order to make those attacks polynomial-time, we can rely on *approximating* the partial averages and consequently the corresponding rejection probabilities. To get the polynomial-time version of the attack of Construction 1.3.13 we can pursue the following idea. For every prefix $d_{\leq i}$, the polynomial-time attacker first approximates the partial average $\hat{f}[d_{\leq i}]$ by sampling a sufficiently large polynomial number of random continuations ${d^{(1)}}_{\leq n-i}, \ldots {d^{(\ell)}}_{\leq n-i}$ and getting the average $\mathbb{E}_{j\in[\ell]}[f(d_{\leq i}, {d^{(j)}}_{\leq n-i}]$ as an approximation for the partial average. By the Hoeffding inequality, this average is a good approximation of $\hat{f}[d_{\leq i}]$ with

exponentially high probability. Consequently, the rejection probabilities can be approximated well making the final distributions statistically close to the distribution of the ideal attack, meaning that the amount of bias is close to the ideal bias as well.

We now formalize the ideas above.

Definition 1.3.19 (Semi-ideal oracle $\tilde{f}[\cdot]$)**.** Let D be a distribution. If for all $d_{\leq i} \in \mathrm{Supp}(D)^i$ we have $\tilde{f}_\xi[d_{\leq i}] \in \hat{f}[d_{\leq i}] \pm \xi$, then, we call $\tilde{f}_\xi[\cdot]$ an ξ-approximation of $\hat{f}[\cdot]$. For simplicity, and when it is clear from the context, we simply write $\tilde{f}[\cdot]$ and call it a *semi-ideal* oracle.

The following lemma immediately follows from the Hoeffding inequality.

Lemma 1.3.20 (Approximating $\hat{f}[\cdot]$ in polynomial-time)**.** *Consider an algorithm that on inputs $d_{\leq i}$ and ξ performs as follows where $\ell = -10\ln(\xi/2)/\xi^2$.*

1. *Sample $(d^1_{\leq n-i}, \ldots, d^\ell_{\leq n-i}) \leftarrow (D^{n-i+1})^\ell$.*
2. *Output $\tilde{f}_\xi[d_{\leq i}] = \mathbb{E}_{j\in[\ell]}\, f(d_{\leq i}, d^j_{\leq n-i})$.*

Then it holds that $\Pr[|\tilde{f}_\xi[d_{\leq i}] - \hat{f}[d_{\leq i}]| \geq \xi] \leq \xi$.

The above lemma implies that if f is polynomial-time computable and D is polynomial-time samplable, any q-query algorithm can approximate the semi-ideal oracle $\tilde{f}[\cdot]$ in time $\mathrm{poly}(q\cdot n/\xi)$ and total error (of failing in one of the queries) by at most ξ. Based on this approximation of $\tilde{f}[\cdot]$, we now describe our polynomial-time version of the Ideal p-Res attack in the semi-ideal oracle model of $\tilde{f}[\cdot]$, by essentially using the semi-ideal oracle $\tilde{f}[\cdot]$ instead of the ideal oracle $\hat{f}[\cdot]$.

Construction 1.3.21 (Polynomial-time p-Res)**.** Polynomial-time p-Res is the same as ideal p-Res of Construction 1.3.13 but it calls the semi-ideal oracle $\tilde{f}_\xi[\cdot]$ instead of the ideal oracle $\hat{f}[\cdot]$.

In the following we analyze the bias achieved by the polynomial-time variant of the p-Res algorithm. We simply pretend that all the queries to the semi-ideal oracle are within $\pm\xi$ approximation of the ideal oracle, knowing that the error of ξ-approximating all of the queries is itself at most ξ and can affect the average also by at most $O(\xi)$. First we show that the rejection probabilities are approximated well.

Lemma 1.3.22. *Let $0 < p < 1$, $0 < \xi < 1$, $\alpha, \beta \in [-\xi, \xi]$, and $\hat{f}[d_{\leq i-1}]$, $\hat{f}[d_{\leq i}]$, $\tilde{f}_\xi[d_{\leq i-1}]$, $\tilde{f}_\xi[d_{\leq i}] \in [0,1]$ such that $\tilde{f}_\xi[d_{\leq i-1}] = \hat{f}[d_{\leq i-1}] + \alpha$ and $\tilde{f}_\xi[d_{\leq i}] = \hat{f}[d_{\leq i}] + \beta$. Let $r[.]$ and $\tilde{r}[.]$ respectively be the rejection probabilities of the Ideal and Polynomial-time p-Res. Then, for every $d_{\leq i} \in \mathrm{Supp}(D)^i$, $|r[d_{\leq i}] - \tilde{r}[d_{\leq i}]| \leq O(\xi)$.*

Proof. We have,

$$|r[d_{\leq i}] - \tilde{r}[d_{\leq i}]| = \left| \frac{1 - \hat{f}[d_{\leq i}])}{2 + p\cdot(1 + \hat{f}[d_{\leq i-1}])} - \frac{1 - \tilde{f}_\xi[d_{\leq i}]}{2 + p\cdot(1 + \tilde{f}_\xi[d_{\leq i-1}])} \right|,$$

where we can compute the following for the right hand side,

$$
\begin{aligned}
&= \left| \frac{(2+p)(\tilde{f}_\xi[d_{\le i}] - \hat{f}[d_{\le i}]) + p \cdot (\tilde{f}_\xi[d_{\le i-1}] - \hat{f}[d_{\le i-1}])}{(2+p\cdot(1+\hat{f}[d_{\le i-1}]))\cdot(2+p\cdot(1+\tilde{f}_\xi[d_{\le i-1}]))} \right. \\
&\quad \left. + \frac{p\cdot(\hat{f}[d_{\le i-1}]\tilde{f}_\xi[d_{\le i}] - \tilde{f}_\xi[d_{\le i-1}]\hat{f}[d_{\le i}])}{(2+p\cdot(1+\hat{f}[d_{\le i-1}]))\cdot(2+p\cdot(1+\tilde{f}_\xi[d_{\le i-1}]))} \right| \\
&\le \frac{\left|(2+p)(\tilde{f}_\xi[d_{\le i}] - \hat{f}[d_{\le i}])\right| + \left|p\cdot(\tilde{f}_\xi[d_{\le i-1}] - \hat{f}[d_{\le i-1}])\right|}{4} \\
&\quad + \frac{\left|p\cdot(\hat{f}[d_{\le i-1}]\tilde{f}_\xi[d_{\le i}] - \tilde{f}_\xi[d_{\le i-1}]\hat{f}[d_{\le i}])\right|}{4} \\
&\le \frac{(2+p)\xi + p\xi + p\left|\hat{f}[d_{\le i-1}](\hat{f}[d_{\le i}]+\beta) - (\hat{f}[d_{\le i-1}]+\alpha)\hat{f}[d_{\le i}]\right|}{4} \\
&= \frac{2\xi + 2p\xi + p\left|\beta\hat{f}[d_{\le i-1}] - \alpha\hat{f}[d_{\le i}]\right|}{4} \le \frac{2\xi + 2p\xi + p\cdot(|\beta| + |-\alpha|)}{4} \le 3\xi/2\,.
\end{aligned}
$$

□

Now we want to argue that when we approximate the p-resetting tampering algorithm's rejection probabilities as proved in Lemma 1.3.22, it leads to 'close probabilities' of sampling final outputs. We prove the following general lemma that will be also useful for the case of Polynomial-time p-Tam attack. For the case of p-resetting, we only need the special case of $k = 1$.

Notation. For $p \in [0,1]$ and distributions X, Y, by $Z \equiv (1-p)X + pY$ we denote the distribution Z in which we sample from X with probability $1-p$, and otherwise (i.e., with probability p) we sample from Y.

Definition 1.3.23 ((p,k,ρ)-variations)**.** For any distribution D, function $\rho\colon \mathrm{Supp}(D) \to [0,1]$, and $k \in \mathbb{N}$, the (p,k,ρ)-variation of D is $D_{p,k,\rho} \equiv (1-p)D + pZ$, where Z is defined as follows.

1. Sample $(d_1, \ldots, d_k) \leftarrow D^k$.
2. For $i \in \{1, \ldots, k\}$, go sequentially over $d_1, \ldots, d_k$, and with probability $\rho[d_i]$ return d_i and exit.
3. If nothing was returned after reading all the k samples, return a fresh sample $d_{k+1} \leftarrow D$.

Lemma 1.3.24 (Implication of approximating rejection probabilities)**.** *Let D be a distribution and $\rho : \mathrm{Supp}(D) \to [0,1]$ and $\rho' : \mathrm{Supp}(D) \to [0,1]$ be two functions such that $\forall d \in \mathrm{Supp}(D), |\rho(d) - \rho'(d)| \le \xi$. Then, for every $k \in \mathbb{N}$ and every $d \in \mathrm{Supp}(D)$, it holds that,*

$$
\left|\ln\left(\frac{\Pr[D_{p,k,\rho} = d]}{\Pr[D_{p,k,\rho'} = d]}\right)\right| \le \frac{p}{1-p}\cdot(k^2+k)\cdot\xi.
$$

Before proving the lemma above, we note that it indeed implies that the *max divergence* [Dwork et al., 2010] of $D_{p,k,\rho}$ and $D_{p,k,\rho'}$ is at most $O(k^2 \cdot \xi)$.

Proof. Let $a = \mathbb{E}_{d \leftarrow D}[\rho(d)]$ and $a' = \mathbb{E}_{d \leftarrow D}[\rho'(d)]$. We have,

$$\frac{\Pr[D_{p,k,\rho} = d]}{\Pr[D = d]} = (1-p) + p \cdot ((1-a)^k + \sum_{i \in [k-1]} \rho(d) \cdot (1-a)^i).$$

With a similar calculation for $\Pr[D_{p,k,\rho'} = d]$ we get,

$$\begin{aligned}
&\frac{\Pr[D_{p,k,\rho} = d]}{\Pr[D_{p,k,\rho'} = d]} \\
&= \frac{(1-p) + p \cdot ((1-a)^k + \sum_{i \in [k-1]} \rho(d) \cdot (1-a)^i)}{(1-p) + p \cdot ((1-a')^k + \sum_{i \in [k-1]} \rho(d) \cdot (1-a')^i)} \\
&= 1 + \frac{p \cdot ((1-a)^k - (1-a')^k + \sum_{i \in [k-1]} \rho(d) \cdot (1-a)^i - \rho'(d) \cdot (1-a')^i)}{(1-p) + p \cdot ((1-a')^k + \sum_{i \in [k-1]} \rho(d) \cdot (1-a')^i)} \\
&\leq 1 + \frac{p \cdot (k \cdot \xi + \sum_{i \in [k-1]} (2i+1) \cdot \xi)}{1-p} \\
&= 1 + \frac{p}{1-p}(k^2 + k) \cdot \xi \\
&\leq \mathrm{e}^{\frac{p}{1-p}(k^2+k) \cdot \xi}.
\end{aligned}$$

Similarly, we have $\frac{\Pr[D_{p,k,\rho'}=d]}{\Pr[D_{p,k,\rho}=d]} \leq \mathrm{e}^{\frac{p}{1-p}(k^2+k)\xi}$ which implies that,

$$\left| \ln \left(\frac{\Pr[D_{p,k,\rho} = d]}{\Pr[D_{p,k,\rho'} = d]} \right) \right| \leq \frac{p}{1-p} \cdot (k^2 + k) \cdot \xi.$$

□

The following lemma states that the expected values of a function over two distributions that are 'close' (under max divergence) are indeed close real numbers.

Lemma 1.3.25. *Let $X = (X_1, \ldots, X_n)$ and $Y = (Y_1, \ldots, Y_n)$ be two joint distributions such that $\mathrm{Supp}(X) = \mathrm{Supp}(Y)$ and for every prefix $x_{\leq i}$ such that $\Pr[X_i = x_i \mid x_{\leq i-1}] > 0$, we have,*

$$\left| \ln \left(\frac{\Pr[X_i = x_i \mid x_{\leq i-1}]}{\Pr[Y_i = x_i \mid x_{\leq i-1}]} \right) \right| \leq \xi.$$

Then, for any function $f \colon \mathrm{Supp}(X) \to [-1, +1]$ we have,

$$\mathbb{E}[f(X)] \geq \mathbb{E}[f(Y)] - \mathrm{e}^{\xi \cdot n} + 1.$$

Proof. First, we note that for every $x \in \text{Supp}(X)$ it holds that,

$$\left|\ln\left(\frac{\Pr[X=x]}{\Pr[Y=x]}\right)\right| = \left|\sum_{i\in[n]} \ln\left(\frac{\Pr[X_i = x_i \mid x_{\le i-1}]}{\Pr[Y_i = x_i \mid x_{\le i-1}]}\right)\right| \le n\cdot\xi.$$

Now for the difference $\mathbb{E}[f(Y)] - \mathbb{E}[f(X)]$ we have,

$$\begin{aligned}
&\sum_{x\in\text{Supp}(X)} (\Pr[Y=x] - \Pr[X=x])\cdot f(x)\\
&\le \sum_{x\in\text{Supp}(X)} |(\Pr[Y=x] - \Pr[X=x])\cdot f(x)|\\
&\le \sum_{x\in\text{Supp}(X)} \left|\min(\Pr[X=x],\Pr[Y=x])\cdot\left(\mathrm{e}^{\left|\ln(\frac{\Pr[Y=x]}{\Pr[X=x]})\right|}-1\right)\cdot f(x)\right|\\
&\le (\mathrm{e}^{n\cdot\xi}-1)\cdot \sum_{x\in\text{Supp}(X)} |\min(\Pr[X=x],\Pr[Y=x])\cdot f(x)|\\
&\le \mathrm{e}^{n\cdot\xi}-1.
\end{aligned}$$

□

Putting things together. Now we show how to choose the parameters of the Polynomial-time p-Res. Suppose ξ' is the parameter of Theorem 1.3.1. If we choose ξ as the parameter of our attack we can bound the final bias as follows. Firstly, if the approximation algorithm of Lemma 1.3.20 gives us a semi-ideal oracle $\tilde{f}_\xi[.]$, then based on Lemma 1.3.22 we can approximate the rejection probabilities with error at most $O(\xi)$. Then based on Lemma 1.3.24 the attack A_{res} that uses efficient p-Res generates a distribution that is $O(\frac{p}{1-p}\cdot\xi)$-close[16] to the distribution of the attack A_{res} that uses ideal p-Res.

Now we can use Lemma 1.3.25 (for $k=1$) to argue that the bias achieved by the efficient adversary is $(\mathrm{e}^{O(n\cdot\xi\cdot\frac{p}{1-p})}-1)$-close to the bias achieved by the ideal adversary. Also note that, if the approximation algorithm fails to provide a semi-ideal oracle for all queries, then the bias of the efficient attack is at least -2 because the function range is $[-1,+1]$. However, the probability of this event is bounded by $O(n\cdot\xi)$ because adversary needs at most $2n$ number of queries to $\tilde{f}$. Therefore, the difference of bias of the efficient and the ideal adversary is at most $O(n\cdot\xi) + \mathrm{e}^{O(n\cdot\xi\cdot\frac{p}{1-p})} - 1$ which is at most $O(n\cdot\xi + n\cdot\xi\cdot\frac{p}{1-p})$ if the exponent in $\mathrm{e}^{O(n\cdot\xi\cdot\frac{p}{1-p})}$ is at most 1. As a result, if we choose $\xi = o\left(\xi'/(n\cdot\frac{p}{1-p})\right) = o\left(\xi'\cdot(1-p)/(n\cdot p)\right)$, we can indeed guarantee that the bias of the efficient adversary is ξ'-close to bias of ideal adversary.

[16] Since we are assuming $p<1$ is constant $O(\frac{p}{1-p}\cdot\xi)$ simply means $O(\xi)$.

Polynomial-time Variant of our p-Tampering Biasing Attack

Building upon the ideas developed above to make our Ideal p-Res tampering algorithm polynomial time, here we focus on our Ideal p-Tam attack. We start by describing a variant of the original attack of Construction 1.3.7 where we cut the rejection sampling procedure after k iterations.

Construction 1.3.26 (Ideal k-cut p-Tam)**.** Ideal k-cut p-Tam is the same as ideal p-Tam of Construction 1.3.7 but it is forced to stop and return a fresh sample if the first k samples were rejected.

Now we show that the new modified attack of Construction 1.3.26 will lead to a close distribution compared to the original attack of Construction 1.3.7.

Lemma 1.3.27. *Let $\widetilde{S} = (\widetilde{D}_1, \ldots, \widetilde{D}_n)$ be the joint distribution after A_{tam} attack is performed on $S \equiv D^n$ using ideal p-Tam tampering algorithm. Also, let $\widetilde{S}' = (\widetilde{D}'_1, \ldots, \widetilde{D}'_n)$ be the joint distribution after A_{tam} attack is performed on S using Ideal k-cut p-Tam tampering algorithm. For every prefix $d_{\leq i} \in \text{Supp}(D)^i$,*

$$\left|\ln\left(\frac{\Pr[\widetilde{D}_i = d_i \mid d_{\leq i-1}]}{\Pr[\widetilde{D}'_i = d_i | d_{\leq i-1}]}\right)\right| \leq \frac{p}{(1-p)^2 \cdot (2-p)^{k-1}}.$$

Proof. Let $r[d_{\leq i}] = \frac{1-\hat{f}[d_{\leq i}]}{3-p-(1-p)\cdot\hat{f}[d_{\leq i-1}]}$ and $c[d_{\leq i-1}] = \frac{1-\hat{f}[d_{\leq i-1}]}{3-p-(1-p)\cdot\hat{f}[d_{\leq i-1}]}$ as it was defined in proof of Claim 1.3.9. We have,

$$\begin{aligned}\frac{\Pr[\widetilde{D}'_i = d_i \mid d_{\leq i-1}]}{\Pr[D = d_i]} &= (1-p)+ \\ &\quad p \cdot \left((c[d_{\leq i-1}])^k + \sum_{j\in[k-1]} (1 - r[d_{\leq i}]) \cdot (1 - c[d_{\leq i}])^j)\right) \\ &= (1-p) + p \cdot \left((c[d_{\leq i-1}])^k + \frac{(1 - r[d_{\leq i}]) \cdot (1 - c[d_{\leq i-1}]^k)}{1 - c[d_{\leq i-1}]}\right).\end{aligned}$$

Also, in the proof of Claim 1.3.9 we showed that,

$$\frac{\Pr[\widetilde{D}_i = d_i \mid d_{\leq i-1}]}{\Pr[D = d_i]} = 1 - p + p \cdot \left(\frac{1 - r[d_{\leq i}]}{1 - c[d_{\leq i-1}]}\right).$$

Therefore, we conclude that,

$$\begin{aligned}\frac{\Pr[\widetilde{D}'_i = d_i \mid d_{\leq i-1}]}{\Pr[\widetilde{D}_i = d_i \mid d_{\leq i-1}]} &= \frac{(1-p) + p \cdot \left((c[d_{\leq i-1}])^k + \frac{(1-r[d_{\leq i}])\cdot(1-c[d_{\leq i-1}]^k)}{1-c[d_{\leq i-1}]}\right)}{1 - p + p \cdot \left(\frac{1-r[d_{\leq i}]}{1-c[d_{\leq i-1}]}\right)} \\ &= 1 + \frac{p \cdot \left(\frac{(r[d_{\leq i}]-c[d_{\leq i-1}])\cdot c[d_{\leq i-1}]^k}{1-c[d_{\leq i-1}]}\right)}{1 - p + p \cdot \left(\frac{1-r[d_{\leq i}]}{1-c[d_{\leq i-1}]}\right)}.\end{aligned}$$

We also know that $c[d_{\leq i-1}] \leq \frac{1}{2-p}$ because $\hat{f}[d_{\leq i-1}] \in [-1,+1]$. So we have,

$$\begin{aligned}\frac{\Pr[\widetilde{D}'_i = d_i \mid d_{\leq i-1}]}{\Pr[\widetilde{D}_i = d_i \mid d_{\leq i-1}]} &= 1 + \frac{p \cdot \left(\frac{(r[d_{\leq i}] - c[d_{\leq i-1}]) \cdot c[d_{\leq i-1}]^k}{1-c[d_{\leq i-1}]}\right)}{1-p+p\cdot\left(\frac{1-r[d_{\leq i}]}{1-c[d_{\leq i-1}]}\right)} \\ &\leq 1 + \frac{p \cdot c[d_{\leq i-1}]^k}{(1-p)\cdot(1-c[d_{\leq i-1}])} \\ &\leq 1 + \frac{p}{(1-p)^2\cdot(2-p)^{k-1}} \leq \mathrm{e}^{\frac{p}{(1-p)^2(2-p)^{k-1}}}.\end{aligned}$$

Also for the inverse ratio, we have,

$$\begin{aligned}\frac{\Pr[\widetilde{D}_i = d_i \mid d_{\leq i-1}]}{\Pr[\widetilde{D}'_i = d_i \mid d_{\leq i-1}]} &= 1 + \frac{p \cdot \left(\frac{(c[d_{\leq i-1}] - r[d_{\leq i}]) \cdot c[d_{\leq i-1}]^k}{1-c[d_{\leq i-1}]}\right)}{(1-p)+p\cdot\left((c[d_{\leq i-1}])^k + \frac{(1-r[d_{\leq i}])\cdot(1-c[d_{\leq i-1}]^k)}{1-c[d_{\leq i-1}]}\right)} \\ &\leq 1 + \frac{p \cdot c[d_{\leq i-1}]^k}{(1-p)\cdot(1-c[d_{\leq i-1}])} \\ &\leq 1 + \frac{p}{(1-p)^2\cdot(2-p)^{k-1}} \leq \mathrm{e}^{\frac{p}{(1-p)^2\cdot(2-p)^{k-1}}}.\end{aligned}$$

Therefore, we can finally conclude that,

$$\left|\ln\left(\frac{\Pr[\widetilde{D}_i = d_i \mid d_{\leq i-1}]}{\Pr[\widetilde{D}'_i = d_i | d_{\leq i-1}]}\right)\right| \leq \frac{p}{(1-p)^2\cdot(2-p)^{k-1}}.$$

□

Lemma 1.3.28. *Let $\widetilde{S} = (\widetilde{D}_1, \ldots, \widetilde{D}_n)$ be the joint distribution after $\mathsf{A}_{\mathrm{tam}}$ attack is performed on $S \equiv D^n$ using ideal p-Tam tampering algorithm. Also, let $\widetilde{S}' = (\widetilde{D}'_1, \ldots, \widetilde{D}'_n)$ be the joint distribution after $\mathsf{A}_{\mathrm{tam}}$ attack is performed on S using Ideal k-cut p-Tam tampering algorithm where $k = \frac{\ln(2-p)-2\ln((1-p)\cdot\xi)}{\ln(2-p)}$. Then, it holds that,*

$$\mathbb{E}[f(\widetilde{S}')] \geq \mathbb{E}[f(\widetilde{S})] - \mathrm{e}^{n\cdot\xi} + 1.$$

Proof. Using Lemma 1.3.27, for every prefix $d_{\leq i} \in \mathrm{Supp}(D)^i$ we have,

$$\left|\ln\left(\frac{\Pr[\widetilde{D}_i = d_i \mid d_{\leq i-1}]}{\Pr[\widetilde{D}'_i = d_i | d_{\leq i-1}]}\right)\right| \leq \frac{p}{(1-p)^2\cdot(2-p)^{k-1}} \leq \xi.$$

Now, using Lemma 1.3.25 we get $\mathbb{E}[f(\widetilde{S}')] \geq \mathbb{E}[f(\widetilde{S})] - \mathrm{e}^{n\cdot\xi} + 1$. □

We can now describe the actual efficient variant of our Ideal p-Tam attack.

Construction 1.3.29 (Polynomial-time k-cut p-Tam)**.** Efficient k-cut p-Tam is the same as Ideal k-cut p-Tam of Construction 1.3.26 but it it calls the semi-ideal oracle $\tilde{f}_\xi[\cdot]$ instead of the ideal oracle $\hat{f}[\cdot]$.

Lemma 1.3.30. *Let $0 < p < 1$. Let $0 < \xi < 1$. Let $\alpha, \beta \in [-\xi, \xi]$. Let $\hat{f}[d_{\le i-1}]$, $\hat{f}[d_{\le i}]$, $\tilde{f}_\xi[d_{\le i-1}]$, $\tilde{f}_\xi[d_{\le i}] \in [0,1]$ such that $\tilde{f}_\xi[d_{\le i-1}] = \hat{f}[d_{\le i-1}] + \alpha$ and $\tilde{f}_\xi[d_{\le i}] = \hat{f}[d_{\le i}] + \beta$. Let $r[.]$ and $\tilde{r}[.]$ respectively be the rejection probabilities of the Ideal and Efficient k-cut p-Tam. Then, for every $d_{\le i} \in \mathrm{Supp}(D)^i$ we have $|r[d_{\le i}] - \tilde{r}[d_{\le i}]| \le O(\xi)$.*

Proof. The proof is similar to the proof of Lemma 1.3.22. We have,

$$|r[d_{\le i}] - \tilde{r}[d_{\le i}]| = \left| \frac{1 - \hat{f}[d_{\le i}])}{3 - p - (1-p)\hat{f}[d_{\le i-1}]} - \frac{1 - \tilde{f}_\xi[d_{\le i}]}{3 - p - (1-p)\tilde{f}_\xi[d_{\le i-1}]} \right| ,$$

where we can compute the following for the right hand side,

$$\begin{aligned}
&= \left| \frac{(1 - \hat{f}[d_{\le i}])(3 - p - (1-p)\tilde{f}_\xi[d_{\le i-1}])}{(3 - p - (1-p)\hat{f}[d_{\le i-1}])(3 - p - (1-p)\tilde{f}_\xi[d_{\le i-1}])} \right. \\
&\quad \left. - \frac{(1 - \tilde{f}_\xi[d_{\le i}])(3 - p - (1-p)\hat{f}[d_{\le i-1}])}{(3 - p - (1-p)\hat{f}[d_{\le i-1}])(3 - p - (1-p)\tilde{f}_\xi[d_{\le i-1}])} \right| \\
&\le \left| \frac{(1-p)(\hat{f}[d_{\le i-1}] - \tilde{f}_\xi[d_{\le i-1}]) + (3-p)(\tilde{f}_\xi[d_{\le i}] - \hat{f}[d_{\le i}])}{(3 - p - (1-p))(3 - p - (1-p))} \right. \\
&\quad \left. + \frac{(1-p)(\tilde{f}_\xi[d_{\le i-1}]\hat{f}[d_{\le i}] - \hat{f}[d_{\le i-1}]\tilde{f}_\xi[d_{\le i}])}{(3 - p - (1-p))(3 - p - (1-p))} \right| \\
&\le \frac{(1-p)\xi + (3-p)\xi + (1-p)\left| \left(\hat{f}[d_{\le i-1}] + \alpha\right)\hat{f}[d_{\le i}] - \hat{f}[d_{\le i-1}]\left(\hat{f}[d_{\le i}] + \beta\right) \right|}{4} \\
&\le \frac{4\xi + |\alpha| + |\beta|}{4} \le 3\xi/2 .
\end{aligned}$$

□

Putting things together. Now we show how to choose the parameters of the Efficient k-cut p-Tam. Suppose ξ' is the parameter of Theorem 1.3.1. If we choose ξ as the parameter of our attack we can bound the final bias as follows. Firstly, if the approximation algorithm of Lemma 1.3.20 gives us a semi-ideal oracle $\tilde{f}_\xi[.]$, then based on Lemma 1.3.30 we can approximate the rejection probabilities with error at most $O(\xi)$. Then based on Lemma 1.3.24 the attack $\mathsf{A}_{\mathrm{tam}}$ that uses the efficient k-cut p-Tam generates a distribution that is $O(\frac{p}{1-p} \cdot k^2 \cdot \xi)$-close to the distribution of the attack $\mathsf{A}_{\mathrm{tam}}$ that uses ideal k-cut p-Tam.

By Lemma 1.3.25, the bias of an efficient adversary is $\left(\mathrm{e}^{O(n \cdot \xi \cdot k^2 \cdot \frac{p}{1-p})} - 1\right)$-close to the bias of the ideal adversary. Also note that, if the approximation algorithm fails to provide a semi-ideal oracle for all queries, then the bias of efficient attack is at least -2 because the function range is $[-1, +1]$. However, the probability

of this event is bounded by $O(k \cdot n \cdot \xi)$ because the adversary needs at most $(k+1) \cdot n$ number of queries to $\tilde{f}$. Therefore, the difference of bias of the efficient and the ideal adversary is at most $O(k \cdot n \cdot \xi) + \mathrm{e}^{O(k^2 \cdot n \cdot \xi \cdot \frac{p}{1-p})} - 1$ which is at most $O(n \cdot \xi + k^2 \cdot n \cdot \xi \cdot \frac{p}{1-p})$ if the exponent in $\mathrm{e}^{O(k^2 \cdot n \cdot \xi \cdot \frac{p}{1-p})}$ is at most 1. As a result, if we choose $\xi = o(\xi'/(k^2 \cdot n \cdot \frac{p}{1-p})) = o(\xi' \cdot (1-p)/(k^2 \cdot n \cdot p))$, we can indeed guarantee that the bias of the efficient adversary (that uses efficient k-cut p-Tam tampering algorithm) is ξ'-close to the bias of the ideal adversary (that uses ideal k-cut p-Tam).

Now we want to select our other parameter k. Based on Lemma 1.3.28, if we choose $k = \omega\left(\frac{\ln((1-p)\xi')}{\ln(2-p)}\right)$ the bias of the attack $\mathsf{A}_{\mathrm{tam}}$ that uses the ideal k-cut p-Tam would be ξ'-close to the bias of the attack $\mathsf{A}_{\mathrm{tam}}$ that uses the ideal p-Tam. Therefore, the bias of the $\mathsf{A}_{\mathrm{tam}}$ that uses efficient k-cut attack is $2 \cdot \xi'$-close to the bias of $\mathsf{A}_{\mathrm{tam}}$ that uses ideal p-Tam.

1.4 PAC Learning under p-Tampering and p-Budget Attacks

In this section, we study the non-targeted case where PAC learning could be defined. We show that realizable problems that are PAC learnable (without attacks), are usually PAC learnable under p-tampering attacks as well. Essentially we bound the probability of some bad event happening (see Definition 1.4.2) in a manner similar to Occam algorithms [Blumer et al., 1987] by relying on the realizability assumption and relying on the specific property of the p-tampering attacks. In particular, we crucially rely on the fact that any p-tampering distribution $\widetilde{D}$ of a distribution D contains a $(1-p) \cdot D$ measure in itself. In fact, we show (see Theorem 1.4.7) that in a close scenario to p-tampering in which the adversary can choose the ($\leq p$ fraction of the) tampering locations, PAC learning might suddenly become impossible. This shows that the 'mistake-free' nature of p-tampering is indeed *not* enough for PAC learnability.[17]

1.4.1 Definitions

Definition 1.4.1. For a learning problem $\mathsf{P} = (\mathcal{X}, \mathcal{Y}, \mathcal{D}, \mathcal{H}, \ell oss)$, distribution $D \in \mathcal{D}$, and training sequence $\mathcal{S} = ((x_1, y_1), \ldots, (x_n, y_n)) \leftarrow D^n$, we say that the event $\mathrm{Bad}_\varepsilon(D, \mathcal{S})$ holds, if there exists an $h \in \mathcal{H}$ such that $h(x_i) = y_i$ for every $i \in [n]$ and $\mathsf{Risk}_D(h) > \varepsilon$.

Definition 1.4.2 (Special PAC Learnability)**.** A realizable learning problem $\mathsf{P} = (\mathcal{X}, \mathcal{Y}, \mathcal{D}, \mathcal{H}, \ell oss)$ is called *special* $(\varepsilon(n), \delta(n))$-PAC learnable if for all $D \in \mathcal{D}$, $n \in \mathbb{N}$, $\Pr_{\mathcal{S} \leftarrow D^n}[\mathrm{Bad}_\varepsilon(D, \mathcal{S})] \leq \delta(n)$. Special $(\varepsilon(n), \delta(n))$-PAC learnability under poisoning attacks is defined similarly, where we demand the inequality to hold for every $\mathsf{A} \in \mathcal{A}_D$ tampering with the training set $\widetilde{\mathcal{S}} \leftarrow \mathsf{A}(\mathcal{S})$.

[17] We note that bounded-budget noise and in fact malicious has also been discussed outside of PAC learning; e.g., [Angluin et al., 1997a] in the membership query model of Angluin [Angluin, 1987].

It is easy to see that if P is special $(\varepsilon(n), \delta(n))$-PAC learnable, then it is $(\varepsilon(n), \delta(n))$-PAC learnable through a 'canonical' learner L who simply finds and outputs a hypothesis h consistent with the training sample set $\mathcal{S}$. Such an h always exists due to the realizability assumption. In fact, many *efficient* PAC learning results follow this very recipe.[18] That motivates our next definition.

Definition 1.4.3 (Efficient Realizability)**.** We say that the problem $\mathsf{P} = (\mathcal{X}, \mathcal{Y}, \mathcal{D}, \mathcal{H}, \ell oss)$ is *efficiently* realizable, if there is a polynomial-time algorithm M, such that for all $D \in \mathcal{D}$, and all $\mathcal{S} \leftarrow D^n$, $M(\mathcal{S})$ outputs some $h \in \mathcal{H}$ such that $\mathsf{Risk}_D(h) = 0$.

Here we define two types of tampering attackers who *do* have control over which examples they tamper with, yet with a 'bounded budget' limiting the number of such instances. Our definitions are inspired by the notions of *adaptive corruption* [Canetti et al., 1996b] and *strong* adaptive corruption defined by Goldwasser, Kalai, and Park [Goldwasser et al., 2015b] in the secure multi-party (coin-flipping) protocols.

Definition 1.4.4 (p-budget attacks)**.** The class of *strong* p-budget (tampering) attacks $\mathcal{A}^p_{\text{bud}} = \cup_{D \in \mathcal{D}} \mathcal{A}_D$ is defined as follows. For $D \in \mathcal{D}$, any $\mathsf{A} \in \mathcal{A}_D$ has a (randomized) tampering algorithm Tam such that:

1. Given access to a sampling oracle for distribution D, $\mathsf{Tam}^D(\cdot)$ always outputs something in $\text{Supp}(D)$.
2. For a training sequence $\mathcal{S} = (d_1, \ldots, d_n)$, the tampered output $\widetilde{\mathcal{S}} = (\widetilde{d}_1, \ldots, \widetilde{d}_n)$ is generated by A inductively, over $i \in [n]$, as $\widetilde{d}_i \leftarrow \mathsf{Tam}^D(1^n, \widetilde{d}_1, \ldots, \widetilde{d}_{i-1}, d_i)$.
3. The number of locations that Tam actually changes d_i is $|\{i \mid d_i \neq \widetilde{d}_i\}| \leq p \cdot n$.

Weak p-budget tampering attacks are defined similarly, with the following difference. The tampering algorithm's execution $\mathsf{Tam}^D(1^n, \widetilde{d}_1, \ldots, \widetilde{d}_{i-1})$ is *not* given d_i, but instead it could either output $o_i \in \text{Supp}(D)$, in which case we let $\widetilde{d}_i = o_i$, or it outputs a special symbol $\perp$, in which case we will have $\widetilde{d}_i = d_i$. Finally, since the weak p-budget attacker should make its decisions without the knowledge of d_i, we shall have $|\{i \mid \perp \neq o_i\}| \leq p \cdot n$.[19]

1.4.2 Results

We first prove that PAC learning is possible under weak p-budget (poisoning) attacks. We then show that this implies a similar possibility result under p-tampering attacks. We then prove that a similar result does *not* hold for *strong* p-budget attacks in general. Our positive result (Theorem 1.4.5) holds even if

[18] For example, properly learning monomials [Valiant, 1984], or using 3-CNF formulae to learn 3-term DNF formulae [Pitt and Valiant, 1988]; the latter is an example of realizable but not proper learning. As an example where the realizability assumption does not necessarily hold, see e.g., [Diochnos, 2016], for learning monotone monomials under a class of distributions - including uniform.

[19] The reason that we did not use the condition $|\{i \mid d_i \neq \widetilde{d}_i\}| \leq p \cdot n$ is the weak p-budget case is that, if the attacker chooses to tamper with the i'th location and simply happens to pick the same $o_i = d_i$, it should still count against its total budget.

the tampering algorithm is given all the history of tampered and untampered blocks (i.e., it is given given input $(1^n, \widetilde{d}_1, \ldots, \widetilde{d}_{i-1}, d_1, \ldots, d_i)$), and our impossibility result (Theorem 1.4.7) holds even if the tampering algorithm is given only d_i.

Theorem 1.4.5 (PAC learning under weak p-budget attacks)**.** *If a realizable problem* $\mathsf{P} = (\mathcal{X}, \mathcal{Y}, \mathcal{D}, \mathcal{H}, \ell oss)$ *is* $(\varepsilon(n), \delta(n))$*-special PAC learnable, then for any* $p \in (0,1)$*,* P *is also* $(\varepsilon\left(n \cdot (1-p)\right), \delta\left(n \cdot (1-p)\right))$*-special PAC learnable under* weak *p-budget (poisoning) attacks.*

Proof. Without loss of generality, we can assume that the tampering algorithm of the adversary is deterministic (otherwise, we can fix the randomness to what is the best for the adversary, and we get a deterministic one again.) For $i \in [n]$ let D_i be the random variable corresponding to the ith example before performing the tampering algorithm and let $(\hat{D}_1, \ldots, \hat{D}_n)$ be the joint distribution of the training sequence after performing the tampering algorithm. Also let T_i be a Boolean random variable which is equal to 1 if the adversary picks to choose the i'th example and $T_i = 0$ otherwise. Using the notation of Definition 1.4.4, $T_i = 0$ if $o_i = \perp$, and $T_i = 1$ otherwise. For $i \in [(1-p) \cdot n]$ let U_i be the random variable corresponding to the index of the i'th zero in the sequence $T_1, \ldots, T_n$, and let $W_i \equiv \hat{D}_{U_i}$. We prove that the joint distribution $(W_1, \ldots, W_{(1-p)\cdot n})$ is distributed identically to $D^{(1-p)\cdot n}$. For every $i \in [(1-p) \cdot n]$ and $d_{\leq i} \in \mathrm{Supp}(D^i)$ we have,

$$\Pr[W_i = d_i \mid W_{\leq i-1} = d_{\leq i-1}]$$

which is

$$\sum_{j=1}^{n} \Pr[\hat{D}_j = d_i \mid W_{\leq i-1} = d_{\leq i-1} \wedge U_i = j] \cdot \Pr[U_i = j]\,.$$

Based on the assumption that the tampering algorithm of the adversary is deterministic, we know that T_i is a function of $D_{\leq i-1}$. On the other hand, D_i is independent of $D_{\leq i-1}$, so D_i and T_i are independent. Therefore, for all predicates $R\colon \mathrm{Supp}(D_{\leq i-1}) \to [0,1]$ such that $R(D_{\leq i-1}) = 1$ implies $T_i = 0$ (i.e., $\Pr[T_i = 0 \mid R(D_{\leq i-1}) = 1] = 1$) we have,

$$\Pr[\hat{D}_i = d \mid R(D_{\leq i-1}) = 1] = \Pr[D_i = d \mid R(D_{\leq i-1}) = 1] = \Pr[D_i = d].$$

It is clear that $W_{\leq i-1} = d_{\leq i-1} \wedge U_i = j$ is a predicate of $D_{\leq j-1}$ as it is a predicate of $\hat{D}_{\leq j-1}$ and $T_{\leq j}$. Also this predicate implies $T_j = 0$, therefore we have,

$$\Pr[W_i = d_i \mid W_{\leq i-1} = d_{\leq i-1}]$$

which is,

$$\sum_{j=1}^{n} \Pr[\hat{D}_j = d_i \mid W_{\leq i-1} = d_{\leq i-1} \wedge U_i = j] \cdot \Pr[U_i = j]$$

which in turn is,

$$\sum_{j=1}^{n} \Pr[D_j = d_i] \cdot \Pr[U_i = j] = \Pr[D = d_i]\,.$$

The above implies $(W_1, \ldots, W_{(1-p)\cdot n}) \equiv D^{(1-p)\cdot n}$.

Now let $\hat{\varepsilon}(n) = \varepsilon\left((1-p)\cdot n\right)$ and $\hat{\delta}(n) = \delta\left((1-p)\cdot n\right)$. Consider the two sets,

$$\begin{cases} \mathsf{Good}_1 &= \{\mathcal{S} \in \mathrm{Supp}(D^n) \colon \overline{\mathrm{Bad}_{\hat{\varepsilon}(n)}(D, \mathcal{S})}\} \\ \mathsf{Good}_2 &= \{\mathcal{S} \in \mathrm{Supp}(D^{(1-p)\cdot n}) \colon \overline{\mathrm{Bad}_{\hat{\varepsilon}(n)}(D, \mathcal{S})}\} \end{cases}.$$

Based on the definition of the event Bad (Definition 1.4.1) we know that,

$$\Pr\left[(\hat{D}_1, \ldots, \hat{D}_n) \in \mathsf{Good}_1 \mid (W_1, \ldots, W_{(1-p)\cdot n}) \in \mathsf{Good}_2\right] = 1.$$

Therefore we have,

$$\begin{aligned} \Pr\left[(\hat{D}_1, \ldots, \hat{D}_n) \in \mathsf{Good}_1\right] &\geq \Pr\left[(W_1, \ldots, W_{(1-p)\cdot n}) \in \mathsf{Good}_2\right] \\ &= \Pr[D^{(1-p)\cdot n} \in \mathsf{Good}_2] \geq 1 - \hat{\delta}(n). \end{aligned}$$

□

Using Theorem 1.4.5, we now prove the following theorem about p-tampering attacks.

Theorem 1.4.6 (PAC learning under weak p-tampering attacks)**.** *For any $p \in (0,1)$, if a realizable problem $\mathsf{P} = (\mathcal{X}, \mathcal{Y}, \mathcal{D}, \mathcal{H}, \ell oss)$ is $(\varepsilon(n), \delta(n))$-special PAC learnable, then for any $q \in (0, 1-p)$, P is also $(\varepsilon'(m), \delta'(m))$-special PAC learnable under p-tampering poisoning attacks for $\varepsilon'(m) = \varepsilon(m\cdot(1-p-q)), \delta'(m) = \mathrm{e}^{-2m\cdot q^2} + \delta(m\cdot(1-p-q))$. Thus, if P is efficiently realizable and special PAC learnable, then P is also efficiently PAC learnable under p-tampering.*

Proof. Consider a p tampering attacker. By the Hoeffding inequality of Lemma 1.2.6, the probability that this attacker tampers with more than $(p+q)\cdot m$ input instances is at most $\mathrm{e}^{-2m\cdot q^2}$. Therefore, with probability $1 - \mathrm{e}^{-2m\cdot q^2}$, this attacker is a *special* case of a weak $(p+q)$-budget attacker, as it does *not* choose the locations of the attack, and thus cannot choose the tampering locations based on the content of the training examples.

Therefore, we can obtain the same bounds of Theorem 1.4.5, but we shall use $p+q$ as the budget and also add $\mathrm{e}^{-2m\cdot q^2}$ to the confidence error. □

Theorem 1.4.7 (PAC learning under strong p-budget attacks)**.** *For any constant $p \in (0,1)$, there is a problem $\mathsf{P} = (\mathcal{X}, \mathcal{Y}, \mathcal{D}, \mathcal{H}, \ell oss)$ that is PAC learnable (under no attack), but it is not PAC learnable under strong p-budget attacks.*

Proof. Suppose $\mathcal{X} = [k]$ where $k = \lceil \frac{2}{p} \rceil$. Let $\mathcal{Y} = \{0,1\}$, and suppose $\mathcal{D}$ consists of all $(x, c(x))_{x \leftarrow \mathcal{X}}$ where $x \leftarrow \mathcal{X}$ is an example drawn from $\mathcal{X}$ uniformly at random and c is an arbitrary function (concept) in $\mathcal{Y}^{\mathcal{X}}$. Let the hypothesis class $\mathcal{H}$ contain all of $\mathcal{Y}^{\mathcal{X}}$, and $\ell oss(b_0, b_1) = |b_0 - b_1|$ is the natural loss for classifiers.

PAC learnability of P trivially follows from the fact that $|\mathcal{X}| = k$ is finite. Therefore, enough samples will reveal the concept function c (defined through D) completely with overwhelming probability for large enough samples n. Consider a concept class which consists of only two functions c_0 and c_1 such that,

$$c_0(i) = 0, \forall i \in [k], \text{ and}$$

$$c_1(i) = \begin{cases} 0 & i \in [k-1] \\ 1 & i = k. \end{cases}$$

Now we propose a strong p-budget adversary A_{sb} (sb stands for (strong budgeted)) that replaces every pair $(k, *)$ it sees with $(k-1, 0)$ until it runs out of its budget which is $p \cdot n$ examples. We denote the distribution of examples after the attack is performed by $A_{sb}(D^n)$. Let us define an event E which is 0 if the adversary runs out of budget at some point and is 1 if she does not run out of budget. Note that if c_0 is being used then the adversary will not do any thing at all and cannot run out of budget. If c_1 is used we can bound the probability of the adversary running out of its budget using Chernoff bound as follows,

$$\Pr[\mathsf{E}] \geq 1 - \mathrm{e}^{\frac{-n}{3k}}.$$

Let L be a learning algorithm that is going to learn a concept c sampled uniformly from $\{c_0, c_1\}$ by looking at n labeled examples sampled from $A_{sb}(D_c^n)$ where $D_c \equiv (d, c(d))_{d \leftarrow U_{[k]}}$. We have,

$$\Pr_{\substack{c \leftarrow \{c_0, c_1\} \\ h \leftarrow L(A_{sb}(D_c^n))}} [h(k) = c(k) \mid \mathsf{E}] \leq \frac{1}{2}.$$

The reason is that two conditional distributions $(A_{sb}(D_{c_0}^n) \mid \mathsf{E})$ and $(A_{sb}(D_{c_1}^n) \mid \mathsf{E})$ are identical, and there is no way for the learning algorithm to find out which of these distributions are being used. Therefore,

$$\begin{aligned}\mathop{\mathbb{E}}_{\substack{c\leftarrow\{c_0,c_1\}\\ h\leftarrow L(A_{sb}(D_c^n))}}[\mathsf{Risk}_{D_c}(h)] &\geq \frac{1}{k}\cdot \Pr_{\substack{c\leftarrow\{c_0,c_1\}\\ h\leftarrow L(A_{sb}(D_c^n))}}[h(k)\neq c(k)]\\ &\geq \frac{1}{k}\cdot \Pr_{\substack{c\leftarrow\{c_0,c_1\}\\ h\leftarrow L(A_{sb}(D_c^n))}}[h(k)\neq c(k)\mid \mathsf{E}]\cdot \Pr[\mathsf{E}]\\ &\geq \frac{1-\mathrm{e}^{\frac{n}{3k}}}{2k}.\end{aligned}$$

Now let $\varepsilon_c(n)$ and $\delta_c(n)$ be the error and confidence that L provides when using n examples sampled from $A(D_c^n)$. We know that,

$$\mathop{\mathbb{E}}_{h\leftarrow L(A_{sb}(D_c^n))}[\mathsf{Risk}_{D_c}(h)] \leq \varepsilon_c(n)+\delta_c(n)$$

which implies,

$$\mathop{\mathbb{E}}_{\substack{c\leftarrow\{c_0,c_1\}\\ h\leftarrow L(A_{sb}(D_c^n))}}[\mathsf{Risk}_{D_c}(h)] \leq \frac{\varepsilon_{c_0}(n)+\delta_{c_0}(n)+\varepsilon_{c_1}(n)+\delta_{c_1}(n)}{2}.$$

Therefore we have,

$$\varepsilon_{c_0}(n)+\delta_{c_0}(n)+\varepsilon_{c_1}(n)+\delta_{c_1}(n) \geq \frac{1-\mathrm{e}^{\frac{-n}{3k}}}{k}$$

which means for any learning algorithm L, one of these values will remain at least $\Omega(1/k)=\Omega(p)$ no matter how many examples the algorithm uses.

□

1.5 Summary and Open Questions

In this chapter, we studied poisoning attacks from a theoritical prospective. Our main contribution was to improved the efficient (polynomial-time) p-tampering biasing attack of [Mahloujifar and Mahmoody, 2017b] to achieve better bias in *polynomial time* and for *real-valued* bounded functions with output in $[0,1]$. This main result allowed us to get improved polynomial-time targeted p-tampering attacks against learners. As in [Mahloujifar and Mahmoody, 2017b], our attacks apply to any learning problem P and any learner L for P.

We also studied the power of p-tampering attacks in the *non-targeted* setting where the adversary's goal is simply to increase the risk of the generated hypothesis. We showed that in this model, p-tampering attacks cannot prevent PAC learnability in 'realizable' settings. We also studied PAC learning under influence of more powerful adversaries who might *choose* the location of training examples that are tampered with but

are still limited to choose $\leq p \cdot n$ such examples. We conclude this section with some natural directions for future work that remain open following our work.

Bounds for attacking specific problems and/or specific learners. The bounds of Corollaries 1.3.4 and 1.3.2 apply to *any* PAC learning problem P and *any* learner L for problem P. Therefore, one can possibly get much stronger bounds for *specific* learning problems, and even for a fixed learning problem P, one can get even better bounds if specific learning algorithms are attacked.

Learning under p-tampering without realizability. The result of Theorems 1.4.5 and 1.4.6 require the realizability assumption to hold for the learning problem P. In what settings do these results extend without the realizability assumption?

Learning under *targeted* p-tampering. Theorems 1.4.5 and 1.4.6 both apply to the case of *non-targeted* poisoning attacks, where the adversary does *not* know the final test example. A natural open question is whether, at least for specific natural cases, this result extends even to the targeted case, where the adversary's tampering strategy could depend on the final test example drawn from the same distribution D as that of training.

Chapter 2

Multi-party Poisoning Attacks

2.1 Introduction

Learning from a set $\mathcal{T} = \{d_1 = (a_1, b_1), \ldots, d_n = (a_n, b_n)\}$ of training examples in a way that the predictions generalize to instances beyond $\mathcal{T}$ is a fundamental problem in learning theory. The goal here is to produce a hypothesis h in such a way that $h(a)$, with high probability, predicts the "correct" label b, where the pair $(a, b) = d$ is sampled from the target (test) distribution D. In the most natural setting, the examples in the training data set $\mathcal{T}$ are also generated from the same distribution D, however this is not always the case (e.g., due to noise in the data).

Multi-party poisoning. In the distributed setting [McMahan and Ramage, 2017, McMahan et al., 2016, Bonawitz et al., 2017, Konečnỳ et al., 2016], the training data $\mathcal{T}$ might be coming from various sources; e.g., it can be generated by m data providers $P_1, \ldots, P_m$ in an online way, while at the end a fixed algorithm, called the aggregator G, generates the hypothesis h based on $\mathcal{T}$. The goal of $P_1, \ldots, P_m$ is to eventually help G construct a hypothesis h that does well (e.g. in the case of classification) in predicting the label b of a given instance a, where $(a, b) \leftarrow D$ is sampled from the final test distribution. The data provided by each party P_i might even be of "different type", so we cannot simply assume that the data provided by P_i is necessarily sampled from the same distribution D. To model this more general setting, we let D_i model the distribution from which the training data $\mathcal{T}_i$ (of P_i) is sampled. Poisoning attacks can naturally be defined in the distributed setting as well [Fung et al., 2018, Bagdasaryan et al., 2018, Blanchard et al., 2017, Hayes and Ohrimenko, 2018] to model adversaries who partially control the training data $\mathcal{T}$. These works, however, focus on attacking and defending specific learning tasks. This leads us to the central question of this section.

What is the inherent *provable power of poisoning attacks in the multi-party setting?*

Answering the above question is critical for understanding the *limits* of provable security against multi-party poisoning.

2.1.1 Summary of Results

We first formalize a new general model multi-party poisoning. We then prove the existence of universal data poisoning attacks in the multi-party setting that apply to any task.

New attack model: (k,p)-poisoning attacks. our first contribution of this section is to formalize a general notion that covers multi-party poisoning attackers that corrupt k out of m data provider parties and furthermore, for each message sent by a corrupted party, the adversary still generates data that is "close" to the honestly generated data. More formally, a (k,p)-poisoning attacker A can first choose to corrupt k of the parties. Then, if a corrupted $\widetilde{P}_i$ is supposed to send the next message, then the adversary will sample $d \leftarrow \widetilde{D}$ for a maliciously chosen distribution $\widetilde{D}$ that is guaranteed to be p to the original distribution D_i in total variation distance. Our (k,p)-poisoning attacks include the so called "p-tampering" attacks of [Mahloujifar et al., 2018a] as special case by letting $k = m$ (m is the number of parties). Moreover, (k,p)-attacks also include the standard model of k static corruption in secure multi-party computation (in cryptography) letting $p = 1$. Our main result in this sections is to prove the *universal* power of (k,p)-poisoning as follow. We show that in *any* m-party learning protocol, there exist a (k,p)-poisoning adversary that increases probability of the produced hypothesis h having a bad property B (e.g., failing on a particular target instance known to the adversary).

(For the formal version of Theorem 2.1.1, see Theorem 2.2.6.)

Theorem 2.1.1 (Power of (k,p)-poisoning attacks–**informal**). *Let $\Pi = (P_1, \ldots, P_m)$ be an m-party learning protocol for an m-party learning problem. Also let B be a bad property defined over the output of the protocol. There is a polynomial time (k,p)-poisoning attack A such that, given oracle access to the data distribution of the parties, A can increase the probability of B from μ to $\mu^{1-kp/m}$.*

Example. By corrupting half of the parties (i.e., $p = 1, k = m/2$) the adversary can increase the probability of any bad event B from $1/100$ to $1/10$.

Universal nature of our attack. Our attacks are *universal* in the sense that they could be applied to *any* learning algorithm for *any* learning task, and they are *dimension-independent* as they applied to any data distribution. On the other hand, our universal attacks rely on an initial vulnerability of arbitrary small *constant* probability that is then amplified through the poisoning attack. As a result, although recent poisoning attacks (e.g., see [Koh et al., 2018]) obtain *stronger* bounds in their attack against specific defenses, our attacks apply to *any* algorithm with any built in defenses.

Deriving attacks on federated learning as special case. Since we allow the distribution of each party in the multi-party case to be completely dependent on that party, our attacks cover the case of model poisoning in federated learning [Bagdasaryan et al., 2018, Bhagoji et al., 2018], in which each party sends something *other* than their plain share of data, as special case. In fact, multiple works have already demonstrated the power of poisoning attacks and defences in the federated learning setting (e.g., see [Fung et al., 2018, Bhagoji et al., 2018, Chen et al., 2018, 2017, Guerraoui et al., 2018, Yin et al., 2018a, Tomsett et al., 2019, Cirincione and Verma, 2019, Han and Zhang, 2019]). Some of these attacks obtain stronger quantitative bounds in their attacks, however this is anticipated as these works investigate attacks on *specific* learners, while a crucial property of our attack is that our attacks come with *provable* bounds and are *universal* in that they apply to *any* learning task and *any* hypothesis class (including neural nets as special case), if there is an initial $\Omega(1)$ vulnerability (for some bad property) over the generated hypothesis.

Note that, our attacks actually do not need the exact history of examples that are used by parties, and only need to know the updates sent by the parties during the course of protocol. Suppose an uncorrected party randomizes its local model (e.g., for differential privacy purposes) and shares an update u_i with the server. Knowledge of u_i is enough for our attacker. One might go even further and ask what if the updates are sent in a secure/private way? Interestingly, our attack work in that model too as it only needs to know the effect of the updates on the central model at the end of round $i-1$ (because all attack wants is to perform a random continuation on the intermediate model).

It also worth mentioning that our attack requires sampling oracles from distributions of all the parties. This might seem that we are giving the adversary too much power. However, we think the right way to define security of federated learning is by giving the adversary everything that hat might be leaked to them. This way of defining security is inspired by cryptography. For instance, when modeling the "chosen plaintext" security of encryption schemes, adversary is given access to an encryption oracle, while one might question how realistic it is. Analogously, In federated learning, the adversary can potentially gather some statistics about the distribution of other parties and learn them over time. However, as mentioned above, we do not need to give adversary access to the actual data of honest parties. Only the public effect of them on the shared model is needed.

2.1.2 Technical Overview

Previous universal poisoning attacks of Mahloujifar and Mahmoody [2017b], Mahloujifar et al. [2018c] for the *single* party case are designed in a setting in which each training example is chosen by the adversary with *independent* probability p. We first describe where exactly the ideas of these works come short of extending

to the multiparty case, and then we explain how to borrow ideas from attacks on coin-tossing protocols in cryptography Ben-Or and Linial [1989], Haitner and Omri [2014] and obtain the desired attacks of this section.

p-tampering attacks and their shortcoming. For starters, let us assume that the adversary gets to corrupt and control k *randomly* selected parties. In this case, it is easy to see that, at the end every single message in the protocol Π between the parties $P_1, \ldots, P_m$ is controlled with exactly probability $p = k/m$ by the adversary A (even though these probabilities are correlated). Thus, at a high level it seems that we should be able to use the p-tampering attacks of Mahloujifar and Mahmoody [2017b], Mahloujifar et al. [2018c] to degrade the quality of the produced hypothesis. However, the catch is that the proof of p-tampering attacks of Mahloujifar and Mahmoody [2017b], Mahloujifar et al. [2018c] (and the bitwise version of Austrin et al. [2017]) crucially rely on the assumption that each message (which in our context corresponds to a training example) is tamperable with *independent* probability p, while corrupting k random parties, leads to tamperable messages in a correlated way.

We prove our main results by first proving a general result about the power of "biasing" adversaries whose goal is to increase the expected value of a random process by controlling each incoming "segment" (aka block) of the random process with probability q (think of q as $\approx p \cdot k/m$). These segments/blocks correspond to single or multiple training examples shared during the learning. As these biasing attacks generalize p-tampering attacks, we simply call them *generalized* p-tampering attacks. We now describe this attack model and clarify how it can be used to obtain Theorem 2.1.1.

Generalized p-tampering: new model for biasing attacks. In this section we introduce generalized p-tampering (biasing) attacks that are defined for any random process $\overline{\mathbf{x}} \equiv (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ and a function $f(\overline{\mathbf{x}}) \in [0,1]$ defined over this process. In order to explain the attack model, first consider the setting where there is no attacker. Now, given a prefix $x_1, \ldots, x_{i-1}$ of the blocks, the next block x_i is simply sampled from its conditional probability distribution $(\mathbf{x}_i \mid x_1, \ldots, x_{i-1})$. (Looking ahead, think of x_i as the i'th training example provided by one of the parties in the interactive learning protocol.) Now, imagine an adversary who enters the game and whose goal is to increase the expected value of a function $f(\mathbf{x}_1, \ldots, \mathbf{x}_n)$ defined over the random process $\overline{\mathbf{x}}$ by tampering with the block-by-block sampling process of $\overline{\mathbf{x}}$ described above. Before the attack starts, there will be a a list $S \subseteq [n]$ of "tamperable" blocks that is *not* necessarily known to the A in advance, but will become clear to him as the game goes on. Indeed, this set S itself will be first sampled according to some fixed distribution $\mathbf{S}$, and the crucial condition we require is that $\Pr[i \in \mathbf{S}] = p$ holds for all $i \in [n]$. After $S \leftarrow \mathbf{S}$ is sampled, the sequence of blocks $(x_1, \ldots, x_n)$ will be sampled block-by-block as follows. Assuming (inductively) that $x_1, \ldots, x_{i-1}$ are already sampled so far, if $i \in S$, then A gets to fully control x_i and determine its value, but if $i \notin S$, then x_i is simply sampled from its original conditional distribution

$(\mathbf{x}_i \mid x_1, \ldots, x_{i-1})$. At the end, the function f is computed over the (adversarially) sampled sequence.

We now explain the intuitive connection between generalized p-tampering attacks and (k, p)-poisoning attacks. The main idea is that we will use a generalized q-tampering attack for $q = p \cdot k/m$ over the random process that lists the sequence of training data provided by the parties during the protocol. Let $\mathbf{S}$ be the distribution over $[n]$ that picks its members through the following algorithm. First choose a set of random parties $\{Q_1, \ldots, Q_k\} \subseteq \{P_1, \ldots, P_m\}$, and then for each message x_j that belongs to Q_i, include the corresponding index j in the final sampled $S \leftarrow \mathbf{S}$ with independent probability p. It is easy to see that $\mathbf{S}$ eventually picks every message with (marginal) probability $q = p \cdot k/m$, but it is also the case that these inclusions are not independent events. Finally, to use the power of generalized p-tampering attacks over the described $\mathbf{S}$ and the random process of messages coming from the parties to get the results of Theorem 2.1.1, roughly speaking, we let a function f model the loss function applied over the produced hypothesis. Therefore, to prove Theorem 2.1.1 it is sufficient to prove Theorem 2.1.1 below which focuses on the power of generalized p-tampering biasing attacks.

Theorem 2.1.2 (Power of generalized p-tampering-**informal**). *Suppose $\overline{\mathbf{x}} \equiv (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ is a joint distribution such that, given any prefix, the remaining blocks could be efficiently sampled in polynomial time. Also let $f\colon \mathrm{Supp}(\overline{\mathbf{x}}) \mapsto [0, 1]$. Then, for any set distribution $\mathbf{S}$ for which $\Pr[i \in \mathbf{S}] = p$ for all i, there is a polynomial-time generalized p-tampering attack (over tampered blocks in $\mathbf{S}$) that increases the average of f over its input from μ to $\mu' \approx \mu^{-p} \cdot \mathbb{E}[f(\overline{x})^{1+p}]$. In particular, if f is boolean function $\mu' \approx \mu^{1-p}$.*

(The formal statement of Theorem 2.1.1 above follows from Theorem 2.3.2 and Lemma 2.3.6.)

Bitwise vs. blockwise attacks. It is easy to see that in the definition of generalized p-tampering attacks, it does not matter whether we define the attack bit-by-bit or block-by-block. The reason is that, even if we break down each block into smaller bits, then still each bit shall eventually fall into the set of tamperable bits, and the model allows correlation between the inclusion and exclusion of each block/bit into the final tamperable set. This is in contrast to the p-tampering model for which this equivalence is not true. In fact, optimal bounds achievable by bitwise p-tampering as proved in Austrin et al. [2017] are *impossible* to achieve in the blockwise p-tampering setting Mahloujifar and Mahmoody [2017b]. Despite this simplification, we still prefer to use a blockwise presentation of the random process, as this way of modeling the problem allows better tracking measures for the attacker's sample complexity.

Ideas Behind the Tampering Attack of Theorem 2.1.1. To prove Theorem 2.1.1 we use ideas from Haitner and Omri [2014], Ben-Or and Linial [1989] in the context of coin-tossing attacks and generalize them using new techniques to obtain our generalized p-tampering attacks.

Rejection sampling attack. The simplified version of our attack can be described as follows. Based on the nature of this attack, we call it the "rejection sampling" (RS) attack. For any prefix of already sampled blocks $(x_1, \ldots, x_{i-1})$, suppose the adversary is given the chance of controlling the next i'th block. The RS tampering then works as follows:

1. Let $x'_i, \ldots, x'_n$ be a random continuation of the random process, conditioned on $(x_1, \ldots, x_{i-1})$.
2. If $s = f(x_1, \ldots, x_{i-1}, x'_i, \ldots, x'_n)$, then if $s = 1$ output y_i, and otherwise (i.e., if $s = 0$) go to Step 1 and repeat the sampling process.

The above attack is inspired by the two-party attack of Haitner and Omri [2014]. Our main contribution is to do the following steps. (1) First, analyze this attack in the generalized tampering setting and show its power, which implies the multiparty case as special case. This already gives an alternative, and in our eyes simpler, proof of the classic result of Ben-Or and Linial [1989] (2) We then extend this attack and its analysis to the *real-output* setting. (3) Finally, we show how to approximate this attack in polynomial time.

2.2 Multi-Party Poisoning Attacks: Definitions and Main Results

Notation. We use bold font (e.g., $\mathbf{x}, \mathbf{S}, \boldsymbol{\alpha}$) to represent random variables, and usually use same non-bold letters for denoting samples from these distributions. We use $d \leftarrow D$ to denote the process of sampling d from the random variable D. By $\mathbb{E}[\boldsymbol{\alpha}]$ we mean the expected value of $\boldsymbol{\alpha}$ over the randomness of $\boldsymbol{\alpha}$, and by $\mathbb{V}[\boldsymbol{\alpha}]$ we denote the variance of random variable $\boldsymbol{\alpha}$. We might use a "processed" version of $\boldsymbol{\alpha}$, and use $\mathbb{E}[f(\boldsymbol{\alpha})]$ and $\mathbb{V}[f(\boldsymbol{\alpha})]$ to denote the expected value and variance, respectively, of $f(\boldsymbol{\alpha})$ over the randomness of $\boldsymbol{\alpha}$. A learning problem $(\mathcal{A}, \mathcal{B}, D, \mathcal{H})$ is specified by the following components. The set $\mathcal{A}$ is the set of possible *instances*, $\mathcal{B}$ is the set of possible *labels*, D is distribution over $\mathcal{A} \times \mathcal{B}$.[1] The set $\mathcal{H} \subseteq \mathcal{B}^{\mathcal{A}}$ is called the *hypothesis space* or *hypothesis class*. An *example* s is a pair $s = (a, b)$ where $x \in \mathcal{A}$ and $y \in \mathcal{B}$. We consider *loss functions* $\ell oss \colon \mathcal{B} \times \mathcal{B} \mapsto \mathbb{R}_+$ where $\ell oss(b', b)$ measures how different the 'prediction' y' (of some possible hypothesis $h(a) = y'$) is from the true outcome y. We call a loss function *bounded* if it always takes values in $[0, 1]$. A natural loss function for classification tasks is to use $\ell oss(b', b) = 0$ if $y = y'$ and $\ell oss(b', b) = 1$ otherwise. The *risk* of a hypothesis $h \in \mathcal{C}$ is the expected loss of h with respect to D, namely $\mathsf{Risk}(h) = \mathbb{E}_{(a,b)\leftarrow D}[\ell oss(h(a), b)]$. The *average error* which quantifies the total error of the protocol is defined as $\mathsf{Err}(D) = \Pr_{h\leftarrow \Pi, (a,b)\leftarrow D}[\ell oss(h(a), b)]$.

Some Useful Inequalities. The following well-known variant of the inequality for the arithmetic mean and the geometric mean could be derived from the Jensen's inequality.

[1] By using joint distributions over $\mathcal{A} \times \mathcal{B}$, we jointly model a set of distributions over $\mathcal{A}$ and a concept class mapping $\mathcal{A}$ to $\mathcal{B}$ (perhaps with noise and uncertainty).

Lemma 2.2.1 (Weighted AM-GM inequality)**.** *For any $n \in \mathbb{N}$, let $z_1, ..., z_n$ be a sequence of non-negative real numbers and let $w_1, ..., w_n$ be such that $w_i \geq 0$ for every $i \in [n]$ and $\sum_{i=1}^{n} w_i = 1$. Then, it holds that*

$$\sum_{i=1}^{n} w_i z_i \geq \prod_{i=1}^{n} z_i^{w_i} \ .$$

The following lemma provides a tool for lower bounding the gap between the two sides of Jensen's inequality, also known as the Jensen gap.

Lemma 2.2.2 (Lower bound for Jensen gap Liao and Berg [(accepted in 2017])**.** *Let $\boldsymbol{\alpha}$ be a real-valued random variable,* $\mathrm{Supp}(\boldsymbol{\alpha}) \subseteq [0,1]$*, and $\mathbb{E}[\boldsymbol{\alpha}] = \mu$. Let $\varphi(\cdot)$ be twice differentiable on $[0,1]$, and let $h_b(a) = \frac{\varphi(a)-\varphi(b)}{(a-b)^2} - \frac{\varphi'(a)}{a-b}$. Then,*

$$\mathbb{E}[\varphi(\boldsymbol{\alpha})] - \varphi(\mu) \geq \mathbb{V}[\boldsymbol{\alpha}] \cdot \inf_{a \in [0,1]} \{h_\mu(a)\} \ .$$

Definition 2.2.3 (Multi-party learning protocols)**.** An m-party learning protocol Π for the m-party learning problem $(\mathcal{D}, \mathcal{H})$ consists of an aggregator function G and m (interactive) data providers $\mathcal{P} = \{P_1, \ldots, P_m\}$. For each data provider P_i, there is a distribution $D_i \in \mathcal{D}$ that models the (honest) distribution of labeled samples generated by P_i, and there is a final (test) distribution D that $\mathcal{P}, G$ want to learn jointly. The protocol runs in r rounds and at each round, based on the protocol Π, one particular data owner P_i broadcasts a single labeled example $(a, b) \leftarrow D_i$.[2] In the last round, the aggregator function G maps the the messages to an output hypothesis $h \in \mathcal{H}$.

Now, we define poisoning attackers that target multi-party protocols. We formalize a more general notion that includes p-tampering attacks and k-party corruption as special case.

Definition 2.2.4 (Multi-party (k,p)-poisoning attacks)**.** A (k,p)-poisoning attack against an m-party learning protocol Π is defined by an adversary A who can control a subset $\mathcal{C} \subseteq [m]$ of the parties where $|\mathcal{C}| = k$. The attacker A shall pick the set $\mathcal{C}$ at the beginning. At each round j of the protocol, if a data provider $P_i \in \mathcal{C}$ is supposed to broadcast the next example from its distribution D_i, the adversary can partially control this sample using the tampered distribution $\widetilde{D}$ such that $|\widetilde{D} - D_i| \leq p$ in total variation distance. Note that the distribution $\widetilde{D}$ can depend on the history of examples broadcast so far, but the requirement is that, conditioned on this history, the malicious message of adversary modeled by distribution $\widetilde{D}$, is at most p-statistically far from D_i. We use Π_{A} to denote the protocol in presence of A. We also define the following notions. A is a *plausible* adversary, if it always holds that $\mathrm{Supp}(\widetilde{D}) \subseteq \mathrm{Supp}(D_i)$. A is *efficient*

[2] We can directly model settings where more data is exchanged in one round, however, we stick to the simpler definition w.l.o.g.

if it runs in polynomial time in the total length of the messages exchanged during the protocol (from the beginning till end).

Remark 2.2.5 (Static vs. adaptive corruption)**.** Definition 2.2.4 focuses on corrupting k parties statically. A natural extension of this definition in which the set $\mathcal{C}$ is chosen *adaptively* Canetti et al. [1996a] while the protocol is being executed can also be defined naturally. In this section, however, we focus on static corruption, and leave the possibility of improving our results in the adaptive case for future work.

We now formally state our result about the power of (k, p)-poisoning attacks.

Theorem 2.2.6 (Power of efficient multi-party poisoning)**.** *In any m-party protocol Π for parties $\mathcal{P} = \{P_1, \ldots, P_m\}$, for any $p \in [0, 1]$ and $k \in [m]$, the following hold where M is the total length of the messages exchanged.*

1. *For any bad property $B : \mathcal{H} \to \{0, 1\}$, there is a* plausible *(k, p)-poisoning attack A that runs in time* $\mathrm{poly}(M/\varepsilon)$ *and increases the probability of B from μ (in the no-attack setting) to*

$$\mu' \geq \mu^{1-p} - \varepsilon.$$

2. *If the (normalized) loss function is bounded (i.e., it outputs in $[0, 1]$), then there is a* plausible*, (k, p)-poisoning A that runs in time* $\mathrm{poly}(M/\varepsilon)$ *and increases the average error of the protocol as*

$$\begin{aligned} \mathsf{Err}_{\mathsf{A}}(D) &\geq \mathsf{Err}(D)^{-p} \cdot \mathop{\mathbb{E}}_{h \leftarrow \Pi}[\mathsf{Risk}(h, D)^{1+p}] \\ &\geq \mathsf{Err}(D) + \frac{p \cdot k}{2m} \cdot \nu - \varepsilon \end{aligned}$$

where $\nu = \mathbb{V}_{h \leftarrow \Pi}[\mathsf{Risk}(h, D)]$ and $\mathbb{V}[\cdot]$ is the variance.

Allowing different distributions in different rounds. In Definition 2.2.4, we restrict the adversary to remain "close" to D_i for each message sent out by one of the corrupted parties. A natural question is: what happens if we allow the parties distributions to be different in different rounds. For example, in a round j, a party P_i might send *multiple* training examples $D^{(j)} = \left(d_1^{(j)}, d_2^{(j)}, \ldots, d_k^{(j)}\right)$, and we want to limit the *total* statistical distance between the distribution of the larger message $D^{(j)}$ from D_i^k (i.e., k iid samples from D_i).[3] We emphasize that, our results extend to this more general setting as well. In particular, the proof of Theorem 2.2.6 directly extends to a more general setting where we can allow the honest distribution D_i of each party i to also depend on the round j in which these messages are sent. Thus, we can use a

[3]Note that, even if each block in $\left(d_1^{(j)}, d_2^{(j)}, \ldots, d_k^{(j)}\right)$ remains p-close to D_i, their joint distribution could be quite far from D_i^k.

round-specific distribution $D_i^{(j)}$ to model the joint distribution of *multiple* samples $D^{(j)} = \left(d_1^{(j)}, d_2^{(j)}, \ldots, d_k^{(j)}\right)$ that are sent out in the j'th round by the party P_i. This way, we can obtain the stronger form of attacks that remain statistically close to the joint (correct) distribution of the (multi-sample) messages sent in a round. In fact, as we will discuss shortly $D^{(j)}$ might be of completely different type.

Allowing randomized aggregation. The aggregator G is a simple function that maps the transcript of the exchanged messages to a hypothesis h. A natural question is: what happens if we generalize this to the setting where G is allowed to be randomized. We note that in Theorem 2.2.6, Part 2 can allow G to be randomized, but Parts 1 and 3 need deterministic aggregation. The reason is that for those parts, we need the transcript to determine the confidence and average error functions. One general way to make up for randomized aggregation is to allow the parties to inject randomness into the transcript as they run the protocol by sending messages that are not necessarily learning samples from their distribution D_i. As described above, our attacks extend to this more general setting as well. Otherwise, we will need the adversary to be able to also depend on the randomness of G, but that is also a reasonable assumption if the aggregation is used using public beacon that could be obtained by the adversary as well.

Before proving Theorem 2.2.6, we need to develop our main result about the power of generalized p-tampering attacks. In Section 2.3, we develop such tools, and then in Section 2.3.2 we prove Theorem 2.2.6.

2.3 Multi-Party Poisoning via Generalized p-Tampering

To prove our Theorem 2.2.6 we interpret the multi-party learning protocol as a coin tossing protocol in which the final bit is 1 if h has the (bad) property B. We define a corresponding attack model in coin tossing protocols that can be directly used to obtain the desired goal; this model is called *generalized* p-tampering. Below, we formally state our main result about the power of generalized p-tampering attacks. We start by formalizing some notation and definitions.

Notation. By $\mathbf{x} \equiv \mathbf{y}$ we denote that the random variables $\mathbf{x}$ and $\mathbf{y}$ have the same distributions. Unless stated otherwise, by using a bar over a variable, we emphasize that it is a vector. By $\overline{\mathbf{x}} \equiv (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$ we refer to a joint distribution over vectors with n components. For a joint distribution $\overline{\mathbf{x}} \equiv (\mathbf{x}_1, \ldots, \mathbf{x}_n)$, we use $\mathbf{x}_{\leq i}$ to denote the joint distribution of the first i variables $\overline{\mathbf{x}} \equiv (\mathbf{x}_1, \ldots, \mathbf{x}_i)$. Also, for a vector $\overline{x} = (x_1 \ldots x_n)$ we use $x_{\leq i}$ to denote the prefix $(x_1, \ldots, x_i)$. For a randomized algorithm $L(\cdot)$, by $y \leftarrow L(x)$ we denote the randomized execution of L on input x outputting y. For a distribution $(\mathbf{x}, \mathbf{y})$, by $(\mathbf{x} \mid \mathbf{y})$ we denote the conditional distribution $(\mathbf{x} \mid \mathbf{y} = y)$. By $\mathrm{Supp}(D) = \{d \mid \Pr[D = d] > 0\}$ we denote the support set of D. By

$T^D(\cdot)$ we denote an algorithm $T(\cdot)$ with oracle access to a sampler for D that upon every query returns fresh samples from D. By D^n we denote the distribution that returns n iid samples from D.

Definition 2.3.1 (Valid prefixes)**.** Let $\overline{\mathbf{x}} \equiv (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ be an arbitrary joint distribution. We call $x_{\leq i} = (x_1, \ldots, x_i)$ a *valid prefix* for $\overline{\mathbf{x}}$ if there exist $x_{i+1}, \ldots, x_n$ such that $(x_1, \ldots, x_n) \in \mathrm{Supp}(\overline{\mathbf{x}})$. $\mathrm{ValPref}(\overline{\mathbf{x}})$ denotes the set of all valid prefixes of $\overline{\mathbf{x}}$.

Definition 2.3.2 (Tampering with random processes)**.** Let $\overline{\mathbf{x}} \equiv (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ be an arbitrary joint distribution. We call a (potentially randomized and possibly computationally unbounded) algorithm Tam an (online) *tampering algorithm for* $\overline{\mathbf{x}}$ if given any prefix $x_{\leq i-1} \in \mathrm{ValPref}(\overline{\mathbf{x}})$, we have

$$\Pr_{x_i \leftarrow \mathsf{Tam}(x_{\leq i-1})}[x_{\leq i} \in \mathrm{ValPref}(\overline{\mathbf{x}})] = 1 \ .$$

Namely, $\mathsf{Tam}(x_{\leq i-1})$ outputs x_i such that $x_{\leq i}$ is again a valid prefix. We call Tam an *efficient* tampering algorithm for $\overline{\mathbf{x}}$ if it runs in time $\mathrm{poly}(N)$ where N is maximum bit length to represent any $\overline{x} \in \mathrm{Supp}(\overline{\mathbf{x}})$.

Definition 2.3.3 (Online samplers)**.** We call OnSam an *online sampler* for $\overline{\mathbf{x}} \equiv (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ if for all $x_{\leq i-1} \in \mathrm{ValPref}(\overline{\mathbf{x}})$, $\mathsf{OnSam}(n, x_{\leq i-1}) \equiv \mathbf{x}_i$. Moreover, we call $\overline{\mathbf{x}} \equiv (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ *online samplable* if it has an online sampler that runs in time $\mathrm{poly}(N)$ where N is maximum bit length of any $\overline{x} \in \mathrm{Supp}(\overline{\mathbf{x}})$.

Notation for tampering distributions. Let $\overline{\mathbf{x}} \equiv (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ be an arbitrary joint distribution and Tam a tampering algorithm for $\overline{\mathbf{x}}$. For any subset $S \subseteq [n]$, we define $\overline{\mathbf{y}} \equiv \langle \overline{\mathbf{x}} \parallel \mathsf{Tam} \rangle S$ to be the joint distribution that is the result of online tampering of Tam over set S, where $\overline{\mathbf{y}} \equiv (\mathbf{y}_1, \ldots, \mathbf{y}_n)$ is sampled inductively as follows. For every $i \in [n]$, suppose $y_{\leq i-1}$ is the previously sampled block. If $i \in S$, then the i^{th} block $\mathbf{y}_i$ is generated by the tampering algorithm $\mathsf{Tam}(y_{\leq i-1})$, and otherwise, $\mathbf{y}_i$ is sampled from $(\mathbf{x}_i \mid \mathbf{x}_{i-1} = y_{\leq i-1})$. For any *distribution* $\mathbf{S}$ over subsets of $[n]$, by $\langle \overline{\mathbf{x}} \parallel \mathsf{Tam} \rangle \mathbf{S}$ we denote the random variable that can be sampled by first sampling $S \leftarrow \mathbf{S}$ and then sampling $\overline{y} \leftarrow \langle \overline{\mathbf{x}} \parallel \mathsf{Tam} \rangle S$.

Definition 2.3.4 (p-covering)**.** Let $\mathbf{S}$ be a distribution over the subsets of $[n]$. We call $\mathbf{S}$ a *p-covering* distribution on $[n]$ (or simply p-covering, when n is clear from the context), if for all $i \in [n], \Pr_{S \leftarrow \mathbf{S}}[i \in S] = p$.

The following theorem states the power of generalized p-tampering attacks.

Theorem 2.3.5 (Biasing of bounded functions through generalizing p-tampering)**.** *Let $\mathbf{S}$ be a p-covering distribution on $[n]$, $\overline{\mathbf{x}} \equiv (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ be a joint distribution, $f \colon \mathrm{Supp}(\overline{\mathbf{x}}) \mapsto [0,1]$, and $\mu = \mathbb{E}[f(\overline{\mathbf{x}})]$. Then, for any $\varepsilon \in [0,1]$, there exists a tampering algorithm Tam_ε that, given oracle access to f and any online sampler OnSam for $\overline{\mathbf{x}}$, it runs in time $\mathrm{poly}(N/\varepsilon)$, where N is the bit length of any $\overline{x} \leftarrow \overline{\mathbf{x}}$, and for $\overline{\mathbf{y}}_\varepsilon \equiv$*

$\langle \overline{\mathbf{x}} \parallel \mathsf{Tam}_{\varepsilon}^{f,\mathsf{OnSam}} \rangle \mathbf{S}$, *it holds that*

$$\mathbb{E}\left[f(\overline{\mathbf{y}}_{\varepsilon})\right] \geq \mu^{-p} \cdot \mathbb{E}\left[f(\overline{\mathbf{x}})^{1+p}\right] - \varepsilon \ .$$

Special case of Boolean functions. When the function f is Boolean, we get $\mu^{-p} \cdot \mathbb{E}[f(\overline{\mathbf{x}})^{1+p}] = \mu^{1-p} \geq \mu(1 + \Omega_{\mu}(p))$, which matches the bound proved in Ben-Or and Linial [1989] for the special case of $p = k/n$ for integer $k \in [n]$ and for $\mathbf{S}$ that is uniformly random subset of $[n]$ of size k. (The same bound for the case of 2 parties was proved in Haitner and Omri [2014] with extra properties). Even for this case, compared to Ben-Or and Linial [1989], Haitner and Omri [2014] our result is more general, as we can allow $\mathbf{S}$ with arbitrary $p \in [0, 1]$ *and* achieve a polynomial time attack given oracle access to an online sampler for $\overline{\mathbf{x}}$. The work of Haitner and Omri [2014] also deals with polynomial time attackers for the special case of 2 parties, but their efficient attackers use a different oracle (i.e., OWF inverter), and it is not clear whether or not their attack extend to the case of more then 2 parties. Finally, both Ben-Or and Linial [1989], Haitner and Omri [2014] prove their bound for the *geometric* mean of the averages for different $S \leftarrow \mathbf{S}$, while we do so for their arithmetic mean, but we emphasize that this is enough for all of our applications.

The bounds of Theorem 2.3.2 for both cases rely on the quantity $\mu' = \mu^{-p} \cdot \mathbb{E}[f(\overline{\mathbf{x}})^{1+p}]$. A natural question is: how large is μ' compared to μ? As discussed above, for the case of Boolean f, we already know that $\mu' \geq \mu$, but that argument does not apply to the real-output f. A simple application of Jensen's inequality shows that $\mu \leq \mu'$ in general, but that still does not mean that $\mu' \gg \mu$.

General case of real-output functions: relating the bias to the variance. If $\mathbb{V}[f(\overline{\mathbf{x}})] = 0$, then no tampering attack can achieve any bias, so any gap achieved between μ' and μ shall somehow depend on the variance of $f(\overline{\mathbf{x}})$. In the following, we show that this gap does exist and that $\mu' - \mu \geq \Omega(p \cdot \mathbb{V}[f(\overline{\mathbf{x}})])$. similar results (relating the bias the the variance of the original distribution) were previously proved Mahloujifar et al. [2018c], Mahloujifar and Mahmoody [2017b], Austrin et al. [2014b] for the special case of p-tampering attacks (i.e., $\mathbf{S}$ chooses every $i \in [n]$ independently with probability p). Here we obtain a more general statement that holds for any p-covering set structure $\mathbf{S}$.

Using Lemma 2.3.6 below for $\boldsymbol{\alpha} \equiv f(\overline{\mathbf{x}})$, we immediately get $\Omega(p \cdot \mathbb{V}[f(\overline{\mathbf{x}})])$ lower bounds for the bias achieved by (both versions of) the attackers of Theorem 2.3.2 for the general case of real-valued functions and arbitrary p-covering set distribution $\mathbf{S}$. See full version of paper for the proof.

Lemma 2.3.6. *Let* $\boldsymbol{\alpha}$ *be any real-valued random variable over* $\mathrm{Supp}(\boldsymbol{\alpha}) \subseteq [0, 1]$*, and* $p \in [0, 1]$*. Let* $\mu = \mathbb{E}[\boldsymbol{\alpha}]$ *be the expected value of* $\boldsymbol{\alpha}$*,* $\nu = \mathbb{V}[\boldsymbol{\alpha}]$ *be the variance of* $\boldsymbol{\alpha}$*. Then, it holds that*

$$\mu^{-p} \cdot \mathbb{E}[\boldsymbol{\alpha}^{1+p}] - \mu \geq \frac{p \cdot (p+1)}{2 \cdot \mu^{p}} \cdot \nu \geq \frac{p}{2} \cdot \nu \ .$$

2.3.1 Proving Theorem 2.3.2 For Computationally Unbounded Adversaries.

The construction below describes a computationally unbounded biasing algorithm that achieves the bounds of Theorem 2.3.2. Please see the full version of paper for the proof of computationally bounded setting where we carefully approximate the construction bellow by a computationally bounded polynomial-time biasing adversary.

Construction 2.3.7 (Rejection-sampling tampering)**.** Let $\overline{\mathbf{x}} \equiv (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ and $f\colon \mathrm{Supp}(\overline{\mathbf{x}}) \mapsto [0,1]$. The *rejection sampling* tampering algorithm RejSam^f works as follows. Given the valid prefix $y_{\leq i-1} \in \mathrm{ValPref}(\overline{\mathbf{x}})$, the tampering algorithm would do the following:

1. Sample $y_{\geq i} \leftarrow (\mathbf{x}_{\geq i} \mid y_{\leq i-1})$ by using the online sampler for f.

2. If $s = f(y_1, \ldots, y_n)$, then with probability s output y_i, otherwise go to Step 1 and repeat the process.

We will first prove a property of the rejection sampling algorithm when applied on every block.

Definition 2.3.8 (Notation for partial expectations of functions)**.** Suppose $f\colon \mathrm{Supp}(\overline{\mathbf{x}}) \mapsto \mathbb{R}$ is defined over a joint distribution $\overline{\mathbf{x}} \equiv (\mathbf{x}_1, \ldots, \mathbf{x}_n)$, $i \in [n]$, and $x_{\leq i} \in \mathrm{ValPref}(\overline{\mathbf{x}})$. Then, using a small hat, we define the notation $\hat{f}[x_{\leq i}] = \mathbb{E}_{\overline{x} \leftarrow (\overline{\mathbf{x}} | x_{\leq i})}[f(\overline{x})]$. (In particular, for $\overline{x} = x_{[n]}$, we have $\hat{f}[\overline{x}] = f(\overline{x})$.)

Claim 2.3.9. *If* $\langle \overline{\mathbf{x}} \parallel \mathsf{RejSam}^f \rangle[n] \equiv \overline{\mathbf{y}}^{[n]} \equiv (\mathbf{y}_1, \ldots, \mathbf{y}_n)$*. Then, for every valid prefix* $y_{\leq i} \in \mathrm{ValPref}[\overline{\mathbf{x}}]$*,*

$$\frac{\Pr[\mathbf{y}_{\leq i} = y_{\leq i}]}{\Pr[\mathbf{x}_{\leq i} = y_{\leq i}]} = \frac{\hat{f}[y_{\leq i}]}{\mu} \ .$$

Proof. Based on the description of RejSam^f, for any $y_{\leq i} \in \mathrm{ValPref}(\overline{\mathbf{x}})$ the following equation holds for the probability of sampling y_i conditioned on prefix $y_{\leq i-1}$.

$$\begin{aligned}\Pr[\mathbf{y}_i = y_i \mid y_{\leq i-1}] = \Pr[\mathbf{x}_i = y_i \mid y_{\leq i-1}] \cdot \hat{f}[y_{\leq i}] \\ + (1 - \hat{f}[y_{\leq i-1}]) \cdot \Pr[\mathbf{y}_i = y_i \mid y_{\leq i-1}].\end{aligned}$$

The first term in this equation corresponds to the probability of selecting and accepting in the first round of sampling and the second term corresponds to the probability of selecting and accepting in any round except the first round. Therefore we have

$$\Pr[\mathbf{y}_i = y_i \mid y_{\leq i-1}] = \frac{\hat{f}[y_{\leq i}]}{\hat{f}[y_{\leq i-1}]} \cdot \Pr[\mathbf{x}_i = y_i \mid y_{\leq i-1}] \ ,$$

which implies that

$$\begin{aligned}\Pr[\mathbf{y}_{\le i} = y_{\le i}] &= \prod_{j\in[i]} \left(\frac{\hat{f}[y_{\le j}]}{\hat{f}[y_{\le j-1}]}\right) \cdot \Pr[\mathbf{x}_{\le i} = y_{\le i}] \\ &= \frac{\hat{f}[y_{\le i}]}{\mu} \cdot \Pr[\mathbf{x}_{\le i} = y_{\le i}] \ .\end{aligned}$$

□

Now, we prove two properties for *any* tampering algorithm (not just rejection sampling) over a p-covering distribution.

Lemma 2.3.10. *Let* $\mathbf{S}$ *be* p*-covering for* $[n]$ *and* $\overline{y} \in \mathrm{Supp}(\overline{\mathbf{x}})$*. For any* $S \in \mathrm{Supp}(\mathbf{S})$ *and an arbitrary tampering algorithm* Tam *for* $\overline{\mathbf{x}}$*, let* $\overline{\mathbf{y}}^S \equiv \langle \overline{\mathbf{x}} \parallel \mathsf{Tam}\rangle S$*. Then,*

$$\prod_{S\in 2^{[n]}} \left(\frac{\Pr[\overline{\mathbf{y}}^S = \overline{y}]}{\Pr[\overline{\mathbf{x}} = \overline{y}]}\right)^{\Pr[\mathbf{S}=S]} = \left(\frac{\Pr[\overline{\mathbf{y}}^{[n]} = \overline{y}]}{\Pr[\overline{\mathbf{x}} = \overline{y}]}\right)^p \ .$$

Proof. For every $y_{\le i} \in \mathrm{ValPref}(\overline{\mathbf{y}}^{[n]}) \subseteq \mathrm{ValPref}(\overline{\mathbf{x}})$ define $\rho[y_{\le i}]$ as

$$\rho[y_{\le i}] = \frac{\Pr[\mathbf{y}_i^{[n]} = x_i \mid \mathbf{y}^{[n]}{}_{\le i-1} = y_{\le i-1}]}{\Pr[\mathbf{x}_i = x_i \mid \mathbf{x}_{\le i-1} = y_{\le i-1}]} \ .$$

Then, for all $\overline{y} \in \mathrm{ValPref}(\overline{\mathbf{y}}^S) \subseteq \mathrm{ValPref}(\overline{\mathbf{x}})$ we have

$$\Pr[\overline{\mathbf{y}}^S = y] = \Pr[\overline{\mathbf{x}} = y] \cdot \prod_{i\in S} \rho[y_{\le i}] \ .$$

Therefore we have

$$\prod_{S\in 2^{[n]}} \left(\frac{\Pr[\overline{\mathbf{y}}^S = \overline{y}]}{\Pr[\overline{\mathbf{x}} = \overline{y}]}\right)^{\Pr[\mathbf{S}=S]} = \left(\prod_{i\in[n]} \rho[y_{\le i}]\right)^p \ .$$

□

Claim 2.3.11. *Suppose* $\mathbf{S}$ *is* p*-covering on* $[n]$*,* $\overline{\mathbf{y}}^S \equiv \langle \overline{\mathbf{x}} \parallel \mathsf{Tam}\rangle S$ *for any* $S \leftarrow \mathbf{S}$*, and* $\overline{\mathbf{y}} \equiv \langle \overline{\mathbf{x}} \parallel \mathsf{Tam}\rangle \mathbf{S}$ *for an arbitrary tampering algorithm* Tam *for* $\overline{\mathbf{x}}$*. Then, it holds that*

$$\mathbb{E}[f(\overline{\mathbf{y}})] \ge \sum_{\overline{y}\in\mathrm{Supp}(\overline{\mathbf{x}})} \Pr[\overline{\mathbf{x}} = \overline{y}] \cdot f(\overline{y}) \cdot \left(\frac{\Pr[\overline{\mathbf{y}}^{[n]} = \overline{y}]}{\Pr[\overline{\mathbf{x}} = \overline{y}]}\right)^p \ .$$

Proof. Let $h_{S,\overline{y}} = \frac{\Pr[\overline{\mathbf{y}}^S = \overline{y}]}{\Pr[\overline{\mathbf{x}} = \overline{y}]}$. Also let $\mathcal{Z} \subseteq \mathrm{Supp}(\overline{\mathbf{x}})$. Note that $\mathrm{Supp}(\overline{\mathbf{y}}^S) \subseteq \mathcal{Z}$ for any $S \subseteq [n]$. Therefore, we have $\mathbb{E}[f(\overline{\mathbf{y}})] = \mathbb{E}_{S \leftarrow \mathbf{S}} \mathbb{E}_{\overline{y} \leftarrow \overline{\mathbf{y}}^S}[f(y)]$ is equal to

$$
\begin{aligned}
&\sum_{S \in 2^{[n]}} \Pr[\mathbf{S} = S] \cdot \sum_{\overline{y} \in \mathcal{Z}} \Pr[\overline{\mathbf{y}}^S = \overline{y}] \cdot f(\overline{y}) \\
&= \sum_{S \in 2^{[n]}} \Pr[\mathbf{S} = S] \cdot \sum_{\overline{y} \in \mathcal{Z}} h_{S,\overline{y}} \cdot \Pr[\overline{\mathbf{x}} = \overline{y}] \cdot f(\overline{y}) \\
&= \sum_{\overline{y} \in \mathcal{Z}} \Pr[\overline{\mathbf{x}} = \overline{y}] \cdot f(\overline{y}) \cdot \sum_{S \in 2^{[n]}} \Pr[\mathbf{S} = S] \cdot h_{S,\overline{y}} \\
&\qquad \text{(by AM-GM inequality)} \\
&\geq \sum_{\overline{y} \in \mathcal{Z}} \Pr[\overline{\mathbf{x}} = \overline{y}] \cdot f(\overline{y}) \cdot \prod_{S \in 2^{[n]}} h_{S,\overline{y}}^{\Pr[\mathbf{S}=S]} \\
&\qquad \text{(by } p\text{-covering of } \mathbf{S} \text{ and Lemma 2.3.10)} \\
&= \sum_{\overline{y} \in \mathcal{Z}} \Pr[\overline{\mathbf{x}} = \overline{y}] \cdot f(\overline{y}) \cdot \left(\frac{\Pr[\overline{\mathbf{y}}^{[n]} = \overline{y}]}{\Pr[\overline{\mathbf{x}} = \overline{y}]} \right)^p .
\end{aligned}
$$

□

We now prove the main result using the one-rejection sampling tampering algorithm and also relying on the p-covering property of $\mathbf{S}$. In particular, if $\overline{\mathbf{y}} \equiv \langle \overline{\mathbf{x}} \parallel \mathsf{RejSam}^f \rangle \mathbf{S}$, then by Claims 2.3.11 and 2.3.9 we have

$$
\begin{aligned}
\mathbb{E}[f(\overline{\mathbf{y}})] &\geq \sum_{\overline{y} \in \mathrm{Supp}(\overline{\mathbf{x}})} \left(\frac{\Pr[\overline{\mathbf{y}}^{[n]} = \overline{y}]}{\Pr[\overline{\mathbf{x}} = \overline{y}]} \right)^p \cdot \Pr[\overline{\mathbf{x}} = \overline{y}] \cdot f(\overline{y}) \\
&\qquad \text{(by Claim 2.3.9)} \\
&= \sum_{\overline{y} \in \mathrm{Supp}(\overline{\mathbf{x}})} \left(\frac{f(\overline{y})}{\mu} \right)^p \cdot \Pr[\overline{\mathbf{x}} = \overline{y}] \cdot f(\overline{y}) \\
&= \mu^{-p} \cdot \sum_{\overline{y} \in \mathrm{Supp}(\overline{\mathbf{x}})} \Pr[\overline{\mathbf{x}} = \overline{y}] \cdot f(\overline{y})^{1+p} \\
&= \mu^{-p} \cdot \mathbb{E}[f(\overline{\mathbf{x}})^{1+p}] .
\end{aligned}
$$

2.3.2 Obtaining (k, p)-Poisoning Attacks: Proof of Theorem 2.2.6 using Theorem 2.3.2

In this section, we formally prove Theorem 2.2.6 using Theorems 2.3.2. We first prove the first part of theorem about the boolean property.

Proof of Theorem 2.2.6 part 1. For a subset $C \subseteq [m]$ let $P_C = \{P_i; i \in C\}$ and R_C be the subset of rounds where one of the parties in P_C sends an example. Also for a subset $S \subseteq [n]$, we define $\mathbf{Bion}(S, p)$ to be a distribution over all the subsets of S, where each subset $S' \subseteq S$ hast the probability $p^{|S'|} \cdot (1-p)^{|S|-|S'|}$. Now, consider the covering $\mathbf{S}$ of the set $[n]$ which is distributed equivalent to the following process. First sample a uniform subset C of $[m]$ of size k. Then sample and output a set S sampled from $\mathbf{Bion}(R_C, p)$. $\mathbf{S}$ is clearly a $(p \cdot \frac{k}{m})$-covering. We use this covering to prove the theorem. For $j \in [n]$ let $w(j)$ be the index of the provider at round j and let $D_{w(j)}$ be the designated distribution of the jth round and let $\overline{\mathbf{x}} = D_{w(1)} \times \cdots \times D_{w(n)}$.

We define a function $f : \mathrm{Supp}(\overline{\mathbf{x}}) \to \{0, 1\}$, which is a Boolean function and is 1 if the output of the protocol has the property B, and otherwise it is 0. Now we use Theorem 2.3.2. We know that $\mathbf{S}$ is a $(p \cdot \frac{k}{m})$-covering for $[n]$. Therefore of Theorem 2.3.2, there exist an $\mathrm{poly}(m/\varepsilon)$ time tampering algorithm Tam_ε that changes $\overline{\mathbf{x}}$ to $\overline{\mathbf{y}} \equiv \langle \overline{\mathbf{x}} \parallel \mathsf{Tam}_\varepsilon^{f,\mathsf{OnSam}} \rangle \mathbf{S}$ where $\mathbb{E}[f(\overline{\mathbf{y}})] \geq \mathbb{E}[f(\overline{\mathbf{y}})]^{1-pk/m} - \varepsilon$.

By an averaging argument, we can conclude that there exist a set $C \in [m]$ of size k for which the distribution $\mathbf{Bion}(R_C, p)$ produces average output at least $\mathbb{E}[f(\overline{\mathbf{y}})]^{1-pk/m} - \varepsilon$. Note that the measure of empty set in $\mathbf{Bion}(R_C, p)$ is exactly equal to $1-p$ which means with probability $1-p$ the adversary will not tamper with any of the blocks, therefore, the statistical distance $|\overline{\mathbf{x}} - \langle \overline{\mathbf{x}} \parallel \mathsf{Tam}_\varepsilon^{f,\mathsf{OnSam}} \rangle \mathbf{Bion}(R_C, p)|$ is at most p. This concludes the proof. □

Now we prove the second part using Theorem 2.3.2 and Lemma 2.3.6.

Proof of Theorem 2.2.6 part 2. Now we prove the second part. The second part is very similar to first part except that the function that we define here is a real valued function. Consider the function $f_2 : \mathrm{Supp}(\overline{\mathbf{x}}) \to [0, 1]$ which is defined to be the risk of the output hypotheses. Now by Theorem 2.3.2 and Lemma 2.3.6, we know that there is tampering algorithm Tam_ε that changes $\overline{\mathbf{x}}$ to $\overline{\mathbf{y}} \equiv \langle \overline{\mathbf{x}} \parallel \mathsf{Tam}_\varepsilon^{f_2,\mathsf{OnSam}} \rangle \mathbf{S}$ such that

$$\mathbb{E}[f_2(\overline{\mathbf{y}})] \geq \mu_2 + \frac{p \cdot k}{2m} \cdot \nu - \varepsilon.$$

By a similar averaging argument we can conclude the proof. □

2.4 Proof of Theorem 2.3.2

In the following subsections, we focus on proving Theorem 2.3.2. We first prove a version of theorem for computationally unbounded adversaries and then will extend it to computationally bounded setting. At the end we prove Lemma 2.3.6.

2.4.1 Proving Theorem 2.3.2 for Polynomially Bounded Attacks

In this section, we prove the second item of Theorem 2.3.2. Namely, we show an efficient tampering algorithm whose average is ε-close to the average of RejSam. We define this attack as follows:

Construction 2.4.1 (k-rejection-sampling tampering)**.** Let $\overline{\mathbf{x}} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ be a joint distribution and $f\colon \mathrm{Supp}(\overline{\mathbf{x}}) \mapsto [0,1]$. The *$k$-rejection sampling* tampering algorithm RejSam_k^f works as follows. Given the valid prefix $y_{\leq i-1} \in \mathrm{ValPref}(\overline{\mathbf{x}})$, the tampering algorithm would do the following for k times:

1. Sample $y_{\geq i} \leftarrow (\mathbf{x}_{\geq i} \mid y_{\leq i-1})$ by using the online sampler for f.
2. Let $s = f(y_1, \ldots, y_n)$; with probability s output y_i, otherwise go to Step 1.

If no y_i was output during any of the above k iterations then output a fresh sample $y_i \leftarrow (\mathbf{x}_i \mid y_{\leq i-1})$.

The output distribution of RejSam_k on any input, converges to the rejections sampling tampering algorithm RejSam for sufficiently large $k \to \infty$.

Notation. Below, use the notation $\overline{\mathbf{z}} = \langle \overline{\mathbf{x}} \parallel \mathsf{RejSam}_k^f \rangle S$ and $\mu_k = \mathbb{E}[f(\overline{\mathbf{z}})]$.

We will prove the following claim which will directly completes the proof of second part of Theorem 2.3.2.

Claim 2.4.2. *Let $\overline{\mathbf{x}} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ be a joint distribution and $f\colon \mathrm{Supp}(\overline{\mathbf{x}}) \mapsto [0,1]$. For any $\varepsilon \in [0,1]$, let $k \geq \frac{16 \ln(2n/\varepsilon)}{\varepsilon^2 \mu^2}$. Then RejSam_k runs in time $O(k) = \mathrm{poly}(N/(\varepsilon \cdot \mu))$, where $N \geq n$ is the total bit-length of representing $\overline{\mathbf{x}}$, and for $\overline{\mathbf{z}} \equiv \langle \overline{\mathbf{x}} \parallel \mathsf{RejSam}_k^{f,\mathsf{OnSam}} \rangle \mathbf{S}$ it holds that*

$$\mathbb{E}[f(\overline{\mathbf{z}})] \geq \mu^{-p} \cdot \mathbb{E}[f(\overline{\mathbf{x}})^{1+p}] - \varepsilon \ .$$

Proof. It is easy to see why RejSam_k runs in time $O(k)$ and thus we will focus on proving the expected value of the output of the k-rejection sampling tampering algorithm. To that end, we start by providing some definitions relevant to our analysis.

Definition 2.4.3. For $\delta \geq 0$, let

$$\mathrm{High}(\delta) = \left\{ \overline{x} \mid \overline{x} \in \mathrm{Supp}(\overline{\mathbf{x}}) \wedge \forall i \in [n], \hat{f}[x_{\leq i-1}] \geq \delta \right\}, \ \mathrm{Low}(\delta) = \mathrm{Supp}(\overline{\mathbf{x}}) \setminus \mathrm{High}(\delta) \ ,$$

$$\mathrm{Big}(\delta) = \{ \overline{x} \mid \overline{x} \in \mathrm{Supp}(\overline{\mathbf{x}}) \wedge f(\overline{x}) \geq \delta \}, \ \text{and} \ \ \mathrm{Small}(\delta) = \mathrm{Supp}(\overline{\mathbf{x}}) \setminus \mathrm{Big}(\delta) \ .$$

Claim 2.4.4. *For $\delta_1 \cdot \delta_2 = \delta$, it holds that*

$$\Pr_{\overline{x} \leftarrow \overline{\mathbf{x}}}[\overline{x} \in \mathrm{Big}(\delta_1) \mid \overline{x} \in \mathrm{Low}(\delta)] \leq \delta_2 \ .$$

As a result, it holds that $\Pr_{\overline{x}\leftarrow\overline{\mathbf{x}}}[\overline{x} \in \mathrm{Big}(\delta_1) \wedge \overline{x} \in \mathrm{Low}(\delta)] \leq \delta_2$*, and so*

$$\sum_{\overline{x}\in \mathrm{Big}(\delta_1)\cap \mathrm{Low}(\delta)} \Pr[\overline{x} = \overline{\mathbf{x}}] \leq \delta_2 \ .$$

Proof. Let $t\colon \mathrm{Low}(\delta) \to \mathrm{ValPref}(\overline{\mathbf{x}})$ be such that $t(\overline{x})$ is the smallest prefix $x_{\leq i}$ such that $\hat{f}[x_{\leq i}] \leq \delta$. Now consider the set $T = \{t(\overline{x}) \mid \overline{x} \in \mathrm{Low}(\delta)\}$. For any $w \in T$ we have

$$\delta \geq \hat{f}[w] \geq \Pr_{\overline{x}\leftarrow\overline{\mathbf{x}}}[\overline{x} \in \mathrm{Big}(\delta_1) \mid t(\overline{x}) = w] \cdot \delta_1 \ ,$$

which implies

$$\Pr_{\overline{x}\leftarrow\overline{\mathbf{x}}}[\overline{x} \in \mathrm{Big}(\delta_1) \mid t(\overline{x}) = w] \leq \delta_2 \ .$$

Thus, we have

$$\begin{aligned}
&\Pr_{\overline{x}\leftarrow\overline{\mathbf{x}}}[\overline{x} \in \mathrm{Big}(\delta_1) \mid \overline{x} \in \mathrm{Low}(\delta)] \\
&= \sum_{w\in T} \Pr_{\overline{x}\leftarrow\overline{\mathbf{x}}}[\overline{x} \in \mathrm{Big}(\delta_1) \wedge t(\overline{x}) = w \mid \overline{x} \in \mathrm{Low}(\delta)] \\
&= \sum_{w\in T} \Pr_{\overline{x}\leftarrow\overline{\mathbf{x}}}[\overline{x} \in \mathrm{Big}(\delta_1) \mid \overline{x} \in \mathrm{Low}(\delta) \wedge t(\overline{x}) = w] \cdot \Pr_{\overline{x}\leftarrow\overline{\mathbf{x}}}[t(\overline{x}) = w \mid \overline{x} \in \mathrm{Low}(\delta)] \\
&\leq \sum_{w\in T} \delta_2 \cdot \Pr_{\overline{x}\leftarrow\overline{\mathbf{x}}}[t(\overline{x}) = w \mid \overline{x} \in \mathrm{Low}(\delta)] \leq \delta_2 \ .
\end{aligned}$$

□

Claim 2.4.5. *Let* $x \in \mathrm{High}(\delta)$*, then we have*

$$\Pr[\overline{\mathbf{z}} = \overline{y}] \geq (1 - (1-\delta)^k)^n \cdot \frac{f(\overline{y})}{\mu} \cdot \Pr[\overline{\mathbf{x}} = \overline{y}] \ .$$

Proof. Consider $\mathrm{E}_{k,y_{\leq i}}$ to be the event that RejSam_k outputs one of its first k samples, when performed on $y_{\leq i}$. Then, it holds that

$$\Pr[\mathrm{E}_{k,y_{\leq i}}] = 1 - (1 - \hat{f}[y_{\leq i}])^k \geq 1 - (1-\delta)^k \ .$$

On the other hand, we know that $\Pr[\overline{\mathbf{z}}_{i+1} = y_{i+1} \mid y_{\le i} \wedge \mathrm{E}_{k,y_{\le i}}] = \Pr[\overline{\mathbf{y}}_{i+1} = y_{i+1} \mid y_{\le i}]$. Thus, we have

$$\begin{aligned}\Pr[\overline{\mathbf{z}}_{i+1} = y_{i+1} \mid y_{\le i}] &\ge \Pr[\overline{\mathbf{z}}_{i+1} = y_{i+1} \mid y_{\le i} \wedge \mathrm{E}_{k,y_{\le i}}] \cdot \Pr[\mathrm{E}_{k,y_{\le i}}] \\ &= \Pr[\overline{\mathbf{y}}_{i+1} = y_{i+1} \mid y_{\le i}] \cdot \Pr[\mathrm{E}_{k,y_{\le i}}] \\ &\ge \Pr[\overline{\mathbf{y}}_{i+1} = y_{i+1} \mid y_{\le i}] \cdot (1-(1-\delta)^k)^n.\end{aligned}$$

By multiplying these inequalities for $i \in [n]$ we get $\Pr[\overline{\mathbf{z}} = \overline{y}] \ge (1-(1-\delta)^k)^n \cdot \Pr[\overline{\mathbf{y}} = \overline{x}]$. □

Claim 2.4.6. *For $\delta_1 \cdot \delta_2 = \delta$, it holds that*

$$\mu_k \ge \sum_{\overline{y} \in \mathrm{Supp}(\overline{\mathbf{x}})} \left(\frac{f(\overline{y})}{\mu}\right)^p \cdot \Pr[\overline{\mathbf{x}} = \overline{y}] \cdot f(\overline{y}) - \frac{\delta_1 + \delta_2}{\mu} - n \cdot (1-\delta)^k \ .$$

Proof. Let

$$\mu' = \sum_{\overline{y} \in \mathrm{Low}(\delta) \cap \mathrm{Small}(\delta_1)} \left(\frac{f(\overline{y})}{\mu}\right)^p \cdot \Pr[\overline{\mathbf{x}} = \overline{y}] \cdot f(\overline{y}) \ ,$$

$$\text{and} \quad \mu'' = \sum_{\overline{y} \in \mathrm{Low}(\delta) \cap \mathrm{Big}(\delta_1)} \left(\frac{f(\overline{y})}{\mu}\right)^p \cdot \Pr[\overline{\mathbf{x}} = \overline{y}] \cdot f(\overline{y}) \ .$$

By Claim, 2.3.11 we have

$$\begin{aligned}\mathbb{E}[f(\overline{\mathbf{z}})] &\ge \sum_{\overline{y} \in \mathrm{Supp}(\overline{\mathbf{x}})} \left(\frac{\Pr[\overline{\mathbf{z}}^{[n]} = \overline{y}]}{\Pr[\overline{\mathbf{x}} = \overline{y}]}\right)^p \cdot \Pr[\overline{\mathbf{x}} = \overline{y}] \cdot f(\overline{y}) \\ &\ge \sum_{\overline{y} \in \mathrm{High}(\delta)} \left(\frac{\Pr[\overline{\mathbf{z}}^{[n]} = \overline{y}]}{\Pr[\overline{\mathbf{x}} = \overline{y}]}\right)^p \cdot \Pr[\overline{\mathbf{x}} = \overline{y}] \cdot f(\overline{y}) \\ \text{(by Claim 2.4.5)} \quad &\ge \sum_{\overline{y} \in \mathrm{High}(\delta)} (1-(1-\delta)^k)^{n \cdot p} \cdot \left(\frac{f(\overline{y})}{\mu}\right)^p \cdot \Pr[\overline{\mathbf{x}} = \overline{y}] \cdot f(\overline{y}) \\ &= (1-(1-\delta)^k)^{n \cdot p} \cdot \left(\sum_{\overline{y} \in \mathrm{Supp}(\overline{\mathbf{x}})} \left(\frac{f(\overline{y})}{\mu}\right)^p \cdot \Pr[\overline{\mathbf{x}} = \overline{y}] \cdot f(\overline{y}) - \mu' - \mu''\right).\end{aligned}$$

We have $\mu' \le \delta_1^{1+p}/\mu^p \le \delta_1/\mu$, because $f(\overline{y}) \le \delta_1$ for all $\overline{y} \in \mathrm{Small}(\delta_1)$. Also, by Claim 2.4.4, we get

$$\mu'' \le \sum_{\overline{y} \in \mathrm{Low}(\delta) \cap \mathrm{Big}(\delta_1)} \left(\frac{1}{\mu}\right)^p \cdot \Pr[\overline{\mathbf{x}} = \overline{y}] \le \frac{\delta_2}{\mu^p} \le \frac{\delta_2}{\mu} \ .$$

Therefore, we have

$$\begin{aligned}
\mathbb{E}[f(\overline{\mathbf{z}})] &\geq (1-(1-\delta)^k)^{n\cdot p}\cdot\left(\sum_{\overline{y}\in\mathrm{Supp}(\overline{\mathbf{x}})}\left(\frac{f(\overline{y})}{\mu}\right)^p\cdot\Pr[\overline{\mathbf{x}}=\overline{y}]\cdot f(\overline{y})-\frac{\delta_1+\delta_2}{\mu}\right)\\
\text{(by Bernoulli inequality)}\quad &\geq (1-n\cdot(1-\delta)^k)\cdot\left(\sum_{\overline{y}\in\mathrm{Supp}(\overline{\mathbf{x}})}\left(\frac{f(\overline{y})}{\mu}\right)^p\cdot\Pr[\overline{\mathbf{x}}=\overline{y}]\cdot f(\overline{y})-\frac{\delta_1+\delta_2}{\mu}\right)\\
&\geq \sum_{\overline{y}\in\mathrm{Supp}(\overline{\mathbf{x}})}\left(\frac{f(\overline{y})}{\mu}\right)^p\cdot\Pr[\overline{\mathbf{x}}=\overline{y}]\cdot f(\overline{y})-\frac{\delta_1+\delta_2}{\mu}-n\cdot(1-\delta)^k.
\end{aligned}$$

□

In order to conclude the proof of Claim 2.4.2, we can set $\delta_1=\delta_2=\sqrt{\delta}$ and let $\delta\leq(\varepsilon\mu/4)^2$. Then, given that we have $k\geq\frac{16\ln(2n/\varepsilon)}{\varepsilon^2\mu^2}$, we get

$$\mathbb{E}[f(\overline{\mathbf{z}})]\geq\sum_{\overline{y}\in\mathrm{Supp}(\overline{\mathbf{x}})}\left(\frac{f(\overline{y})}{\mu}\right)^p\cdot\Pr[\overline{\mathbf{x}}=\overline{y}]\cdot f(\overline{y})-\frac{\varepsilon}{2}-\frac{\varepsilon}{2}\,.$$

□

2.4.2 Relating the Bias to the Variance: Proving Lemma 2.3.6

We use Lemma 2.2.2 by letting $\varphi(x)=x^{1+p}$. Thus, we have to minimize the following function on $x\in[0,1]$,

$$g_\mu(x)=\left(x^{1+p}-\mu^{1+p}-(1+p)\cdot\mu^p\cdot(x-\mu)\right)/(x-\mu)^2\,.$$

We now prove that the minimum happens on $x=1$. Note that the function $g_\mu(x)$ is continues on $[0,\mu)$ and $(\mu,0]$ and the limit exists at $x=\mu$ and is equal to $1/2\cdot p\cdot(1+p)\cdot\mu^{-1+p}$. Therefore if we show that g'_μ is negative for $x\in[0,\mu)\cup(\mu,1]$ it implies that, $\forall x\in[0,1]g(x)\geq g(1)$. We have

$$\begin{aligned}
g'_\mu(x)&=\frac{(p-1)\cdot\mu^{p+1}-(p+1)\cdot x\cdot\mu^p+(p+1)\cdot\mu\cdot x^p-(p-1)\cdot x^{p+1}}{(\mu-x)^3}\\
\text{(using } c=x/\mu)\quad &=\mu^{p-2}\cdot\frac{(p-1)-(p+1)\cdot c+(p+1)\cdot c^p-(p-1)\cdot c^{p+1}}{(1-c)^3}\,.
\end{aligned}$$

We prove that the numerator $q(c) = (p-1) - (p+1)\cdot c + (p+1)\cdot c^p - (p-1)\cdot c^{p+1}$ is positive for $c > 1$ and negative for $0 < c < 1$. For $c > 0$, we have

$$\begin{aligned} q'(c) &= -(1+p) + (p+1)\cdot p\cdot c^{p-1} + (1-p)\cdot(p+1)\cdot c^p \\ &= (1+p)\cdot(p\cdot c^{p-1} + (1-p)\cdot c^p - 1) \\ \text{(by AM-GM inequality of Lemma 2.2.1)} &\geq (1+p)\cdot(c^{p\cdot(p-1)}\cdot c^{(1-p)\cdot p} - 1) \\ &= 0\ . \end{aligned}$$

Therefore, q is increasing for $c > 0$ which implies $\forall c \in [0,1], q(c) < q(1) = 0$ and $\forall c > 1, q(c) > q(1) = 0$. We have $\forall x \in [0,\mu)\cup(\mu,1], g'(x) \leq 0$. Therefore we have

$$\forall x \in [0,1], g_u(x) \geq g_u(1)\ . \tag{2.1}$$

Now we prove that $g_\mu(1) \geq \frac{p(1+p)}{2}$. Consider the following function,

$$w(\mu) = g_\mu(1) = \left(1 - \mu^{1+p} - (1+p)\cdot\mu^p\cdot(1-\mu)\right)/(1-\mu)^2\ .$$

We will show that q is a decreasing function for $\mu \in [0,1]$. We have

$$w'(\mu) = \frac{p\cdot(1-\mu^2)\cdot\mu^{p-1} + p^2(1-\mu)^2\cdot\mu^{p-1} + 2\cdot(\mu^p - 1))}{(-1+\mu)^3}\ .$$

We will show that the numerator $s(\mu) = p\cdot(1-\mu^2)\cdot\mu^{p-1} + p^2(1-\mu)^2\cdot\mu^{p-1} + 2\cdot(\mu^p - 1)$ is negative for $\mu \in [0,1]$. We have $s'(\mu) = p(p^2-1)\cdot(1-\mu)^2\cdot\mu^{p-2}$ which is negative for $\mu \in [0,1]$. This implies that $\forall\mu \in [0,1], s(\mu) \geq s(1) = 0$. Therefore, w is a decreasing function, and we obtain

$$\forall\mu \in [0,1], g_\mu(1) = w(\mu) \geq \lim_{u\to 1} w(u) = \frac{p(1+p)}{2}\ . \tag{2.2}$$

Now, we conclude that

$$\begin{aligned} \mu^{-p}\cdot\mathbb{E}[\boldsymbol{\alpha}^{1+p}] - \mu &= \mu^{-p}\left(\mathbb{E}[\boldsymbol{\alpha}^{1+p}] - \mu^{1+p}\right) \\ \text{(by Lemma 2.2.2)} \quad &\geq \mu^{-p}\left(\inf_{x\in[0,1]}\{g_\mu(x)\}\cdot\nu\right) \\ \text{(by Inequality 2.1)} \quad &\geq \mu^{-p}\cdot g_\mu(1)\cdot\nu \\ \text{(by Inequality 2.2)} \quad &\geq \frac{p\cdot(1+p)}{2\cdot\mu^p}\cdot\nu\ . \end{aligned}$$

Chapter 3

Strong Adaptive Poisoning Attacks

3.1 Poisoning Attacks from Concentration of Product Measures

In this section, we design new poisoning attacks against any deterministic learning algorithm, by using the concentration of space in the domain of training data. We start by defining the confidence and error parameters of learners.

3.1.1 Definition of Confidence and Chosen-Instance Error

Definition 3.1.1 (Probably approximately correct learning)**.** An algorithm L is an $(\varepsilon(\cdot), \delta(\cdot))$-PAC learner for a classification problem $(\mathcal{X}, \mathcal{Y}, D, \mathcal{H}, \mathcal{C})$, if for all $c \in \mathcal{C}$ and $m \in \mathbb{N}$, we have

$$\Pr_{\substack{\mathcal{T} \leftarrow (D, c(D))^m \\ h \leftarrow L(\mathcal{T})}} [\mathsf{Risk}(h, c) > \varepsilon(m)] \leq \delta(m).$$

The function $\varepsilon(\cdot)$ is the error parameter, and $1 - \delta(m)$ is the confidence of the learner L.

Now, we formally define the class of poisoning attacks and their properties.

Definition 3.1.2 (Strong Adaptive poisoning attacks)**.** Let $(\mathcal{X}, \mathcal{Y}, D, \mathcal{H}, \mathcal{C})$ be a classification with a learning algorithm L. Then, a poisoning adversary A for $(L, \mathcal{X}, \mathcal{Y}, D, \mathcal{H}, \mathcal{C})$ is an algorithm that takes as input a training set $\mathcal{T} \leftarrow (D, c(D))^m$ and outputs a modified training set $\mathcal{T}' = A(\mathcal{T})$ of the same size[1]. We also interpret $\mathcal{T}$ and $\mathcal{T}$ as vectors with m coordinates with a large alphabet and let HD be the Hamming distance for such vectors of m coordinates. For any $c \in \mathcal{C}$, we define the following properties for A.

- A is called *plausible* (with respect to c), if $y = c(x)$ for all $(x, y) \in \mathcal{T}'$.

[1] Requiring the sets to be equal only makes our *negative* attacks *stronger*.

- A has *tampering budget* $b \in [m]$ if for all $\mathcal{T} \leftarrow (D, c(D))^m, \mathcal{T}' \leftarrow A(\mathcal{T})$, we have

$$\mathsf{HD}(\mathcal{T}', \mathcal{T}) \leq b.$$

- A has *average* tampering budget b, if we have:

$$\mathop{\mathbb{E}}_{\substack{\mathcal{T} \leftarrow (D, c(D))^m \\ \mathcal{T}' \leftarrow A(\mathcal{T})}} [\mathsf{HD}(\mathcal{T}', \mathcal{T}))] \leq b.$$

Before proving our results about the power of poisoning attacks, we need to define the confidence function of a learning algorithm under such attacks.

Definition 3.1.3 (Confidence function and its adversarial variant)**.** For a learning algorithm L for a classification problem $(\mathcal{X}, \mathcal{Y}, D, \mathcal{H}, \mathcal{C})$, we use Conf_A to define the *adversarial confidence* in the presence of a poisoning adversary A. Namely,

$$\mathsf{Conf}_A(m, c, \varepsilon) = \Pr_{\substack{\mathcal{T} \leftarrow (D, c(D))^m \\ h \leftarrow L(A(\mathcal{T}))}} [\mathsf{Risk}(h, c) \leq \varepsilon].$$

By $\mathsf{Conf}(\cdot)$, we denote L's *confidence function* without any attack; namely, $\mathsf{Conf}(\cdot) = \mathsf{Conf}_I(\cdot)$ for the trivial (identity) attacker I that does not change the training data.

Definition 3.1.4 (Chosen instance (average) error and its adversarial variant)**.** For a classification problem $(\mathcal{X}, \mathcal{Y}, D, \mathcal{H}, \mathcal{C})$, and a learning algorithm L, a chosen instance $x \in \mathcal{X}$, a concept $c \in \mathcal{C}$ and for some $m \in \mathbb{N}$, the *chosen-instance* error of x in presence of a poisoning adversary A is

$$\mathsf{Err}_A(m, c, x) = \Pr_{\substack{\mathcal{T} \leftarrow (D, c(D))^m \\ h \leftarrow L(A(\mathcal{T}))}} [h(x) \neq c(x)].$$

The *chosen-instance* error for x (without attacks) is then defined as $\mathsf{Err}(m, c, x) = \mathsf{Err}_I(m, c, x)$ using the trivial adversary that outputs its input.

3.1.2 Decreasing Confidence and Increasing Chosen-Instance Error through Poisoning

The following theorem formalizes our poisoning attack. We emphasize that by choosing the adversary *after* the concept function is fixed, we allow the adversary to depend on the concept class. This is also the case in e.g., p-tampering poisoning attacks of Mahloujifar et al. [2018a] (describesd in Section 1 of this part).

However, there is a big distinction between our attacks here and those of Mahloujifar et al. [2018a], as our attackers need to know the *entire* training sequence before tampering with them, while the attacks of Mahloujifar et al. [2018a] were online.

Theorem 3.1.5. *For any classification problem $(\mathcal{X}, \mathcal{Y}, D, \mathcal{H}, \mathcal{C})$, let L be a deterministic learner, $c \in \mathcal{C}$ and $\varepsilon \in [0,1]$. Also let $\mathsf{Conf}(m, \varepsilon, c) = 1 - \delta$ be the original confidence of L for error probability ε.*

1. *For any $\gamma \in [0,1]$, there is a plausible poisoning adversary A with tampering budget at most $\sqrt{-\ln(\delta \cdot \gamma) \cdot m}$ such that, A makes the adversarial confidence to be as small as γ:*

$$\mathsf{Conf}_A(\varepsilon, c, m) \leq \gamma.$$

2. *There is a plausible poisoning adversary A with* average *tampering budget $\sqrt{-\ln(\delta) \cdot m/2}$ eliminating all the confidence:*

$$\mathsf{Conf}_A(\varepsilon, c, m) = 0.$$

Before proving Theorem 3.1.5, we introduce a notation.

Notation. For $\overline{x} = (x_1, \ldots, x_m) \in \mathcal{X}^m$ we use $(\overline{x}, c(\overline{x}))$ to denote $\big((x_1, c(x_1)), \ldots, (x_m, c(x_m))\big)$.

Proof of Theorem 3.1.5. We first prove Part 1. Let $\mathcal{F} = \{\overline{x} \in \mathcal{X}^m \mid L((\overline{x}, c(\overline{x})) = h, \mathsf{Risk}(h, c) > \varepsilon\}$, and let $\mathcal{F}_b$ be the b expansion of $\mathcal{F}$ under Hamming distance inside $\mathcal{X}^m$.

We now define an adversary A that fulfills the statement of Part 1 of Theorem 3.1.5. Given a training set $\mathcal{T} = (\overline{x}, c(\overline{x}))$, the adversary A does the following.

Case 1: If $\overline{x} \in \mathcal{F}_b$, it selects an arbitrary $\overline{x}' \in \mathcal{F}$ where $\mathsf{HD}(\overline{x}, \overline{x}') \leq b$ and outputs $\mathcal{T}' = (\overline{x}', c(\overline{x}'))$.

Case 2: If $\mathcal{T} \notin \mathcal{F}_b$, it does nothing and outputs $\mathcal{T}$.

By definition, A is using tampering budget at most b, as its output is always in a Hamming ball of radius b centered at $\overline{x}$. In addition, A is a plausible attacker, as it always uses correct labels.

We now show that A decreases the confidence as stated. Note that by the definition of Conf we have $\mathsf{Conf}(\varepsilon, c, m) = 1 - D^{(m)}(\mathcal{F})$ where $D^{(m)}$ is the product distribution measuring according to m independently samples from D. By Lemma 2.2.5, we have $D^{(m)}(\mathcal{F}_b) \geq 1 - \frac{e^{-b^2/m}}{D^{(m)}(\mathcal{F})}$ which by letting $b = \sqrt{-\ln(\delta \cdot \gamma) \cdot m}$ implies that

$$D^{(m)}(\mathcal{F}_b) \geq 1 - \gamma. \tag{3.1}$$

We also know that if A goes to Case 1, it always selects some $\overline{x}' \in \mathcal{F}$, and that means that the generated hypothesis using A's output will have a Risk greater than or equal to ε. Also, if A goes to Case 2 then it will output the original training set which means the generated hypothesis will have a Risk less than ε. Therefore, we have

$$\mathsf{Conf}_A(\varepsilon, c, m) = 1 - \Pr_{\overline{x} \leftarrow D^{(m)}}[\text{Case 1}] = 1 - D^{(m)}(\mathcal{F}_b) \leq \gamma.$$

Before proving Part 2, we state the following claim, which we prove using McDiarmid Inequality.

Claim 3.1.6. *For any product distribution $\boldsymbol{\lambda} = \boldsymbol{\lambda}_1 \times \cdots \times \boldsymbol{\lambda}_m$ where $(\mathrm{Supp}(\boldsymbol{\lambda}), \mathsf{HD}, \boldsymbol{\lambda})$ is a nice metric probability space and any set $\mathcal{S} \subseteq \mathrm{Supp}(\boldsymbol{\lambda})$ where $\boldsymbol{\lambda}(\mathcal{S}) = \varepsilon$, we have*

$$\mathop{\mathbb{E}}_{\overline{x} \leftarrow \boldsymbol{\lambda}}[\mathsf{HD}(\overline{x}, \mathcal{S})] \leq \sqrt{\frac{-\ln(\varepsilon) \cdot m}{2}}.$$

Proof of Claim 3.1.6. We define function $f(\overline{x}) = \mathsf{HD}(\overline{x}, \mathcal{S})$. Because $(\mathrm{Supp}(\boldsymbol{\lambda}), \mathsf{HD}, \boldsymbol{\lambda})$ is a nice metric probability space by assumption, f is a measurable function. Moreover, it is easy to see that for every pair $(\overline{x}, \overline{x}')$ we have $|f(\overline{x}) - f(\overline{x}')| \leq \mathsf{HD}(\overline{x}, \overline{x}')$ (i.e., f is Lipschitz). Now if we let $a = \mathbb{E}_{\overline{x} \leftarrow \boldsymbol{\lambda}}[f(\overline{x})]$, by using Lemma 2.2.6, we get

$$\varepsilon = \boldsymbol{\lambda}(\mathcal{S}) = \Pr_{\overline{x} \leftarrow \boldsymbol{\lambda}}[f(\overline{x}) = 0] = \Pr_{\overline{x} \leftarrow \boldsymbol{\lambda}}[f(\overline{x}) \leq 0] \leq \mathrm{e}^{-2a^2/m}$$

simply because for all $\overline{x} \in \mathcal{S}$ we have $f(\overline{x}) = 0$. Thus, we get $a \leq \sqrt{-\ln(\varepsilon) \cdot m/2}$. □

Now we prove Part 2. We define an adversary A that fulfills the statement of the second part of the theorem. Given a training set $\mathcal{T} = (\overline{x}, c(\overline{x}))$ the adversary selects some $\overline{x}' \in \mathcal{F}$ such that $\mathsf{HD}(\overline{x}, \overline{x}') = \mathsf{HD}(\overline{x}, \mathcal{F})$ (i.e., one of the closest points in $\mathcal{F}$ under Hamming distance). The adversary then outputs $\mathcal{T}' = (\overline{x}', c(\overline{x}'))$. It is again clear that this attack is plausible, as the tampered instances are still within the support set of the correct distribution. Also, it is the case that $\mathsf{Conf}_A(\varepsilon, c, m) = 0$, as the adversary always selects $\overline{x}' \in \mathcal{F}$. To bound the average budget of A we use Claim 3.1.6. By the description of A, we know that the average number of changes that A makes to $\overline{x}$ is equal to $\mathbb{E}_{\overline{x} \leftarrow D^{(m)}}[\mathsf{HD}(\overline{x}, \mathcal{F})]$ which, by Claim 3.1.6, is bounded by $\sqrt{-\ln(\varepsilon) \cdot m/2}$. □

Remark 3.1.7 (Attacks for any undesired predicate)**.** As should be clear from the proof of Theorem 3.1.5, this proof directly extends to any setting in which the adversary wants to increase the probability of any "bad" event B defined over the hypothesis h, if h is produced deterministically based on the training set $\mathcal{T}$. More generally, if the learning rule is not deterministic, we can still increase the probability of any bad event B if B is defined directly over the training data $\mathcal{T}$. This way, we can increase the probability of bad predicate B, where B is defined over the *distribution* of the hypotheses.

We now state our results about the power of poisoning attacks that increase the *average* of the error probability of learners. Our attacks, in this case, need to know the final text instance x, which makes our attacks *targeted* poisoning attacks Barreno et al. [2006].

Theorem 3.1.8. *For any classification problem $(\mathcal{X}, \mathcal{Y}, D, \mathcal{H}, \mathcal{C})$, let L be a deterministic learner, $x \in \mathcal{X}$, $c \in \mathcal{C}$, and let $\varepsilon = \mathsf{Err}(m, c, x)$ be the chosen-instance error of x without any attack.*

1. *For any $\gamma \in (0, 1]$, there is a plausible poisoning adversary A with budget $\sqrt{-\ln(\varepsilon \cdot \gamma) \cdot m}$ such that*

$$\mathsf{Err}_A(m, c, x) \geq 1 - \gamma.$$

2. *There is a plausible poisoning adversary A with average budget $\sqrt{-\ln(\varepsilon) \cdot m}$ such that*

$$\mathsf{Err}_A(m, c, x) = 1.$$

Proof of Theorem 3.1.8. The proof is very similar to the proof of Theorem 3.1.5. We only have to change the description of $\mathcal{F}$ as

$$\mathcal{F} = \{\overline{x} \in \mathcal{X}^m \mid h = L(\overline{x}, c(\overline{x})), h(x) \neq c(x)\},$$

and then everything directly extends to the new setting. □

First now remark on the power of poisoning attacks of Theorems 3.1.5 and 3.1.8.

Remark 3.1.9 (Asymptotic power of our poisoning attacks)**.** We note that, in Theorem 3.1.5 as long as the initial confidence is $1 - 1/\operatorname{poly}(n)$, an adversary can decrease it to $1/\operatorname{poly}(n)$ (or to 0, in the average-budget case) using only tampering budget $\widetilde{O}(\sqrt{n})$. Furthermore, if the initial confidence is at most $1 - \exp(-o(n))$ (i.e., subexponentially far from 1) it can be made subexponentially small $\exp(-o(n))$ (or even 0, in the average-budget case) using only a sublinear $o(n)$ tampering budget. The same remark holds for Theorem 3.1.8 and average error. Namely, if the initial average error for a test example is $1/\operatorname{poly}(n)$, an adversary can decrease increase it to $1 - 1/\operatorname{poly}(n)$ (or to 1, in the average-budget case) using only tampering budget $\widetilde{O}(\sqrt{n})$, and if the initial average error is at least $\exp(-o(n))$ (i.e., subexponentially large), it can be made subexponentially close to one: $1 - \exp(-o(n))$ (or even 1, in the average-budget case) using only a sublinear $o(n)$ tampering budget. The damage to average error is even more devastating, as typical PAC learning arguments usually do not give anything more than a $1/\operatorname{poly}(n)$ error.

Chapter 4

Poisoning Attacks against Privacy

4.1 Introduction

Collaborative ML. Machine learning is revolutionizing nearly every discipline from healthcare to finance to manufacturing and marketing. However, one of the limiting factors in ML is availability of large quantities of quality data. Smaller entities may not be able to collect enough data to build reliable models, and even larger entities may have subpopulations for which they do not have enough data to get accurate results. This has prompted calls for collaborative learning, where many parties combine datasets to train a joint model.

However, much of this data involves either private data about individuals or confidential enterprise information. Moreover, the folk-lore belief that ML models are sufficiently complex that it is hard to extract information about the training data, is being increasingly challenged by recent research on unintended memorization and leakage attacks on ML models Carlini et al. [2018], He et al. [2019]. This prompts the following question:

If I as a company/hospital allow a model to be trained on my (confidential) data and released, what am I revealing about my data?

Note that there are two concerns here: information revealed during the training process, and information revealed by the resulting model. The former can be addressed using a number of cryptographic techniques including Secure Multi-Party Computation Yao [1986], Goldreich et al. [1987], trusted hardware (e.g. Intel SGX), and Homomorphic Encryption Gentry [2009], while the latter is an issue regardless of the techniques used, even if one has a trusted party to perform the training.

Information leakage from ML models There has been a series of work looking at to what extent a model leaks information about a certain *individual record* in the training set, including work on using

differential privacy Dwork et al. [2006] to define what it means for a training algorithm to preserve privacy of these individuals and technically how that can be achieved.

However, leaking information on individuals is not the only concern in this context. In many cases even the aggregate information is sensitive. For example, consider an email classification system which identifies spam or phishing mail. Such a model would be expected to reveal which words commonly indicate spam, and companies might be comfortable sharing this information. However, if this model also leaked information about how often a particular word or set of words appeared in the training data, that could potentially be very sensitive. Similarly, a financial company might be willing to share a model to detect fraud, but might not be willing to reveal the volume of various types of transactions. Or a number of smaller companies might be willing to share a model to help target customers for price reductions etc, however such companies might not be willing to share specific sales numbers for different types of products.

This inspires a line of work started in Ateniese et al. [2015], Ganju et al. [2018a] that looks at *property inference* attacks, in which the attacker is trying to learn aggregate information about a dataset. In particular, we focus here, as did Ganju et al. [2018a], on an attacker who is trying to determine the frequency of a particular feature in the dataset. Notice that this type of aggregate leakage is a global property of the training dataset and is not mitigated by adding differential-privacy mechanisms.

Leakage from combining datasets One of the questions that is not addressed in any of the above works is:

Does releasing a model trained by combining my dataset with other parties' data leak more than one trained only on my own data

In particular, we consider the possibility that one party may modify their data before submitting it in order to learn more from the final model. Note that this is not prevented by any of the cryptographic or hardware assisted solutions: in all of these there is no practical way to guarantee that the data that is entered is actually correct.

This type of *training data poisoning* attack has been extensively studied in the context of security of ML models, i.e., where the goal of the attacker is to train the model to miss-classify certain datapoints Nelson et al. [2008], Mei and Zhu [2015], Jagielski et al. [2018], but to the best of our knowledge ours is the first work that looks at poisoning attacks that aim to compromise privacy of the training data.

Black box or white box model access Finally, one might ask:

Can I prevent this leakage by keeping the model hidden and only allowing other parties to query it?

We show that even in the *black box* model, where the attacker is only allowed to make a limited number of queries to the trained model and learn the results, these attacks can be very successful. "Black box" attacks is sometimes used to refer to attacks which also have access to model's confidence values on each query Sablayrolles et al. [2019]. We emphasize here that we use the stricter notion of black box and our attacker will use only the model predictions.

Theory and experimentation The research in adversarial machine learning has been something of an arms race in recent years, with proposed heuristic defenses followed by new attacks and in turn new defenses. Here we aim to take a more principled approach, where we first present and analyse a theoretical attack which we can show succeeds against any Bayes optimal classifier. Then, we verify this result experimentally by implementing it and testing it against models with different architectures, and show that it does indeed succeed with very high probability. The experiments show that these attacks are real, whereas the theoretical analysis gives intuition for the attacks and hopefully will help to offer some suggestion for where we might look in the future to either strengthen the attack or think about mitigations.

Summary of results In this section we consider an attacker who is allowed to first submit a set of "poisoned" data of its choice, which will be combined with the victim dataset and used to train a model. The attacker can then make a series of black box queries. Finally, the attacker's goal is to determine whether a particular feature is above or below a particular threshold.

1. We first describe a theoretical attack that works as long as the trained model is a Bayes optimal classifier. The rough intuition is that if we can introduce some poisoned data at the training phase in which the label is correlated with the target feature then this will change the Bayes optimal classifier a bit. In particular, the ambiguous cases (those data points which occur in the original training set/underlying distribution equally often with either label) will be slightly more likely to have a particular label when the poisoned data is included in the training set. We can choose these poison points in a way such that this shift is noticeably different for the cases when the target feature occurs with frequency below or above the threshold.

   The key idea is that the *amount* that the classifier changes depends on the frequency of the target feature in the dataset. For example, if high education individuals are rare in the original training dataset, then adding poisoned information about such individuals will shift the optimal predictions for them much more noticeably as compared to a dataset with an abundance of such individuals. We formalize this intuition by giving a concrete attack in this model and analyzing it's effectiveness. Note that our attack is agnostic to the architecture of the trained model since it is completely blackbox.

2. Real model are not exactly Bayes optimal, so there is a question about whether the above result occur in practice. To explore this we implement our attack on various target properties on Census and Enron dataset. The objective of the attacker in all these experiments is to distinguish if there is a higher frequency of the target property or not. Our attack succeeded for various ranges of *higher* vs. *lower* frequencies, e.g. (5% from 10%), (40% from 60%), (30% from 70%). We experiment with three types of target properties:

   (a) Property that is present in the training dataset as a feature, e.g., Gender and Race in Census.

   (b) Property that is not used as a feature in the training data, e.g., Negative emails in Enron.

   (c) Property that is random: for this experiment, we added a random binary feature to both the Census and the Enron data is completely uncorrelated to the classification task.

   Experimenting with these variety of target properties demonstrate the power of our attack. The whitebox attack in Ganju et al. [2018a] used Gender and Race as their target properties on Census dataset, so these experiments demonstrate the effectiveness of our attack and also provide a reference for comparison. In our attacks, we were able to achieve above 90% attack accuracy with about 10% poisoning in all of these experiments without degrading the quality of the trained model. We run our experiments for logistic regression and more complex architectures.

3. Since the work closest to ours is whitebox attack in Ganju et al. [2018a], we experimentally compare its performance our attack. With $\leq 10\%$ poisoning and 50 shadow models we beat the accuracy the white-box attack for the target properties Gender and Race in the Census data that uses 1000 shadow models. For the Random property, the blackbox completely outperforms the whitebox. So our attack improves the performance of the white-box attack both in accuracy and running time, and of course in the access model which is fully black box.

4.2 Related Work

It is quite well known by now that understanding what ML models actually memorize from their training data is not trivial. As discussed above, there is a rich line of work that tries to investigate privacy leakage from ML models under different threat models. Here we provide some more detail on several of the works which seem most related to ours. For a comprehensive survey on the other privacy attacks on neural networks, please see He et al. [2019].

The global property inference attacks of Ateniese et al. [2015], Ganju et al. [2018a] are the most relevant to us: here the adversary's goal is to infer sensitive global properties of the training dataset from the trained

model that the model producer did not intend to share. We have already described some examples above. Yet another example of a global property of a dataset is the following: whether the training data had a higher or lower representation of a particular gender. Property inference attacks were first formulated and studied in Ateniese et al. [2015]. However, this initial approach did not scale well to deep neural networks, so Ganju et al. [2018a] proposed a modified attack that is more efficient. The main differences from our attack are in the threat model: 1) our adversary can poison a portion of the training data and 2) in Ateniese et al. [2015], Ganju et al. [2018a] the adversary has whitebox access to the model meaning that it is given all of the weights in the neural net, while our adversary has only blackbox access to the trained model as described above. We experimentally compare our attack performance and accuracy with that of Ganju et al. [2018a] in Section 4.6.

Another closely related attack is the more recent subpopulation attack Jagielski [2019]. Here the adversary's goal is to poison part of the training data in such a way that only the predictions on inputs coming from a certain subpopulation in the data are impacted. To achieve this, the authors poison the data based on a filter function that specifies the target subpopulation. An example filter function that the authors suggest is the following: in a dataset with race and gender features, an adversary may want to harm the performance specifically for black men. So it will choose data from the underlying distribution where "race" = "black" and "gender" = "male". To poison the training data, the adversary picks some sample data that satisfy the filter function and adds this to the training set with flipped labels. The hope is that if the filter function represents a good enough separation, then the learning algorithm will be able to learn the poisoned function for the targeted subpopulation. While this attack is not targeting any global property leakage, a natural question is whether we can enhance their poisoning strategy with our attack to learn some global property of the target subpopulation. We leave this as future work.

In Melis et al. [2019] the authors studied property leakage in the federated learning framework. In federated learning the process proceeds through multiple rounds. In each round each of the $n > 2$ takes the intermediate model and uses their own data to locally compute an update. These updates are all collected by a central party and used to form the next intermediate model. The threat model in Melis et al. [2019] is the following: $n > 2$ parties participate in a ML training using federated learning where one of the participant is the adversary. The adversary uses the model updates revealed in each round of the federated training and tries to infer properties of the training data that are true of a subpopulation but not of the population as a whole. We note that in this threat model, the adversary gets to see a lot more information than on our model, so this result is not directly comparable to ours.

Finally, we note that our work is similar to Sablayrolles et al. [2019] in spirit, where the authors develop

Bayes optimal strategies for membership inference for *grey box* membership inference attack[1]. The attack the authors study is membership-inference (as opposed to property inference), so the goal and technique of the attack and the assumptions they make about the model are completely different from ours.

4.3 Threat model

Before going through the threat model, we introduce some useful notation.

Notation. We use calligraphic letter (e.g $\mathcal{T}$) to denote sets and capital letters (e.g. D) to denote distributions. We use (X, Y) to denote the joint distribution of two random variables (e.g. the distribution of labeled instances). To indicate the equivalence of two distributions we use $D_1 \equiv D_2$. By $x \leftarrow X$ we denote sampling x from X and by $\Pr_{x \leftarrow X}$ we denote the probability over sampling x from X. We use $\text{Supp}(X)$ to denote the support set of distribution X. For a distribution of labeled instances $D \equiv (X, Y)$ and a predictor $h \colon \text{Supp}(X) \to \text{Supp}(Y)$ we use $\mathsf{Risk}(h, D) = \Pr_{(x,y) \leftarrow D}[h(x) \neq y]$ to denote the average of (0-1) loss over the distribution. We use $p \cdot D_1 + (1-p) \cdot D_2$ to denote the weighted mixture of D_1 and D_2 with weights p and $(1-p)$.

Property Inference: Consider a learning algorithm $L \colon (\mathcal{X} \times \mathcal{Y})^* \to \mathcal{H}$ that maps datasets in $\mathcal{T} \in (\mathcal{X} \times \mathcal{Y})^*$ to a hypothesis class $\mathcal{H}$. Also consider a Boolean property $f \colon \mathcal{X} \to \{0, 1\}$. We consider adversaries who aim at finding information about the statistics of the property f over dataset $\mathcal{T} \in (\mathcal{X} \times \mathcal{Y})^*$, that is used to train a hypothesis $h \in \mathcal{H}$. In particular, the goal of the adversary is to learn information about $\hat{f}(T)$ which is the fraction of data entries in $\mathcal{T}$ that has the property f over data entries. More specifically the adversary tries to distinguish between $\hat{f}(\mathcal{T}) \geq t_0$ or $\hat{f}(\mathcal{T}) \leq t_1$ for some $t_0 > t_1 \in [0, 1]$. We are interested in Black-box setting where the adversary can only query the trained model on several points to see the output label.

Formal Model: To formalize this, we use distributions D_-, D_+ to model the underlying distribution of the dataset for instances with $f(x) = 0$ and $f(x) = 1$ respectively. Then we consider two distributions made by mixing D_-, D_+ at different ratios, i.e.,

$$D_t \equiv t \cdot D_+ + (1-t) \cdot D_-$$

D_t is the distribution where t fraction of the points have $f(x) = 1$. The adversary's goal is to determine the value t for the distribution that has been used to train a model that adversary has access to.

[1]We note that, even though the authors claimed that there results hold for black-box attack, in reality, they addressed grey-box attacks as the adversary in their model also learned the confidence and not just the prediction

For example, the two distributions could be D_{t_0} and D_{t_1} obtained by mixing D_- and D_+ in ratios t_0 and t_1 respectively.

In this attack, as in previous work Ateniese et al. [2015], Ganju et al. [2018a] we assume that the adversary can sample from D_-, D_+, i.e. he knows how data is correlated and distributed except for the target property.

Property Inference with Poisoning: We consider the poisoning model where adversary can contribute pn "poisoned" points to a n-entry dataset T that is used to train the model. To the best of our knowledge, this is the first time that poisoning attacks against privacy of machine learning are modeled and studied.

In order to measure the power of adversary in this model we define the following adversarial game between a challenger C and an adversary A. Our game mimics the classic indistinguishably game style used in cryptographic literature. As described above, L is the learning algorithm, n is the size of the training dataset of which p fraction are poisoned points selected by an adversary. D_-, D_+ are the distributions of elements x with $f(x) = 0, 1$ respectively, and the goal of the attacker is to tell whether the victim dataset contains less than t_0 fraction of points from D_- or more than t_1 fraction of points from D_-. The distinguishing game bellow formalizes this adversarial model.

$\mathsf{PIWP}(L, n, p, D_-, D_+, t_0, t_1)$:

1. C select a bit $b \in \{0, 1\}$ uniformly at random.

   then samples a dataset of size $(1-p) \cdot n$: $\mathcal{T}_{\mathsf{clean}} \leftarrow D_{t_b}^{(1-p)n}$

2. Given all the parameters in the game, A selects a poisoning dataset T_{poison} of size pn and sends it to C.

3. C then trains a model $M \leftarrow L(T_{\mathsf{poison}} \cup T_{\mathsf{clean}})$.

4. A adaptively queries the model on a sequence of points $x_1, \ldots, x_m$ and receives $y_1 = M(x_1), \ldots, y_m = M(x_m)$.

5. A then outputs a bit b' and wins the game if $b = b'$.

We aim to construct an adversary that succeed with probability significantly above 1/2.

Remark 4.3.1. Note that the attacker is assumed to be able to sample from D_- and D_+, the distribution of items with $f(x) = 0$ and with $f(x) = 1$. This assumption is same as what is used in previous property and membership inference attacks against privacy of machine learning. The reason we assume this for the attacker

is because we want to have a worst case analysis of the privacy and we should give the adversary all the oracles that we cannot prevent him to have. This is how security games are usually modeled in cryptography and we follow the same path here. However, removing this assumption and having a success full attack that does not need this knowledge is an interesting open question that we also mention in the Conclusion section.

4.4 Attack against Bayes-Optimal Classifiers

In this section, we will introduce a theoretical attack with provable guarantees for Bayes-optimal classifiers. A Bayes-optimal classifier for a distribution $D \equiv (X, Y)$ is defined to be a classifier that provides the best possible accuracy, given the uncertainty of D. Bellow, we first define the notion of Bayes error and Bayed-optimal classifiers.

Definition 4.4.1 (Bayes Error and Bayes-optimal classifier)**.** Let $D \equiv (X, Y)$ be distribution over $\mathcal{X} \times \mathcal{Y}$. The Bayes error of D is defined to be the optimal error for a deterministic predictor of $\mathcal{Y}$ from $\mathcal{X}$. Namely,

$$\mathsf{Bayes}(D) = \inf_{h\colon \mathcal{X}\to\mathcal{Y}} \Pr_{(x,y)\leftarrow D}[h(x) \neq y].$$

Also the Bayes-optimal classifier for D is defined to be a hypothesis h that minimizes the error over the distribution. Such classifier always exists if $\mathcal{Y}$ is a finite set. In particular, the following function is a Bayes-optimal classifier for any such D

$$\forall x \in \mathcal{X} : h^*_D(x) = \operatorname*{argmax}_{y\in\mathcal{Y}} \Pr[Y = y \mid X = x].$$

The Bayes error is the best error that a classifier can hope for. High performance learning algorithms would try to achieve the error rates close Bayes error by mimicking the behavior of Bayes-optimal classifier. Bellow, we assume a fictional learning algorithm who can learn the bayes-optimal classifier for any distribution, and will show that even such a high quality learning algorithm is susceptible to a certain attack that we describe.

Definition 4.4.2 (Fictional learning algorithm)**.** The fictional learning algorithm L^* is an algorithm that given a dataset $\mathcal{T} \leftarrow D^n$ of size n sampled from an arbitrary distribution $D \equiv (X, Y)$, outputs a model M such that

$$\mathsf{Risk}(M, D) = \mathsf{Bayes}(D).$$

We know that the fictional algorithm L^* cannot exist because of the *No free lunch* Wolpert [1996] theorem. However, for understanding the attack we imagine this learning algorithm exists and try to attack it. After

that we will see that we can attack any learning algorithm that outputs some model "close" to the Bayes optimal classifier, for a class of (not all) distributions.

Theorem 4.4.3. *Let L^* be the fictional learning algorithm of definition 4.4.2. Let $D \equiv (X, Y)$ be a distribution over $\mathcal{X} \times \mathcal{Y}$ and $f : \mathcal{X} \to \{0, 1\}$ be a property over instances. For a any $p, t_1 < t_2 \in [0, 1]$, if*

$$\Pr_{x \leftarrow D_+}[\frac{1}{2} - \frac{p}{2t_2(1-p)} \geq \Pr[Y = 1 \mid X = x] \geq \frac{1}{2} - \frac{p}{2t_1(1-p)}] > 0$$

then for $D_+ = (X, Y \mid f(X) = 1)$ and $D_- = (X, Y \mid f(X) = 0)$, there is an adversary A who wins the security game $\mathsf{PIWP}(n, L^*, D_-, D_+, p, t_1, t_2)$ with probability 1.

4.4.1 Attack Description

In this section we prove Theorem 4.4.3. We first describe an attack and then show how it proves Theorem 4.4.3.

The rough intuition behind the attack is the following. If an adversary can introduce some poisoned data at the training phase to introduce correlation of the target property with the label, then, this will change L^* a bit for the ambiguous cases (i.e., those which occur in the training set equally often with label 0 and 1). The adversary can choose these poison points in a way such that this shift is noticeably different for the cases when the target property occurs with frequency t_1 vs t_2. Thus, this attack exploits the *inherent leakage* in the model to infer information about the target property

Let the original data distribution of clean samples be $D \equiv (X, Y)$. Our adversary A will pick the poison data by i.i.d. sampling from a a distribution $D_A \equiv (X_A, Y_A)$. Note that this adversary is weak in a sense that it does not control the poison set but only controls the distribution from which the poison set is sampled. Such an adversary will change the distribution of instances to a distribution $\tilde{D}$ such that $\tilde{D}$ is a weighted mixture of D and D_A. More precisely,

$$\tilde{D} \equiv \begin{cases} D, & \text{With probability } (1-p), \\ D_A & \text{With probability } p \end{cases}$$

Now we describe the distribution D_A. The adversary first picks a value $m_A \in [0, 1]$ and chooses $X_A \equiv m_A \cdot X_+ + (1 - m_A) \cdot X_-$ where $X_+ = (X \mid f(X) = 1)$ and $X_- = (X \mid f(X) = 0)$. The adversary also picks an adversarial rule $h_a : \mathcal{X} \to \mathcal{Y}$ and sets its poisoning distribution as $D_A = (X_A, h_A(X_A))$. Later we will see how the choice of m_A and h_A would help the adversary to optimize the attack.

Now consider the algorithm L^* and its guarantees for $\tilde{D}$. On a dataset $\tilde{\mathcal{T}} \leftarrow \tilde{D}^n$, the algorithm L^* is guaranteed to output of model $\tilde{M} = L^*(\tilde{T})$ that has the Bayes-optimal error on $\tilde{D}$. Consider joint distribution $(\tilde{X}, \tilde{Y})$ such that $\tilde{D} \equiv (\tilde{X}, \tilde{Y})$. Consider a Bayes-optimal classifier on $\tilde{D}$ is defined as follows:

$$\tilde{h}(x) = \begin{cases} 1 & \text{if } \Pr[\tilde{Y} = 1 \mid \tilde{X} = x] > 0.5, \\ 0 & \text{if } \Pr[\tilde{Y} = 1 \mid \tilde{X} = x] \leq 0.5. \end{cases} \tag{4.1}$$

Because both $\tilde{h}$ and $\tilde{M}$ are Bayes-optimal classifiers, the probability of them disagreeing with each other should be zero. Namely,

$$\Pr[\tilde{M}(X) \neq \tilde{h}(X)] = 0. \tag{4.2}$$

Now we want to calculate $\tilde{h}$ and to see the effect of advesary on the behavior of trained model. Specifically, we calculate $\Pr[\tilde{Y} = 1 \mid \tilde{X} = x]$ to see the effect of poisoning on the classifier. Let E be the event that $\tilde{D}$ is sampled from the poison distribution. We have

$$\begin{aligned} \Pr[\tilde{Y} = 1 \mid \tilde{X} = x] &= \Pr[\tilde{Y} = 1 \mid \tilde{X} = x \mathsf{AND} E] \cdot \Pr[E \mid \tilde{X} = x] \\ &\quad + \Pr[\tilde{Y} = 1 \mid \tilde{X} = x \mathsf{AND} \bar{E}] \cdot \Pr[\bar{E} \mid \tilde{X} = x] \\ &= \Pr[Y_A = 1 \mid X_A = x] \cdot \Pr[E \mid \tilde{X} = x] \\ &\quad + \Pr[Y = 1 \mid X = x] \cdot \Pr[\bar{E} \mid \tilde{X} = x] \\ &= \Pr[h_A(x) = 1 \mid X_A = x] \cdot \Pr[E \mid \tilde{X} = x] \\ &\quad + \Pr[Y = 1 \mid X = x] \cdot \Pr[\bar{E} \mid \tilde{X} = x] \end{aligned} \tag{4.3}$$

Now we should calculate the probability $\Pr[E \mid \tilde{X} = x]$. We have

$$\Pr[E \mid \tilde{X} = x] = \frac{\Pr[\tilde{X} = x \mid E] \cdot \Pr[E]}{\Pr[\tilde{X} = x \mid E] \cdot \Pr[E] + \Pr[\tilde{X} = x \mid \bar{E}] \cdot \Pr[\bar{E}]} \tag{4.4}$$

$$= \frac{\Pr[X_A = x] \cdot p}{\Pr[X_A = x] \cdot p + \Pr[X = x] \cdot (1 - p)} \tag{4.5}$$

On the other hand for all $x \in \mathcal{X}$ such that $f(x) = 1$ we have

$$\Pr[X_A = x] = m_A \cdot \Pr[X_+ = x]. \tag{4.6}$$

and for all $x \in \mathcal{X}$ such that $f(x) = 0$ we have

$$\Pr[X_A = x] = (1 - m_A) \cdot \Pr[X_- = x]. \tag{4.7}$$

Now let $t = \Pr[f(X)]$, for all $x \in \mathcal{X}$ such that $f(x) = 1$ we have

$$\Pr[X = x] = t \cdot \Pr[X_+ = x]. \tag{4.8}$$

and for all $x \in \mathcal{X}$ such that $f(x) = 0$ we have

$$\Pr[X = x] = (1 - t) \cdot \Pr[X_- = x]. \tag{4.9}$$

Now combining Equations 4.5, 4.6, 4.7, 4.8 and 4.9 we for all $x \in \mathcal{X}$ such that $f(x) = 1$ we have

$$\Pr[E \mid \tilde{X} = x] = \frac{m_A \cdot p}{m_A \cdot p + t \cdot (1 - p)} \tag{4.10}$$

and for all $x \in \mathcal{X}$ such that $f(x) = 0$ we have

$$\Pr[E \mid \tilde{X} = x] = \frac{(1 - m_A) \cdot p}{(1 - m_A) \cdot p + (1 - t) \cdot (1 - p)} \tag{4.11}$$

It should be already clear that the behavior of final model would depend on t. The strategy of our adversary is to pick m_A and h_A in a way that maximizes the dependence on t. In next section, we will see how we can implement a concrete attack with this goal, but here just to finish the theorem, we instantiate $m_A = 0$ and $h_A(x) = 1$. Then we have the following claim:

Claim 4.4.4. *If adversary sets $m_A = 1$ and $h_A(x) = 1$ then for any $x \in \mathcal{X}$ such that $f(x) = 1$ we have*

$$\Pr[\tilde{Y} \mid \tilde{X} = x] = \frac{p}{p + t(1 - p)} + \frac{t(1 - p)}{p + t(1 - p)} \cdot \Pr[Y = 1 \mid X = x].$$

Proof. The proof directly follows from combining Equations 4.3 and 4.10. □

Now the following claim shows how the adversary can exploit the dependence of the model on t and infer between t_1 and t_2.

Claim 4.4.5. *If for some $x \in \mathcal{X}$ such that $f(x) = 1$*

$$\frac{1}{2} - \frac{p}{2t_2(1 - p)} \geq \Pr[Y = 1 \mid X = x] \geq \frac{1}{2} - \frac{p}{2t_1(1 - p)}$$

we have

$$\begin{cases} \tilde{h}(x) = 0 & \text{if } t \geq t_2, \\ \tilde{h}(x) = 1 & \text{if } t \leq t_1 \end{cases}$$

Proof. By putting the numbers in Claim 4.4.4 we get if $t > t_2$ then $\Pr[Y = 1 \mid X = x] \leq 0.5$ and if $t < t_1$ then $\Pr[Y = 1 \mid X = x] \geq 0.5$. Therefore, using Equation 4.1 the proof is complete. □

And finally, we show that if the probability of xs that satisfy the condition of Claim 4.4.4 is higher than 0, then the adversary can distinguish t_1 from t_2.

Claim 4.4.6. *If*

$$\Pr_{x \leftarrow X_+}[\frac{1}{2} - \frac{p}{2t_2(1-p)} \geq \Pr[Y = 1 \mid X = x] \geq \frac{1}{2} - \frac{p}{2t_1(1-p)}] > 0$$

then at least for one point $x^* \in \mathcal{X}$, we have

$$\begin{cases} \tilde{M}(x^*) = 0 & \text{if } t \geq t_2, \\ \tilde{M}(x^*) = 1 & \text{if } t \leq t_1. \end{cases}$$

Proof. Using Equation 4.2 we know that the probability of $\tilde{M}$ disagreeing with $\tilde{h}$ is zero. Since we have

$$\Pr_{x \leftarrow X_+}[\frac{1}{2} - \frac{p}{2t_2(1-p)} \geq \Pr[Y = 1 \mid X = x] \geq \frac{1}{2} - \frac{p}{2t_1(1-p)}] > 0$$

there is at least one point x^* that satisfies the conditions in the probability and we have $\tilde{M}(x^*) = \tilde{h}(x^*)$ for both cases of $t \leq t_1$ and $t \geq t_2$. This, together with Claim 4.4.5 implies

$$\begin{cases} \tilde{M}(x^*) = 0 & \text{if } t \geq t_2, \\ \tilde{M}(x^*) = 1 & \text{if } t \leq t_1. \end{cases}$$

□

This finishes the proof of Theorem 4.4.3. This is because the adversary can infer whether $t < t_1$ or $t > t_2$ by only querying the x^*.

Remark 4.4.7. Although we only described the attack for our fictional algorithm, However, note that the assumption behind our attack is not that strong as we only need the learning algorithm to output the Bayes-optimal for *one* (poisoned) distribution. One can try to make the assumption more relax and prove the a similar result for the almost optimal Bayes classifiers. However, we do not take that approach and instead

show the effectiveness of our attack and its idea by designing a concrete attack (next section) and running experiments on real world datasets and simple learning algorithms.

4.5 A concrete attack

Here we describe the concrete attack we use in our experiments. We note that there are many possible variations on this attack; what we have presented here is just one configuration that demonstrates that the attack is feasible with high accuracy. We leave exploration of some of the other variations to future work.

High level approach Recall that the attacker is assumed to be able to sample from D_- and D_+, the distribution of items with $f(x) = 0$ and with $f(x) = 1$. As in the theoretical attack described in Section 4.4, the poisoned data is generated by sampling from D_- and D_+ and introducing correlation between the target feature $f(x)$ and the label y. The fraction of poisoned data that is sampled from each of these distributions, in particular m_A fraction of D_+ and $(1 - m_A)$ fraction of D_-, depend on the fractions t_0 and t_1 that the attacker is trying to distinguish. In particular, we set $m_A = \frac{\pm 1(1/2 - \frac{t_0+t_1}{2})+1}{2}$ of its poison points from D_- and $1 - m_A$ points from D_+. Note that m_A is always either 0 or 1, based on the way we pick its value. This means that the adversary either always picks all the poison points from D_+ or from D_-.

Now let us look at the correlation rule between the label and the target feature. In our concrete attack this correlation rule can either be $h_A(x) = 1$ or $h_A(x) = 0$. Based on our theoretical study, it is in the adversary's best interest to introduce the correlation rule that is different from the correlation rule that is dominant in the victim's training data. But of course, the adversary do not know which rule is dominant in the victim's training dataset. So he does the following: observe the correlation between poison points and their label introduce the opposite correlation in the poison set. For example, if most of the poison points are labeled as 1, pick $h_A(x) = 0$ as the poison rule.

The next challenge here is in choosing the query points. As in Section 4.4, we want to find borderline points, i.e. x's which are roughly equally likely to have label $y = 0$ or $y = 1$. Since we only have sampling access to the distribution, we don't know this probability exactly. Instead we estimate the probability that the label is 0 or 1 by training an ensemble of models and then evaluating all of them on x; if roughly half of the models predict label $y = 0$, we consider it a borderline point. We also include the poisoning points in the points that adversary queries.

In the theoretical analysis, we only need 1 query to determine which of the two distributions was used to train the model: D_{t_0} or D_{t_1}. However, in this concrete attack there is a lot of noise and uncertainty introduced by the above sampling process and by the fact that the model is not necessarily exactly Bayes

optimal. Thus, we find a set of borderline points to use for our queries and combine the responses on all of them to make our prediction.

This raises the question of how to combine these responses. To do this we train another model to determine the best weights to place on the responses. Here we use a shadow model approach: we train a series of "shadow" models at t_0 and t_1, and apply the above poisoning approach using the real poisoned data generated above. Then we query each of the models on each of the chosen query points. Finally, we generate a training set where these responses are the features and the label are 0 or 1 depending on whether the "shadow" model used was from D_{t_0} or D_{t_1}, and train a model on this set. The adversary will guess $\hat{f}(T) = t_0$ or $\hat{f}(T) = t_1$ based on the output of this model on the real response values.

Here we describe the concrete attack for distinguishing $D_{.3}$ from $D_{.7}$ with initial victim training set size n and poisoning rate p (i.e. pn poisoned points out of n total training points).

Choosing poisoned data: Recall that the attacker has sampling access to D_-, D_+, the distribution of samples with $f(x) = 0$ and $f(x) = 1$ respectively. The attack will proceed as follows:

1. Sample pn samples from D_+ if $(t_0 + t_1) < 1$, or pn samples from D_- where $t = (t_0 + t_1) \geq 1$
2. Observe the fraction of samples with label 1 and let l the majority label.
3. Modify each sample in the poison data and set the label equal to $1 - l$
4. Output the resulting pn items as the poisoned set T_{poison}

Choosing the black box queries:

1. Sample a thousand data sets $T_1, \ldots, T_{1000}$ each composed half of elements sampled from D_- and half of elements sampled from D_+.
2. Train $M_1, \ldots, M_{1000}$ using $T_1, \ldots, T_{1000}$.
3. Set $T_q = \emptyset$.
4. While $|T_q| < 1000$ repeat the following two steps:
   - If $|T_q| < 500$ queries, sample $x \leftarrow D_-$, otherwise sample $x \leftarrow D_+$.
   - if
   $$300 \leq \sum_{i=1}^{1000} M_i(x) \leq 700$$
   $T_q = T_q \cup \{x\}$.

5. Set $T_q = T_q \cup T_{\mathsf{poison}}$
6. Output T_q as the set of black box queries to make.

Guessing $\hat{f}(T)$ given responses $R_q = y_1, \ldots y_q$: The attack proceeds as follows:

1. Sample data sets $T_1^{30\%}, \ldots, T_{100}^{30\%}$ with size n from $D_{.30}$ and $T_1^{70\%}, \ldots, T_{100}^{70\%}$ with size n from $D_{.70}$. (Note that the adversary can generate samples from these distributions given sampling access to D_-, D_+: e.g. to sample from $D_{.3}$, first choose bit b from a distribution that is 1 with probability .3, then sample from D_b.)
2. Train $M_1^{30\%}, \ldots, M_{100}^{30\%}$ using $T_1^{30\%} \cup T_{\mathsf{poison}}, \ldots, T_{100}^{30\%} \cup T_{\mathsf{poison}}$ and $M_1^{70\%}, \ldots, M_{100}^{70\%}$ using $T_1^{70\%} \cup T_{\mathsf{poison}}, \ldots, T_{100}^{70\%} \cup T_{\mathsf{poison}}$.
3. Query all models on T_q to get $R_1^{70\%}, \ldots, R_{100}^{70\%}$ and $R_1^{30\%}, \ldots, R_{100}^{30\%}$.
4. Train a linear model on $R_1^{70\%}, \ldots, R_{100}^{70\%}$ and $R_1^{30\%}, \ldots, R_{100}^{30\%}$ with labels 0 and 1 respectively to get M_A with appropriate regularization terms (We use ℓ_2 regulizer with weight $2 \cdot \sqrt{1/m}$ where m is the number of shadow models).
5. Evaluate $M_A(R_q)$ and output the result.

4.6 Experimental Evaluation

Here we evaluate the performance and the accuracy of our attack described in Section 4.5.

4.6.1 Experimental Setup

DataSets We have run our experiments on two datasets:

Census: The primary dataset that we use for our experiments is the US Census Dataset Frank et al. [2011]. The US Census Income Dataset contains census data extracted from the 1994 and 1995 population surveys conducted by the U.S. Census Bureau. This dataset includes 299,285 records with 41 demographic and employment related attributes such as race, gender, education, occupation, marital status and citizenship. The classification task is to predict whether a person earns over $50,000 a year based on the census attributes.

Enron: The second dataset that we use to validate our attack on a completely different dataset is the Enron email dataset Klimt and Yang [2004]. This dataset contains 33717 emails. The classification task here was to classify a mail as spam or not spam based on the words used in the mail. We use 200 words to use as features using tf-idf. The accuracy of spam detection with logistic regression on this dataset is 96%.

We used the census dataset as is, as compared to Ganju et al. [2018a] where they preprocess the census dataset and run their experiments with balanced labels (50% low income and 50% high income). We notice that in the original dataset, the labels are not balanced (around 90% low income and 10% high income) which makes the task much harder.

Target Property In Table 4.1, we summarize the features we experimented with. In all these experiments, the attacker's objective is to distinguish if there is a higher frequency of the target feature or not. Bellow is a summary list of all these properties. Note that in different experiments, different ratios are used to demonstrate the effectiveness of the attack in distinguishing different values and we did not hand pick our experiments.

DataSet	Target Feature	Distinguish between
Census	Random binary	0.05 vs 0.15
Census	Gender	0.6 vs 0.4 female
Census	Race	0.1 vs 0.25 black
Enron	Random binary	0.7 vs 0.3
Enron	Negative sentiment	0.10 vs 0.05

Table 4.1: Target Features

Random binary: We note that to understand the power of this attack on a feature that is completely random and uncorrelated to the classification task (hence, should not be leaked by an ideal model), we did a set of experiments where we added a random binary feature to both the census data and set that as the target feature. Note that this feature is not correlated with any other feature in the dataset and the model should not depend on it to make its decision. This is actually true in experiments and as we will see, the attack of Ganju et al. [2018b] does not perform better than random guess on this property.

Gender: Gender is one of the features in census data with values Male and Female. In our attack, we try to infer whether the dataset has 40% or 60% females included. We picked this property because it could be used as a tool to identify gender bias in a dataset by only looking at the blackbox queries to the trained model.

Race: Another feature that we attack in the census dataset is Gender. In this target property, the adversary tries to infer between two different distribution, one of which has 10% black population and the other one has 25%. Again, we chose this because the race distribution in a population could be a sensitive information in many datasets.

Negative Sentiment: In one of our target properties, we try to infer how many fraction of emails in the enron email dataset have negative sentiment. To do this, we use the sentiment analysis tool in *python nltk* to identify emails with positive and negative sentiment. Note that unlike all the other target properties, the Negative sentiment feature is not part of the dataset which makes it much harder for the attacker.

4.6.2 Black-box queries

As mentioned before, we are interested in the privacy leakage caused by black-box access to the model. Namely, the adversary can query the model on a number of points and infer information about target property using the label prediction of the model on those queries (See Section 4.3 for more details). Our model does not require any other information other than predicted label (e.g. confidence score, derivative of the model, etc). The query points are selected from points that their confidence falls into a certain interval, suggested by our theoretical attack (See claim 4.4.5 and also Section 4.5.) For Enron experiments, we use 500 number of query points and for census data experiments we use 1000 number of query points.

4.6.3 Target model architectures

Most of our experiments use logistic regression as the model architecture for training. The main reason we picked logistic regression was that because it is much faster to train compare to Neural Networks. However, we also have a few experiments on the more complex models. In particular, we test our attack on fully connected neural networks with up to 6 hidden layers. We have to note that since our attack is black-box, we do not need any assumption over the target model architecture other than the fact that it will have high accuracy. This is again in contrast with the previous work of Ganju et al. [2018b] that only work on fully connected neural networks.

4.6.4 Shadow model training

Our shadow model training step is quite simple. As described in Section 4.5, we train a series of shadow models with a fixed poisoning set. Then we use the shadow models to train a simple linear attack model over the predictions on the queries. We use linear models since our theoretical results suggest a simple linear

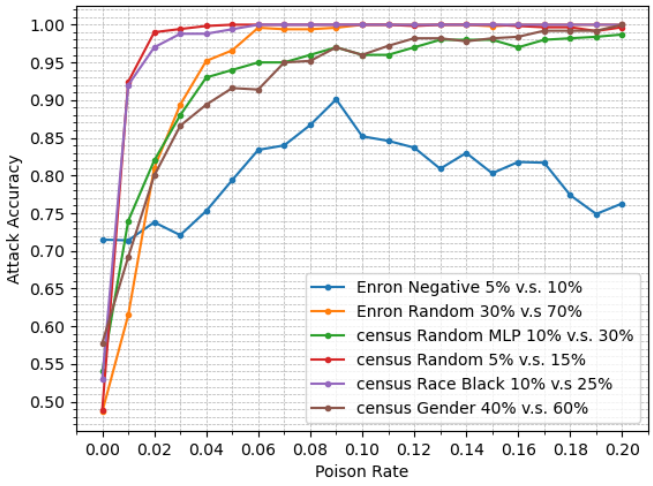

Figure 4.1: Poison rate vs attack accuracy

model over the queries would be able to do the job. We use ℓ_2 and ℓ_1 regularization for our linear models to reduce the number of effective queries as much as possible. Note that this choice of simple linear models is contrast with the attack of Ganju et al. [2018b] that uses complex models (e.g. set neural networks) to train the attack model which runs much slower.

4.6.5 Accuracy Perfomance

In the following experiments, we evaluate the performance of our attack and compare it with the attack of Ganju et al. [2018a]. In the rest of the manuscript, we denote the attack of Ganju et al. [2018a] as WBAttack. We first evaluate how the different parameters, namely, poisoning rate, training set size, number of shadow models (defined in Sec 4.3) and the complexity of the target model affect its accuracy. To understand the effect of each parameter, for each set of experiments, we fix a set of parameters and vary one.

Poisoning Rate In Fig. 4.1, we have 6 experimets where we fix the model to be logistic regression for all of them except one (Census random MLP) which uses a 5 layer perceptron with hidden layers sizes 32, 16 and 8. In all the experiments we set the number of shadow models to be 500 and the training size to be 1000. We vary the poisoning rate from 0% upto 20%. The number of black-box queries is set to 500 for experiments on Enron and 1000 for experiments on Census. The attack accuracy for the other target features are quite low when there is no poisoning as well. But with increase in poisoning rate, the attack accuracy dramatically improves and for all features, the accuracy reaches around 0.9 with just 10% poisoning. All the experiments are repeated 5 times

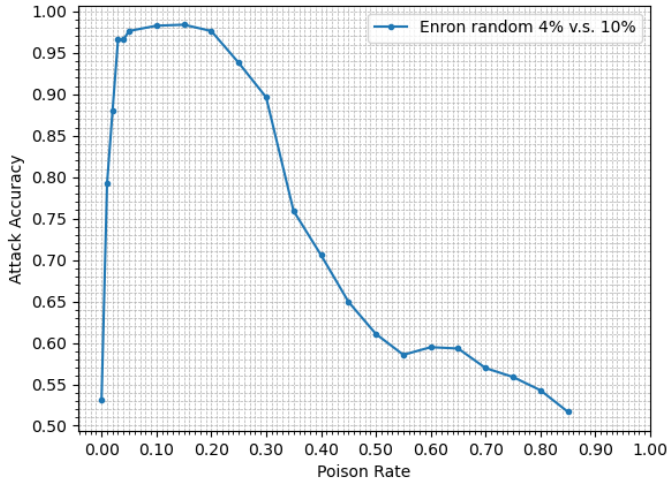

Figure 4.2: Poison rate vs attack accuracy

Note that the Enron negative sentiment experiment seems like an anomaly in Figure 4.1. However, we want to mention that the drop of accuracy with more poison points could be anticipated. To understand this, one can think about the extreme case where 100% of the distribution is the poison data, which means there is no information about the clean distribution in the trained model. This especially happens for the properties that have very weak signal in the behavior of the final model. The Enron negative sentiment property produces the weakest signal among all the experiments because (1) the feature does not exist in the dataset and (2) it has the smallest difference in percentage among all the other experiments (5% vs 10%). In order to verify the understanding that the accuracy of the attack would drop with more poison points, we tried various poisoning rates on Enron dataset with random feature. Figure 4.2 shows this phenomenon.

Number of Shadow Models The next set of experiments (See Fig 4.3) are to see the effect of number of shadow models on the accuracy of the attack. For this experiments, we vary the number of shadow models from 50 to 2000. We notice that increasing the number of shadow models increases the attack accuracy and about 500 shadow models are sufficient to get close to maximum accuracy. Note that in this experiment we set the poisoning ratio to small values so that we can see the trend better. If larger poison ratio were chosen, in most experiments the attack reaches the maximum of 1 with very small number of shadow models and it is hard to see the trend. For instance, with 10% percent poisoning, the experiments with random feature (both census and Enron) would reach 100% accuracy with only 50 number of shadow models. This small number of shadow models makes the running time of the attack much lower.

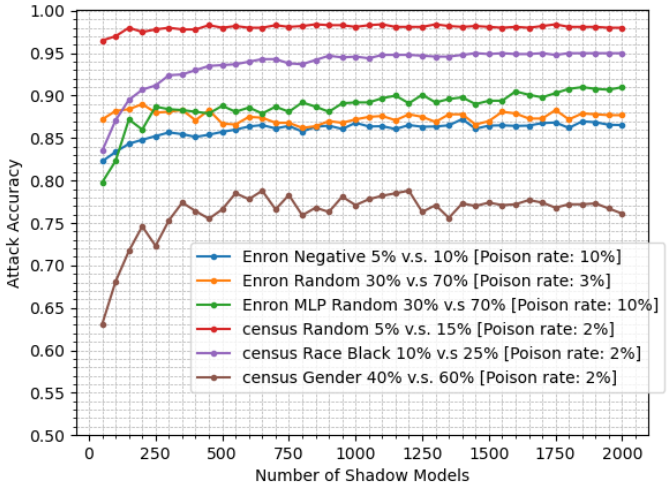

Figure 4.3: Number of shadow models vs attack accuracy

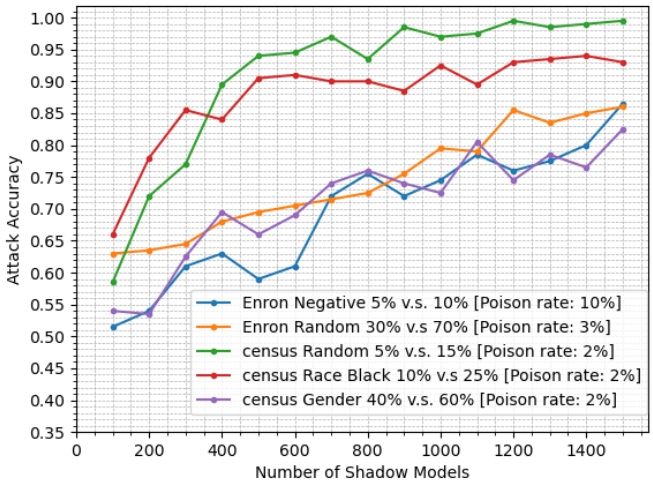

Figure 4.4: Training Set Size vs attack accuracy

Training Set size In Fig. 4.4, we wanted to see the effect of training size on the effectiveness of the attack. Note that our theoretical attack suggests that larger training size should actually improve the attack because the models trained on larger training sets would have smaller generalization error and hence would be closer to a Bayes-optimal classifier. In fact, our experiments verify this theoretical insight. In our experiments, we vary the training set size from 100 to 1500 and the upward trend is quite easy to observe. Again, we have selected the poisoning rate and the number of shadow models in a way that the attack does not get accuracy 1.0 for small training sizes.

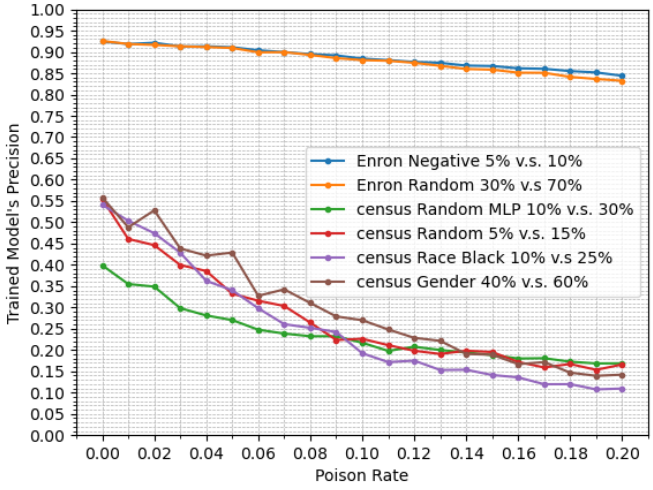

Figure 4.5: Precision

Undetectablilty of the Attack Recall that in our threat model, the adversary is able to poison a fraction of the training data. If the target model quality degrades significantly due to this poisoning, then it becomes easily detectable. Therefore, for the effectiveness of this attack, it is important that the quality of the model does not degrade. We experimentally confirm that this is indeed true with our poisoning strategy. See Fig 4.5 and Fig 4.6 for the precision and recall rate for the model Logistic Regression where the poison rate varies from 0% to 20% (for 500 shadow models and training set size of 1000). In general, the experiments show that the precision tends to decrease with a rather low slope and recall tend to increase by adding more poison data. Note that for census data, the slope of decrease in precision is much higher than Enron. We think this is because of the uncertain nature of Census dataset and the fact that we do not balance the label distribution in Census data. However, it also worth mentioning that for all experiments in Census data, 4-5% poisoning is sufficient to get attack accuracy more than 90%. This means that, if one considers the drop in precision versus the attack accuracy, the census data is not much worse than enron.

Complexity of Target Models While in most of our experiments we fix the target model to be logistic regression, here we experiment with more complex architectures to see how our attack performs. We summarize the results in Table 4.2. Based on our theoretical analysis, the effectiveness of the attack would depend on its performance in generalization to the training data distribution. Therefore, we expect the effectiveness of the attack to drop with more complex networks as the generalization error would increase when the number of parameters in the model increases. This might sound counter intuitive as the privacy problems are usually tied with over fitting and unintended memorizationCarlini et al. [2018]. However, our

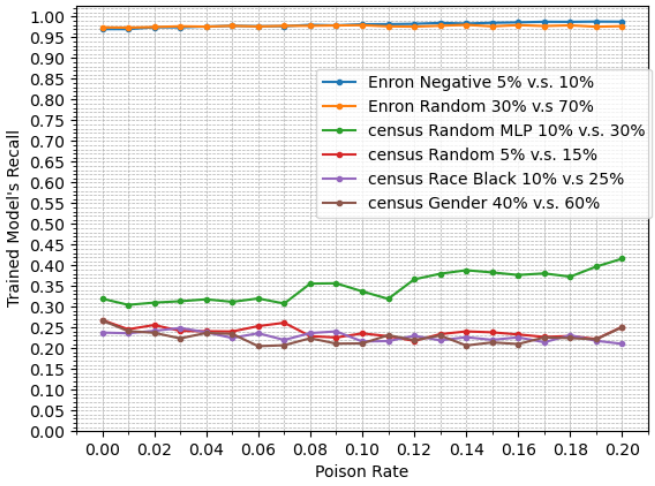

Figure 4.6: Recall

experiments verify our theoretical intuition. We observe that we more layers, the accuracy of the attack tends to drop. However, this issue could be resolved if one uses larger training size as the larger training size could compensate the generalization error caused by higher number of parameters and overfitting. For instance, in the last row of Table 4.2 the accuracy increases significantly when we set the training size to 10000 and use more shadow models.

Architecture			**Performance**	
Hidden Layers	**Layer sizes**	**Training Size**	**Attack Accuracy**	**Shadow Models**
1	[2]	1000	1.0	600
2	[4 2]	1000	0.97	600
3	[8 4 2]	1000	0.94	600
4	[16 8 4 2]	1000	0.88	600
5	[32 16 8 4 2]	1000	0.81	600
5	[32 16 8 4 2]	10000	0.92	1000

Table 4.2: Complexity of target models vs attack accuracy on Census Data

4.6.6 Comparison with WBAttack

Since the work closest to ours is WBAttack, even though it is a white-box attack, we experimentally compare the performance of WBAttack to ours. For this comparison, we run the *vector* attack in WBAttack. In Table 4.3, we see how our *black-box attack performance* compares with WBAttack. Notice that blac-kbox with no poisoning (first 3 rows of the table) performs much worse that WBAttack on race and gender. However, WBAttack also performs poorly on the random feature. In fact, the strength of our attack is to find a way

Experiment parameters	White-box Performance		Black-box Performance		
Feature	**# Shadows**	**Accuracy**	**# Shadows**	**Poison**	**Accuracy**
Census-Random	1000	.52	1000	0	.5
Census-Gender	1000	.96	1000	0	.61
Census-Race	1000	.95	1000	0	.55
Census-Random	1000	.52	100	0.05	1.0
Census-Gender	1000	.96	100	0.03	.99
Census-Race	1000	.95	100	0.05	.97
Census-Random	1000	.52	50	0.1	1.0
Census-Gender	1000	.96	50	0.1	1.0
Census-Race	1000	.95	50	0.1	.98

Table 4.3: Comparison on Logistic Regression. The training size in all experiments is 1000.

to infer information about features similar to random that do not have high correlation with the label. As we see in the columns bellow, with very small ratio of poisoning our attack get accuracy 1.0 on the random target property. It also beats the performance of WBAttack on other experiments by very few number of poison points. Note that our attack also requires much fewer number of poison points. For example with 10% poisoning, only 50 number of shadow models would beat the accuracy WBAttack that uses 1000 number of shadow models. The small number of shadow models would be important in scenarios where the adversary does not have access to a lot of similar data. So in summary, our attack can improve the performance WBAttack both in accuracy and number of shadow models, and of course in the access model which is fully black box. The cost of these improvements is allowing the adversary to choose a fraction of training set which is not an uncommon scenario in multi-party learning applications.

Part II

Inference-time Attacks

Chapter 1

Definitions of Adversarial Risk and Robustness

In recent years, modern machine learning tools (e.g., neural networks) have pushed to new heights the classification results on traditional datasets that are used as testbeds for various machine learning methods.[1] As a result, the properties of these methods have been put into further scrutiny. In particular, studying the *robustness* of the trained models in various adversarial contexts has gained special attention, leading to the active area of *adversarial* machine learning.

Within adversarial machine learning, one particular direction of research that has gained attention in recent years deals with the study of the so-called *adversarial perturbations* of the test instances. This line of work was particularly popularized, in part, by the work of Szegedy et al. [Szegedy et al., 2014] within the context of deep learning classifiers, but the same problem can be asked for general classifiers as well. Briefly, when one is given a particular instance x for classification, an adversarial perturbation x' for that instance is a new instance with minimal changes in the features of x so that the resulting perturbed instance x' is misclassified by the classifier h. The perturbed instance x' is commonly referred to as an *adversarial example* (for the classifier h). Adversarial machine learning has its roots at least as back as in [Lowd and Meek, 2005, Nelson et al., 2010a,b]. However, the work of [Szegedy et al., 2014] revealed pairs of images that differed slightly so that a human eye could not identify any real differences between the two, and yet, contrary to what one would naturally expect, machine learning classifiers would predict different labels for the classifications of such pairs of instances. It is perhaps this striking resemblance to the human eye of the pairs of images that were provided in [Szegedy et al., 2014] that really gave this new push for intense investigations within the

[1] For example, http://rodrigob.github.io/are_we_there_yet/build/ has a summary of state-of-the-art results.

context of adversarial perturbations. Thus, a very intense line of work started, aiming to understand and explain the properties of machine learning classifiers on such adversarial perturbations; e.g., [Goodfellow et al., 2015, Moosavi-Dezfooli et al., 2016, Bastani et al., 2016, Carlini and Wagner, 2017b, Madry et al., 2018]. These attacks are also referred to as *evasion* attacks [Nelson et al., 2012, Biggio et al., 2014, Goodfellow et al., 2015, Carlini and Wagner, 2017b, Xu et al., 2018]. There is also work that aims at making the classifiers more robust under such attacks [Papernot et al., 2016b, Xu et al., 2018], yet newer attacks of Carlini and Wagner [Carlini and Wagner, 2017a] broke many proposed defenses.

As the current literature contains multiple definitions of adversarial risk and robustness, we start by giving a taxonomy for these definitions based on their direct goals. More specifically, suppose x is an original instance that the adversary perturbs into a "close" instance x'. Suppose $h(x), h(x')$ are the predictions of the hypothesis $h(\cdot)$ and $c(x), c(x')$ are the true labels of x, x' defined by the concept function $c(\cdot)$. To call x' a successful "adversarial example", a natural definition would compare the predicted label $h(x')$ with some other "anticipated answer". However, what $h(x')$ is exactly compared to is where various definitions of adversarial examples diverge. We observe in Section 4.2 that the three possible definitions (based on comparing $h(x')$ with either of $h(x)$, $c(x)$ or $c(x')$) lead to three different ways of defining adversarial risk and robustness. We then identify one of them (that compares $h(x)$ with $c(x')$) as the one guaranteeing misclassification by pushing the instances to the *error region*. We also discuss natural conditions under which these definitions coincide. However, these conditions do not hold *in general*.

Defining Adversarial Risk and Robustness

Here we briefly explain the core ideas behind some previous definitions and how they compare with our new definitions. For simplicity we focus on the case of *classification* problems where $c(x)$ is the correct label for an instance $x \in \mathcal{X}$. The adversary aims to perturb the instance x into an instance x' such that x' is close to x under some metric and moreover x' is misclassified by the trained hypothesis h. By *risk* we refer to the probability by which the adversary can get a misclassified x' 'close' to x, and by *robustness* we refer to the minimum change to x that can guarantee x' is misclassified.

Risk and robustness based on the original prediction. The work of [Szegedy et al., 2014] modeled adversary's job as perturbing the instance x into the close instance x' such that $h(x') \neq h(x)$.[2] (See Definition 2.4.1 for a formalization.) Note that, x' would indeed be a misclassification if $c(x') \neq h(x)$, but by substituting this condition with $h(x') \neq h(x)$ one can use optimization methods to find such a close point x' solely based on the parameters of the trained model h. However, this approach is based on two implicit assumptions:

[2] [Szegedy et al., 2014] focuses on the $||\cdot||_2$ norm, and also studies the *targeted* perturbations where the adversary has a specific target label in mind.

(1) that the hypothesis h is correct on all the original untampered examples $x \leftarrow D$; we call this the *initial correctness* assumption, and that (2) the ground truth $c(\cdot)$ does not change under small perturbations of x into x' (i.e., $c(x) = c(x')$); we call this the *truth proximity* assumption. However, if either of these to implicit assumptions do not hold, we cannot use the condition $h(x') \neq h(x)$ as adversary's goal. We refer to this approach for defining risk and robustness as the *prediction-change* (PC) approach.

Risk based on the original truth. The recent work of [Madry et al., 2018] proposed a new way of defining an adversarial loss and risk based on adversary's ability to obtain an x' close to x such that $c(x) \neq h(x')$; namely, here we compare the new prediction to the original actual label. This definition has the advantage that it no longer relies on the implicit assumption of initial correctness for $x \leftarrow D$. However, x' is misclassified only under the truth proximity assumption; that is, one still needs to assume $c(x') = c(x)$. We refer to this approach for defining adversarial risk and robustness as the *original-truth* (OT) approach.

Our approach: error region. In this chapter, we propose defining adversarial risk and robustness (see Definition 2.3.1) directly based on requiring the adversary to push the instances into the error region *regardless* of whether or not the truth proximity or the initial correctness assumptions hold. Namely, our adversary simply aims to perturb x into a close x' such that $h(x') \neq c(x')$. The main motivation for revisiting these notions, is that in broader settings where either of the truth proximity or the initial correctness assumptions do not hold, our new definitions of adversarial risk and robustness are still meaningful while the previous definitions no longer guarantee misclassification of x'. We call this the *error-region* (ER) approach.

1.1 General Definitions of Adversarial Risk and Robustness

Notation. We use calligraphic letters (e.g., $\mathcal{X}$) for sets and capital non-calligraphic letters (e.g., D) for distributions. By $x \leftarrow D$ we denote sampling x from D. In a classification problem $\mathcal{P} = (\mathcal{X}, \mathcal{Y}, \mathcal{D}, \mathcal{C}, \mathcal{H})$, the set $\mathcal{X}$ is the set of possible *instances*, $\mathcal{Y}$ is the set of possible *labels*, $\mathcal{D}$ is a set of distributions over $\mathcal{X}$, $\mathcal{C}$ is a class of *concept* functions, and $\mathcal{H}$ is a class of *hypotheses*, where any $f \in \mathcal{C} \cup \mathcal{H}$ is a mapping from $\mathcal{X}$ to $\mathcal{Y}$. An *example* is a *labeled instance*. We did not state the loss function explicitly, as we work with classification problems, however all main three definitions of this section directly extend to arbitrary loss functions. For $x \in \mathcal{X}, c \in \mathcal{C}, D \in \mathcal{D}$, the *risk* or *error* of a hypothesis $h \in \mathcal{H}$ is the expected (0-1) loss of (h, c) with respect to D, namely $\mathsf{Risk}(h, c, D) = \Pr_{x \leftarrow D}[h(x) \neq c(x)]$. We are usually interested in learning problems with a fixed distribution $\mathcal{D} = \{D\}$, as we are particularly interested in robustness of learning under the uniform distribution U_n over $\{0,1\}^n$. Note that since we deal with negative results, fixing the distribution only makes our results stronger. As a result, whenever $\mathcal{D} = \{D\}$, we omit D from the risk notation and simply write $\mathsf{Risk}(h, c)$. We usually work with problems $\mathcal{P} = (\mathcal{X}, \mathcal{Y}, D, \mathcal{C}, \mathcal{H}, \mathsf{d})$ that include a metric d over the instances.

For a set $\mathcal{S} \subseteq \mathcal{X}$ we let $\mathsf{d}(x, \mathcal{S}) = \inf\{\mathsf{d}(x, y) \mid y \in \mathcal{S}\}$ and $\mathcal{B}all_r(x) = \{x' \mid \mathsf{d}(x, x') \leq r\}$. By HD we denote Hamming distance for pairs of instances from $\{0,1\}^n$. Finally, we use the term *adversarial instance* to refer to an adversarially perturbed instance x' of an originally sampled instance x when the label of the adversarial example is either not known or not considered.

1.1.1 Different Definitions and Qualitative Comparisons

Below we present formal definitions of adversarial risk and robustness. In all of these definitions we will deal with attackers who perturb the initial test instance x into a *close* adversarial instance x'. We will measure how much an adversary can increase the *risk* by perturbing a given input x into a *close* adversarial example x'. These definitions differ in when to call x' a successful adversarial example. First we formalize the main definition that we use in this chapter based on adversary's ability to push instances to the error region.

Definition 1.1.1 (Error-region risk and robustness)**.** Let $\mathcal{P} = (\mathcal{X}, \mathcal{Y}, D, \mathcal{C}, \mathcal{H}, \mathsf{d})$ be a classification problem (with metric d defined over instances $\mathcal{X}$).

- **Risk.** For any $r \in \mathbb{R}_+, h \in \mathcal{H}, c \in \mathcal{C}$, the *error-region risk* under r-perturbation is

$$\mathsf{Risk}_r^{\mathrm{ER}}(h, c) = \Pr_{x \leftarrow D}[\exists x' \in \mathcal{B}all_r(x), h(x') \neq c(x')] .$$

  For $r = 0$, $\mathsf{Risk}_r^{\mathrm{ER}}(h, c) = \mathsf{Risk}(h, c)$ becomes the standard notion of risk.

- **Robustness.** For any $h \in \mathcal{H}, x \in \mathcal{X}, c \in \mathcal{C}$, the *error-region robustness* is the expected distance of a sampled instance to the error region, formally defined as follows

$$\mathsf{Rob}^{\mathrm{ER}}(h, c) = \mathop{\mathbb{E}}_{x \leftarrow D}[\inf\{r \colon \exists x' \in \mathcal{B}all_r(x), h(x') \neq c(x')\}] .$$

Definition 2.3.1 requires the adversarial instance x' to be *misclassified*, namely, $h(x') \neq c(x')$. So, x' clearly belongs to the error region of the hypothesis h compared to the ground truth c. This definition is implicit in the work of Gilmer et al. [2018b]. In what follows, we compare our main definition above with previously proposed definitions of adversarial risk and robustness found in the literature and discuss when they are (or when they are not) equivalent to Definition 2.3.1.

Definitions based on hypothesis's prediction change (PC risk and robustness). Many works, including the works of [Szegedy et al., 2014, Fawzi et al., 2018] use a definition of robustness that compares classifier's prediction $h(x')$ with the prediction $h(x)$ on the original instance x. Namely, they require

$h(x') \neq h(x)$ rather than $h(x') \neq c(x')$ in order to consider x' an adversarial instance. Here we refer to this definition (that does not depend on the ground truth c) as *prediction-change* (PC) risk and robustness (denoted as $\mathsf{Risk}_r^{\mathrm{PC}}(h)$ and $\mathsf{Rob}^{\mathrm{PC}}(h)$). We note that this definition captures the error-region risk and robustness if we *assume* the initial correctness (i.e., $h(x) = c(x)$) of classifier's prediction on all $x \leftarrow X$ and "truth proximity", i.e., that $c(x) = c(x')$ holds for all x' that are "close" to x. Both of these assumptions are valid in some natural scenarios. For example, when input instances consist of images that look similar to humans (if used as the ground truth $c(\cdot)$) and if h is also correct on the original (non-adversarial) test examples, then the two definitions (based on error region or prediction change) coincide. But, these assumptions do not hold in *in general*.

We note that there is also a work in the direction of finding adversarial instances that may potentially fool humans that have limited time to decide for their label, as in [Elsayed et al., 2018]. The images of [Elsayed et al., 2018] are sufficiently 'confusing' that answers of the form "I do not know" are very plausible from the humans that are asked. This fuzzy classification that allows "I do not know" answers is reminiscent of the *limited membership query* model of Sloan and Turán [Sloan and Turán, 1994] (which is a worst-case version of the *incomplete membership query* model of Angluin and Slonim [Angluin and Slonim, 1994]; see also [Angluin et al., 1997b] and [Sloan et al., 2010] for further related discussions) as well as of the model of learning from a *consistently ignorant teacher* of Frazier et al. [Frazier et al., 1996].

Definitions based on the notion of corrupted instance (CI risk and robustness). The works of [Mansour et al., 2015, Feige et al., 2015, 2018, Attias et al., 2018] study the robustness of learning models in the presence of *corrupted inputs*. A more recent framework was developed in [Madry et al., 2018, Schmidt et al., 2018] for modeling risk and robustness that is inspired by robust optimization [Ben-Tal et al., 2009] (with an underlying metric space) and model adversaries that corrupt the the original instance in (exponentially more) ways. When studying adversarial perturbations using corrupted instances, we define adversarial risk by requiring the adversarial instance x' to satisfy $h(x') \neq c(x)$. The term "corrupted instance" is particularly helpful as it emphasizes on the fact that the goal (of the classifier) is to find the *true* label of the *original* (uncorrupted) instance x, while we are only given a corrupted version x'. Hence, we refer to this definition as the *corrupted instance* (CI) risk and robustness and denote them by $\mathsf{Risk}_r^{\mathrm{CI}}(h, c)$ and $\mathsf{Rob}^{\mathrm{CI}}(h, c)$. The advantage of this definition compared to the prediction-change based definitions is that here, we no longer need to assume the initial correctness assumption. Namely, only if the "truth proximity" assumption holds, then we have $c(x) = c(x')$ which together with the condition $h(x') \neq c(x)$ we can conclude that x' is indeed misclassified. However, if small perturbations can change the ground truth, $c(x')$ can be different from $c(x)$, in which case, it is no long clear whether x' is misclassified or not.

Stronger definitions with more restrictions on adversarial instance. The corrupted-input definition requires an adversarial instance x' to satisfy $h(x') \neq c(x)$, and the error-region definition requires $h(x') \neq c(x')$. What if we require *both* of these conditions to call x' a true adversarial instance? This is indeed the definition used in the work of Suggala et al. [Suggala et al., 2018a], though more formally in their work, they subtract the original risk (without adversarial perturbation) from the adversarial risk. This definition is certainly a *stronger* guarantee for the adversarial instance. Therefore, we simply refer to risk and robustness under this condition as *strong* adversarial risk and robustness. As this definition is a hybrid of the error-region and corrupted-instance definitions, we do not make a direct study of this definition and only focus on the other three definitions described above.

How about when the classifier h is 100% correct? We emphasize that when h happens to be the same function as c, (the error region) Definition 2.3.1 implies h has zero *adversarial* risk and infinite adversarial robustness $\mathsf{Rob}^{\mathrm{ER}}(h, c) = \infty$. This is expected, as there is no way an adversary can perturb any input x into a misclassified x'. However, both of the definitions of risk and robustness based on prediction change [Szegedy et al., 2014] and corrupted instance [Mansour et al., 2015, Madry et al., 2018] could compute large risk and small robustness for such h. In fact, in a recent work [Tsipras et al., 2018a] it is shown that for definitions based on corrupted input, correctness might be *provably at odds* with robustness in some cases. Therefore, even though all these definitions could perhaps be used to approximate the risk and robustness when we do not have access to the ground truth c' on the new point x', in this chapter we separate the *definition* of risk and robustness from how to compute/approximate them, so we will use Definition 2.3.1 by default.

1.1.2 Various Aspects of the Attack Models

We emphasize that the definitions of Section 1.1.1 are all *information theoretic* and do not address the *efficiency* of the adversary who perturbs the original instance x into x'. Moreover, there are other aspects of the attack that are implicit in the definitions Section 1.1.1 in terms of what adversary does or does not have access to during the course of the attack. Below, we briefly point out these other aspects.

- **Efficiency.** This aspect of an attack could come in two flavor. One way to mathematically formalize "efficient" attacks is to use polynomial-time attacks as it is done in cryptography. Another way is to use information theoretic attacks without the efficiency requirements. Security against information theoretic attacks are stronger, while attacks of polynomial-time form are stronger.

- **Information access.** The other aspect of the attack is about *what adversary has access to* during the attack and *how it can access* this information. We separate these aspects as follows.

- **What to access.** In general, we can consider attacks that do or do not access to either of the ground truth c, the hypothesis h, or distribution D.
- **How to access.** If the attack can depend on a function f (e.g., $f = h$ or $f = c$) or a distribution D it can still access this information in various forms. An information theoretic attack can completely depend on the full description of f, while an efficient (polynomial time attack) can use oracle access to f (regardless of efficiency of f itself) or a sampler for D. In fact, if f (or a sampler for a distribution D) has a compact representation, then an efficient attacker can also fully depend on f or D if that representation is given.

Going back to the definitions of Section 4.2, by "$\exists x' \in \mathcal{B}all_r(x), P(x')$" we simply state the *existence* of a close instance x' with a property $P(x')$ while it might be computationally infeasible to actually *find* such an x'. Moreover, the definitions of Section 4.2 assume the adversary has full access to f, c, D.

Chapter 2

Evasion Attacks from Concentration of Measure

2.1 Introduction

Learning how to classify instances based on labeled examples is a fundamental task in machine learning. The goal is to find, with high probability, the correct label $c(x)$ of a given test instance x coming from a distribution D. Thus, we would like to find a good-on-average "hypothesis" h (also called the trained model) that minimizes the error probability $\Pr_{x \leftarrow D}[h(x) \neq c(x)]$, which is referred to as the risk of h with respect to the ground truth c. Due to the explosive use of learning algorithms in real-world systems (e.g., using neural networks for image classification) a more modern approach to the classification problem aims at making the learning process, from training till testing, more *robust*. Namely, as we discussed in previous section, even if the instance x is perturbed in a limited way into x' by an adversary A, we would like to have the hypothesis h still predict the right label for x'; hence, minimizing the "adversarial risk"

$$\Pr_{x \leftarrow D}[h(x') \neq c(x') \text{ for some } x' \text{ "close" to } x]$$

of the hypothesis h under such perturbations, where "close" is defined by a metric. An attack to increase the risk is called an "evasion attack" (see e.g., Biggio et al. [2014], Carlini and Wagner [2017b]) due to the fact that x' "evades" the correct classification. One major motivation behind this problem comes from scenarios such as image classification, in which the adversarially perturbed instance x' would still "look similar" to the original x, at least in humans' eyes, even though the classifier h might now misclassify x' Goodfellow et al.

[2018]. In fact, starting with the work of Szegedy et al. Szegedy et al. [2014] an active line of research (e.g., see Biggio et al. [2013, 2014], Goodfellow et al. [2015], Papernot et al. [2016b], Carlini and Wagner [2017b], Xu et al. [2018]) investigated various attacks and possible defenses to resist such attacks. The race between attacks and defenses in this area motivates a study of whether or not such robust classifiers could ever be found, if they exist at all.

The state of affairs in attacks and defenses with regard to the robustness of learning systems in both the evasion contexts leads us to our main question:

> *What are the inherent limitations of defense mechanisms for evasion attacks? Equivalently, what are the inherent power of such attacks?*

Understanding the answer to the above question is fundamental for finding the right bounds that robust learning systems can indeed achieve, and achieving such bounds would be the next natural goal.

Related prior work. In the context of evasion attacks, the most relevant to our main question above are the recent works of Gilmer et al. Gilmer et al. [2018b], Fawzi et al. Fawzi et al. [2018], and Diochnos et al. Diochnos et al. [2018c]. In all of these works, isoperimetric inequalities for specific metric probability spaces (i.e., for uniform distributions over the n-sphere by Gilmer et al. [2018b], for isotropic n-Gaussian by Fawzi et al. [2018], and for uniform distribution over the Boolean hypercube by Diochnos et al. [2018c]) were used to prove that problems on such input spaces are always vulnerable to adversarial instances.[1] The work of Schmidt et al. Schmidt et al. [2018] shows that, at least in some cases, being robust to adversarial instances requires more data. However, the work of Bubeck et al. Bubeck et al. [2018b] proved that *assuming the existence* of classifiers that are robust to evasion attacks, they *could* be found by "few" training examples in an information theoretic way.

2.1.1 Summary of Results

In this section, we draw a connection between the general phenomenon of "concentration of measure" in metric measured spaces and both evasion and poisoning attacks. A concentrated metric probability space $(\mathcal{X}, \mathsf{d}, D)$ with metric d and measure D has the property that for any set $\mathcal{S}$ of measure at least half ($D(\mathcal{S}) \geq 1/2$), most of the points in $\mathcal{X}$ according to D, are "close" to $\mathcal{S}$ according to d (see Definition 2.2.4). We prove that for any learning problem defined over such a concentrated space, no classifier with an initial constant error (e.g., 1/100) can be robust to adversarial perturbations. Namely, we prove the following theorem. (See Theorem 2.3.2 for a formalization.)

[1] More formally, Gilmer et al. Gilmer et al. [2018b] designed specific problems over (two) n-spheres, and proved them to be hard to learn robustly, but their proof extend to any problem defined over the uniform distribution over the n-sphere. Also, Fawzi et al. Fawzi et al. [2018] used a different notion of adversarial risk that only considers the hypothesis h and is independent of the ground truth c, however their proofs also extend to the same setting as ours.

Theorem 2.1.1 (Informal)**.** *Suppose $(\mathcal{X}, \mathsf{d}, D)$ is a concentrated metric probability space from which the test instances are drawn. Then for any classifier h with $\Omega(1)$ initial "error" probability, there is an adversary who changes the test instance x into a "close" one and increases the risk to ≈ 1.*

In Theorem 2.1.1, the "error" could be any undesired event over h, c, x where h is the hypothesis, c is the concept function (i.e., the ground truth) and x is the test instance.

The intuition behind the Theorem 2.1.1 is as follows. Let $\mathcal{E} = \{x \in \mathcal{X} \mid h(x) \neq c(x)\}$ be the "error region" of the hypothesis h with respect to the ground truth concept $c(\cdot)$ on an input space $\mathcal{X}$. Then, by the concentration property of $\mathcal{X}$ and that $D(\mathcal{E}) \geq \Omega(1)$, we can conclude that at least half of the space $\mathcal{X}$ is "close" to $\mathcal{E}$, and by one more application of the same concentration property, we can conclude that indeed most of the points in $\mathcal{X}$ are "close" to the error region $\mathcal{E}$. Thus, an adversary who launches an evasion attack, can indeed push a typical point x into the error region by little perturbations. This above argument, is indeed inspired by the intuition behind the previous results of Gilmer et al. [2018b], Fawzi et al. [2018], and Diochnos et al. [2018c] all of which use isoperimetric inequalities for *specific* metric probability spaces to prove limitations of robust classification under adversarial perturbations. Indeed, one natural way of proving concentration results is to use isoperimetric inequalities that characterize the shape of sets with minimal boundaries (and thus minimal measure after expansion). However, we emphasize that bounds on concentration of measure could be proved even if no such isoperimetric inequalities are known, and e.g., *approximate* versions of such inequalities would also be sufficient. Indeed, in addition to proofs by isoperimetric inequalities, concentration of measure results are proved using tools from various fields such as differential geometry, bounds on eigenvalues of the Laplacian, martingale methods, etc, Milman and Schechtman [1986]. Thus, by proving Theorem 2.1.1, we pave the way for a wide range of results against robust classification for learning problems over *any* concentrated space. To compare, the results of Gilmer et al. [2018b], Fawzi et al. [2018], Diochnos et al. [2018c] have better *constants* due to their use of isoperimetric inequalities, while we achieve similar asymptotic bounds with worse constants but in broader contexts.

Lévy families. A well-studied class of concentrated metric probability spaces are the so-called Lévy families (see Definition 4.3.1) and one special case of such families are known as *normal* Lévy families. In such spaces, when the dimension (seen as the diameter of, or the typical norm of vectors in $(\mathcal{X}, \mathsf{d})$) is n, if we expand sets with measure $1/2$ by distance b, they will cover measure at least $1 - k_1 \mathrm{e}^{-k_2 b^2/n}$ for some universal constants k_1, k_2. When translated back into the context of adversarial classification using our Theorem 2.1.1, we conclude that any learning task defined over a normal Lévy metric space $(\mathcal{X}, \mathsf{d}, D)$ guarantees the existence of (misclassified) adversarial instances that are only $\widetilde{O}(\sqrt{n})$-far from the original instance x, assuming that the original error of the classifier is only polynomially large $\geq 1/\operatorname{poly}(n)$. Interestingly, all the above-mentioned

classifier-independent results on the existence of adversarial instances follow as special cases by applying our Theorem 2.1.1 to known normal Lévy families (i.e., the n-sphere, isotropic n-Gaussian, and the Boolean hypercube under Hamming distance). However, many more examples of normal Lévy families are known in the literature (e.g., the unit cube, the unit sphere, the special orthogonal group, symmetric group under Hamming distance, etc.) for which we immediately obtain new results, and in fact, it seems that "natural" probabilistic metric spaces are more likely to be Lévy families than not! In Section 2.3.2, we list some of these examples and give citation where more examples could be found.[2]

Robustness against *average-case* limited perturbation. We also prove variants of Theorem 2.1.1 that deal with the *average* amount of perturbation done by the adversary with the goal of changing the test instance x into a misclassified x'. Indeed, just like the notion of adversarial risk that, roughly speaking, corresponds to the concentration of metric spaces with a *worst-case* concentration bound, the robustness of a classifier h with an average-case bound on the perturbations corresponds to the concentration of the metric probability space using an average-case bound on the perturbation. In this section we introduce the notion of *target-error* robustness in which the adversary targets a specific error probability and plans its (average-case bounded) perturbations accordingly (see Theorem 2.3.5).

Relation to hardness of robust image classification. Since a big motivation for studying the hardness of classifiers against adversarial perturbations comes from the challenges that have emerged in the area of image classifications, here we comment on possible ideas from our work that might be useful for such studies. Indeed, a natural possible approach is to study whether or not the metric measure space of the images is concentrated or not. We leave such studies for interesting future work. Furthermore, the work of Fawzi et al. [2018] observed that vulnerability to adversarial instances over "nice" distributions (e.g., n-Gaussian in their work, and any concentrated distribution in our work) can *potentially* imply attacks on real data *assuming* that the data is generated with a smooth generative model using the mentioned nice distributions. So, as long as one such mapping could be found for a concentrated space, our impossibility results can potentially be used for deriving similar results about the generated data (in this case image classification) as well.

The special case of product distributions. One natural family of metric probability spaces for which Theorem 2.1.1 entails new impossibility results are *product* measure spaces under Hamming distance. Results of Amir and Milman [1980], Milman and Schechtman [1986], Talagrand [1995] show that such metric probability spaces are indeed normal Lévy. Therefore, we immediately conclude that, in any learning task, if the instances

[2]More formally, in Definition 4.3.1, the concentration function is $e^{-k_2 b^2 \cdot n}$, however in many natural examples that we discuss in Section 2.3.2, the original norm required to be a Lévy family is ≈ 1, while the (expected value of the) "natural" norm is $\approx n$ where n is the dimension. (See Remark 2.3.9.)

come from any product space of dimension n, then an adversary can perturb them to be misclassified by only changing $O(\sqrt{n})$ of the "blocks" of the input. A special case of this result covers the case of Boolean hypercube that was recently studied by Diochnos et al. [2018c]. However, here we obtain impossibilities for *any* product space. As we will see below, concentration in such spaces are useful beyond evasion attacks.

2.2 Preliminaries

2.2.1 Basic Concepts and Notation

Definition 2.2.1 (Notation for metric spaces)**.** Let $(\mathcal{X}, \mathsf{d})$ be a metric space. We use the notation $\mathsf{Diam}^{\mathsf{d}}(\mathcal{X}) = \sup\{\mathsf{d}(x,y) \mid x, y \in \mathcal{X}_i\}$ to denote the diameter of $\mathcal{X}$ under d, and we use $\mathcal{B}all_b^{\mathsf{d}}(x) = \{x' \mid \mathsf{d}(x,x') \leq b\}$ to denote the ball of radius b centered at x. When d is clear from the context, we simply write $\mathsf{Diam}(\mathcal{X})$ and $\mathcal{B}all_b(x)$. For a set $\mathcal{S} \subseteq \mathcal{X}$, by $\mathsf{d}(x,\mathcal{S}) = \inf\{\mathsf{d}(x,y) \mid y \in \mathcal{S}\}$ we denote the distance of a point x from $\mathcal{S}$.

Unless stated otherwise, all integrals in this chapter are Lebesgue integrals.

Definition 2.2.2 (Nice metric probability spaces)**.** We call $(\mathcal{X}, \mathsf{d}, D)$ a metric probability space, if D is a Borel probability measure over $\mathcal{X}$ with respect to the topology defined by d. Then, for a Borel set $\mathcal{E} \subseteq \mathcal{X}$, the *b-expansion* of $\mathcal{E}$, denoted by $\mathcal{E}_b$, is defined as[3]

$$\mathcal{E}_b = \{x \mid \mathsf{d}(x,\mathcal{E}) \leq b\}\,.$$

We call $(\mathcal{X}, \mathsf{d}, D)$ a *nice* metric probability space, if the following conditions hold.

1. **Expansions are measurable.** For every D-measurable (Borel) set $\mathcal{E} \in \mathcal{X}$, and every $b \geq 0$, its b-expansion $\mathcal{E}_b$ is also D-measurable.

2. **Average distances exist.** For every two Borel sets $\mathcal{E}, \mathcal{S} \in \mathcal{X}$, the average minimum distance of an element from $\mathcal{S}$ to $\mathcal{E}$ exists; namely, the integral $\int_{\mathcal{S}} \mathsf{d}(x,\mathcal{E}) \cdot dD(x)$ exists.

At a high level, and as we will see shortly, we need the first condition to define adversarial risk and need the second condition to define (a generalized notion of) robustness. Also, we remark that one can weaken the second condition above based on the first one and still have risk and robustness defined, but since our goal in this chapter is *not* to do a measure theoretic study, we are willing to make simplifying assumptions that hold on the actual applications, if they make the presentation simpler.

[3]The set $\mathcal{E}_b$ is also called the b-flattening or b-enlargement of $\mathcal{E}$, or simply the b-ball around A.

Definition 2.2.3 (Nice classification problems)**.** We call $(\mathcal{X}, \mathcal{Y}, D, \mathcal{C}, \mathcal{H}, \mathsf{d})$ a *nice* classification problem, if the following two conditions hold:

1. $(\mathcal{X}, \mathsf{d}, D)$ is a nice metric probability space.

2. For every $h \in \mathcal{H}, c \in \mathcal{C}$, their error region $\{x \in \mathcal{X} \mid h(x) \neq c(x)\}$ is D-measurable.

The second condition above is satisfied, e.g., if the set of labels $\mathcal{Y}$ (which is usually finite) is countable, and for all $y \in \mathcal{Y}, f \in \mathcal{H} \cup \mathcal{C}$, the set $\{x \in \mathcal{X} \mid f(x) = y\}$ is D-measurable.

2.2.2 The Concentration Function and Some Bounds

We now formally define the (standard) notion of concentration function.

Definition 2.2.4 (Concentration function)**.** Let $(\mathcal{X}, \mathsf{d}, D)$ be a metric probability space and $\mathcal{E} \subseteq \mathcal{X}$ be a Borel set. The *concentration function* is then defined as

$$\boldsymbol{\alpha}(b) = 1 - \inf \{D(\mathcal{E}_b) \mid D(\mathcal{E}) \geq 1/2\}.$$

Variations of the following Lemma 2.2.5 below are in Amir and Milman [1980], Milman and Schechtman [1986], but the following version is due to Talagrand Talagrand [1995] (in particular, see Equation 2.1.3 of Proposition 2.1.1 in Talagrand [1995]).

Lemma 2.2.5 (Concentration of product spaces under Hamming distance)**.** *Let $D \equiv D_1 \times \cdots \times D_n$ be a product probability measure of dimension n and let the metric be the Hamming distance. For any D-measurable $\mathcal{S} \subseteq \mathcal{X}$ such that the b-expansion $\mathcal{S}_b$ of $\mathcal{S}$ under Hamming distance is also measurable,*

$$D(\mathcal{S}_b) \geq 1 - \frac{\mathrm{e}^{-b^2/n}}{D(\mathcal{S})}.$$

Lemma 2.2.6 (McDiarmid inequality)**.** *Let $D \equiv D_1 \times \cdots \times D_n$ be a product probability measure of dimension n, and let $f \colon \mathrm{Supp}(D) \mapsto \mathbb{R}$ be a measurable function such that $|f(x) - f(y)| \leq 1$ whenever x and y* only *differ in one coordinate. If $a = \mathbb{E}_{x \leftarrow D}[f(x)]$, then*

$$\Pr_{x \leftarrow D}[f(x) \leq a - b] \leq \mathrm{e}^{-2 \cdot b^2/n}.$$

2.3 Evasion Attacks: Finding Adversarial Examples from Concentration

In this section, we formally prove our main results about the existence of evasion attacks for learning problems over concentrated spaces. We start by formalizing the notions of risk and robustness.

Definition 2.3.1 (Adversarial risk and robustness)**.** Let $(\mathcal{X}, \mathcal{Y}, D, \mathcal{C}, \mathcal{H}, \mathsf{d})$ be a nice classification problem. For $h \in \mathcal{H}$ and $c \in \mathcal{C}$, let $\mathcal{E} = \{x \in \mathcal{X} \mid h(x) \neq c(x)\}$ be the error region of h with respect to c. Then, we define:

- **Adversarial risk.** For $b \in \mathbb{R}_+$, the (error-region) *adversarial risk* under b-perturbation is

$$\mathsf{Risk}_b(h, c) = \Pr_{x \leftarrow D}[\exists\, x' \in \mathcal{B}all_b(x) \cap \mathcal{E}] = D(\mathcal{E}_b).$$

  We might call b the "budget" of an imaginary "adversary" who perturbs x into x'. Using $b = 0$, we recover the standard notion of risk: $\mathsf{Risk}(h, c) = \mathsf{Risk}_0(h, c) = D(\mathcal{E})$.

- **Target-error robustness.** Given a target error $\rho \in (0, 1]$, we define the ρ-error robustness as the expected perturbation needed to increase the error to ρ; namely,

$$\begin{aligned}\mathsf{Rob}_\rho(h, c) &= \inf_{D(\mathcal{S}) \geq \rho} \left\{ \mathop{\mathbb{E}}_{x \leftarrow D} [\mathbf{1}_{\mathcal{S}}(x) \cdot \mathsf{d}(x, \mathcal{E})] \right\} \\ &= \inf_{D(\mathcal{S}) \geq \rho} \left\{ \int_{\mathcal{S}} \mathsf{d}(x, \mathcal{E}) \cdot dD(x) \right\},\end{aligned}$$

  where $\mathbf{1}_{\mathcal{S}}(x)$ is the characteristic function of membership in $\mathcal{S}$. Letting $\rho = 1$, we recover the notion of full robustness $\mathsf{Rob}(h, c) = \mathsf{Rob}_1(h, c) = \mathbb{E}_{x \leftarrow D}[\mathsf{d}(x, \mathcal{E})]$ that captures the expected amount of perturbations needed to *always* change x into a misclassified x' where $x' \in \mathcal{E}$.

As discussed in the introduction, starting with Szegedy et al. [2014], many papers (e.g., the related work of Fawzi et al. [2018]) use a definitions of risk and robustness that *only* deal with the hypothesis/model and is independent of the concept function. In Diochnos et al. [2018c], that definition is formalized as "prediction change" (PC) adversarial risk and robustness. In Appendix 2.4, we show that using the concentration function $\boldsymbol{\alpha}(\cdot)$ and our proofs of this section, one can also bound the PC risk and robustness of hypotheses assuming that we have a concentration function. Then, by plugging in any concentration function (e.g., those of Lévy families) and obtain the desired upper/lower bounds.

In the rest of this section, we focus on misclassification as a necessary condition for the target adversarial example. So, in the rest of this section, we use Definition 2.3.1 to prove our results.

2.3.1 Increasing Risk and Decreasing Robustness by Adversarial Perturbation

We now formally state and prove our result that the adversarial risk can be large for any learning problem over concentrated spaces. Note that, even though the following is stated using the concentration function, having an *upper bound* on the concentration function suffices for using it. Also, we note that all the results of this section extend to settings in which the "error region" is substituted with any "bad" event modeling an undesired region of instances based on the given hypothesis h and concept function c; though the most natural bad event is that error $h(x) \neq c(x)$ occurs.

Theorem 2.3.2 (From concentration to large adversarial risk)**.** *Let $(\mathcal{X}, \mathcal{Y}, D, \mathcal{C}, \mathcal{H}, \mathsf{d})$ be a nice classification problem. Let $h \in \mathcal{H}$ and $c \in \mathcal{C}$, and let $\varepsilon = \Pr_{x \leftarrow D}[h(x) \neq c(x)]$ be the error of the hypothesis h with respect to the concept c. If $\varepsilon > \boldsymbol{\alpha}(b)$ (i.e., the original error is more than the concentration function for the budget b), then the following two hold.*

1. **Reaching adversarial risk at least half.** *Using only tampering budget b, the adversary can make the adversarial risk to be more than half; namely, $\mathsf{Risk}_b(h, c) > 1/2$.*
2. **Reaching adversarial risk close to one.** *If in addition we have $\gamma \geq \boldsymbol{\alpha}(b_2)$, then the adversarial risk for the total tampering budget $b_1 + b_2$ is $\mathsf{Risk}_{b_1+b_2}(h, c) \geq 1 - \gamma$.*

Proof of Theorem 2.3.2. Let $\mathcal{E} = \{x \in \mathcal{X} \mid h(x) \neq c(x)\}$ be the error region of (h, c), and so it holds that $\varepsilon = D(\mathcal{E})$. To prove Part 1, suppose for sake of contradiction that $\mathsf{Risk}_b(h, c) \leq 1/2$. Then, for $\mathcal{S} = \mathcal{X} \setminus \mathcal{E}_b$, it holds that $D(\mathcal{S}) = 1 - D(\mathcal{E}_b) = 1 - \mathsf{Risk}_b(h, c) \geq 1/2$. By the assumption $D(\mathcal{E}) > \boldsymbol{\alpha}(b)$, we have $D(\mathcal{S}_b) \geq 1 - \boldsymbol{\alpha}(b) > 1 - \varepsilon$. So, there should be $x \in \mathcal{S}_b \cap \mathcal{E}$, which in turn implies that there is a point $y \in \mathcal{S}$ such that $\mathsf{d}(y, x) \leq b$. However, that is a contraction as $\mathsf{d}(y, x) \leq b$ implies that y should be in $\mathcal{E}_b = \mathcal{X} \setminus \mathcal{S}$.

To prove Part 2, we rely on Part 1. By Part 1, if we use a tampering budget b_1, we can increase the adversarial risk to $\mathsf{Risk}_{b_1}(h, c) > 1/2$, but then because of the second assumption $\gamma \geq \boldsymbol{\alpha}(b_2)$, it means that by using b_2 more budget, we can expand the error region to measure $\geq 1 - \gamma$. □

The above theorem provides a general result that applies to *any* concentrated space. So, even though we will compute explicit bounds for spaces such as Lévy families, Theorem 2.3.2 could be applied to any other concentrated space as well, leading to stronger or weaker bounds than what Lévy families offer. Now, in the following, we go after finding general relations between the concentration function and the robustness of the learned models.

Simplifying notation. Suppose $(\mathcal{X}, \mathsf{d}, D)$ is a nice metric probability space. Since our risk and robustness definitions depend only on the error region, for any Borel set $\mathcal{E} \subseteq \mathcal{X}$ and $b \in \mathbb{R}_+$, we define its *b-tampering*

risk as $\mathsf{Risk}_b(\mathcal{E}) = D(\mathcal{E}_b)$, and for any such $\mathcal{E}$ and $\rho \in (0,1]$, we define the ρ-error robustness as $\mathsf{Rob}_\rho(\mathcal{E}) = \inf_{D(\mathcal{S})\geq\rho} \left\{\int_{\mathcal{S}} \mathsf{d}(x,\mathcal{E}) \cdot dD(x)\right\}$.

The following lemma provides a very useful tool for going from adversarial risk to robustness; hence, allowing us to connect concentration of spaces to robustness. In fact, the lemma could be of independent interest, as it states a relation between *worst-case* concentration of metric probability spaces to their *average-case* concentration with a *targeted* amount of measure to cover.

Lemma 2.3.3 (From adversarial risk to target-error robustness)**.** *For a nice metric probability space* $(\mathcal{X}, \mathsf{d}, D)$*, let* $\mathcal{E} \subseteq \mathcal{X}$ *be a Borel set. If* $\rho = \mathsf{Risk}_\ell(\mathcal{E})$*, then we have*

$$\mathsf{Rob}_\rho(\mathcal{E}) = \rho \cdot \ell - \int_{z=0}^{\ell} \mathsf{Risk}_z(\mathcal{E}) \cdot dz.$$

First, we make a few comments on using Lemma 2.3.3.

Special case of full robustness. Lemma 2.3.3 can be used to compute the full robustness also as

$$\mathsf{Rob}(\mathcal{E}) = \mathsf{Rob}_1(\mathcal{E}) = \ell - \int_{z=0}^{\ell} \mathsf{Risk}_z(\mathcal{E}) \cdot dz, \tag{2.1}$$

using any $\ell \geq \mathsf{Diam}(\mathcal{X})$, because for such ℓ we will have $\mathsf{Risk}_\ell(\mathcal{E}) = 1$. In fact, even if the diameter is not finite, we can always use $\ell = \infty$ and rewrite the two terms as

$$\mathsf{Rob}(\mathcal{E}) = \int_{z=0}^{\infty} (1 - \mathsf{Risk}_z(\mathcal{E})) \cdot dz, \tag{2.2}$$

which might or might not converge.

When we only have *lower bounds* for adversarial risk. Lemma 2.3.3, as written, requires the exact amount of risk for the initial set $\mathcal{E}$. Now, suppose we only have a lower bound $L_z(\mathcal{E}) \leq \mathsf{Risk}_z(\mathcal{E})$ for the adversarial risk. In this case, we can still use Lemma 2.3.3, but it will only give us an *upper bound* on the ρ-error robustness using any ℓ such that $\rho \leq L_\ell(\mathcal{E})$ as follows,

$$\mathsf{Rob}_\rho(\mathcal{E}) \leq \ell - \int_{z=0}^{\ell} L_z(\mathcal{E}) \cdot dz. \tag{2.3}$$

Note that, even though the above bound looks similar to that of full robustness in Equation 2.1, in Inequality 2.3 we can use $\ell < \mathsf{Diam}(\mathcal{X})$, which leads to a smaller total bound on the ρ-error robustness.

Proof of Lemma 2.3.3. Let $\nu(\mathcal{S}) = \int_{\mathcal{S}} \mathsf{d}(x, \mathcal{E}) \cdot dD(x)$. Based on the definition of robustness, we have

$$\mathsf{Rob}_\rho(\mathcal{E}) = \inf_{D(\mathcal{S}) \geq \rho} [\nu(\mathcal{S})] .$$

For the fixed $\mathcal{E}$, let $m_{\mathcal{S}} = \sup \{\mathsf{d}(x, \mathcal{E}) : x \in \mathcal{S}\}$, and let $F_{\mathcal{S}} : \mathbb{R} \to \mathbb{R}$ be the cumulative distribution function for $\mathsf{d}(x, \mathcal{E})$ over $\mathcal{S}$, namely $F_{\mathcal{S}}(z) = D(\mathcal{E}_z \cap \mathcal{S})$. Whenever $\mathcal{S}$ is clear from the context we simply write $m = m_{\mathcal{S}}, F(\cdot) = F_{\mathcal{S}}(\cdot)$. Before continuing the proof, we prove the following claim.

Claim 2.3.4. *Let F be a cumulative distribution function of a random variable. For any $m \in \mathbb{R}^+$,*

$$\int_{z=0}^{m} z \cdot dF(z) + \int_{z=0}^{m} F(z) \cdot dz = m \cdot F(m)$$

where the left integral shall be interpreted as Lebesgue integral over the Lebesgue–Stieltjes measure associated with the cumulative distribution function $F(\cdot)$.

Proof of Claim 2.3.4. Claim 2.3.4 follows from the integration-by-parts (extension) for Lebesgue integral over the Lebesgue–Stieltjes measure. □

Now, we have

$$\begin{aligned} \nu(\mathcal{S}) &= \int_{\mathcal{S}} \mathsf{d}(x, \mathcal{E}) \cdot dD(x) = \int_{z=0}^{m} z \cdot dF(z) \\ \text{(by Claim 2.3.4)} \quad &= m \cdot F(m) - \int_{z=0}^{m} F(z) \cdot dz. \end{aligned}$$

Indeed, for the special case of $\mathcal{S} = \mathcal{E}_\ell$ we have $m_{\mathcal{S}} = \ell, F_{\mathcal{S}}(m_{\mathcal{S}}) = F_{\mathcal{S}}(\ell) = D(\mathcal{S}) = \rho$. Thus,

$$\nu(\mathcal{E}_\ell) = m_{\mathcal{E}_\ell} \cdot F_{\mathcal{E}_\ell}(m_{\mathcal{E}_\ell}) - \int_{z=0}^{m_{\mathcal{E}_\ell}} F_{\mathcal{E}_\ell}(z) \cdot dz = \ell \cdot \rho - \int_{z=0}^{\ell} \mathsf{Risk}_z(\mathcal{E}) \cdot dz,$$

and so the robustness can be bounded from above as

$$\mathsf{Rob}_\rho(\mathcal{E}) = \inf_{D(\mathcal{S}) \geq \rho} [\nu(\mathcal{S})] \leq \nu(\mathcal{E}_\ell) = \ell \cdot \rho - \int_{z=0}^{\ell} \mathsf{Risk}_z(\mathcal{E}) \cdot dz. \tag{2.4}$$

We note that, if we wanted to prove Lemma 2.3.3 for the *special* case of *full* robustness (i.e., $\ell \geq \mathsf{Diam}(\mathcal{X}), D(\mathcal{E}_\ell) = \rho = 1$), the above concludes the proof. The rest of the proof, however, is necessary for the more interesting case of target-error robustness. At this point, all we have to prove is a similar *lower* bound

for any $\mathcal{S}$ where $D(\mathcal{S}) \geq \rho$, so in the following assume $\mathcal{S}$ is one such set. By definition, it holds that

$$\forall\, z \in [0, m], F(z) \leq D(\mathcal{E}_z) = \mathsf{Risk}_z(\mathcal{E}) \tag{2.5}$$

and

$$F(m) = D(\mathcal{S}) \geq \rho. \tag{2.6}$$

First, we show that

$$\int_{z=\ell}^{m} (F(z) - F(m)) \cdot dz \leq 0. \tag{2.7}$$

The inequality above clearly holds if $m \geq \ell$. We prove that if $\ell > m$ then the integral is equal to 0. We know that $F(\ell) \leq D(\mathcal{E}_\ell) = \rho$, therefore $F(m) \geq \rho \geq F(\ell)$. We also know that F is an increasing function and $\ell > m$ therefore $F(m) = \rho = F(\ell)$. So we have $\forall z \in [m, \ell], F(z) = \rho$ which implies $\int_{z=\ell}^{m}(F(z) - F(m)) \cdot dz = 0$. Now, we get

$$\begin{aligned}
\nu(\mathcal{S}) &= m \cdot F(m) - \int_{z=0}^{m} F(z) \cdot dz \\
&= \ell \cdot F(m) - \int_{z=0}^{\ell} F(z) \cdot dz - \int_{z=\ell}^{m} (F(z) - F(m)) \cdot dz \\
\text{(by Inequality 2.6)} \quad &\geq \ell \cdot \rho - \int_{z=0}^{\ell} F(z) \cdot dz - \int_{z=\ell}^{m} (F(z) - F(m)) \cdot dz \\
\text{(by Inequality 2.5)} \quad &\geq \ell \cdot \rho - \int_{z=0}^{\ell} \mathsf{Risk}_z(\mathcal{E}) \cdot dz - \int_{z=\ell}^{m} (F(z) - F(m)) \cdot dz \\
\text{(by Inequality 2.7)} \quad &\geq \ell \cdot \rho - \int_{z=0}^{\ell} \mathsf{Risk}_z(\mathcal{E}) \cdot dz.
\end{aligned}$$

The above lower bound on $\mathsf{Rob}_\rho(\mathcal{E})$ and the upper bound of Inequality 2.3 conclude the proof. □

We now formally state our result that concentration in the instance space leads to small robustness of classifiers. Similarly to Theorem 2.3.2, we note that even though the following theorem is stated using the concentration function, having an upper bound on the concentration function would suffice.

Theorem 2.3.5 (From concentration to small robustness)**.** *Let $(\mathcal{X}, \mathcal{Y}, D, \mathcal{C}, \mathcal{H}, \mathsf{d})$ be a nice classification problem. Let $h \in \mathcal{H}$ and $c \in \mathcal{C}$, and let $\varepsilon = \Pr_{x \leftarrow D}[h(x) \neq c(x)]$ be the error of the hypothesis h with respect to the concept c. Then if $\varepsilon > \boldsymbol{\alpha}(b_1)$ and $1 - \rho \geq \boldsymbol{\alpha}(b_2)$, we have*

$$\mathsf{Rob}_\rho(\mathcal{E}) \leq (1 - \varepsilon) \cdot b_1 + \int_{z=0}^{b_2} \boldsymbol{\alpha}(z) \cdot dz.$$

Proof of Theorem 2.3.5. By Theorem 2.3.2, we know that $\mathsf{Risk}_{b_1}(\mathcal{E}) = D(\mathcal{E}_{b_1}) \geq \frac{1}{2}$ which implies $\mathsf{Risk}_{b_1+b_2}(\mathcal{E}) = \mathsf{Risk}_{b_2}(\mathcal{E}_{b_1}) \geq \rho$. If we let $\rho^* = \mathsf{Risk}_{b_1+b_2}(\mathcal{E})$, then we have

$$
\begin{aligned}
& \mathsf{Rob}_{\rho}(\mathcal{E}) \leq \mathsf{Rob}_{\rho^*}(\mathcal{E}) \\
\text{(by Lemma 2.3.3)} \quad & = \int_{z=0}^{b_1+b_2} (\rho^* - \mathsf{Risk}_z(\mathcal{E})) \cdot dz \\
& = \int_{z=0}^{b_1} (\rho^* - \mathsf{Risk}_z(\mathcal{E})) \cdot dz + \int_{b_1}^{b_1+b_2} (\rho^* - \mathsf{Risk}_z(\mathcal{E})) \cdot dz \\
& \leq (\rho^* - \gamma) \cdot b_1 + \int_{b_1}^{b_1+b_2} (\rho^* - \mathsf{Risk}_z(\mathcal{E})) \cdot dz \\
& = (\rho^* - \gamma) \cdot b_1 + \int_{z=0}^{b_2} (\rho^* - \mathsf{Risk}_z(\mathcal{E}_{b_1})) \cdot dz \\
\text{(by Theorem 2.3.2)} \quad & \leq (\rho^* - \gamma) \cdot b_1 + \int_{z=0}^{b_2} (\rho^* - 1 + \boldsymbol{\alpha}(z)) \cdot dz \\
& = (\rho^* - \varepsilon) \cdot b_1 + (\rho^* - 1) \cdot b_2 + \int_{z=0}^{b_2} \boldsymbol{\alpha}(z) \cdot dz \\
& \leq (1 - \varepsilon) \cdot b_1 + \int_{z=0}^{b_2} \boldsymbol{\alpha}(z) \cdot dz.
\end{aligned}
$$

□

2.3.2 Normal Lévy Families as Concentrated Spaces

In this subsection, we study a well-known special case of concentrated spaces called normal Lévy families, as a rich class of concentrated spaces, leading to specific bounds on the risk and robustness of learning problems whose test instances come from any normal Lévy family. We start by formally defining normal Lévy families.

Definition 2.3.6 (Normal Lévy families)**.** A family of metric probability spaces $(\mathcal{X}_n, \mathsf{d}_n, D_n)_{i\in\mathbb{N}}$ with corresponding concentration functions $\boldsymbol{\alpha}_n(\cdot)$ is called a (k_1, k_2)-*normal Lévy family* if

$$\boldsymbol{\alpha}_n(b) \leq k_1 \cdot \mathrm{e}^{-k_2 \cdot b^2 \cdot n}.$$

The following theorem shows that classifying instances that come from a normal Lévy family has the inherent vulnerability to perturbations of size $O(1/\sqrt{n})$

Theorem 2.3.7 (Risk and robustness in normal Lévy families)**.** *Let $(\mathcal{X}_n, \mathcal{Y}_n, D_n, \mathcal{C}_n, \mathcal{H}_n, \mathsf{d}_n)_{n\in\mathbb{N}}$ be a nice classification problem with a metric probability space $(\mathcal{X}_n, \mathsf{d}_n, D_n)_{n\in\mathbb{N}}$ that is a (k_1, k_2)-normal Lévy family. Let $h \in \mathcal{H}_n$ and $c \in \mathcal{C}_n$, and let $\varepsilon = \Pr_{x\leftarrow D}[h(x) \neq c(x)]$ be the error of the hypothesis h with respect to the concept c.*

1. **Reaching adversarial risk at least half.** *If* $b > \sqrt{\ln(k_1/\varepsilon)}/\sqrt{k_2 \cdot n}$, *then* $\mathsf{Risk}_b(h, c) \geq 1/2$.

2. **Reaching Adversarial risk close to one.** *If* $b > \left(\sqrt{\ln(k_1/\varepsilon)} + \sqrt{\ln(k_1/\gamma)}\right)/\sqrt{k_2 \cdot n}$, *then it holds that* $\mathsf{Risk}_b(h, c) \geq 1 - \gamma$.

3. **Bounding target-error robustness.** *For any* $\rho \in [\frac{1}{2}, 1]$, *we have*

$$\mathsf{Rob}_\rho(h, c) \leq \frac{(1-\varepsilon)\sqrt{\ln(k_1/\varepsilon)} + \mathsf{erf}\left(\sqrt{\ln(k_1/(1-\rho))}\right) \cdot k_1\sqrt{\pi}/2}{\sqrt{k_2 \cdot n}}.$$

Proof of Theorem 2.3.7. Proof of Part 1 is similar to (part of the proof of) Part 2, so we focus on Part 2. To prove Part 2, let $b_2 = \sqrt{\frac{\ln(k_1/\gamma)}{k_2 \cdot n}}$ and $b_1 = b - b_2 > \sqrt{\frac{\ln(k_1/\varepsilon)}{k_2 \cdot n}}$. Then, we get $k_1 \cdot \mathrm{e}^{-k_2 \cdot b_2^2 \cdot n} = \gamma$ and $k_1 \cdot \mathrm{e}^{-k_2 \cdot b_1^2 \cdot n} < \varepsilon$. Therefore, by directly using Part 2 of Theorem 2.3.2 and Definition 4.3.1 (of normal Lévy families), we conclude that $\mathsf{Risk}_b(h, c) \geq 1 - \gamma$ for $b = b_1 + b_2$.

We now prove Part 3. By Theorem 2.3.5, we have

$$\mathsf{Rob}_\rho(h, c) \leq (1-\varepsilon) \cdot b_1 + k_1 \cdot \int_0^{b_2} \mathrm{e}^{-k_2 \cdot z^2 \cdot n} \cdot dz = (1-\varepsilon) \cdot b_1 + \frac{k_1 \cdot \sqrt{\pi}}{2\sqrt{n \cdot k_2}} \cdot \mathsf{erf}\left(b_2 \cdot \sqrt{n \cdot k_2}\right).$$

□

Here we remark on its interpretation in an asymptotic sense, and discuss how much initial error is needed to achieve almost full adversarial risk.

Corollary 2.3.8 (Asymptotic risk and robustness in normal Lévy families)**.** *Let* P_n *be a nice classification problem defined over a metric probability space that is a normal Lévy family, and let* ε *be the error probability of a hypothesis* h *with respect to some concept function* c.

1. **Starting from constant error.** *If* $\varepsilon \geq \Omega(1)$, *then for any constant* γ, *one can get adversarial risk* $1 - \gamma$ *for* h *using only* $O(1/\sqrt{n})$ *perturbations, and full robustness of* h *is also* $O(1/\sqrt{n})$.

2. **Starting from sub-exponential error.** *If* $\varepsilon \geq \exp(-o(n))$, *then one can get adversarial risk* $1 - \exp(-o(n))$ *for* h *using only* $o(1)$ *perturbations, and full robustness is also* $o(1)$.

Remark 2.3.9 (How much perturbation is needed? $O(\sqrt{n})$ or $O(1/\sqrt{n})$?)**.** The amount of perturbation in normal Lévy families needed to (almost certainly) misclassify the adversarial example is $O(1/\sqrt{n})$, but this is also the case that "typically" metric probability spaces become normal Lévy under a "normalized" metric; meaning that the diameter (or more generally the average of distances of random pairs) is $\Theta(1)$. (E.g., when working with the unit n-sphere.) However, in some occasions, the "natural" metrics over those spaces is

achieved by scaling up the typical distances to $\Theta(n)$ (e.g., the Hamming distance in the Boolean hypercube). In that case, the bounds of Theorem 2.3.7 also get scaled up to $O(\sqrt{n})$ (for constants ε, γ).

Examples of Normal Lévy Families.

Here, we list some natural metric probability spaces that are known to be normal Lévy families. For the references and more examples we refer the reader to excellent sources Ledoux [2001], Giannopoulos and Milman [2001], Milman and Schechtman [1986]. There are other variants of Lévy families, e.g., those called Lévy (without the adjective "normal") or *concentrated* Lévy families Alon and Milman [1985] with stronger concentration, but we skip them and refer the reader to the cited sources and general tools of Theorems 2.3.2 and 2.3.5 on how to apply *any* concentration of measure results to get bounds on risk and robustness of classifiers.

- **Unit sphere under Euclidean or Geodesic distance**. The unit n-spheres $\mathbb{S}^n$ (of radius 1 in $\mathbb{R}^{n+1}$), under the geodesic distance (or Euclidean distance) and the normalized rotation-independent uniform measure is a normal Lévy family. Lévy was first Lévy [1951] to notice that the isoperimetric inequality for $\mathbb{S}^n$ makes it (what is now known as a) Lévy family.

- $\mathbb{R}^n$ **under Gaussian distribution and Euclidean distance.** $\mathbb{R}^n$ with Euclidean distance and n-dimensional Gaussian measure (where expected Euclidean length is 1) is a normal Lévy family. This follows from the Gaussian isoperimetric inequality Borell [1975], Sudakov and Tsirel'son [1978].

- **Unit cube and unit ball under Euclidean distance.** Both the unit cube $[0,1]^n$ and the unit n-ball (of radius 1) are normal Lévy families under normalized Euclidean distance (where the diameter is 1) and normalized Lebesgue distributions (see Propositions 2.8 and 2.9 in Ledoux [2001]).

- **Special orthogonal group.** The special orthogonal group $\mathrm{SO}(n)$ (i.e., the subgroup of the orthogonal group $\mathrm{O}(n)$ containing matrices with determinant one) equipped with the Hilbert-Schmidt metric and the Haar probability measure is a normal Lévy family.

- **Product distributions under Hamming distance.** Any product distribution D^n with normalized Hamming distance is a normal Lévy family Amir and Milman [1980], Milman and Schechtman [1986], Talagrand [1995]. In particular, the Boolean hypercube $\{0,1\}^n$ with normalized Hamming distance and uniform distribution is a normal Lévy family Amir and Milman [1980].[4] In the next section, we will use the concentration of product spaces to obtain *poisoning* attacks against learners.

[4] This also follows from the isoperimetric inequality of Harper [1966].

- **Symmetric group under Hamming distance.** The set of all permutations Π^n under Hamming distance and the uniform distribution forms a *non-product* Lévy family.

2.4 Risk and Robustness Based on Hypothesis's Prediction Change

As we discussed in previous section, the work of Szegedy et al. Szegedy et al. [2014], as well as a big portion of subsequent work on adversarial examples, relies on defining adversarial risk and robustness of a hypothesis h based on the amount of adversarial perturbations that change the prediction of h. Their definition is independent of the concept function c determining the ground truth. In particular, for a given example $(x, c(x))$ where the prediction of the hypothesis is $h(x)$ (that might indeed be different from $c(x)$), an adversarial perturbation of x is r such that for the instance $x' = x + r$ we have $h(x') \neq h(x)$ (where $h(x')$ may or may not be equal to $c(x')$). Hence, since the attacker only cares about changing the prediction of the hypothesis h, we refer to adversarial properties (be it adversarial perturbations, adversarial risk, adversarial robustness) under this definition as adversarial properties based on "prediction change" (PC for short)– as opposed to adversarial properties based on the "error region" in Definition 2.3.1.

In this section, we show that using the concentration function $\boldsymbol{\alpha}(\cdot)$ and our proofs of Section 2.3, one can also bound the PC risk and robustness of hypotheses assuming that we have a concentration function. Then, one can use any concentration function (e.g., those of Lévy families) and obtain the desired upper/lower bounds, just as how we did so for the the results of Subsection 2.3.2.

Focusing on the hypothesis class. Whenever we consider a classification problem $(\mathcal{X}, \mathcal{Y}, D, \mathcal{H}, \mathsf{d})$ without explicitly denoting the concept class $\mathcal{C}$, we mean that $(\mathcal{X}, \mathcal{Y}, D, \mathcal{C}, \mathcal{H}, \mathsf{d})$ is nice for the trivial set $\mathcal{C}$ of constant functions that output either of $y \in Y$. The reason for this definition is that basically, below we will require some concept class, and all we want is that preimages of specific labels under any h are measurable sets, which is implied if the problem is nice with the simple $\mathcal{C}$ described.

Definition 2.4.1 (Prediction-change adversarial risk and robustness)**.** Let $(\mathcal{X}, \mathcal{Y}, D, \mathcal{H}, \mathsf{d})$ be a nice classification problem. For $h \in \mathcal{H}$, and $\ell \in \mathcal{Y}$, we define $h^\ell = \{x \in \mathcal{X} \mid h(x) = \ell\}$. Then, for any $h \in H$, we define the following.

- **Prediction change (PC) risk.** The *PC risk* under b-perturbation is

$$\mathsf{Risk}_b^{\mathrm{PC}}(h) = \Pr_{x \leftarrow D}\left[\exists\, x' \in \mathcal{B}all_b(x), h(x) \neq h(x')\right].$$

- **Target-label PC risk.** For $\ell \in \mathcal{Y}$ and $b \in \mathbb{R}_+$, the *ℓ-label (PC) risk* under b-perturbation is

$$\mathsf{Risk}_b^\ell(h) = \Pr_{x \leftarrow D}\left[\exists\, x' \in \mathcal{B}all_b(x) \cap h^\ell\right] = D(h_b^\ell).$$

- **PC robustness.** For a given non-constant $h \in \mathcal{H}$, we define the *PC robustness* as the expected perturbation needed to change the labels as follows

$$\mathsf{Rob}^{\mathrm{PC}}(h) = \mathop{\mathbb{E}}_{\substack{x \leftarrow D \\ \ell = h(x)}}\left[\mathsf{d}(x, \mathcal{X} \setminus h^\ell)\right].$$

- **Target-label PC robustness.** For $\ell \in \mathcal{Y}$ and a given non-constant $h \in \mathcal{H}$, we define the *ℓ-label (PC) robustness* as the expected perturbation needed to make the label always ℓ defined as

$$\mathsf{Rob}^\ell(h) = \mathop{\mathbb{E}}_{x \leftarrow D}\left[\mathsf{d}(x, h^\ell)\right].$$

Theorem 2.4.2 (PC risk and robustness in concentrated spaces)**.** *Let $(\mathcal{X}, \mathcal{Y}, D, \mathcal{H}, \mathsf{d})$ be a nice classification problem. For any $h \in \mathcal{H}$ that is* not *a constant function, the following hold.*

1. *Let $\varepsilon \in [0, 1/2]$ be such that $D(h^\ell) \le 1 - \varepsilon$ for all $\ell \in \mathcal{Y}$. If $\boldsymbol{\alpha}(b_1) < \varepsilon/2$ and $\boldsymbol{\alpha}(b_2) \le \gamma/2$, then for $b = b_1 + b_2$ we have*
$$\mathsf{Risk}_b^{\mathrm{PC}}(h) \ge 1 - \gamma.$$

2. *If $\boldsymbol{\alpha}(b_1) < D(h^\ell)$ and $\boldsymbol{\alpha}(b_2) \le \gamma$ then for $b = b_1 + b_2$ we have*
$$\mathsf{Risk}_b^\ell(h) \ge 1 - \gamma.$$

3. *If $\mathsf{Risk}_b^{\mathrm{PC}}(h) \ge \frac{1}{2}$, then*
$$\mathsf{Rob}^{\mathrm{PC}}(h) \le b + \int_0^\infty \boldsymbol{\alpha}(z) \cdot dz.$$

4. *If $\boldsymbol{\alpha}(b) < D(h^\ell)$, then*
$$\mathsf{Rob}^\ell(h) \le b + \int_0^\infty \boldsymbol{\alpha}(z) \cdot dz.$$

Proof. We prove the parts in order.

1. Let $b = b_1 + b_2$. Also, for a set $\mathcal{Z} \subseteq \mathcal{Y}$, let $h^{\mathcal{Z}} = \cup_{\ell \in \mathcal{Z}} h^\ell$. Because for all $\ell \in \mathcal{Y}$ we have $D(h^\ell) \le 1 - \varepsilon$, it can be shown that there is a set $\mathcal{Y}^1 \subset \mathcal{Y}$ such that $D(h^{\mathcal{Y}^1}) \in (\varepsilon/2, 1/2]$. Let $\mathcal{X}^1 = \left\{x \in \mathcal{X} \mid h(x) \in \mathcal{Y}^1\right\}$

and $\mathcal{X}^2 = \mathcal{X} \setminus \mathcal{X}^1$. We know that $D(\mathcal{X}^1) > \varepsilon/2$, so

$$D(\mathcal{X}_b^1) \geq 1 - \gamma/2.$$

On the other hand, we know that $D(\mathcal{X}^2) \geq 1/2$, therefore we have

$$D(\mathcal{X}_b^2) \geq D(\mathcal{X}_{b_2}^2) \geq 1 - \gamma/2.$$

By a union bound we conclude that

$$D(\mathcal{X}_b^1 \cap \mathcal{X}_b^2) \geq 1 - \gamma$$

which implies that $\mathsf{Risk}_b^{\mathrm{PC}}(h) \geq 1-\gamma$. The reason is that for any $x \in \mathcal{X}_b^1 \cap \mathcal{X}_b^2$ there are $x^1, x^2 \in \mathcal{B}all(x, b)$ such that $h(x^1) \in \mathcal{Y}^1$ and $h(x^2) \in \mathcal{Y} \setminus \mathcal{Y}^1$ which means either $h(x) \neq h(x^1)$ or $h(x) \neq h(x^2)$.

2. The proof Part 2 directly follows from the definition of $\boldsymbol{\alpha}$ and an argument identical to that of Part 2 of Theorem 2.3.2.

3. Let $\mathcal{E} = \{x \in \mathcal{X} \mid \exists\, x' \in \mathcal{B}all_b(x), h(x) \neq h(x')\}$. We know that $D(\mathcal{E}) \geq 1/2$, therefore by Theorem 2.3.5 we have

$$\mathsf{Rob}(\mathcal{E}) \leq \int_0^\infty \boldsymbol{\alpha}(z) \cdot dz.$$

On the other hand, for every $x \in \mathcal{X}$ where $\ell = h(x)$, we have $\mathsf{d}(x, \mathcal{X} \setminus h^\ell) \leq b + \mathsf{d}(x, \mathcal{E})$ because we know that for any $x' \in \mathcal{E}$ there exist some $x'' \in \mathcal{B}all(x', b)$ such that $h(x') \neq h(x'')$. Therefore, we get that either $h(x') \neq h(x)$ or $h(x'') \neq h(x)$, which implies $\mathsf{d}(x, \mathcal{X} \setminus h^{h(x)}) \leq b + \mathsf{d}(x, \mathcal{E})$. Thus, we have

$$\mathsf{Rob}^{\mathrm{PC}}(h) \leq b + \mathsf{Rob}(\mathcal{E}) \leq b + \int_0^\infty \boldsymbol{\alpha}(z) \cdot dz.$$

4. Part 4 follows from an argument that is identical to that of Theorem 2.3.5. □

the following corollary directly follows Theorem 2.4.2 above and Definition 4.3.1 of Lévy families, just the same way Corollary 2.3.8 could be derived from Theorems 2.3.2 and 2.3.5 (by going through a variant of Theorems 2.3.7 for PC risk and robustness that we skip) to get asymptotic bounds of risk and robustness of classification tasks over Lévy spaces.

Corollary 2.4.3 (Asymptotic PC risk and robustness in normal Lévy families)**.** *Let P_n be a nice classification problem defined over a metric probability space that is a normal Lévy family.*

1. **PC risk and robustness.** *If for all $\ell \in \mathcal{Y}$ it holds that $D(h^\ell) \leq 0.99$ (i.e., h is not almost constant), then the amount of perturbations needed to achieve PC risk 0.99 is $O(1/\sqrt{n})$ and the (full) PC robustness of h is also $O(1/\sqrt{n})$.*

2. **Target-label PC risk and robustness.** *If a particular label ℓ happens with constant probability $D(h^\ell) = \Omega(1)$, then the perturbation needed to increase ℓ-label PC risk to 0.99 and the ℓ-label PC robustness of h are both at most $O(1/\sqrt{n})$. Furthermore, if $D(h^\ell) \geq \exp(-o(n))$ is subexponentially large, then the perturbation needed to increase the ℓ-label PC risk to $1 - \exp(-o(n))$ and the ℓ-label PC robustness of h are at both most $o(1)$.*

Chapter 3

Measuring Concentration for real-world distributions

3.1 Introduction

that under certain assumptions regarding the data distribution and the perturbation metric, adversarial examples are theoretically inevitable. As a result, for a broad set of theoretically natural metric probability spaces of inputs, there is no classifier for the data distribution that achieves adversarial robustness. For example, we showed that if the inputs come from any Normal Lévy family [Lévy, 1951], any classifier with a noticable test error will be vulnerable to small (i.e., sublinear in the typical norm of the inputs) perturbations.

Although such theoretical findings seem discouraging to the goal of developing robust classifiers, all these impossibility results depend on assumptions about data distributions that might not hold for cases of interest. Our work develops a general method for testing properties of concrete datasets against these theoretical assumptions.

Summary of Results Our work shrinks the gap between theoretical analyses of robustness of classification for theoretical data distributions and understanding the intrinsic robustness of actual datasets. Indeed, quantitative estimates of the intrinsic robustness[1] of benchmark image datasets such as MNIST and CIFAR-10 can provide us with a better understanding of the threat of adversarial examples for natural image distributions and may suggest promising directions for further improving classifier robustness. Our main

[1]See Definition 3.2.2 for the formal definition of intrinsic robustness. The term robustness has been used with different meanings in previous works (e.g., in Diochnos et al. [2018b], it refers to the average distances to the error region). However, all such uses refer to a desirable property of the classifier in being resilient to adversarial perturbations, which is the case here as well. See Diochnos et al. [2018b] for a taxonomy of different definitions.

technical contribution is a general method to evaluate the concentration of a given input distribution μ based on a set of data samples. We prove that by simultaneously increasing the sample size m and a complexity parameter T, the concentration of the empirical measure converges to the actual concentration of μ (Section 3.3). Using this method, we perform experiments to demonstrate the existence of robust error regions Compared with state-of-the-art robustly trained models, our estimated intrinsic robustness shows that, for most settings, there exists a large gap between the robust error achieved by the best current models and the theoretical limits implied by concentration. This suggests the concentration of measure is not the only reason behind the vulnerability of existing classifiers to adversarial perturbations. Thus, either there is room for improving the robustness of image classifiers (even with non-zero classification error) or a need for deeper understanding of the reasons for the gap between intrinsic robustness and the actual robustness achieved by robust models, at least for the datasets like the image classification benchmarks used in our experiments.

Related Work We are aware of only one previous work that attempts to heuristically estimate these properties. To extend their theoretical impossibility result to the practical distributions, Gilmer et al. [2018b] studied MNIST dataset to

Notation Lowercase boldface letters such as $\boldsymbol{x}$ are used to denote vectors, and $[n]$ is used to represent $\{1, 2, \ldots, n\}$. For any set $\mathcal{A}$, let $\mathsf{Pow}(\mathcal{A})$, $|\mathcal{A}|$ and $\mathbb{1}_{\mathcal{A}}(\cdot)$ be the set of measurable subsets of $\mathcal{A}$, cardinality and indicator function of $\mathcal{A}$, respectively. For any $\boldsymbol{x} \in \mathbb{R}^n$, the ℓ_∞-norm and ℓ_2-norm of $\boldsymbol{x}$ are defined as $\|\boldsymbol{x}\|_\infty = \max_{i\in[n]} |x_i|$ and $\|\boldsymbol{x}\|_2 = (\sum_{i\in[n]} x_i^2)^{1/2}$ respectively. Let $(\mathcal{X}, \mu)$ be a probability space and $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be some distance metric defined on $\mathcal{X}$. Define the empirical measure with respect to a set $\mathcal{S}$ sampled from μ as $\hat{\mu}_{\mathcal{S}}(\mathcal{A}) = \sum_{\boldsymbol{x}\in\mathcal{S}} \mathbb{1}_{\mathcal{A}}(\boldsymbol{x})/|\mathcal{S}|, \forall \mathcal{A} \subseteq \mathcal{X}$. Let $\mathcal{B}all(\boldsymbol{x}, \epsilon) = \{\boldsymbol{x}' \in \mathcal{X} : d(\boldsymbol{x}', \boldsymbol{x}) \leq \epsilon\}$ be the ball around $\boldsymbol{x}$ with radius ϵ. For any subset $\mathcal{A} \subseteq \mathcal{X}$, define the ϵ-expansion $\mathcal{A}_\epsilon = \{\boldsymbol{x} \in \mathcal{X} \colon \exists\, \boldsymbol{x}' \in \mathcal{B}all(\boldsymbol{x}, \epsilon) \cap \mathcal{A}\}$. The collection of the ϵ-expansions for members of any $\mathcal{G} \subseteq \mathsf{Pow}(\mathcal{X})$ is defined and denoted as $\mathcal{G}_\epsilon = \{\mathcal{A}_\epsilon : \mathcal{A} \in \mathcal{G}\}$.

3.2 Robustness and Concentration of Measure

In this chapter, we work with the following definition of *adversarial risk*:

Definition 3.2.1 (Adversarial Risk)**.** Let $(\mathcal{X}, \mu)$ be the probability space of instances and f^* be the underlying ground-truth. The *adversarial risk* of a classifier f in metric d with strength ϵ is defined as

$$\mathsf{AdvRisk}_\epsilon(f, f^*) = \Pr_{\boldsymbol{x} \leftarrow \mu} \left[\exists\, \boldsymbol{x}' \in \mathcal{B}all(\boldsymbol{x}, \epsilon) \text{ s.t. } f(\boldsymbol{x}') \neq f^*(\boldsymbol{x}')\right].^2$$

For $\epsilon = 0$, which allows no perturbation, the notion of adversarial risk coincides with traditional risk.

Definition 3.2.2 (Intrinsic Robustness)**.** Consider the same setting as in Definition 3.2.1. Let $\mathcal{F}$ be some family of classifiers, then the *intrinsic robustness* is defined as the maximum adversarial robustness that can be achieved within $\mathcal{F}$, namely

$$\mathrm{Rob}_\epsilon(\mathcal{F}, f^*) = 1 - \inf_{f \in \mathcal{F}} \left\{\mathsf{AdvRisk}_\epsilon(f, f^*)\right\}.$$

In this chapter, we specify $\mathcal{F}$ as the family of imperfect classifiers that have risk at least $\alpha \in (0, 1)$.

Previous work shows a connection between concentration of measure and the intrinsic robustness with respect to some families of classifiers (Gilmer et al. [2018b], Fawzi et al. [2018], Mahloujifar et al. [2018b], Shafahi et al. [2018c]). The concentration of measure on a metric probability space is defined by a concentration function as follows.

Definition 3.2.3 (Concentration Function)**.** Consider a metric probability space $(\mathcal{X}, \mu, d)$. Suppose $\epsilon > 0$ and $\alpha \in (0, 1)$ are given parameters, then the *concentration function* of the probability measure μ with respect to ϵ, α is defined as

$$h(\mu, \alpha, \epsilon) = \inf_{\mathcal{E} \in \mathsf{Pow}(\mathcal{X})} \left\{\mu(\mathcal{E}_\epsilon) \colon \mu(\mathcal{E}) \geq \alpha\right\}.$$

Note that the standard notion of concentration function (e.g., see Talagrand [1995]) is related to a special case of Definition 3.2.3 by fixing $\alpha = 1/2$.

Generalizing the result of Gilmer et al. [2018b] about instances drawn from spheres, Mahloujifar et al. [2018b] showed that, in general, if the metric probability space of instances is concentrated, then any classifier with 1% risk incurs large adversarial risk for small amount of perturbations.

Theorem 3.2.4 (Mahloujifar et al. [2018b])**.** *Let $(\mathcal{X}, \mu)$ be the probability space of instances and f^* be the underlying ground-truth. For any classifier f, we have*

$$\mathsf{AdvRisk}_\epsilon(f, f^*) \geq h(\mu, \mathsf{Risk}(f, f^*), \epsilon).$$

[2]Note that bounding l_p norm might be restrictive for the adversary [Gilmer et al., 2018a] and this definition only covers a subset of possible adversaries.

In order for this theorem to be useful, we need to know the concentration function. The behavior of this function is studied extensively for certain theoretical metric probability spaces [Ledoux, 2001, Milman and Schechtman, 1986]. However, it is not known how to measure the concentration function for arbitrary metric probability spaces. In this chapter, we provide a framework to (algorithmically) bound the concentration function from i.i.d. samples from a distribution. Namely, we want to solve the following optimization task using our i.i.d. samples:

$$\underset{\mathcal{E}\in\mathsf{Pow}(\mathcal{X})}{\text{minimize}} \;\; \mu(\mathcal{E}_\epsilon) \quad \text{subject to } \mu(\mathcal{E}) \geq \alpha. \tag{3.1}$$

We aim to estimate the minimum possible adversarial risk, which captures the intrinsic robustness for classification in terms of the underlying distribution μ, conditioned on the fact that the original risk is at least α. Note that solving this optimization problem only shows the possibility of existence of an error region $\mathcal{E}$ with certain (small) expansion. This means that there could potentially exist a classifier with risk at least α and adversarial risk equal to the solution of the optimization problem of (3.1). Actually *finding* such an optimally robust classifier (with error α) using a learning algorithm might be a much more difficult task or even infeasible. We do not consider that problem in this chapter.

3.3 Method for Measuring Concentration

In this section, we present a method to measure the concentration of measure on a metric probability space using i.i.d. samples. To measure concentration, there are two main challenges:

1. Measuring concentration appears to require knowledge of the density function of the distribution, but we only have a data set sampled from the distribution.

2. Even with the density function, we have to find the best possible subset among all the subsets of the space, which seems infeasible.

We show how to overcome these challenges and find the actual concentration in the limit by first empirically simulating the distribution and then narrowing down our search space to a specific collection of subsets. Our results show that for a carefully chosen family of sets, the set with minimum expansion can be approximated using polynomially many samples. On the other hand, the minimum expansion convergence to the actual concentration (without the limits on the sets) as the complexity of the collection goes to infinity.

Before stating our main theorems, we introduce two useful definitions. The following definition captures the concentration function for a specific collection of subsets.

Definition 3.3.1 (Concentration Function for a Collection of Subsets)**.** Consider a metric probability space $(\mathcal{X}, \mu, d)$. Let $\epsilon > 0$ and $\alpha \in (0, 1)$ be given parameters, then the *concentration function* of the probability measure μ with respect to ϵ, α and a collection of subsets $\mathcal{G} \subseteq \mathsf{Pow}(\mathcal{X})$ is defined as

$$h(\mu, \alpha, \epsilon, \mathcal{G}) = \inf_{\mathcal{E} \in \mathcal{G}} \{\mu(\mathcal{E}_\epsilon) \colon \mu(\mathcal{E}) \geq \alpha\}.$$

When $\mathcal{G} = \mathsf{Pow}(\mathcal{X})$, we write $h(\mu, \alpha, \epsilon)$ for simplicity.

We also need to define the notion of complexity penalty for a collection of subsets. The complexity penalty for a collection of subsets captures the rate of the uniform convergence for the subsets in that collection. One can get such uniform convergence rates using the VC dimension or Rademacher complexity of the collection.

Definition 3.3.2 (Complexity Penalty)**.** Let $\mathcal{G} \subseteq \mathsf{Pow}(\mathcal{X})$ be a collection of subsets of $\mathcal{X}$. A function $\phi \colon \mathbb{N} \times \mathbb{R} \to [0, 1]$ is a complexity penalty for $\mathcal{G}$ iff for any probability measure μ supported on $\mathcal{X}$ and any $\delta \in [0, 1]$, we have

$$\Pr_{S \leftarrow \mu^m}[\exists\, \mathcal{E} \in \mathcal{G} \;\text{ s.t. }\; \|\mu(\mathcal{E}) - \hat{\mu}_S(\mathcal{E})\| \geq \delta] \leq \phi(m, \delta).$$

Theorem 3.3.3 shows how to overcome the challenge of measuring concentration from finite samples, when the concentration is defined with respect to specific families of subsets. Namely, it shows that the empirical concentration is close to the true concentration, if the underlying collection of subsets is not too complex. The proof of Theorem 3.3.3 is provided in Appendix 3.4.1.

Theorem 3.3.3 (Generalization of Concentration)**.** *Let $(\mathcal{X}, \mu, d)$ be a metric probability space and $\mathcal{G} \subseteq \mathsf{Pow}(\mathcal{X})$. For any $\delta, \alpha, \epsilon \in [0, 1]$, we have*

$$\Pr_{S \leftarrow \mu^m}[h(\mu, \alpha - \delta, \epsilon, \mathcal{G}) - \delta \leq h(\hat{\mu}_S, \alpha, \epsilon, \mathcal{G}) \leq h(\mu, \alpha + \delta, \epsilon, \mathcal{G}) + \delta] \geq 1 - 2\big(\phi(m, \delta) + \phi_\epsilon(m, \delta)\big)$$

where ϕ and ϕ_ϵ are complexity penalties for $\mathcal{G}$ and $\mathcal{G}_\epsilon$ respectively.

Remark 3.3.4. Theorem 3.3.3 shows that if we narrow down our search to a collection of subsets $\mathcal{G}$ such that both $\mathcal{G}$ and $\mathcal{G}_\epsilon$ have small complexity penalty, then we can use the empirical distribution to measure concentration of measure for that specific collection. Note that the generalization bound of Theorem 3.3.3 depends on complexity penalties for both $\mathcal{G}$ and $\mathcal{G}_\varepsilon$. Therefore, in order for this theorem to be useful, the collection $\mathcal{G}$ must be chosen in a careful way. For example, if $\mathcal{G}$ has bounded VC dimension, then $\mathcal{G}_\epsilon$ might still have a very large VC dimension. Alternatively, $\mathcal{G}$ might denote the collection of subsets that are decidable by a neural network of a certain size. In that case, even though there are well known complexity penalties for such collections (see Neyshabur et al. [2017]), the complexity of their *expansions* is unknown. In fact, relating

the complexity penalty for expansion of a collection to that of the original collection is tightly related to generalization bounds in the adversarial settings, which has also been the subject of several recent works [Cullina et al., 2018, Attias et al., 2018, Montasser et al., 2019, Yin et al., 2018b, Raghunathan et al., 2019].

The following theorem, proved in Appendix 3.4.2, states that if we gradually increase the complexity of the collection and the number of samples together, the empirical estimate of concentration converges to actual concentration, as long as several conditions hold. Theorem 2.3.2 and the techniques used in its proof are inspired by the work of Scott and Nowak [2006] on learning minimum volume sets.

Theorem 3.3.5. *Let $\{\mathcal{G}(T)\}_{T\in\mathbb{N}}$ be a family of subset collections defined over a space $\mathcal{X}$. Let $\{\phi^T\}_{T\in\mathbb{N}}$ and $\{\phi_\epsilon^T\}_{T\in\mathbb{N}}$ be two families of complexity penalty functions such that ϕ^T and ϕ_ϵ^T are complexity penalties for $\mathcal{G}(T)$ and $\mathcal{G}_\epsilon(T)$ respectively, for some $\epsilon \in [0,1]$. Let $\{m(T)\}_{T\in\mathbb{N}}$ and $\{\delta(T)\}_{T\in\mathbb{N}}$ be two sequences such that $m(T) \in \mathbb{N}$ and $\delta(T) \in [0,1]$.*

Consider a sequence of datasets $\{S_T\}_{T\in\mathbb{N}}$, where S_T consists of $m(T)$ i.i.d. samples from a measure μ supported on $\mathcal{X}$. Also let $\alpha \in [0,1]$ be such that h is locally continuous w.r.t the second parameter at point $(\mu, \alpha, \epsilon, \mathsf{Pow}(\mathcal{X}))$. If all the following hold,

1. $\sum_{T=1}^{\infty} \phi^T(m(T), \delta(T)) < \infty$
2. $\sum_{T=1}^{\infty} \phi_\epsilon^T(m(T), \delta(T)) < \infty$
3. $\lim_{T\to\infty} \delta(T) = 0$
4. $\lim_{T\to\infty} h(\mu, \alpha, \epsilon, \mathcal{G}(T)) = h(\mu, \alpha, \epsilon)$

then with probability 1, we have $\lim_{T\to\infty} h(\hat{\mu}_{S_T}, \alpha, \epsilon, \mathcal{G}(T)) = h(\mu, \alpha, \epsilon)$.

Remark 3.3.6. In Theorem 2.3.2, the first two conditions restrict the growth rate for the complexity of the collections. Namely, we need the complexity penalties $\phi^T(m(T), \delta(T))$ and $\phi_\epsilon^T(m(T), \delta(T))$ to rapidly approach 0 as $T \to \infty$, which means the complexity of $\mathcal{G}(T)$ and $\mathcal{G}_\epsilon(T)$ should grow at a slow rate. The third condition requires that our generalization error goes to zero as we increase T. Note that the complexity penalty is a decreasing function with respect to δ, which means condition 3 makes achieving the first two conditions harder. However, since the complexity penalty is a function of both δ and sample size, we can still increase the sample size with a faster rate to satisfy the first two conditions. Finally, the fourth condition requires our approximation error goes to 0 as we increase T. Note that this condition holds for any family of collections of subsets that is a universal approximator (e.g., decision trees or neural networks). However, in order for our theorem to hold, we also need all the other conditions. In particular, we cannot use decision trees or neural networks as our collection of subsets, because we do not know if there is a complexity penalty for them that satisfies condition 2.

3.3.1 Special Case of ℓ_∞

In this subsection, we show how to instantiate Theorem 2.3.2 for the case of ℓ_∞. Below, we introduce a special collection of subsets characterized by the *complement of a union of hyperrectangles*:

Definition 3.3.7 (Complement of union of hyperrectangles)**.** For any positive integer T, the collection of subsets specified by the *complement of a union of T n-dimensional hyperrectangles* is defined as

$$\mathcal{CR}(T,n) = \left\{\mathbb{R}^n \setminus \cup_{t=1}^{T} \mathcal{R}ect(\boldsymbol{u}^{(t)}, \boldsymbol{r}^{(t)}) \colon \forall t \in [T], (\boldsymbol{u}^{(t)}, \boldsymbol{r}^{(t)}) \in \mathbb{R}^n \times \mathbb{R}^n_{\geq 0}\right\},$$

where $\mathcal{R}ect(\boldsymbol{u}, \boldsymbol{r}) = \left\{\boldsymbol{x} \in \mathcal{X} : \forall j \in [n], |x_j - u_j| \leq r_j/2\right\}$ denotes the hyperrectangle centered at $\boldsymbol{u}$ with $\boldsymbol{r}$ representing the edge size vector. When n is free of context, we simply write $\mathcal{CR}(T)$.

Recall that our goal is to find a subset $\mathcal{E} \in \mathbb{R}^n$ such that $\mathcal{E}$ has measure at least α and the ϵ_∞-expansion of $\mathcal{E}$ under ℓ_∞ has the minimum measure. To achieve this goal, we approximate the distribution μ with an empirical distribution $\hat{\mu}_S$, and limit our search to the special collection $\mathcal{CR}(T)$ (though our goal is to find the minimum concentration around arbitrary subsets). Namely, what we find is still an *upper bound* on the concentration function, and it is an upper bound that we know it converges the actual value in the limit. Our problem thus becomes the following optimization task:

$$\underset{\mathcal{E} \in \mathcal{CR}(T)}{\text{minimize}} \;\; \hat{\mu}_{\mathcal{S}}(\mathcal{E}_{\epsilon_\infty}) \quad \text{subject to } \hat{\mu}_{\mathcal{S}}(\mathcal{E}) \geq \alpha. \tag{3.2}$$

The following theorem provides the key to our empirical method by providing a convergence guarantee. It states that if we increase the number of rectangles and the number of samples together in a careful way, the solution to the problem using restricted sets converges to the true concentration.

Theorem 3.3.8. *Consider a nice metric probability space $(\mathbb{R}^n, \mu, \ell_\infty)$. Let $\{S_T\}_{T\in\mathbb{N}}$ be a family of datasets such that for all $T \in \mathbb{N}$, S_T contains at least T^4 i.i.d. samples from μ. For any ϵ_∞ and $\alpha \in [0,1]$, if h is locally continuous w.r.t the second parameter at point $(\mu, \alpha, \epsilon_\infty)$, then with probability 1 we get*

$$\lim_{T\to\infty} h(\hat{\mu}_{S_T}, \alpha, \epsilon_\infty, \mathcal{CR}(T)) = h(\mu, \alpha, \epsilon_\infty).$$

Note that the size of S_T is selected as T^4 to guarantee conditions 1 and 2 are satisfied in Theorem 2.3.2. In fact, we can tune the parameters more carefully to get T^2, instead of T^4, but the convergence will be slower. See Appendix 3.4.3 for the proof.

3.3.2 Special Case of ℓ_2

This subsection demonstrates how to apply Theorem 2.3.2 to the case of ℓ_2. The following definition introduces the collection of subsets characterized by a *union of balls*:

Definition 3.3.9 (Union of Balls)**.** For any positive integer T, the collection of subsets specified by a *union of T n-dimensional balls* is defined as

$$\mathcal{B}(T,n) = \left\{ \cup_{t=1}^{T} \mathcal{B}all(\boldsymbol{u}^{(t)}, \boldsymbol{r}^{(t)}) \colon \forall t \in [T], (\boldsymbol{u}^{(t)}, \boldsymbol{r}^{(t)}) \in \mathbb{R}^n \times \mathbb{R}^n_{\geq 0} \right\}.$$

When n is free of context, we simply write $\mathcal{B}(T)$.

By restricting our search to the collection of a union of balls $\mathcal{B}(T)$ and replacing the underlying distribution μ with the empirical one $\hat{\mu}_{\mathcal{S}}$, our problem becomes the following optimization task

$$\underset{\mathcal{E}\in\mathcal{B}(T)}{\text{minimize}} \ \hat{\mu}_{\mathcal{S}}(\mathcal{E}_{\epsilon_2}) \quad \text{subject to } \hat{\mu}_{\mathcal{S}}(\mathcal{E}) \geq \alpha. \tag{3.3}$$

Theorem 3.3.10, proven in Appendix 3.4.4, guarantees that if we increase the number of balls and samples together in a careful way, the solution to the empirical problem (3.3) converges to the true concentration.

Theorem 3.3.10. *Consider a nice metric probability space $(\mathbb{R}^n, \mu, \ell_2)$. Let $\{S_T\}_{T\in\mathbb{N}}$ be a family of datasets such that for all $T \in \mathbb{N}$, S_T contains at least T^4 i.i.d. samples from μ. For any ϵ_2 and $\alpha \in [0,1]$, if h is locally continuous w.r.t the second parameter at point $(\mu, \alpha, \epsilon_2)$, then with probability 1 we get*

$$\lim_{T\to\infty} h(\hat{\mu}_{S_T}, \alpha, \epsilon_2, \mathcal{B}(T)) = h(\mu, \alpha, \epsilon_2).$$

3.4 Proofs of Theorems in Section 3.3

In this section, we prove Theorems 3.3.3, 2.3.2, 3.3.8 and 3.3.10.

3.4.1 Proof of Theorem 3.3.3

Proof. Define $g(\mu, \alpha, \epsilon, \mathcal{G}) = \text{argmin}_{\mathcal{E}\in\mathcal{G}} \{\mu(\mathcal{E}_\epsilon) \colon \mu(\mathcal{E}) \geq \alpha\}$, and let $\mathcal{E} = g(\mu, \alpha+\delta, \epsilon, \mathcal{G})$ and $\hat{\mathcal{E}} = g(\hat{\mu}_S, \alpha, \epsilon, \mathcal{G})$. (Note that these sets achieving the minimum might not exist, in which case we select a set for which the expansion is arbitrarily close to the infimum and every step of the proof will extend to this variant).

By the definition of the complexity penalty we have

$$\Pr_{S\leftarrow\mu^m}\left[\|\mu(\hat{\mathcal{E}})-\hat{\mu}_S(\hat{\mathcal{E}})\|\geq\delta\right]\leq\phi(m,\delta)$$

which implies

$$\Pr_{S\leftarrow\mu^m}[\mu(\hat{\mathcal{E}})\leq\alpha-\delta]\leq\phi(m,\delta).$$

Therefore, by the definition of h we have

$$\Pr_{S\leftarrow\mu^m}[\mu(\hat{\mathcal{E}}_\epsilon)\leq h(\mu,\alpha-\delta,\epsilon,\mathcal{G})]\leq\phi(m,\delta). \tag{3.4}$$

On the other hand, based on the definition of ϕ_ϵ we have

$$\Pr_{S\leftarrow\mu^m}\left[\|\mu(\hat{\mathcal{E}}_\epsilon)-\hat{\mu}_S(\hat{\mathcal{E}}_\epsilon)\|\geq\delta\right]\leq\phi_\epsilon(m,\delta). \tag{3.5}$$

Combining Equation 3.4 and Equation 3.5, and by a union bound we get

$$\Pr_{S\leftarrow\mu^m}[\hat{\mu}_S(\hat{\mathcal{E}}_\epsilon)\leq h(\mu,\alpha-\delta,\epsilon,\mathcal{G})-\delta]\leq\phi(m,\delta)+\phi_\epsilon(m,\delta)$$

which by the definition of $\hat{\mathcal{E}}$ implies that

$$\Pr_{S\leftarrow\mu^m}[h(\hat{\mu}_S,\alpha,\epsilon,\mathcal{G})\leq h(\mu,\alpha-\delta,\epsilon,\mathcal{G})-\delta]\leq\phi(m,\delta)+\phi_\epsilon(m,\delta). \tag{3.6}$$

Now we bound the probability for the other side of our inequality. By the definition of the notion of complexity penalty we have

$$\Pr_{S\leftarrow\mu^m}[\|\mu(\mathcal{E})-\hat{\mu}_S(\mathcal{E})\|\geq\delta]\leq\phi(m,\delta)$$

which implies

$$\Pr_{S\leftarrow\mu^m}[\hat{\mu}_S(\mathcal{E})\leq\alpha]\leq\phi(m,\delta).$$

Therefore, by the definition of h we have,

$$\Pr_{S\leftarrow\mu^m}[\hat{\mu}_S(\mathcal{E}_\epsilon)\leq h(\hat{\mu}_S,\alpha,\epsilon,\mathcal{G})]\leq\phi(m,\delta). \tag{3.7}$$

On the other hand, based on the definition of ϕ_ϵ we have

$$\Pr_{S\leftarrow\mu^m}[\|\mu(\mathcal{E}_\epsilon) - \hat{\mu}_S(\mathcal{E}_\epsilon)\| \geq \delta] \leq \phi(m,\delta) + \phi_\epsilon(m,\delta). \tag{3.8}$$

Combining Equations 3.7 and 3.8, by union bound we get

$$\Pr_{S\leftarrow\mu^m}[\mu(\mathcal{E}_\epsilon) \leq h(\hat{\mu}_S,\alpha,\epsilon,\mathcal{G}) - \delta] \leq \phi(m,\delta) + \phi_\epsilon(m,\delta)$$

which by the definition of $\mathcal{E}$ implies

$$\Pr_{S\leftarrow\mu^m}[h(\mu,\alpha+\delta,\epsilon,\mathcal{G}) \leq h(\hat{\mu}_S,\alpha,\epsilon,\mathcal{G}) - \delta] \leq \phi(m,\delta) + \phi_\epsilon(m,\delta). \tag{3.9}$$

Now combining Equations 3.6 and 3.9, by union bound we have

$$\Pr_{S\leftarrow\mu^m}[h(\mu,\alpha-\delta,\epsilon,\mathcal{G}) - \delta \leq h(\hat{\mu}_S,\alpha,\epsilon,\mathcal{G}) \leq h(\mu,\alpha+\delta,\epsilon,\mathcal{G}) + \delta] \geq 1 - 2\left(\phi(m,\delta) + \phi_\epsilon(m,\delta)\right).$$

□

3.4.2 Proof of Theorem 2.3.2

In this section, we prove Theorem 2.3.2 using ideas similar to ideas used in Scott and Nowak [2006]. Before proving the theorem, we lay out the following lemma which will be used in the proof.

Lemma 3.4.1 (Borel-Cantelli Lemma)**.** *Let $\{E_T\}_{T\in\mathbb{N}}$ be a series of events such that*

$$\sum_{T=1}^{\infty} \Pr[E_T] < \infty$$

Then with probability 1, only finite number of events will occur.

Now we are ready to prove Theorem 2.3.2.

Proof of Theorem 2.3.2. Define E_T to be the event that

$$h(\mu,\alpha-\delta(T),\epsilon,\mathcal{G}(T)) - \delta(T) > h(\hat{\mu}_{S_T},\alpha,\epsilon) \text{ or } h(\mu,\alpha+\delta(T),\epsilon,\mathcal{G}(T)) + \delta(T) < h(\hat{\mu}_{S_T},\alpha,\epsilon,\mathcal{G}).$$

Based on Theorem 3.3.3 we have $\Pr[E_T] \leq 2 \cdot (\phi^T(m(T), \delta(T)) + \phi_\epsilon^T(m(T), \delta(T)))$. Therefore, by Conditions 1 and 2 we have

$$\sum_{T=1}^{\infty} \Pr[E_T] \leq 2 \left(\sum_{T=1}^{\infty} \phi^T \left(m(T), \delta(T)\right) + \phi_\epsilon^T \left(m(T), \delta(T)\right) \right) < \infty.$$

Now by Lemma 3.4.1, we know there exist with measure 1 some $j \in \mathbb{N}$, such that for all $T \geq j$,

$$h(\mu, \alpha - \delta(T), \epsilon, \mathcal{G}(T)) - \delta(T) \leq h(\hat{\mu}_{S_T}, \alpha, \epsilon, \mathcal{G}(T)) \leq h(\mu, \alpha + \delta(T), \epsilon, \mathcal{G}(T)) + \delta(T).$$

The above implies that

$$\lim_{T\to\infty} h(\mu, \alpha - \delta(T), \epsilon, \mathcal{G}(T)) - \delta(T) \leq \lim_{T\to\infty} h(\hat{\mu}_{S_T}, \alpha, \epsilon, \mathcal{G}(T)) \leq \lim_{T\to\infty} h(\mu, \alpha + \delta(T), \epsilon, \mathcal{G}(T)) + \delta(T).$$

We know that

$$\begin{aligned} \lim_{T\to\infty} h(\mu, \alpha - \delta(T), \epsilon, \mathcal{G}(T)) &= \lim_{T_1\to\infty} \lim_{T_2\to\infty} h(\mu, \alpha - \delta(T_1), \epsilon, \mathcal{G}(T_2)) \\ \text{(By condition 4)} \quad &= \lim_{T_1\to\infty} h(\mu, \alpha - \delta(T_1), \epsilon) \\ \text{(By local continuity and condition 3)} \quad &= h(\mu, \alpha, \epsilon). \end{aligned}$$

Similarly, we have

$$\lim_{T\to\infty} h(\mu, \alpha + \delta(T), \epsilon, \mathcal{G}(T)) = h(\mu, \alpha, \epsilon).$$

Therefore we have,

$$\lim_{T\to\infty} h(\mu, \alpha, \epsilon) - \delta(T) \leq \lim_{T\to\infty} h(\hat{\mu}_{S_T}, \alpha, \epsilon, \mathcal{G}(T)) \leq \lim_{T\to\infty} h(\mu, \alpha, \epsilon) + \delta(T)$$

which by condition 3 implies

$$\lim_{T\to\infty} h(\hat{\mu}_{S_T}, \alpha, \epsilon, \mathcal{G}(T)) = h(\mu, \alpha, \epsilon).$$

□

3.4.3 Proof of Theorem 3.3.8

Proof. This theorem follows from our general Theorem 2.3.2. We show that the choice of parameters here satisfies all four conditions of Theorem 2.3.2.

If we let $\mathcal{G}(T)$ to be the collection of subsets specified by complement of union of T hyperrectangles. Then $\mathcal{G}_\epsilon(T)$ will be the collection of of subsets specified by complement of union of T hyperrectangles that are

bigger than ϵ in each coordinate. Therefore we have $\mathcal{G}_\epsilon(T) \subset \mathcal{G}(T)$. We know that the VC dimension of $\mathcal{G}(T)$ is $d_T = O(nT\log(T))$ because the VC dimension of all hyperrectangles is $O(n)$ and the functions formed by T fold union of functions in a VC class is at most $n \cdot T\log(T)$ (See Eisenstat and Angluin [2007]). Therefore, by VC inequality we have

$$\Pr_{S\leftarrow\mu^m}\left[\sup_{\mathcal{E}\in\mathcal{G}(T)} |\mu(\mathcal{E}) - \hat{\mu}_S(\mathcal{E})| \geq \delta\right] \leq 8e^{nT\log(T)\log(m) - m\delta^2/128}.$$

Therefore $\Phi^T(m,\delta) = 8e^{nT\log(T)\log(m)-m\delta^2/128}$ is a complexity penalty for both $\mathcal{G}(T)$ and $\mathcal{G}_\epsilon(T)$. Hence, if we define $\delta(T) = 1/T$ and $m(T) \geq T^4$, then the first three conditions of Theorem 2.3.2 are satisfied. The fourth condition is also satisfied by the universal consistency of histogram rules (See Devroye et al. [2013], Ch. 9). □

3.4.4 Proof of Theorem 3.3.10

Proof. Similar to Theorem 3.3.8 This theorem follows from our general Theorem 2.3.2. We show that the choice of parameters here satisfies all four conditions of Theorem 2.3.2.

If we let $\mathcal{G}(T)$ to be the collection of subsets specified by union of T balls. Then $\mathcal{G}_\epsilon(T)$ will be the collection of of subsets specified by union of T balls with diameter at least ϵ. Similar to the proof of Theorem 3.3.8, we have $\mathcal{G}_\epsilon(T) \subset \mathcal{G}(T)$. We know that the VC dimension of all balls is $O(n)$ so using the fact that $\mathcal{G}(T)$ is T fold union of balls, the VC dimension of $\mathcal{G}(T)$ is $d_T = O(nT\log(T))$ (See Eisenstat and Angluin [2007]). Therefore, by VC inequality we have complexity penalties similar to those of Theorem 3.3.8 for both $\mathcal{G}(T)$ and $\mathcal{G}_\epsilon(T)$. Hence, if we define $\delta(T) = 1/T$ and $m(T) \geq T^4$, then the first three conditions of Theorem 2.3.2 are satisfied. The fourth condition is also satisfied by the universal consistency of kernel-based rules (See Devroye et al. [2013] , Ch. 10). □

Chapter 4

Lower Bounds for Adversarially Robust PAC Learning

4.1 Introduction

A fundamental question in robust learning is whether one can design learning algorithms that achieve "uniform converegence" even under such adversarial perturbations. Namely, we want to know when we can learn a robust classifier h that still correctly classifies its inputs even if they are adversarially perturbed in a limited way. Indeed, one can ask when (ε, δ) PAC (probably approximately correct) learning [Valiant, 1984] is possible in adversarial settings. More formally, the goal here is to learn a robust h from the data set $\mathcal{S}$ consisting of m independently sampled labeled (non-adversarial) instances in such a way that, with probability $1 - \delta$ over the learning process, the produced h has error at most ε even under "limited" adversarial perturbations of the input. This limitation is carefully defined by some metric d defined over the input space $\mathcal{X}$ and some upper bound "budget" b on the amount of perturbations that the adversary can introduce. That is, we would like to minimize

$$\mathsf{AdvRisk}(h) = \Pr_{x \leftarrow D}[\exists\ \widetilde{x} \colon d(x, \widetilde{x}) \leq b, h(\widetilde{x}) \neq c(\widetilde{x})] \leq \varepsilon$$

where $\mathsf{AdvRisk}$ is the "adversarial" risk, and $c(\cdot)$ is the ground truth (i.e., the concept function). As we discussed in Section 1 of this part, there are multiple definitions of adversarial examples. Here we mention two of those definitions again for reader's convenience.

Error-Region Adversarial Risk The above notion of adversarial risk has been used implicitly or explicitly in previous work [Gilmer et al., 2018b, Diochnos et al., 2018b, Degwekar and Vaikuntanathan, 2019, Ford

et al., 2019] and was formalized by Diochnos et al. [2018b] as the "error-region" adversarial risk, because the adversary's goal here is to push $\widetilde{x}$ into the error region

$$\mathcal{E} = \{x \mid h(x) \neq c(x)\}\,.$$

Corrupted-Input Adversarial Risk Another notion of adversarial risk (that is similar, but still different from the error-region adversarial risk explained above) has been used in many works such as [Feige et al., 2015, Madry et al., 2018, Bubeck et al., 2018a] in which the perturbed $\widetilde{x}$ is interpreted as a "corrupted input". Namely, here the goal of the learner is to find the label of the original *untampered* point x by only having its corrupted version $\widetilde{x}$, and thus adversary's success criterion is to reach $d(x, \widetilde{x}) \leq b, h(\widetilde{x}) \neq c(x)$. Hence, in that setting, the goal of the learner is to find an h that minimizes

$$\Pr_{x \leftarrow D}[\exists\ \widetilde{x} \colon d(x, \widetilde{x}) \leq b, h(\widetilde{x}) \neq c(x)].$$

It is easy to see that, if the ground truth $c(x)$ does not change under b-perturbations, $c(x) = c(\widetilde{x})$, the two notions of error-region and corrupted-input adversarial risk will be equal. In particular, this is the case for practical distributions of interest, such as images or voice, where sufficiently-small perturbations do not change human's judgment about the true label. However, if b-perturbations can change the ground truth, $c(x) \neq c(\widetilde{x})$, the two definitions are incomparable.

Why PAC Learning under General Perturbation Is Meaningful We emphasize that, even if the b-perturbation *could* change the ground truth's judgement, asking whether a learning problem is PAC learnable or not is very meaningful. In fact, the problem is still "realizable" under the right definition (for the general setting) because if one happens to learn the concept class c completely and output the hypothesis $h = c$, then h will have adversarial risk *zero* under the error-region definition. In other words, the ground truth can still be *predicted* robustly. Thus, it is a natural question to ask whether one can learn a hypothesis h that has small adversarial risk even under perturbations that are still small in magnitude compared to the size of the original sample x.

Previous Work Several works have already studied PAC learning with provable guarantees under adversarial perturbations [Bubeck et al., 2018b, Cullina et al., 2018, Feige et al., 2018, Attias et al., 2018, Khim and Loh, 2018, Yin et al., 2018b, Montasser et al., 2019]. However, all these works use the *corrupted-input* notion of adversarial risk. In particular, it is proved by Attias et al. [2018] that robust learning might require more data, but it was also shown by Attias et al. [2018], Bubeck et al. [2018b] that in natural settings, if robust

classification is feasible, robust classifiers could be found with a sample complexity that is only *polynomially* larger than that of normal learning. This leads us to our central question:

> *What problems are PAC learnable under evasion attacks that perturb instances into the error region? If PAC learnable, what is their sample complexity?*

Note that previous positive (or negative) results about PAC learning under the corrupted-input definition do not answer our question above, as we study general arbitrary perturbation budgets allowed to the adversary. Also, when the ground truth can also change under that amount of perturbation we have to use the error-region definition. More technically, we note that positive results about adversarial PAC learning (cited above) do not answer our question for the following reason. When the allowed perturbation is limited to keep the ground truth c robust, then the two definition are equivalent, yet, when the budget gets larger, then a positive result proved using the corrupted-input definition would simply mean that there is a way to learn a hypothesis h that has only ε adversarial risk more than the "best possible" h^*. However, this could be just a side affect that any h^* under the corrupted-input definition (and certain amount of allowed perturbations) could have very large (even $1-\varepsilon$) adversarial risk, making the job of agnostic learning trivial (to output anything). That is why, when we work with arbitrary perturbation budget, we need to employ the error-region definition, which still allows $c = h$ to have small adversarial risk, which is the intuitive decision as well.

4.1.1 Summary of Results

In this section, we initiate a formal study of PAC learning under adversarial perturbations, where the goal of the adversary is to increase the error-region adversarial risk using small (sublinear $o(\|x\|)$) perturbations of the inputs x. Therefore, in what follows, whenever we refer to adversarial risk, by default it means the error-region variant. Before we proceed, so that we can better put our work into perspective, we first give a short description explaining our main contributions in previous work that we have done that is related to the work of this chapter.

Putting our Work into Perspective Our work in Mahloujifar et al. [2018a] dealt with clean-label "poisoning" attacks in situations where the adversary has the opportunity to substitute $\approx p$ *randomly selected* fraction of the training examples, with some "adversarial" ones of their choosing but the labels of the injected training examples need to respect the ground truth c (and hence the term "clean-label"). Such attacks are called p-tampering. In particular, the adversary can also effectively reduce the sample size by repeating training examples at the randomly selected p fraction of the changed examples. Our work in Mahloujifar et al. [2018a] is connected to the second part of our work here, where we formalize and study hybrid attacks.

In Diochnos et al. [2018b] we provided a taxonomy of definitions that are used for the computation of adversarial examples and ultimately for the computation of the adversarial risk and robustness of learned classifiers. In addition, we showed that when misclassification is really the goal of an adversarial perturbation, then there is a natural problem (under the uniform distribution over $\{0,1\}^n$) where only the error region definition computes the adversarial risk and robustness correctly. As a result we decided to use the error-region adversarial risk and robustness by default. Finally in that work, we computed inherent bounds that classifiers have on risk and robustness (based on the error-region) when again the distribution is uniform over $\{0,1\}^n$ – these bounds were information-theoretic.

In Mahloujifar et al. [2018b] extended the previous (information-theoretic) inherent bounds that classifiers have on adversarial risk and robustness, from the uniform distribution over $\{0,1\}^n$, to information-theoretic bounds on any Normal Lévy families (which, for example, include product distributions over $\{0,1\}^n$ and many more examples), using the phenomenon of concentration of measure. In the same work, we showed that the same phenomenon of concentration of measure allows an adversary to substitute a sublinear amount of training examples (that is, in a poisoning attack) and increase the probability of any bad property (e.g., misclassifying a particular test instance) from some non-negligible value (say 1%) to almost certainty (say 99%) by changing only $\approx \sqrt{n}$ of the examples, using correct labels.

The works of Mahloujifar and Mahmoody [2019c], Etesami et al. [2019a] extended the above information-theoretic results on poisoning and evasion attacks by explicitly providing efficient (polynomial-time) attacks on product distributions, so that the perturbation budget used in the attack scheme matches the information-theoretic bounds from the previous work of Mahloujifar et al. [2018b].

In Mahloujifar et al. [2019e], it was shown how to empirically approximate (more specifically, upper bound) the concentration of a distribution of inputs given only (black-box) samples from the distribution. This is relevant to the line of work in which concentration of measure plays a key role in the hardness of adversarially robust learning, because one would need to know whether specific input distributions of interest (e.g., MNIST) are concentrated or not.

As the description above shows, our previous work on adversarial examples has focused on the power of an attacker. However, once one fixes the perturbation budget for the attacker, a natural question to ask is to what extent a learner can *defend* the hypothesis that it forms – that is, flip the table of the point of view of the analysis. Indeed, our first result in this section shows that a PAC learner needs exponentially many training examples in order to form a robust hypothesis when the attacker can substitute only a sublinear amount of the coordinates of the test instance. In the second part of the paper we introduce hybrid attacks and see that essentially a learner is helpless to form a robust hypothesis when the attacker has access both to

the training as well as to the testing phase.

We are now ready to provide more details for the results of this current work.

Result 1: Exponential Lower Bound on Sample Complexity Suppose the instances of a learning problem come from a metric probability space $(\mathcal{X}, D, \mathsf{d})$ where D is a distribution and d is a metric defining some norm $\|\cdot\|$. Suppose the input instances have norms $\|x\| \approx n$ where n is a parameter related (or is in fact equal) to the data dimension. One natural setting of study for PAC learning is to study attackers that can only perturb x by a *sublinear* amount $o(\|x\|) = o(n)$ (e.g., $\sqrt{n}$).

Our first result is to prove a strong lower bound for the sample complexity of PAC learning in this setting. We prove that for many theoretically natural input spaces of high dimension n (e.g., isotropic Gaussian in dimension n under ℓ_2 perturbations), PAC learning of certain problems under sublinear perturbations of the test instances requires *exponentially* many samples in n, even though the problem in the no-attack setting is PAC learnable using polynomially many samples. This holds e.g., when we want to learn half spaces in dimension n under such distributions (which is possible in the no-attack setting). We note that even though PAC learning is defined for all distributions, proving such lower bound for a specific input distribution D over $\mathcal{X}$ only makes the negative result *stronger*. Our lower bound is in contrast with previously proved results [Attias et al., 2018, Bubeck et al., 2018b, Montasser et al., 2019, Cullina et al., 2018] in which the gap between the sample complexity of the normal and robust learning is only *polynomial*. However, as mentioned before, all these previous results are proved using the *corrupted-input* variant of adversarial risk.

Our result extends to any learning problem where input space $\mathcal{X}$, the metric d and the distribution D defined over them, and the class of concept functions $\mathcal{C}$ have the following two conditions.

1. The inputs $\mathcal{X}$ under the distribution D and small perturbations measured by the metric d forms a *concentrated* metric probability space [Ledoux, 2001, Milman and Schechtman, 1986]. A concentrated space has the property that relatively small events (e.g., of measure 0.1) under small (e.g., smaller than the diameter of the space) perturbations expand to cover almost all measure ≈ 1 of the input space.

2. The set of concept functions $\mathcal{C}$ is complex enough to allow proving lower bounds for the sample complexity for (distribution-dependent) PAC learners in the *no-attack* setting under the *same distribution* D. Distribution-dependent sample complexity lower bounds are known for certain settings [Long, 1995, Balcan and Long, 2013, Sabato et al., 2013], however, we use a more relaxed condition that can be applied to broader settings. In particular, we require that for a sufficiently small ε, there are two concept functions c_1, c_2 that are equal for $1 - \varepsilon$ fraction of inputs sampled from D (see Definition 4.3.3).

Having the above two conditions, our proof proceeds as follows **(I)** We show that the (normal) risk $\mathsf{Risk}(h)$ of a hypothesis produced by *any* learning algorithm with sub-exponential sample complexity cannot be as large as an inverse polynomial over the dimension. **(II)** We then use ideas from the works (e.g., see [Mahloujifar et al., 2018b]) to show that such sufficiently large risk will expand into a large *adversarial* risk of almost all inputs, due to the measure concentration the input space.

Remark 4.1.1 (Approximation error in error-region robust learning)**.** If a learning problem is *realizable* in the no-attack setting, i.e., there is a hypothesis h that has risk zero over the test instances, it means that the same hypothesis h will have adversarial (true) risk zero over the test instances as well, because any perturbed point is still going to be correctly classified. This is in contrast with corrupted-input notion of adversarial risk that even in realizable problems, the smallest corrupted-input (true) adversarial risk could still be large, and even at odds with correctness [Tsipras et al., 2018b]. This means that our results rule out (efficient) PAC learning even in the *agnostic* setting as well, because in the realizable setting there is at least one hypothesis with error-region adversarial risk zero while (as we prove), in some settings learning a model with adversarial risk (under sublinear perturbations) close to zero requires exponentially many samples.

Result 2: Ruling Out PAC Learning under Hybrid Attacks We then study PAC learning under adversarial perturbations that happen during *both* training and testing phases. We formalize *hybrid* attacks in which the final evasion attack is preceded by a poisoning attack [Biggio et al., 2012, Papernot et al., 2016a]. This attack model bears similarities to "trapdoor attacks" [Gu et al., 2017] in which a poisoning phase is involved before the evasion attack, and here we give a formal definition for PAC learning under such attacks. Our definition of hybrid attacks is general and can incorporate any notion of adversarial risk, but our results for hybrid attacks use the *error-region* adversarial risk.

Under hybrid attacks, we show that PAC learning is sometimes *impossible* all together, even though it is possible without such attacks. For example, even if the VC dimension of the concept class is bounded by n, if the adversary is allowed to poison only $1/n^{10}$ fraction of the m training examples, then it can do so in such a way that a subsequent evasion attack could then increase the adversarial risk to ≈ 1. This means that PAC learning is in fact impossible under such hybrid attacks.

We also note that classical results about malicious noise [Valiant, 1985, Kearns and Li, 1993b] and nasty noise [Bshouty et al., 2002] could be interpreted as ruling out PAC learning under poisoning attacks. However, there are two differences: **(I)** The adversary in these previous works needs to change a *constant* fraction of the training examples, while our attacker changes only an *arbitrarily small* inverse polynomial fraction of them. **(II)** Our poisoning attacker only *removes* a fraction of the training set, and hence it does *not* add

any misclassified examples to the pool. Thus the poisoning attack used here is a clean/correct label attack [Mahloujifar et al., 2018a, Shafahi et al., 2018a].

4.2 Adversarially Robust PAC Learning

Notation. By $\widetilde{O}(f(n))$ we refer to the set of all functions of the form $O(f(n)\log(f(n))^{O(1)})$.

Our learning problems $\mathcal{P}_n = (\mathcal{X}_n, \mathcal{Y}_n, \mathcal{C}_n, \mathcal{D}_n, \mathcal{H}_n)$ are usually parameterized by n where n denotes the "data dimension" or (closely) capture the bit length of the instances. Thus, the "efficiency" of the algorithms could depend on n. Even in this case, for simplicity of notation, we might simply write $\mathcal{P} = (\mathcal{X}, \mathcal{Y}, \mathcal{C}, \mathcal{D}, \mathcal{H})$. By default, we will have $\mathcal{C} \subseteq \mathcal{H}$, in which case we call $\mathcal{P}$ *realizable*. This means that for any training set for $c \in \mathcal{C}, D \in \mathcal{D}$, there is a hypothesis that has empirical and true risk zero; though finding such h might be challenging.

Evasion Attacks An evasion attacker A is one that changes the test instance x, denoted as $\widetilde{x} \leftarrow \mathsf{A}(x)$. The behavior and actions taken by A could, in general, depend on the choices of $D \in \mathcal{D}, c \in \mathcal{C}$, and $h \in \mathcal{H}$. As a result, in our notation, we provide A with access to D, c, h by giving them as special inputs to A,[1] denoting the process as $\widetilde{x} \leftarrow \mathsf{A}[D, c, h](x)$. We use calligraphic font $\mathcal{A}$ to denote a *class/set* of attacks. For example, $\mathcal{A}$ could contain all attackers who could change test instance x by at most b perturbations under a metric defined over $\mathcal{X}$.

Poisoning Attacks A poisoning attacker A is one that changes the training sequence as $\widetilde{\mathcal{S}} \leftarrow \mathsf{A}(\mathcal{S})$. Such attacks, in general, might add examples to $\mathcal{S}$, remove examples from $\mathcal{S}$, or do both. The behavior and actions taken by A could, in general, depend on the choices of $D \in \mathcal{D}, c \in \mathcal{C}$ (but not on $h \in \mathcal{H}$, as it is not produced by the learner at the time of the poisoning attack)[2]. As a result, we provide implicit access to D, c by giving them as special inputs to A, denoting the process as $\widetilde{\mathcal{S}} \leftarrow \mathsf{A}[D, c](\mathcal{S})$. We use calligraphic font $\mathcal{A}$ to denote a *class/set* of attacks. For example, $\mathcal{A}$ could contain attacks that change $1/n$ fraction of $\mathcal{S}$ only using clean labels [Mahloujifar et al., 2018b, Shafahi et al., 2018a].

Hybrid Attacks A hybrid attack $\mathsf{A} = (\mathsf{A}_1, \mathsf{A}_2)$ is a two phase attack in which A_1 is a poisoning attacker and A_2 is an evasion attacker. One subtle point is that A_2 is also aware of the internal state of A_1, as they are a pair of coordinating attacks. More formally, A_1 outputs an extra "state" information st which will be

[1] This dependence is information theoretic, and for example, A might want to find $\widetilde{x}$ that is misclassified, in which case its success is defined as $h(\widetilde{x}) \neq c(\widetilde{x})$ which depends on both h, c.

[2] For example, an attack model might require A to choose its perturbed instances still using *correct/clean* labels, in which case the attack is restricted based on the choice of c).

given as an extra input to A_2. As discussed above, A_1 can depend on D, c, and A_2 can depend on D, c, h as defined for evasion and poisoning attacks.

We now define PAC learning under adversarial perturbation attacks. To do so, we need to first define our notion of adversarial risk. We will do so by employing the *error-region* notion adversarial risk as formalized in Diochnos et al. [2018b] adversary aims to misclassify the perturbed instance $\widetilde{x}$.

Definition 4.2.1 (Error-region (adversarial) risk)**.** Suppose A is an evasion adversary and let D, c, h be fixed. The *error-region* (adversarial) risk is defined as follows.

$$\mathsf{AdvRisk}_{\mathsf{A}}(D, c, h) = \Pr_{x \leftarrow D, \widetilde{x} \leftarrow \mathsf{A}[D,c,h](x)}[h(\widetilde{x}) \neq c(\widetilde{x})].$$

For randomized h, the above probability is also over the randomness of h chosen after $\widetilde{x}$ is selected.

We now define PAC learning under hybrid attacks, from which one can derive also the definition of PAC learning under evasion attacks and under poisoning attacks.

Definition 4.2.2 (PAC learning under hybrid attacks)**.** Suppose $\mathcal{P}_n = (\mathcal{X}_n, \mathcal{Y}_n, \mathcal{C}_n, \mathcal{D}_n, \mathcal{H}_n)$ is a realizable classification problem, and suppose $\mathcal{A}$ is a class of hybrid attacks for $\mathcal{P}_n$. $\mathcal{P}_n$ is PAC learnable with sample complexity $\mathsf{m}(\varepsilon, \delta, n)$ under hybrid attacks of $\mathcal{A}$, if there is a learning algorithm L such that for every n, $0 < \varepsilon, \delta < 1, c \in \mathcal{C}, D \in \mathcal{D}$, and $(\mathsf{A}_1, \mathsf{A}_2) \in \mathcal{A}$, if $m = \mathsf{m}(\varepsilon, \delta, n)$, then

$$\Pr_{\substack{\mathcal{S} \leftarrow (D, c(D))^m, \\ (\widetilde{\mathcal{S}}, \mathsf{st}) \leftarrow \mathsf{A}_1[D,c](\mathcal{S}), \\ h \leftarrow L(\widetilde{\mathcal{S}})}} \left[\mathsf{AdvRisk}_{\mathsf{A}_2[D,c,h,\mathsf{st}]}(h, c, D) > \varepsilon\right] \leq \delta.$$

PAC learning under (pure) poisoning attacks or evasion attacks could be derived from Definition 4.2.2 by letting either of A_1 or A_2 be a trivial attack that does no tampering at all.

We also note that one can obtain other definitions of PAC learning under evasion or hybrid attacks in Definition 4.2.2 by using other forms of adversarial risk, e.g., corrupted-input adversarial risk [Feige et al., 2015, 2018, Madry et al., 2018, Schmidt et al., 2018, Attias et al., 2018]

4.3 Lower Bounds for PAC Learning under Evasion and Hybrid Attacks

Before proving our main results, we need to recall the notion of Normal Lévy families, and define a desired and common property of set of concept functions with respect to the distribution of inputs.

Notation. Let $(\mathcal{X}, \mathsf{d})$ be a metric space. For $\mathcal{S} \subseteq \mathcal{X}$, by $\mathsf{d}(x, \mathcal{S}) = \inf\{\mathsf{d}(x, y) \mid y \in \mathcal{S}\}$ we denote the distance of a point x from $\mathcal{S}$. We also let $\mathcal{S}_b = \{y \mid \mathsf{d}(x, y) \leq b, x \in \mathcal{S}\}$ be the *b-expansion* of $\mathcal{S}$. When there is also a measure D defined over the metric space $(\mathcal{X}, \mathsf{d})$, the *concentration function* is defined and denoted as $\boldsymbol{\alpha}(b) = 1 - \inf\{\Pr_D[\mathcal{E}_b] \mid \Pr_D[\mathcal{E}] \geq 1/2\}$.

Definition 4.3.1 (Normal Lévy families)**.** A *family* of metric probability spaces $(\mathcal{X}_n, \mathsf{d}_n, D_n)_{i\in\mathbb{N}}$ with concentration function $\boldsymbol{\alpha}_n(\cdot)$ is called a *normal Lévy family* if there are k_1, k_2, such that[3]

$$\boldsymbol{\alpha}_n(b) \leq k_1 \cdot \mathrm{e}^{-k_2 \cdot b^2/n}$$

Examples. Many natural metric probability spaces are Normal Lévy families. For example, all the following examples under normalized distance (to make the typical norms $\approx n$) are normal Lévy families as stated in Definition 4.3.1: the unit n-sphere with uniform distribution under the Euclidean or geodesic distance, $\mathbb{R}^n$ under Gaussian distribution and Euclidean distance, $\mathbb{R}^n$ under Gaussian distribution and Euclidean distance, the unit n-cube and unit n-ball under the uniform distribution and Euclidean distance, any product distribution of dimension n under the Hamming distance. See [Ledoux, 2001, Giannopoulos and Milman, 2001, Milman and Schechtman, 1986] for more examples.

The following lemma was proved in Mahloujifar et al. [2018b] when Normal Lévy input spaces.

Lemma 4.3.2. *Let the input space of a hypothesis classifier h be a Normal Lévy family $(\mathcal{X}_n, \mathsf{d}_n, D_n)_{i\in\mathbb{N}}$. If the risk of h with respect to the ground truth concept function c is bigger than α, $\mathsf{Risk}(D_n, c, h) \geq \alpha$, and if an adversary A can perturb instances by up to b in metric d_n for*

$$b = \sqrt{n/k_2} \cdot \left(\sqrt{\ln(k_1/\alpha)} + \sqrt{\ln(k_1/\beta)}\right),$$

then the adversarial risk is $\mathsf{AdvRisk}_{\mathsf{A}}(D, h, c) \geq 1 - \beta$.

Definition 4.3.3 (α-close function families)**.** Suppose D is a distribution over $\mathcal{X}$, and let $\mathcal{C}$ be a set of functions from $\mathcal{X}$ to some set $\mathcal{Y}$. We call $\mathcal{C}$ *α-close* with respect to D, if there are $c_1, c_2 \in \mathcal{C}$ such that $\Pr_{x\leftarrow D}[c_1(x) \neq c_2(x)] = \alpha$.

Examples. The set of homogeneous half spaces in $\mathbb{R}^n$ are α-close for all $\alpha \in (0, 1]$ under any of the following natural distributions: uniform over the unit sphere, uniform inside the unit ball, and isotropic Gaussian. This can be proved by picking two half spaces that their disagreement region under the mentioned

[3] Another common formulation of Normal Lévy families uses $\boldsymbol{\alpha}_n(b) \leq k_1 \cdot \mathrm{e}^{-k_2 \cdot b^2 \cdot n}$, but here we scale the distances up by n to achieve "typical norms" to be $\approx n$, which is the dimension.

distributions is exactly α. The set of (monotone, or not necessarily monotone) conjunctions are α-close for $\alpha = 2^{-k}$ for all $k \in \{2, \ldots, n\}$ under the uniform distribution over $\{0,1\}^n$. This can be proved by looking at $c_1 = x_1 \wedge \ldots \wedge x_{k-1}$ and $c_2 = x_1 \wedge \ldots \wedge x_{k-1} \wedge x_k = c_1 \wedge x_k$. Since all the variables that appear in c_1 also appear in c_2, we have that $\Pr_{x \leftarrow \{0,1\}^n}[c_1(x) \neq c_2(x)]$ is equal to $\Pr_{x \leftarrow \{0,1\}^n}[(c_1(x) = 1) \wedge (c_2(x) = 0)]$, and as a consequence this is equal to $2^{-(k-1)} - 2^{-k} = 2^{-k}$.

We now state and prove our main results. Theorem 2.3.2 is stated in the *asymptotic* form considering attack families that attack the problem for sufficiently large index $n \in \mathbb{N}$ of the problem. We describe a quantitative variant afterwards (Lemma 4.3.5).

Theorem 4.3.4 (Limits of adversarially robust PAC learning)**.** *Suppose $\mathcal{P}_n = (\mathcal{X}, \mathcal{Y}, \mathcal{C}, \mathcal{D}, \mathcal{H})$ is a realizable classification problem and that $\mathcal{X}$ is a Normal Lévy Family (Definition 4.3.1) over D and a metric d, and that $\mathcal{C}$ is $\Theta(\alpha)$-close with respect to D for all $\alpha \in [2^{-\Theta(n)}, 1]$. Then, the following hold even for PAC learning with parameters $\varepsilon = 0.9, \delta = 0.49$.*

1. Sample complexity of PAC learning robust fo evasion attacks:

   (a) **Exponential lower bound:** *Any PAC learning algorithm that is robust against* all *attacks with a sublinear tampering $b = o(n)$ budget under the metric d requires exponential sample complexity $m \geq 2^{\Omega(n)}$.*

   (b) **Super-polynomial lower bound:** *PAC learning that is robust against against all tampering attacks with budget $b = \widetilde{O}(\sqrt{n})$, requires at least $m \geq n^{\omega(1)}$ many samples.*

2. Ruling out PAC learning robust to hybrid attacks:

   Suppose the tampering budget of the evasion adversary can be any $b = \widetilde{O}(\sqrt{n})$, and let $\mathcal{B}_\lambda$ be any class of poisoning attacks that can remove $\lambda = \lambda(n)$ fraction of the training examples for an (arbitrary small) inverse polynomial $\lambda(n) \geq 1/\operatorname{poly}(n)$. Let $\mathcal{R}$ be the class of hybrid attacks that first do a poisoning by some $\mathsf{B} \in \mathcal{B}_\lambda$ and then an evasion by some adversary of budget $b = \widetilde{O}(\sqrt{n})$. Then, $\mathcal{P}_n$ is not *PAC learnable (regardless of sample complexity) under hybrid attacks in $\mathcal{R}$.*

As we will see, Part 1a and Part 1b of Theorem 2.3.2 are special cases of the following more quantitative lower bound that might be of independent interest.

Lemma 4.3.5. *For the setting of Theorem 2.3.2, if the tampering budget is $b = \rho \cdot n$, for a fixed function $\rho = \rho(n) = o(1)$, then any PAC learning algorithm for $\mathcal{P}_n$ under evasion attacks of tampering budget $b = b(n)$, even for parameters $\varepsilon = 0.9, \delta = 0.49$ requires sample complexity at least*

$$m(n) \geq 2^{\Omega(\rho^2 \cdot n)}.$$

Examples. Here we list some natural scenarios that fall into the conditions of Theorem 2.3.2. All examples of Normal Lévy families listed after Definition 4.3.1 together with the concept class of half spaces satisfy the conditions of Theorem 2.3.2 and hence cannot be PAC learned using a $\mathrm{poly}(n)$ number of samples. The reason is that one can always find two half spaces whose symmetric difference has measure exactly ε. Moreover, as discussed in examples following Definition 4.3.3, even discrete problems such as learning monotone-conjunctions under the uniform distribution (and Hamming distance as perturbation metric) fall into the conditions of Theorem 2.3.2, for which a lower bound on their sample complexity (or even impossibility) of robust PAC learning could be obtained.

Remark 4.3.6 (Evasion-robust PAC learning in the RAM computing model with real numbers)**.** We remark that if we allow (truly) real numbers represent the concept and hypothesis classes, one can even *rule out* PAC learning (not just lower bounds on sample complexity) under similar perturbations describe in Part 1. Indeed, by inspecting the same proof of Theorem 2.3.2 for Part 1 one can get such results, e.g., for learning half-spaces in dimension n when inputs come from isotropic Gaussian. However, we emphasize that such (seemingly) stronger lower bounds are not realistic, as in real settings, we eventually work with *finite* precision to represent the concept functions (of half spaces). This makes the set of concept functions *finite*, in which case the test error eventually reaches *zero*, using perhaps exponentially many samples. Theorem 2.3.2, however, has the useful feature that it applies even in those settings, as long as the concept functions are rich enough to allow the sufficiently close (but not too close) pairs under the distribution D according to Definition 4.3.3.

In what follows, we will first prove Lemma 4.3.5. We will then use Lemma 4.3.5 to prove Theorem 2.3.2.

Proof of Lemma 4.3.5. Let $m = \mathsf{m}(0.9, 0.49, n)$ be the sample complexity of the (presumed) learner L that achieves (ε, δ)-PAC learning for $\varepsilon = 0.9, \delta = 0.49$. If $m = 2^{\Omega(n)}$ already, we are done, as it is even larger than what Lemma 4.3.5 states, so let $m = 2^{o(n)}$, and we will derive a contradiction. Since the distribution D is fixed, in the discussion below, we simply denote $\mathsf{Risk}(D, h, c)$ as $\mathsf{Risk}(h, c)$.

Recall that, by assumption, for all $\varepsilon \in [2^{-\Theta(n)}, 1]$, there are $c_1, c_2 \in \mathcal{C}$ that are $\Theta(\varepsilon)$-close under the distribution D. Because $m = 2^{o(n)}$, it holds that $1/m \geq \omega(2^{-\Theta(n)})$, and so there are $c_1, c_2 \in \mathcal{C}$ such that for $\Delta(c_1, c_2) = \{x \in \mathcal{X} \mid c_1(x) \neq c_2(x)\}$ we have

$$\Omega\left(\frac{1}{m}\right) \leq \Pr_{x \leftarrow D}[x \in \Delta(c_1, c_2)] \leq \frac{1}{100m}.$$

Now, consider m i.i.d. samples that are given to the learner L as a training set $\mathcal{S}$. With probability at least 0.99 of the sampling of $\mathcal{S}$, all $x \in \mathcal{S}$ would be outside $\Delta(c_1, c_2)$, in which case L would have no way to distinguish c_1 from c_2. So, if we pick $c \leftarrow \{c_1, c_2\}$ at random and pick test instance $x \leftarrow (D \mid \Delta(c_1, c_2))$, the

hypothesis $h = L(\mathcal{S})$ fails with probability at least $0.99/2$. Thus, we can fix the choice of $c \in \{c_1, c_2\}$, such that with probability $0.99/2 > 0.49$ we get a $h \leftarrow L(\mathcal{S})$ where

$$\begin{aligned}\mathsf{Risk}(h, c) = \Pr_{x \leftarrow D}[h(x) \neq c(x)] &\geq \frac{1}{2} \cdot \Pr_{x \leftarrow D}[x \in \Delta(c_1, c_2)] \\ &\geq \Omega\left(\frac{1}{m}\right).\end{aligned}$$

For this fixed c and any such learned hypothesis h with $\mathsf{Risk}(h, c) = \Omega(1)/m$, by Lemma 4.3.2, the adversarial risk reaches $\mathsf{AdvRisk}_{\mathcal{A}_b}(h, c) \geq 0.99$ by an attack $\mathsf{A} \in \mathcal{A}_b$ that has tampering budget:

$$b = O(\sqrt{n}) \cdot \left(\sqrt{\ln(O(m))} + \sqrt{O(1)}\right) \leq t \cdot (\sqrt{n \cdot \ln m})$$

for universal constant t. But, we said at the beginning that the tampering budget of the adversary is $\rho(n) \cdot n$. Therefore, it should be that

$$\rho(n) \cdot n < t \cdot (\sqrt{n \cdot \ln m}),$$

as otherwise the evasion-robust PAC learner is not actually robust as stated. Thus, we get

$$m \geq e^{\rho(n)^2 \cdot n/t} = 2^{\Omega(\rho(n)^2 \cdot n)}$$

which finishes the proof of Lemma 4.3.5. □

We now prove Theorem 2.3.2 using Lemma 4.3.5.

Proof of Theorem 2.3.2. Using Lemma 4.3.5, we will first prove Part 1a, then Part 1b, and then Part 2. Throughout, $\varepsilon = 0.9, \delta = 0.49$ are fixed, so the sample complexity $m = m(n)$ is a function of n.

Proving Part 1a. We claim that PAC learning resisting all $b = o(n)$-tampering attacks requires sample complexity $m \geq 2^{\Omega(n)}$. The reason is that, otherwise, there will be an infinite sequence of values $n_1 < n_2 < \ldots$ for n for which $m = m(n_i) \leq 2^{\gamma(n_i) \cdot (n_i)}$ for $\gamma(n) = o(1)$. However, in that case, if we let $\rho(n) = \gamma(n)^{1/3}$, because $\rho(n) = o(n)$, by Lemma 4.3.5, the sample complexity is

$$m(n_i) \geq 2^{\Omega(\rho(n_i)^2 \cdot n_i)} = \omega\left(2^{\gamma(n_i) \cdot n_i}\right).$$

However, this is a contradiction as we previously assumed $m(n_i) \leq 2^{\gamma(n_i) \cdot (n_i)}$.

Proving Part 1b. Suppose the adversary can tamper instances with budget $b(n) = \kappa(n) \cdot \sqrt{n}$ for $\kappa(n) \in \text{polylog}(n)$. Since we can rewrite $b(n) = \rho(n) \cdot n$ for $\rho(n) = \kappa(n)/\sqrt{n}$, then by Lemma 4.3.5, the sample complexity of L should be at least

$$m(n) \geq 2^{\Omega(\rho(n)^2 \cdot n)} = 2^{\Omega(\kappa(n)^2)}.$$

Therefore, if we choose $\kappa(n) = \log(n)^2$, the sample complexity of L becomes $m \geq n^{\log n} \geq n^{\omega(1)}$.

Proving Part 2. Let be $c_1, c_2 \in \mathcal{C}$ be such that for $\Delta(c_1, c_2) = \{x \in \mathcal{X} \mid c_1(x) \neq c_2(x)\}$ we have

$$\Omega(\lambda) \leq \Pr_{x \leftarrow D(c_1,c_2)}[x \in \Delta(c_1, c_2)] \leq \lambda.$$

Consider a poisoning attacker A_1 that given a data set $\mathcal{S}$, it removes any (x, y) from $\mathcal{S}$ such that $x \in \Delta(c_1, c_2)$. Note that the (expected) number of such examples is $\Pr[x \in \Delta(c_1, c_2)] \leq \lambda$. Let $\widetilde{\mathcal{S}}$ be the modified training set. The learner $L(\widetilde{\mathcal{S}})$ now has now way to distinguish between c_1 and c_2. Thus, like in Lemma 4.3.5, we can fix $c \in \{c_1, c_2\}$, such that $L(\widetilde{\mathcal{S}})$ always produces h where

$$\begin{aligned}\mathsf{Risk}(h, c) = \Pr_{x \leftarrow D}[h(x) \neq c(x)] &\geq \frac{1}{2} \cdot \Pr_{x \leftarrow D}[x \in \Delta(c_1, c_2)] \\ &\geq \Omega(\lambda).\end{aligned}$$

For this fixed c and any such learned hypothesis h with $\mathsf{Risk}(h, c) = \Omega(\lambda)$, by Lemma 4.3.2, the adversarial risk (under attacks) reaches $\mathsf{AdvRisk}_{\mathcal{A}_b}(h, c) \geq 0.99$ by an attack $\mathsf{A} \in \mathcal{A}_b$ that changes test instances x by at most b for

$$b = O(\sqrt{n}) \cdot \left(\sqrt{\ln(O(1/\lambda))} + \sqrt{O(1)}\right) \leq O(\sqrt{n \cdot \ln(1/\lambda)}).$$

Since $\lambda = 1/\operatorname{poly}(n)$, it holds that $b = \widetilde{O}(\sqrt{n})$. □

4.4 Extensions

In this section, we describe some extensions to Theorem 2.3.2 in various directions.

Extension to Randomized Predictors In Theorem 2.3.2, we ruled out PAC learning (or its small sample complexity) even for very large values $\varepsilon = 0.9, \delta = 0.49$. One might argue that proving such lower bound could not be impossible because a trivial hypothesis (for the setting where $\mathcal{Y} = \{0, 1\}$) can achieve $\varepsilon = 0.5$ by outputting random bits. However, this trivial predictor is *randomized*, while Theorem 2.3.2 is proved for

deterministic hypotheses. For the case of randomized hypotheses, one can adjust the proof of Theorem 2.3.2 to get similar lower bounds for $\varepsilon = 0.49, \delta = 0.49$ as follows.

In the proof of Theorem 2.3.2 we first showed that small sample complexity implies the existence of c that with probability > 0.49 it will have an error region with a non-negligible measure. When the hypothesis is randomized, however, we cannot work with the traditional notion of error region, because on every point $x \in \mathcal{X}$, the hypothesis could be wrong $h(x) \neq c(x)$ with some probability in $[0, 1]$. We can, however, work with the relaxed notion of "approximate error" region, defined as $\mathcal{AE}(h, c) = \{x \mid \Pr_h[h(x) \neq c(x)] \geq 1/2\}$, where the probability is over the randomness of h.

In proofs of both Lemma 4.3.5 and Theorem 2.3.2 we deal with two close concept functions c_1, c_2 that are "indistinguishable" for the hypothesis h and then conclude that for each point $x \in \Delta(c_1, c_2)$, h makes a mistake on at least one of c_1, c_2. If h is randomized, we cannot say this anymore, but we can still say that for each such point $x \in \Delta(c_1, c_2)$, for at least one of c_1, c_2, $h(x)$ is wrong with probability at least 0.5. Therefore, we get the same lower bound on the size of the $\mathcal{AE}$ as we got in Lemma 4.3.5 and Theorem 2.3.2. However, expanding the set $\mathcal{AE}$ instead of an actual error-region, implies that the adversarially perturbed points $\widetilde{x}$ that fall into $\mathcal{AE}$ are now misclassified with probability 0.5. Thus, at least 0.99 fraction of inputs can be perturbed into $\mathcal{AE}$ to be misclassified with probability > 0.49.

Lower Bound for PAC Learning of a "Typical" Concept Function Theorem 2.3.2 only proves the *existence* of at least *one* concept function $c \in \mathcal{C}$ for which the (presumed) robust PAC learner will either fail (to PAC learn) or will need large sample complexity. Now, suppose concept functions themselves come from a (natural) distribution and we only want to robustly PAC learn *most* of them. Indeed, we can extend the proof of Theorem 2.3.2 to show that for natural settings, the impossibility result extends to at least *half* of the concept functions, not just a few pathological cases.

To extend Theorem 2.3.2 to the more general "typical" failure over $c \leftarrow \mathcal{C}$ (stated as Claim 4.4.2 below) we need the following definition as an extension to Definition 4.3.3.

Definition 4.4.1 (Uniformly α-close function families)**.** Suppose D is a distribution over $\mathcal{X}$, and let $\mathcal{C}$ be a set of functions from $\mathcal{X}$ to some set $\mathcal{Y}$. We call $\mathcal{C}$ *uniformly α-close* with respect to D, if there is a joint distribution $(\mathbf{c}_1, \mathbf{c}_2)$ where both coordinates are *uniformly* distributed over $\mathcal{C}$, and that for all $(c_1, c_2) \leftarrow (\mathbf{c}_1, \mathbf{c}_2)$, it both holds that $c_1, c_2 \in \mathcal{C}$ and that $\Pr_{x \leftarrow D}[c_1(x) \neq c_2(x)] = \alpha$.

Claim 4.4.2. *In Theorem 2.3.2 and Lemma 4.3.5, make the only change in the setting as follows. The concept class $\mathcal{C}$ now satisfies the stronger condition of being* uniform *α-close with respect to D. Then, the same limitations of PAC learning hold for at least measure half of $c \leftarrow \mathcal{C}$.*

Here we sketch why Claim 4.4.2 holds. The difference is that now, instead of knowing the *existence* of an α-close pair (c_1, c_2), we have *distribution* $(\mathbf{c}_1, \mathbf{c}_2)$ samples from which satisfy the α-close property. Therefore, for all samples $(c_1, c_2) \leftarrow (\mathbf{c}_1, \mathbf{c}_2)$, at least one of c_1 or c_2 is "bad" for the (presumed) PAC learner L (with the same proof before). But, since each of the coordinates in $(\mathbf{c}_1, \mathbf{c}_2)$ is marginally uniform, therefore, at least measure $1/2$ of $c \leftarrow \mathcal{C}$ is bad for L.

Example Consider the uniform measure over homogeneous half spaces in dimension n as the set of concept functions $\mathcal{C}$: choose a point w in the unit sphere and select the half space $\{x \mid \langle x, w\rangle \geq 0\}$. It is easy to see that $\mathcal{C}$ with such measure is uniformly α-close with respect to the isotropic Gaussian distribution (or uniform distribution over the unit sphere). Thus, Claim 4.4.2 applies to this case.

4.5 Conclusion and Open Questions

We examined evasion attacks, where the adversary can perturb instances during test time, as well as hybrid attacks where the adversary can perturb instances during both training and test time. For evasion attacks we gave an exponential lower bound on the sample complexity even when the adversary can perturb instances by an amount of $o(n)$, where n is the data dimension capturing the "typical" norm of an input. For hybrid attacks, PAC learning is ruled out altogether when the adversary can poison a small fraction of the training examples and still perturb the test instance by a sublinear amount $o(n)$ (or even $\widetilde{O}(\sqrt{n})$).

Our result shows a different behavior when it comes to PAC learning for error-region adversarial risk compared to previously used notions of adversarial robustness based on corrupted inputs. In particular, in the error-region variant of adversarial risk, realizable problems stay realizable, as normal risk zero for a hypothesis h also implies (error-region) adversarial risk zero for the same h. This makes our results more striking, as they apply to agnostic learning as well.

Open Questions Our Theorem 2.3.2 relies on a level of tampering to be at least $\widetilde{O}(\sqrt{n})$ to imply the super-polynomial lower bounds. One natural question is to find the exact threshold of perturbations needed that triggers super-polynomial lower bounds on sample complexity.

Another important direction is to study the sample complexity of PAC learning (with concrete parameters ε, δ) for practical distributions such as images or voice. Our lower bounds of this chapter are only proved for theoretically natural distributions that are provably concentrated in high dimension. Mahloujifar et al. [2019e], presents a method for empirically approximating the concentration of such distributions given i.i.d. samples from them.

Finally, we ask if similar results could be proved for corrupted-input adversarial risk. Note that previous work studying learning under corrupted-input adversarial risk [Bubeck et al., 2018b, Cullina et al., 2018, Feige et al., 2018, Attias et al., 2018, Khim and Loh, 2018, Yin et al., 2018b, Montasser et al., 2019] focus on agnostic learning, by aiming to get close to the "best" robust classifier. However, it is not clear how good the best classifier is. It remains open to find out when we can learn robust classifiers (under corrupted-input risk) in which the *total* adversarial risk is small.

Part III

Computational Complexity of Attacks

Chapter 1

Can Adversarially Robust Learning Leverage Computational Hardness?

1.1 Introduction

Is adversarially robust classification possible? As we saw in Parts 1 and 2, recently, started by Gilmer et el. Gilmer et al. [2018b] and followed by Fawzi et al. [2018], Diochnos et al. [2018c], Shafahi et al. [2018b], Mahloujifar et al. [2018b], it was shown that for many natural metric probability spaces of instances (e.g., uniform distribution over $\{0,1\}^n, [0,1]^n$, unit n-sphere, or isotropic Gaussian in dimension n, all with "normalized" Euclidean or Hamming distance) adversarial examples of sublinear perturbations exist for almost all test instances. Indeed, as shown by Mahloujifar, Diochnos, and Mahmoody Mahloujifar et al. [2018b], if the instances are drawn from any "normal Lévy family" Milman and Schechtman [1986] of metric probability spaces (that include all the above-mentioned examples), and if there exists an initial non-negligible risk for the generated hypothesis classifier h, an adversary can perturb an initial instance x into an adversarial one x' that is only $\approx \sqrt{n}$-far (which is sublinear in n) from x and that x' is misclassified.

Is *computationally* robust classification possible? All the above-mentioned sublinear-perturbation attacks of Fawzi et al. [2018], Diochnos et al. [2018c], Shafahi et al. [2018b], Mahloujifar et al. [2018b], in both evasion and poisoning models, were *information theoretic* (i.e., *existential*). Namely, they only show the existence of such adversarial instances for evasion attacks or that they show the existence of such adversarial poisoned data with sublinear perturbations for poisoning attacks. In this chapter, we study the next natural question; can we overcome these information theoretic (existential) lower bounds by relying on the fact that

the adversary is computationally bounded? Namely, can we design solutions that resist *polynomial-time* attacks on the robustness of the learning algorithms? More specifically, the general question studied in our work is as follows.

> *Can we make classifiers robust to* computationally bounded *adversarial perturbations (of sublinear magnitude) that occur during the training or the test phase?*

1.1.1 Summary of Results

In this chapter, we prove strong barriers against basing the robustness of classifiers, in both evasion and poisoning settings, on computational intractability based on a new notion that we introduce name "computational concentration of measure". We state two theorem on the power of polynomial time poisoning and evasion attacks. We prove out Theorems based by assuming that product distributions are computationally concentrated. In next section, we will prove the computational concentration for product distributions. Namely, we show that in settings that computational concentration of measure (i.e., any problem for which the instances are drawn from a product distribution and that their distances are measured by Hamming distance) adversarial examples could be found in *polynomial time*. This result applies to any learning task over these distributions. In the poisoning attacks' setting, we show that for any learning task and any distribution over the labeled instances, if the goal of the adversary is to decrease the confidence of the learner or to increase its error on any particular instance x, it can always do so in polynomial time by only changing $\approx \sqrt{m}$ of the labeled instances and replacing them with yet correctly labeled examples. Below we describe both of these results at a high level.

Theorem 1.1.1 (Informal: polynomial-time evasion attacks)**.** *Let $\mathcal{P}$ be a classification problem in which the test instances are drawn from a product distribution $D \equiv \mathbf{u}_1 \times \cdots \times \mathbf{u}_n$. Suppose c is a concept function (i.e., ground truth) and h is a hypothesis that has a constant $\Omega(1)$ error in predicting c. Then, there is a* polynomial-time *(black-box) adversary that perturbs only $\approx O(\sqrt{n})$ of the* blocks *of the instances and make them misclassified with probability ≈ 1.*

(See Theorem 1.3.3 for the formal version of the following theorem.)

The above theorem covers many natural distributions such as uniform distributions over $\{0,1\}^n$ or $[0,1]^n$ or the isotropic Gaussian of dimension n, so long as the distance measure is Hamming distance. Also, as we will see in Theorem 1.3.3, the initial error necessary for our polynomial-time evasion attack could be as small as $1/\operatorname{poly}(\log n)$ to keep the perturbations $\widetilde{O}(\sqrt{n})$, and even initial error $\omega(\log n/\sqrt{n})$ is enough to keep the perturbations sublinear $o(n)$. Finally, by "black-box" we mean that our attacker only needs oracle access to

the hypothesis h, the ground truth c, and distribution D.[1] This black-box condition is similar to the one defined in previous work of Papernot et al. Papernot et al. [2017], however the notion of black box in some other works (e.g., see Ilyas et al. [2018]) are more relaxed and give some additional data, such as a vector containing probabilities assigned to each label, to the adversary as well.

We also note that, even though *learning* is usually done with respect to a *family* of distributions (e.g., all distributions), working with a particular distribution in our *negative* results make them indeed *stronger*.

We now describe our main result about polynomial-time poisoning attacks. See Theorem 1.3.6 for the formal version of the following theorem.

Theorem 1.1.2 (Informal: polynomial-time poisoning attacks)**.** *Let $\mathcal{P}$ be a classification problem with a deterministic learner L that is given m labeled examples of the form $(x, c(x))$ for a concept function c (determining the ground truth).*

- **Decreasing confidence.** *For any risk threshold $\varepsilon \in [0,1]$, let ρ be the probability that L produces a hypothesis of risk at most ε, referred to as the ε-confidence of L. If ρ is at most $1 - \Omega(1)$, then there is a* polynomial-time *adversary that replaces at most $\approx O(\sqrt{m})$ of the training examples with other correctly classified examples and makes the ε-confidence go down to any constant $O(1) \approx 0$.*
- **Increasing chosen-instance[2] error.** *For any fixed test instance x, if the average error of the hypotheses generated by L over instance x is at least $\Omega(1)$, then there is a* polynomial-time *adversary that replaces at most $\approx O(\sqrt{m})$ of the training examples with other correctly classified examples and increases this average error to any constant ≈ 1.*

Moreover, both attacks above are online *and* black-box*.*

Generalization to arbitrary predicates. More generally, and similarly to the information theoretic attacks of Mahloujifar et al. [2018b], the two parts of Theorem 1.1.2 follow as special cases of a more general result, in which the adversary has a particular efficiently checkable *predicate* in mind defined over the hypothesis (e.g., mislabelling on a particular x or having more than ε risk). We show that the adversary can significantly increase the probability of this bad event if it originally happens with any (arbitrary small) constant probability.

Other features of our poisoning attacks. Similarly to the previous attacks of Mahloujifar et al. Mahloujifar and Mahmoody [2017b], Mahloujifar et al. [2018b,e], both poisoning attacks of Theorem 1.1.2 have the following features.

[1] As mentioned, we need to give our adversary oracle access to a sampler for the instance distribution $\mathbf{x}$ as well, though this distribution is usually polynomial-time samplable.

[2] Poisoning attacks in which the instance is chosen are also called *targeted* Barreno et al. [2006].

1. Our attacks are online; i.e., during the attack, the adversary is only aware of the training examples sampled *so far* when it decides about the next tampering decision. So, these attacks can be launched against online learners in a way that the tampering happens concurrently with the learning process (see Wang and Chaudhuri [2018] for an in-depth study of attacks against online learners). The information theoretic attacks of Mahloujifar et al. [2018b] were "off-line" as the adversary needed the full training sequence before attacking.

2. Our attacks only use *correct labels* for instances that they inject to the training set (see Shafahi et al. [2018a] where attacks of this form are studied in practice).

3. Our attacks are black-box Papernot et al. [2017], as they use the learning algorithm L and concept c as oracles.

1.1.2 Technique: Computational Concentration of Measure in Product Spaces

In a concentrated spaces (e.g., in normal Lévy families) of dimension n, for any sufficiently large set $\mathcal{S}$ (of, say constant measure) the "typical" minimum distance of the space points to $\mathcal{S}$ is sublinear $o(n)$ ($O(\sqrt{n})$ in normal Lévy families). A computational version of this statement shall find such "close" points in $\mathcal{S}$ in polynomial time. The main technical contribution of our work is to prove such computational concentration of measure for any product distribution under the Hamming distance. Namely, we prove the following result about biasing Boolean functions defined over product spaces using polynomial time tampering algorithms. (See Theorem 1.3.1 for a formal variant.)

Theorem 1.1.3 (Informal: computational concentration of products)**.** *Let $\overline{\mathbf{u}} \equiv \mathbf{u}_1 \times \ldots \mathbf{u}_n$ be any product distribution of dimension n and let $f\colon \text{Supp}(\overline{\mathbf{u}}) \mapsto \{0,1\}$ be any Boolean function with expected value $\mathbb{E}[f(\overline{\mathbf{u}})] = \Omega(1)$. Then, there is a* polynomial-time *tampering adversary who only tampers with $O(\sqrt{n})$ of the blocks of a sample $\overline{u} \leftarrow \overline{\mathbf{u}}$ and increases the average of f over the tampered distribution to ≈ 1.*

Once we prove Theorem 1.1.3, we can also use it directly to obtain *evasion* attacks that find adversarial examples, so long as the test instances are drawn from a product distribution and that the distances over the instances are measured by Hamming distance. Indeed, using concentration results (or their stronger forms of isoperimetric inequalities) was the key method used in Section 2 of Part 2 to show the existence of adversarial examples. Thus, our Theorem 1.3.1 is a natural tool to be used in this context as well, as it simply shows that similar (yet not exactly equal) bounds to those proved by the concentration of measure can be achieved algorithmically using polynomial time adversaries.

Relation to approximate nearest neighbor search. We note that computational concentration of measure (e.g., as proved in Theorems 1.1.3 and 1.3.1 for product spaces under Hamming distance) bears similarities to the problem of "approximate nearest neighbor" (ANN) search problem Indyk and Motwani [1998], Andoni and Indyk [2006], Andoni and Razenshteyn [2015], Andoni et al. [2018] in high dimension. Indeed, in the ANN search problem, we are given a set of points $\mathcal{P} \subseteq \mathcal{X}$ where $\mathcal{X}$ is the support set of a metric probability space (of high dimension). We then want to answer approximate near neighbor queries quickly. Namely, for a given $x \in \mathcal{X}$, in case there is a point $y \in \mathcal{P}$ where x and y are "close", the algorithm should return a point y' that is comparably close to x. Despite similarities, (algorithmic proofs of) computational concentration of measure are different in two regards: (1) In our case the set $\mathcal{P}$ could be huge, so it is not even possible to be given as input, but we rather have *implicit* access to $\mathcal{P}$ (e.g., by oracle access). (2) We are not necessarily looking for point by point approximate solutions; we only need the *average* distance of the returned points in $\mathcal{P}$ to be within some (nontrivial) asymptotic bounds.

Proof of Theorem 1.3.1 The proof of this Theorem evolved in two papers. First Mahloujifar and Mahmoody [2018a] studied the concentration of product and achieved sub-optimal bounds for it. Then, in Etesami et al. [2020] we could get almost optimal concentration results for product distributions. Both proofs are inspired by the recent work of Kalai et al. Kalai et al. [2018a] in the context of attacks against coin tossing protocols. Indeed, Kalai et al. [2018a] proved that in any coin tossing protocol in which n parties send a single message each, there is always an adversary who can corrupt up to $\approx \sqrt{n}$ of the players adaptively and almost fix the output to 0 or 1, making progress towards resolving a conjecture of Ben-Or and Linial Ben-Or and Linial [1989]. In next Section we provide a full proof of Theorem 1.3.1. In the rest of this Chapter, we just show how we can get our poisoning and evasion attacks from Theorem 1.3.1.

1.2 Preliminaries

The following definition based on the definitions given in Mahloujifar and Mahmoody [2017b], Mahloujifar et al. [2018a,b].

Definition 1.2.1 (Confidence, chosen-instance error, and their adversarial variants)**.** Let L be a learning algorithm for a classification problem $\mathcal{P} = (\mathcal{X}, \mathcal{Y}, D, \mathcal{C}, \mathcal{H})$, m be the sample complexity of L, and $c \in \mathcal{C}$ be any concept. We define the (adversarial) confidence function and chosen-instance error as follows.

- **Confidence function.** For any error function $\varepsilon = \varepsilon(m)$, the *adversarial confidence* in the presence of a adversary A is defined as

$$\mathsf{Conf}_{\mathsf{A}}(m, c, \varepsilon) = \Pr_{\substack{\mathcal{S} \leftarrow (D, c(D))^m \\ h \leftarrow L(\mathsf{A}(\mathcal{S}))}} [\mathsf{Risk}(h, c) < \varepsilon].$$

  By $\mathsf{Conf}(\cdot)$ we denote the confidence without any attack; namely, $\mathsf{Conf}(\cdot) = \mathsf{Conf}_I(\cdot)$ for the trivial (identity function) adversary I that does not change the training data.

- **Chosen-instance error.** For a fixed test instance $x \in \mathcal{X}$, the *chosen-instance* error (over instance x) in presence of a poisoning adversary A is defined as

$$\mathsf{Err}_{\mathsf{A}}(m, c, x) = \Pr_{\substack{\mathcal{S} \leftarrow (D, c(D))^m \\ h \leftarrow L(\mathsf{A}(\mathcal{S}))}} [h(x) \neq c(x)].$$

  By $\mathsf{Err}(\cdot)$ we denote the chosen-instance error (over x) without any attacks; namely, $\mathsf{Err}(\cdot) = \mathsf{Err}_I(\cdot)$ for the trivial (identity function) adversary I.

1.2.1 Basic Definitions for Tampering Algorithms

Our tampering adversaries follow a close model to that of p-budget adversaries defined in Mahloujifar et al. [2018a]. Such adversaries, given a sequence of blocks, select at most p fraction of the locations in the sequence and change their value. The p-budget model of Mahloujifar et al. [2018a] works in an online setting in which, the adversary should decide for the ith block, only knowing the first $i-1$ blocks. In this chapter, we define both online and offline attacks that work in a closely related budget model in which we only bound the *expected* number of tampered blocks. We find this notion more natural for the robustness of learners.

Definition 1.2.2 (Online and offline tampering)**.** We define the following two tampering attack models.

- **Online attacks.** Let $\overline{\mathbf{u}} \equiv \mathbf{u}_1 \times \cdots \times \mathbf{u}_n$ be an arbitrary product distribution.[3] We call a (potentially randomized and computationally unbounded) algorithm OnTam an *online tampering* algorithm for $\overline{\mathbf{u}}$, if given any $i \in [n]$ and any $u_{\leq i} \in \mathrm{Supp}(\mathbf{u}_1) \times \cdots \times \mathrm{Supp}(\mathbf{u}_i)$, it holds that

$$\Pr_{v_i \leftarrow \mathsf{OnTam}(u_{\leq i})} [v_i \in \mathrm{Supp}(\mathbf{u}_i)] = 1 \ .$$

  Namely, $\mathsf{OnTam}(u_{\leq i})$ outputs (a candidate i^{th} block) v_i in the support set of $\mathbf{u}_i$.[4]

[3] We restrict the case of online attacks to product distribution as they will have simpler notations and that they cover our main applications, however they can be generalized to arbitrary joint distributions as well with a bit more care.

[4] Looking ahead, this restriction makes our attacks stronger in the case of poisoning attacks by always picking correct lables during the attack.

- **Offline attacks.** For an arbitrary joint distribution $\overline{\mathbf{u}} \equiv (\mathbf{u}_1 \ldots, \mathbf{u}_n)$ (that might or might not be a product distribution), we call a (potentially randomized and possibly computationally unbounded) algorithm OffTam an *offline tampering* algorithm for $\overline{\mathbf{u}}$, if given any $\overline{u} \in \mathrm{Supp}(\overline{\mathbf{u}})$, it holds that

$$\Pr_{\overline{v} \leftarrow \mathsf{OnTam}(\overline{u})}[\overline{v} \in \mathrm{Supp}(\overline{\mathbf{u}})] = 1 \ .$$

  Namely, given any $\overline{u} \leftarrow \overline{\mathbf{u}}$, $\mathsf{OnTam}(\overline{u})$ always outputs a vector in $\mathrm{Supp}(\overline{\mathbf{u}})$.

- **Efficiency of attacks.** If $\overline{\mathbf{u}}$ is a joint distribution coming from a *family* of distributions (perhaps based on the index $n \in N$), we call an online or offline tampering algorithm *efficient*, if its running time is $\mathrm{poly}(N)$ where N is the total bit length of any $\overline{u} \in \mathrm{Supp}(\overline{\mathbf{u}})$.

- **Notation for tampered distributions.** For any joint distribution $\overline{\mathbf{u}}$, any $\overline{u} \leftarrow \overline{\mathbf{u}}$, and for any tampering algorithm Tam, by $\langle \overline{u} \parallel \mathsf{Tam} \rangle$ we refer to the distribution obtained by running Tam over $\overline{u}$, and by $\langle \overline{\mathbf{u}} \parallel \mathsf{Tam} \rangle$ we refer to the final distribution by also sampling $\overline{u} \leftarrow \overline{\mathbf{u}}$ at random. More formally,

  - For an offline tampering algorithm OffTam, the distribution $\langle \overline{u} \parallel \mathsf{OffTam} \rangle$ is sampled by simply running OffTam on the whole $\overline{u}$ and obtaining the output $(v_1, \ldots, v_n) \leftarrow \mathsf{OffTam}(u_1, \ldots, u_n)$.
  - For an online tampering algorithm OnTam and input $\overline{u} = (u_1, \ldots, u_n)$ sampled from a *product* distribution $\mathbf{u}_1 \times \ldots \mathbf{u}_n$, we obtain the output $(v_1, \ldots, v_n) \leftarrow \langle \overline{u} \parallel \mathsf{OnTam} \rangle$ *inductively*: for $i \in [i]$, sample $v_i \leftarrow \mathsf{OnTam}(v_1, \ldots, v_{i-1}, u_i)$.[5]

- **Average budget of tampering attacks.** Suppose d is a metric defined over $\mathrm{Supp}(\overline{\mathbf{u}})$. We say an online or offline tampering algorithm Tam has *average budget* (at most) b, if

$$\mathop{\mathbb{E}}_{\substack{\overline{u} \leftarrow \overline{\mathbf{u}}, \\ \overline{v} \leftarrow \langle \overline{u} \parallel \mathsf{Tam} \rangle}} [\mathsf{d}(\overline{u}, \overline{v})] \leq b.$$

  If no metric d is specified, we use Hamming distance over vectors of dimension n.

1.3 Polynomial-time Attacks from Computational Concentration of Products

In this section, we will first formally state our main technical tool, Theorem 1.3.1, that underlies our polynomial-time evasion and poisoning attacks. Namely, we will prove that product distributions are

[5] By limiting our online attackers to product distributions, we can sample the whole sequence of "untampered" values $(u_1, \ldots, u_n)$ at the beginning; otherwise, for general random processes in which the distribution of blocks are correlated, we would need to sample $(u_1, \ldots, u_n)$ and $(v_1, \ldots, v_n)$ *jointly* by sampling u_i *conditioned on* $v_1, \ldots, v_{i-1}$.

"computationally concentrated" under the Hamming distance, in the sense that any subset with constant probability, is "computationally close" to most of the points in the probability space. We will then use this tool to obtain our attacks against learners. We will finally prove our main technical tool.

Theorem 1.3.1 (Computational concentration of product distributions)**.** *Let* $\overline{\mathbf{u}} \equiv \mathbf{u}_1 \times \cdots \times \mathbf{u}_n$ *be any product distribution and* $f\colon \mathrm{Supp}(\overline{\mathbf{u}}) \mapsto \{0,1\}$ *be any Boolean function over* $\overline{\mathbf{u}}$*, and let* $\mu = \mathbb{E}[f(\overline{\mathbf{u}})]$ *be the expected value of* f*. Then, for any* ρ *where* $\mu < \rho < 1$*, there is an* online *tampering algorithm* OnTam *generating the tampering distribution* $\overline{\mathbf{v}} \equiv \langle \overline{\mathbf{u}} \parallel \mathsf{OnTam} \rangle$ *with the following properties.*

1. **Achieved bias.** $\mathbb{E}[f(\overline{\mathbf{v}})] \geq \rho$.
2. **Efficiency.** *Having oracle access to* f *and a sampler for* $\overline{\mathbf{u}}$*,* $\mathsf{OnTam} = \mathsf{OnTam}^{f,\overline{\mathbf{u}}}$ *runs in time* $\mathrm{poly}\left(\frac{n \cdot \ell}{\mu \cdot (1-\rho)}\right)$ *where* ℓ *is the maximum bit length of any* $u_i \in \mathrm{Supp}(\mathbf{u}_i)$ *for any* $i \in [n]$.
3. **Average budget.** $\mathsf{OnTam} = \mathsf{OnTam}^{f,\overline{\mathbf{u}}}$ *uses average budget* $O(\sqrt{n \cdot \ln(1/\mu(1-\rho))})$.

In the rest of this section, we will use Theorem 1.3.1 to prove limitations of robust learning in the presence of polynomial-time poisoning and evasion attackers. We will prove Theorem 1.3.1 in the next section.

1.3.1 Polynomial-time Evasion Attacks

The following definition of robustness against adversarial perturbations of the input is based on the previous definitions used in Gilmer et al. [2018b], Bubeck et al. [2018b], Diochnos et al. [2018c], Mahloujifar et al. [2018b] in which the adversary aims at *misclassification* of the adversarially perturbed instance by trying to push them into the error region.

We define the following definition for a fixed distribution D (as our negative results are for simplicity stated for such cases) but a direct generalization can be obtained for any *family* of distributions over the instances. Moreover, we only give a definition for the "black-box" type of attacks (again because our attacks are black-box) but a more general definition can be given for non-black-box attacks as well.

Definition 1.3.2 (Computational (error-region) evasion robustness)**.** Let $\mathcal{P} = (\mathcal{X}, \mathcal{Y}, D, \mathcal{C}, \mathcal{H}, \mathsf{d})$ be a classification problem. Suppose the components of $\mathcal{P}$ are indexed by $n \in \mathbb{N}$, and let $0 < \mu(n) < \rho(n) \leq 1$ for functions $\mu(n)$ and $\rho(n)$ that for simplicity we denote by μ and ρ. We say that the μ-to-ρ *evasion robustness* of $\mathcal{P}$ is at most $b = b(n)$, if there is a (perhaps computationally unbounded) tampering oracle algorithm $\mathsf{A}^{(\cdot)}$ such that for all $h \in \mathcal{H}, c \in \mathcal{C}$ with error region $\mathcal{E} = \mathcal{E}(h,c), \Pr[D \in \mathcal{E}] \geq \mu$, we have the following.

1. Having oracle access to h, c and a sampler for D, the oracle adversary $\mathsf{A} = \mathsf{A}^{h,c,D}(x)$ reaches adversarial risk to at least ρ (for the choice of c, h). Namely, $\Pr_{x \leftarrow D}[\mathsf{A}^{h,c,D}(x) \in \mathcal{E}] \geq \rho$.

2. The average budget of the adversary $\mathsf{A} = \mathsf{A}^{h,c,D}$ (with oracle access to h, c and a sampler for D) is at most b for samples $x \leftarrow D$ and with respect to metric d.

The μ-to-ρ *computational* evasion robustness of $\mathcal{P}$ is at most $b = b(n)$, if the same statement holds for an *efficient* (i.e., PPT) oracle algorithm A.

Evasion robustness of *problems* vs. that of *learners*. Computational evasion robustness as defined in Definition 1.3.2 directly deals with learning problems regardless of what learning algorithm is used for them. The reason for such a choice is that in this chapter, we prove *negative* results demonstrating the *limitations* of computational robustness. Therefore, limiting the robustness of a learning problems *regardless* of their learner is a stronger result. In particular, any negative result (i.e., showing attackers with small tampering budget) about μ-to-ρ (computational) robustness of a learning problem $\mathcal{P}$, immediately implies that any learning algorithm L for $\mathcal{P}$ that produces hypothesis with risk $\approx \mu$ can always be attacked (efficiently) to reach adversarial risk ρ.

Now we state and prove our main theorem about evasion attacks. Note that the proof of this theorem is identical to the reduction shown in Mahloujifar et al. [2018b]. The difference is that, instead of using original concentration inequalities, we use our new results about *computational* concentration of product measures under hamming distance and obtain attacks that work in polynomial time.

Theorem 1.3.3 (Limits on computational evasion robustness)**.** *Let $\mathcal{P} = (\mathcal{X}, \mathcal{Y}, D, \mathcal{C}, \mathcal{H}, \mathsf{d})$ be a classification problem in which the instances' distribution $D \equiv \mathbf{u}_1 \times \cdots \times \mathbf{u}_n$ is a product distribution of dimension n and d is the Hamming distance over vectors of dimension n. Let $0 < \mu = \mu(n) < \rho = \rho(n) \leq 1$ be functions of n. Then, the μ-to-ρ* computational *evasion robustness of $\mathcal{P}$ is at most*

$$b = O(\sqrt{n \cdot \ln(1/\mu \cdot (1-\rho))}).$$

In particular, if $\mu(n) = \omega(1/\operatorname{poly}(n))$ and $\rho(n) = 1 - 1/\operatorname{poly}(n)$, then then $b = \widetilde{O}(\sqrt{n})$.

Proof. We first define a Boolean function $f\colon : \mathcal{X} \to [0,1]$ as follows:

$$f(x) = \begin{cases} 1 & c(x) \neq h(x), \\ 0 & c(x) = h(x). \end{cases}$$

It is clear that $\mathbb{E}[f(D)] = \Pr[D \in \mathcal{E}] \geq \mu$. Therefore, by using Theorem 1.3.1, we know there is an tampering algorithm $A_{\mu}^{f,D}$ that runs in time $\operatorname{poly}(n \cdot \ell/\mu \cdot (1-\rho))$ and increases the average of f to ρ while using average

budget at most $O(\sqrt{n \cdot \ln(1/\mu(1-\rho))}$. Note that A needs oracle access to $f(\cdot)$ which is computable by oracle access to $h(\cdot)$ and $c(\cdot)$. □

Remark 1.3.4 (Computationally bounded prediction-change evasion attacks)**.** As we mentioned in the introduction, some works studying adversarial examples (e.g., Szegedy et al. [2014], Fawzi et al. [2018]) study robustness by only comparing the prediction of the hypothesis over the adversarial example with its own prediction on the honest example, and so their definition is independent of the ground truth c. (In the terminology of Diochnos et al. [2018c], such attacks are called *prediction-change* attacks.) Here we point out that our biasing attack of Theorem 1.3.1 can be used to prove limits on the robustness against such evasion attacks as well. In particular, in Mahloujifar et al. [2018b], it was shown that using concentration of measure, one can obtain existential (information theoretic) prediction-change attacks (even of the "targeted" form in which the target label is selected). By combining the arguments of Mahloujifar et al. [2018b] and plugging in our computationally bounded attack of Theorem 1.3.1 one can obtain impossibility results for basing the robustness of hypotheses on computational hardness.

1.3.2 Polynomial-time Poisoning Attacks

The following definition formalizes the notion of robustness against computationally bounded poisoning adversaries. Our definition is based on those of Mahloujifar and Mahmoody [2017b], Mahloujifar et al. [2018a] who studied online poisoning attacks and that of Mahloujifar et al. [2018b] who studied offline poisoning attacks.

Definition 1.3.5 (Computational poisoning robustness)**.** Let $\mathcal{P} = (\mathcal{X}, \mathcal{Y}, D, \mathcal{C}, \mathcal{H})$ be a classification problem with a learner L of sample complexity m. Let $0 < \mu = \mu(m) < \rho = \rho(m) \leq 1$ be functions of m.

- **Computational confidence robustness.** For $\varepsilon = \varepsilon(m)$, we say that the ρ-to-μ *ε-confidence robustness* of the learner L is at most $b = b(m)$, if there is a (computationally unbounded) tampering algorithm A such that for all $c \in \mathcal{C}$ for which $\mathsf{Conf}(m, c, \varepsilon) \leq \rho$, the following two conditions hold.

  1. The average budget of $\mathsf{A} = \mathsf{A}^{L,c,D}$ (who has oracle access to L, c and a sampler for D) tampering with the distribution $(D, c(D))^m$ is at most b.

  2. The adversarial confidence for $\varepsilon' = 99 \cdot \varepsilon/100$ is at most $\mathsf{Conf}_{\mathsf{A}}(m, c, \varepsilon') \leq \mu$ when attacked by the oracle adversary $\mathsf{A} = \mathsf{A}^{L,c,D}$.[6]

[6] The computationally-unbounded variant of this definition as used in Mahloujifar et al. [2018b] uses $\varepsilon' = \varepsilon$ instead of $\varepsilon' = 99 \cdot \varepsilon/100$, but as observed by Mahloujifar et al. [2018a], due to the computational bounded nature of our attack we need to have a small gap between ε' and ε.

The ρ-to-μ *computational* ε-confidence robustness of the learner L is at most $b = b(n)$, if the same statement holds for an *efficient* (i.e., PPT) oracle algorithm A .

- **Computational chosen-instance robustness.** For an instance $x \leftarrow D$, we say that the μ-to-ρ *chosen-instance robustness* of the learner L for x is at most $b = b(m)$, if there is a (computationally unbounded) tampering oracle algorithm A (that could depend on x) such that for all $c \in \mathcal{C}$ for which $\mathcal{E}(m, c, x) \geq \mu$, the following two conditions hold.

  1. The average budget of $\mathsf{A} = \mathsf{A}^{L,c,D}$ (who has oracle access to L, c and a sampler for D) tampering with the distribution $(D, c(D))^m$ is at most b.
  2. Adversary $\mathsf{A} = \mathsf{A}^{L,c,D}$ increases the chosen-instance error to $\mathsf{Err}_{\mathsf{A}}(m, c, x) \geq \rho$.

  The μ-to-ρ *computational* chosen-instance robustness of the learner L for instance x is at most $b = b(n)$, if the same thing holds for an *efficient* (i.e., PPT) oracle algorithm A.

Now we state and prove our main theorem about poisoning attacks. Again, the proof of this theorem is identical to the reduction from shown in Mahloujifar et al. [2018b]. The difference is that here we use our new results about *computational* concentration of product measures under hamming distance and get attacks that work in polynomial time. Another difference is that our attacks here are online due the online nature of our martingale attacks on product measures.

Theorem 1.3.6 (Limits on computational poisoning robustness)**.** *Let $\mathcal{P} = (\mathcal{X}, \mathcal{Y}, D, \mathcal{C}, \mathcal{H})$ be a classification problem with a* deterministic polynomial-time *learner L. Let $0 < \mu = \mu(m) < \rho = \rho(m) \leq 1$ be functions of m, where m is the sample complexity of L.*

- **Confidence robustness.** *Let $\varepsilon = \varepsilon(m) \geq 1/\operatorname{poly}(m)$ be the risk threshold defining the confidence function. Then, the ρ-to-μ* computational *ε-confidence robustness of the learner L is at most $b = O(\sqrt{m \cdot \ln(1/\mu(1-\rho))})$.*

- **Chosen-instance robustness.** *For any instance $x \leftarrow D$, the μ-to-ρ* computational *chosen-instance robustness of the learner L for x is at most $b = O(\sqrt{m \cdot \ln(1/(1-\rho)\mu)})$.*

In particular, in both cases above if $\mu(m) = \omega(1/\operatorname{poly}(m))$ and $\rho(m) = 1 - 1/\operatorname{poly}(m)$, then $b = \widetilde{O}(\sqrt{m})$.

Moreover, the polynomial time attacker A bounding the computational poisoning robustness in both cases above has the following features: **(1)** A *is online, and* **(2)** A *is plausible; namely, it never uses any wrong labels in its poisoned training data.*

Proof. We first prove the case of chosen-instance robustness. We define a Boolean function $f\colon \mathcal{X}^m \to [0,1]$ as follows:

$$f_1(x_1,\ldots,x_m) = \begin{cases} 1 & h = L((x_1,c(x_1)),\ldots,(x_n,c(x_m)) \wedge h(x) \neq c(x), \\ 0 & h = L((x_1,c(x_1)),\ldots,(x_n,c(x_m)) \wedge h(x) = c(x). \end{cases}$$

It is clear that $\mathbb{E}[f_1(D^m)] = \mathcal{E}(m,c,x) \geq \mu$. Therefore, by using Theorem 1.3.1, we know there is a PPT tampering Algorithm $A_2^{f_1(\cdot),\mu}$ that runs in time $\text{poly}(m \cdot \ell/(\mu \cdot (1-\rho)))$, and increase the average of f_1 to ρ while using average budget at most $O(\sqrt{m \cdot \ln(1/\mu(1-\rho))})$. Note that A_1 needs oracle access to $f_1(\cdot)$ which is computable by oracle access to the learning algorithm $L(\cdot)$ and concept $c(\cdot)$. Now we prove the case of confidence robustness. Again we define a Boolean function $f_2\colon \mathcal{X}^m \to [0,1]$ as follows:

$$f_2(x_1,\ldots,x_m) = \begin{cases} 1 & h = L((x_1,c(x_1)),\ldots,(x_n,c(x_m)) \wedge \Pr[h(D) \neq c(D)] \geq \varepsilon, \\ 0 & h = L((x_1,c(x_1)),\ldots,(x_n,c(x_m)) \wedge \Pr[h(D) \neq c(D)] < \varepsilon. \end{cases}$$

We have $\mathbb{E}[f_2(D^m)] = 1 - \mathsf{Conf}(m,c,\varepsilon) \geq 1-\rho$. Therefore, by using Theorem 1.3.1, we know there is a PPT tampering Algorithm $A_2^{f_2(\cdot),\mu}$ that runs in time $\text{poly}(m \cdot \ell/(1-\rho) \cdot \mu)$, and increase the average of f_2 to $1-\mu$ while using average budget at most $O(\sqrt{m \cdot \ln(2/\mu(1-\rho))})$. Note that A_2 needs oracle access to $f_2(\cdot)$, which requires the adversary to know the exact error of a hypothesis. Computing the exact error is not possible in polynomial time but using an emprical estimator, the adversary can find an approximation of the error which is sufficient for the attack (See Corollary 3 of Mahloujifar et al. [2018a]). □

Chapter 2

Optimal Bounds for Computational Concentration of Measure

2.1 Introduction

As we saw in previous section, computational concentration of measure is an important tool for robust learning. In this section we try to achieve optimal computational concentration results. Let $(\mathcal{X}, \mathsf{d}, D)$ be a metric probability space in which d is a metric over $\mathcal{X}$, and D is a probability measure over $\mathcal{X}$. The concentration of measure phenomenon Ledoux [2001], Milman and Schechtman [1986] states that many natural metric probability spaces of high dimension are concentrated in the following sense. Any set $\mathcal{S} \subseteq \mathcal{X}$ of "not too small" probability $D(\mathcal{S}) \geq \varepsilon$ is "close" (according to d) to "almost all" points ($1-\delta$ measure according to D).

A well-studied class of concentrated spaces is the set of product spaces in which the measure $D = D_1 \times \ldots D_n$ is a product measure of dimension n, and the metric d is Hamming distance of dimension n; namely, $\mathsf{HD}(\overline{u}, \overline{v}) = |\{i \colon u_i \neq v_i\}|$ for vectors $\overline{u} = (u_1, \ldots, u_n), \overline{v} = (v_1, \ldots, v_n)$. More specifically, it is known, e.g., by results implicit in Amir and Milman [1980], Milman and Schechtman [1986] and explicit in McDiarmid [1989], Talagrand [1995], and weaker versions known as blowing-up lemma proved in Ahlswede et al. [1976], Margulis [1974], Marton [1986], that any such metric probability space is a so-called Normal Lévy family Lévy [1951], Alon and Milman [1985]. Namely, for any $\mathcal{S}$ of probability $D(\mathcal{S}) \geq \varepsilon$, at least $1-\delta$ fraction of the points (under the product measure D) are $O(\sqrt{n \cdot \ln(1/\varepsilon\delta)})$-close in Hamming distance to $\mathcal{S}$. Previous proofs of measure concentration, and in particular those proofs for product spaces are *information theoretic*, and only show the *existence* of a "close" such point $\overline{y} \in \mathcal{S}$ to most of $\overline{x} \leftarrow D$ sampled according to D. Naive sampling of points around $\overline{x}$ will likely *not* fall into $\mathcal{S}$ (see Section 2.6).

Motivated by finding polynomial-time attacks on the "robustness" of machine learning algorithms, recently Mahloujifar and Mahmoody Mahloujifar and Mahmoody [2019a] (See previous section) studied a *computational* variant of the measure concentration in which the mapping from a given point $\overline{x} \leftarrow D$ to its close neighbor $\overline{y} \in \mathcal{S}$ is supposed to be computed by an efficient polynomial-time algorithm $A^{\mathcal{S},D}(\overline{x}) = \overline{y}$ that has oracle access to test membership in $\mathcal{S}$ and a sampling oracle from the measure D.[1] It was shown in Mahloujifar and Mahmoody [2019a] that if $\mathcal{S}$ is large enough, then the measure computationally concentrates around $\mathcal{S}$. In particular, it was shown that if $\Pr[\mathcal{S}] \geq 1/\operatorname{polylog}(n)$, then $A^{\mathcal{S},D}(\overline{x})$ finds $\overline{y}$ with Hamming distance $\widetilde{O}(\sqrt{n})$ from $\overline{x}$, and instead if $\mathcal{S}$ is at least $\Pr[\mathcal{S}] \geq \omega(1/\sqrt{n})$, then A finds $\overline{y}$ with Hamming distance $o(n)$. Consequently, it was left open to prove computational concentration of measure around any smaller sets of "non-negligible" $1/\operatorname{poly}(n)$ probability, e.g., of measure $1/n$.

2.1.1 Summary of Results

In this chapter, we resolve the open question about the computational concentration of measure in product spaces under Hamming distance and prove (tight up to constant) computational concentration for all ranges of initial probabilities $\Pr[\mathcal{S}]$ for the target set $\mathcal{S}$. Namely, we prove the following result matching what information theoretic concentration of product spaces guarantees up to a constant factor, while the mapping is done algorithmically. As we deal with algorithms, without loss of generality, we focus on discrete distributions.[2]

Theorem 2.1.1 (Main result)**.** *There is an algorithm $A^{\mathcal{S},D}_{\varepsilon,\delta}(\cdot)$ called MUCIO (short for "MUltiplicative Conditional Influence Optimizer") that given access to a membership oracle for any set $\mathcal{S}$ and a sampling oracle from any product measure D of dimension n, it achieves the following. If $\Pr[\mathcal{S}] \geq \varepsilon$, given ε and δ, the algorithm $A^{\mathcal{S},D}_{\varepsilon,\delta}(\cdot)$ runs in time $\operatorname{poly}(n/\varepsilon\delta)$, and with probability $\geq 1-\delta$ given a random point $\overline{x} \leftarrow D$, it maps $\overline{x}$ to a point $\overline{y} \in \mathcal{S}$ of bounded Hamming distance $\mathsf{HD}(\overline{x},\overline{y}) \leq O(\sqrt{n \cdot \ln(1/\varepsilon\delta)})$.*

See Theorem 2.3.2 for a more general version of Theorem 2.1.1.

For the special case that $\varepsilon, \delta = 1/\operatorname{poly}(n)$ (implying $\mathcal{S}$ has a non-negligible measure) the algorithm MUCIO of Theorem 2.1.1 achieves its goal in $\operatorname{poly}(n)$ time, while it changes only $\widetilde{O}(\sqrt{n})$ of the coordinates.

Our work can be seen as another example of works in computer science that make previously existential proofs algorithmic. A good example of a similar successful effort is the active line of work started from Moser [2009], Moser and Tardos [2010] that presented algorithmic proofs of Lovász's local lemma, leading to algorithms that efficiently find objects that previously where only shown to exist using Lovász's local lemma.

[1] In case of product measure, oracle access to a sampler from $D = D_1 \times \dots D_n$ is equivalent to having such samplers for all D_i.

[2] Note that even seemingly non-discrete distributions like Gaussian, when used as input to efficient algorithms, are necessarily rounded to limited precision and thus end up being discrete.

The work of Impagliazzo and Kabanets [2010] also approaches measure concentration from an algorithmic perspective, but their goal is to algorithmically find witness for *lack* of concentration.

Extensions

In this chapter we also prove several extensions to our main result in different directions expanding a direct study of computational concentration as an independent direction.

Extension to random processes and coin-tossing attacks.. We prove a more general result than Theorem 2.1.1 in which the perturbed object is a random process. Namely, suppose $\overline{\mathbf{w}} \equiv (\mathbf{w}_1, \ldots, \mathbf{w}_n)$ is a discrete (non-product) random process in which, given the history of blocks $w_1, \ldots, w_{i-1}$, the i^{th} block w_i is sampled from its corresponding random variable $(\mathbf{w}_i \mid w_1, \ldots, w_{i-1})$. Suppose $\Pr_{\overline{w} \leftarrow \overline{\mathbf{w}}}[\overline{w} \in \mathcal{S}] \geq \varepsilon$ for an arbitrary set $\mathcal{S}$. A natural question is: how much can an adversary increase the probability of falling into $\mathcal{S}$, if it is allowed to partially tamper with the online process of sampling $w_1, \ldots, w_n$ up to $K < n$ times? In other words, the adversary has a limited budget of K, and in the i^{th} step, it can use one of its budget, and in exchange it gets to override the originally (honestly) sampled value $w_i \leftarrow (\mathbf{w}_i \mid w_1, \ldots, w_{i-1})$ by a new value. Note that if the adversary does a tampering, the changed value will *substitute* w_i and will affect the way the future blocks of the random process are sampled, e.g., in the next sampling of $w_{i+1} \leftarrow (\mathbf{w}_{i+1} \mid w_1, \ldots, w_i)$.

Our generalized version of Theorem 2.1.1 (stated in Theorem 2.3.2) shows that in the above setting of tampering with random processes, an adversary with budget $O(\sqrt{n \cdot \ln(1/\varepsilon\delta)})$ can indeed change the distribution of the random process and make the resulting tampered sequence end up in $\mathcal{S}$ with probability at least $1 - \delta$, while the adversary also runs in time $\text{poly}(n/\varepsilon\delta)$. Previously, Mahloujifar and Mahmoody [2019a] also showed a similar less tight result for random processes, but their result was limited to the setting that $\mathcal{S}$ is sufficiently large $\Pr[\mathcal{S}] \geq \omega(1/\sqrt{n})$.

The variant of Theorem 2.1.1 for random processes allows us to attack cryptographic coin-tossing protocols Ben-Or and Linial [1989], Cleve and Impagliazzo [1993], Maji et al. [2010], Berman et al. [2014], Haitner and Omri [2014], Kalai et al. [2018b] in which n parties $P_1, \ldots, P_n$ each send a single message during a total of n rounds, and the full transcript $M = (m_1, \ldots, m_n)$ determines a bit b. The goal of an attacker is to corrupt up to K of the parties and bias the bit b towards its favor. Our results show that even if the original bit b had a small probability of being 1, $\Pr_{\text{no-attack}}[b = 1] \geq \varepsilon = 1/\text{poly}(n)$, then a $\text{poly}(n)$-time attacker who can corrupt up to $\widetilde{O}(\sqrt{n})$ parties and change their messages can bias the output bit b all the way up to make it $\Pr_{\text{attack}}[b = 1] \geq 1 - 1/\text{poly}(n)$. The corruption model here was first introduced by Goldwasser, Kalai and

Park Goldwasser et al. [2015b] and is called *strong* adaptive corruption, because the adversary has the option to first see the message m_i before deciding to corrupt (or not corrupt) P_i to change its message m_i (or not). [3]

Weighted Hamming distance.. In another extension to our Theorem 2.1.1 (see Theorem 2.3.2) we allow the Hamming distance to have different costs α_i when changing the i^{th} coordinate for any vector $\overline{\alpha} = (\alpha_1, \ldots, \alpha_n)$ of ℓ_2 norm $\sum_i \alpha_i^2 = n$. In Talagrand's inequality Talagrand [1995], it is proved that even if $\overline{\alpha}_{\overline{x}}$ can completely depend on the original point $\overline{x}$, we still can conclude that most points are "close" to any sufficiently large set $\mathcal{S}$, when the distance from $\overline{x}$ to $\mathcal{S}$ is measured by the $\overline{\alpha}_{\overline{x}}$-weighted Hamming distance. An algorithmic version of Talagrand's inequality, then, shall find a close point $\overline{y} \in \mathcal{S}$ to $\overline{x}$ measured by $\overline{\alpha}_{\overline{x}}$-weighed Hamming distance. Interestingly, our proof allows the coordinate α_i to completely depend on $(x_1, \ldots, x_{i-1})$, but falls short of proving an algorithmic version of Talagrand's inequality, if possible at all.

Reductions and other metric probability spaces.. Motivated by proving computational concentration of measure in other metric probability spaces, as well as designing a machinery for this goal, we define a new model of *algorithmic reductions* between computational concentration of measure in different metric probability spaces. This notion, whose definition has some subtle algorithmic aspects, requires *two* (inverse) polynomial-time mappings one of which is an algorithmic Lipschitz mapping and the other one is an algorithmic coupling connecting the two distributions. As an application, we apply this notion of reduction to obtain computational concentration of measure for high-dimensional Gaussian distributions under the ℓ_1 distance. We prove this exemplary case by revisiting the proof of Bobkov [1997] who proved the *information theoretic* reduction from the concentration of Gaussian distributions under the ℓ_1 distance to that of Hamming cube. We show how the core ideas of Bobkov [1997] could be extended to obtain all the algorithmic components that are needed for a computational variant. Although there are known results on concentration of Gaussian distribution ℓ_1 in information theoretic regime, this is the first time (to the best of our knowledge) that a computational variant of concentration is proved for Gaussian spaces. We envision the same machinery can be applied to more information theoretic results for obtaining new computational variants; we leave doing so for future work. See Theorem 2.4.2 for the formal statement.

Computational concentration around mean.. As measure concentration is usually proved for concentration around mean of a function $f(\cdot)$ when the inputs come from certain distributions, we show how to use our main result of Theorem 2.3.2 to obtain algorithmic concentration results for that setting as well. Namely, at a high level, we show that in certain settings (where concentration is known to follow from those settings) one can algorithmically find the right minimal perturbations to sampled points $\overline{x}$ so that the new perturbed

[3] If each message m_i is a bit, it turns out that our attack can be modified to an attack that is not strong.

point $\overline{x}'$ gives us the average of the concentrated function: $f(\overline{x}') \approx \mathbb{E}_{x \leftarrow D}[f(\overline{x})]$. Sometimes doing so is trivial (e.g., in case of Chernoff bound, when f is simply the addition of i.i.d. sampled Boolean values, as one can greedily change Boolean variables to decrease/increase their summation) but sometimes doing so is not straightforward. In particular, we prove a computational variant of McDiarmid's inequality. Namely, we show how to modify $\sqrt{n}$ coordinates of a vector $\overline{x} \leftarrow D$ sampled from a product distribution D of dimension n, such that $f(\overline{x}')$ gets arbitrary (i.e., $1/\operatorname{poly}(n)$) close to the average $\mu = \mathbb{E}_{\overline{x} \leftarrow D}[f(\overline{x})]$ for a function f that is Lipschitz under Hamming distance. (Note that the Lipschitz property is needed for the McDiarmid inequality as well). See Theorem 2.5.1 for the formal statement.

Lower bounds for simple methods.. We also prove exponential lower bounds on the query complexity of natural, yet restricted, classes of algorithms. Two such classes stand out: One is non-adaptive algorithms where the queries made do not depend on the answer of previous queries. Another, natural class of algorithms are algorithms where all the queried points are at the distance where an acceptable final output may be at that distance. These lower bounds shed light on why perhaps some of the ideas behind our algorithm MUCIO are necessary, and that some simpler more straightforward algorithms are not as efficient.

Polynomial-time biasing attacks against extractors.. At a high level, our biasing attacks on random processes are also related to impossibility results on extracting randomness from blockwise Santha-Vazirani sources Santha and Vazirani [1986], Chor and Goldreich [1988], Beigi et al. [2017], Reingold et al. [2004], Dodis et al. [2004] and specifically the p-tampering and p-resetting attacks of Bentov et al. [2016], Mahloujifar and Mahmoody [2017b], Mahloujifar et al. [2018a]. In those attacks, an attacker might get to tamper each incoming block with an *independent* probability p, and they can achieve a bias of magnitude $O(p)$ (in polynomial time). However, our attackers *can choose* which blocks are the target of their tampering substitutions, but then achieve much stronger bias and almost fixing the output with much smaller $o(n)$ number of tamperings.

Polynomial-time Attacks on Robust Learning

Our results also have implications on (limits) of robust learning, which is also the focus of the work of Mahloujifar and Mahmoody [2019a] where computational concentration of measure was also studied. We refer the reader to Mahloujifar and Mahmoody [2019a] for a more in-depth treatment of the literature and settings for (attacks on) robust learning. For sake of completeness, below we describe the basic setting of such attacks and briefly discuss the implication of our computational concentration results to robust learning attacks.

Suppose L is a (deterministic) learning algorithm, taking as input a training set T consisting of m iid sampled and labeled examples $T = \{x_i, c(x_i)\}_{i\in[m]}$ where $x_i \leftarrow D$ for $i \in [m]$, and that $c(\cdot)$ is a concept function to be learned. Let $h = L(T)$ be the hypothesis that the learner produces based on the training set T. Main attacks against robustness of learners are studied during the training phase or the testing phase of a learning process. We describe the settings and previous work before explaining the implication of our new computational concentration results to those settings.

Poisoning attacks. In a so-called data poisoning attack Barreno et al. [2006], Biggio et al. [2012], which is tightly related to Valiant's malicious noise model Valiant [1985], Kearns and Li [1993b], Bshouty et al. [2002], the adversary only tampers with the training phase and substitutes a small $p < 1$ fraction of the examples in T with other arbitrary examples, leading to a poisoned data set $\widetilde{T}$. The goal of the adversary, in general, is to make $L(\widetilde{T})$ produce a "bad" hypothesis $h \in \widetilde{H}$ (e.g., bad might mean having large risk or making a mistake on a particular test x during the test time) where $\widetilde{H} \subseteq H$ includes the set of all undesired hypothesis. It was shown by Mahloujifar et al. [2018b] that the concentration of measure in product spaces (under Hamming distance) implies that in any such learning process, so long as $\Pr_T[L(T) \in \widetilde{H}] \geq \varepsilon$, then an adversary A who changes $O(\sqrt{m \cdot \ln(1/\varepsilon\delta)})$ of the training examples (and substitute them with still correctly labeled data) can increase the probability of producing a bad hypothesis in $\widetilde{H}$ to $\Pr_{\widetilde{T}\leftarrow\mathsf{A}(T)}[L(\widetilde{T}) \in \widetilde{H}] \geq \delta$. It was left open whether such attack can be made polynomial time, or that perhaps computational intractability can be leveraged to prevent such attacks. The work of Mahloujifar and Mahmoody [2019a] showed how to make such attacks polynomial time, only for the setting where the probability of falling into $\widetilde{H}$ was already not too small, and in particular at least $\omega(1/\sqrt{n})$, and also with looser bounds. Our Theorem 2.1.1 shows how to get such polynomial time evasion attacks for *any non-negligible* probability $\varepsilon \geq 1/\operatorname{poly}(n)$. In fact, as stated in Theorem 2.1.1, our attack's complexity can gracefully adapt to ε. More formally, we prove the following theorem.

Theorem 2.1.2 (Polynomial time poisoning)**.** *Suppose L is a deterministic learner of sample complexity n that takes data set $T = \{x_i, c(x_i)\}_{i\in[m]}$ of size n that is i.i.d. sampled from the distribution $(x, c(x))_{x\leftarrow D}$ and produces a hypothesis $h \in H$. Suppose, we have $\Pr[h \in \widetilde{H}] \geq \varepsilon$ for a set $\widetilde{H} \subseteq H$ of "bad" hypotheses, and where the probability is taken over the randomness of sampling T. Suppose testing membership of $h \in \widetilde{H}$ can be done in time t, and suppose an adversary A is allowed to obtain more samples from the distribution $(x, c(x))$ for $x \leftarrow D$ through a given oracle and that it is allowed to run in time $t \cdot \operatorname{poly}(\frac{n}{\varepsilon\cdot\delta})$. Then, such adversary A can change T into $\widetilde{T}$ that is $O(\sqrt{n \cdot \ln(1/\varepsilon\delta)})$-close in Hamming distance to T such that, we now have $\Pr[L(\widetilde{T}) \in \widetilde{H}] \geq 1 - \delta$, where the probability is over the sampling of T and $\widetilde{T}$.*

The previous attacks of Mahloujifar et al. [2018b], Mahloujifar and Mahmoody [2019a] and our newer attacks of this chapter do not contradict recent exciting works in defending against poisoning attacks Diakonikolas et al. [2016], Lai et al. [2016], Diakonikolas et al. [2018a], Prasad et al. [2018], as those defenses either focus on learning parameters of distributions or, even in the classification setting, they aim to bound the *risk* of the hypothesis, while we increase the probability of a bad Boolean property.[4]

Evasion attacks. In another active line of work, other types of attacks on learners are studied in which the adversary enters the game during the test time. In such so-called *evasion* attacks Biggio et al. [2014], Carlini and Wagner [2017b], Szegedy et al. [2014], Goodfellow et al. [2018] that find "adversarial examples", the goal of the adversary is to perturb the test input x into a "close" input $\widetilde{x}$ under some metric d (perhaps because this small perturbation is imperceptible to humans) in such a way that this tampering makes the hypothesis h make a mistake. In Mahloujifar et al. [2018b], it was also shown that the concentration of measure can potentially lead to inherent evasion attacks, as long as the input metric probability space $(\mathcal{X}, \mathsf{d}, D)$ is concentrated. This holds e.g., if the space is a Normal Lévy family Lévy [1951], Alon and Milman [1985]. The work of Mahloujifar and Mahmoody [2019a] showed the existence of polynomial time evasion attacks with sublinear perturbations for classification tasks in which the input distribution is a n-dimensional product space (e.g., the uniform distribution over the hypercube) under Hamming distance. But their attacks could be applied only when the original risk ε of the hypothesis h is at least $\varepsilon = \omega(1/\sqrt{n})$. However, standard PAC learners (e.g., based on empirical risk minimization) can indeed achieve polynomially small risk $\varepsilon = 1/\operatorname{poly}(m)$ where m is the sample complexity. Our Theorem 2.1.1 shows how to obtain polynomial-time attacks even in the low-risk regime $\varepsilon = 1/\operatorname{poly}(n)$[5] and perturb given samples $x \leftarrow D$ in $\widetilde{O}(\sqrt{n})$ coordinates and make the perturbed adversarial instance $\widetilde{x}$ misclassified with high probability.

Our results of Section 2.4 show that one can also obtain polynomial time evasion attacks for classifiers whose inputs come from metric probability spaces that use metrics other than Hamming distance (e.g., Gaussian under ℓ_1). Using the reductionist approach of Section 2.4 one can perhaps obtain more such results. Our attacks, however, do not rule out the possibility of robust classifiers for specific input distributions such as images or voice that is the subject of recent intense research Szegedy et al. [2014], Carlini and Wagner [2017b], Moosavi-Dezfooli et al. [2016], but they shed light on barriers for robustness in theoretically natural settings. See Bubeck et al. [2018b], Degwekar and Vaikuntanathan [2019] for more discussion on other possible barriers for robust learning.

[4] In fact, the challenge in those works is to obtain polynomial-time *learners* in settings where inefficient robust methods were perhaps known in the robust statistics literature. The focus here, however, is to obtain polynomial-time *attacks*.

[5] Note that in the "high dimensional" setting where input dimension n is huge, we can see the sample complexity m bounded, which implies $\varepsilon \geq 1/\operatorname{poly}(m)$ if $\varepsilon = 1/\operatorname{poly}(n)$.

2.1.2 Technical Overview

In this subsection, we describe the challenges and key ideas behind the proof of Theorem 2.1.1 and some of its extensions. The extension for the concentration around mean (see Section 2.5) follows directly from the main result about concentration around noticeably large sets. Thus, we only focus on explaining ideas behind some other extensions to our result; namely how to obtain new results through carefully defined algorithmic reductions, and proving limits for the power of simple methods for proving computational concentration.

Setting.. (The reader might find the explanations for our notation at the beginning of Section 2.2 useful.) Suppose $\overline{\mathbf{w}} \equiv (\mathbf{w}_1 \times \cdots \times \mathbf{w}_n)$ is a random variable with a product distribution of dimension n.[6] Also, suppose the set $\mathcal{S} \subseteq \text{Supp}(\mathbf{w})$ is denoted by its characteristic function f, where $f(\overline{w}) = 1$ iff $\overline{w} \in \mathcal{S}$. The goal of the tampering algorithm Tam is to change as few as possible of the sampled blocks $\overline{w} = (w_1, \ldots, w_n) \leftarrow \overline{\mathbf{w}}$ making the new vector $\overline{v} = (v_1, \ldots, v_n)$ such that $f(\overline{v}) = 1$ with high probability (over the both steps of sampling $\overline{w}$ and obtaining $\overline{v}$ from it).

Our starting point is the previous attack of Mahloujifar and Mahmoody [2019a] that only proved computational concentration around large sets of measure $\Pr[\mathcal{S}] \geq \omega(1/\sqrt{n})$. The result of Mahloujifar and Mahmoody [2019a], in turn, was built upon techniques developed in the work of Komargodski, Kalai, and Raz Kalai et al. [2018b] that presented an alternative simpler proof for a previously known result of Lichtenstein et al. Lichtenstein et al. [1989]. Below, we first describe the high level ideas behind the approach of Mahloujifar and Mahmoody [2019a], Kalai et al. [2018b], and then we describe why that approach breaks down when $\mathcal{S}$ gets smaller than $1/\sqrt{n}$, and thus fails to obtain the optimal information theoretic bounds for concentration. We then describe our new techniques to bypass this challenge and obtain computational concentration with optimal bounds.

The high-level approach of Mahloujifar and Mahmoody [2019a]. As it turns out, the tampering algorithm of Mahloujifar and Mahmoody [2019a], as well as ours, do not need to know $w_{i+1}, \ldots, w_n$ when deciding to change w_i (into a different $v_i \neq w_i$) or leaving it as is (i.e., $w_i = v_i$). So, a useful notation to use is the partial expected values, capturing the chance of falling into $\mathcal{S}$ (i.e., $f(\overline{w}) = 1$) over the randomness of the remaining blocks.

$$\hat{f}[w_1, \ldots, w_i] = \mathop{\mathbb{E}}_{(w_{i+1}, \ldots, w_n) \leftarrow (\mathbf{w}_{i+1}, \ldots, \mathbf{w}_n)} [f(w_1, \ldots, w_n)].$$

[6] As discussed above, our results extend to random processes as well, when formalized carefully, but for simplicity we focus on the interesting special case of product distributions.

One obvious reason for working with $\hat{f}[\cdot]$ quantities is that they can be *approximated* with arbitrary small $\pm 1/\operatorname{poly}(n)$ additive error. This can be done using the sampling oracle of the distribution of $\overline{\mathbf{w}} \equiv \mathbf{w}_1 \times \cdots \times \mathbf{w}_n$ and the oracle $f(\cdot)$ determining membership in $\mathcal{S}$.

At a high level, the idea behind the attack of Mahloujifar and Mahmoody [2019a] is to change w_i only if this change allows us to increase $\hat{f}[\cdot]$ *additively* by $+\lambda$ for a parameter $\lambda \approx 1/\sqrt{n}$. We first describe this attack, and then explain its challenges against obtaining optimal bounds and how we resolve them.

At a high level, the attack of Mahloujifar and Mahmoody [2019a] tampers with the i^{th} block (i.e., w_i), if just before or just after looking at w_i, we conclude that we can increase $\hat{f}[\cdot]$ by λ.

Construction 2.1.3. (Attack of Mahloujifar and Mahmoody [2019a] oracle $\hat{f}[\cdot]$) Suppose that we are given a prefix $v_{\leq i-1}$ that is finalized, and we are also given a candidate value w_i for the i'th block (supposedly sampled from $\mathbf{w}_i$) and we want to decide to keep it if $v_i = w_i$ or change it if $v_i \neq w_i$. Let $\lambda > 0$ be a parameter of the attack to be chosen later, $v_i^* = \operatorname{argmax}_{y_i} \hat{f}[v_{\leq i-1}, y_i]$ be the choice for i'th block that maximizes $\hat{f}[v_{\leq i}]$, and let $f^* = \hat{f}[v_{\leq i-1}, v_i^*]$.

1. (Case 1) If $f^* \geq \hat{f}[v_{\leq i-1}]+\lambda$, then output $v_i = v_i^*$ (regardless of w_i).
2. (Case 2) Otherwise, if (by looking at w_i) $\hat{f}[v_{\leq i-1}, w_i] \leq \hat{f}[v_{\leq i-1}]-\lambda$, then again output $v_i = v_i^*$.
3. (Case 3) Otherwise, keep the value w_i and output $v_i = w_i$.

Why this attack biases $f(\cdot)$ towards 1? For simplicity, suppose $\Pr[\mathcal{S}] = 1/2$. Suppose we "color" different $i \in [n]$ depending on whether the tampering algorithm changes the i^{th} block w_i or not. If $v_i \neq w_i$ (tampering happened), color i green, denoted by $i \in G$, and otherwise color i red, denoted as $i \in R = [n] \setminus G$. A simple yet extremely useful observation is that we can write $f(\overline{v})$ as the sum of the *changes* in $\hat{f}[v_{\leq i}]$ between consecutive i. Namely, if we let $\hat{g}(v_{\leq i}) = \hat{f}[v_{\leq i}] - \hat{f}[v_{\leq i-1}]$, then

$$\hat{f}[v_{\leq n}] - \hat{f}[\varnothing] = f(\overline{v}) - 1/2 = \sum_{i \in [n]} \hat{g}(v_{\leq i}).$$

This means that we have to study the effect of the green and red coordinates i on how $\hat{g}(v_{\leq i})$ behaves, because that will tell us how the final output bit is determined and distributed.

Construction 2.1.3 is designed so that, whenever i is green, the partial expectation oracle $\hat{f}[v_{\leq i}]$ jumps up at least by λ (This is easy to observe in Case 1 of the attack, but needs a carefull calculation for Case 2.). So, the only damage (leading to falling outside $\mathcal{S}$) could come from the red coordinates and how they change $\hat{f}[v_{\leq i}]$ downwards. Let us now focus on the red coordinates $i \in R$. A simple inspection of Construction 2.1.3 shows that, the change in $\hat{f}[\cdot]$ captured by $\hat{g}(v_{\leq i})$ is bounded in absolute value by λ, and that is the result

of no-tampering for a block. Therefore, the summation of $\hat{g}(v_{\leq i})$ for red coordinates i would cancel out each other and, by the Azuma inequality, the probability that this summation is more than 1 is at most $\exp(-1/(n \cdot \lambda^2)$. So, by choosing $\lambda \ll 1/\sqrt{n}$, the red coordinates cannot control the final bit, as with high probability this summation is less than one. This means that the outcome (whenever the red coordinates do not fix the function) should be 1, because the green coordinates only increase the $\hat{f}[\cdot]$ function.

Why the attack is efficient? The efficiency of the attack follows from its effectiveness and the same argument described above. Namely, whenever the green coordinates are determining the output, it means that their total sum of of $\hat{g}(v_{\leq i})$ is going from a specific number in $[0, +1]$ to 1, and each time they jump up by at least λ, so they cannot be more than n/λ green steps. Since we chose $\lambda = 1/\sqrt{n}$, the efficiency follows as well.

The challenge when $\Pr[\mathcal{S}] = \mathbb{E}[\overline{w}] = \varepsilon$ is too small. The issue with the above approach is that whenever ε is too small (not around 1/2) we need to pick λ much smaller, so that the summation (i.e., the effect of the red coordinates does not make the function reach zero). Simple calculation shows that after the threshold $\varepsilon \approx 1/\sqrt{n}$, the number of tampered (green) blocks would grow too much and eventually become *more* than n. However, note that when we reach n tamperings, it means the attack's efficiency is meaningless.

Our Approach (MUCIO: MUltiplicative Conditional Influence Optimizer)

Main step 1: tampering with *multiplicatively* influential blocks. Our first key idea is to judge whether a block is influential (and thus tamper it) based how much it can change the partial expectations in a *multiplicative* way. (This is related to the notion of a *log-likelihood ratio* in statistics and information theory.) Construction 2.1.4 below describes this simple change. However, as we will see, doing this simple change will have big advantages as well as new challenges to be resolved. We will describe both the advantages and thew new challenges after the construction.

Construction 2.1.4. (*Multiplicative* online tampering using oracle $\hat{f}[\cdot]$) The key difference between this attack and that of Construction 2.1.3 is that here, in order to judge whether tampering with the current i^{th} block is worth it or not, we make the decision based on the *multiplicative* gain (in how $\hat{f}[\cdot]$ changes) that this would give us. Namely, for the same setting of Construction 2.1.3, we do as follows.

1. (Case 1) If $f^* \geq e^{\lambda} \cdot \hat{f}[v_{\leq i-1}]$, then output $v_i = v_i^*$ (regardless of w_i).
2. (Case 2) Otherwise, if $\hat{f}[v_{\leq i-1}, w_i] \leq e^{-\lambda} \cdot \hat{f}[v_{\leq i-1}]$, then output $v_i = v_i^*$.
3. (Case 3) Otherwise, keep the value w_i and output $v_i = w_i$.

Main advantage: the output is fully biased. We first describe what advantages the above change gives us, and then will discuss the remaining challenges. The key insight into why this is a better approach is that the tampering algorithm of Construction 2.1.4 will *always* lead to obtaining $f(\overline{v}) = 1$ at the end (i.e., we always end up in $\mathcal{S}$). In order to see why this is a big difference, notice that if $\hat{f}[w_{\leq 0}] = \varepsilon$ is very small at the beginning and we tamper only based on additive differences (as is done in Construction 2.1.3), there is a possibility that we do *not* tamper with the first block and end up at $\hat{f}[w_{\leq 1}] = 0$. Such a problem does not happen when we decide on tampering based on multiplicative improvement, and every tiny chance of falling into $\mathcal{S}$ is taken advantage of.

Only few tamperings happen. To analyze the number of tamperings that occur in the "idealized" attack of Construction 2.1.4 we keep track of $\ln\left(\hat{f}[v_{\leq i}]/\hat{f}[v_{\leq i-1}]\right)$ as we go. We know that the output of function under the attack is always 1 which means:

$$\sum_{i=1}^{n} \ln\left(\frac{\hat{f}[v_{\leq i}]}{\hat{f}[v_{\leq i-1}]}\right) = \ln\left(\frac{\hat{f}[v_{\leq n}]}{\hat{f}[\varnothing]}\right) = \ln\left(\frac{1}{\hat{f}[\varnothing]}\right).$$

We again categorize the indices i to red and green. Green set indicates the locations that the algorithm tampers with w_i and red is the set of locations that tampering has not happened and $v_i = w_i$. For the red locations, we prove the following inequality that plays a key role in our analysis of the attack. One interpretation of this inequality is that we will now use $\ln(1/\hat{f}[v_{\leq i-1}])$ as a potential function that allows us keep track of, and control, the number of tamperings.

$$\ln(1/\hat{f}[v_{\leq i-1}]) - \mathop{\mathbb{E}}_{v_i \leftarrow \mathbf{v}[v_{\leq i-1}]}[\ln(1/\hat{f}[v_{\leq i}])] \geq -\frac{\lambda^2}{2}.$$

This inequality follows from a Jensen Gap inequality on the natural logarithm function. For green locations, we increase $\ln(\hat{f}[v_{\leq i}])$ whenever we tamper by at least λ. Therefore, the overall effect of green locations on $\sum_{i=1}^{n} \ln(\hat{f}[v_{\leq i}]/\hat{f}[v_{\leq i-1}])$ will be

$$\lambda \cdot \mathbb{E}[\#\text{ of tampering}].$$

Combining these together we get the following:

$$\lambda \cdot \mathbb{E}[\#\text{ of tampering}] - \frac{n \cdot \lambda^2}{2} \leq \ln(1/\varepsilon).$$

Now we can optimize λ to get the best inequality on the expected number of tampering.

New challenge: obtaining good multiplicative approximations when $\hat{f}[\cdot]$ gets too small. Construction 2.1.4 increases the average to 1 (i.e., we always end up in $\mathcal{S}$) with small number of tampering. However we cannot

implement that construction in polynomial time. The problem is that it is hard to instantiate the oracle $\hat{f}[v_{\leq i}]$ polynomial-time when the partial average gets close to 0. To solve this issue, we add a step to the construction that makes the algorithm abort if the partial average goes below some threshold.

Construction 2.1.5. (Online tampering *with abort* TamAb using partial-expectations oracle) This construction is identical to Construction 2.1.4, except that whenever the fixed prefix has a too small partial expectation $\hat{f}[v_{\leq i-1}, w_i]$ (based on a new parameter τ) we will abort. Also, in that case the tampering algorithm does not tamper with any future v_i block either. Namely, we add the following "Case 0" to the previous steps:

- (Case 0) If $\hat{f}[v_{\leq i-1}, w_i] \leq e^{-\tau} \cdot \varepsilon$ abort. If had aborted before, do nothing.

Main step 2: showing that reaching low expectations is unlikely under the attack. To argue that the new construction does not hurt the performance of our algorithm by much, we show that the probability of getting a low $\hat{f}[v_{\leq i}]$ is small because of the way our algorithm works. The idea is that, our algorithm always guarantees that

$$-\lambda \leq \ln\left(\hat{f}[v_{\leq i}]/\hat{f}[v_{\leq i-1}]\right) \leq \lambda.$$

We also show that

$$\mathbb{E}[\ln(\hat{f}[v_{\leq i}]/\hat{f}[v_{\leq i-1}])] \geq -\frac{\lambda^2}{2}.$$

This means that the sequence of $\ln\left(\frac{\hat{f}[v_{\leq i}]}{\hat{f}[v_{\leq i-1}]}\right)$ forms an "approximate" sub-martingale difference sequence. We can use Azuma inequality to show that sum of this sequence will remain bigger than some small threshold, with high probability. After all, we can bound the probability of getting into Case 0 to be very small.

More Computational Concentration Results through Algorithmic Reductions

Here we explain a technical overview of our generic reduction technique. Let $S_1 = (\mathcal{X}_1, \mathsf{d}_1, D_1)$ and $S_2 = (\mathcal{X}_2, \mathsf{d}_2, D_2)$ be two metric probability spaces. In addition, assume we already know some level of computational concentration proved for S_2, and that we want to prove (some level of) computational concentration for S_1 through a reduction. In Section 2.4, we formalize a generic framework to prove such reductions. The main ingredients of such algorithmic reduction are two polynomial time mappings $\mathbf{f}\colon \mathcal{X}_1 \to \mathcal{X}_2$ and $\mathbf{g}\colon \mathcal{X}_2 \to \mathcal{X}_1$ with 3 properties. The first property (roughly speaking) requires that the pushforward of D_1 under $\mathbf{f}$ is an approximation of D_2 and the pushforward of D_2 under $\mathbf{g}$ is an approximation of D_1, namely $\mathbf{f}_*(D_1) \approx D_2$ and $\mathbf{g}_*(D_2) \approx D_1$. This property guarantees that if we sample a point from one space and use the mapping and go to the other space, we get a distribution close to the probability measure of the second

space. (This can be interpreted as an algorithmic coupling.) The second property requires that the mapping $\mathbf{g}$ is Lipschitz. The third property requires that $\mathbf{g}(\mathbf{f}(x))$ is close to x. The idea behind why such reduction (as a collection of these mappings) work is as follows. We are given a point x_1 in S_1 and we want to find a close x_2 such that x_2 falls inside a subset $\mathcal{S}$. To do that we first map x_1 to a point x'_1 in S_2 using $\mathbf{f}$. We know that S_2 is computationally concentrated and we can efficiently find a close x'_2 such that x'_2 falls into an specific subset $\mathcal{S}'$. Then we use $\mathbf{g}$ to go back to a point x_2 in S_1. The second and third properties together guarantee that x_1 and x_2 are close, because x'_1 and x'_2 are close. At the same time, the first condition guarantees that x_2 will hit $\mathcal{S}$ if we select $\mathcal{S}'$ in a careful way. See Theorem 2.4.2 for more details.

We use this general framework to prove computational concentration bounds for Gaussian spaces under ℓ_1 norm. We reduce the computational concentration of Gaussian distribution under ℓ_1 to the computational concentration of the Boolean Hamming cube. For this goal, we show how to build two mappings $\mathbf{f}$ and $\mathbf{g}$ from an n-dimensional Gaussian space to a n^2-dimensional Hamming cube and vice versa, following the footsteps of a reduction by Bobkov [1997] who proved an information theoretic variant of this result. Here we show that the algorithmic ingredients that are necessary, in addition to the ideas already in Bobkov [1997], could indeed be obtained. The main idea behind this mappings is the fact that the number of 1's in a sample from n-dimensional hamming cube approximately forms a Gaussian distribution centered around $\frac{n}{2}$. Therefore, we can map each dimension of the Gauss space to a n-dimensional hamming cube and vice versa. Here we observe that we can use the same idea and build the mappings in a way that achieves the three properties mentioned above. See Section 2.4 for more details.

Lower Bounds for Simple Methods

To prove exponential lower bounds on the query complexity of too-simple algorithms, we consider the half-space $\mathcal{S}$ in the Hamming cube consisting of those points with below-average Hamming weight.

A uniformly random point $\overline{x}$ in the cube, with high probability has Hamming distance $\Omega(\sqrt{n})$ from the set $\mathcal{S}$. Now, if for such a point $\overline{x}$, we hope to find a close point in $\mathcal{S}$ simply by sampling uniformly at random among points close to x, we fail except with exponentially small probability. For only random points with distance $n^{1-o(1)}$ have a significant chance of changing the weight of point $\overline{x}$ by $\Omega(\sqrt{n})$, whereas the information-theoretic bound says there exists a point of distance $O(\sqrt{n})$ that changes the weight by $\Omega(\sqrt{n})$.

To achieve lower bounds for more general classes of algorithms, we use a random half-space instead of a fixed half-space. This gives us exponential lower bounds for non-adaptive attacks as well as attacks that query about $\mathcal{S}$-membership of points outside a ball of size $d = O(\sqrt{n \cdot \ln(1/\varepsilon\delta)})$ even when we are interested in finding a point in the intersection of $\mathcal{S}$ and this ball. Notice that MUCIO avoids this last restriction by

surveying the influence of the first coordinate on the totality of points, while it ends up changing only a small fraction of the coordinates.

2.2 Preliminaries

General notation. We use calligraphic letters (e.g., $\mathcal{X}$) for sets. By default, all distributions and random variables in this chapter are discrete. We use bold letters (e.g., $\mathbf{w}$) to denote random variables that return a sample from a corresponding discrete distribution. By $w \leftarrow \mathbf{w}$ we denote sampling w from the random variable $\mathbf{w}$. By $\mathrm{Supp}(\mathbf{w})$ we denote the support set of $\mathbf{w}$. For an event $\mathcal{S} \subseteq \mathrm{Supp}(\mathbf{w})$, the probability function of $\mathbf{w}$ for $\mathcal{S}$ is denoted as $\Pr[\mathbf{w} \in \mathcal{S}] = \Pr_{w \leftarrow \mathbf{w}}[w \in \mathcal{S}]$. For a randomized algorithm $R(\cdot)$, by $y \leftarrow R(x)$ we denote the randomized execution of R on input x outputting y. By $\mathbf{u} \equiv \mathbf{v}$ we denote that the random variables $\mathbf{u}$ and $\mathbf{v}$ have the same distributions. Unless stated otherwise, we denote vectors by using a bar over a variable. By $\overline{\mathbf{w}} \equiv (\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_n)$ we refer to a sequence of n *jointly sampled* random variables. For a vector $\overline{w} = (w_1 \ldots w_n)$, we use $w_{\leq i}$ to denote the prefix $(w_1, \ldots, w_i)$, and we use the same notation $\mathbf{w}_{\leq i}$ for jointly distributed random variables. For a jointly distributed random variables $(\mathbf{u}, \mathbf{v})$, by $(\mathbf{u} \mid v)$ we denote the conditional distribution $(\mathbf{u} \mid \mathbf{v} = v)$. For a random variable $\mathbf{u}$, by $T^{\mathbf{u}}(\cdot)$ we denote an oracle-aided algorithm $T^{(\cdot)}(\cdot)$ that can query fresh sample from $\mathbf{u}$. By $\mathbf{u} \times \mathbf{v}$ we refer to the product distribution in which $\mathbf{u}$ and $\mathbf{v}$ are sampled independently. For a real-valued random variable $\mathbf{x}$, by $\mathbb{E}[\mathbf{x}]$ we refer to the expected value of $\mathbf{x}$, and by $\mathbb{V}[\mathbf{x}]$ we denote its variance. By $\mathrm{negl}(n)$ we denote some function that is negligible in input n; namely for all $k \in \mathbb{N}$, $\mathrm{negl}(n) \leq O(1/n^k)$.

Notation on random processes and online samplers. Let $\overline{\mathbf{w}} \equiv (\mathbf{w}_1, \ldots, \mathbf{w}_n)$ be a sequence of jointly distributed random variables. We can interpret the distribution of $\overline{\mathbf{w}}$ as a random process in which the i^{th} block w_i is sampled from the marginal distribution $(\mathbf{w}_i \mid w_{\leq i-1})$. For simplicity, we use notation $\mathbf{w}[w_{\leq i-1}] \equiv (\mathbf{w}_i \mid w_{\leq i-1})$ to refer to this marginal conditional distribution. (Note that i is dropped from the distribution's name, relying on the input $w_{\leq i-1}$ that uniquely determines i.) We can interpret $w_{\leq i-1}$ as a "node" in a tree of depth i, and the sampling $w_i \leftarrow \mathbf{w}[w_{\leq i-1}]$ can be seen as the process of sampling the next child according to the distribution of $\mathbf{w}[w_{\leq i-1}]$. Alternatively, describing the distributions of the random variables $\mathbf{w}[w_{\leq i-1}]$ defines the distribution of $\overline{\mathbf{w}}$. For random variable $\overline{\mathbf{w}} \equiv (\mathbf{w}_1, \ldots, \mathbf{w}_n)$ we sometimes refer to the random variable $\mathbf{w}[w_{\leq i-1}]$ as the *online* sampler for $\overline{\mathbf{w}}$, because it returns fresh samples form the next block, given the previously fixed prefix $w_{\leq i-1}$.

Definition 2.2.1 (Online tampering)**.** Let $\overline{\mathbf{w}} \equiv (\mathbf{w}_1, \ldots, \mathbf{w}_n)$ be a sequence of jointly distributed random variables, and let $\mathbf{w}[w_{\leq i-1}]$ be the online sampler for $\overline{\mathbf{w}}$ for all $i \in [n]$ and all $w_{\leq i-1} \in \mathrm{Supp}(\mathbf{w}_{\leq i-1})$. Online tampering algorithms for $\overline{\mathbf{w}}$ and their properties are defined as follows.

- **Online tampering.** We call a (potentially randomized and computationally unbounded) algorithm Tam an *online tampering* algorithm for $\overline{\mathbf{w}}$, if for all $i \in [n]$ and $w_{\leq i} \in \mathrm{Supp}(\mathbf{w}_{\leq i})$, it holds that

$$\Pr[\mathsf{Tam}(w_{\leq i}) \in \mathrm{Supp}(\mathbf{w}[w_{\leq i-1}])] = 1 \ .$$

  Namely, $\mathsf{Tam}(w_{\leq i})$ always outputs a candidate i^{th} block that still falls into $\mathrm{Supp}(\mathbf{w}[w_{\leq i-1}])$.

- **Resulting tampered distribution.** For an online tampering algorithm Tam for $\overline{\mathbf{w}}$, by $(\overline{\mathbf{u}}, \overline{\mathbf{v}}) \equiv \langle \overline{\mathbf{w}} \parallel \mathsf{Tam} \rangle$ we refer to the jointly distributed sequence of random varaibles defined as follows. For $i = 1, 2, \ldots, n$, we first sample $u_i \leftarrow \mathbf{w}[v_{\leq i-1}]$, and then we obtain $v_i \leftarrow \mathsf{Tam}(v_{\leq i-1}, u_i)$ as the (possibly different than u_i) choice of the tampering algorithm Tam for the i^{th} block (that will override u_i). At the end, we output the pair of sequences $(\overline{u} = u_{\leq n}, \overline{v} = v_{\leq n})$ as the sample from $(\overline{\mathbf{u}}, \overline{\mathbf{v}})$.

  Notation. For simplicity, we use $\mathbf{v}[v_{\leq i-1}]$ to denote $(\mathbf{v}_i \mid v_{\leq i-1})$ and use $(\mathbf{w}, \mathbf{v})[v_{\leq i-1}]$ to denote the *jointly* distributed random variables from which (u_i, v_i) are sampled conditioned on the prefix $v_{\leq i-1}$. The notation allows us to use $\mathbf{v}[v_{\leq i-1}], (\mathbf{w}, \mathbf{v})[v_{\leq i-1}]$ similarly to how we use online samplers.[7]

- **Budget of tampering attacks.** Let d be a metric defined over $\mathrm{Supp}(\overline{\mathbf{u}})$ as vectors of dimension n. We say a tampering algorithm Tam has *budget* (at most) b, if

$$\Pr_{(\overline{u},\overline{v}) \leftarrow \langle \overline{\mathbf{w}} \parallel \mathsf{Tam} \rangle} [\mathsf{d}(\overline{u}, \overline{v}) \leq b] = 1.$$

  We say that Tam has *average budget* (at most) b, if the following weaker condition holds

$$\mathop{\mathbb{E}}_{(\overline{u},\overline{v}) \leftarrow \langle \overline{\mathbf{w}} \parallel \mathsf{Tam} \rangle} [\mathsf{d}(\overline{u}, \overline{v})] \leq b.$$

- **Algorithmic efficiency of attacks.** If $\overline{\mathbf{w}} = \overline{\mathbf{w}}_n$ is a member from a *family* defined for all $n \in \mathbb{N}$, we call an online or offline tampering algorithm *efficient*, if its running time is $\mathrm{poly}(N)$ where N is the total bit-length representation of any $\overline{w} \in \mathrm{Supp}(\overline{\mathbf{w}}_n)$.

[7] Note that are *not* defining a similar notation of the form $\mathbf{u}[v_{\leq i-1}]$ for $\overline{\mathbf{u}}$. Firstly, this is not needed as $\mathbf{w}[v_{\leq i-1}]$ already provides a sampler for u_i. Moreover, such notation would be inconsistent with our notation for online samplers for random processes based on joint distributions, because the notation would implicitly interpret $v_{\leq i-1}$ as previous samples from $\mathbf{u}_{\leq i-1}$.

Definition 2.2.2. (Partial expectations) Suppose $f\colon \mathrm{Supp}(\overline{\mathbf{w}}) \mapsto \mathbb{R}$ for $\overline{\mathbf{w}} \equiv (\mathbf{w}_1, \ldots, \mathbf{w}_n)$, $i \in [n]$, and $w_{\leq i} \in \mathrm{Supp}(\mathbf{w}_{\leq i})$. Then (using a small hat) we define the notation $\hat{f}[w_{\leq i}] = \mathbb{E}_{\overline{w} \leftarrow (\overline{\mathbf{w}} | w_{\leq i})}[f(\overline{w})]$ to define the expected value of f for a sample from $\overline{\mathbf{w}}$ given the prefix $w_{\leq i}$. In particular, for $\overline{w} = w_{\leq n}$, we have $\hat{f}[\overline{w}] = f(\overline{w})$, and also $\hat{f}[\varnothing] = \mathbb{E}[f(\overline{\mathbf{w}})]$.

Lemma 2.2.3. *(Hoeffding's lemma) Let* $\mathbf{x}$ *be a random variable such that* $\Pr[a \leq \mathbf{x} \leq b] = 1$ *and* $\mathbb{E}[\mathbf{x}] = 0$. *Then, it holds that* $\mathbb{E}[e^{\mathbf{x}}] \leq e^{(b-a)^2/8}$.

Lemma 2.2.4. *Let* $\mathbf{x}$ *be a random variable where* $\Pr\left[e^{-\lambda} \leq \mathbf{x}\right] = 1$ *and* $\Pr\left[\mathbf{x} \leq e^{\lambda}\right] \geq 1-\delta$ *and* $\Pr[\mathbf{x} \leq c] = 1$. *Then,* $\mathbb{E}[\ln(\mathbf{x})] \geq \ln(\mathbb{E}[\mathbf{x}] - \delta \cdot c) - \lambda^2/2$.

Proof. Let $\mathbb{E}[\min(\ln(\mathbf{x}), \lambda)] = s$. Consider a random variable $\mathbf{y} \equiv \min(\ln(\mathbf{x}), \lambda) - s$. We have $\mathbb{E}[\mathbf{y}] = 0$ and $-\lambda - s \leq \mathbf{y} \leq \lambda - s$. Therefore, by Lemma 2.2.3 we have

$$\mathbb{E}[e^{\mathbf{y}}] \leq e^{\lambda^2/2}.$$

On the other hand, we have $\mathbb{E}[e^{\mathbf{y}}] = \mathbb{E}\left[e^{\min(\ln(\mathbf{x}),\lambda)-s}\right] = \mathbb{E}\left[\min\left(\mathbf{x}, e^{\lambda}\right)\right] \cdot e^{-s}$. Thus, we have $\mathbb{E}\left[\min\left(\mathbf{x}, e^{\lambda}\right)\right] \cdot e^{-s} \leq e^{\lambda^2/2}$ which implies $e^{-s} \leq e^{\lambda^2/2 - \ln\left(\mathbb{E}\left[\min\left(\mathbf{x}, e^{\lambda}\right)\right]\right)}$, and so $s \geq \ln\left(\mathbb{E}\left[\min\left(\mathbf{x}, e^{\lambda}\right)\right]\right) - \lambda^2/2$. Therefore we have, $s \geq \ln(\mathbb{E}[\mathbf{x}] - \delta \cdot c) - \lambda^2/2$. □

Lemma 2.2.5. *Let* $\mathbf{x}$ *be a random variable where* $\Pr[e^{-\lambda} \leq \mathbf{x}] = 1$ *and* $\Pr[\mathbf{x} \leq e^{\lambda}] \geq 1 - \delta$ *and* $\Pr[\mathbf{x} \leq c] = 1$. *Then,* $\mathbb{E}[1/\mathbf{x}] \leq \frac{e^{\lambda^2}}{\mathbb{E}[\mathbf{x}] - \delta \cdot c}$.

Proof. Let $\mathbb{E}[\min(\ln(\mathbf{x}), \lambda)] = s$. Consider a random variable $\mathbf{y} = \min(\ln(\mathbf{x}), \lambda) - s$. Similar to proof of Lemma 2.2.4 we have $s \geq \ln(\mathbb{E}[\mathbf{x}] - \delta \cdot c) - \lambda^2/2$. Now consider another random variable $\mathbf{y}' \equiv -\mathbf{y}$. Again by using Hoeffding Lemma we have $\mathbb{E}[e^{\mathbf{y}'}] \leq e^{\lambda^2/2}$ which means

$$\mathbb{E}[e^{-\min(\ln(\mathbf{x}),\lambda)}] \cdot e^{s} \leq e^{\lambda^2/2}$$

which implies

$$\mathbb{E}[\max(1/\mathbf{x}, e^{-\lambda})] \leq e^{\lambda^2/2} \cdot e^{-s} \leq \frac{e^{\lambda^2}}{\mathbb{E}[\mathbf{x}] - \delta \cdot c}.$$

□

The following lemma is implied by Theorem 3.13 from Mcdiarmid et al. [1998].

Lemma 2.2.6. *(Azuma's inequality for sub-martingales) Let* $\bar{\mathbf{t}} \equiv (\mathbf{t}_1, \ldots, \mathbf{t}_n)$ *be a sequence of* n *jointly distributed random variables such that for all* $i \in [n]$, $\Pr[|\mathbf{t}_i| \leq c_i] \geq 1 - \xi$, *for all* $t_{\leq i-1} \leftarrow \mathbf{t}_{\leq i-1}$, *and that*

$\mathbb{E}[\mathbf{t}_i \mid t_{\leq i-1}] \geq -\gamma_i$. *If* $\gamma = \sum_{i=1}^{n} \gamma_i$, *then we have*

$$\Pr\left[\sum_{i=1}^{n} \mathbf{t}_i \leq -s\right] \leq \mathrm{e}^{\frac{-(s-\gamma)^2}{2\sum_{i=1}^{n} c_i^2}} + n \cdot \xi.$$

2.3 Optimal Computational Concentration for Hamming Distance

In this section, we formally state and prove our main result, which is the computational concentration of measure in any product space under Hamming distance.

Definition 2.3.1. (Weighted Hamming Distance) For $\overline{\alpha} = (\alpha_1, \ldots, \alpha_n) \in \mathbb{R}_+^n$, the $\overline{\alpha}$-weighted Hamming distance between vectors of dimension n is denoted by $\mathsf{HD}_{\overline{\alpha}}(\cdot, \cdot)$ and is defined as

$$\mathsf{HD}_{\overline{\alpha}}(\overline{u}, \overline{v}) = \sum_{i \in [n], u_i \neq v_i} \alpha_i.$$

Theorem 2.3.2. *Let* $(\alpha_1, \ldots, \alpha_n) \in \mathbb{R}^n$ *be such that* $\sum_{i=1}^{n} \alpha_i^2 = n$. *Then, there is a (uniform) oracle-aided randomized algorithm* Tam *such that the following holds. Suppose* $f\colon \mathrm{Supp}(\overline{\mathbf{w}}) \mapsto \{0,1\}$ *is a Boolean function for random variable* $\overline{\mathbf{w}} \equiv (\mathbf{w}_1, \ldots, \mathbf{w}_n)$, *and that* $\Pr[f(\overline{\mathbf{w}}) = 1] = \varepsilon$. *Then, the oracle-aided algorithm* $\mathsf{Tam}^{\mathbf{w}[\cdot], f(\cdot)}(\varepsilon, \delta, \cdot)$ *(also denoted by* Tam *for simplicity) with access to the online sampler* $\mathbf{w}[\cdot]$ *for* $\overline{\mathbf{w}}$ *and* $f(\cdot)$ *as oracles is an online tampering algorithm for* $\overline{\mathbf{w}}$ *and has the following features:*

1. $\Pr[f(\overline{\mathbf{v}}) = 1] \geq 1 - \delta$ *where* $\overline{\mathbf{v}}$ *is the tampered sequence, i.e.,* $\Pr_{(\overline{u},\overline{v}) \leftarrow \langle \overline{\mathbf{w}} \parallel \mathsf{Tam} \rangle}[f(\overline{v}) = 1] \geq 1 - \delta$.
2. Tam*'s tampering budget in* α*-weighed Hamming distance* $\mathsf{HD}_{\overline{\alpha}}$ *is* $O(\sqrt{n \cdot \ln(1/\varepsilon\delta)})$.
3. Tam *runs in time* $\mathrm{poly}(N/\varepsilon\delta)$ *where* N *is the total bit representation of any* $\overline{w} \leftarrow \overline{\mathbf{w}}$.

Remark 2.3.3. (Corollary for product distributions) If the original random variable $\overline{\mathbf{w}} = (\mathbf{w}_1, \ldots, \mathbf{w}_n)$ in Theorem 2.3.2 is a product, $\overline{\mathbf{w}} = (\mathbf{w}_1 \times \cdots \times \mathbf{w}_n)$, then the distribution of the samples $\mathbf{u}$ obtained through $(\overline{u}, \overline{v}) \leftarrow \langle \overline{\mathbf{w}} \parallel \mathsf{Tam} \rangle$ would be identical to that of $\overline{\mathbf{w}}$. Namely, we can simply think of the samples $\overline{u}$ as the original *untampered* vector sampled from $\overline{\mathbf{w}}$, and $\overline{v}$ would be the perturbed vector.

In the rest of this section, we prove Theorem 2.3.2.

2.3.1 Proof Using Promised Approximate Partial Expectation Oracles

The following result works in the model where the approximate partial-expectations oracle $\tilde{f}(\cdot)$ is available to the online tampering algorithm AppTam.

Consider three oracles $\tilde{f}(v_{\le i})$, $m(v_{\le i})$ and $\tilde{f}^*(v_{\le i}) = \tilde{f}(v_{\le i}, m(v_{\le i}))$ with the guarantee that for all $v_{\le i} \in \mathrm{Supp}(\mathbf{w}_{\le i})$ we have 5 conditions:

1. $\left|\ln \tilde{f}(v_{\le i}) - \ln \hat{f}[v_{\le i}]\right| \le \gamma$,
2. $\tilde{f}^*(v_{\le i}) = \tilde{f}(v_{\le i}, m(v_{\le i})) \ge \tilde{f}(v_{\le i})$,
3. $\Pr\left[\tilde{f}(v_{\le i}, \mathbf{w}[v_{\le i}]) \ge \tilde{f}^*(v_{\le i})\right] \le \gamma \cdot \tilde{f}(v_{\le i})$,
4. $0 \le \tilde{f}(v_{\le i}) \le 1$,
5. $\tilde{f}(v_{\le n}) = f(v_{\le n})$.

The first condition states that the approximate partial expectation oracle has a small multiplicative error. The second and third conditions state that $m(v_{\le i-1})$ is a good approximation of some $v*$ that maximized $\tilde{f}(v_{\le i}, v^*)$. Now we construct an algorithm using these oracles.

Construction 2.3.4. (Online tampering using promised *approximate* partial-expectations oracle) Recall that we are given a prefix $v_{\le i}$ that is finalized, and we are also given a candidate value u_i for the i'th block (supposedly sampled from $\mathbf{w}[v_{\le i}]$) and we want to decide to keep $v_i = u_i$ or change it. Let $\lambda > 0$ be a parameter of the attack to be chosen later, $v^*_{i+1} = \tilde{f}^*(v_{\le i})$ and let $\tilde{f}^* = \tilde{f}^*(v_{\le i-1})$ be that maximum.

1. (Case 1) If $\tilde{f}^* \ge e^{\lambda\alpha_i} \cdot \tilde{f}(v_{\le i-1})$, then output $v_i = v^*_i$ (regardless of u_i).
2. (Case 2) Otherwise, if $\tilde{f}(v_{\le i-1}, u_i) \le e^{-\lambda\alpha_i} \cdot \tilde{f}(v_{\le i-1})$, then output $v_i = v^*_i$.
3. (Case 3) Otherwise keep the value u_i and output $v_i = u_i$.

Claim 2.3.5. *(Average case analysis of Construction 2.3.4) Let $\mathbf{k}_i$ be the Boolean random variable that $\mathbf{k}_i = 1$ iff the tampering over the i'th block happens, and let $\mathbf{K}_{\overline{\alpha}} = \sum_{i\in[n]} \alpha_i \cdot \mathbf{k}_i$ capture the resulting $\mathsf{HD}_{\overline{\alpha}}$ distance between the jointly sampled $\overline{u}$ and $\overline{v}$. Also let $\tilde{\varepsilon} = \tilde{f}(\varnothing)$. Then, it holds that*

$$\ln(1/\tilde{\varepsilon}) \ge \mathbb{E}[\mathbf{K}_{\overline{\alpha}}] \cdot \lambda - \lambda^2 n/2 + n \cdot \ln(1 - 3\gamma).$$

Corollary of Claim 2.3.5. By setting $\lambda = \sqrt{2\ln(1/\varepsilon)/n}$ and $\gamma = \frac{1}{6n}$ we obtain $\mathbb{E}[\mathbf{K}_{\overline{\alpha}}] \le \sqrt{2n\ln(1/\varepsilon)} + 1$.

We prove the following stronger statement that implies Claim 2.3.5.

Claim 2.3.6. *Let $v_{\le i-1}$ be fixed. Then,*

$$\ln(1/\tilde{f}(v_{\le i-1})) - \mathop{\mathbb{E}}_{v_i \leftarrow \mathbf{v}[v_{\le i-1}]}\left[\ln\left(1/\tilde{f}(v_{\le i})\right)\right] \ge \Pr[\mathbf{k}_i] \cdot (\alpha_i\lambda) - \frac{\alpha_i^2\lambda^2}{2} + \ln(1 - 3\gamma).$$

Proof. (Proof of Claim 2.3.5 using Claim 2.3.6.) A key property of Construction 2.3.4 is that, because the tampering algorithm does not allow the function reach 0, the final sequence $\overline{v}$ always makes the function 1, namely

$$\Pr[f(\mathbf{v}_{\leq n}) = 1] = 1. \tag{2.1}$$

Using the above equation, Claim 2.3.5 follows from Claim 2.3.6 and linearity of expectation as follows.

$$\begin{aligned}
\ln(1/\tilde{\varepsilon}) &= \ln(1/\tilde{f}(\varnothing)) - \mathbb{E}[\ln(1)] \\
&= \ln(1/\tilde{f}(\varnothing)) - \mathbb{E}[\ln(1/\tilde{f}(\mathbf{v}_{\leq n}))] \\
&= \sum_{i\in[n]} \left[\mathbb{E}[\ln(1/\tilde{f}(\mathbf{v}_{\leq i-1})) - \mathbb{E}[\ln(1/\tilde{f}(\mathbf{v}_{\leq i}))]\right] \\
&\geq \sum_{i\in[n]} \left[\mathbb{E}[\alpha_i \cdot \mathbf{k}_i] \cdot \lambda - \frac{\alpha_i^2 \cdot \lambda^2}{2} + \ln(1-3\gamma)\right] \\
&= \mathbb{E}[\mathbf{K}_{\overline{\alpha}}] \cdot \lambda - \frac{n \cdot \lambda^2}{2} + \ln(1-3\gamma) \cdot n.
\end{aligned}$$

□

Now we prove Claim 2.3.6.

Proof. (Proof of Claim 2.3.6) There are two cases:

- If tampering of Case 1 happens, then we have $\Pr[\mathbf{k}_i = 1] = 1$, and

$$\begin{aligned}
&\ln(1/\tilde{f}(v_{\leq i-1})) - \mathop{\mathbb{E}}_{v_i \leftarrow \mathbf{v}[v_{\leq i-1}]}[\ln(1/\tilde{f}(v_{\leq i}))] \\
&\quad \geq \ln(1/\tilde{f}(v_{\leq i-1})) - \ln(1/\tilde{f}^*) \\
&\quad \geq \lambda\alpha_i
\end{aligned}$$

  Thus, in this case Claim 2.3.6 follows trivially.

- If tampering of Case 1 does not happen, it means that $\tilde{f}^*$ is bounded from above. In the following, we focus on this case and all the probabilities and expectations are conditioned on Case 1 not happening; namely, we have $\tilde{f}^* \leq \tilde{f}(v_{\leq i-1}) \cdot e^{\lambda\cdot\alpha_i}$.

Let $I(v_{\leq i})$ be the indicator function for the set $\left\{v_{\leq i} \colon \tilde{f}(v_{\leq i}) \leq e^{-\lambda\alpha_i} \cdot \tilde{f}(v_{\leq i-1})\right\}$. Now consider the random variable $\mathbf{t}$ for a fixed $v_{\leq i-1}$ as follows

$$\mathbf{t} \equiv \frac{\max\left(e^{-\alpha_i\lambda} \cdot \tilde{f}(v_{\leq i-1}), \tilde{f}(v_{\leq i-1}, \mathbf{w}[v_{\leq i-1}])\right)}{\tilde{f}(v_{\leq i-1})}.$$

We have

$$\Pr_{(u_i,v_i)\leftarrow(\mathbf{w},\mathbf{v})[v_{\leq i-1}]}\left[\tilde{f}(v_{\leq i}) \geq \mathbf{t}\cdot e^{\lambda\alpha_i\cdot I(v_{\leq i-1},u_i)}\right] = 1.$$

This is correct because we are either in Case 2, which means $I(v_{\leq i}) = 1$ and

$$\tilde{f}(v_{\leq i}) = \tilde{f}^* \geq \tilde{f}(v_{\leq i-1}) \geq \tilde{f}(v_{\leq i-1}, u_i)\cdot e^{\lambda\cdot\alpha_i}$$

or we are in Case 3 which means $I(v_{\leq i}) = 0$ and

$$\tilde{f}(v_{\leq i}) = \tilde{f}(v_{\leq i-1}, u_i).$$

Therefore, by linearity of expectation we have

$$\begin{aligned}
&\mathop{\mathbb{E}}_{v_i\leftarrow\mathbf{v}[v_{\leq i-1}]}[\ln(\tilde{f}(v_{\leq i}))]\\
&\geq \mathop{\mathbb{E}}_{u_i\leftarrow\mathbf{w}[v_{\leq i-1}]}\left[\ln\left(\mathbf{t}\cdot e^{\lambda\cdot\alpha_i\cdot I(v_{\leq i-1},u_i)}\right)\right]\\
&= \mathop{\mathbb{E}}_{u_i\leftarrow\mathbf{w}[v_{\leq i-1}]}[\ln(\mathbf{t})] + \lambda\cdot\alpha_i\cdot\mathop{\mathbb{E}}_{u_i\leftarrow\mathbf{w}[v_{\leq i-1}]}[I(v_{\leq i-1},u_i)]\\
&= \mathop{\mathbb{E}}_{u_i\leftarrow\mathbf{w}[v_{\leq i-1}]}[\ln(\mathbf{t})] + \lambda\cdot\alpha_i\cdot\mathbb{E}[\mathbf{k}_i]. \qquad (2.2)
\end{aligned}$$

It holds that

$$\Pr\left[e^{-\lambda\alpha_i} \leq \mathbf{t}\right] = 1. \qquad (2.3)$$

We also know by condition 3 of the $\tilde{f}^*(\cdot)$ oracle that

$$\Pr[\tilde{f}^*(v_{\leq i-1}) \geq \tilde{f}(v_{\leq i-1}, \mathbf{w}[v_{\leq i-1}])] \geq 1 - \gamma\cdot\tilde{f}(v_{\leq i-1})$$

which together with $\tilde{f}^*(v_{\leq i-1}) \leq \tilde{f}(v_{\leq i-1})\cdot e^{\lambda\alpha_i}$ implies

$$\Pr[\mathbf{t} \leq e^{\lambda\cdot\alpha_i}] \leq 1 - \gamma\cdot\tilde{f}(v_{\leq i-1}). \qquad (2.4)$$

We also know that

$$\Pr\left[\mathbf{t} \leq \frac{1}{\tilde{f}(v_{\leq i-1})}\right] = 1. \qquad (2.5)$$

We also have

$$\begin{aligned}\mathbb{E}\left[\mathbf{t}\cdot\tilde{f}(v_{\leq i-1})\right] &= \mathbb{E}\left[\max\left(e^{-\lambda}\cdot\tilde{f}(v_{\leq i-1}),\tilde{f}(v_{\leq i-1},\mathbf{w}[v_{\leq i-1}])\right)\right]\\ &\geq \mathbb{E}\left[\tilde{f}(v_{\leq i-1},\mathbf{w}[v_{\leq i-1}])\right]\\ &\geq \mathbb{E}\left[\hat{f}[v_{\leq i-1},\mathbf{w}[v_{\leq i-1}]]\right]\cdot e^{-\gamma}\\ &= \hat{f}[v_{\leq i-1}]\cdot e^{-\gamma}\\ &\geq \tilde{f}(v_{\leq i-1})\cdot e^{-2\gamma}.\end{aligned}$$

which implies

$$\mathbb{E}[\mathbf{t}]\geq e^{-2\gamma}\geq 1-2\gamma. \tag{2.6}$$

Therefore using 2.3, 2.4, 2.5 and 2.6 and applying Lemma 2.2.4 we get,

$$\begin{aligned}\mathbb{E}[\ln(\mathbf{t})] &\geq \ln\left(\mathbb{E}[\mathbf{t}]-\gamma\cdot\tilde{f}(v_{\leq i-1})\cdot\frac{1}{\tilde{f}(v_{\leq i-1})}\right)-\frac{\alpha_i^2\cdot\lambda^2}{2}\\ &\geq \ln(1-3\gamma)-\frac{\alpha_i^2\cdot\lambda^2}{2}.\end{aligned} \tag{2.7}$$

Combining Equations (2.2) and (2.7), we get

$$\begin{aligned}\mathop{\mathbb{E}}_{v_i\leftarrow\mathbf{v}[v_{\leq i-1}]}\left[\ln(\tilde{f}(v_{\leq i}))\right] &\geq \ln(\tilde{f}(v_{\leq i-1}))\\ &+\lambda\cdot\alpha_i\cdot\mathbb{E}[\mathbf{k}_i]-\frac{\lambda^2\cdot\alpha_i^2}{2}+\ln(1-3\gamma)\end{aligned}$$

which finishes the proof. □

Claim 2.3.7. *(Worst case analysis of Construction 2.3.4) Let $\mathbf{k}_i$ be the Boolean random variable that $\mathbf{k}_i=1$ iff the tampering over the i'th block happens, and let $\mathbf{K}_{\overline{\alpha}}=\sum_{i\in[n]}\alpha_i\cdot\mathbf{k}_i$ capture the resulting $\mathsf{HD}_{\overline{\alpha}}$ distance between the jointly sampled $\overline{u}$ and $\overline{v}$. Also let $\tilde{\varepsilon}=\tilde{f}(\varnothing)$. Then, it holds that*

$$\Pr[\mathbf{K}\geq k]\leq\frac{e^{(\sum_{i=1}^n\alpha_i^2)\lambda^2-k\lambda}}{\tilde{\varepsilon}\cdot(1-2\gamma)^n}.$$

Proof. We prove this claim by induction on n. Let $A(n,k,\tilde{\varepsilon})$ be a function that indicates the maximum probability of using more than k budget, over all random processes with boolean outcome of length n, and

average $\tilde{\varepsilon}$. We want to inductively show that

$$A(n,k,\tilde{\varepsilon}) \leq \frac{e^{(\sum_{i=1}^{n}\alpha_i^2)\cdot\lambda^2-k\lambda}}{\tilde{\varepsilon}\cdot(1-2\gamma)^n}.$$

Consider different cases that might happen during the tampering of first block. If we tamper on first block through Case I, we have

$$\Pr[\mathbf{K}\geq k] \leq A(n-1,k-\alpha_1,\tilde{f}^*(\varnothing))$$

And by induction hypothesis we have

$$\begin{aligned}
A(n-1,k-\alpha_1,\tilde{f}^*(\varnothing)) &\leq \frac{e^{(\sum_{i=2}^{n}\alpha_i^2)\lambda^2-k\lambda+\lambda\cdot\alpha_i}}{\tilde{f}^*(\varnothing)\cdot(1-2\gamma)^n}\\
&\leq \frac{e^{(\sum_{i=2}^{n}\alpha_i^2)\lambda^2-k\lambda+\lambda\cdot\alpha_i}}{e^{\lambda\alpha_1}\cdot\tilde{\varepsilon}\cdot(1-2\gamma)^n}\\
&\leq \frac{e^{(\sum_{i=1}^{n}\alpha_i^2)\cdot\lambda^2-k\lambda}}{\tilde{\varepsilon}\cdot(1-2\gamma)^n}.
\end{aligned}$$

So the induction goes through for Case 1. Consider the random variable $\mathbf{t}$ for a fixed $v_{\leq i-1}$ as follows

$$\mathbf{t} \equiv \max(e^{-\lambda\cdot\alpha_i}\cdot\tilde{\varepsilon},\tilde{f}(u_{\leq 1})).$$

If we are not in Case 1, then we have,

$$\begin{aligned}
\Pr[\mathbf{K}\geq k] &= \Pr[\mathbf{K}\geq k \mid \text{ Case 3}]\cdot\Pr[\text{Case 3}]\\
&\quad+\Pr[\mathbf{K}\geq k \mid \text{ Case 2}]\cdot\Pr[\text{Case 2}]\\
&\leq \mathbb{E}\left[A\left(n-1,k,\tilde{f}(u_{\leq 1})\right)\mid \text{ Case 3}\right]\cdot\Pr[\text{Case 3}]\\
&\quad+\mathbb{E}\left[A\left(n-1,k-\alpha_1,\tilde{f}^*(\varnothing)\right)\mid \text{ Case 2}\right]\cdot\Pr[\text{Case 2}]\\
&\leq \mathbb{E}\left[\frac{e^{(\sum_{i=2}^{n}\alpha_i^2)\cdot\lambda^2-k\lambda}}{\tilde{f}(u_{\leq 1})-2(n-1)\gamma}\mid \text{ Case 3}\right]\cdot\Pr[\text{Case 3}]\\
&\quad+\mathbb{E}\left[\frac{e^{(\sum_{i=2}^{n}\alpha_i^2)\cdot\lambda^2-k\lambda+\lambda\cdot\alpha_1}}{\tilde{f}^*(\varnothing)\cdot(1-2\gamma)^{n-1}}\mid \text{ Case 2}\right]\cdot\Pr[\text{Case 2}]\\
&\leq \mathbb{E}\left[\frac{e^{(\sum_{i=2}^{n}\alpha_i^2)\cdot\lambda^2-k\lambda}}{\tilde{f}(u_{\leq 1})\cdot(1-2\gamma)^{n-1}}\mid \text{ Case 3}\right]\cdot\Pr[\text{Case 3}]\\
&\quad+\mathbb{E}\left[\frac{e^{(\sum_{i=2}^{n}\alpha_i^2)\cdot\lambda^2-k\lambda+\lambda\cdot\alpha_1}}{\mathbf{t}\cdot e^{\lambda\cdot\alpha_i}\cdot(1-2\gamma)^{n-1}}\mid \text{ Case 2}\right]\cdot\Pr[\text{Case 2}]\\
&\leq e^{(\sum_{i=2}^{n}\alpha_i^2)\cdot\lambda^2-k\lambda}\cdot\mathbb{E}\left[\frac{1}{\mathbf{t}\cdot(1-2\gamma)^{n-1}}\right] \qquad (2.8)
\end{aligned}$$

We know that $\mathbb{E}[\max(e^{-\lambda\cdot\alpha_i}\cdot\tilde{\varepsilon},\tilde{f}(u_{\leq 1}))] \geq \mathbb{E}[\tilde{f}(u_{\leq 1})] \geq \tilde{\varepsilon}\cdot e^{-\gamma}$. Now we can use Lemma 2.2.5 and get

$$\mathbb{E}[\frac{1}{\max(e^{-\lambda\cdot\alpha_1}\cdot\tilde{\varepsilon},\tilde{f}(u_{\leq 1}))}] \leq \frac{e^{\alpha_1^2\cdot\lambda^2}}{\tilde{\varepsilon}\cdot(e^{-\gamma}-\gamma)} \leq \frac{e^{\alpha_1^2\cdot\lambda^2}}{\tilde{\varepsilon}\cdot(1-2\gamma)} \tag{2.9}$$

Combining Equations 2.8 and 2.9 we get,

$$\Pr[\mathbf{K}\geq k] \leq \frac{e^{(\sum_{i=1}^n \alpha_i^2)\lambda^2-k\lambda}}{\tilde{\varepsilon}(1-2\gamma)^n}$$

which finishes the proof. □

Tampering with Abort

The Construction 2.3.4 achieves average close to 1 with small number of tampering. However we cannot implement that construction it in polynomial time. The problem is that it is hard to instantiate the oracle $\tilde{f}(\cdot)$ and $\tilde{f}^*(\cdot)$ in polynomial time when the partial average gets close to 0. Following we add a step to our construction to address this issue. Then we will show that this additional step will not hurt the performance of the algorithm by much.

Construction 2.3.8. (Online tampering *with abort* $\mathsf{AppTamAb}$ using promised approximate partial-expectations oracle) This construction is identical to Construction 2.3.4, except that whenever the fixed prefix has a too small approximate partial expectation $\tilde{f}(v_{\leq i-1},u_i)$ (based on a parameter τ) we will abort. Also, in that case the tampering algorithm does not tamper with any future v_i block either. Namely, we add the following "Case 0" to the previous steps:

- (Case 0) If $\tilde{f}(v_{\leq i-1},u_i) \leq e^{-\tau}\cdot\tilde{\varepsilon}$ abort ($\tilde{\varepsilon}=\tilde{f}(\varnothing)$). If had aborted before, do nothing.

Average and worst case analysis of Construction 2.3.8. The average number of tampering of Construction 2.3.8 is trivially less than average number of tampering of Construction 2.3.4. Therefore, the same bound of Claim 2.3.5 still applies to Construction 2.3.8 as well. Also, the probability of number of tampering going beyond some threshold does not increase compared to Construction 2.3.4 which means the same bound of Claim 2.3.7 hold here.

Claim 2.3.9. *The probability of ever aborting during sampling* $(\overline{u},\overline{v}) \leftarrow \langle \overline{\mathbf{w}} \parallel \mathsf{TamAb}\rangle$ *is at most* $n\cdot e^{-\frac{(\tau-n\cdot\lambda^2/2)^2}{2\cdot n\cdot\lambda^2}}$. *As a result, we also have*

$$\mathop{\mathbb{E}}_{(\overline{u},\overline{v})\leftarrow\langle\overline{\mathbf{w}}\parallel\mathsf{TamAb}\rangle}[f(\overline{v})] \geq 1-n\cdot e^{-\frac{(\tau-n\cdot\lambda^2/2)^2}{2\cdot n\cdot\lambda^2}} - n^2\gamma.$$

Proof. For proof please see full version. □

2.3.2 Putting Things Together

In this subsection we show how to instantiate parameters of Construction 2.3.8 so that we can get polynomial time attack. We first show how to instantiate the oracles. To compute oracle $\tilde{f}(v_{\leq i})$, we sample $\frac{8}{\gamma^3 \cdot e^{-\tau} \cdot \tilde{\varepsilon}}$ random continuation and take the average over all of them. By Hoeffding inequality, if $\hat{f}[v_{\leq i}] \geq e^{-\tau} \cdot \tilde{\varepsilon}$ we get the following:

$$\Pr[|\ln(\tilde{f}(v_{\leq i})) - \ln(\hat{f}[v_{\leq i}])| \geq \gamma] \leq \gamma.$$

For $m(v_{\leq i})$ and $\tilde{f}^*(v_{\leq i})$ oracle, sample $\frac{1}{\gamma^2 \cdot e^{-\tau} \cdot \tilde{\varepsilon}}$ number of v_{i+1} and take the maximum over $\tilde{f}(v_{\leq i+1})$. This way, we can easily bound the probability of Conditions 2 or 3 not happening by γ for all $v_{\leq i}$ that $\tilde{f}(v_{\leq i}) \geq e^{-\tau} \cdot \tilde{\varepsilon}$. Note that in both of these oracle, we are ignoring the case where $\tilde{f}(v_{\leq i})$ is smaller than the threshold that causes the construction to abort. This enables us to achieve high confidence on our oracles. Using these oracles, we can bound the average of function, average budget and worst case budget of construction 2.3.8 as follows. Based on Claim 2.3.9 we have

$$\mathop{\mathbb{E}}_{(\overline{u},\overline{v}) \leftarrow \langle \overline{\mathbf{w}} \parallel \mathsf{TamAb} \rangle} [f(\overline{v})] \geq 1 - n \cdot e^{-\frac{(\tau - n \cdot \lambda^2/2)^2}{2 \cdot n \cdot \lambda^2}} - n^2\gamma - 2n \cdot \gamma.$$

The last $-2n \cdot \gamma$ is added to the right hand side to capture the probability of any of the algorithm's oracle calls failing. For the average budget, following Claim 2.3.5 we have,

$$\mathbb{E}[\mathbf{K}_{\overline{\alpha}}] \leq \frac{\ln(1/\tilde{\varepsilon}) + \lambda^2 n/2 - n \cdot \ln(1 - 3\gamma)}{\lambda} + 2 \cdot n \cdot \gamma.$$

And for the worst case budget, following Claim 2.3.7 we have

$$\Pr[\mathbf{K} \geq k] \leq \frac{e^{n\lambda^2 - k\lambda}}{\tilde{\varepsilon} - 2\gamma} + 2n \cdot \gamma.$$

Instantiating the Average Case Algorithm: Now if we set $\lambda = \sqrt{-2\ln(\varepsilon)/n}$, $\tau = \ln(1/\tilde{\varepsilon}) + \sqrt{4\ln(\delta/2n) \cdot \ln(\tilde{\varepsilon})}$ and $\gamma = \frac{\delta}{24n^2}$ then we can provide the oracles in time $poly(n/\varepsilon \cdot \delta)$ and we get:

$$\mathop{\mathbb{E}}_{(\overline{u},\overline{v}) \leftarrow \langle \overline{\mathbf{w}} \parallel \mathsf{TamAb} \rangle} [f(\overline{v})] \geq 1 - \delta$$

and

$$\mathbb{E}[\mathbf{K}_{\overline{\alpha}}] \leq \sqrt{-2n\ln(\varepsilon)} + \delta.$$

Instantiating the Worst Case Algorithm: Also, for the worst case attacks. If we select the tampering budget $k = \sqrt{2n \cdot \ln(\delta/8) \cdot \ln(\varepsilon/2)}$ and then let $\lambda = k/2n$. For $\tau = \ln(1/\tilde{\varepsilon}) + \sqrt{4\ln(\delta/2n) \cdot \ln(\tilde{\varepsilon})}$ and $\gamma = \min(\delta/24n^2, \varepsilon/4n)$ we get an algorithm that runs in time $poly(n/\varepsilon \cdot \delta)$, uses at most k tamperings and increases the average as follows

$$\mathop{\mathbb{E}}_{(\overline{u},\overline{v}) \leftarrow \langle \overline{\mathbf{w}} \parallel \mathsf{TamAb} \rangle} [f(\overline{v})] \geq 1 - \delta.$$

2.4 Algorithmic Reductions for Computational Concentration

In this section, we show a generic framework to prove computational concentration for a metric probability space by reducing its computational concentration to that of another metric probability space. We first define an embedding with some properties.

Definition 2.4.1. Let $S_1 = (\mathcal{X}_1, \mathsf{d}_1, D_1)$ and $S_2 = (\mathcal{X}_2, \mathsf{d}_2, D_2)$ be two metric probability spaces. We call a pair of mappings $(\mathbf{f}, \mathbf{g})$ (where $\mathbf{f}$ and $\mathbf{g}$ are potentially randomized) an (α, b, w) computational concentration (CC) reduction from S_1 to S_2 if the following hold:

- **Probability embedding.** The pushforward $\mathbf{f}_*(D_1)$ is α-close (in statistical distance) to D_2 and $\mathbf{g}_*(D_2)$ is α-close to D_1.
- **Almost Lipschitz property of g.** with probability 1 over all $x, x' \leftarrow D_2$, $\mathsf{d}_1(\mathbf{g}(x), \mathbf{g}(x')) \leq w \cdot \mathsf{d}_2(x, x') + b$.
- **Almost inverse mappings.** With probability at least $1 - \alpha$ for $x_1 \leftarrow D_1$, and all $x_2 \leftarrow \mathbf{f}(x_1)$, it holds that $\mathsf{d}_1(x_1, \mathbf{g}(x_2)) \leq b$.

Now we have the following lemma which how to reduce computational concentration on a metric probability space by reducing it to computational concentration on another metric probability space using the embedding between them.

Theorem 2.4.2. *Let $S_2 = (\mathcal{X}_2, \mathsf{d}_2, D_2)$ be a metric probability space and let $A_2^{\mathcal{S}(\cdot)} : \mathcal{X}_2 \to \mathcal{X}_2$ be an oracle algorithm such that for any subset $\mathcal{S} \subseteq \mathcal{X}_2$ we have $\mathsf{d}_2(A_2^{\mathcal{S}(\cdot)}(x), x) \leq k$ and*

$$\Pr_{x \leftarrow D_2}[A_2^{\mathcal{S}(\cdot)}(x) \in \mathcal{S}] \geq c(D_2(\mathcal{S}))$$

for a function $c\colon [0,1] \to [0,1]$. If $(\mathbf{f}, \mathbf{g})$ is an (α, b, w) CC reduction from $S_1 = (\mathcal{X}_1, \mathsf{d}_1, D_1)$ to $S_2 = (\mathcal{X}_2, \mathsf{d}_2, D_2)$, then there is an oracle algorithm $A_1^{\mathcal{S}(\cdot)}\colon \mathcal{X}_1 \to \mathcal{X}_1$ such that for any subset $\mathcal{S} \subseteq \mathcal{X}_1$ we have

$\mathsf{d}_1(A_1^{\mathcal{S}(\cdot)}(x), x) \leq w \cdot k + 2b$ *and*

$$\Pr_{x \leftarrow D_1}[A_1^{\mathcal{S}(\cdot)}(x) \in \mathcal{S}] \geq c(D_1(\mathcal{S})/2 - \alpha) - 2\alpha - \mathrm{negl}(n).$$

Furthermore, if A_2, $\mathbf{f}$ *and* $\mathbf{g}$ *run in time* $\mathrm{poly}(\frac{n}{\varepsilon})$, *then* A_1 *also runs in time* $\mathrm{poly}(\frac{n}{\varepsilon})$.

Proof. We define algorithm $A_1^{\mathcal{S}(\cdot)}$ on input x_1 as follows: A_1 first computes $f(x_1)$ to get x_1'. Then it creates a set $\mathcal{S}' = \{x \in \mathcal{X}_2 \colon \Pr[g(x) \in \mathcal{S}] \geq 1/2\}$ and runs $A_2^{\mathcal{S}'(\cdot)}$ on x_1' to get x_2'. Then, it computes $g(x_2')$ for at most n times until it gets some $x_2 \in \mathcal{S}$. If $\mathsf{d}_1(x_1, x_2) \leq w \cdot k + 2b$ it outputs x_2, otherwise it outputs x_1. We have

$$\begin{aligned}
&\Pr_{x_1 \leftarrow D_1}[A_1^{\mathcal{S}(\cdot)}(x_1) \in \mathcal{S}] \\
&\geq \Pr_{x_1 \leftarrow D_1}[A_2^{\mathcal{S}'(\cdot)}(f(x_1)) \in \mathcal{S}'] - 2^{-n} \\
&\qquad - \Pr_{x_1 \leftarrow D_1}[\mathsf{d}_1(x_1, x_2) \geq w \cdot k + 2b] \\
&\quad \geq \Pr_{x_1' \leftarrow D_2}[A_2^{\mathcal{S}'(\cdot)}(x_1') \in \mathcal{S}'] - \alpha - 2^{-n} \\
&\qquad - \Pr_{x_1 \leftarrow D_1}[\mathsf{d}_1(x_1, x_2) \geq w \cdot k + 2b] \\
&\geq c(D_2(\mathcal{S}')) - 2^{-n} - \alpha \\
&\qquad - \Pr_{x_1 \leftarrow D_1}[\mathsf{d}_1(x_1, x_2) \geq w \cdot k + 2b] \\
&\geq c(D_1(\mathcal{S})/2 - \alpha) - 2^{-n} - \alpha \\
&\qquad - \Pr_{x_1 \leftarrow D_1}[\mathsf{d}_1(x_1, x_2) \geq w \cdot k + 2b].
\end{aligned}$$

Note that the oracle $\mathcal{S}'(\cdot)$ cannot be implemented in polynomial time, but it could be approximated with negligible error in polynomial time. In particular, we can implement oracle $\mathcal{S}''(\cdot)$ oracle, that for every x, $\mathcal{S}''(x) = \mathcal{S}'(x)$ with probability $1 - \mathrm{negl}(n)$ over the randomness of $\mathcal{S}''$. On the other hand, we have

$$\begin{aligned}
\mathsf{d}_1(A_1(x_1), x_1) &= \mathsf{d}(x_2, x_1) \\
&\leq \mathsf{d}_1(x_2, g(x_1')) + \mathsf{d}_1(g(x_1'), x_1) \\
&\leq w \cdot \mathsf{d}_2(x_2', x_1')) + b + \mathsf{d}_1(g(x_1'), x_1) \\
&\leq w \cdot k + b + \mathsf{d}_1(g(x_1'), x_1).
\end{aligned}$$

By almost inverse property of g we know that $\Pr_{x_1 \leftarrow D_1}[\mathsf{d}_1(x_1, g(x_1')) > b] \leq \alpha$ which implies $\Pr_{x_1 \leftarrow D_1}[\mathsf{d}_1(A_1(x_1), x_1)) \geq w \cdot k + 2b] \leq \alpha$. Therefore we have,

$$\begin{aligned}
&\Pr_{x_1 \leftarrow D_1}[A_1^{\mathcal{S}(\cdot)}(x_1) \in \mathcal{S}] \\
&\geq c(D_1(\mathcal{S})/2 - \alpha) - 2^{-n} - \alpha \\
&\quad - \Pr_{x_1 \leftarrow D_1}[\mathsf{d}_1(x_1, x_2) \geq w \cdot k + 2b] \\
&\geq c(D_1(\mathcal{S})/2 - \alpha) - 2^{-n} - 2\alpha
\end{aligned}$$

□

The following construction shows an embedding from Gaussian distribution to hamming cube. Using this embedding and Lemma 2.4.2 we get computational concentration for the Gaussian distribution. The following embedding uses ideas similar to Bobkov [1997].

Construction 2.4.3. (CC reduction from (Gaussian, ℓ_1) to Hamming cube) We construct f and g as follows.

f : Let n be an even number. Given a point $x = (x_1, \ldots, x_n)$ sampled from Gaussian space of dimension n, do the following:

1. If $\exists i; |x_i| \geq \sqrt{n}/2$, output 0^{n^2}.
2. Otherwise, for each $i \in n$ compute $a_i = [\frac{x_i}{\sqrt{n}} + \frac{n}{2}]$ then uniformly sample some $y_i \in \{0,1\}^n$ such that y_i has exactly a_i number of 1s. Then append y_i s to get $y = (y_1| \ldots |y_n)$.

g : Let $y = (y_1| \ldots |y_n)$ be a Boolean vector of size n^2 (each y_i has size n). Let a_i be the number of 1s in y_i. Then sample $x = (x_1, \ldots, x_n)$ from Gaussian space conditioned on $\frac{2a_i - n}{2\sqrt{n}} \leq x_i < \frac{2a_i - n + 1}{2\sqrt{n}}$

Claim 2.4.4. *The embedding of Construction 2.4.3 is an* $(\mathrm{negl}(n), 1/\sqrt{n}, 1/\sqrt{n})$ *CC reduction from* n*-dimensional isotropic Gaussian space (where the standard deviation of each coordinate is* $\sqrt{n}$ *and the metric is* ℓ_1*) to Hamming cube (i.e., Boolean hypercube under Hamming distance).*

Proof. The embedding property of these mappings is proved in Bobkov [1997]. The mappings f and g are clearly polynomial time in n and the Almost Lipschitz and Inverse Mappings properties are straightforward. □

The following Corollary follows from Lemma 2.4.2, Claim 2.4.4 and Theorem 2.3.2.

Corollary 2.4.5. *(Computational concentration of Gaussian under* ℓ_1*) There is an algorithm* $A_{\varepsilon,\delta}^{\mathcal{S},D}(\cdot)$ *that given access to a membership oracle for any set* $\mathcal{S}$ *and a sampling oracle from an isotropic Gaussian measure*

D of dimension n, it achieves the following. If $\Pr[\mathcal{S}] \geq \varepsilon$, given ε and δ, the algorithm $A^{\mathcal{S},D}_{\varepsilon,\delta}(\cdot)$ runs in time $\mathrm{poly}(n/\varepsilon\delta)$, and with probability $\geq 1-\delta$ given a random point $\overline{x} \leftarrow D$, it maps $\overline{x}$ to a point $\overline{y} \in \mathcal{S}$ of bounded ℓ_1 distance $\ell_1(\overline{x},\overline{y}) \leq O(\sqrt{n \cdot \ln(1/\varepsilon\delta)})$.

2.4.1 Case of Gaussian or Sphere under ℓ_2

A reduction may also be used to obtain a (non-optimal) computational concentration of measure for the multi-dimensional Gaussian distribution under the ℓ_2 metric.

Theorem 2.4.6. *There is an algorithm $A^{\mathcal{S},D}_{\varepsilon,\delta}(\cdot)$ that given access to a membership oracle for any set $\mathcal{S}$ and a sampling oracle from an isotropic Gaussian measure D of dimension n, where each coordinate has variance 1, it achieves the following. If $\Pr[\mathcal{S}] \geq \varepsilon$, given $\varepsilon, \delta \geq 1/n^{O(1)}$, the algorithm $A^{\mathcal{S},D}_{\varepsilon,\delta}(\cdot)$ runs in time $\mathrm{poly}(n)$, and with probability $\geq 1-\delta$ given a random point $\overline{x} \leftarrow D$, it maps $\overline{x}$ to a point $\overline{y} \in \mathcal{S}$ of bounded ℓ_2 distance $\ell_2(\overline{x},\overline{y}) \leq O(n^{1/4} \log^{O(1)} n)$.*

Proof. Since $\epsilon \geq 1/n^{O(1)}$, at most $\epsilon/2$ and $\delta/2$ fraction of the points have a coordinate of size $\geq O(\sqrt{\log n})$. So ignoring points having such large coordinates, we may assume $\Pr[\mathcal{S}] \geq \epsilon/2$ while every point of $\mathcal{S}$ has coordinates as small as $O(\sqrt{\log n})$, and we may assume the point we are mapping also has small coordinates (except our algorithm should now work for $1-\delta/2$ fraction of the points instead of for $1-\delta$ fraction.)

Now, when each coordinate is $O(\sqrt{\log n})$, the l_2 distance between two points is at most $O(\sqrt{d_H \log n})$, where d_H is the Hamming distance of the two points. Now, the theorem follows from our main theorem for Hamming distance. □

We should note that the above computational bound is not information-theoretically tight, since for the Gaussian ℓ_2 metric probability space, where each coordinate has variance 1, the right bound is $O(\sqrt{\ln(1/(\epsilon\delta))})$. (This follows e.g. from the Gaussian isoperimetric inequality proved in Sudakov and Tsirel'son [1978], Borell [1975], which shows the half-space is isopermetrically optimal for the Gaussian distribution.)

Finally, the following shows that our results are not limited to product spaces, and may for example be applied to computational concentration of measure for the high-dimensional sphere.

Theorem 2.4.7. *There is an algorithm $A^{\mathcal{S},D}_{\varepsilon,\delta}(\cdot)$ that given access to a membership oracle for any set $\mathcal{S}$ and a sampling oracle from the uniform measure D on the unit sphere of dimension n, it achieves the following. If $\Pr[\mathcal{S}] \geq \varepsilon$, given $\varepsilon, \delta \geq 1/n^{O(1)}$, the algorithm $A^{\mathcal{S},D}_{\varepsilon,\delta}(\cdot)$ runs in time $\mathrm{poly}(n)$, and with probability $\geq 1-\delta$ given a random point $\overline{x} \leftarrow D$, it maps $\overline{x}$ to a point $\overline{y} \in \mathcal{S}$ of bounded ℓ_2 distance $\ell_2(\overline{x},\overline{y}) \leq O(n^{-1/4} \log^{O(1)} n)$.*

Proof. First, we note that a random Gaussian vector, where each coordinate has variance 1, has ℓ_2 norm $\sqrt{n} + O(n^{1/4})$ except for arbitrary inverse polynomial probability.

So given $\overline{x}$, we can map it to a new vector $\overline{x}'$ with the same direction as $\overline{x}$ but with a random length of distribution square root of chi square, so that the new vector has the Gaussian distribution. We also map the set $\mathcal{S}$ to the set $\mathcal{S}' = \{r \cdot s : r \in n^{1/2} + O(n^{1/4}), s \in \mathcal{S}\}$, where the new set still has probability $\geq \epsilon/2$ under the Gaussian distribution. By the computational concentration of measure for the Gaussian, we know that we can map, with probability $1 - \delta/2$, $\overline{x}'$ to a point $\overline{y}' \in \mathcal{S}'$ of distance $n^{1/4} \log^{O(1)} n$ from $\overline{x}'$ in ℓ_2. Let $\overline{y}$ be the projection of $\overline{y}'$ onto the unit sphere. Therefore

$$\begin{aligned} d_{\ell_2}(\overline{x}, \overline{y}) &\leq d_{\ell_2}(\overline{x}, \overline{x}'/\sqrt{n}) \\ &\quad + d_{\ell_2}(\overline{x}'/\sqrt{n}, \overline{y}'\sqrt{n}) + d_{\ell_2}(\overline{y}'/\sqrt{n}, \overline{y}) \\ &= O(n^{1/4} \log^{O(1)} n). \end{aligned}$$

□

These types of relations between concentration of measure of Gaussian and uniform sphere measures has been well-known information-theoretically, e.g. see [Ledoux, 2001, page 2] where concentration for Gaussian is derived from concentration for sphere. In the above we showed a similar relation for *computational* concentration of measure, this time deriving for the sphere from the Gaussian.

2.5 Computational Concentration around Mean

Let $(\mathcal{X}, \mathsf{d}, D)$ be a metric probability space and $f \colon \mathcal{X} \mapsto \mathbb{R}$ a measurable function (with respect to D). For any Borel set $\mathcal{T} \subseteq \mathbb{R}$, an parameters $k, \delta \in \mathbb{R}_+$, one can define a computational problem as follows. Given oracle access to a sampler from D, d and function $f(\cdot)$, map a given input $x \in \mathcal{X}$ algorithmically to $y \in \mathcal{Y}$, such that: (1) $\mathsf{d}(x, y) \leq k$, and (2) $f(y) \in \mathcal{T}$ for $1 - \delta$ fraction of $x \in \mathcal{X}$ according to D. If we already know that $(\mathcal{X}, \mathsf{d}, D)$ is (ε, δ, k) (computationally) concentrates, and if $\Pr_{x \leftarrow D}[f(x) \in \mathcal{T}] \geq \varepsilon$, then it implies that by changing x by at most distance k into a new point y, we can (algorithmically) get $f(y) \in \mathcal{T}$, by defining $\mathcal{S} = f^{-1}(\mathcal{T})$ and noting that $\Pr_D[\mathcal{S}] \geq \varepsilon$. This algorithm needs oracle access to $\mathcal{S}$

Computational concentration around mean. Again, let $(\mathcal{X}, \mathsf{d}, D)$ be a metric probability space and let $f \colon \mathcal{X} \mapsto \mathbb{R}$ be measurable. Now suppose $\eta = \mathbb{E}_{x \leftarrow D}[f(x)]$. If we already know, by information theoretic concentration bounds, that $\Pr_{x \leftarrow D}[|f(x) - \eta| \leq T] \geq 1 - \delta$, then it means that a trivial algorithm that does not even change given $x \leftarrow D$, finds a point where $f(x)$ is T-close to the average η. However, this becomes nontrivial, if the goal of the algorithm is to find y that is close to x, and that $f(y)$ is much *closer* to the mean η than what x achieves. In particular, suppose we somehow know that $\Pr_{x \leftarrow D}[|f(x) - \eta| \leq t] \geq \varepsilon$

for $t \ll T, \varepsilon \ll 1-\delta$. (Such results usually follow from the same concentration inequalities proving $\Pr_{x \leftarrow D}[|f(x)-\eta| \leq T] \approx 1$.) The smaller t is, the "higher quality" the point x has in terms of $f(x)$ being closer to the mean. This means the set $\mathcal{S} = \{x \colon |f(x)-\eta| \leq t\}$ has D measure at least ε. Therefore, if the space $(\mathcal{X}, \mathsf{d}, D)$ is (ε, δ, k) computationally concentrated, then we can conclude that there is an efficient algorithm (whose running time can polynomially depend on $1/\varepsilon\delta$ and) that maps $1-\delta$ fraction of $x \leftarrow D$ to a point $y \in \mathcal{S}$. Different, but similar, statements about one-sided concentration can be made as well, if we start from weaker conditions of the form $\Pr_{x \leftarrow D}[f(x) > \eta + t] \leq \varepsilon$ (or $\Pr_{x \leftarrow D}[f(x) < \eta - t] \leq 1-\varepsilon$) leading to a weaker conclusion: we can map x to a point y satisfies $f(x) \geq \eta - t$ (or $f(x) \leq \eta + t$).

Finally, we note that even if the mean η is *not* known to the mapping algorithm A, good approximations of it can be obtained by repeated sampling and taking their average. So for simplicity, and without loss of generality, the reader can assume that η is known to the mapping algorithm A.

Special case of Lipschitz functions: algorithmic proofs of concentration. When $f \colon \mathcal{X} \mapsto \mathbb{R}$ is Lipschitz, i.e., $|f(x)-f(y)| \leq \mathsf{d}(x,y)$, computational concentration around a set like $\mathcal{S} = \{x \colon |f(x)-\eta| \leq t\}$ (or similar one-sided variants) means something stronger than before. We now have an algorithm that *indirectly proves* the concentration around η by efficiently finding points that are almost at the border defined by η. Namely, the Lipschitz now implies that $|f(x)-f(y)| \leq k$, whenever $|x-y| \leq k$. Therefore, the algorithm A mapping x to y is also proving that $1-\delta$ measure of the space $(\mathcal{X}, D)$ is mapped under f to a point that is $k+t$ close to average η.

All the above arguments are general and apply to any metric probability space. Below, we discuss an special case of a "McDiarmid type" inequality in more detail to demonstrate the power of this argument.

Theorem 2.5.1. *(An algorithmic variant of McDiarmid inequality) Suppose $D \equiv D_1 \times \cdots \times D_n$ is a product measure on a product space $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_n$, and let $f \colon \mathcal{X} \mapsto \mathbb{R}$ be such that $|f(\overline{x}) - f(\overline{x}')| \leq \alpha_i$ whenever $\overline{x}$ and $\overline{x}'$ only differ in the i^{th} coordinate. Let $a = \|\overline{\alpha}\|_2$ for $\overline{\alpha} = (\alpha_1, \ldots, \alpha_n)$. Let $\eta = \mathbb{E}_{x \leftarrow D}[f(\overline{x})]$ and $\mathcal{S} = \{x \colon f(\overline{x}) \leq \eta + \varepsilon \cdot a\}$. Then there is an algorithm $A_{\varepsilon,\delta}^{D, f(\overline{x})}(\cdot)$ running in time $\operatorname{poly}(n/\varepsilon\delta)$ that uses oracle access to f and a sampler from D, and it holds that*

$$\Pr_{x \leftarrow D, y \leftarrow A_{\varepsilon,\delta}^{D,f}(\overline{x})} \Big[\overline{y} \in \mathcal{S} \quad \text{and} \quad |f(\overline{x}) - f(\overline{y})| \leq O\left(\sqrt{m \cdot \log(1/\varepsilon\delta)}\right) \Big] \geq 1-\delta.$$

Corollaries for special cases. Theorem 2.5.1 implies a similar result when the quality of the destination region is base on the ℓ_1 norm; namely, $\mathcal{S} = \{x \colon f(\overline{x}) \leq \eta + \varepsilon \cdot \|\alpha\|_1\}$, but this follows from the same statement

since $\|\alpha\|_2 \leq \|\alpha\|_1$. In addition, for the special case where $\alpha_i = 1$ for all i,[8] and let $\gamma, \delta = 1/\operatorname{poly}(n)$ be arbitrarily small inverse polynomials. In that case, Theorem 2.5.1, shows that for $1-\delta$ fraction of $\overline{x} \leftarrow D$, we can map $\overline{x}$ to $\overline{y}$ in $\operatorname{poly}(n)$ time in such a way that $f(\overline{y}) \leq \mathbb{E}[f(D)] + \gamma$ and $|f(\overline{x}) - f(\overline{y})| \leq \widetilde{O}(\sqrt{n})$. If we choose $\gamma < 1/2$, due to the Lipschitz condition, we can also find some $\overline{y}$ for which $f(\overline{y}) \in \mathbb{E}[f(D)] \pm 1$. This is possible by first finding some $\overline{y}$ where $f(\overline{y}) \leq \mathbb{E}[f(D)] + \gamma$, and then go back over the coordinates in which $\overline{x}$ and $\overline{y}$ differ and only changing some of them to get $\overline{y}'$ where $f(\overline{y}') \in \mathbb{E}[f(D)] \pm 1$, and output $\overline{y}'$ instead. We note that, however, that whenever we want to choose $\gamma < 1/2$, we need to also choose $\varepsilon < 1/(2n)$. For this range of small ε, we *cannot* use the computational concentration results of Mahloujifar and Mahmoody [2019a], but we can indeed use the stronger computational concentration results of this chapter that prove computational concentration around any non-negligible event.

Proof of Theorem 2.5.1. For starters, suppose η is given. In that case, we first observe that $\Pr_D[\mathcal{S}] \geq 1 - e^{-2\varepsilon^2} = \Theta(\varepsilon^2)$ by McDiarmid's inequality itself. We can then apply Theorem 2.3.2.

When η is not given, we can find a sufficiently good approximation of it, such that $\eta' \in \eta \pm \|\alpha\|_2 \cdot \varepsilon/10$ (in time $\operatorname{poly}(n/\varepsilon\delta)$ and error probability $\delta/10$) and use it instead of η. Obtaining such η' can be done because any x, x' satisfy $|f(x) - f(x')| \leq \|\alpha\|_1$. Therefore, we can obtain $\eta' \in \eta \pm \lambda \cdot \|\alpha\|_1$ in time by sampling $\ell = \operatorname{poly}(n/\lambda\delta)$ (for sufficiently large ℓ) many points $\overline{x}_1, \ldots, \overline{x}_\ell \leftarrow D$ and letting $\eta' = \mathbb{E}_{i \leftarrow \ell} f(\overline{x}_i)$. The only catch is that we want $\eta' \in \eta \pm \varepsilon \cdot \|\alpha\|_2$. However, since it holds that $\|\alpha\|_2 \leq \|\alpha\|_1 \cdot \sqrt{n}$, we can choose $\lambda = \varepsilon/\sqrt{n}$, and use the same procedure to obtain $\eta' \in \eta \pm \lambda \cdot \|\alpha\|_1$ with probability $1 - \delta/10$ in time $\operatorname{poly}(n/\varepsilon\delta)$. □

2.6 Limits of Nonadaptive Methods for Proving Computational Concentration

In this section, we consider three restricted types of attacks and prove exponential lower bounds on their running time. The attacks are

- **I.i.d. queries:** An attack where given $\overline{x}$, we query i.i.d. points whose distribution may depend on $\overline{x}$, until one of these points lies in $\mathcal{S}$. The analysis of this attack boils down to analysis of a single-query attack where we want to maximize the probability of $\mathcal{S}$-membership of the queried point.

- **Non-adaptive queries:** An attack where given $\overline{x}$, we output a list of points, and query all the points in this list. Since the points in the list are determined before the querying, this attack is non-adaptive.

[8] For example, this could be the setting of Hoeffding's inequality in which each coordinate D_i is arbitrarily distributed over $[0,1]$, and $f(\overline{x}) = \sum_{i \in [n]} x_i$, where $\overline{x} = (x_1, \ldots, x_n)$

It is easy to see (and we give a proof below) how lower bounding this type of attack reduces to the previous type of attack.

- **Querying only points close enough to have a chance to be output:** If we are interested in finding a point at distance $\leq d$ from $\overline{x}$, one may be tempted to limit the queried points to points at distance $\leq d$ from $\overline{x}$. We show how lower bounding this type of attack reduces to the previous type of attack.

Theorem 2.6.1. *(Lower bound for non-adaptive algorithms) Let D be the uniform probability distribution on $\{1,-1\}^n$, and let $\varepsilon = 1/2$ and $\delta < 1/2$ be constants. There does not exist any non-adaptive algorithm A that given $\overline{x} \leftarrow D$, the algorithm outputs $m = n^{O(1)}$ (random) points $\overline{y}^1, \ldots, \overline{y}^m$, all within Hamming distance $n^{1-\Omega(1)}$ of $\overline{x}$, such that given any set $\mathcal{S}$ with $\Pr[\mathcal{S}] \geq \varepsilon$, one of these m points lies in $\mathcal{S}$ with probability $1-\delta$ over the randomness of x and randomness of $\overline{y}^1, \ldots, \overline{y}^m$.*

Proof. Assume for the sake of contradiction that such an algorithm A exists. Consider the following modified algorithm: given $\overline{x}$, run A to produce $\overline{y}^1, \ldots, \overline{y}^m$, and then let $\overline{z}^1$ be one of those m vectors uniformly at random. To produce $\overline{z}^2$, run A independently afresh, and let $\overline{z}^2$ be one of the m freshly produced vectors. We can continue in this way, and produce the vectors $\overline{z}^1, \ldots, \overline{z}^{m'}$ as the output of the modified algorithm. By the assumption, for any constant $\delta' \in (\delta, 1/2)$, with probability $1-\delta'$ over the randomness of $\overline{x}$, algorithm A has success probability at least $1/n$, hence each $\overline{z}^i$ lies in $\mathcal{S}$ with probability $\geq 1/mn$. Hence for these $\overline{x}$, if we choose $m' = mn^2$, with probability $1-(1-1/mn)^{m'} = 1-o(1)$, the modified algorithm succeeds. Therefore, the average success probability of the algorithm is $\geq 1-\delta'-o(1) \geq 1/2+\Omega(1)$.

The above argument shows that we only need to look at algorithms where $\overline{y}^1, \ldots, \overline{y}^m$ are independent given $\overline{x}$. Thus, it is enough to show that there does not exist a random mapping from $\overline{x}$ to a vector $\overline{y}$ in such a way that with probability $1-\delta$ over the randomness of $\overline{x}$, the probability $\Pr[\overline{y} \in \mathcal{S}]$ is non-negligible (since m is polynomial in n).

For the sake of contradiction, assume such a mapping from $\overline{x}$ to $\overline{y}$ exists. Let $\mathcal{S}$ be a random half-space, i.e. $\mathcal{S} = \{\overline{z} : \sum_{i=1}^n a_i z_i \leq 0\}$ for a uniformly random vector $a = (a_1, \ldots, a_n) \in \{-1,1\}^n$. We will show that for every $\overline{x}$, with probability δ over the randomness of a, the probability $\Pr[\overline{y} \in \mathcal{S}]$ is negligible. By an averaging argument, this shows that there exists a half-space $\mathcal{S}$ such that with probability δ over the randomness of $\overline{x}$, $\Pr[\overline{y} \in \mathcal{S}]$ is negligible, completing the proof.

As mentioned above, we want to show that for every $\overline{x}$, a random half-space is troublesome for the algorithm. By symmetry, without loss of generality, we may assume $\overline{x} = (1,1,\ldots,1)$. Let $\eta = (\eta_1, \ldots, \eta_n) = (\overline{x}-\overline{y})/2$ be the characteristic vector for the coordinates for which $\overline{y}$ is different from $\overline{x}$. We note that $\overline{y} \in \mathcal{S}$ iff $\sum_i a_i - 2\sum_i a_i \eta_i \leq 0$. We know that with probability $\delta + \Omega(1)$ over the randomness of a, we have $\sum_i a_i \geq$

$\Omega(\sqrt{n})$. (This easily follows from the central limit theorem.) Now, conditioned on η, the sum $\sum_i \eta_i a_i$ is actually a sum of $n^{1-\Omega(1)}$-many ± 1 independent random variables of mean zero, so $\Pr[\sum_i \eta_i a_i \geq \Omega(\sqrt{n})]$ is a negligible, actually exponentially small, probability. This implies over the randomness of a and η, $\Pr[\sum_i \eta_i a_i \geq \Omega(\sqrt{n})]$ is negligible. Thus, except for an $o(1)$ fraction of random half-spaces, $\Pr[\sum_i \eta_i a_i \geq \Omega(\sqrt{n})]$ is negligible over the randomness of $\overline{y}$. Thus, with probability at least $\delta + \Omega(1) - o(1) \geq \delta$ over the randomness of a, we have both

- $\sum_i a_i \geq \Omega(\sqrt{n})$, and
- $\Pr[\sum_i a_i \eta_i = \Omega(\sqrt{n})]$ is negligible over the randomness of $\overline{y}$.

In this case, $\overline{y}$ does not lie in $\mathcal{S}$ except with non-negligible. □

Remark 2.6.2. It can be seen that the above theorem holds whenever ε and δ are positive constants such that $\varepsilon + \delta < 1$. It can be seen that the above theorem does not hold when $\varepsilon + \delta > 1$ since when we set $\overline{y} = \overline{x}$, our failure probability δ is exactly $1 - \varepsilon$.

Lemma 2.6.3. *Given a radius r, assume an adaptive algorithm A, given $\overline{x}$, wants to find a vector $\overline{y} \in \mathcal{S}$ in the ball of radius r around $\overline{x}$. Furthermore, assume that the algorithm does not make any $\mathcal{S}$-membership oracle queries regarding points outside the ball. Then, we can transform the algorithm into a non-adaptive algorithm with the same performance.*

Proof. When the algorithm ever queries about a point $\overline{y}$ (and by assumption $\overline{y}$ is in the ball), if the oracle says that $\overline{y} \in S$, then we are done (since we have found our desired point.) So the algorithm may always pretend that the result of each membership query about each queried point is that the point is not in $\mathcal{S}$. This equivalent algorithm is non-adaptive. □

Corollary 2.6.4. *In the $\{0,1\}^n$ uniform product space, when we want to find a point $\overline{y} \in \mathcal{S}$ at distance $n^{1/2+\varepsilon}$ from a random $\overline{x}$ (for some $\varepsilon \in (0, 1/2)$), to be query-efficient, we need to query about $\mathcal{S}$-membership of points having distance more than $n^{1/2+\varepsilon}$.*

The above corollary says that even though we are interested in points in a ball of certain radius around $\overline{x}$, we have to query about points outside that ball. When we notice that we are not assuming any structure on the set $\mathcal{S}$ other than it should have some minimum mass, the above corollary becomes all the more surprising!

Chapter 3

Separating Computational and Statistical Robustness for Inference-time Attacks

3.1 Introduction

Our results in Part 2 suggest that perhaps the existence of adversarial example is due to fundamental reasons that might be inevitable. In Part 2, Section 2 we showed that for natural theoretical distributions (e.g., isotropic Gaussian of dimension n) and natural metrics over them (e.g., ℓ_0, ℓ_1 or ℓ_2), adversarial examples are inevitable. Namely, the concentration of measure phenomenon [Ledoux, 2001, Milman and Schechtman, 1986] in such metric probability spaces imply that small perturbations are enough to map almost all the instances x into a close x' that is misclassified. This line of work, however, does not yet say anything about "natural" distributions of interest such as images or voice, as the precise nature of such distributions are yet to be understood.

Can lessons from cryptography help? Given the pessimistic state of affairs, researchers have asked if we could use lessons from cryptography to make progress on this problem [Madry, 2018, Goldwasser, 2018, Mahloujifar and Mahmoody, 2018b]. Indeed, numerous cryptographic tasks (e.g. encryption of long messages) can only be realized against attackers that are computationally bounded. In particular, we know that all encryption methods that use a short key to encrypt much longer messages are insecure against computationally unbounded adversaries. However, when restricted to computationally bounded adversaries

this task becomes feasible and suffices for numerous settings. This insight has been extremely influential in cryptography. Nonetheless, despite attempts to build on this insight in the learning setting, we have virtually no evidence on whether this approach is promising. Thus, in this chapter we study the following question:

Could we hope to leverage computational hardness for the benefit of adversarially robust learning by rendering successful attacks computationally infeasible?

Taking a step in realizing this vision, we provide formal definitions for *computational* variants of robust learning. Following the cryptographic literature, we provide a game based definition of computationally robust learning. Very roughly, a game-based definition consists of two entities: a *challenger* and an *attacker*, that interact with each other. In our case, as the first step the challenger generates independent samples from the distribution at hand, use those samples to train a learning algorithm, and obtain a hypothesis h. Additionally, the challenger samples a fresh challenge sample x from the underlying distribution. Next, the challenger provides the attacker with oracle access to $h(\cdot)$ and x. At the end of this game, the attacker outputs a value x' to the challenger. The attacker declares this execution as a "win" if x' is obtained as a small perturbation of x and leads to a misclassification. We say that the learning is computationally robust as long as no attacker from a class of adversaries can "win" the above game with a probability much better than some base value. (See Definition 3.3.1.) This definition is very general and it implies various notions of security by restricting to various classes of attackers. While we focus on polynomially bounded attackers in this chapter, we remark that one may also naturally consider other natural classes of attackers based on the setting of interest (e.g. an attacker that can only modify certain part of the image).

What if adversarial examples are actually easy to find? Mahloujifar and Mahmoody [2019b] studied this question, and showed that as long as the input instances come from a product distribution, and if the distances are measured in Hamming distance, adversarial examples with sublinear perturbations can be found in polynomial time. This result, however, did not say anything about other distributions or metrics such as ℓ_∞. Thus, it was left open whether computational hardness could be leveraged in any learning problem to guarantee its robustness.

3.1.1 Summary of Results

From computational hardness to computational robustness. In this chapter, we show that computational hardness can indeed be leveraged to help robustness. In particular, we present a learning problem $\mathcal{P}$ that has a classifier $h_{\mathcal{P}}$ that is *only* computationally robust. In fact, let Q be *any* learning problem that has a classifier with "small" risk α, but that adversarial examples *exist* for classifier h_{Q} with higher probability $\beta \gg \alpha$ under the ℓ_0 norm (e.g., Q could be *any* of the well-studied problems in the literature with a vulnerable

classifier $h_{\mathcal{Q}}$ under norm ℓ_0). Then, we show that there is a "related" problem $\mathcal{P}$ and a related classifier $h_{\mathcal{P}}$ that has *computational* risk (i.e., risk in the presence of *computationally bounded* tampering adversaries) at most α, but the risk of $h_{\mathcal{P}}$ will go up all the way to $\approx \beta$ if the tampering attackers are allowed to be computationally unbounded. Namely, computationally bounded adversaries have a much smaller chance of finding adversarial examples of small perturbations for $h_{\mathcal{P}}$ than computationally unbounded attackers do. (See Theorem 3.4.2.)

The computational robustness of the above construction relies on allowing the hypothesis to sometimes "detect" tampering and output a special symbol $\star$. The goal of the attacker is to make the hypothesis output a wrong label and *not* get detected. Therefore, we have proved, along the way, that allowing tamper detection can also be useful for robustness. Allowing tamper detection, however, is not always an option. For example a real-time decision making classifier (e.g., classifying a traffic sign) that *has to* output a label, even if it detects that something might be suspicious about the input image. We prove that even in this case, there is a learning problem $\mathcal{P}$ with binary labels and a classifier h for $\mathcal{P}$ such that computational risk of h is almost zero, while its information theoretic risk is $\approx 1/2$, which makes classifiers' decisions under attack meaningless. (See Theorem 3.4.8).

Extension: existence of learning *problems* that are computationally robust. Our result above applies to certain classifiers that "separate" the power of computationally bounded vs. that of computationally unbounded attackers. Doing so, however, does not rule out the possibility of finding information theoretically robust classifiers for the *same* problem. So, a natural question is: can we extend our result to show the existence of learning tasks for which *any* classifier is vulnerable to unbounded attackers, while computationally robust classifiers for that task exist? At first, it might look like an impossible task, in "natural" settings, in which the ground truth function c itself is robust under the allowed amount of perturbations. (For example, in case of image classification, Human is the robust ground truth). Therefore, we cannot simply extend our result in this setting to rule out the existence of robust classifiers, since they might simply exist (unless one puts a limits on the complexity of the learned model, to exclude the ground truth function as a possible hypothesis).

However, one can still formulate the question above in a meaningful way as follows: Can we have a learning task for which any *polynomial time* learning algorithm (with polynomial sample complexity) is forced to produce (with high probability) hypotheses with low robustness against unbounded attacks? Indeed, in this chapter we also answer this question affirmatively, as a corollary to our main result, by also relying on recent results proved in recent exciting works of [Bubeck et al., 2018c,a, Degwekar and Vaikuntanathan, 2019].

In summary, our work provides credence that perhaps restricting attacks to computationally bounded adversaries holds promise for achieving computationally robust machine learning that relies on computational hardness assumptions as is currently done in cryptography.

From computational robustness back to computational hardness. Our first result shows that computational hardness can be leveraged in some cases to obtain nontrivial computational robustness that beats information theoretic robustness. But how about the reverse direction; are computational hardness assumptions necessary for this goal? We also prove such reverse direction and show that nontrivial computational robustness implies computationally hard problems in **NP**. In particular, we show that a non-negligible gap between the success probability of computationally bounded vs. that of unbounded adversaries in attacking the robustness of classifiers implies strong average-case hard distributions for class **NP**. Namely, we prove that if the distribution D of the instances in learning task is efficiently samplable, and if a classifier h for this problem has computational robustness α, information theoretic robustness β, and $\alpha < \beta$, then one can efficiently sample from a distribution S that generates Boolean formulas $\phi \leftarrow S$ that are satisfiable with overwhelming probability, yet no efficient algorithm can find the satisfying assignments of $\phi \leftarrow S$ with a non-negligible probability. (See Theorem 3.5.2 for the formal statement.)

What world do we live in? As explained above, our main question is whether adversarial examples could be prevented by relying on computational limitations of the adversary. In fact, even if adversarial examples exist for a classifier, we might be living in either of two completely different worlds. One is a world in which computationally unbounded adversaries can find adversarial examples (almost) whenever they exist and they would be as powerful as information-theoretic adversaries. Another world is one in which machine learning could leverage computational hardness. Our work suggests that computational hardness can potentially help robustness for certain learning problems; thus, we are living in the better world. Whether or not we can achieve *computational* robustness for *practical* problems (such as image classification) that beats their information-theoretic robustness remains an intriguing open question. A related line of work [Bubeck et al., 2018c,a, Degwekar and Vaikuntanathan, 2019] studied other "worlds" that we might be living in, and studied whether adversarial examples are due to the computational hardness of *learning* robust classifiers. They designed learning problems demonstrating that in some worlds, robust classifiers might exist, while they are hard to be obtained efficiently. We note however, that the goal of those works and our work are quite different. They deal with how computational constraints might be an issue and *prevent* the learner from reaching its goal, while our focus is on how such constraints on adversaries can *help* us achieve robustness guarantees that are not achievable information theoretically.

What does our result say about robustifying other natural learning tasks? Our results only show the *existence* of a learning task for which computational robustness is very meaningful. So, one might argue that this is an ad hoc phenomenon that might not have an impact on other practical problems (such as image classification). However, we emphasize that prior to our work, there was *no* provable evidence that computational hardness can play any positive role in robust learning. Indeed, our results also shed light on how computational robustness can potentially be applied to other, perhaps more natural learning tasks. The reason is that the space of all possible ways to tamper a high dimensional vector is *exponentially* large. Lessons from cryptography, and the construction of our learning task proving our main result, suggest that, in such cases, there is potentially a huge gap between the power of computationally bounded vs. unbounded search algorithms. On the other hand, there are methods proposed by researchers that *seem* to resist attacks that try to find adversarial examples [Madry et al., 2018], while the certified robustness literature is all focused on modeling the adversary as a computationally unbounded entity who can find adversarial examples within a certain distance, so long as they exist [Raghunathan et al., 2018, Wong and Kolter, 2018, Sinha et al., 2018, Wong et al., 2018]. Our result shows that, perhaps we shall start to consider *computational* variants of certification methods that focus on computationally bounded adversaries, as by doing so we might be able to prove better robustness bounds for methods that are designed already.

Techniques

We prove our main result about the possibility of computationally robust classifiers (Theorem 3.4.2) by "wrapping" an arbitrary learning problem Q with a vulnerable classifier by adding *computational* certification based on cryptographic digital signatures to test instances. A digital signature scheme (see Definition 3.2.2) operates based on two generated keys $(\mathsf{vk}, \mathsf{sk})$, where sk is private and is used for *signing* messages, and vk is public and is used for *verifying* signatures. Such schemes come with the guarantee that a computationally bounded adversary with the knowledge of vk cannot sign new messages on its own, even if it is given signatures on some previous messages. Digital signature schemes can be constructed based on the assumption that one-way functions exist.[1] Below we describe the ideas behind this result in two steps.

Initial Attempt. Suppose D_{Q} is the distribution over $\mathcal{X} \times \mathcal{Y}$ of a learning problem Q with input space $\mathcal{X}$ and label space $\mathcal{Y}$. Suppose D_{Q} had a hypothesis h_{Q} that can predict correct labels reasonably well, $\Pr_{(x,y)\leftarrow D_{\mathsf{Q}}}[h(x) \neq y] \leq \alpha$. Suppose, at the same time, that a (perhaps computationally unbounded) adversary A can perturb test instances like x into a close adversarial example x' that is now likely to be misclassified by

[1] Here, we need signature schemes with "short" signatures of poly-logarithmic length over the security parameter. They could be constructed based on exponentially hard one-way functions [Rompel, 1990] by picking the security parameter sub-exponentially smaller that usual and using universal one-way hash functions to hash the message to poly-logarithmic length..

h_{Q},

$$\Pr_{(x,y)\leftarrow D_{\mathsf{Q}}}[h(x') \neq y; x' = \mathsf{A}(x)] \geq \beta \gg \alpha.$$

Now we describe a related problem $\mathcal{P}$, its distribution of examples $D_{\mathcal{P}}$, and a classifier $h_{\mathcal{P}}$ for $\mathcal{P}$. To sample an example from $D_{\mathcal{P}}$, we first sample $(x, y) \leftarrow D_{\mathsf{Q}}$ and then modify x to $\overline{x} = (x, \sigma_x)$ by attaching a *short* signature $\sigma_x = \mathrm{Sign}(\mathsf{sk}, x)$ to x. The label y of $\overline{x}$ remains the same as that of x. Note that sk will be kept secret to the sampling algorithm of $D_{\mathcal{P}}$. The new classifier $h_{\mathcal{P}}$ will rely on the public parameter vk that is available to it. Given an input $\overline{x} = (x, \sigma_x)$, $h_{\mathcal{P}}$ first checks its integrity by verifying that the given signature σ_x is valid for x. If the signature verification does not pass, $h_{\mathcal{P}}$ rejects the input as adversarial without outputting a label, but if this test passes, $h_{\mathcal{P}}$ outputs $h_{\mathsf{Q}}(x)$.

To successfully find an adversarial example $\overline{x}'$ for $h_{\mathcal{P}}$ through a small perturbation of $\overline{x} = (x, \sigma)$ sampled as $(\overline{x}, y) \leftarrow D_{\mathcal{P}}$, an adversary A can pursue either of the following strategies. **(I)** One strategy is that A tries to find a new signature $\sigma' \neq \sigma_x$ for the same x, which will constitute as a sufficiently small perturbation as the signature is short. Doing so, however, is not considered a successful attack, as the label of $\overline{x}'$ remains the same as that of the true label of the untampered point $\overline{x}$. **(II)** Another strategy is to perturb the x part of $\overline{x}$ into a close instance x' and then trying to find a correct signature σ' for it, and outputting $\overline{x}' = (x', \sigma')$. Doing so would be a successful attack, because the signature is short, and thus $\overline{x}'$ would indeed be a close instance to $\overline{x}$. However, doing this is computationally infeasible, due to the very security definition of the signature scheme. Note that $(x'\sigma')$ is a forgery for the signature scheme, which a computationally bounded adversary cannot construct because of the security of the underlying signature scheme. This means that the *computational* risk of $h_{\mathcal{P}}$ would remain at most α.

We now observe that information theoretic (i.e., computationally unbounded) attackers can succeed in finding adversarial examples for $h_{\mathcal{P}}$ with probability at least $\beta \gg \alpha$. In particular, such attacks can first find an adversarial example x' for x (which is possible with probability β over the sampled x), construct a signature σ' for x', and then output (x', σ'). Recall that an unbounded adversary can construct a signature σ' for x' using exhaustive search.

Actual construction. One main issue with the above construction is that it needs to make vk publicly available, as a public parameter to the hypothesis (after it is sampled as part of the description of the distribution $D_{\mathcal{P}}$). Note that it is computationally hard to construct the hypothesis described above without knowing vk. The problem with revealing vk to the learner is that the distribution of examples should come with some extra information other than samples. However, in the classical definition of a learning problem, the learner only has access to samples from the distribution. In fact, if we were allowed to pass some extra information to the learner, we could pass the description of a robust classifier (e.g. the ground truth) and

the learning task becomes trivial. The other issue is that the distribution $D_{\mathcal{P}}$ is not *publicly samplable* in polynomial time, because to get a sample from $D_{\mathcal{P}}$ one needs to use the signing key sk, but that key is kept secret. We resolve these two issues with two more ideas. The first idea is that, instead of generating *one* pair of keys $(\mathsf{vk}, \mathsf{sk})$ for $D_{\mathcal{P}}$ and keeping sk_D secret, we can generate a fresh pair of keys $(\mathsf{vk}_x, \mathsf{sk}_x)$ every time that we sample $(x, y) \leftarrow D_{\mathsf{Q}}$ and attach vk_x also to the actual instance $\overline{x} = (x, \sigma_x, \mathsf{vk}_k)$. The modified hypothesis $h_{\mathcal{P}}$ also uses this key and verifies (x, σ_x) using vk_x. This way, the distribution $D_{\mathcal{P}}$ is publicly samplable, and moreover, there is no need for making vk available as a public parameter. However, this change of the distribution $D_{\mathcal{P}}$ introduces a new possible way to attack the scheme and to find adversarial examples. In particular, now the adversary can try to perturb vk_x into a close string vk' for which it knows a corresponding signing key sk', and then use sk' to sign an adversarial example x' for x and output $(x', \sigma', \mathsf{vk}')$. However, to make this attack impossible for the attacker under small perturbations of instances, we use error correction codes and employ an encoding $[\mathsf{vk}_x]$ of the verification key (instead of vk_x) that needs too much change before one can fool a decoder to decode to any other $\mathsf{vk}' \neq \mathsf{vk}_x$. But as long as the adversary cannot change vk_x, the adversary cannot attack the robustness computationally. (See Construction 3.4.1.)

To analyze the construction above (see Theorem 3.4.2), we note that the computationally bounded adversary would need to change $\Omega(|x|)$ number of bits in $(x, \sigma, [\mathsf{vk}])$ to get $(x', \sigma', [\mathsf{vk}'])$ where $x \neq x'$. This is because the encoded $[\mathsf{vk}]$ would need $\Omega(|x|)$ number of perturbations to change the encoded vk, and if vk remains the same it is hard computationally to find a valid signature. On the other hand, a computationally unbounded adversary can focus on perturbing x into x' and then forge a short signatures for it, which could be as small as $\mathrm{poly}(\log(|x|))$ perturbations.

Extension to problems, rather than specific classifiers for them. Note that the construction above could be wrapped around any learning problem. In particular, we can pick an original problem that is not (information theoretically) robustly learnable in polynomial time. These problems, which we call them robust-hard are studied recently in [Bubeck et al., 2018c] and [Degwekar and Vaikuntanathan, 2019] where they construct such robust-hard problems to show the effect of computational limitation in robust learning (See Definition 3.3.5 and 3.3.6). Here, using their construction as the original learning problem, and wrapping it with our construction, we can strengthen our result and construct a learning problem that is not robustly learnable by *any* polynomial time learning algorithm, yet it has a classifier that is computationally robust. See Corollary 3.4.3 for more details.

Computational robustness without tamper detection. The computational robustness of the constructed classifier relies on sometimes detecting tampering attacks and not outputting a label. We give an

alternative construction for a setting that the classifier *always* has to output a label. We again use digital signatures and error correction codes as the main ingredient of our construction but in a different way. The main difference is that we have to repeat the signature multiple times to prevent the adversary from changing all of the signatures. The caveat of this construction is that it is no longer a wrapper around an arbitrary learning problem. See Construction 3.4.7 for more details.

3.2 Useful Tools

Here, we define the notions of one-way function, one-time signature and error correcting code.

Definition 3.2.1 (One-way function)**.** A function $f\colon \{0,1\}^* \to \{0,1\}^*$ is one-way if it can be computed in polynomial time and the inverse of f is hard to compute. Namely, there is a polynomial time algorithm M such that

$$\Pr[x \leftarrow \{0,1\}^n ; M(x) = f(x)] = 1$$

and for any polynomial time algorithm A there is a negligible function $\mathrm{negl}(\cdot)$ such that we have,

$$\Pr[x \leftarrow \{0,1\}^n ; y = f(x); f(\mathsf{A}(y)) = x] \leq \mathrm{negl}(|x|).$$

The assumption that one-way functions exist is standard and omnipresent in cryptography as a *minimal* assumption, as many cryptographic tasks imply the existence of OWFs [Goldreich, 2007, Katz and Lindell, 2014].

Definition 3.2.2 (One-time signature schemes)**.** A one-time signature scheme $S = (\mathrm{KGen}, \mathrm{Sign}, \mathsf{Verify})$ consists of three probabilistic polynomial-time algorithms as follows:

- $\mathrm{KGen}(1^\lambda)$[2] $\to (\mathsf{sk}, \mathsf{vk})$
- $\mathrm{Sign}(\mathsf{sk}, m) \to \sigma$
- $\mathsf{Verify}(\mathsf{vk}, \sigma, m) \to \{0,1\}$

which satisfy the following properties:

[2]By 1^λ we mean an string of bits of size λ that is equal to 1 at each location. Note that λ is the security parameter that controls the security of the scheme. As λ increases the task of finding a forgery for a signature becomes harder.

1. **Completeness:** For every m

$$\Pr[(sk, vk) \leftarrow \mathrm{KGen}(1^\lambda); \sigma \leftarrow \mathrm{Sign}(sk, m); \\ \mathsf{Verify}(\mathsf{vk}, \sigma, m) = 1] = 1.$$

2. **Unforgeability:** For every positive polynomial s, for every λ and every pair of circuits (A_1, A_2) with size $s(\lambda)$ the following probability is negligible in λ:

$$\Pr[(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathrm{KGen}(1^\lambda); \\ (m, st) \leftarrow A_1(1^\lambda, \mathsf{vk}); \\ \sigma \leftarrow \mathrm{Sign}(\mathsf{sk}, m); \\ (m', \sigma') \leftarrow A_2(1^\lambda, \mathsf{vk}, st, m, \sigma); \\ m \neq m' \wedge \mathsf{Verify}(\mathsf{vk}, \sigma', m') = 1] \leq \mathrm{negl}(\lambda).$$

Definition 3.2.3 (Error correction codes)**.** An error correction code with code rate α and error rate β consists of two algorithms Encode and Decode as follows.

- The encode algorithm Encode takes a Boolean string m and outputs a Boolean string c such that $|c| = |m|/\alpha$.
- The decode algorithm Decode takes a Boolean string c and outputs either $\perp$ or a Boolean string m. It holds that for all $m \in \{0,1\}^*$, $c = \mathsf{Encode}(m)$ and c' where $\mathsf{HD}(c, c') \leq \beta \cdot |c|$, it holds that $\mathsf{Decode}(c') = m$.

3.3 Defining Computational Risk and Computationally Robust Learning

Notation. We use calligraphic letters (e.g., $\mathcal{X}$) for sets and capital non-calligraphic letters (e.g., D) for distributions. By $d \leftarrow D$ we denote sampling d from D. For a randomized algorithm $R(\cdot)$, $y \leftarrow R(x)$ denotes the randomized execution of R on input x outputting y. A classification problem $\mathcal{P} = (\mathcal{X}, \mathcal{Y}, D, \mathcal{H})$ is specified by the following components: set $\mathcal{X}$ is the set of possible *instances*, $\mathcal{Y}$is the set of possible *labels*, $D \in \mathcal{D}$ is a *joint* distribution over $\mathcal{X} \times \mathcal{Y}$, and $\mathcal{H}$ is the space of hypothesis. For simplicity we work with problems that have a single distribution D (e.g., D is the distribution of labeled images from a data set like MNIST or

CIFAR-10). A learner L for problem P is an algorithm that takes a dataset $\mathcal{S} \leftarrow D^m$ as input and outputs a hypothesis $h \in \mathcal{H}$. We did not state the loss function explicitly, as we work with classification problems and use the zero-one loss by default. For a learning problem $\mathcal{P} = (\mathcal{X}, \mathcal{Y}, D, \mathcal{H})$, the *risk* or *error* of a hypothesis $h \in \mathcal{H}$ is $\mathsf{Risk}_{\mathcal{P}}(h) = \Pr_{(x,y)\leftarrow D}[h(x) \neq y]$. We are usually interested in learning problems $\mathcal{P} = (\mathcal{X}, \mathcal{Y}, D, \mathcal{H})$ with a specific metric d defined over $\mathcal{X}$ for the purpose of defining adversarial perturbations of bounded magnitude controlled by d. In that case, we might simply write $\mathcal{P} = (\mathcal{X}, \mathcal{Y}, D, \mathcal{H})$, but d is implicitly defined over $\mathcal{X}$. Finally, for a metric d over $\mathcal{X}$, we let $\mathsf{d}_b(x) = \{x' \mid \mathsf{d}(x, x') \leq b\}$ be the ball of radius b centered at x under the metric d. By default, we work with Hamming distance $\mathsf{HD}(x, x') = |\{i\colon x_i \neq x'_i\}|$, but our definitions can be adapted to any other metrics. We usually work with *families* of problems $\mathcal{P}_n$ where n determines the length of $x \in \mathcal{X}_n$ (and thus input lengths of $h \in \mathcal{H}_n, c \in \mathcal{C}_n, \mathsf{d}_n$). We sometimes use a special notation $\Pr[x \leftarrow X; E(x)]$ to define $\Pr_{x\leftarrow X}[E(x)]$ that is the probability of and event E over a random variable X. We also might use a combination of multiple random variables, for examples $\Pr[x \leftarrow X; y \leftarrow Y; E(x, y)]$ denotes the same thing as $\Pr_{x\leftarrow X, y\leftarrow Y}[E(x, y)]$. Order of sampling of X and Y matters Y might depend on X.

Allowing tamper detection. In this chapter, we expand the standard notion of hypotheses and allow $h \in \mathcal{H}$ to output a special symbol $\star$ as well (without adding $\star$ to $\mathcal{Y}$), namely we have $h\colon \mathcal{X} \mapsto \mathcal{Y} \cup \{\star\}$. This symbol can be used by the classifier h to denote "out of distribution" points, or any form of tampering, without outputting an actual label. In natural scenarios, $h(x) \neq \star$ when x is not an adversarially tampered instance. However, we allow this symbol to be output by h even in no-attack settings as long as its probability is small enough.

We follow the tradition of game-based security definitions in cryptography [Naor, 2003, Shoup, 2004, Goldwasser and Kalai, 2016, Rogaway and Zhang, 2018]. Games are the most common way that security is defined in cryptography. These games are defined between a challenger Chal and an adversary A. Consider the case of a signature scheme. In this case the challenger Chal is a signature scheme Π and an adversary A is given oracle access to the signing functionality (i.e. adversary can give a message m_i to the oracle and obtains the corresponding signature σ_i). Adversary A wins the game if he can provide a valid signature on a message that was not queried to the oracle. The security of the signature scheme is then defined informally as follows: any probabilistic polynomial time/size adversary A can win the game by probability that is bounded by a negligible $n^{-\omega(1)}$ function on the security parameter. We describe a security game for tampering adversaries with bounded tampering budget in HD, but the definition is more general and can be used for other adversary classes.

Definition 3.3.1 (Security game of adversarially robust learning)**.** Let $\mathcal{P}_n = (\mathcal{X}_n, \mathcal{Y}_n, D_n, \mathcal{H}_n)$ be a classification problem where the components are parameterized by n. Let L be a learning algorithm with sample complexity $m = m(n)$ for $\mathcal{P}_n$. Consider the following game between a challenger Chal, and an adversary A with tampering budget $b = b(n)$.

1. Chal samples m i.i.d. examples $\mathcal{S} \leftarrow D_n^m$ and gets hypothesis $h \leftarrow L(\mathcal{S})$ where $h \in \mathcal{H}_n$.
2. Chal then samples a test example $(x, y) \leftarrow D_n$ and sends (x, y) to the adversary A.
3. Having oracle access (or oracle gates, in case of circuits) to hypothesis h and a sampler for D_n, the adversary obtains the adversarial instance $x' \leftarrow \mathsf{A}^{h(\cdot), D_n}(x)$ and outputs x'.

Winning conditions: In case $x = x'$, the adversary A wins if $h(x) \neq y$,[3] and in case $x \neq x'$, the adversary wins if all the following hold:

1. $\mathsf{HD}(x, x') \leq b$,
2. $h(x') \neq y$, and
3. $h(x') \neq \star$.

Why separating winning conditions for $x = x'$ from $x \neq x'$? One might wonder why we separate the winning condition for the two cases of $x = x'$ and $x \neq x'$. The reason is that $\star$ is supposed to capture tamper detection. So, if the adversary does not change x and the hypothesis outputs $h(x) = \star$, this is an error, and thus should contribute to the risk. More formally, when we evaluate risk, we have $\mathsf{Risk}_{\mathcal{P}}(h) = \Pr_{(x,y)\leftarrow D}[h(x) \neq y]$, which implicitly means that outputting $\star$ contributes to the risk. However, if adversary's perturbs to $x' \neq x$ leads to $h(x') = \star$, it means the adversary has *not* succeeded in its attack, because the tampering is detected. In fact, if we simply require the other 3 conditions to let adversary win, the notion of "adversarial risk" (see Definition 3.3.2) might be even less than the normal risk, which is counter intuitive.

Alternative definitions of winning for the adversary. The winning condition for the adversary could be defined in other ways as well. In our Definition 3.3.1, the adversary wins if $\mathsf{d}(x, x') \leq b$ and $h(x') \neq y$. This condition is inspired by the notion of corrupted input [Feige et al., 2015], is extended to metric spaces in [Madry et al., 2018], and is used in and many subsequent works. An alternative definition for adversary's goal, formalized in [Diochnos et al., 2018b] and used in [Gilmer et al., 2018b, Diochnos et al., 2018b, Bubeck et al., 2018a, Degwekar and Vaikuntanathan, 2019] requires $h(x')$ to be different from the true label of x' (rather than x). This condition requires the misclassification of x', and thus, x' would belong to the "error-region"

[3]Note that, if $h(x) \neq y$, without loss of generality, the adversary A can output $x' = x$

of h. Namely, if we let $c(x) = y$ be the ground truth function, the error-region security game requires $h(x') \neq c(x')$. Another stronger definition of adversarial risk is given by Suggala et al. [2018b] in which the requirement condition requires both conditions: (1) the ground truth should not change $c(x) = c(x')$, and that (2) x' is misclassified. For natural distributions like images or voice, where the ground truth is robust to small perturbations, all these three definitions for adversary's winning are equivalent.

Stronger attack models. In the attack model of Definition 3.3.1, we only provided the label y of x to the adversary and also give her the sample oracle from D_n. A stronger attacker can have access to the "concept" function $c(x)$ which is sampled from the distribution of y given x (according to D_n). This concept oracle might not be efficiently computable, even in scenarios that D_n is efficiently samplable. In fact, even if D_n is not efficiently samplable, just having access to a large enough pool of i.i.d. sampled data from D_n is enough to run the experiment of Definition 3.3.1. In alternative winning conditions (e.g., the error-region definition) for Definition 3.3.1 discussed above, it makes more sense to also include the ground truth concept oracle $c(\cdot)$ given as oracle to the adversary, as the adversary needs to achieve $h(x') \neq c(x')$. Another way to strengthen the power of adversary is to give him non-black-box access to the components of the game (see Papernot et al. [2017]). In definition 3.3.1, by default, we model adversaries who have black-box access to $h(\cdot), D_n$, but one can define non-black-box (white-box) access to each of $h(\cdot), D_n$, if they are polynomial size objects.

Diochnos et al. [2018b] focused on bounded perturbation adversaries that are unbounded in their running time and formalized notions of of adversarial risk for a given hypothesis h with respect to the b-perturbing adversaries. Using Definition 3.3.1, in Definition 3.3.2, we retrieve the notions of standard risk, adversarial risk, and its (new) computational variant.

Definition 3.3.2 (Adversarial risk of hypotheses and learners)**.** Suppose L is a learner for a problem $\mathcal{P} = (\mathcal{X}, \mathcal{Y}, D, \mathcal{H})$. For a class of attackers $\mathcal{A}$ we define

$$\mathsf{AdvRisk}_{\mathcal{P},\mathcal{A}}(L) = \sup_{\mathsf{A} \in \mathcal{A}} \Pr[\mathsf{A} \text{ wins}]$$

where the winning is in the experiment of Definition 3.3.1. When the attacker A is fixed, we simply write $\mathsf{AdvRisk}_{\mathcal{P},\mathsf{A}}(L) = \mathsf{AdvRisk}_{\mathcal{P},\{\mathsf{A}\}}(L)$. For a trivial attacker I who outputs $x' = x$, it holds that $\mathsf{Risk}_{\mathcal{P}}(L) = \mathsf{AdvRisk}_{\mathcal{P},I}(L)$. When $\mathcal{A}$ includes attacker that are *only* bounded by b perturbations, we use notation $\mathsf{AdvRisk}_{\mathcal{P},b}(L) = \mathsf{AdvRisk}_{\mathcal{P},\mathcal{A}}(L)$, and when the adversary is further restricted to all s-size (oracle-aided) circuits, we use notation $\mathsf{AdvRisk}_{\mathcal{P},b,s}(L) = \mathsf{AdvRisk}_{\mathcal{P},\mathcal{A}}(L)$. When L is a learner that outputs a fixed hypothesis h, by substituting h with L, we obtain the following similar notions for h, which will be denoted as $\mathsf{Risk}_{\mathcal{P}}(h)$, $\mathsf{AdvRisk}_{\mathcal{P},\mathcal{A}}(h)$, $\mathsf{AdvRisk}_{\mathcal{P},b}(h)$, and $\mathsf{AdvRisk}_{\mathcal{P},b,s}(h)$.

Definition 3.3.3 (Computationally robust learners and hypotheses)**.** Let $\mathcal{P}_n = (\mathcal{X}_n, \mathcal{Y}_n, D_n, \mathcal{H}_n)$ be a family of classification parameterized by n. We say that a learning algorithm L is a *computationally robust* learner with risk at most $R = R(n)$ against $b = b(n)$-perturbing adversaries, if for any polynomial $s = s(n)$, there is a negligible function $\mathrm{negl}(n) = n^{-\omega(1)}$ such that

$$\mathsf{AdvRisk}_{\mathcal{P}_n,b,s}(L) \leq R(n) + \mathrm{negl}(n).$$

Note that the size of circuit used by the adversary controls its computational power and that is why we are enforcing it to be a polynomial. Again, when L is a learner that outputs a fixed hypothesis h_n for each n, we say that the family h_n is a computationally robust hypothesis with risk at most $R = R(n)$ against $b = b(n)$-perturbing adversaries, if L is so. In both cases, we might simply say that L (or h) has *computational risk* at most $R(n)$.

Remark 3.3.4 (Alternative definition without the negligible term for concrete adversary runtime)**.** We remark that, when the class of adversary is a finite set, and when we work with a concrete setting of parameter (as opposed to the asymptotic setting of Definition 3.3.2) one can opt to work with concrete bounds and a version that drops the negligible probability negl on the right hand side of the inequality and asks for the probability of winning to be simply stated as $\mathsf{AdvRisk}_{\mathcal{P}_n,b,s}(L) \leq R(n)$ for s-sized oracle-aided circuit adversaries or s-time oracle-aided Turing machines. However, in the asymptotic setting, one can work with very large polynomials for small security parameters, in which case there is little difference between information theoretic adversaries versus computationally bounded ones. In that case, the negligible additive term will subsume any large advantage that such adversaries might have for small security parameters. In this chapter, we opt to work with the above asymptotic definition together with the negligible additive term. Moreover, the negligible probability usually comes up in computational reductions, and hence it simplifies the statement of our theorems, but we emphasize that both forms of the definition of computational risk (for concert as well as asymptotic settings) are equally appealing and valid on their own.

PAC learning under computationally bounded tampering adversaries. Recently, several works studied generalization under adversarial perturbations from a theoretical perspective [Bubeck et al., 2018b, Cullina et al., 2018, Feige et al., 2018, Attias et al., 2018, Khim and Loh, 2018, Yin et al., 2018b, Montasser et al., 2019, Diochnos et al., 2019], and hence they implicitly or explicitly revisited the "probably approximately corect" (PAC) learning framework of Valiant [2013] under adversarial perturbations. Here we comment that, one can derive variants of those definitions for *computationally bounded* attackers, by limiting their adversaries as done in our Definition 3.3.3. In particular, we call a learner L an (ε, δ) PAC learner for a

problem $\mathcal{P}$ and computationally bounded b-perturbing adversaries, if with probability $1-\delta$, L outputs a hypothesis h that has computational risk at most ε.

Bellow we formally define the notion of robust-hard learning problems which captures the inherent vulnerability of a learning problem to adversarial attacks due to computational limitations of the learning algorithm. This definition are implicit in works of [Degwekar and Vaikuntanathan, 2019, Bubeck et al., 2018c]. In Section 3.4, we use these robust-hard problems to construct learning problems that are inherently non-robust in presence of computationally unbounded adversaries but have robust classifiers against computationally bounded adversaries.

Definition 3.3.5 (Robust-hard learning problems)**.** A learning problem $\mathcal{P}_n = (\mathcal{X}_n, \mathcal{Y}_n, D_n, \mathcal{H}_n)$ is robust-hard w.r.t budget $b(n)$ if for any learning algorithm L that runs in poly(n) we have

$$\mathsf{AdvRisk}_{\mathcal{P}_n, b(L)} \geq 1 - \mathrm{negl}(n).$$

Theorem 3.3.6 (Degwekar and Vaikuntanathan [2019], Bubeck et al. [2018c])**.** *There exist a Learning problem $\mathcal{P}_n = (\mathcal{X}_n, \mathcal{Y}_n, D_n, \mathcal{H}_n)$ and a sub-linear budget $b(n)$ such that $\mathcal{P}_n$ is robust-hard w.r.t b unless one-way functions do not exist. (See appendix for the definition of one-way functions)*

Discussion on falsifiability of computational robustness. If the learner L is polynomial time, and that the distribution D_n is samplable in polynomial time (e.g., by sampling y first and then using a generative model to generate x for y), then the the computational robustness of learners as defined based on Definitions 3.3.3 and 3.3.1 is a "falsifiable" notion of security as defined by Naor [2003]. Namely, if an adversary claims that it can break the computational robustness of the learner L, it can prove so in polynomial time by participating in a challenge-response game and winning in this game with a noticeable (non-negligible) probability more than $R(n)$. This feature is due to the crucial property of the challenger in Definition 3.3.1 that is a polynomial time algorithm itself, and thus can be run efficiently. Not all security games have efficient challengers (e.g., see Pandey et al. [2008]).

3.4 From Computational Hardness to Computational Robustness

In this section, we will first prove our main result that shows the existence of a learning problem with classifiers that are only computationally robust. We first prove our result by starting from *any* hypothesis that is vulnerable to adversarial examples; e.g., this could be any of the numerous algorithms shown to be

susceptible to adversarial perturbations. Our constructions use error correction codes and cryptographic signatures. For definitions of these notions refer to section 3.2.

3.4.1 Computational Robustness with Tamper Detection

Our first construction uses hypothesis with tamper detection (i.e, output $\star$ capability). Our construction is based on cryptographic signature schemes with short (polylogarithmic) signatures.

Construction 3.4.1. Let $\mathsf{Q} = (\{0,1\}^d, \mathcal{Y}, D, \mathcal{H})$ be a learning problem and $h \in \mathcal{H}$ a classifier for Q. We construct a family of learning problems $\mathcal{P}_n$ (based on the fixed problem Q) with a family of classifiers h_n. In our construction we use signature scheme $(\mathrm{KGen}, \mathrm{Sign}, \mathsf{Verify})$ for which the bit-length of vk is λ and the bit-length of signature is $\ell(\lambda) = \mathrm{polylog}(\lambda)$ [4]. We also use an error correction code $(\mathsf{Encode}, \mathsf{Decode})$ with code rate $\mathsf{cr} = \Omega(1)$ and error rate $\mathsf{er} = \Omega(1)$.

1. The space of instances for $\mathcal{P}_n$ is $\mathcal{X}_n = \{0,1\}^{n+d+\ell(n)}$.
2. The set of labels is $\mathcal{Y}_n = \mathcal{Y}$.
3. The distribution D_n is defined by the following process: first sample $(x, y) \leftarrow D$, $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathrm{KGen}(1^{n \cdot \mathsf{cr}})$, $\sigma \leftarrow \mathrm{Sign}(\mathsf{sk}, x)$, then encode $[\mathsf{vk}] = \mathsf{Encode}(\mathsf{vk})$ and output $((x, \sigma, [\mathsf{vk}]), y)$.
4. The classifier $h_n \colon \mathcal{X}_n \to \mathcal{Y}_n$ is defined as

$$h_n(x, \sigma, [\mathsf{vk}]) = \begin{cases} h(x) & \text{if } \mathsf{Verify}\left(\mathsf{Decode}([\mathsf{vk}]), x, \sigma\right), \\ \star & \textit{otherwise.} \end{cases}$$

Theorem 3.4.2. *For family $\mathcal{P}_n$ of Construction 3.4.1, the family of classifiers h_n is computationally robust with adversarial risk at most $\mathsf{Risk}_Q(h) = \alpha$ against adversaries with budget $\mathsf{er} \cdot n$. (Recall that er is the error rate of the error correction code.) On the other hand h_n is not robust against information theoretic adversaries of budget $b + \ell(n)$, if h itself is not robust to b perturbations:*

$$\mathsf{AdvRisk}_{\mathcal{P}_n, b+\ell(n)}(h_n) \geq \mathsf{AdvRisk}_{\mathsf{Q},b}(h).$$

Theorem 3.4.2 means that, the h_n is computationally robust for adversarial budget as large as $\Omega(n)$ (if we choose a code with constant error correction rate) while it has small information theoretic adversarial robustness for budget value as small as $b + \mathrm{polylog}(n) \leq \mathrm{polylog}(n)$ (note that b is a constant here) if we choose a signature scheme with short signatures of poly-logarithmic length.

[4] Such signatures exist assuming exponentially hard one-way functions [Rompel, 1990].

Before proving Theorem 3.4.2 we state the following corollary about robust-hard learning problems.

Corollary 3.4.3. *If the underlying problem Q in Construction 3.4.1 is robust-hard w.r.t sublinear budget $b(n)$, then for* any *polynomial learning algorithm L for $\mathcal{P}_n$ we have*

$$\mathsf{AdvRisk}_{\mathcal{P}_n, b+\ell(n)}(L) \geq 1 - \mathrm{negl}(n).$$

On the other hand, the family of classifiers h_n for $\mathcal{P}_n$ is computationally robust with risk at most α against adversaries with linear budget.

The above corollary follows from Theorem 3.4.2 and definition of robust-hard learning problems. The significance of this corollary is that it provides an example of a learning problem that could not be learnt robustly with *any* polynomial time algorithm. However, the same problem has a classifier that is robust against computationally bounded adversaries. This construction uses a robust-hard learning problem that is proven to exist based on cryptographic assumptions [Bubeck et al., 2018c, Degwekar and Vaikuntanathan, 2019]. Now we prove Theorem 3.4.2.

Proof. (of Theorem 3.4.2) We first prove the following claim about the risk of h_n.

Claim 3.4.4. *For problem $\mathcal{P}_n$ we have*

$$\mathsf{Risk}_{\mathcal{P}_n}(h_n) = \mathsf{Risk}_{\mathsf{Q}}(h) = \alpha.$$

Proof. The proof follows from the completeness of the signature scheme. We have,

$$\begin{aligned}\mathsf{Risk}_{\mathcal{P}_n}(h_n) &= \Pr[((x, \sigma, [\mathsf{vk}]), y) \leftarrow D_n;\ h_n(x, \sigma, [\mathsf{vk}]) \neq y] \\ &= \Pr[(x, y) \leftarrow D;\ h(x) \neq y] = \mathsf{Risk}_{\mathsf{Q}}(h).\end{aligned}$$

□

Now we prove the computational robustness of h_n.

Claim 3.4.5. *For family $\mathcal{P}_n$, and for any polynomial $s(\cdot)$ there is a negligible function* negl *such that for all $n \in \mathbb{N}$*

$$\mathsf{AdvRisk}_{\mathcal{P}_n, \mathsf{er}\cdot n, s}(h_n) \leq \alpha + \mathrm{negl}(n).$$

Proof. Let $\mathsf{A}_{\{n\in\mathbb{N}\}}$ be the family of circuits maximizing the adversarial risk for h_n for all $n \in \mathbb{N}$. We build a sequence of circuits $\mathsf{A}^1_{\{n\in\mathbb{N}\}}$, $\mathsf{A}^2_{\{n\in\mathbb{N}\}}$ such that A^1_n and A^2_n are of size at most $s(n) + poly(n)$. A^1_n just samples

a random $(x, y) \leftarrow D$ and outputs (x, y). A_n^2 gets x, σ and vk, calls A_n to get $(x', \sigma', \mathsf{vk}') \leftarrow \mathsf{A}_n((x, \sigma, [\mathsf{vk}]), y)$ and outputs (x', σ'). Note that A_n^2 can provide all the oracles needed to run A_n if the sampler from D, h and c are all computable by a circuit of polynomial size. Otherwise, we need to assume that our signature scheme is secure with respect to those oracles and the proof will follow. We have,

$$\begin{aligned}\mathsf{AdvRisk}_{\mathcal{P}_n,\mathsf{er}\cdot n,s}(h_n) = \Pr[((x,\sigma,[\mathsf{vk}]),y) \leftarrow D_n;\ (x',\sigma',\mathsf{vk}') \leftarrow A((x,\sigma,[\mathsf{vk}]),y));\\ (x',\sigma',\mathsf{vk}') \in \mathsf{HD}_{\mathsf{er}\cdot n}(x,\sigma,[\mathsf{vk}]) \wedge h_n(x',\sigma',\mathsf{vk}') \neq \star \wedge h_n(x',\sigma',\mathsf{vk}') \neq y].\end{aligned}$$

Note that $(x', \sigma', \mathsf{vk}') \in \mathsf{HD}_{\mathsf{er}\cdot n}(x, \sigma, [\mathsf{vk}])$ implies that $\mathsf{Decode}(\mathsf{vk}') = \mathsf{vk}$ based on the error rate of the error correcting code. Also $h_n(x', \sigma', \mathsf{vk}') \neq \star$ implies that σ' is a valid signature for x' under verification key vk. Therefore, we have,

$$\begin{aligned}&\mathsf{AdvRisk}_{\mathsf{er}\cdot n,s}(h_n)\\ &\leq \Pr[(\mathsf{sk},\mathsf{vk}) \leftarrow \mathrm{KGen}(1^n);\ (x,y) \leftarrow \mathsf{A}_1(1^n);\ \sigma \leftarrow \mathrm{Sign}(\mathsf{sk},x);\ (x',\sigma') \leftarrow \mathsf{A}_2(x,\sigma,\mathsf{vk});\\ &\qquad \mathsf{Verify}(\mathsf{vk},x',\sigma') \wedge h_n(x',\sigma',[\mathsf{vk}]) \neq y]\\ &\leq \Pr[(\mathsf{sk},\mathsf{vk}) \leftarrow \mathrm{KGen}(1^n);\ x \leftarrow \mathsf{A}_1(1^n);\ \sigma \leftarrow \mathrm{Sign}(\mathsf{sk},x);\ (x',\sigma') \leftarrow \mathsf{A}_2(x,\sigma,\mathsf{vk});\\ &\qquad \mathsf{Verify}(\mathsf{vk},x',\sigma') \wedge x' \neq x] + \mathsf{Risk}_{\mathcal{P}_n}(h_n).\end{aligned}$$

Thus, by the unforgeability of the one-time signature scheme we have

$$\mathsf{AdvRisk}_{\mathcal{P}_n,\mathsf{er}\cdot n,s}(h_n) \leq \mathsf{Risk}_{\mathcal{P}_n}(h_n) + \mathrm{negl}(n),$$

which by Claim 3.4.4 implies

$$\mathsf{AdvRisk}_{\mathsf{er}\cdot n,s}(h_n) \leq \alpha + \mathrm{negl}(n).$$

□

Now we show that h_n is not robust against computationally unbounded attacks.

Claim 3.4.6. *For family $\mathcal{P}_n$ and any $n, b \in \mathbb{N}$ we have*

$$\mathsf{AdvRisk}_{\mathcal{P}_n,b+\ell(n)}(h_n) \ \geq\ \mathsf{AdvRisk}_{\mathsf{Q},b(h)}.$$

Proof. For any $((x, \sigma, [\mathsf{vk}]), y)$ define $A(x, \sigma, [\mathsf{vk}]) = (x', \sigma', [\mathsf{vk}])$ where x' is the closes point to x where $h(x) \neq y$ and σ' is a valid signature such that $\mathsf{Verify}(\mathsf{vk}, x^*, \sigma') = 1$. Based on the fact that the size of signature is $\ell(n)$,

we have $\mathsf{HD}(A(x,\sigma,[\mathsf{vk}]),(x,\sigma,[\mathsf{vk}])) \le \ell(n) + \mathsf{HD}(x,x')$. Also, it is clear that $h_n(A(x,\sigma,[\mathsf{vk}])) \neq \star$ because σ' is a valid signature. Also, $h_n(A(x,\sigma,[\mathsf{vk}])) \neq c_n(A(x,\sigma,[\mathsf{vk}]))$. Therefore we have

$$
\begin{aligned}
&\mathsf{AdvRisk}_{\mathcal{P}_n, b+\ell(n)}(h_n) \\
&= \Pr[((x,\sigma,[\mathsf{vk}]),y) \leftarrow D_n; \exists (x',\sigma') \in \mathsf{HD}_{b+\ell(n)}(x,\sigma), h(x') \neq y \wedge h(x') \neq \star \wedge \mathsf{Verify}(\mathsf{vk},\sigma',x')] \\
&\ge \Pr[((x,\sigma,[\mathsf{vk}]),y) \leftarrow D_n; \exists x' \in \mathsf{HD}_b(x), h(x') \neq y \wedge h(x') \neq \star] \\
&= \mathsf{AdvRisk}_{\mathsf{Q},b}(h).
\end{aligned}
$$

□

This concludes the proof of Theorem 3.4.2. □

3.4.2 Computational Robustness without Tamper Detection

The following theorem shows an alternative construction that is incomparable to Construction 3.4.1, as it does not use any tamper detection. On the down side, the construction is not defined with respect to an arbitrary (vulnerable) classifier of a natural problem.

Construction 3.4.7 (Computational robustness without tamper detection)**.** Let D be a distribution over $\{0,1\}^{\mathsf{cr}\cdot n} \times \{0,1\}$ with a balanced "label" bit: $\Pr_{(x,y)\leftarrow D}[y=0] = 1/2$. We construct a family of learning problems $\mathcal{P}_n$ with a family of classifiers h_n. In our construction we use a signature scheme $(\mathrm{KGen}, \mathrm{Sign}, \mathsf{Verify})$ for which the bit-length of vk is λ and the bit-length of signature is $\ell(\lambda) = \mathrm{polylog}(\lambda)$ and an error correction code $(\mathsf{Encode}, \mathsf{Decode})$ with code rate $\mathsf{cr} = \Omega(1)$ and error rate $\mathsf{er} = \Omega(1)$.

1. The space of instances for $\mathcal{P}_n$ is $\mathcal{X}_n = \{0,1\}^{2n+n\cdot\ell(n)}$.
2. The set of labels is $\mathcal{Y}_n = \{0,1\}$.
3. The distribution D_n is defined as follows: first sample $(x,y) \leftarrow D$, then sample $(\mathsf{sk},\mathsf{vk}) \leftarrow \mathrm{KGen}(1^{n\cdot\mathsf{cr}})$ and compute $[\mathsf{vk}] = \mathsf{Encode}(\mathsf{vk})$. Then compute $[x] = \mathsf{Encode}(x)$. If $y = 0$ sample a random $\sigma \leftarrow \{0,1\}^{\ell(n)}$ that is not a valid signature of x w.r.t vk. Then output $(([x],\sigma^n,[\mathsf{vk}]),0)$. Otherwise compute $\sigma \leftarrow \mathrm{Sign}(\mathsf{sk},x)$ and output $(([x],\sigma^n,[\mathsf{vk}]),1)$.
4. The classifier $h_n \colon \mathcal{X}_n \to \mathcal{Y}_n$ is defined as

$$
h_n(x',\sigma_1,\ldots,\sigma_n,\mathsf{vk}') = \begin{cases} 1 & \text{if } \exists i \in [n]; \mathsf{Verify}\left(\mathsf{Decode}(\mathsf{vk}'),\mathsf{Decode}(x'),\sigma_i\right), \\ 0 & \textit{otherwise.} \end{cases}
$$

Theorem 3.4.8. *For family $\mathcal{P}_n$ of Construction 3.4.7, the family of classifiers h_n has risk 0 and is computationally robust with risk at most 0 against adversaries of budget $\mathsf{er} \cdot n$. On the other hand h_n is not robust against information theoretic adversaries of budget $\ell(n)$:*

$$\mathsf{AdvRisk}_{\mathcal{P}_n,\ell(n)}(h_n) \geq 1/2.$$

Note that reaching adversarial risk 1/2 makes the classifier's decisions meaningless as a random coin toss achieves this level of accuracy.

Proof. First it is clear that for problem $\mathcal{P}_n$ we have $\mathsf{Risk}_{\mathcal{P}_n}(h_n) = 0$. Now we prove the computational robustness of h_n.

Claim 3.4.9. *For family $\mathcal{P}_n$, and for any polynomial $s(\cdot)$ there is a negligible function* negl *such that for all $n \in \mathbb{N}$*

$$\mathsf{AdvRisk}_{\mathcal{P}_n,\mathsf{er}\cdot n,s}(h_n) \leq \mathrm{negl}(n).$$

Proof. Similar to proof of Claim 3.4.5 we prove this based on the security of the signature scheme. Let $\mathsf{A}_{\{n\in\mathbb{N}\}}$ be the family of circuits maximizing the adversarial risk for h_n for all $n \in \mathbb{N}$. We build a sequence of circuits $\mathsf{A}^1_{\{n\in\mathbb{N}\}}$ and $\mathsf{A}^2_{\{n\in\mathbb{N}\}}$ such that A^1_n and A^2_n are of size at most $s(n) + \mathrm{poly}(n)$. A^1_n just asks the signature for $0^{\mathsf{cr}\cdot n}$. A^2_n gets vk and does the following: It first samples $(x, y) \leftarrow D$, computes encodings $[x] = \mathsf{Encode}(x)$ and $[\mathsf{vk}] = \mathsf{Encode}(\mathsf{vk})$ and if $y = 0$, it samples a random $\sigma \leftarrow \{0,1\}^{\ell(n)}$ then calls A_n on input $([x], \sigma^n, [\mathsf{vk}])$ to get $(x', (\sigma_1, \ldots, \sigma_n), \mathsf{vk}') \leftarrow \mathsf{A}_n(([x], \sigma^n, [\mathsf{vk}]), y)$. Then it checks all σ_i's and if there is any of them that $\mathsf{Verify}(\mathsf{vk}, \sigma_i, x) = 1$ it outputs (x, σ_i), otherwise it aborts and outputs $\perp$. If $y = 0$ it aborts and outputs $\perp$. Note that A^2_n can provide all the oracles needed to run A_n if the sampler from D, h and c are all computable by a circuit of polynomial size. Otherwise, we need to assume that our signature scheme is secure with respect to those oracles and the proof will follow. We have,

$$\begin{aligned}
&\mathsf{AdvRisk}_{\mathcal{P}_n,\mathsf{er}\cdot n,s}(h_n) \\
&= \Pr[(([x], \sigma^n, [\mathsf{vk}]), y) \leftarrow D_n;\ (x', (\sigma_1, \ldots, \sigma_n), \mathsf{vk}') \leftarrow A_n(([x], \sigma^n, [\mathsf{vk}]), y)); \\
&\qquad (x', (\sigma_1, \ldots, \sigma_n), \mathsf{vk}') \in \mathsf{HD}_{\mathsf{er}\cdot n}([x], \sigma^n, [\mathsf{vk}]) \wedge h_n(x', (\sigma_1, \ldots, \sigma_n), \mathsf{vk}') \neq y].
\end{aligned}$$

Because of the error rate of the error correcting code, $(x', (\sigma_1, \ldots, \sigma_n), \mathsf{vk}') \in \mathsf{HD}_{\mathsf{er}\cdot n}(x, \sigma^n, [\mathsf{vk}])$ implies that $\mathsf{Decode}(\mathsf{vk}') = \mathsf{vk}$ and $\mathsf{Decode}(x') = x$. Also $h_n(x', (\sigma_1, \ldots, \sigma_n), \mathsf{vk}') \neq y$ implies that $y = 0$. This is because if $y = 1$, the adversary has to make all the signatures invalid which is impossible with tampering budget $\mathsf{cr} \cdot n$. Therefore y must be 1 and one of the signatures in $(\sigma_1, \ldots, \sigma_n)$ must pass the verification because the

prediction of h_λ should be 1. Therefore we have

$$\begin{aligned}\mathsf{AdvRisk}_{\mathcal{P}_n,\mathsf{er}\cdot n,s}(h_n) &\leq \Pr[((x,\sigma^n,[\mathsf{vk}]),y) \leftarrow D_n;\ (x',(\sigma_1,\ldots,\sigma_n),\mathsf{vk}') \leftarrow A((x,\sigma,[\mathsf{vk}]),y));\\ &\qquad y=0 \wedge \exists i \mathsf{Verify}(\mathsf{vk},\sigma_i,x)]\\ &\leq \Pr[(\mathsf{sk},\mathsf{vk}) \leftarrow \mathrm{KGen}(1^n);\ 0^{\mathsf{cr}\cdot n} \leftarrow \mathsf{A}_1(1^n,\mathsf{vk});\ \sigma \leftarrow \mathrm{Sign}(\mathsf{sk},0^{\mathsf{cr}\cdot n});\\ &\qquad (x,\sigma_i) \leftarrow \mathsf{A}_2(\mathsf{vk});\ \mathsf{Verify}(\mathsf{vk},x,\sigma_i)]\end{aligned}$$

Thus, by the unforgeability of the one-time signature scheme we have

$$\mathsf{AdvRisk}_{\mathcal{P}_n,\mathsf{er}\cdot n,s}(h_n) \leq \mathrm{negl}(n).$$

□

Now we show that h_n is not robust against computationally unbounded attacks.

Claim 3.4.10. *For family $\mathcal{P}_n$ and any $n \in \mathbb{N}$ we have*

$$\mathsf{AdvRisk}_{\mathcal{P}_n,\ell(n)}(h_n) = 0.5.$$

Proof. For any $(([x],\sigma^n,[\mathsf{vk}]),y)$ define $A([x],\sigma^n,[\mathsf{vk}])$ as follows: If $y=1$, A does nothing and outputs $([x],\sigma^n,[\mathsf{vk}])$. If $y=0$, A search all possible signatures to find a signature σ' such that $\mathsf{Verify}(\mathsf{vk},\sigma',x)=1$. It then outputs $([x],(\sigma',\sigma^{n-1}),[\mathsf{vk}])$. Based on the fact that the size of signature is $\ell(n)$, we have $\mathsf{HD}((x,(\sigma',\sigma^{n-1}),[\mathsf{vk}]),(x,\sigma^n,[\mathsf{vk}])) \leq \ell(n)$. Also, it is clear that $h_n(x,(\sigma',\sigma^{n-1}),[\mathsf{vk}])=1$ because the first signature is always a valid signature. Therefore we have

$$\begin{aligned}\mathsf{AdvRisk}_{\mathcal{P}_n,\ell(n)}(h_n) &\geq \Pr[(([x],\sigma^n,[\mathsf{vk}]),y) \leftarrow D_n; h(A(([x],\sigma^n,[\mathsf{vk}]))) \neq y]\\ &= \Pr[(([x],\sigma^n,[\mathsf{vk}]),y) \leftarrow D_n; 1 \neq y]\\ &= 0.5.\end{aligned}$$

□

This concludes the proof of Theorem 3.4.8. □

3.5 Average-Case Hardness of NP from Computational Robustness

In this section, we show a converse result to those in Section 3.4, going from useful computational robustness to deriving computational hardness. Namely, we show that if for there is a learning problem whose computational risk is noticeably more than its information theoretic risk, then **NP** is hard even on average.

Definition 3.5.1 (Hard samplers for **NP**)**.** For the following definition, A Boolean formula ϕ over some Boolean variables $x_1, \ldots, x_k$ is satisfiable, if there is an assignment to $x_1, \ldots, x_k \in \{0,1\}$, for which ϕ evaluates to 1 (i.e, TRUE). We use some standard canonical encoding of such Boolean formulas and fix it, and we refer to $|\phi|$, the size of ϕ, as the bit-length of this representation for formula ϕ. Let SAT be the language/set of all satisfiable Boolean formulas. Suppose $S(1^n, r)$ is a polynomial time randomized algorithm that takes 1^n and randomness r, runs in time $\mathrm{poly}(n)$, and outputs Boolean formulas of size $\mathrm{poly}(n)$. We call S a *hard (instance) sampler* for **NP** if,

1. For a negligible function negl it holds that $\Pr_{\phi \leftarrow S}[\phi \in \mathrm{SAT}] = 1 - \mathrm{negl}(n)$.
2. For every poly-size circuit A, there is a negligible function negl, such that

$$\Pr_{\phi \leftarrow S, t \leftarrow \mathsf{A}(\phi)}[\phi(t) = 1] = \mathrm{negl}(n).$$

The following theorem is stated for computationally robust learning, but the same proof holds for computationally robust hypotheses as well.

Theorem 3.5.2 (Hardness of **NP** from computational robustness)**.** *Let $\mathcal{P}_n = (\mathcal{X}_n, \mathcal{Y}_n, D_n, \mathcal{H}_n)$ be a learning problem. Suppose there is a (uniform) learning algorithm L for $\mathcal{P}_n$ such that:*

1. *L is* computationally *robust with risk at most α under $b(n)$-perturbations.*
2. *$\mathsf{AdvRisk}_{\mathcal{P}_n, b(n)}(L) \geq \beta(n)$; i.e., information-theoretic adversarial risk of L is at least $\beta(n)$.*
3. *$\beta(n) - \alpha \geq \varepsilon$ for $\varepsilon = 1/\mathrm{poly}(n)$.*
4. *D_n is efficiently samplable by algorithm S.*
5. *For any $x, x' \in \mathcal{X}_n$ checking $\mathsf{d}(x, x') \leq b(n)$ is possible in polynomial time.*

Then, there is a hard sampler for **NP**.

Before proving Theorem 3.5.2, we recall a useful lemma. The same proof of amplification of (weak to strong) one-way functions by Yao [1982] and described in [Goldreich, 2007], or the parallel repetition of verifiable puzzles [Canetti et al., 2005, Holenstein and Schoenebeck, 2011] can be used to prove the following lemma.

Lemma 3.5.3 (Amplification of verifiable puzzles)**.** *Suppose S is a distribution over Boolean formulas such that for every poly-size adversary A, for sufficiently large n, it holds that solving the puzzles generated by S are weakly hard. Namely, $\Pr_{\phi \leftarrow S(1^n, r_1)}[\phi(t) = 1; t \leftarrow \mathsf{A}(\phi)] \leq \varepsilon$ for $\varepsilon = 1/\operatorname{poly}(n)$. Then, for any polynomial-size adversary A, there is a negligible function negl, such that the probability that A can simultaneously solve all of $k = n/\varepsilon$ puzzles $\phi_1, \ldots, \phi_k$ that are independently sampled from S is at most $\operatorname{negl}(n)$.*

Proof. (of Theorem 3.5.2.) First consider the following sampler S_1. (We will modify S_1 later on).

1. Sample m examples $\mathcal{S} \leftarrow D_n^m$.
2. Run L to get $h \leftarrow L(\mathcal{S})$.
3. Sample another $(x, y) \leftarrow D_n$
4. Using the Cook-Levin reduction, get a Boolean formula $\phi = \phi_{h,x,y}$ such that $\phi \in \text{SAT}$, if (1) $\mathsf{d}(x', x') \leq b(n)$ and (2) $h(x') \neq y$. This is possible because using h, x, y, both conditions (1) and (2) are efficiently checkable.
5. Output ϕ.

By the assumptions of Theorem 3.5.2, it holds that $\Pr_{\phi \leftarrow S_1}[\phi \in \text{SAT}] \geq \beta(n)$ while for any poly-size algorithm A, it holds that $\Pr_{\phi \leftarrow S_1, t \leftarrow \mathsf{A}(\phi)}[\phi(t) = 1] \leq \alpha$. So, S_1 almost gets the conditions of a hard sampler for **NP**, but only with a weak sense.

Using standard techniques, we can amplify the ε-gap between $\alpha, \beta(n)$. The algorithm S_2 works as follows. (This algorithm assumes the functions $\alpha, \beta(n)$ are efficiently computable, or at least there is an efficiently computable threshold $\tau \in [\alpha + 1/\operatorname{poly}(n), \beta(n) - 1/\operatorname{poly}(n)]$.)

1. For $k = n/\varepsilon^2$, and all $i \in [k]$, get $\phi_i \leftarrow S_1$.
2. Using the Cook-Levin reduction get a Boolean formula $\phi = \phi_{\phi_1, \ldots, \phi_k}$ such $\phi \in \text{SAT}$, if there is a solution to satisfy at least $\tau = (\alpha + \beta(n))/2$ of the formulas $\phi_1, \ldots, \phi_k$. More formally, $\phi \in \text{SAT}$, if there is a vector $\bar{t} = (t_1, \ldots, t_k)$ such that $|\{i \colon \phi_i(t_i) = 1\}| \geq \tau$. This is possible since verifying $\bar{t}$ is efficiently possible.

By the Chernoff-Hoeffding bound,

$$\Pr_{\phi \leftarrow S_2}[\phi \in \text{SAT}] \geq 1 - e^{-(\varepsilon/2)^2 \cdot n/\varepsilon^2} \geq 1 - e^{-n/4}.$$

Proving the second property of the hard sampler S is less trivial, as it needs an efficient *reduction*. However, we can apply a weak bound here and then use Lemma 3.5.3. We first claim that for any poly-size adversary A,

$$\Pr_{\phi \leftarrow S_2, t \leftarrow \mathsf{A}(\phi)}[\phi(t) = 1] \leq 1 - \varepsilon/3. \tag{3.1}$$

To prove Equation 3.1, suppose for sake of contradiction that there is such adversary A. We can use A and solve $\phi' \leftarrow S_1$ with probability more than $\alpha + \Omega(\varepsilon)$ which is a contradiction. Given ϕ', The reduction is as follows.

1. Choose $i \leftarrow [k]$ at random.
2. Sample $k-1$ instances $\phi_1, \ldots, \phi_{i-1}, \phi_{i+1}, \ldots, \phi_k \leftarrow S_1$ independently at random.
3. Let $\phi_i = \phi'$.
4. Ask A to solve $\phi_{\phi_1,\ldots,\phi_k}$, and if A's answer gave a solution for $\phi_i = \phi'$, output this solution.

Since A cannot guess i, a simple argument shows that the above reduction succeeds with probability $\alpha + \varepsilon/2 - \varepsilon/3 = \alpha + \varepsilon/6$. Now that we have a puzzle generator S_2 that has satisfiable puzzles with probability $1 - \text{negl}(n)$ and efficient algorithms can solve its solutions by probability at most $\varepsilon/2$, using Lemma 3.5.3, we can use another direct product and design sampler S that samples $2n/\varepsilon$ independent instances from S_2 and asks for solutions to all of them. Because we already established that $\Pr_{\phi \leftarrow S_2}[\phi \in \text{SAT}] \geq 1 - \text{negl}(n)$, the puzzles sampled by S are also satisfiable by probability $1 - n \cdot \text{negl}(n) = 1 - \text{negl}(n)$, but efficient algorithms can still find the solution only with probability that is $\text{negl}(n)$. □

3.6 Conclusion

The assumption of computationally-bounded adversaries has been the key to modern cryptography. In fact, without this assumption modern cryptographic primitives would not be possible. this chapter investigates whether this assumption helps in the context of robust learning and demonstrates that is indeed the case (i.e., computational hardness can be leveraged in robust learning). We hope that this chapter is the first-step in leveraging computational hardness in the context of robust learning.

Several intriguing questions remain, such as:

- Our Construction 3.4.2 takes a natural learning problem, but then it modifies it. Can computational robustness be achieved for natural problems, such as image classification?
- Theorem 3.5.2 shows that computational hardness is necessary for nontrivial computational robustness. However, this does not still mean we can get cryptographic primitives back from such problems. Can we obtain cryptographically useful primitives, such as *one-way functions*, from such computational robustness?

Bibliography

Rudolf Ahlswede, Peter Gács, and János Körner. Bounds on conditional probabilities with applications in multi-user communication. *Probability Theory and Related Fields*, 34(2):157–177, 1976.

Noga Alon and Vitali D Milman. $\lambda 1$, isoperimetric inequalities for graphs, and superconcentrators. *Journal of Combinatorial Theory, Series B*, 38(1):73–88, 1985.

D Amir and VD Milman. Unconditional and symmetric sets inn-dimensional normed spaces. *Israel Journal of Mathematics*, 37(1-2):3–20, 1980.

Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*, pages 459–468. IEEE, 2006.

Alexandr Andoni and Ilya Razenshteyn. Optimal data-dependent hashing for approximate near neighbors. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 793–801. ACM, 2015.

Alexandr Andoni, Piotr Indyk, and Ilya Razenshteyn. Approximate nearest neighbor search in high dimensions. *arXiv preprint arXiv:1806.09823*, 2018.

Dana Angluin. Queries and Concept Learning. *Machine Learning*, 2(4):319–342, 1987.

Dana Angluin and Donna K. Slonim. Randomly Fallible Teachers: Learning Monotone DNF with an Incomplete Membership Oracle. *Machine Learning*, 14(1):7–26, 1994.

Dana Angluin, Martins Krikis, Robert H. Sloan, and György Turán. Malicious Omissions and Errors in Answers to Membership Queries. *Machine Learning*, 28(2-3):211–255, 1997a.

Dana Angluin, Mārtiņš Kriķis, Robert H. Sloan, and György Turán. Malicious Omissions and Errors in Answers to Membership Queries. *Machine Learning*, 28(2-3):211–255, 1997b.

Giuseppe Ateniese, Luigi V. Mancini, Angelo Spognardi, Antonio Villani, Domenico Vitali, and Giovanni Felici. Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers. *IJSN*, 10(3):137–150, 2015. doi: 10.1504/IJSN.2015.071829. URL https://doi.org/10.1504/IJSN.2015.071829.

Idan Attias, Aryeh Kontorovich, and Yishay Mansour. Improved generalization bounds for robust learning. *arXiv preprint arXiv:1810.02180*, 2018.

Yonatan Aumann and Yehuda Lindell. Security against covert adversaries: Efficient protocols for realistic adversaries. *Theory of cryptography*, pages 137–156, 2007.

Per Austrin, Kai-Min Chung, Mohammad Mahmoody, Rafael Pass, and Karn Seth. On the impossibility of cryptography with tamperable randomness. In *International Cryptology Conference*, pages 462–479. Springer, 2014a.

Per Austrin, Kai-Min Chung, Mohammad Mahmoody, Rafael Pass, and Karn Seth. On the impossibility of cryptography with tamperable randomness. In *International Cryptology Conference*, pages 462–479. Springer, 2014b.

Per Austrin, Kai-Min Chung, Mohammad Mahmoody, Rafael Pass, and Karn Seth. On the impossibility of cryptography with tamperable randomness. *Algorithmica*, 79(4):1052–1101, Dec 2017. ISSN 1432-0541. doi: 10.1007/s00453-016-0219-7. URL https://doi.org/10.1007/s00453-016-0219-7.

Pranjal Awasthi, Maria Florina Balcan, and Philip M. Long. The power of localization for efficiently learning linear separators with noise. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, pages 449–458. ACM, 2014.

Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. *arXiv preprint arXiv:1807.00459*, 2018.

Maria-Florina Balcan and Phil Long. Active and passive learning of linear separators under log-concave distributions. In *Conference on Learning Theory*, pages 288–316, 2013.

Marco Barreno, Blaine Nelson, Russell Sears, Anthony D Joseph, and J Doug Tygar. Can machine learning be secure? In *Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, pages 16–25. ACM, 2006.

Osbert Bastani, Yani Ioannou, Leonidas Lampropoulos, Dimitrios Vytiniotis, Aditya V. Nori, and Antonio Criminisi. Measuring Neural Net Robustness with Constraints. In *NIPS*, pages 2613–2621, 2016.

Salman Beigi, Omid Etesami, and Amin Gohari. Deterministic randomness extraction from generalized and distributed santha–vazirani sources. *SIAM Journal on Computing*, 46(1):1–36, 2017.

M. Ben-Or and N. Linial. Collective coin flipping. *Randomness and Computation*, 5:91–115, 1989.

Aharon Ben-Tal, Laurent El Ghaoui, and Arkadi S. Nemirovski. *Robust Optimization*. Princeton Series in Applied Mathematics. Princeton University Press, October 2009.

Gyora M. Benedek and Alon Itai. Learnability with Respect to Fixed Distributions. *Theoretical Computer Science*, 86(2):377–390, 1991.

Iddo Bentov, Ariel Gabizon, and David Zuckerman. Bitcoin beacon. *arXiv preprint arXiv:1605.04559*, 2016.

Itay Berman, Iftach Haitner, and Aris Tentes. Coin flipping of any constant bias implies one-way functions. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, pages 398–407. ACM, 2014.

Arjun Nitin Bhagoji, Supriyo Chakraborty, Prateek Mittal, and Seraphin Calo. Analyzing federated learning through an adversarial lens. *arXiv preprint arXiv:1811.12470*, 2018.

Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, pages 1467–1474. Omnipress, 2012.

Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Srndic, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion Attacks against Machine Learning at Test Time. In *ECML/PKDD*, pages 387–402, 2013.

Battista Biggio, Giorgio Fumera, and Fabio Roli. Security evaluation of pattern classifiers under attack. *IEEE transactions on knowledge and data engineering*, 26(4):984–996, 2014.

Peva Blanchard, Rachid Guerraoui, Julien Stainer, et al. Machine learning with adversaries: Byzantine tolerant gradient descent. In *Advances in Neural Information Processing Systems*, pages 119–129, 2017.

Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Occam's Razor. *Information Processing Letters*, 24(6):377–380, 1987.

Anselm Blumer, A. Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *Journal of the ACM*, 36(4):929–965, October 1989.

Sergey G Bobkov. An isoperimetric inequality on the discrete cube, and an elementary proof of the isoperimetric inequality in gauss space. *The Annals of Probability*, 25(1):206–214, 1997.

Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1175–1191. ACM, 2017.

Christer Borell. The brunn-minkowski inequality in gauss space. *Inventiones mathematicae*, 30(2):207–216, 1975.

Nader H. Bshouty, Nadav Eiron, and Eyal Kushilevitz. PAC learning with nasty noise. *Theoretical Computer Science*, 288(2):255–275, 2002.

Sébastien Bubeck, Yin Tat Lee, Eric Price, and Ilya Razenshteyn. Adversarial examples from cryptographic pseudo-random generators. *arXiv preprint arXiv:1811.06418*, 2018a.

Sébastien Bubeck, Eric Price, and Ilya Razenshteyn. Adversarial examples from computational constraints. *arXiv preprint arXiv:1805.10204*, 2018b.

Sébastien Bubeck, Eric Price, and Ilya Razenshteyn. Adversarial examples from computational constraints. *arXiv preprint arXiv:1805.10204*, 2018c.

Ran Canetti, Uri Feige, Oded Goldreich, and Moni Naor. Adaptively secure multi-party computation. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 639–648. ACM, 1996a.

Ran Canetti, Uriel Feige, Oded Goldreich, and Moni Naor. Adaptively secure multi-party computation. pages 639–648, 1996b.

Ran Canetti, Shai Halevi, and Michael Steiner. Hardness amplification of weakly verifiable puzzles. In *Theory of Cryptography Conference*, pages 17–33. Springer, 2005.

Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 3–14. ACM, 2017a.

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, 2017b.

Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Xiaodong Song. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *USENIX Security Symposium*, 2018.

Moses Charikar, Jacob Steinhardt, and Gregory Valiant. Learning from untrusted data. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 47–60. ACM, 2017.

Melissa Chase, Esha Ghosh, and Saeed Mahloujifar. Poisoning attacks against privacy of collaborative learning. *Under Submission*, 2020.

Lingjiao Chen, Hongyi Wang, Zachary Charles, and Dimitris Papailiopoulos. Draco: Byzantine-resilient distributed training via redundant gradients. In *International Conference on Machine Learning*, pages 902–911, 2018.

Yudong Chen, Lili Su, and Jiaming Xu. Distributed statistical machine learning in adversarial settings: Byzantine gradient descent. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 1(2):44, 2017.

Herman Chernoff. A Measure of Asymptotic Efficiency for Tests of a Hypothesis Based on the sum of Observations. *Annals of Mathematical Statistics*, 23(4):493–507, 1952.

Benny Chor and Oded Goldreich. Unbiased bits from sources of weak randomness and probabilistic communication complexity. pages 429–442.

Benny Chor and Oded Goldreich. Unbiased bits from sources of weak randomness and probabilistic communication complexity. *SIAM Journal on Computing*, 17(2):230–261, 1988.

Gregory Cirincione and Dinesh Verma. Federated machine learning for multi-domain operations at the tactical edge. In *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, volume 11006, page 1100606. International Society for Optics and Photonics, 2019.

Richard Cleve and Russell Impagliazzo. Martingales, collective coin flipping and discrete control processes. *In other words*, 1:5, 1993.

Daniel Cullina, Arjun Nitin Bhagoji, and Prateek Mittal. Pac-learning in the presence of evasion adversaries. *arXiv preprint arXiv:1806.01471*, 2018.

Akshay Degwekar and Vinod Vaikuntanathan. Computational limitations in robust classification and win-win results. *arXiv preprint arXiv:1902.01086*, 2019.

Luc Devroye, László Györfi, and Gábor Lugosi. *A Probabilistic Theory of Pattern Recognition*. Springer Science & Business Media, 2013.

Ilias Diakonikolas and Daniel M Kane. Recent advances in algorithmic high-dimensional robust statistics. *arXiv preprint arXiv:1911.05911*, 2019.

Ilias Diakonikolas, Gautam Kamath, Daniel M Kane, Jerry Li, Ankur Moitra, and Alistair Stewart. Robust estimators in high dimensions without the computational intractability. In *Foundations of Computer Science (FOCS), 2016 IEEE 57th Annual Symposium on*, pages 655–664. IEEE, 2016.

Ilias Diakonikolas, Daniel M Kane, and Alistair Stewart. Statistical query lower bounds for robust estimation of high-dimensional Gaussians and Gaussian mixtures. In *Foundations of Computer Science (FOCS), 2017 IEEE 58th Annual Symposium on*, pages 73–84. IEEE, 2017.

Ilias Diakonikolas, Gautam Kamath, Daniel M Kane, Jerry Li, Jacob Steinhardt, and Alistair Stewart. Sever: A robust meta-algorithm for stochastic optimization. *arXiv preprint arXiv:1803.02815*, 2018a.

Ilias Diakonikolas, Daniel M Kane, and Alistair Stewart. List-decodable robust mean estimation and learning mixtures of spherical Gaussians. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1047–1060. ACM, 2018b.

Ilias Diakonikolas, Weihao Kong, and Alistair Stewart. Efficient algorithms and lower bounds for robust linear regression. *arXiv preprint arXiv:1806.00040*, 2018c.

Dimitrios Diochnos, Saeed Mahloujifar, and Mohammad Mahmoody. Adversarial risk and robustness: General definitions and implications for the uniform distribution. In *Advances in Neural Information Processing Systems*, pages 10359–10368, 2018a.

Dimitrios Diochnos, Saeed Mahloujifar, and Mohammad Mahmoody. Adversarial risk and robustness: General definitions and implications for the uniform distribution. In *Advances in Neural Information Processing Systems*, pages 10359–10368, 2018b.

Dimitrios I. Diochnos. On the Evolution of Monotone Conjunctions: Drilling for Best Approximations. In *ALT*, pages 98–112, 2016.

Dimitrios I. Diochnos, Saeed Mahloujifar, and Mohammad Mahmoody. Adversarial Risk and Robustness: General Definitions and Implications for the Uniform Distribution. In *Conference on Neural Information Processing Systems (NIPS)*, 2018c.

Dimitrios I Diochnos, Saeed Mahloujifar, and Mohammad Mahmoody. Lower bounds for adversarially robust pac learning. *arXiv preprint arXiv:1906.05815*, 2019.

Yevgeniy Dodis and Yanqing Yao. Privacy with imperfect randomness. In *Annual Cryptology Conference*, pages 463–482. Springer, 2015.

Yevgeniy Dodis, Shien Jin Ong, Manoj Prabhakaran, and Amit Sahai. On the (Im)possibility of Cryptography with Imperfect Randomness. In *FOCS: IEEE Symposium on Foundations of Computer Science (FOCS)*, 2004.

Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam D. Smith. Calibrating noise to sensitivity in private data analysis. In Shai Halevi and Tal Rabin, editors, *Theory of Cryptography, Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006, Proceedings*, volume 3876 of *Lecture Notes in Computer Science*, pages 265–284. Springer, 2006. doi: 10.1007/11681878_14. URL https://doi.org/10.1007/11681878_14.

Cynthia Dwork, Guy N Rothblum, and Salil Vadhan. Boosting and differential privacy. In *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*, pages 51–60. IEEE, 2010.

Andrzej Ehrenfeucht, David Haussler, Michael J. Kearns, and Leslie G. Valiant. A General Lower Bound on the Number of Examples Needed for Learning. *Information and Computation*, 82(3):247–261, 1989.

David Eisenstat and Dana Angluin. The vc dimension of k-fold union. *Information Processing Letters*, 101 (5):181–184, 2007.

Gamaleldin F. Elsayed, Shreya Shankar, Brian Cheung, Nicolas Papernot, Alex Kurakin, Ian J. Goodfellow, and Jascha Sohl-Dickstein. Adversarial Examples that Fool both Computer Vision and Time-Limited Humans. *NIPS*, 2018.

Omid Etesami, Saeed Mahloujifar, and Mohammad Mahmoody. Computational concentration of measure: Optimal bounds, reductions, and more. *CoRR*, abs/1907.05401, 2019a. URL http://arxiv.org/abs/1907.05401.

Omid Etesami, Saeed Mahloujifar, and Mohammad Mahmoody. Computational concentration of measure: Optimal bounds, reductions, and more. *arXiv preprint arXiv:1907.05401*, 2019b. To appear in SODA 2020.

Omid Etesami, Saeed Mahloujifar, and Mohammad Mahmoody. Computational concentration of measure: Optimal bounds, reductions, and more. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 345–363. SIAM, 2020.

Alhussein Fawzi, Hamza Fawzi, and Omar Fawzi. Adversarial vulnerability for any classifier. *arXiv preprint arXiv:1802.08686*, 2018.

Uriel Feige, Yishay Mansour, and Robert Schapire. Learning and inference in the presence of corrupted inputs. In *Conference on Learning Theory*, pages 637–657, 2015.

Uriel Feige, Yishay Mansour, and Robert E Schapire. Robust inference for multiclass classification. In *Algorithmic Learning Theory*, pages 368–386, 2018.

Nic Ford, Justin Gilmer, Nicolas Carlini, and Dogus Cubuk. Adversarial examples are a natural consequence of test error in noise. *arXiv preprint arXiv:1901.10513*, 2019.

Andrew Frank, Arthur Asuncion, et al. Uci machine learning repository, 2010. *URL http://archive. ics. uci. edu/ml*, 15:22, 2011.

Michael Frazier, Sally A. Goldman, Nina Mishra, and Leonard Pitt. Learning from a Consistently Ignorant Teacher. *Journal of Computer and System Sciences*, 52(3):471–492, 1996.

Clement Fung, Chris JM Yoon, and Ivan Beschastnikh. Mitigating sybils in federated learning poisoning. *arXiv preprint arXiv:1808.04866*, 2018.

Karan Ganju, Qi Wang, Wei Yang, Carl A. Gunter, and Nikita Borisov. Property inference attacks on fully connected neural networks using permutation invariant representations. In *CCS '18*, 2018a.

Karan Ganju, Qi Wang, Wei Yang, Carl A. Gunter, and Nikita Borisov. Property inference attacks on fully connected neural networks using permutation invariant representations. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, page 619–633, New York, NY, USA, 2018b. Association for Computing Machinery. ISBN 9781450356930. doi: 10.1145/3243734.3243834. URL https://doi.org/10.1145/3243734.3243834.

Sanjam Garg, Somesh Jha, Saeed Mahloujifar, and Mohammad Mahmoody. Adversarially robust learning could leverage computational hardness. *arXiv preprint arXiv:1905.11564*, 2019.

Sanjam Garg, Somesh Jha, Saeed Mahloujifar, and Mahmoody Mohammad. Adversarially robust learning could leverage computational hardness. In *Algorithmic Learning Theory*, pages 364–385, 2020.

Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 169–178. ACM, 2009. doi: 10.1145/1536414.1536440. URL https://doi.org/10.1145/1536414.1536440.

Apostolos A Giannopoulos and Vitali D Milman. Euclidean structure in finite dimensional normed spaces. *Handbook of the geometry of Banach spaces*, 1:707–779, 2001.

Justin Gilmer, Ryan P Adams, Ian Goodfellow, David Andersen, and George E Dahl. Motivating the rules of the game for adversarial example research. *arXiv preprint arXiv:1807.06732*, 2018a.

Justin Gilmer, Luke Metz, Fartash Faghri, Samuel S Schoenholz, Maithra Raghu, Martin Wattenberg, and Ian Goodfellow. Adversarial Spheres. *arXiv preprint arXiv:1801.02774*, 2018b.

Oded Goldreich. *Foundations of cryptography: volume 1, basic tools*. Cambridge university press, 2007.

Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred V. Aho, editor, *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, pages 218–229. ACM, 1987. doi: 10.1145/28395.28420. URL https://doi.org/10.1145/28395.28420.

Shafi Goldwasser. From idea to impact, the crypto story: What's next? *IACR Distinguished Lecture at CRYPTO conference*, August 2018.

Shafi Goldwasser and Yael Tauman Kalai. Cryptographic assumptions: A position paper. In *Theory of Cryptography Conference*, pages 505–522. Springer, 2016.

Shafi Goldwasser, Yael Tauman Kalai, and Sunoo Park. Adaptively secure coin-flipping, revisited. In *International Colloquium on Automata, Languages, and Programming*, pages 663–674. Springer, 2015a.

Shafi Goldwasser, Yael Tauman Kalai, and Sunoo Park. Adaptively secure coin-flipping, revisited. In *International Colloquium on Automata, Languages, and Programming*, pages 663–674. Springer, 2015b.

Carlos R. González and Yaser S. Abu-Mostafa. Mismatched training and test distributions can outperform matched ones. *Neural Computation*, 27(2):365–387, 2015.

Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples. In *ICLR*, 2015. URL http://arxiv.org/abs/1412.6572.

Ian J. Goodfellow, Patrick D. McDaniel, and Nicolas Papernot. Making machine learning robust against adversarial inputs. *Communications of the ACM*, 61(7):56–66, 2018.

Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017.

Rachid Guerraoui, Sébastien Rouault, et al. The hidden vulnerability of distributed learning in byzantium. In *International Conference on Machine Learning*, pages 3518–3527, 2018.

Iftach Haitner and Eran Omri. Coin flipping with constant bias implies one-way functions. *SIAM Journal on Computing*, 43(2):389–409, 2014.

Iftach Haitner, Yuval Ishai, Eyal Kushilevitz, Yehuda Lindell, and Erez Petrank. Black-box constructions of protocols for secure computation. Cryptology ePrint Archive, Report 2010/164, 2010. http://eprint.iacr.org/2010/164.

Yufei Han and Xiangliang Zhang. Robust federated training via collaborative machine teaching using trusted instances. *arXiv preprint arXiv:1905.02941*, 2019.

Lawrence H Harper. Optimal numberings and isoperimetric problems on graphs. *Journal of Combinatorial Theory*, 1(3):385–393, 1966.

Jamie Hayes and Olga Ohrimenko. Contamination attacks and mitigation in multi-party machine learning. In *Advances in Neural Information Processing Systems*, pages 6604–6615, 2018.

Yingzhe He, Guozhu Meng, Kai Chen, Xingbo Hu, and Jinwen He. Towards privacy and security of deep learning systems: A survey, 2019.

Wassily Hoeffding. Probability Inequalities for Sums of Bounded Random Variables. *Journal of the American Statistical Association*, 58(301):13–30, March 1963.

Thomas Holenstein and Grant Schoenebeck. General hardness amplification of predicates and puzzles. In *Theory of Cryptography Conference*, pages 19–36. Springer, 2011.

Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. Black-box adversarial attacks with limited queries and information. *arXiv preprint arXiv:1804.08598*, 2018.

Russell Impagliazzo and Valentine Kabanets. Constructive proofs of concentration bounds. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 617–631. Springer, 2010.

Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613. ACM, 1998.

Matthew Jagielski. Subpopulation data poisoning attacks. 2019.

Matthew Jagielski, Alina Oprea, Battista Biggio, Chang Liu, Cristina Nita-Rotaru, and Bo Li. Manipulating machine learning: Poisoning attacks and countermeasures for regression learning. In *2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA*, pages 19–35. IEEE Computer Society, 2018. doi: 10.1109/SP.2018.00057. URL https://doi.org/10.1109/SP.2018.00057.

Yael Tauman Kalai, Ilan Komargodski, and Ran Raz. A lower bound for adaptively-secure collective coin-flipping protocols. In *32nd International Symposium on Distributed Computing*, 2018a.

Yael Tauman Kalai, Ilan Komargodski, and Ran Raz. A lower bound for adaptively-secure collective coin-flipping protocols. In *32nd International Symposium on Distributed Computing*, 2018b.

Jonathan Katz and Yehuda Lindell. *Introduction to modern cryptography*. Chapman and Hall/CRC, 2014.

Michael Kearns and Ming Li. Learning in the presence of malicious errors. *SIAM Journal on Computing*, 22 (4):807–837, 1993a.

Michael J. Kearns and Ming Li. Learning in the Presence of Malicious Errors. *SIAM J. on Computing*, 22 (4):807–837, 1993b.

Justin Khim and Po-Ling Loh. Adversarial risk bounds for binary classification via function transformation. *arXiv preprint arXiv:1810.09519*, 2018.

Bryan Klimt and Yiming Yang. The enron corpus: A new dataset for email classification research. *European Conference on Machine Learning*, pages 217–226, 2004.

Pang Wei Koh, Jacob Steinhardt, and Percy Liang. Stronger data poisoning attacks break data sanitization defenses. *arXiv preprint arXiv:1811.00741*, 2018.

Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

Kevin A Lai, Anup B Rao, and Santosh Vempala. Agnostic estimation of mean and covariance. In *Foundations of Computer Science (FOCS), 2016 IEEE 57th Annual Symposium on*, pages 665–674. IEEE, 2016.

Michel Ledoux. *The Concentration of Measure Phenomenon*. Number 89 in Mathematical Surveys and Monographs. American Mathematical Society, 2001.

Paul Lévy. *Problèmes concrets d'analyse fonctionnelle*, volume 6. Gauthier-Villars Paris, 1951.

JG Liao and Arthur Berg. Sharpening Jensen's inequality. *The American Statistician*, (accepted in 2017). URL https://arxiv.org/abs/1707.08644.

David Lichtenstein, Nathan Linial, and Michael Saks. Some extremal problems arising from discrete control processes. *Combinatorica*, 9(3):269–287, 1989.

Philip M Long. On the sample complexity of pac learning half-spaces against the uniform distribution. *IEEE Transactions on Neural Networks*, 6(6):1556–1559, 1995.

Daniel Lowd and Christopher Meek. Adversarial learning. In *KDD*, pages 641–647, 2005.

Aleksander Madry. Machine learning and security: The good, the bad, and the hopeful. *Talk given at workshop: Beyond Crypto, A TCS Perspective*, August 2018.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards Deep Learning Models Resistant to Adversarial Attacks. In *ICLR*, 2018.

Saeed Mahloujifar and Mohammad Mahmoody. Blockwise p-tampering attacks on cryptographic primitives, extractors, and learners. Cryptology ePrint Archive, Report 2017/950, 2017a. https://eprint.iacr.org/2017/950.

Saeed Mahloujifar and Mohammad Mahmoody. Blockwise p-Tampering Attacks on Cryptographic Primitives, Extractors, and Learners. In *Theory of Cryptography Conference*, pages 245–279. Springer, 2017b.

Saeed Mahloujifar and Mohammad Mahmoody. Can adversarially robust learning leverage computational hardness? *arXiv preprint arXiv:1810.01407*, 2018a.

Saeed Mahloujifar and Mohammad Mahmoody. Can adversarially robust learning leverage computational hardness? *arXiv preprint arXiv:1810.01407*, 2018b.

Saeed Mahloujifar and Mohammad Mahmoody. Can adversarially robust learning leverage computational hardness? In Aurélien Garivier and Satyen Kale, editors, *Proceedings of the 30th International Conference on Algorithmic Learning Theory*, volume 98 of *Proceedings of Machine Learning Research*, pages 581–609, Chicago, Illinois, 22–24 Mar 2019a. PMLR. URL http://proceedings.mlr.press/v98/mahloujifar19a.html.

Saeed Mahloujifar and Mohammad Mahmoody. Can adversarially robust learning leveragecomputational hardness? In *Algorithmic Learning Theory*, pages 581–609, 2019b.

Saeed Mahloujifar and Mohammad Mahmoody. Can adversarially robust learning leverage computational hardness? *Algorithmic Learning Theory (ALT)*, 2019c.

Saeed Mahloujifar, Dimitrios I Diochnos, and Mohammad Mahmoody. Learning under p-Tampering Attacks. In *ALT*, pages 572–596, 2018a.

Saeed Mahloujifar, Dimitrios I Diochnos, and Mohammad Mahmoody. The curse of concentration in robust learning: Evasion and poisoning attacks from concentration of measure. *arXiv preprint arXiv:1809.03063*, 2018b.

Saeed Mahloujifar, Dimitrios I Diochnos, and Mohammad Mahmoody. Learning under p-tampering attacks. In *Algorithmic Learning Theory*, pages 572–596, 2018c.

Saeed Mahloujifar, Dimitrios I Diochnos, and Mohammad Mahmoody. Adversarial risk and robustness: General definitions and implications for the uniform distribution. *In Conference on Neural Information Processing Systems (NIPS)*, 2018d.

Saeed Mahloujifar, Mohammad Mahmoody, and Ameer Mohammed. Multi-party poisoning through generalized p-tampering. *arXiv preprint arXiv:1809.03474*, 2018e.

Saeed Mahloujifar, Dimitrios I Diochnos, and Mohammad Mahmoody. The curse of concentration in robust learning: Evasion and poisoning attacks from concentration of measure. *Conference on Artificial Intelligence (AAAI)*, 2019a.

Saeed Mahloujifar, Dimitrios I Diochnos, and Mohammad Mahmoody. Learning under p-tampering poisoning attacks. *Annals of Mathematics and Artificial Intelligence*, pages 1–34, 2019b.

Saeed Mahloujifar, Mohammad Mahmoody, and Ameer Mohammed. Universal multi-party poisoning attacks. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 4274–4283. PMLR, 2019c. URL http://proceedings.mlr.press/v97/mahloujifar19a.html.

Saeed Mahloujifar, Xiao Zhang, Mohammad Mahmoody, and David Evans. Empirically measuring concentration: Fundamental limits on intrinsic robustness. *Advances in Neural Information Processing Systems*, 2019d.

Saeed Mahloujifar, Xiao Zhang, Mohammad Mahmoody, and David Evans. Empirically measuring concentration: Fundamental limits on intrinsic robustness. *Safe Machine Learning workshop at ICLR*, 2019e.

Hemanta K Maji, Manoj Prabhakaran, and Amit Sahai. On the computational complexity of coin flipping. In *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*, pages 613–622. IEEE, 2010.

Yishay Mansour, Aviad Rubinstein, and Moshe Tennenholtz. Robust probabilistic inference. In *Proceedings of the twenty-sixth annual ACM-SIAM symposium on Discrete algorithms*, pages 449–460. Society for Industrial and Applied Mathematics, 2015.

Grigorii Aleksandrovich Margulis. Probabilistic characteristics of graphs with large connectivity. *Problemy peredachi informatsii*, 10(2):101–108, 1974.

Katalin Marton. A simple proof of the blowing-up lemma (corresp.). *IEEE Transactions on Information Theory*, 32(3):445–446, 1986.

Colin McDiarmid. On the method of bounded differences. *Surveys in combinatorics*, 141(1):148–188, 1989.

Colin Mcdiarmid, M Habib, J Ramirez-Alfonsin, and B Reed. Probabilistic methods for algorithmic discrete mathematics. *Algorithms and Combinatorics Series*, 16:1–46, 1998.

Brendan McMahan and Daniel Ramage. Federated learning: Collaborative machine learning without centralized training data. *Google Research Blog*, 2017.

H Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, et al. Communication-efficient learning of deep networks from decentralized data. *arXiv preprint arXiv:1602.05629*, 2016.

Shike Mei and Xiaojin Zhu. Using machine teaching to identify optimal training-set attacks on machine learners. In Blai Bonet and Sven Koenig, editors, *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*, pages 2871–2877. AAAI Press, 2015. URL http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9472.

Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Exploiting unintended feature leakage in collaborative learning. *2019 IEEE Symposium on Security and Privacy (SP)*, May 2019. doi: 10.1109/sp.2019.00029. URL http://dx.doi.org/10.1109/SP.2019.00029.

Vitali D Milman and Gideon Schechtman. *Asymptotic theory of finite dimensional normed spaces*, volume 1200. Springer, 1986.

Omar Montasser, Steve Hanneke, and Nathan Srebro. Vc classes are adversarially robustly learnable, but only improperly. *arXiv preprint arXiv:1902.04217*, 2019.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks. In *CVPR*, pages 2574–2582, 2016.

Robin A Moser. A constructive proof of the lovász local lemma. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 343–350. ACM, 2009.

Robin A Moser and Gábor Tardos. A constructive proof of the general lovász local lemma. *Journal of the ACM (JACM)*, 57(2):11, 2010.

Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008.

Moni Naor. On cryptographic assumptions and challenges. In *Annual International Cryptology Conference*, pages 96–109. Springer, 2003.

Blaine Nelson, Marco Barreno, Fuching Jack Chi, Anthony D. Joseph, Benjamin I. P. Rubinstein, Udam Saini, Charles A. Sutton, J. Doug Tygar, and Kai Xia. Exploiting machine learning to subvert your spam filter. In Fabian Monrose, editor, *First USENIX Workshop on Large-Scale Exploits and Emergent Threats, LEET '08, San Francisco, CA, USA, April 15, 2008, Proceedings*. USENIX Association, 2008. URL http://www.usenix.org/events/leet08/tech/full_papers/nelson/nelson.pdf.

Blaine Nelson, Benjamin I. P. Rubinstein, Ling Huang, Anthony D. Joseph, and J. D. Tygar. Classifier Evasion: Models and Open Problems. In *PSDM*, pages 92–98, 2010a.

Blaine Nelson, Benjamin I. P. Rubinstein, Ling Huang, Anthony D. Joseph, and J. D. Tygar. Classifier Evasion: Models and Open Problems. In *PSDM*, pages 92–98, 2010b.

Blaine Nelson, Benjamin IP Rubinstein, Ling Huang, Anthony D Joseph, Steven J Lee, Satish Rao, and JD Tygar. Query strategies for evading convex-inducing classifiers. *Journal of Machine Learning Research*, 13(May):1293–1332, 2012.

Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nati Srebro. Exploring generalization in deep learning. In *Advances in Neural Information Processing Systems*, pages 5947–5956, 2017.

Omkant Pandey, Rafael Pass, and Vinod Vaikuntanathan. Adaptive one-way functions and applications. In *Annual International Cryptology Conference*, pages 57–74. Springer, 2008.

Nicolas Papernot, Patrick McDaniel, Arunesh Sinha, and Michael Wellman. Towards the science of security and privacy in machine learning. *arXiv preprint arXiv:1611.03814*, 2016a.

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, 2016b.

Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pages 506–519. ACM, 2017.

Leonard Pitt and Leslie G. Valiant. Computational limitations on learning from examples. *Journal of the ACM*, 35(4):965–984, 1988.

Adarsh Prasad, Arun Sai Suggala, Sivaraman Balakrishnan, and Pradeep Ravikumar. Robust estimation via robust gradient estimation. *arXiv preprint arXiv:1802.06485*, 2018.

Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Certified defenses against adversarial examples. *arXiv preprint arXiv:1801.09344*, 2018.

Aditi Raghunathan, Sang Michael Xie, Fanny Yang, John C Duchi, and Percy Liang. Adversarial training can hurt generalization. *arXiv preprint arXiv:1906.06032*, 2019.

C Radhakrishna Rao. Information and the accuracy attainable in the estimation of statistical parameters. In *Breakthroughs in statistics*, pages 235–247. Springer, 1992.

Omer Reingold, Salil Vadhan, and Avi Wigderson. A note on extracting randomness from santha-vazirani sources. *Unpublished manuscript*, 2004.

Phillip Rogaway and Yusi Zhang. Simplifying game-based definitions. In *Annual International Cryptology Conference*, pages 3–32. Springer, 2018.

John Rompel. One-way functions are necessary and sufficient for secure signatures. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 387–394. ACM, 1990.

Benjamin I.P. Rubinstein, Blaine Nelson, Ling Huang, Anthony D. Joseph, Shing-hon Lau, Satish Rao, Nina Taft, and J.D. Tygar. Antidote: understanding and defending against poisoning of anomaly detectors. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, pages 1–14. ACM, 2009a.

Benjamin I.P. Rubinstein, Blaine Nelson, Ling Huang, Anthony D. Joseph, Shing-hon Lau, Satish Rao, Nina Taft, and J.D. Tygar. Stealthy poisoning attacks on pca-based anomaly detectors. *ACM SIGMETRICS Performance Evaluation Review*, 37(2):73–74, 2009b.

Sivan Sabato, Nathan Srebro, and Naftali Tishby. Distribution-dependent sample complexity of large margin learning. *The Journal of Machine Learning Research*, 14(1):2119–2149, 2013.

Alexandre Sablayrolles, Matthijs Douze, Cordelia Schmid, Yann Ollivier, and Hervé Jégou. White-box vs black-box: Bayes optimal strategies for membership inference. *ArXiv*, abs/1908.11229, 2019.

Miklos Santha and Umesh V. Vazirani. Generating quasi-random sequences from semi-random sources. *J. Comput. Syst. Sci.*, 33(1):75–87, 1986.

Ludwig Schmidt, Shibani Santurkar, Dimitris Tsipras, Kunal Talwar, and Aleksander Madry. Adversarially Robust Generalization Requires More Data. *arXiv preprint arXiv:1804.11285*, 2018.

Clayton D Scott and Robert D Nowak. Learning minimum volume sets. *Journal of Machine Learning Research*, 7(Apr):665–704, 2006.

Ali Shafahi, W Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein. Poison frogs! targeted clean-label poisoning attacks on neural networks. In *Advances in Neural Information Processing Systems*, pages 6103–6113, 2018a.

Ali Shafahi, W Ronny Huang, Christoph Studer, Soheil Feizi, and Tom Goldstein. Are adversarial examples inevitable? *arXiv preprint arXiv:1809.02104*, 2018b.

Ali Shafahi, W Ronny Huang, Christoph Studer, Soheil Feizi, and Tom Goldstein. Are adversarial examples inevitable? *arXiv preprint arXiv:1809.02104*, 2018c.

Shiqi Shen, Shruti Tople, and Prateek Saxena. A uror: defending against poisoning attacks in collaborative deep learning systems. In *Proceedings of the 32nd Annual Conference on Computer Security Applications*, pages 508–519. ACM, 2016.

Victor Shoup. Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332, 2004. https://eprint.iacr.org/2004/332.

Aman Sinha, Hongseok Namkoong, and John Duchi. Certifiable distributional robustness with principled adversarial training. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=Hk6kPgZA-.

Robert H. Sloan. Four Types of Noise in Data for PAC Learning. *Information Processing Letters*, 54(3): 157–162, 1995.

Robert H. Sloan and György Turán. Learning with Queries but Incomplete Information (Extended Abstract). In *COLT*, pages 237–245, 1994.

Robert H. Sloan, Balázs Szörényi, and György Turán. *Learning Boolean Functions with Queries*, pages 221–256. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 2010. doi: 10.1017/CBO9780511780448.010.

Vladimir N Sudakov and Boris S Tsirel'son. Extremal properties of half-spaces for spherically invariant measures. *Journal of Soviet Mathematics*, 9(1):9–18, 1978.

Arun Sai Suggala, Adarsh Prasad, Vaishnavh Nagarajan, and Pradeep Ravikumar. On Adversarial Risk and Training. *arXiv preprint arXiv:1806.02924*, 2018a.

Arun Sai Suggala, Adarsh Prasad, Vaishnavh Nagarajan, and Pradeep Ravikumar. On adversarial risk and training. *arXiv preprint arXiv:1806.02924*, 2018b.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *ICLR*, 2014. URL http://arxiv.org/abs/1312.6199.

Michel Talagrand. Concentration of measure and isoperimetric inequalities in product spaces. *Publications Mathématiques de l'Institut des Hautes Etudes Scientifiques*, 81(1):73–205, 1995.

Richard Tomsett, Kevin Chan, and Supriyo Chakraborty. Model poisoning attacks against distributed machine learning systems. In *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, volume 11006, page 110061D. International Society for Optics and Photonics, 2019.

Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness May Be at Odds with Accuracy. *arXiv preprint arXiv:1805.12152*, 2018a.

Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. *stat*, 1050:11, 2018b.

Leslie Valiant. *Probably Approximately Correct: Nature's Algorithms for Learning and Prospering in a Complex World*. Basic Books, Inc., New York, NY, USA, 2013.

Leslie G. Valiant. A Theory of the Learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

Leslie G. Valiant. Learning disjunctions of conjunctions. In *IJCAI*, pages 560–566, 1985.

John Von Neumann. 13. various techniques used in connection with random digits. *Appl. Math Ser*, 12: 36–38, 1951.

Yizhen Wang and Kamalika Chaudhuri. Data poisoning attacks against online learning. *arXiv preprint arXiv:1808.08994*, 2018.

David H Wolpert. The lack of a priori distinctions between learning algorithms. *Neural computation*, 8(7): 1341–1390, 1996.

Eric Wong and Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference on Machine Learning*, pages 5283–5292, 2018.

Eric Wong, Frank Schmidt, Jan Hendrik Metzen, and J Zico Kolter. Scaling provable adversarial defenses. *arXiv preprint arXiv:1805.12514*, 2018.

Huang Xiao, Battista Biggio, Gavin Brown, Giorgio Fumera, Claudia Eckert, and Fabio Roli. Is feature selection secure against training data poisoning? In *ICML*, pages 1689–1698, 2015.

Huan Xu and Shie Mannor. Robustness and generalization. *Machine Learning*, 86(3):391–423, 2012.

Weilin Xu, David Evans, and Yanjun Qi. Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks. In *NDSS*, 2018.

Keisuke Yamazaki, Motoaki Kawanabe, Sumio Watanabe, Masashi Sugiyama, and Klaus-Robert Müller. Asymptotic bayesian generalization error when training and test distributions are different. In *ICML*, pages 1079–1086, 2007.

Andrew C Yao. Theory and application of trapdoor functions. In *23rd Annual Symposium on Foundations of Computer Science (SFCS 1982)*, pages 80–91. IEEE, 1982.

Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27-29 October 1986*, pages 162–167. IEEE Computer Society, 1986. doi: 10.1109/SFCS.1986.25. URL https://doi.org/10.1109/SFCS.1986.25.

Dong Yin, Yudong Chen, Kannan Ramchandran, and Peter Bartlett. Byzantine-robust distributed learning: Towards optimal statistical rates. *arXiv preprint arXiv:1803.01498*, 2018a.

Dong Yin, Kannan Ramchandran, and Peter Bartlett. Rademacher complexity for adversarially robust generalization. *arXiv preprint arXiv:1810.11914*, 2018b.

Xiaoyong Yuan, Pan He, Qile Zhu, and Xiaolin Li. Adversarial examples: Attacks and defenses for deep learning. *IEEE transactions on neural networks and learning systems*, 2019.