On Security and Privacy Implications of Modifying Machine Learning Datasets

---

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

Ji Gao

May 2022

APPROVAL SHEET

This

Dissertation

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

Doctor of Philosophy

Author: Ji Gao

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

Advisor: Mohammad Mahmoody

Advisor:

Committee Member: David Evans

Committee Member: Amin Karbasi

Committee Member: Jundong Li

Committee Member: Michael D. Porter

Committee Member: Haifeng Xu

Committee Member:

Accepted for the School of Engineering and Applied Science:

Jennifer L. West, School of Engineering and Applied Science

May 2022

Abstract

Machine learning is being increasingly applied to many security and privacy sensitive tasks. Datasets, as the raw material of the machine learning working flow, determine the properties of learned models. In this dissertation, we aim to understand security and privacy implications of different types of modifications made to training datasets. Specifically, we focus on the "small change" regime, where the modification to the training dataset is minimal. We study the following points.

1. *Security implications* of malicious modifications to the dataset (also called poisoning attacks). We study this scenario from both the view of the learner and the view of the adversary.

   - From the view of the learner, we formally define the learnability under instance-targeted poisoning attacks. Our main result shows that PAC learning and certification are achievable if adversary's budget scales sub-linearly with the sample complexity.
   - From the view of the adversary, we prove the existence of polynomial-time adversaries that can amplify any vulnerability with non-negligible chance of happening already.

2. *Privacy implications* of benign modifications to the dataset. Motivated to meet new legal requirements, many machine learning methods are recently extended to support machine *unlearning*, i.e., updating models as if certain examples are removed from their training sets, and meet new legal requirements. We formalize (various forms of) *deletion inference* and *deletion reconstruction* attacks, in which the adversary aims to either identify which record is deleted or to reconstruct the deleted records. We then present successful deletion inference and reconstruction attacks for a variety of machine learning models and tasks.

   The results presented in the dissertation are primarily from the following three papers [1, 2, 3].

To my family.

Acknowledgments

First of all, I would like to thank my advisor, Dr. Mohammad Mahmoody, for his constant guidance and support. Mohammad teaches me a lot of things. He is meticulous in doing research, always striving for excellence, which encourages me to pursue better research work.

I am grateful for the great support I received from my Ph.D. committee members and mentors in the research community. I would like to thank my committee members, Dr. David Evans, Dr. Amin Karbasi, Dr. Jundong Li, Dr. Michael D. Porter, and Dr. Haifeng Xu, for their support of my dissertation and Ph.D. study. I sincerely appreciate their insightful questions, valuable suggestions, and precious time. I also would like to thank Dr. Yanjun Qi, Dr. Mary Lou Soffa, and Dr. Hongning Wang for their guidance and advice during my Ph.D. journey.

I would also like to thank my collaborators. I could never have finished my research alone. I especially would like to thank my wife, Meiyi Ma, who is also one of my collaborators. During my Ph.D. journey, we spent a lot of time together working and discussing research ideas. She always supports me, even during the hardest days. I would like to express my deepest thank to her here. This dissertation also features collaborative works with Dr. Omid Etesami, Dr. Sanjam Garg, Dr. Saeed Mahloujifar, and Dr. Prashant Nalini Vasudevan.

Last but not least, I would like to thank my parents for their constant support and encouragement. Their constant belief in me always lights my path.

Contents

List of tables

List of figures

Notation

Throughout the dissertation we use the following notation. We use calligraphic letters (e.g., $\mathcal{X}$) for sets. Let $\mathbb{N} = \{0, 1, \ldots\}$ denote the set of integers, $\mathcal{X}$ the input/instance space, and $\mathcal{Y}$ the space of labels. By $\mathcal{Y}^{\mathcal{X}}$ we denote a set of all functions from $\mathcal{X}$ to $\mathcal{Y}$. By $\mathcal{H} \subset \mathcal{Y}^{\mathcal{X}}$ we denote the set of hypothesis functions from $\mathcal{X}$ to $\mathcal{Y}$. We use D to denote a distribution over $\mathcal{X} \times \mathcal{Y}$. By $e \sim D$ we state that e is distributed/sampled according to distribution D. For a set $\mathcal{S}$, the notation $e \sim \mathcal{S}$ means that e is uniformly sampled from $\mathcal{S}$. By D^m we denote a product distribution over m i.i.d. samples from D. By $D_{\mathcal{X}}$ we denote the projection of D over its first coordinate (i.e., the marginal distribution over $\mathcal{X}$).

Regarding machine learning, for a function $h \in \mathcal{Y}^{\mathcal{X}}$ and an example $e = (x, y) \in \mathcal{X} \times \mathcal{Y}$, we use $\ell(h, e)$ to denote the loss of predicting $h(x) \in \mathcal{Y}$ while the correct label for x is y. Loss will always be non-negative, and when it is in $[0, 1]$, we call it bounded. For classification problems, unless stated differently, we use the 0-1 loss, i.e., $\ell(h, e) = \mathbb{1}[h(x) \neq y]$. We use $\mathcal{S} \in (\mathcal{X} \times \mathcal{Y})^*$ to denote a training "set", even though more formally it is in fact a sequence. We use Lrn to denote a learning algorithm that (perhaps randomly) maps a training set $\mathcal{S} \sim D^m$ of any size m to some $h \in \mathcal{Y}^{\mathcal{X}}$. We call a learner Lrn *proper* (with respect to hypothesis class $\mathcal{H}$) if it always outputs some $h \in \mathcal{H}$. $\mathsf{Lrn}(\mathcal{S})(x)$ denotes the prediction on x by the hypothesis returned by $\mathsf{Lrn}(\mathcal{S})$. When Lrn is randomized, by $y \sim \mathsf{Lrn}(\mathcal{S})(x)$ we state that y is the prediction when the randomness of Lrn is chosen uniformly. For a randomized Lrn and the random seed r (of the appropriate length), Lrn_r denotes the deterministic learner with the hardwired randomness r. For a hypothesis $h \in \mathcal{H}$, a loss function ℓ, and a distribution D over $\mathcal{X} \times \mathcal{Y}$, the population (a.k.a. true) risk of h over D (with respect to the loss ℓ) is defined as $\mathrm{Risk}(h, D) = \mathbb{E}_{e \sim D}[\ell(h, e)]$, and the empirical risk of h over $\mathcal{S}$ is defined as $\mathrm{Risk}(h, \mathcal{S}) = \mathbb{E}_{e \sim \mathcal{S}}[\ell(h, e)]$. For a hypothesis class $\mathcal{H}$, we say that the realizability assumption holds for a distribution D if there exists an $h \in \mathcal{H}$ such that $\mathrm{Risk}(h, D) = 0$.

To add clarity to the text, we use a diamond "$\diamond$" to denote the end of a technical definition.

Chapter 1

Introduction

1.1 Machine learning and datasets

Machine learning applications are everywhere these days, from self-driving cars [5] to image/text classification [6, 7], even helping judges in making legal decisions [8]. The role of machine learning in our daily life seems to grow everyday. Consequently, malicious parties now have a lot more motivation to heavily invest in making learning systems deviate from their prescribed schemes into algorithms and final decisions that serve the interest of those entities. This has shifted the landscape towards having much more focus on *security* and *privacy* aspects of machine learning, compared to traditional machine learning that only deals with benign settings.

The fundamental task of machine learning is learning predictive models by generalizing from empirical data. In its classic form, a learning algorithm or *learner*, denoted by Lrn, first takes a collection of examples $\mathcal{S} = \{e_1, \ldots, e_m\}$ as the input to the algorithm, where $\mathcal{S}$ is known as the *training dataset* and each individual e_i is known as the *training example*. (Supervised) learning algorithms then generalize the knowledge from the training dataset, and try to minimize the error of predicting the correct label with a machine learning *model/predictor/hypothesis* h on test examples.

More specifically, the machine learning problem is defined with an *input/domain set* $\mathcal{X}$ and a *output/label* set $\mathcal{Y}$. A training example $e_i = (x_i, y_i)$ is an instance-label pair, where instance $x_i \in \mathcal{X}$ is the input to the model h, and label $y_i \in \mathcal{Y}$ is the expected output. We assume the examples are generated by a distribution D, and the training dataset is an i.i.d. sample from D, i.e., $\mathcal{S} = \{e_1, \ldots, e_m\} \sim D^m$. Furthermore, we assume a ground-truth labeling function f exists, that for each $(x_i, y_i) \sim D$, we have $y_i = f(x_i)$. Then the learner Lrn is a (possibly randomized) function that

takes a training dataset $\mathcal{S}$ as input, and returns a model $h \sim \mathsf{Lrn}(\mathcal{S})$ as output, where $h : \mathcal{X} \to \mathcal{Y}$.

The goal of machine learning is to make the predicted model h close to the ground-truth labeling function f. More specifically, suppose a *loss* function $\ell(h, e)$ is defined to measure the error of model h on example e for the learning task. We then define the *risk* function $\mathrm{Risk}(h, D)$ as the expected loss of model h, that is, $\mathrm{Risk}(h, D) = \mathbb{E}_{e \sim D}[\ell(h, e)]$. This (true/population) risk function measures the quality of the predictions of the model h over the data distribution D. Therefore, the goal of machine learning is to minimize this risk.

In machine learning, the step of producing the model h from training data is called the *training stage*, and the step of predicting test data is called the *testing stage*. Note that from the process of machine learning we just introduced, the learning algorithm Lrn has no direct knowledge about data distribution D. The only input to the algorithm is the training data set $\mathcal{S}$. As the raw material of this working flow, training dataset *determines* the properties of the learned model. Even a single example in the training dataset could be very important for the training of machine learning model and could lead to large deviation in the predictions.

In this thesis, we focus on the security and privacy impact of modifications on the training dataset, that is, how the action of adding, removing, or replacing examples in the training dataset influence the security and privacy of the learned model. Modifications on the training dataset is (not surprisingly) very common in the real practice. As it turn out that we will see even small modifications could have strong impacts on the security and privacy of the learned models.

We start by describing different types of modifications and their potential implications.

1.2 Modifications of datasets

Modifications (or updates) on datasets are common. There are many possible causes of such modifications, including the following cases.

- **Data is constantly updated.** For some learning tasks, data may be time-sensitive. As an example, in active learning [9], the learning algorithm is constantly asking human labels for the purpose of better learning. When a new label is returned, the dataset is updated to include this new example. In this case, the training dataset is constantly modified (and so is the learned model).

- **Data owner requests modification on the dataset.** The owner of the data (e.g., user of online services) and the trainer of the machine learning model (e.g., service providers such

as Google or Facebook) are often different parties. Data owners may request to remove their data from the dataset. In fact, there are recent legal requirements (e.g., the European Union's GDPR [10] or California's CCPA [11]) that aim to protect such right of the data owner to erase their data.

- **Malicious attack on the training dataset.** Adversarial parties may want to modify the training dataset to control the induced machine learning model. For example, in federated learning [12, 13], multiple parties contribute to the training dataset. Each party sets up a local model for their own data collection, and updates a centralized model with gradients from their local model. As a result, the centralized model is trained with multiple datasets without direct access to the datasets. A malicious party, in this scenario, can craft malicious inputs and add them into the dataset in order to attack the centralized model.

In a summary, modifications on training datasets can be separated into two main categories by the intention of the modification, namely,

- **Malicious modifications.** A malicious modification, named as a poisoning attack in the literature, is made by an adversary that aims to induce erroneous behaviors from model.
- **Benign modifications.** Benign modifications, which is typically made by the learner, aims to update the model into a newer version and to incorporate or remove information from model.

These different scenarios have different security and privacy implications . Therefore, we discuss these two scenarios separately in the following sections.

1.3 Security implications of malicious modifications

Recently, the classic machine learning setting has been revisited by allowing adversarial manipulations that tamper with the process, while the learner still aims to make correct predictions. In general, adversarial tamperings can take place in both the training stage or the testing stage of models. In the testing stage, adversaries can identify examples that is incorrectly predicted by the machine learning models. Such attack is passive: Whether the attack is successful or not depends on the machine learning algorithm and the predicted model.

Our interest in this thesis is on a form of training-stage attacks, also known as poisoning or causative attacks [14, 15, 16, 17]. In particular, poisoning adversaries may partially change the

training set $\mathcal{S}$ into another training set $\mathcal{S}'$ in such a way that the "quality" of the returned hypothesis h' by the learning algorithm Lrn, that is trained on $\mathcal{S}'$ instead of $\mathcal{S}$, degrades significantly.

Depending on the context, the way we measure the quality of the poisoning attack may differ. For instance, the quality of h' may refer to the expected error of h' when test data points are sampled from the distribution D. In the case, the adversary targets to maximize the population risk $\text{Risk}(h, D)$ of the predicted model. It could also refer to the error on a particular test point x, known to the adversary but unknown to the learning algorithm Lrn. In this setting, an adversary could craft its strategy based on the knowledge of the target instance x, making the adversary more powerful.

For the first scenario, a classic result from statistical learning [18] shows that applying the Empirical Risk Minimization (ERM) rule will lead to a Probably Approximately Correct (PAC) algorithm that is robust to poisoning adversaries who could change $b(m) = o(m)$ number of examples.

The latter scenario, which is the main focus of this thesis, is known as (instance) *targeted poisoning* [14]. We then study this more challenging scenario from both the view of the learner Lrn and the view of the adversary A, in Chapter 3 and Chapter 4 respectively.

1.4 Privacy implications of benign modifications

As described in Section 1.2, benign modifications are common in various application scenarios. We want to study the security and privacy implications of these benign modifications. We will show that in one specific type of modification - *machine unlearning*, the privacy implication of such modification can be substantial.

There are a lot of reasons for users to keep their data private. Since many machine (especially deep) learning applications use a huge amount of data, privacy concerns have long been raised about their data usage. The concerns aggravate when machine learning techniques are applied to sensitive fields, for example, medical profiles, financial data, and legal records. Leakage of user data in those fields will have gruesome impact. Not surprisingly, many techniques have been proposed to enhance the privacy of machine learning algorithms.

In the scope of our thesis we focus on machine unlearning. Machine unlearning refers to the operation that forgets specific example in the model, as if it was never inside the training dataset. When the owner of an example wants to withdraw the example from the model, the model shall be updated to not include any information about that example, as if the data was never in the training dataset. Recent legal requirements (e.g., the European Union's GDPR [10] or California's CCPA [11]) aim to make such privacy considerations mandatory. The question of *how* such privacy concerns

can be formally modeled and enforced is the subject of ongoing study [19, 20, 21, 22]. In particular, upon a deletion request for an example $e \in \mathcal{S}$, according to the machine unlearning framework the learner will need to update the model $h_{\mathcal{S}} = \mathsf{Lrn}(\mathcal{S})$ to h_{-e} such that $h_{-e} = \mathsf{Lrn}(\mathcal{S} \setminus \{e\})$ (ideally) has the same distribution as training a model from scratch using $\mathcal{S} \setminus \{e\}$.

Initially, it might seem like a perfect deletion of an example e from a model $h_{\mathcal{S}}$ and releasing h_{-e} instead should *help* with preventing all the leakage about the particular deleted example e. After all, we are *removing* e from the learning process of the model accessible to the adversary. However, the adversary now could potentially access *both* models $h_{\mathcal{S}}$ and h_{-e}, and so it might be able to extract even *more* information about the deleted example e compared to the setting in which the adversary could only access $h_{\mathcal{S}}$ or h_{-e} alone. As a simplified contrived example, suppose the examples $\mathcal{S} = \{e_1, \ldots, e_n\}$ are real-valued vectors, and suppose the ML model $h_{\mathcal{S}}$ (perhaps upon many queries) somehow reveals the summation $\sum_{i \in [n]} e_i$. In this case, if the set $\mathcal{S}$ is sampled from a distribution with sufficient entropy, the trained model $h_{\mathcal{S}}$ might potentially provide a certain degree of privacy for examples in $\mathcal{S}$. However, if one of the examples e_i is deleted from $h_{\mathcal{S}}$, then because the updated model h_{-e_i} also returns the updated summation $\sum_{j \neq i} e_j$, then an adversary who extracts both of these summations can reconstruct the deleted record e_i completely. In other words, the very task of deletion might in fact *harm* the privacy of the very deleted example e_i.

In this thesis we ask: How vulnerable are ML algorithms to leak information about the deleted examples, if an adversary gets to interact with the models both before and after the deletion updates? We will study the problem in Chapter 5.

1.5 Dissertation structure

In Chapter 2 we introduce related works, and how this dissertation advances the state-of-the-art research. After that, in Chapter 3 we study the learnability of machine learning algorithms under instance-targeted poisoning. Next, in Chapter 4 we study the power of polynomial-time instance-targeted adversaries with very small budget size with the goal of targeted-error amplification. We study the privacy implication of machine unlearning in Chapter 5. Finally, in Chapter 6 we summarize the dissertation and discuss its broader impacts.

Chapter 2

Related work

Here we summarize the literature that is related to our dissertation.

2.1 Poisoning attacks

Studying poisoning attacks on machine learning models can be traced back to Barreno et al. [14], which proposes the term *poisoning attacks* for the first time. Later, Biggio et al. [23] presented a poisoning attack strategy on Support Vector Machine (SVM) classifiers, which crafts inputs and inserts them in the training dataset by a gradient ascent method. Following their work, Mei and Zhu [24] proposes a more general optimization framework that crafts poisoning inputs for machine learning algorithms. Their method is optimal if the learning model is trained with a convex loss function. Recently, data poisoning attacks are widely applied to different machine learning scenarios, including recommendation systems [25], graph-based models [26] and federated learning [27].

A typical variant of poisoning attacks on recent state-of-the-art learning models is backdoor attacks, which is a combination of training-stage adversary and testing-stage adversary. In a backdoor attack, malicious parties injects malicious examples to induce specific malicious behavior in the models. The model acts normal when its input is a benign sample, but acts maliciously (e.g., wrongly returns some specific label) when the input includes the pre-defined trigger [28]. In image classification tasks, the trigger is often some specific object, pattern, or texture. Badnets [29] first proposes the backdoor attack on deep learning model in this manner, in which the poisoning samples is constructed by adding the trigger on random benign samples. Chen et al. [30] proposes an attack with minimal, human-unaware trigger. Following this direction, more recent works includes clean-label backdoor [31, 32], dynamic backdoor [33], and distributed backdoor [34].

Note that most proposed poisoning attacks are empirical, that is, gives no theoretical guarantee on learner's error or the attack's performance.

Certification under poisoning attacks. Given a poisoning attack, the predictions of a learning algorithm may or may not change. To this end, Steinhardt et al. [35] initiated the study of *certification* against poisoning attacks, studying the conditions under which a learning algorithm can certifiably obtain an expected low risk. To extend these results to the instance-targeted positing scenario, Rosenfeld et al. [36] recently addressed the *instance targeted* (a.k.a., pointwise) certification with the goal of providing certification guarantees about the prediction of *specific* instances when the adversary can poison the training data. While the instance-targeted certification has sparked a new line of research [37, 38, 39, 40] with interesting insights, the existing works do not address the fundamental question of when, and under what conditions, learnability and certification are achievable under the instance-targeted poisoning attack.

2.2 Privacy of machine learning models

The data-rich nature of machine learning poses a high requirement for preserving data privacy in the model. Differential privacy [41, 42, 43] provides a framework to provably limit the information that would leak about the used training examples. For deep learning, Abadi et al. [44] proposes differentially Private Stochastic Gradient Descent (DP-SGD) algorithm, which allows the training of a learning model with differentially private guarantee. However, applying differential privacy guarantee to machine learning has a cost: major utility loss on the predicted model [45, 46, 47, 48, 49, 50].

A different angle to enhance data privacy in machine learning is to redesign the learning framework so that that the learner is separated with the data. One such example is federated learning [13], which sets up local models for each data collection, and updates the centralized model with gradients from each local model. These methods are irrelevant to our topic here, as we focus on the privacy implication of dataset modification on the predicted model, where model is considered as a black-box.

Membership inference attacks. A typical type of attack on the privacy of machine learning model is membership inference attacks (MIA). Membership of an example is critical to the privacy of a machine learning model. For example, the occurrence of an example in the dataset of a disease study means the data owner has the specific disease [51]. Shokri et al. [4] first define membership inference attacks on machine learning models. They use additional samples from the data distribution to train

several shadow models, and then generate membership records of the samples used in training the shadow models. They then train an attack model from the collected membership records to predict the membership of a specific sample. Their method is later noted to succeed when the generalization gap is large, i.e., highly overfitted. Long et al. [52] shows that even for well-generalized model, there exist several examples that are more vulnerable than other examples and can be identified by the adversary. Yu et al. [53] shows that if data augmentation is included in the model, apply membership inference on the additional augmented data can improves the inference accuracy.

Machine unlearning In light of the recent attention to the "right to erasure" or the "right to be forgotten," also stressed by legal requirements such as GDPR and CCPA, a new line of work has emerged with the goal of *unlearning* or simply *deleting* examples from machine learning models [54, 55, 56, 22, 57, 58, 59, 60]. In this setting, upon a deletion request for a example $e \in \mathcal{S}$, one needs to update the model h to h_{-e} such that h_{-e} is (ideally) the same as training a model from scratch using $\mathcal{S} \setminus \{e\}$. We call the deletion as *perfect deletion* when the $h_{-e} = \mathsf{Lrn}(\mathcal{S} \setminus \{e\})$. These machine unlearning literature propose various methods that focus on how to approach the perfect deletion for different machine learning algorithms and tasks. Therefore, they mostly focus on the property of the model after deletion h_{-e}, seeing the machine unlearning process from a static point of view.

In our work, we revisit the machine unlearning process from a dynamic point of view, that an adversary could observe both h and h_{-e}. We then analysis the privacy implication from this point of view. Similar to our work, recent interesting works of Salem et al. [61] and Chen et al. [62] also study privacy of ML models under updates. The work of [61] focuses on online learning where the model updates add content (rather than removing them) and uses generative models to design their reconstruction attacks. The more recent work of [62], like this work, focuses on deletion updates (i.e., machine unlearning) and is focused on inference attacks. In comparison, our work has a more theoretical flavor and presents a variety of formal definitions (in both cases of inference and reconstruction) that model various subtle aspects of the threat; in particular, we give a comprehensive comparison with the deletion compliance framework of Garg et al [22] and, consequently, to differential privacy (through the lens of deletion compliance) as positive ways to prevent such attacks.

Data auditing One interesting concept related to our topic is data auditing, which can also be view as a requirement posed by the data-protection regulations such as GDPR. Such data-protection

regulations give users the right to know how their data is processed. Therefore, in data auditing, users and the legal enforcement seek proof of how their data is processed. As an example, Ateniese et al.[63] proposes a technique to acquire proof of the data possession on some untrusted cloud storage services.

When comes machine learning services, things become more interesting. A good machine learning model should protect data privacy, so private information about data should be kept from the adversary. However, user would want proof for their data being processed by the model, which contradicts the privacy requirement in some sense. For this topic, Song and Shmatikov [64] proposes an auditing algorithm which learns and predict whether an example is part of the dataset. Their scenario, however, is close to the membership inference where the auditor can be view as a variant of a membership inference adversary. Can we design a framework that encode necessary information into the model, that let the user and law enforcement to verify the membership of an example, but not the adversary? It is an interesting potential future direction to consider.

Chapter 3

Learnability under targeted poisoning

3.1 Introduction

Let $\mathcal{H}$ consists of a hypothesis class of classifiers $h : \mathcal{X} \to \mathcal{Y}$ where $\mathcal{X}$ denotes the instances domain and $\mathcal{Y}$ the labels domain. We would like to study the learnability of $\mathcal{H}$ under instance-targeted poisoning attacks. But before discussing the problem in that setting, we recall the notion of PAC learning *without* attacks.

Informally speaking, $\mathcal{H}$ is "Probably Approximately Correct" learnable (PAC learnable for short) if there is a learning algorithm Lrn such that for every distribution D over $\mathcal{X} \times \mathcal{Y}$, if D can be learned with $\mathcal{H}$ (i.e., the so-called realizability assumption holds) then with high probability over sampling any sufficiently large set $\mathcal{S} \sim D^m$, Lrn maps $\mathcal{S}$ to a hypothesis $h \in \mathcal{H}$ with "arbitrarily small" risk under the distribution D. Lrn is called *improper* if it is allowed to output functions outside $\mathcal{H}$, and it is a *distribution-specific* learner, if it is only required to work when the marginal distribution $D_{\mathcal{X}}$ on the instance domain $\mathcal{X}$ is fixed e.g., to be isotropic Gaussian. (See Section 3.3 and Definition 3.3.5 for formal definitions.)

Suppose that before the example $(x, y) \sim D$ is tested, an adversary who is aware of (x, y) (and hence, is *targeting* the instance x) can craft a poisoned set $\mathcal{S}'$ from $\mathcal{S}$ by *arbitrarily changing* up to b of the training examples in $\mathcal{S}$. Now, the learning algorithm encounters $\mathcal{S}'$ as the training set and the hypothesis it returns is, say, $h' \in \mathcal{H}$ in the proper learning setting. Now, the predicted label of x, i.e., $y' = h'(x)$, may no longer be equal to the correct label y.

Main questions. In this chapter, we would like to study under what conditions on the class complexity $\mathcal{H}$, budget b, and different (weak/strong) forms of instance-targeted poisoning attacks, one can achieve (proper/improper) PAC learning. In particular, the learner's goal is to still be correct, with high probability, on *most* test instances, despite the existence of the attack. A stronger goal than robustness is to also *certify* the predictions $h(x) = y$ with a lower bound k on how much an instance-targeted poisoning adversary needs to change the training set $\mathcal{S}$ to eventually flip the decision on x into $y' \neq y$. Here, we also keep an eye on when robust learners can be enhanced to provide such guarantees, leading to *certifiably robust* learners.

We should highlight that all the aforementioned methods [36, 37, 38, 39, 40] mainly considered practical methods that allow predictions for individual instances under specific conditional assumptions about the model's performance at the decision time that can be only verified empirically, but it is not clear (provably) if such conditions would actually happen during the prediction moment. Here, we avoid such assumptions and address the question of under what conditions on the *problem's setting*, the learnability is possible provably.

Our contribution. Our contributions are as follows.

Formalism. We provide a precise and general formalism for the notions of certification and PAC learnability under instance-targeted attacks. These formalisms are based on a careful treatment of the notions of *risk* and *robustness* defined particularly for learners under instance-targeted poisoning attacks. The definitions carefully consider various attack settings, e.g., based on whether the adversary's perturbation can depend on learner's randomness or not, and also distinguish between various forms of certification (to hold for *all* training sets, or just *most* training sets.)

Distribution-independent setting. We then study the problem of robust learning and certification under instance-targeted poisoning attacks in the distribution-independent setting. Here, the learner shall produce "good" models for *any* distribution over the examples, as long as the distribution can be learned by at least one hypothesis $h \in \mathcal{H}$ (i.e., the realizable setting). We separate our studies here based on the subtle distinction between two cases: Adversaries who can base their perturbation also for a *fixed* randomness of the learner (the default attack setting), and those whose perturbation would be retrained using *fresh* randomness (called weak adversaries). In the first setting, We show that as long as the hypothesis class $\mathcal{H}$ is (properly or improperly) PAC learnable under the 0-1 loss and the strong adversary's budget is $b = o(m)$, where m is the number of samples in the training set, then the hypothesis class $\mathcal{H}$ is always *improperly* PAC learnable under the instance-targeted attack with certification (Theorem 3.4.4). This result is inspired by the recent work of [37] and

comes with certification. We then show that the limitation on $b(m) = o(m)$ is inherent in general, as when $\mathcal{H}$ is the set of homogeneous hyperplanes, if $b(m) = \Omega(m)$, then robust PAC learning against instance-targeted poisoning is impossible in a strong sense (Theorem 3.4.7). m. We then show that if the adversary is "weak" and is *not* aware of learner's randomness, if the hypothesis class $\mathcal{H}$ is properly PAC learnable and the weak adversary's budget is $b = o(m)$, then $\mathcal{H}$ is also properly PAC learnable under instance-targeted attacks (Theorem 3.4.3). This result, however, does *not* come with certification guarantees.

Distribution-specific learning. We then study robust learning under instance-targeted poisoning when the instance distribution is fixed. We show that when the projection of the marginal distribution $D_{\mathcal{X}}$ is the uniform distribution over the unit sphere (e.g., d-dimensional isotropic Gaussian), the hypothesis class consists of homogeneous half-spaces, and the strong adversary's budget is $b = c/\sqrt{d}$, then proper PAC learnability under instant-targeted attack is possible iff $c = o(m)$ (see Theorems 3.4.10 and 3.4.11). Note that if we allow d to grow with m to capture the "high dimension" setting, then the mentioned result becomes incomparable to our above-mentioned results for the distribution-independent setting). To prove this result we use tools from measure concentration over the unit sphere in high dimension.

Experiments. We empirically study the robustness of K nearest neighbour, logistic regression, multi-layer perceptron, and convolutional neural network on real data sets. We observe that methods with high standard accuracy (such as convolutional neural network) are indeed more vulnerable to instance-targeted poisoning attacks. This observation might be explained by the fact that more complex models fit the training data better and thus the adversary can more easily confuse them at a specific test instance. A possible interpretation is that models that somehow "memorize" their data could be more vulnerable to targeted poisoning. In addition, we study whether dropout on the inputs and also $L2$-regularization on the output can help the model to defend against instance-targeted poisoning attacks. We observe that adding these regularization to the learner does not help in defending against such attacks.

3.2 Related work

The concurrent work of Blum et al. [65] also studies instance-targeted PAC learning. In particular, they formalize and prove positive and negative results about PAC learnability under instance-targeted poisoning attacks, in which the adversary can add an unbounded number of clean-label examples to the training set. In comparison, we formalize the problem for any prediction task and also for

certification of results. Our main positive and negative results are, however, proved for classification tasks and for adversaries who can change $o(1)$ fraction of the data set. Other theoretical works have also studied instance-targeted poisoning *attacks* (rather than learnability under such attacks) using clean labels [66, 67, 68, 69, 70, 71, 72]. The work of Shafahi et al. [73] studied such (targeted clean-label) attacks empirically, and showed that neural nets can be very vulnerable to them. Finally, Koh and Liang [74] also studied clean label attacks empirically for *non-targeted* settings.

More broadly, some classical works in machine learning can also be interpreted as (non-targeted) data poisoning [75, 76, 77, 18]. In fact, the work of Bshouty et al. [18] studies the same question as in this chapter, but for the *non-targeted setting*. However, making learners robust against such attacks can easily lead to *intractable* learning methods that do *not* run in polynomial time. Recently, starting with the seminal results of [78, 79] and many follow up works (see the survey [16]) it was shown that in some natural settings one can go beyond the intractability barriers and obtain polynomial-time methods to resist non-targeted poisoning. In contrast, we focus on targeted poisoning. We shall also comment that, while our focus is on instance-targeted attacks for prediction tasks, it is not clear how to even define such (targeted) attacks for robust parameter estimation (e.g., learning Gaussians).

Regarding certification, Steinhardt et al. [35] were the first who studied certification of the *overall risk* under the poisoning attack. However, the more relevant to our paper is the work by Rosenfeld et al. [36] who introduced the instance-targeted poisoning attack and applied randomized smoothing for certification in this setting. Empirically, they showed how smoothing can provide robustness against label-flipping adversaries. Subsequently, Levine and Feizi [37] introduced Deep Partition Aggregation (DPA), a novel technique that uses deterministic bagging in order to develop robust predictions against general addition/removal instance-targeted poisoning. [38, 39, 40] further developed randomized bagging/sub-sampling and empirically studied the intrinsic robustness of their methods. predictions.

Finally, we note that while our focus is on *training-time-only* attacks, poisoning attacks can be performed in conjunction with test time attacks, leading to backdoor attacks [29, 80, 81, 82, 83, 71].

3.3 Definitions

Notation. For a hypothesis class $\mathcal{H}$, we call a data set $\mathcal{S} \sim D^m$ *ε-representative* if $\forall h \in \mathcal{H}, |\text{Risk}(h, D) - \text{Risk}(h, \mathcal{S})| \leq \varepsilon$. A hypothesis class has the *uniform convergence* property, if there is a function $m = m^{\mathcal{H}}_{\mathsf{UC}}(\varepsilon, \delta)$ such that for any distribution D, with probability $1 - \delta$ over $\mathcal{S} \sim D^m$, it holds that $\mathcal{S}$ is ε-representative.

For simplicity, we work with deterministic strategies, even though our results could be extended directly to randomized adversarial strategies as well. We use A to denote an adversary who changes the training set $\mathcal{S}$ into $\mathcal{S}' = \mathsf{A}(\mathcal{S})$. This mapping can depend on (the knowledge of) the learning algorithm Lrn or any other information such as a targeted example e as well as the randomness of Lrn. By $\mathcal{A}$ we refer to a *set* (or *class*) of adversarial mappings and by $\mathsf{A} \in \mathcal{A}$ we denote that the adversary A belongs to this class. (See below for examples of such classes.) Our adversaries always will have a budget $b \in \mathbb{N}$ that controls how much they can change the training set $\mathcal{S}$ into $\mathcal{S}'$ under some (perhaps asymmetric) distance metric. To explicitly show the budget, we denote the adversary as A_b and their corresponding classes as $\mathcal{A}_b$. Finally, we let $\mathcal{A}_b(\mathcal{S}) = \{\mathcal{S}' \mid \mathsf{A}_b \in \mathcal{A}_b(\mathcal{S})\}$ be the set of all "adversarial perturbations" of $\mathcal{S}$ when we go over all possible attacks of budget b from the adversary class $\mathcal{A}$.

Adversary classes. Here we define the main adversary classes that we use in this chapter. For more noise models see the work of [77].

- $\mathcal{R}ep_b$ **(b-replacing).** The adversary can replace up to b of the examples in $\mathcal{S}$ (with arbitrary examples) and then put the whole sequence $\mathcal{S}'$ in an arbitrary order. More formally, the adversary is limited to (1) $|\mathcal{S}| = |\mathcal{S}'|$, and (2) by changing the order of the elements in $\mathcal{S}$, one can make the Hamming distance between $\mathcal{S}', \mathcal{S}$ at most b. This is essentially the targeted version of the "nasty noise" model introduced by [18].

- $\mathcal{F}lip_b$ **(b-label flipping).** The adversary can change the label of up to b examples in $\mathcal{S}$ and reorder the final set.

- $\mathcal{A}dd_b$ **(b-adding).** The adversary adds up to b examples to $\mathcal{S}$ and put them in arbitrary order. Namely, the multi-set $\mathcal{S}'$ has size at most $|\mathcal{S}| + b$ and it holds that $\mathcal{S} \subseteq \mathcal{S}'$.

- $\mathcal{R}em_b$ **(b-removing).** The adversary removes up to b examples from $\mathcal{S}$ and puts the rest in an arbitrary order. Namely, as multi-sets $|\mathcal{S}'| \geq |\mathcal{S}| - b$ and $\mathcal{S}' \subseteq \mathcal{S}$.

- $\mathcal{A}dd\mathcal{R}em_b$ **(b-adding-or-removing).** The adversary can remove up to b examples from $\mathcal{S}$, then add up to b arbitrary examples, and then it puts the rest in an arbitrary order. Namely, as multi-sets $|\mathcal{S}' \cap \mathcal{S}| \geq |\mathcal{S}| - b$ and $|\mathcal{S}' \setminus \mathcal{S}| \leq b$.[1]

[1] $\mathcal{R}ep_b$ attacks are essentially as powerful as $\mathcal{A}dd\mathcal{R}em_b$ attack, with the only limitation that they preserve the training set size. Our results of Theorems 3.4.3 and 3.4.4 extend to $\mathcal{A}dd\mathcal{R}em_b$ attacks as well, however we focus on b-replacing attacks for simplicity of presentation.

We now define the notions of risk, robustness, certification, and learnability under targeted poisoning attacks for prediction tasks with a focus on classification. We emphasize that in the definitions below, the notions of targeted-poisoning risk and robustness are defined with respect to a *learner* rather than a hypothesis. The reason is that, very often (and in many natural settings) when the data set is changed by the adversary, the learner needs to return a new hypothesis, reflecting the change in the training data,

Definition 3.3.1 (Instance-targeted poisoning risk)**.** Let Lrn be a possibly randomized learner, $\mathcal{A}_b$ be a class of attacks of budget b. For a training set $\mathcal{S} \in (\mathcal{X} \times \mathcal{Y})^m$, an example $e = (x, y) \in \mathcal{X} \times \mathcal{Y}$, and randomness r, the *targeted poisoning loss* (under attacks $\mathcal{A}_b$) is defined as[2]

$$\ell_{\mathcal{A}_b}(\mathcal{S}, r, e) = \sup_{\mathcal{S}' \in \mathcal{A}_b(\mathcal{S})} \ell(\mathsf{Lrn}_r(\mathcal{S}'), e). \tag{3.1}$$

For a distribution D over $\mathcal{X} \times \mathcal{Y}$, the *targeted poisoning risk* is defined as

$$\mathrm{Risk}_{\mathcal{A}_b}(\mathcal{S}, r, D) = \mathop{\mathbb{E}}_{e \sim D}[\ell_{\mathcal{A}_b}(\mathcal{S}, r, e)].$$

For a bounded loss function with values in $[0, 1]$ (e.g., the 0-1 loss), we define the *correctness* of the learner for the distribution D under targeted poisoning attacks of $\mathcal{A}_b$ as

$$\mathrm{Cor}_{\mathcal{A}_b}(\mathcal{S}, D) = 1 - \mathrm{Risk}_{\mathcal{A}_b}(\mathcal{S}, D).$$

The above formulation implicitly allows the adversary to depend (and hence "know") on the randomness r of the learning algorithm. We also define *weak targeted-poisoning* loss and risk by using *fresh* learning randomness r unknown to the adversary, when doing the retraining:

$$\ell^{\mathrm{wk}}_{\mathcal{A}_b}(\mathcal{S}, e) = \sup_{\mathcal{S}' \in \mathcal{A}_b(\mathcal{S})} \mathop{\mathbb{E}}_{r}[\ell(\mathsf{Lrn}_r(\mathcal{S}'), e)], \qquad \mathrm{Risk}^{\mathrm{wk}}_{\mathcal{A}_b}(\mathcal{S}, D) = \mathop{\mathbb{E}}_{e \sim D}[\ell^{\mathrm{wk}}_{\mathcal{A}_b}(\mathcal{S}, e)].$$

In particular, having a small weak targeted-poisoning risk under the 0-1 loss means that for most of the points $e \sim D$ the decisions are correct, and the prediction on e would not change under any e-targeted poisoning attacks with high probability over a randomized retraining. ◇

We now define robustness of predictions, which is more natural for classification tasks, but we state it more generally.

[2]Note that Equation 3.1 is equivalent to $\ell_{\mathcal{A}_b}(\mathcal{S}, r, e) = \sup_{\mathsf{A} \in \mathcal{A}_b} \ell(\mathsf{Lrn}_r(\mathsf{A}(\mathcal{S}, r, e)), e)$, because we are choosing the attack over $\mathcal{S}$ after fixing r, e.

Definition 3.3.2 (Robustness under instance-targeted poisoning)**.** Consider the same setting as that of Definition 3.3.1, and let $\tau > 0$ be a threshold to model when the loss is "large enough". For a data set[3] $\mathcal{S}$ and learner's randomness r, we call an example $e = (x, y)$ to be τ-*vulnerable* to a targeted poisoning (of attacks in $\mathcal{A}_b$), if the e-targeted adversarial loss is at least τ, namely, $\ell_{\mathcal{A}_b}(\mathcal{S}, r, e) \geq \tau$. For the same $(\mathcal{S}, r, e, \tau)$ we define the *targeted poisoning robustness* (under attacks in $\mathcal{A}$) as the smallest budget b such that e is τ-vulnerable to a targeted poisoning, i.e.,

$$\mathrm{Rob}^{\tau}_{\mathcal{A}}(\mathcal{S}, r, e) = \inf\left\{b \mid \ell_{\mathcal{A}_b}(\mathcal{S}, r, e) \geq \tau\right\}.$$

If no such b exists, we let $\mathrm{Rob}^{\tau}(\mathcal{S}, r, e) = \infty$.[4] When working with the 0-1 loss (e.g., for classification), we will use $\tau = 1$ and simply write $\mathrm{Rob}_{\mathcal{A}}(\cdot)$ instead. Also note that in this case, $\ell(\mathsf{Lrn}_r(\mathcal{S}'), e) \geq 1$ is simply equivalent to $\mathsf{Lrn}_r(\mathcal{S}')(x) \neq y$. In particular, if $e = (x, y)$ is an example and $\mathsf{Lrn}_r(\mathcal{S})$ is already wrong in its prediction of the label for x, then the robustness will be $\mathrm{Rob}_{\mathcal{A}}(\mathcal{S}, r, e) = 0$, as no poisoning will be needed to make the prediction wrong. For a distribution D we define the *expected targeted-poisoning robustness* as $\mathrm{Rob}^{\tau}_{\mathcal{A}}(\mathcal{S}, r, D) = \mathbb{E}_{e\sim D}[\mathrm{Rob}^{\tau}_{\mathcal{A}}(\mathcal{S}, r, e)]$. ◇

We now formalize when a learner provides certifying guarantees for the produced predictions. For simplicity, we state the definition for the case of 0-1 loss, but it can be generalized to other loss functions by employing a threshold parameter τ as it was done in Definition 3.3.2.

Definition 3.3.3 (Certifying predictors and learners)**.** A *certifying predictor* (as a generalization of a hypothesis function) is a function $h\colon \mathcal{X} \to \mathcal{Y} \times \mathbb{N}$, where the second output is interpreted as a claim about the robustness of the prediction. When $h(x) = (y, b)$, we define $h_{\mathrm{pred}}(x) = y$ and $h_{\mathrm{cert}}(x) = b$. If $h_{\mathrm{cert}}(x) = b$, the interpretation is that the prediction y shall not change when the adversary performs a b-budget poisoning perturbation (defined by the attack model) over the training set used to train h.[5] Now, suppose $\mathcal{A}_b$ is an adversary class with budget $b = b(m)$ (where m is the sample complexity) and $\mathcal{A} = \cup_i \mathcal{A}_i$. Also suppose Lrn is a learning algorithm such that $\mathsf{Lrn}_r(\mathcal{S})$ always outputs a certifying predictor for any data set $\mathcal{S} \in (\mathcal{X} \times \mathcal{Y})^{\star}$. We call Lrn a *certifying learner* (under the attacks in $\mathcal{A}$) for a specific data set $\mathcal{S} \in (\mathcal{X} \times \mathcal{Y})^{\star}$ and randomness r, if the following holds. For all $x \sim D$, if $\mathsf{Lrn}_r(\mathcal{S})(x) = (y, b)$ and if we let $e = (x, y)$,[6] then $\mathrm{Rob}_{\mathcal{A}}(\mathcal{S}, r, e) \geq b$. In other words, to change the prediction y on x (regardless of y being a correct prediction or not), any adversary

[3] Even though, in natural attack scenarios the set $\mathcal{S}$ is sampled from D^m, Definitions 3.3.1 and 3.3.2 are more general in the sense that $\mathcal{S}$ is an arbitrary set.

[4] If the adversary's budget allows it to flip all the labels, in natural settings (e.g., when the hypothesis class contains the complement functions and the learner is a PAC learner), no robustness will be infinite for such attacks.

[5] When using a general loss function, b would be interpreted as the attack budget that is needed to increase the loss over the example $e(x, y)$ (where y is the prediction) to τ.

[6] Note that y might not be the right label

needs a budget at least b. We call Lrn a *universal* certifying learner if it is a certifying learning for all data sets $\mathcal{S}$. For an adversary class $\mathcal{A} = \cup_{b\in\mathbb{N}}\mathcal{A}_b$, and a certifying learner Lrn for $(\mathcal{S}, r)$, we define the b-certified correctness of Lrn over $(\mathcal{S}, r, D)$ as the probability of outputting correct predictions while certifying them with robustness at least b. Namely,

$$\mathrm{CCor}_{\mathcal{A}_b}(\mathcal{S}, r, D) = \Pr_{(x,y)\sim D}\left[(y' = y) \wedge (b' \geq b) \text{ where } (y', b') = \mathsf{Lrn}_r(\mathcal{S})(x)\right]. \quad \diamond$$

Remark 3.3.4 (On a potential weaker requirement for certifying learners)**.** Definition 3.3.3 needs a learner to produce a certifying model that is *always* correct in its robustness claims about its own prediction, regardless of whether the prediction itself is correct or wrong. One can imagine a weaker certification requirement in which the provided certified robustness guarantee is only required to hold when the predicted label itself is correct. However, since a learner usually does not really know whether its prediction is correct with full confidence, known methods for certified robustness already achieve the stronger guarantee of in Definition 3.3.3. Also, if one uses that weaker requirement, *robust* PAC learning and *certified* PAC learning (see Definition 3.3.5) become equivalent, as a learner can simply output b as its certifying guarantee when we know that robust PAC learning against targeted b-budget poisoning attacks is possible.

The following definition extends the standard PAC learning framework of [84] by allowing targeted-poisoning attacks and asking the leaner now to have small targeted-poisoning risk. This definition is strictly more general than PAC learning, as the trivial attack that does not change the training set, Definition 3.3.5 below reduces to the standard definition of PAC learning.

Definition 3.3.5 (Learnability under instance-targeted poisoning)**.** Let the function $b\colon \mathbb{N} \to \mathbb{N}$ model adversary's budget as a function of sample complexity m. A hypothesis class $\mathcal{H}$ is *PAC learnable under targeted poisoning attacks in* $\mathcal{A}_b$, if there is a proper learning algorithm Lrn such that for every $\varepsilon, \delta \in (0, 1)$ there is an integer m where the following holds. For every distribution D over $\mathcal{X} \times \mathcal{Y}$, if the realizability condition holds[7] (i.e., $\exists h \in \mathcal{H}, \mathrm{Risk}(h, D) = 0$), then with probability $1 - \delta$ over the sampling of $\mathcal{S} \sim D^m$ and Lrn's randomness r, it holds that $\mathrm{Risk}_{\mathcal{A}_b}(\mathcal{S}, r, D) \leq \varepsilon$.

- **Improper learning.** We say that $\mathcal{H}$ is *improperly* PAC learnable under targeted $\mathcal{A}_b$-poisoning attacks, if the same conditions as above hold but using an improper learner that might output functions outside $\mathcal{H}$.[8]

[7] Note that realizability holds while no attack is launched.
[8] We note, however, that whenever the proper or improper condition is not stated, the default is to be proper.

- **Distribution-specific learning.** Suppose $\mathcal{D}$ is the set of all distributions D over $\mathcal{X} \times \mathcal{Y}$ such that the marginal distribution of D over its first coordinate (in $\mathcal{X}$) is a fixed distribution $D_{\mathcal{X}}$ (e.g., isotropic Gaussian in dimension d). If all the conditions above (resp. for the improper cases) are only required to hold for distributions $D \in \mathcal{D}$, then we say that the hypothesis class $\mathcal{H}$ is PAC learnable (resp. improperly PAC learnable) under instance distribution $D_{\mathcal{X}}$ and targeted $\mathcal{A}_b$-poisoning.

A hypothesis class is *weakly* (improperly and/or distribution-specific) PAC learnable under targeted $\mathcal{A}_b$-poisoning, if with probability $1-\delta$ over the sampling of $\mathcal{S} \sim D^m$, it holds that $\mathrm{Risk}^{\mathrm{wk}}_{\mathcal{A}_b}(\mathcal{S}, D) \leq \varepsilon$. A hypothesis class is *certifiably* (improperly and/or distribution-specific) PAC learnable under targeted $\mathcal{A}_b$-poisoning, if we modify the (ε, δ) learnability condition as follows. With probability $1-\delta$ over $\mathcal{S} \sim D^m$ and randomness r, it holds that (1) Lrn is a certifying learner for $(\mathcal{S}, r)$, and (2) $\mathrm{CCor}_{\mathcal{A}_b}(\mathcal{S}, r, D) \geq 1-\varepsilon$. A hypothesis class is *universally* certifiably PAC learnable, if it is certifiably PAC learnable using a universal certifying learner Lrn. We call the sample complexity of any learner of the forms above *polynomial*, if the sample complexity m is at most $\mathrm{poly}(1/\varepsilon, 1/\delta) = (1/(\varepsilon\delta))^{O(1)}$. We call the learner *polynomial time*, if it runs in time $\mathrm{poly}(1/\varepsilon, 1/\delta)$, which implies the sample complexity is polynomial as well. ◇

Remark 3.3.6 (Generalization to (ε, δ)-PAC learning)**.** Suppose $\varepsilon(m), \delta(m)$ are functions of m. Then one can generalize Definition 3.3.5 to define $(\varepsilon(m), \delta(m))$ PAC learning (under the same settings of Definition 3.3.5) for a given desired $\varepsilon(m), \delta(m)$. Then PAC learnability would simply mean $\varepsilon(m), \delta(m)$ PAC learning for $\varepsilon(m), \delta(m) = o_m(1)$ (i.e., $\varepsilon(m), \delta(m)$ both go to zero, when m goes to infinity). This more fine-grained definition allows one to study *optimal error* bounds in relation to adversary's budget $b(m)$ as well. We leave a more in-depth study of such relations for future work.

Remark 3.3.7 (On defining agnostic learning under instance-targeted poisoning)**.** Definition 3.3.5 focuses on the realizable setting. However, one can generalize this to the agnostic (non-realizable) case by requiring the following to hold with probability $1-\delta$ over $\mathcal{S} \sim D^m$ and randomness r,

$$\mathrm{Risk}_{\mathcal{A}_b}(\mathcal{S}, r, D) \leq \varepsilon + \inf_{h \in \mathcal{H}} \mathrm{Risk}(h, r, D).$$

Note that in this definition the learner wants to achieve *adversarial* risk that is ε-close to the risk under *no attack*. One might wonder if there is an alternative definition in which the learner aims to "ε-compete" with the best *adversarial* risk. However, recall that targeted-poisoning adversarial risk is *not* a property of the hypothesis, and it is rather a property of the learner. This leads to the

following arguably unnatural criteria that needs to hold with probability $1-\delta$ over $S \sim D^m$ and r. (For clarity the learner is explicitly denoted as super-index for $\mathrm{Risk}_{\mathcal{A}_b}$.)

$$\mathrm{Risk}^{\mathsf{Lrn}}_{\mathcal{A}_b}(\mathcal{S}, r, D) \leq \varepsilon + \inf_{L} \mathrm{Risk}^{L}_{\mathcal{A}_b}(\mathcal{S}, r, D)$$

The reason that the above does not trivially hold is that Lrn needs to satisfy this for *all* distributions D (and most $\mathcal{S}$) simultaneously, while the learner L in the right hand side can depend on D and $\mathcal{S}$.

3.4 Our results

We now study the question of learnability under instance-targeted poisoning. We first discuss our positive and negative results in the context of distribution-independent learning. We then turn to the setting of distribution-dependent setting. At the end, we prove some generic relations between risk and robustness, showing how to derive one from the other.

3.4.1 Distribution-independent learning

We start by showing results on distribution-independent learning. We first show that in the realizable setting, for any hypothesis class $\mathcal{H}$ that is PAC-learnable, $\mathcal{H}$ is also PAC learnable under instance-targeted poisoning attacks that can replace up to $b(m) = o(m)$ (e.g., $b(m) = \sqrt{m}$) number of examples arbitrarily. To state the bound of sample complexity of robust learners, we first define the $\lambda(\cdot)$ function based an adversary's budget $b(m)$.

Definition 3.4.1 (The $\lambda(\cdot)$ function)**.** Suppose $b(m) = o(m)$. Then for any real number x, $\lambda(x)$ returns the minimum m where $m'/b(m') \geq x$ for any $m' > m$. Formally,

$$\lambda(x) = \inf_{m \in \mathcal{N}} \left\{ \forall m' \geq m, \frac{m'}{b(m')} \geq x \right\}.$$

Note that because $b(m) = o(m)$, we have $m/b(m) = \omega_m(1)$, so $\lambda(x)$ is well-defined. ◇

Claim 3.4.2 (When λ is polynomially bounded)**.** *If $b(m) = O(x^{1-c})$ for any constant $c > 0$, then $\lambda(x) = O(m^{1/c})$, which means $\lambda(\cdot)$ is a polynomial function. For example, when $b(m) = O(\sqrt{m})$, then $\lambda(x) = O(x^2)$.*

Proof. As $b(m) = O(m^{1-c})$, there exists a number m_0 and a constant q, that for any $m' \geq m_0$, we have $b(m') \leq q \cdot (m')^{1-c}$, which indicates $m'/b(m') \geq q \cdot (m')^c$. By the definition of $\lambda(x)$, we want to show

that for any $m \geq \lambda(x)$, we have $m/b(m) \geq x$. Let $m_1 = (x/q)^{1/c}$, then when $x \geq q \cdot m_0^c$, we have $m_1 \geq m_0$. By Definition 3.4.1, $m_1/b(m_1) \geq q \cdot m_1^c = x$. Therefore, $m_1 \in \{\forall m' \geq m, m'/b(m') \geq x\} \geq \lambda(x)$. Since $m_1 = O(x^{1/c})$, we have $\lambda(x) = O(x^{1/c})$. □

Theorem 3.4.3 (Proper learning under weak instance-targeted poisoning)**.** *Let $\mathcal{H}$ be the PAC learnable class of hypotheses. Then, for adversary budget $b(m) = o(m)$, the same class $\mathcal{H}$ is also PAC learnable using randomized learners under* weak *b-replacing targeted-poisoning attacks. The proper/improper nature of learning remains the same. Specifically, let $m_{\mathsf{Lrn}}(\varepsilon, \delta)$ be the sample complexity of a PAC learner* Lrn *for $\mathcal{H}$. Then, there is a learner* WR *that PAC learns $\mathcal{H}$ under weak b-replacing attacks with sample complexity at most*

$$m_{\mathsf{WR}}(\varepsilon, \delta) = \lambda\left(\max\left\{m_{\mathsf{Lrn}}^2\left(\varepsilon, \frac{\delta}{2}\right), \frac{4}{\delta^2}\right\}\right).$$

Moreover, if $b(m) \leq O(m^{1-\Omega(1)})$, then whenever $\mathcal{H}$ is learnable with a polynomial sample complexity and/or a polynomial-time learner Lrn*, the robust variant* WR *will have the same features as well.*

Proof of Theorem 3.4.3. We first clarify that if $b(m) \leq O(m^{1-\Omega(1)})$, and if $\mathcal{H}$ is learnable with a polynomial sample complexity, then the polynomial sample complexity of the robust variant simply follows from Claim 3.4.2 and the formula for $m_{\mathsf{WR}}(\varepsilon, \delta)$ as stated in the statement of the theorem. Moreover, the polynomial-time nature of our learner (assuming $\mathcal{H}$ is polynomial-time learnable) would be straightforward based on its description below.

The idea is to show that even a simple sub-sampling of the right size from the given training set $\mathcal{S}$, and then training a model over the sub-sample will do what we want. In particular, we will randomly choose k of the elements in $\mathcal{S}$, call it subset $\mathcal{S}_k$, and then run any oracle learner for hypothesis class $\mathcal{H}$. Below, we will first describe how we choose k. We will then prove specific properties about the designed learning algorithm, and finally we will analyze its robustness to weak instance-targeted poisoning attacks (who do not know learner's randomness for retraining). We call the new learner WR, and denote the oracle that provides learners for $\mathcal{H}$, simply as Lrn.

Let $k = k(m) = \sqrt{m/b(m)}$. By the definition of $\lambda(x)$, we have that $\forall m \geq \lambda(x)$, $m/b(m) \geq x$. For simplicity of notation we might write k and b where both are actually functions of m.

Let $m_{\mathsf{Lrn}}(\varepsilon, \delta)$ be the sample complexity of the Lrn which returns a hypothesis with error ε for at least $1 - \delta$ probability. We now show that when the sample complexity $m \geq m_{\mathsf{WR}}(\varepsilon, \delta) = \lambda(\max\{m_{\mathsf{Lrn}}^2(\varepsilon, \delta/2), 4/\delta^2\})$ the learner WR becomes an (ε, δ)-robust PAC learner. Note that by the

definition of $\lambda(\cdot)$, we have

$$\frac{m}{b(m)} \geq \max\left\{m^2_{\mathsf{Lrn}}\left(\varepsilon, \frac{\delta}{2}\right), \frac{4}{\delta^2}\right\}.$$

We then have $\sqrt{m/b(m)} \geq m_{\mathsf{Lrn}}(\varepsilon, \delta/2)$ and $\sqrt{m/b(m)} \geq \frac{2}{\delta}$.

Warm up: PAC learnability without attack. It holds that $k = \sqrt{m/b} \geq m_{\mathsf{Lrn}}(\varepsilon, \delta/2)$. Hence, $\mathsf{WR}(\mathcal{S}) = \mathsf{Lrn}(\mathcal{S}_k)$ will be a PAC learner which returns a hypothesis of at most ε with at least $1 - \delta/2$ probability, in the case no attack happens.

Robustness under weak attacks. Now suppose an adversary can change up to b of the examples through a weak b-replacing attack. The probability that the subset $\mathcal{S}_k$ intersects with any of the k poisoned examples is at most

$$p(m) = \frac{k \cdot b}{m} = \sqrt{\frac{m}{b}} \cdot \frac{b}{m} = \sqrt{\frac{b}{m}} \leq \frac{\delta}{2}.$$

Therefore, with probability at least $1 - p(m)$, none of the poison examples that are introduced by the adversary will land in the subset $\mathcal{S}_k$. In this case by a union bound, when learner Lrn is an $(\varepsilon, \delta/2)$ PAC learner, learner WR will be a $(\varepsilon, \delta/2 + p(m))$ PAC learner under weak b-replacing instance-targeted poisoning attacks. As $\delta/2 + p(m) \leq \delta$, WR with at least $1 - \delta$ probability will return a hypothesis that has at most ε risk under weak b-replacing attacks. □

The above theorem shows that targeted-poisoning-robust proper learning is possible for PAC learnable classes using *private* randomness for the learner if $b(m) = o(m)$. Thus, it is natural to ask the following question: can we achieve the stronger (default) notion of robustness as in Definition 3.3.5 in which the adversarial perturbation can also depend on the (fixed) randomness r of the learner? Also, can this be a learning with certifications? Our next theorem answers these questions positively, yet that comes at the cost of improper learning. Interestingly, the improper nature of the learner used in Theorem (3.4.4) could be reminiscent of the same phenomenon in *test-time* attacks (a.k.a., adversarial example) where, as it was shown by [85], improper learning came to rescue as well.

Theorem 3.4.4 (Improper learning and certification under targeted poisoning)**.** *Let $\mathcal{H}$ be (perhaps improperly) PAC learnable. If b-replacing attacks have their budget limited to $b(m) = o(m)$, then $\mathcal{H}$ is improperly certifiably PAC learnable under b-replacing targeted poisoning attacks. Specifically, let $m_{\mathsf{Lrn}}(\varepsilon, \delta)$ be the sample complexity of a PAC learner for $\mathcal{H}$. Then there is a learner* Rob *that*

universally certifiably PAC learns $\mathcal{H}$ under b-replacing attacks with sample complexity at most

$$m_{\mathsf{Rob}}(\varepsilon,\delta) = 576\lambda\left(\max\left\{m^2_{\mathsf{Lrn}}\left(\frac{\varepsilon}{12},\frac{\varepsilon}{12}\right), \frac{1}{4\varepsilon^2}, \frac{\log\left(\frac{\delta}{2}\right)^2}{\left(\frac{2\sqrt{3}\varepsilon}{3}\right)^4}, \frac{\log_2\left(\frac{2}{\delta}\right)}{576}\right\}\right).$$

Moreover, if $b(m) \leq O(m^{1-\Omega(1)})$ and $\mathcal{H}$ is learnable using a learner with a polynomial sample complexity and/or time, the robust variant Rob *will have the same features as well.*

Before proving Theorem 3.4.3, we define the notion of majority ensembles.

Definition 3.4.5 (Majority ensemble)**.** A *majority ensemble* model h_{ens} is defined over t sub-models $\{h_1,\ldots,h_t\}$ as follows.

$$h_{\mathsf{ens}}(x) = \operatorname*{argmax}_{y\in\mathcal{Y}} \sum_{i=1}^{t} \mathbb{1}[h_i(x) = y].$$

Where $\mathbb{1}[E]$ is the Boolean indicator function that equals 1 if E is true. If no strict majority vote exists, then $h_{\mathsf{ens}}(x) = \perp$ for some fixed output $\perp$. ◇

Proof of 3.4.4. Similar to the proof of Theorem 3.4.3, if $b(m) = O(m^{1-\Omega(1)})$, the relation between polynomial sample complexity and polynomial time aspects of the certifying Rob in relation to the base learner Lrn follows from Claim 3.4.2, the polynomial bound $m_{\mathsf{Rob}}(\varepsilon,\delta)$, and the description of our learner Rob below.

Recall that Lrn is a (ε',δ') PAC learner and our goal is to show that we can obtain (ε,δ)-PAC learning under b-replacing targeted-poisoning attacks. We will indeed show how to achieve $(O(\varepsilon'+\delta'), O(\varepsilon'+\delta'))$-PAC learning under such attacks.

We first describe a learning method in which the b-replacing adversary is *not* allowed to reorder the examples after changing b of the examples in $\mathcal{S}$. Our robust learner in this case is deterministic. We will then discuss how one can retain the result by handling even when the adversary can reorder the examples. Our robust learner for the latter case is randomized and uses a careful hashing method. This learner is inspired by the randomized method first introduced in [37]. In comparison, (1) we need to generalize the hashing method of [37] and carefully choose how to hash *repeated* examples in the data set, and (2) we give a proof of generalization based on adversary's budget.

Attacks that do not reorder the examples. We define the operation *partition* with size k as repeatedly collecting first k items in the data set $\mathcal{S}$ (which is defined as a sequence), that is, when partition data set $\mathcal{S} = e_1, e_2, \ldots, e_m$ with size k, the first partition $\mathcal{S}_1$ will contain examples

$e_1, e_2, \ldots, e_k$, and the second partition $\mathcal{S}_2$ will contain examples $e_{k+1}, e_{k+2}, \ldots, e_{2k}$. Now, let $t = t(m) = \sqrt{b(m) \cdot m}$. RLrn proceeds as follows.

1. Partition the data set $\mathcal{S}$ into t subsets $\mathcal{S}_1, \ldots, \mathcal{S}_t$ with equal size m/t.

2. For each subset $\mathcal{S}_i$ where $i \in [t]$, train a sub-model $h_i = \mathsf{Lrn}(\mathcal{S}_i)$.

3. Returns h_{ens} that is the majority ensemble model of $\{h_1, \ldots, h_t\}$.

If $t = t(m) = \sqrt{m \cdot b(m)}$, $\varepsilon' = \varepsilon/12$, $\delta' = \varepsilon/12$, and $p = \max\{m^2_{\mathsf{Lrn}}(\varepsilon/12, \varepsilon/12), 144/\varepsilon^2, -\log(\delta)/\big(2(\varepsilon/12)^2\big)\}$, we show that $\lambda(p)$ becomes an upper bound on the sample complexity m of a robust PAC learner under b-replacing attacks. By the definition of the function $\lambda(\cdot)$, we have $m/b(m) \geq p$. Therefore, we have $\sqrt{m/b(m)} \geq m_{\mathsf{Lrn}}(\varepsilon/12, \varepsilon/12)$, $\sqrt{m/b(m)} \geq 12/\varepsilon$, and $\sqrt{m/b(m)} \geq -\log(\delta)/\big(2(\varepsilon/12)^2\big)$. For simplicity of notation we might write t and b directly where both are actually functions of m.

We start by showing the learner RLrn has the following two properties:

- PAC learnability of each sub-model without attack: Each set $\mathcal{S}_i$ has m/t examples. Therefore, eventually all the partition sets $\mathcal{S}_i, i \in [t]$ will have enough examples for PAC learning. Specifically, $m/t = \sqrt{m/b} \geq m_{\mathsf{Lrn}}(\varepsilon/12, \varepsilon/12)$.

- Not many sub-models are under attack: An adversary who can replace b examples in these t sets, is indeed affecting only t/b fraction of the subsets, and $t = \sqrt{b \cdot m}$, $b/t = \sqrt{b/m} \leq \varepsilon/12$.

The above arguments show that for each sub-model h_i, we can guarantee (ε', δ')-PAC learning using the number of samples $m_{\mathsf{Lrn}}(\varepsilon/12, \varepsilon/12)$ that falls into the corresponding $\mathcal{S}_i$. Then, we want to argue that the ensemble h_{ens}, which is the majority applied to $h_1, \ldots, h_t$, is indeed $(O(\varepsilon'+\delta'), O(\delta'+\varepsilon'))$-PAC learning even under b-budget changing adversaries (who do not reorder the new set $\mathcal{S}'$).

We will first argue about why the obtained ensemble model *without attack* has small risk, and once we do it, we argue why it has small risk even under b-replacing attacks who do not reorder the output examples.

We start by showing that with high probability, most sub-models have small risk. One might be tempted to use the union bound and conclude that with probability $1 - t \cdot \delta'$ all of $h_1, \ldots, h_t$ have risk at most ε', before arguing about the low risk of their majority. But this is a lose confidence bound as $t \cdot \delta'$ can grow to be larger than one. Hence, we need a more careful analysis. In particular, we use concentration bounds to conclude that with high probability *most* of the sub-models have risk at most ε'. Namely, using the Hoeffding inequality, we can conclude that with probability

at least $1 - e^{-2t\cdot\delta'^2}$, it holds that the fraction of $h_1, \ldots, h_t$ with risk at most ε is at most $2\delta'$. When $m \geq m_{\mathsf{Rob}}(\varepsilon, \delta)$, we have $t = \sqrt{m \cdot b(m)} \geq \sqrt{m/b} \geq -\log(\delta)/\big(2(\varepsilon/12)^2\big) = -\log(\delta)/\big(2\delta'^2\big)$. As $1 - e^{-2t\cdot\delta'^2} \geq 1 - e^{-2\cdot(-\log(\delta)/2\delta'^2)\cdot\delta'^2} = 1 - \delta$. In that case, we can argue about the robustness of the majority ensemble as follows.

Recall that at this stage we are assuming $1-2\delta'$ fraction of the models $h_1, \ldots, h_t$ have risk at most ε. We claim that if we let $\varepsilon = 3(2\delta' + \varepsilon')$, then with probability at least $1 - \varepsilon'$ over $e = (x, y) \sim D$, it holds that at least $2t/3$ of the sub-models $h_1, \ldots, h_t$ give the right answer y on instance x. Otherwise we can derive a contradiction as follows. Suppose more than ε fraction of the examples $e = (x, y) \sim D$ have at least $t/3$ wrong answers among $h_1, \ldots, h_t$, i.e., $\Pr_{(x,y)\sim D}\left[\sum_{i=1}^{t} \mathbb{1}[h_i(x) \neq y] \geq t/3\right] > \varepsilon$. Then, when we pick both $i \sim [t]$, and $e = (x, y) \sim D$ at random and get $h_i(x)$ as answer, we get a wrong answer with probability more than $\varepsilon/3$. On the other hand, this probability cannot be too large, because at most $2\delta'$ fraction of $i \sim [t]$ give a model h_i with risk more than ε', and the rest have risk at most ε', and hence we should have $\varepsilon/3 < 2\delta' + \varepsilon'$, which contradicts $\varepsilon = 3(2\delta' + \varepsilon')$.

Now, we argue that essentially the same bounds above hold even if an adversary goes back and changes b of the examples among the all m examples based on knowing a test example. The only place in the proof that we need to modify is where we obtained $\varepsilon/3 \leq 2\delta' + \varepsilon'$, while now we shall allow the adversary to corrupt b of the t sub-models by planting wrong examples into their pool $\mathcal{S}_i$. This can only corrupt b/t fraction of the t models, leading to the bound $\varepsilon = 3(2\delta' + b/t + \varepsilon')$.

As a summary, with $\varepsilon' = \varepsilon/12$ and $\delta' = \varepsilon/12$, when $m_{\mathsf{Rob}}(\varepsilon, \delta) = \lambda(p)$ and $t = \sqrt{b \cdot m}$, the majority learner is an (ε, δ)-PAC learner to b-replacing attacks that do not reorder the examples, as with probability at least $1 - e^{-2t\,\delta'^2} \geq 1 - \delta$, the robust risk of the learner is at most

$$3\left(2\delta' + \frac{b}{t} + \varepsilon'\right) = 3\left(2\cdot\frac{\varepsilon}{12} + \sqrt{\frac{b}{m}} + \frac{\varepsilon}{12}\right) \leq 3\left(2\cdot\frac{\varepsilon}{12} + \frac{\varepsilon}{12} + \frac{\varepsilon}{12}\right) = \varepsilon.$$

Adding certification. Finally, we define a certifying model h_{cert} that returns certifications larger than b with high probability. Let

$$h_{\mathrm{cert}}(x) = \sum_{i=1}^{t} \mathbb{1}\left\{h_i(x) = y'\right\} - \frac{t}{2}$$

where $y' = h_{\mathsf{ens}}(x)$ and $h_1, \ldots, h_t$ are sub-models in h_{ens}. As the sub-models $h_1, \ldots, h_t$ are trained with separate data sets, for any $b' < h_{\mathrm{cert}}(x)$, the prediction of h_{ens} remains the same, indicates that

h_{cert} always gives correct certification. Now, from the previous analysis, we have

$$\Pr_{\mathcal{S}}\left[\text{CCor}_{\mathcal{R}ep_b}(\mathcal{S}, D) \geq 1-\varepsilon\right] \geq 1-\delta.$$

Therefore, $\mathcal{H}$ is certifiably PAC learnable under $\mathcal{R}ep_b$ attacks with the aforementioned upper bound on its sample complexity.

Attacks that might reorder the examples. The above learner was indeed deterministic, but it leveraged on the fact that the adversary will not reorder the examples, hence most sub-models are robust to adversarial perturbations. For the full-fledged b-replacing adversaries, we will use randomness r that (informally speaking) defines a hash function from $\mathcal{X}\times\mathcal{Y}$ to $[t]$. The hash function can either be a random oracle, or an m-wise independent function (for sake of a polynomial-time learner). We then partition the training set $\mathcal{S}$ into t subsets by using the hash function that looks at *individual* examples to determine where they land among the t subsets $\mathcal{S}_1, \ldots, \mathcal{S}_t$.

Because we did not make any assumptions about distribution D, the training set $\mathcal{S}$ could have multiple instances of the same input if D is concentrated on some examples. If we simply pick a hash function h to map $\mathcal{X}\times\mathcal{Y}$ to $[t]$, it might make the subsets unbalanced and thus lose the i.i.d. property of the distributions generating subsets $\mathcal{S}_i$.

We then slightly revise the rule to evenly distributed these examples as follows. For an example $e_i = (x_i, y_i)$ in the training set $\mathcal{S}$, let O_i be the number of occurrence of the same example (x_i, y_i) in $\mathcal{S}$ (0 if it's the first occurrence). We then use a hash function family $h_K : \mathcal{X}\times\mathcal{Y}\times[m] \to [t]$, where K is a key generated by r. The j-th occurrence of e_i is then mapped into the partition t_i where $t_i = h_K(e_i, j)$.

Following our assumption of the hash function being independently random on all elements in $\mathcal{S}$, each partition $\mathcal{S}_i$ is now an i.i.d. sample of the same distribution. It is because each example in $\mathcal{S}_i$ is independently and identically sampled from $\mathcal{S}$, which is an i.i.d. sample of D. Therefore, with enough number of examples in $\mathcal{S}_i$, by the PAC learnablity of $\mathcal{H}$, each sub-model h_i will be a PAC learner. However, for a pair of (ε, δ), it is not guaranteed that $\mathcal{S}_i$ has enough number of examples for (ε, δ)-PAC learning, because we are using a probabilistic hashing. If some of the sub-models do not have enough examples in their pool $\mathcal{S}_i$, it is then hard to show the majority ensemble model is a good model with error less than ε. To handle this problem, we only train sub-models on the partitions with enough number of examples.

We pick $t = 4\sqrt{b(m)\cdot m}$ be the number of subsets. RLrn proceeds as follows.

1. For the j-th occurrence of the example $e_i \in \mathcal{S}$, add it into partition $\mathcal{S}_{t_i}$ where $t_i = h_K(e_i, j)$.
2. For each subset $\mathcal{S}_i$ that $|\mathcal{S}_i| \geq m/6t$ where $i \in [t]$, train a sub-model $\mathsf{Lrn}(\mathcal{S}_i)$.
3. Denote all the sub-models trained in Step 2 as $h_1, h_2, \ldots, h_{t'}$.
4. Return h_{ens}, the majority ensemble model of $\{h_1, \ldots, h_{t'}\}$.

Here, the majority ensemble model will have t' (instead of t) sub-models, and $t' \leq t$. We now show that when $p' = \max\left\{m^2_{\mathsf{Lrn}}(\varepsilon/12, \varepsilon/12), 1/4\varepsilon^2, \log(\delta/2)^2/((2\sqrt{3}\varepsilon/3)^4), \log_2(2/\delta)/576\right\}$ with the sample complexity bounded by $m \geq m_{\mathsf{Rob}}(\varepsilon, \delta) = 576\lambda(p')$, RLrn is robust to b-replacing attacks that can reorder the examples.

First, we prove that the majority of the partitions $\mathcal{S}_i$ will have enough samples, specifically, at least $t' \geq t/4$ sub-models will have $m/6t$ examples with high probability.

To analyze the probability of $t' \geq t/4$, we first consider a simple bucket and ball setting. Consider there are $2t$ examples (balls) and we partition them into t subsets (buckets). Then the probability that at least $t/2$ buckets are not empty is at least

$$1 - \binom{t}{t/2}\left(\frac{1}{2}\right)^{2t} = 1 - \binom{t}{t/2}\left(\frac{1}{2^t}\right) \cdot \left(\frac{1}{2^t}\right) \geq 1 - \frac{1}{2^t}.$$

It is because if there are $t/2$ empty buckets, then all $2t$ balls should be in the other $t/2$ buckets. The probability is then calculated by taking a union bound over all $\binom{t}{t/2}$ choices of $t/2$ empty buckets in t buckets.

Now, we have m examples in total. We then consider m examples as $m/2t$ rounds of $2t$ examples. Then for each round, at least $t/2$ subsets have at least one example with probability at least $1 - 1/2^t$. Clearly, applying the union bound over all the rounds of examples gives the result that with probability $1 - m/(2t \cdot 2^t)$, every round makes at least $t/2$ buckets non-empty. Then, by a simple counting argument, at the end at least $t/4$ buckets will have at least $m/6t$ examples. (Otherwise, the total number of examples would be fewer than $(t/2)(m/3t)$.)

We now prove some properties for RLrn. Let $\varepsilon' = \delta' = \frac{\varepsilon}{12}$, when $m = \lambda(p')$. Then, we have

- Not many sub-models are under attack: An adversary who can corrupt b of these $t/4$ sets, is indeed corrupting only $4b/t$ fraction of them. We then have $4b/t = \sqrt{b/m} \leq \delta'$.
- PAC learnability of each sub-model without attack: The sub-model that has $m/6t$ examples have enough examples for PAC learning. $m/6t = \sqrt{m/b}/24 \geq m^2_{\mathsf{Lrn}}(\varepsilon/12, \varepsilon/12)$.

- Enough examples: With probability $1 - m/(2t \cdot 2^t)$, at least $t/4$ subsets have at least $m/6t$ examples. We have $m/(2t \cdot 2^t) < 1/2^{(\log_2(2/\delta))} = \delta/2$
- Most sub-models have low risk: By Hoeffding's inequality, with probability at least $1 - e^{-2t \cdot \delta'^2}$, it holds that the fraction of $h_1, \ldots, h_{t'}$ with risk at most ε' is at most $2\delta'$. When $m \geq m_{\mathsf{Rob}}(\varepsilon, \delta)$, we have $t' \geq t/4 \geq \sqrt{m/b}/4 \geq -\log(\delta)/\left(8\delta'^2\right)$

In summary, we show that with probability at least $1 - \delta/2$, we have at least $t/4 = \sqrt{m \cdot b(m)}$ subsets, each subset has at least $m_{\mathsf{Lrn}}(\varepsilon/12, \varepsilon/12)$ examples, and we train an majority ensemble model on it. We then follow the same analysis from the case that the attacks can not reorder the examples. Therefore, with probability at least $1 - \delta/2$, RLrn is a $(\varepsilon, \delta/2)$-PAC learner under b-replacing attacks. By the union bound, RLrn is a $(\varepsilon, \delta/2)$-PAC learner under b-replacing attacks.

As a summary, ensemble learner RLrn achieves a bound similar to the sample complexity bound of the non-reordering attacks. When $m_{\mathsf{Rob}}(\varepsilon, \delta) = 576\lambda\,(p')$, the majority learner is robust to b-replacing attacks that can also reorder the examples.

Finally, when $m \geq 576\lambda\,(p')$, certifying model $h_{\mathrm{cert}}(h_{\mathsf{ens}}, x) = \sum_{i=1}^{t'} \mathbb{1}\{h_i(x) = y'\} - t'/2$ gets

$$\Pr_{\mathcal{S}}\left[\mathrm{CCor}_{\mathcal{R}ep_b}(\mathcal{S}, D) \geq 1 - \varepsilon\right] \geq 1 - \delta$$

over data set $\mathcal{S}$. Therefore, $\mathcal{H}$ is certifiably PAC learnable under $\mathcal{R}ep_b$ attack. □

Extension to $\mathcal{A}dd\mathcal{R}em_b$ attacks. The proofs of Theorems 3.4.3 and 3.4.4 extend to $\mathcal{A}dd\mathcal{R}em_b$ attacks as well when $b = o(m)$. This is because, at a high level, all we care about is that adversarial "changes" (whether they are addition or removal of examples) either do not hit the sub-sampled dataset (in Theorem 3.4.3) or hit few of the sub-samples (in Theorem 3.4.4).

We then show that limiting adversary's budget to $b(m) = o(m)$ is essentially necessary for obtaining positive results in the distribution-independent PAC learning setting, as some hypothesis classes with finite-VC dimension are not learnable under targeted poisoning attacks when $b(m) = \Omega(m)$ in a very strong sense: any PAC learner (without attack) would end up having essentially a risk arbitrary close to 1 under attack for any $b(m) = \Omega(m)$ budget given to a b-replacing adversary.

We use homogeneous halfspace classifiers, defined in Definition 3.4.6 below, as an example of hypothesis classes with finite VC dimension. Then in Theorem 3.4.7, we show that the hypothesis class of halfspaces are not distribution-independently robust learnable against $\Omega(m)$-label flipping instance-targeted attacks.

Definition 3.4.6 (Homogeneous halfspace classifiers)**.** A (homogeneous) halfspace classifier $h_\omega : \mathbb{R}^d \to \{0,1\}$ is defined as $h_\omega(x) = \text{Sign}(\omega \cdot x)$, where ω is a d-dimensional vector. We then call $\mathcal{H}_{\mathsf{half}}$ the class of halfspace classifiers $\mathcal{H}_{\mathsf{half}} = \{h_\omega(x) \colon \omega \in \mathbb{R}^d\}$. For simplicity, we may use ω to refer to both the model parameter and the classifier. ◇

Theorem 3.4.7 (Limits of distribution-independent learnability of halfspaces)**.** *Consider the halfspaces hypothesis set $\mathcal{H} = \mathcal{H}_{\mathsf{half}}$ and we aim to learn any distribution over the unit sphere using $\mathcal{H}$. Let the adversary class be b-replacing with $b(m) = \beta \cdot m$ for any (even very small) constant β. For any (even improper) learner* Lrn *one of the following two conditions holds. Either* Lrn *is* not *a PAC learner for the hypothesis class of half spaces (even without attacks) or there exists a distribution D such that* $\text{Risk}_{\mathcal{F}lip_b}(\mathcal{S}, D) \geq 1 - \sqrt{\sigma}$ *with probability* $1 - \sqrt{\sigma}$ *over the selection of $\mathcal{S}$ of sufficiently large $m \geq m_{\mathsf{Lrn}}(\beta \cdot \sigma/6, \sigma/2)$, where m_{Lrn} is the sample complexity of PAC learner* Lrn*.*

Proof of Theorem 3.4.7. To prove the theorem, we select a distribution D and an $\Omega(m)$-label flipping adversary, that for any PAC learner Lrn, the targeted poisoning risk is high. We first prove the theorem for the ERM rule, and then we discuss how it extends to any PAC learner.

Our scenario is in dimension $d = 3$ with dimensions X, Y, Z. Consider the following distribution D: For $e = (\alpha, c) \sim D$ where α is a point in the 3-dimensional space and c is a label in $\{+1, -1\}$, with probability $1/2$ we sample α uniformly from the unit circle with $z = 1$ (namely $x^2 + y^2 = 1, z = 1$) and we let label $c = +1$ of the sampled point α. In addition, with probability $1/2$ we sample α uniformly from the unit circle $x^2 + y^2 = 1, z = -1$ and let label $c = -1$. This distribution is realizable over the halfspaces hypothesis set, as halfspace $\omega = (0, 0, 1)$ has 0 risk on D. In the following analysis, we call an arc of one of the circles as an interval $\mathcal{I}$. We then define the measure of the interval $\mathcal{I}$ as the probability that a random example $\beta \sim D$ that falls into the interval. Clearly in our setting, an interval $\mathcal{I}$ can be uniquely determined by fixing its measure β and its center point α'. This scenario is shown in Figure 3.1.

Now, assume the adversarial perturbation $\mathcal{S}' = \mathcal{F}lip_b(\mathcal{S})$ (that depends on $e = (\alpha, c)$) wants to fool the learner on the point $\alpha = (x, y, z)$. We now define the adversary $\mathsf{A}_b(\mathcal{S})$, that with a data set $\mathcal{S} \sim D^m$ and target point α, the adversary operates as the following.

- Pick an interval $\mathcal{I}$ of constant measure $\beta/3$ which is centered at $\alpha \in \mathcal{I}$ *in the same circle where α belongs.*
- To make the attack realizable, pick another corresponding interval, where $\mathcal{I}' = \{\alpha' \mid -\alpha' \in \mathcal{I}\}$.
- For all $(\alpha_i, c_i) \in \mathcal{S}$, flip the label if $\alpha_i \in (\mathcal{I} \cup \mathcal{I}')$. Return the new set as $\mathcal{S}'$.

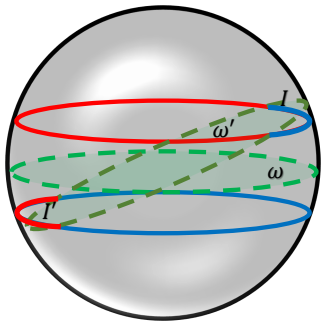


Figure 3.1: Example for proving Theorem 3.4.7. The red circle has label 1, and the blue circle has label −1. ω is the ground-truth halfspace with 0 risk, and ω' is the halfspace that has 0 risk after adversary make replacements.

In total $\mathcal{I}$ and $\mathcal{I}'$ has probability measure $2\beta/3$. Each example in $\mathcal{S}$ has probability $2\beta/3$ to fall into $\mathcal{I} \cup \mathcal{I}'$. Then by the Hoeffding's inequality,

$$\Pr\left[|\mathcal{S} \cap \mathcal{S}'| \leq (1-\beta)\cdot m\right] \geq 1 - e^{-\frac{m}{18}}.$$

That is, with high probability $\mathsf{A}_b(\mathcal{S})$ will modify less than $b(m) = \beta \cdot m$ examples. We then analyze how this adversary fools the learners.

ERM learner. We start from the case that the learner is the ERM learner. As $\mathcal{I}$ and $\mathcal{I}'$ are symmetric to the origin $(0,0,0)$, there exists a halfspace $\omega' \in \mathcal{H}_{\mathsf{half}}$ that passes all the endpoints of arcs $\mathcal{I}$ and $\mathcal{I}'$, which then has 0 empirical risk on $\mathcal{S}'$. With probability at least $1 - 2(1-\beta/6)^m \approx 1$, $\mathcal{S}$ contains two examples from $\mathcal{I}$ that positioned at either side around α, that ω (and all other hypothesis that correctly predicts α) will have non-zero risk on $\mathcal{S}'$. Therefore, ERM will return a hypothesis that incorrectly predicts α.

Extension to any proper PAC learner. We now prove that the same adversary can fool any proper PAC learner with sufficiently large m. Let D' be the "poisoned" distribution, that is, for $(\alpha_1, c_1) \in \mathsf{supp}(D')$ and $(\alpha_1, c_2) \in \mathsf{supp}(D)$. $c_1 = \begin{cases} -c_2 & \alpha' \in \mathcal{I} \cup \mathcal{I}' \\ c_2 & \text{Otherwise} \end{cases}$. Then for $\mathcal{S}' = \mathsf{A}_b(\mathcal{S})$, when $\mathcal{S} \sim D^m$, $\mathcal{S}' \sim (D')^m$.

Now, let $m_{\mathsf{Lrn}}(\varepsilon_1, \delta_1)$ be the sample complexity of $\mathcal{H}_{\mathsf{half}}$ on D'. When $m \geq m_{\mathsf{Lrn}}(\varepsilon_1, \delta_1)$, on the distribution D', $\mathsf{Lrn}(\mathcal{S}')$ holds $\mathrm{Risk}(\mathsf{Lrn}(\mathcal{S}'), D') \leq \varepsilon_1$ with probability at least $1 - \delta_1$.

Let $\varepsilon_1 = \beta/4$. Because hypothesis set $\mathcal{H}$ are halfspaces, the prediction region (the subset of all the examples predicted for a specific label) is also a connected interval. Therefore, if $\mathsf{Lrn}(\mathcal{S}')$ incorrectly

predicts α on $\mathcal{S}'$ (which is, correctly predicts α on the original data set $\mathcal{S}$), as α is at the center of $\mathcal{I}$, at least half of $\mathcal{I}$ (and $\mathcal{I}'$ because of symmetry) is incorrectly predicted, i.e., $\text{Risk}(\mathsf{Lrn}(\mathcal{S}'), D') \geq \beta/3$. This contradicts $\text{Risk}(\mathsf{Lrn}(\mathcal{S}'), D') \leq \varepsilon_1 = \beta/4$. Therefore, for the selected values of ε_1 and δ_1, with a sufficiently large sample complexity $m \geq m_{\mathsf{Lrn}}(\beta/4, \delta_1)$, the probability of α being misclassified becomes at least $1 - \delta_1$, which indicates the adversary succeeds with probability at least $1 - \delta_1$. By averaging, with probability at least $1 - \sqrt{\delta_1}$, we have $\text{Risk}_{\mathcal{F}lip_b}(\mathcal{S}, D) \geq 1 - \sqrt{\delta_1}$.

Extension to any improper PAC learner. Previous method cannot be directly applied to improper PAC learners as we no longer have at least half of $\mathcal{I}$ is incorrectly predicted if α is incorrectly predicted. We now slightly revise $\mathsf{A}_b(\mathcal{S})$ to fool improper PAC learners as well.

To fool an arbitrary improper PAC learner, the adversary will *randomize* the interval $\mathcal{I}$. The revised adversary $\mathsf{A}'_b(\mathcal{S}, \alpha)$ works as the following.

- Compute the interval $\mathcal{I}_0$ which is centered at α with measure $\beta/3$.
- Uniformly pick a random point α_r from $\mathcal{I}_0$.
- Pick the intervals $\mathcal{I}$ symmetrically around α_r with measure $\beta/3$, and let $\mathcal{I}' = \{\beta | -\beta \in \mathcal{I}\}$.

We have $\mathcal{S}' = \mathsf{A}'_b(\mathcal{S})$ where $\mathcal{S} \sim D^m$. Now, let $D'_{\mathcal{I}}$ be the data distribution where the labels of the examples in $\mathcal{I}$ and $\mathcal{I}'$ are flipped, we have $\mathcal{S}' \sim {D'_{\mathcal{I}}}^m$ as one can view the poisoned data set $\mathcal{S}'$ as an i.i.d. sample from the poisoned distribution $D'_{\mathcal{I}}$, which is conditioned on $\mathcal{I}$ and $\mathcal{I}'$. $\mathcal{I}$ and $\mathcal{I}'$, on the other hand, is conditioned on the poisoning target α.

Now, consider a different process that generates the variables in a different order, that the adversary first uniformly picks a interval $\mathcal{I}$ among all the interval with measure $\beta/3$ (and its counterpart $\mathcal{I}'$), and then uniformly samples an example α inside $\mathcal{I}$ and $\mathcal{I}'$. Because the sampling is uniform, the probability of picking a specific combination of $\mathcal{I}$, $\mathcal{I}'$ and α in the second process is equivalent to the probability of picking this combination following the original process, i.e., pick a random α, and then pick $\mathcal{I}$ conditioned on α. Because this equivalence, if α is picked *after* the learner returns a model learned from the data set $\mathcal{S}'$ (since it is sampled from $D'_{\mathcal{I}'}$), the probability of whether $\mathsf{Lrn}(\mathcal{S}')$ incorrectly predicts α remains the same.

We now prove that when m is sufficiently large, attacks succeed with high probability on improper PAC learners. Let $m_{\mathsf{Lrn}}(\varepsilon_1, \delta_1)$ be the sample complexity of $\mathcal{H}_{\mathsf{half}}$ on $D'_{\mathcal{I}}$. When $m \geq m_{\mathsf{Lrn}}(\varepsilon_1, \delta_1)$, on the distribution D', $\mathsf{Lrn}(\mathcal{S}')$ holds $\text{Risk}(\mathsf{Lrn}(\mathcal{S}'), D'_{\mathcal{I}}) \leq \varepsilon_1$ with probability at least $1 - \delta_1$. Since we can equivalently assume α is sampled after $\mathsf{Lrn}(\mathcal{S}')$ is done, the probability of $\mathsf{Lrn}(\mathcal{S}')$ correctly

predicts α on $D'_{\mathcal{I}}$ (which is, incorrectly predicts α on D) is at least $1-\epsilon_1/(\beta/3)$. Let $\epsilon_1 = \sigma \cdot \beta/6$ and $\delta_1 = \sigma/2$.

Therefore, for the selected values of ε_1 and δ_1, with $m \geq m_{\mathsf{Lrn}}(\varepsilon_1, \delta_1)$, the probability of α being misclassified becomes at least $1-\epsilon_1/(\beta/3)-\delta_1 = 1-\sigma/2-\sigma/2 = 1-\sigma$. By averaging, with probability at least $1-\sqrt{\sigma}$, we have $\mathrm{Risk}_{\mathcal{F}lip_b}(\mathcal{S}, D) \geq 1-\sqrt{\sigma}$. □

Remark 3.4.8 (On (ε, δ)-PAC learning with $\varepsilon = \Omega(1)$)**.** Theorem 3.4.7 shows that if adversary's budget scales linearly with the sample complexity m, then one cannot get (ε, δ) PAC learners that are robust against instance-targeted poisoning attacks and that $\varepsilon, \delta = o_m(1)$. However, one can also ask what is the minimum achievable error $\varepsilon(m)$, perhaps as a function of adversary's budget $b(m)$, even when $b(m) = \Omega(m)$. For example, what would be the optimal learning error, if adversary corrupts 1% of the examples. The same proof of Theorem 3.4.7 shows that in this case, any learner that is robust to instance-targeted $\mathcal{R}ep_b$ attacks would need to have $\varepsilon(m) = \Omega(b(m)/m)$. The reason is that if $\varepsilon(m) = o(b(m)/m)$, then one can still choose $\sigma_m = o_m(1)$, while $\varepsilon(m) = (b(m)/m) \cdot \sigma(m)/6, \delta(m) = \sigma(m)/2$ are both $o_m(1)$ as well.

Note that it was already proved by [18] that, if the adversary can corrupt $b = \Omega(m)$ of the examples, even with *non-targeted* adversary, robust PAC learning is impossible. However, in that case, there is a learning algorithm with error $O(b/m)$. So if, e.g., $b = m/1000$, then non-targeted learning is possible for practical purposes. On the other hand, Theorem 3.4.7 shows that any PAC learning algorithm in the *no attack* setting, would have essentially risk 1 under *targeted* poisoning.

Remark 3.4.9 (Other loss functions)**.** Most of our initial results in this chapter are proved for the 0-1 loss as the default for classification. Yet, the written proof of Theorem 3.4.3 holds for any loss function. Theorem 3.4.4 can also likely be extended to other "natural" losses, but using a more complicated "decision combiner" than the majority. In particular, the learner can now output a label for which "most" sub-models will have "small" risk (parameters most/small shall be chosen carefully). The existence of such a label can probably be proved by a similar argument to the written proof of the 0-1 loss. However, this operation is not poly time.

3.4.2 Distribution-specific learning

Our previous results are for distribution-independent learning. This still leaves open to study distribution-specific learning. That is, when the input distribution is fixed, one might able to prove stronger results.

We then study the learnability of halfspaces under instance-targeted poisoning on *the uniform distribution over the unit sphere*. Note that one can map all the examples in the d-dimensional space to the surface of the unit sphere, and their relative position to a homogeneous halfspace remains the same. Hence, one can limit both ω and instance $x \in \mathbb{R}^d \setminus 0^d$ to be unit vectors in $\mathbb{S}^{d-1}$. Therefore, distributions D_X on the unit sphere surface can represent any distribution in the d-dimensional space. For example, a d-dimensional isotropic Gaussian distribution can be equivalently mapped to the uniform distribution over the unit sphere as far as classification with homogeneous halfspaces is concerned. We note that when the attack is *non-targeted*, it was already shown by [18] that whenever $b(m) = o(m)$, then robust PAC learning is possible (if it is possible in the no-attack setting). Therefore, our results below can be seen as extending the results of [18] to the *instance-targeted* poisoning attacks.

Theorem 3.4.10 (Learnability of halfspaces under the uniform distribution)**.** *In the realizable setting, let D be uniform on the d dimensional unit sphere $\mathbb{S}^{d-1}$ and let adversary's budget for $\mathcal{R}ep_{b(m)}$ be $b(m) = cm/\sqrt{d}$. Then for the halfspace hypothesis set $\mathcal{H}_{\mathsf{half}}$, there exists a deterministic proper certifying learner* CLrn *such that the following*

$$\Pr_{S\sim D^m}\left[\mathrm{CCor}_{\mathcal{R}ep_{b(m)}}(S, D) \geq 1 - 2\sqrt{2\pi}\cdot c - \sqrt{2\pi d}\cdot\varepsilon\right]$$

is at least $1-\delta$ for sufficiently large sample complexity $m \geq m^{\mathcal{H}}_{\mathsf{UC}}(\varepsilon,\delta)$, where $m^{\mathcal{H}}_{\mathsf{UC}}$ is the sample complexity of uniform convergence on $\mathcal{H}_{\mathsf{half}}$. So the problem is properly and certifiably PAC learnable under b-replacing instance-targeted poisoning attacks.

For example, when $c = 1/502$, $\varepsilon = c/(100\sqrt{d})$ and $\delta = 0.01$, Theorem 3.4.10 implies that

$$\Pr_{S\sim D^m}\left[\mathrm{CCor}_{\mathcal{R}ep_{b(m)}}(S, D) \geq 99\%\right] \geq 99\%.$$

Proof of Theorem 3.4.10. Without loss of generality, we assume $\omega = (1, 0, 0\ldots, 0) \in \mathcal{H}_{\mathsf{half}}$ denotes the ground-truth halfspace, i.e., $\mathsf{Risk}(\omega, D) = 0$. Therefore, for any data set that is i.i.d. sampled $S \sim D^m$, $\mathsf{Risk}(\omega, S) = 0$. We denote $\beta(m) = b(m)/m = c/\sqrt{d}$ be the fraction of replaced examples in the data set, and for simplicity we may use b and β to represent $b(m)$ and $\beta(m)$ in the following analysis.

We now show that hypothesis class $\mathcal{H}_{\mathsf{half}}$ is properly and certifiably PAC learnable under instance-targeted poisoning attacks on D. The general idea is to prove that for the majority of examples $e = (x, y) \sim D$, the risk of any hypothesis that incorrectly predicts x is large. Let $\mathsf{A}_b(S)$ be an

arbitrary adversary of budget $b(m)$. Since the adversary needs to fool the ERM algorithm, the adversary needs to change the data set from $\mathcal{S}$ to $\mathcal{S}'$, so that the empirical risk of a "bad" hypothesis ω', $\text{Risk}(\omega', \mathcal{S}')$, is lower than the empirical risk of ω, $\text{Risk}(\omega, \mathcal{S}')$. However, since the adversary can only make b changes, we have

$$\text{Risk}(\omega, \mathcal{S}') \leq \text{Risk}(\omega, \mathcal{S}) + \beta = \beta, \quad \text{and} \quad \text{Risk}(\omega', \mathcal{S}') \geq \text{Risk}(\omega', \mathcal{S}) - \beta.$$

Also, according to the uniform convergence property of the hypothesis set, let $m_{\mathsf{UC}}^{\mathcal{H}}(\varepsilon, \delta)$ be the sample complexity of uniform convergence. Then with probability at least $1 - \delta$ over $\mathcal{S}$, we have $\text{Risk}(\omega', D) \leq \text{Risk}(\omega', \mathcal{S}) + \varepsilon$. Therefore, to fool ERM on x with budget b, the adversary needs

$$\exists \omega' \in \mathcal{H}_{\mathsf{half}} \text{ such that } \text{Risk}(\omega', D) \leq 2\beta + \varepsilon \text{ and } \omega'(x) \neq \omega(x). \tag{3.2}$$

We then show that when $m \geq m_{\mathsf{UC}}^{\mathcal{H}}(\varepsilon, \delta)$, for the majority of instances according to D, no such ω' exists if B is sufficiently small.

The intersection of the halfspace ω and the d- dimensional sphere $\mathbb{S}^{d-1}$, i.e., the "equator", is a $(d-1)$-dimensional sphere. Suppose $x = (x_1, x_2, \ldots, x_d)$, let θ be the angle between x, the origin, and the halfspace ω. There exists an unique x' on the equator that has the minimal distance to the x among all the points on the equator, and $\angle xox' = \theta$ where o stands for the origin $\{0, 0, \ldots, 0\}$. For any halfspace ω_1 where x' is on ω_1, the angle between ω and ω_1 is at least θ. Therefore, a halfspace where $\omega'(x) \neq \omega(x)$ has the property that the angle between ω' and ω is at least θ. In that case, since the the risk of ω' on D is at least $\mathsf{Risk}(\omega', D) \geq \theta/\pi$. In the following analysis, we call an example x' around angle θ' of a halfspace ω', if the angle between x', the origin and halfspace ω' is less than θ'.

As the distribution D is uniform, the probability of an example fall into angle θ around the halfspace ω can be calculated by measuring the size of the surface within angle θ, which is then upper bounded by the cylindrical surface size of a cylinder whose bottom is a $(d-1)$-dimensional unit ball and height is 2θ. Let S_{d-1} denotes the surface of the $(d-1)$-dimensional unit sphere, then this cylinder surface has the size of $2\theta S_{d-1}$. We further denote the surface of a d-dimensional ball as S_d. Therefore, the probability of a random example falls into the set within angle θ around ω can be upper bounded by

$$\Pr_{(x,y)\sim D}[x \text{ is within angle } \theta \text{ around } \omega] < \frac{2\theta S_{d-1}}{S_d} < \frac{\theta\sqrt{2d}}{\sqrt{\pi}}.$$

The last inequality follow from the property of spheres. Now, let $\theta_0 = (2\beta + \varepsilon)\pi = 2\pi c/\sqrt{d} + \pi\varepsilon$,

then $\theta_0\sqrt{2d}/\sqrt{\pi} = 2\sqrt{2\pi}\cdot c + \sqrt{2\pi d}\cdot\varepsilon$. Therefore, we have for at least $1-(2\sqrt{2\pi}\cdot c+\sqrt{2\pi d}\cdot\varepsilon)$ of all possible x, all halfspace ω' that $\omega'(x)\neq\omega(x)$ has $\mathrm{Risk}(\omega', D) > 2\beta+\varepsilon$, which according to Equation 3.2, indicates that the adversary needs budget more than b to change the prediction of x.

Finally, we define a certifying model h_{cert} that returns certifications $\geq b$ with high probability. For input $e=(x,y)$ and $\mathcal{S}$, suppose $\omega' = \mathsf{Lrn}(\mathcal{S})$, let θ' be the angle between x and ω', then

$$h_{\mathrm{cert}}(x) = \begin{cases} \max\left\{0, \left(\frac{\theta'}{2\pi} - \frac{\varepsilon}{2}\right)\cdot m\right\} & \frac{\theta'}{\pi} \geq 2\beta+\varepsilon \\ 0 & \text{Otherwise} \end{cases}.$$

Following our analysis, we have $h_{\mathrm{cert}}(x) > b$ for all the examples that are not within angle θ' of ω, which is with high probability. Also, for any x that $\theta'/\pi \geq 2\beta+\varepsilon$, we have $\forall\omega'(x)\neq\omega(x)$, $\mathrm{Risk}(\omega', D)\geq\theta'/\pi$. To flip the prediction on x, the adversary need to replace at least

$$\beta' \geq \frac{\min_{\omega'\in\mathcal{H}}\{\mathrm{Risk}(\omega',\mathcal{S})\}}{2} \geq \frac{\min_{\omega'\in\mathcal{H}}\{\mathrm{Risk}(\omega', D)-\varepsilon\}}{2} \geq \frac{\theta/\pi-\varepsilon}{2} = \frac{\theta'}{2\pi}-\frac{\varepsilon}{2}$$

fractions of any $\mathcal{S}$ that is ε-representative. Therefore, h_{cert} gives a correct certification for all examples for any $\mathcal{S}$ that is ε-representative, and the certification result is larger than b for the majority of examples for any such $\mathcal{S}$.

In summary, when $b = cm/\sqrt{d}$ and $m \geq m^{\mathcal{H}}_{\mathsf{UC}}(\varepsilon,\delta)$, with probability $1-\delta$, there are at least $1-2\sqrt{2\pi}\cdot c-\sqrt{2\pi d}\cdot\varepsilon$ of examples that are robust to any b-replacing instance-targeted poisoning attacks. Therefore, the certifying learner $\mathsf{CLrn}(\mathcal{S})(x) = (\mathsf{Lrn}(\mathcal{S})(x), h_{\mathrm{cert}}(x))$ gets

$$\Pr_{\mathcal{S}\sim D^m}\left[\mathrm{CCor}_{\mathcal{R}ep_{b(m)}}(\mathcal{S}, D) \geq 1-2\sqrt{2\pi}\cdot c-\sqrt{2\pi d}\cdot\varepsilon\right] \geq 1-\delta.$$

Therefore, $\mathcal{H}$ is certifiably and properly PAC learnable under $\mathcal{R}ep_b$ attacks. □

We also show that the above theorem is essentially optimal, as long as we use proper learning. Namely, for any fixed dimension d, with budget $b = O(m/\sqrt{d})$, a b-replacing adversary can guarantee success of fooling the majority of examples. Note that for constant d, when $m\to\infty$, this is just a constant fraction of data being poisoned, yet this constant fraction can be made arbitrary small when $d\to\infty$.

Theorem 3.4.11 (Limits of robustness of PAC learners under the uniform distribution)**.** *In the realizable setting, let D be uniform over the d dimensional unit sphere $\mathbb{S}^{d-1}$. For the halfspace hypothesis set $\mathcal{H}_{\mathsf{half}}$, if $b(m)\geq cm/\sqrt{d}$ for b-label flipping attacks $\mathcal{F}lip_b$, for any proper learner* Lrn

one of the following two conditions holds. Either Lrn *is* not *a PAC learner for the hypothesis class of half spaces (even without attacks), or for sufficiently large* $m \geq m_{\mathsf{Lrn}}(3c/(10\sqrt{d}), \delta)$*, with probability* $1 - \sqrt{\delta + 2e^{-c^2/18}}$ *over the selection of* $\mathcal{S}$ *we have*

$$\mathrm{Risk}_{\mathcal{F}lip_b}(\mathcal{S}, D) \geq 1 - \sqrt{\delta + 2e^{-c^2/18}},$$

where m_{Lrn} *is the sample complexity of the learner* Lrn*.*

For example, when $c = 20$ and $\delta = 0.00009$, we have $\mathrm{Risk}_{\mathcal{F}lip_b}(\mathcal{S}, D) \geq 99\%$.

Proof of Theorem 3.4.11. Let $\omega \in \mathcal{H}_{\mathsf{half}}$ denote the ground-truth halfspace, i.e., $\mathsf{Risk}(\omega, D) = 0$. We now design an adversary that fools the learner Lrn within the budget $b(m)$. We start by proving the theorem for the ERM rule, and then we discuss how it extends to any PAC learner.

According to the concentration of the uniform measure over the unit sphere $\mathbb{S}^{d-1}$ (e.g., see [86]), for any set of measure 0.5 on the sphere, its ρ-neighborhood T_ρ (defined as the set of all the points whose Euclidean distance less or equal to ρ) has measure

$$\mu(T_\rho) \geq 1 - 2e^{-d\rho^2/2}.$$

Therefore, for any halfspace ω, the measure of samples that has ρ distance to ω is at least $1 - 4e^{-d\rho^2/2}$.

Now, given an example x and the training data set $\mathcal{S}$, suppose θ is the angle between x and ω, the adversary $\mathsf{A}_b \in \mathcal{F}lip_b$ act like this:

1. Rotate ω to x by θ. Let ω' denotes the result halfspace (where x landed on).
2. Rotate ω' with another θ in the same direction to the halfspace ω''.
3. For any example from the data set $\mathcal{S}$ that is between ω and ω'', flip its label.
4. Return the data set as $\mathcal{S}'$.

Let $\rho_0 = c/3\sqrt{d}$, then at least $1 - 2e^{-c^2/18}$ of x has at most ρ_0 distance to ω. The probability measure of the surface between ω and ω'' is $2\theta/\pi$, where $2\theta/\pi \leq 2\sin(\theta) \leq 2\rho_0$. Let $m^{\mathcal{H}}_{\mathsf{UC}}(\varepsilon, \delta)$ be the sample complexity of uniform convergence. Then with probability at least $1 - \delta$ over $\mathcal{S}$, we have $\mathrm{Risk}(\omega'', \mathcal{S}) \leq \mathrm{Risk}(\omega'', \mathcal{D}) + \varepsilon \leq 2\rho_0 + \varepsilon$.

Let $\varepsilon = 0.9\rho_0$, then the adversary flips $\mathrm{Risk}(\omega'', \mathcal{S}) \cdot m$ examples, which with probability $1 - \delta$ we have $\mathrm{Risk}(\omega'', \mathcal{S}) \leq 2.9\rho_0 < b/m$. Now, the ERM learner will go for the hypothesis with the

minimal error on $\mathcal{S}'$, which is then ω''. As $\omega''(x) \neq \omega(x)$, the ERM learner will give a wrong answer on x. With probability $1-\delta$, the adversary will complete the attack within budget b on at least $1-2e^{-c^2/18}$ examples, by the union bound, the adversary succeeds on $1-\delta-2e^{-c^2/18}$ examples. Finally, by an averaging argument, we have with probability $1-\sqrt{\delta+2e^{-c^2/18}}$, the adversary succeeds with $1-\sqrt{\delta+2e^{-c^2/18}}$ examples.

Extension to any proper PAC learner To extend the result to any proper PAC learner, we use a similar proof as in Theorem 3.4.7. We show same A_b can be extended to fool any proper PAC learner with high probability.

Let D' be the "poisoned" distribution, that for $\mathcal{S}' = \mathsf{A}_b(\mathcal{S})$, we have $\mathcal{S}' \sim (D')^m$. Then with probability $1-\delta$, we have $\mathrm{Risk}(\mathsf{Lrn}(\mathcal{S}'), \mathcal{D}') \geq \mathrm{Risk}(\mathsf{Lrn}(\mathcal{S}'), \mathcal{S}') - \varepsilon$. Now, let $m_{\mathsf{Lrn}}(\varepsilon_1, \delta_1)$ be the sample complexity of Lrn on D'. When $m \geq m_{\mathsf{Lrn}}(\varepsilon_1, \delta_1)$, on the distribution D', $\mathsf{Lrn}(\mathcal{S}')$ holds $\mathrm{Risk}(\mathsf{Lrn}(\mathcal{S}'), D') \leq \varepsilon_1$ with probability at least $1-\delta_1$.

Let $\varepsilon_1 = 0.9\rho_0 = 3c/10\sqrt{d}$. Because hypothesis set $\mathcal{H}$ are halfspaces, the prediction region (the subset of all the examples predicted for a specific label) is connected. Therefore, if $\mathsf{Lrn}(\mathcal{S}')$ incorrectly predicts x (which is, correctly predicts x on the original data set $\mathcal{S}$), as x is on ω', at least half of the surface between ω and ω'' is incorrectly predicted, i.e., $\mathrm{Risk}(\mathsf{Lrn}(\mathcal{S}'), D') \geq \rho_0$. This contradicts $\mathrm{Risk}(\mathsf{Lrn}(\mathcal{S}'), D') \leq \varepsilon_1 = 0.9\rho_0$. Therefore, with probability $1-\delta_1$, the adversary will complete the attack within budget b on at least $1-2e^{-c^2/18}$ examples, by the union bound, the adversary succeeds on $1-\delta_1-2e^{-c^2/18}$ examples. Finally, by an averaging argument, we have with probability $1-\sqrt{\delta_1+2e^{-c^2/18}}$, the adversary succeeds with $1-\sqrt{\delta_1+2e^{-c^2/18}}$ examples.

□

3.4.3 Relating risk and robustness

Risk uses a worst-case budget to capture what an adversary can do, while robustness does so using an average-case budget. Theorem 3.4.12 below relates the two notions of risk and robustness in the context of targeted poisoning attacks and is inspired by results previously proved for adversarial inputs that are crafted during test-time attacks ([87, 70]). In particular, Theorem 3.4.12 proves that for 0-1 loss, it is equivalent to fully understand either of them to understand the other one and allows to derive numerical values for one through the other.

Theorem 3.4.12 (From risk to robustness and back)**.** *Suppose $\mathcal{S} \in (\mathcal{X} \times \mathcal{Y})^m$ is a training set, Lrn is a learner, D is a distribution over $\mathcal{X} \times \mathcal{Y}$, $\mathcal{A}_b$ is an adversary class with the budget b, and $\mathcal{A} = \cup_{b \in \mathbb{N}} \mathcal{A}_b$. Then the following relations hold.*

1. **From robustness to risk.** *For any non-negative loss function, we have*

$$\mathrm{Risk}_{\mathcal{A}_b}(\mathcal{S}, r, D) = \int_0^\infty \Pr_{e\sim D}\left[\mathrm{Rob}^\tau_{\mathcal{A}}(\mathcal{S}, r, e) \le b\right] \cdot \mathrm{d}\tau.$$

For the special case of 0-1 loss, this simplifies to $\mathrm{Risk}_{\mathcal{A}_b}(\mathcal{S}, r, D) = \Pr_{e\sim D}\left[\mathrm{Rob}_{\mathcal{A}}(\mathcal{S}, r, e) \le b\right]$.

2. **From risk to robustness.** *Suppose we use the 0-1 loss. Suppose* b *is large enough such that* $\mathrm{Risk}_{\mathcal{A}_n}(\mathcal{S}, r, D) = 1$*, or equivalently* $\mathrm{Cor}_{\mathcal{A}_i}(\mathcal{S}, r, D) = 0$ *for* $i \ge b$.[9] *Then, it holds that*

$$\begin{aligned}\mathrm{Rob}_{\mathcal{A}}(\mathcal{S}, r, D) &= b - \sum_{i=0}^{b-1} \mathrm{Risk}_{\mathcal{A}_i}(\mathcal{S}, r, D)\\ &= \sum_{i=0}^{b-1} \mathrm{Cor}_{\mathcal{A}_i}(\mathcal{S}, r, D)\\ &= \sum_{i=0}^{\infty} \mathrm{Cor}_{\mathcal{A}_i}(\mathcal{S}, r, D).\end{aligned}$$

In other words, if we could compute adversarial risks for all b*, we can also compute the average robustness by summing robust correctness.*

Proof of Theorem 3.4.12. We write the proof for deterministic learners who do not have any randomness, but the same exact proof works when a randomness r exists and is fixed.

By Definition 3.3.2, for any threshold τ we have

$$\begin{aligned}\mathrm{Rob}^\tau_{\mathcal{A}}(\mathcal{S}, e) \le b &\iff \sup_{\mathcal{S}'\in\mathcal{A}_b(\mathcal{S})} \{\ell(\mathsf{Lrn}(\mathcal{S}'), e)\} \ge \tau\\ &\iff \exists \mathcal{S}' \in \mathcal{A}_b(\mathcal{S}), \ell(\mathsf{Lrn}(\mathcal{S}')(x), y) \ge \tau.\end{aligned}$$

Also, the so-called expectation through CDF[10] implies that for a non-negative function f and a distribution D, we have

$$\mathop{\mathbb{E}}_{x\sim D}[f(x)] = \int_{\tau=0}^\infty \Pr[f(x) \ge \tau]\,\mathrm{d}\tau \tag{3.3}$$

[9] For example, if the adversarial strategy allows flipping up to b labels, then for $b = m$ the adversary can flip all the labels. For natural hypothesis classes and learning algorithms, changing all the labels allows the adversary to control prediction on all points and so $\mathrm{Risk}_{\mathcal{A}_b}(\mathcal{S}, D) = 1$.

[10] See `https://en.wikipedia.org/w/index.php?title=Expected_value&oldid=1017448479\#Basic_properties` as accessed on May 16, 2021.

Therefore, Part 1 can be proven as follows.

$$\begin{aligned}
\mathrm{Risk}_{\mathcal{A}_b}(\mathcal{S}, D) &= \mathop{\mathbb{E}}_{e\sim D}\left[\ell_{\mathcal{A}_b}(\mathcal{S}, e)\right] \\
\text{(by Definition 3.3.1)} &= \mathop{\mathbb{E}}_{e\sim D}\left[\sup_{\mathcal{S}'\in\mathcal{A}_b(\mathcal{S})}\{\ell(\mathsf{Lrn}(\mathcal{S}'), e)\}\right] \\
\text{(by Equation 3.3)} &= \int_{\tau=0}^{\infty}\Pr_{e\sim D}\left[\sup_{\mathcal{S}'\in\mathcal{A}_b(\mathcal{S})}\{\ell(\mathsf{Lrn}(\mathcal{S}'), e)\}\geq\tau\right]\cdot \mathrm{d}\tau \\
\text{(by Definition 3.3.2)} &= \int_{\tau=0}^{\infty}\Pr_{e\sim D}[\mathrm{Rob}^{\tau}_{\mathcal{A}}(\mathcal{S}, e)\leq b]\cdot \mathrm{d}\tau.
\end{aligned}$$

We now prove Part 2. From Definition 3.3.2, $\mathrm{Rob}_{\mathcal{A}}(\mathcal{S}, e)\in\mathbb{N}\cup\{\infty\}$. We then have

$$\forall i\in\mathbb{R}, \Pr\left[\mathrm{Rob}_{\mathcal{A}}(\mathcal{S}, e)\geq i\right] = \Pr\left[\mathrm{Rob}_{\mathcal{A}}(\mathcal{S}, e)\geq \lceil i\rceil\right], \tag{3.4}$$

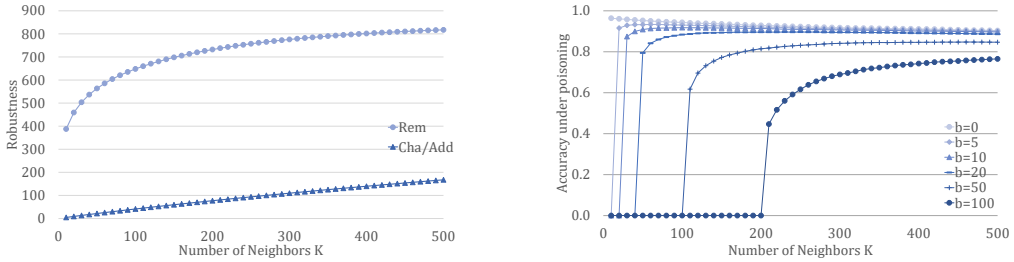
where $\lceil i\rceil$ is the ceiling function that returns the minimum integer above i. Furthermore, recall that b is a large enough number that for any example e, $\forall i\geq b, \mathrm{Risk}_{\mathcal{A}_i}(\mathcal{S}, e) = 1$ and $\mathrm{Cor}_{\mathcal{A}_i}(\mathcal{S}, e) = 0$. We have $\forall e, \Pr\left[\mathrm{Rob}_{\mathcal{A}}(\mathcal{S}, e)\leq b\right] = 1$, i.e., $\mathrm{Rob}_{\mathcal{A}}(\mathcal{S}, e)\leq b$. Then we conclude that,

$$\begin{aligned}
\mathrm{Rob}_{\mathcal{A}}(\mathcal{S}, D) &= \mathop{\mathbb{E}}_{e\sim D}\left[\mathrm{Rob}_{\mathcal{A}}(\mathcal{S}, e)\right] \\
\text{(by Equation 3.3)} &= \int_{\tau=0}^{\infty}\Pr_{e\sim D}\left[\mathrm{Rob}_{\mathcal{A}}(\mathcal{S}, e)\geq\tau\right]\cdot\mathrm{d}\tau \\
\text{(by Equation 3.4)} &= \sum_{i=0}^{\infty}\Pr_{e\sim D}\left[\mathrm{Rob}_{\mathcal{A}}(\mathcal{S}, e) > i\right] \\
&= b - \sum_{i=0}^{b-1}\Pr_{e\sim D}\left[\mathrm{Rob}_{\mathcal{A}}(\mathcal{S}, e)\leq i\right] \\
\text{(by Definition 3.3.2)} &= b - \sum_{i=0}^{b-1}\mathrm{Risk}_{\mathcal{A}_i}(\mathcal{S}, D) \\
\text{(by Definition 3.3.1)} &= \sum_{i=0}^{b-1}\mathrm{Cor}_{\mathcal{A}_i}(\mathcal{S}, D) = \sum_{i=0}^{\infty}\mathrm{Cor}_{\mathcal{A}_i}(\mathcal{S}, D).
\end{aligned}$$

□

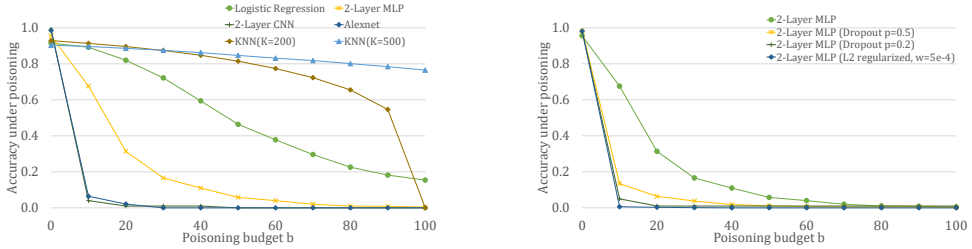
3.5 Experiments

In this section, we study the power of instance-targeted poisoning on the MNIST dataset [88]. We first analyze the robustness of K-Nearest Neighbor model, where the robustness can be efficiently calculated empirically. We then empirically study the accuracy under targeted poisoning for multiple other different learners. Previous empirical analysis on instance-targeted poisoning (e.g., Shafahi et al. [73]) mostly focus on clean-label attacks. Here, we use attacks of any labels, which lead to



(a) (b)

Figure 3.2: Experiment of K-Nearest Neighbors on the MNIST dataset. (a) The trend of Robustness $\text{Rob}(\mathsf{Lrn}_{\mathsf{knn}}, \mathcal{S}_{\mathsf{MNIST}}, \mathcal{D})$ on attacks $\mathcal{R}ep$, $\mathcal{A}dd$, and $\mathcal{R}em$, with the increase of number of neighbors K. (b) Accuracy of K-NN model under $\mathcal{R}ep_b$ with different poisoning budget b.



(a) (b)

Figure 3.3: Accuracy of different learners under $\mathcal{A}dd_b$ instance-targeted poisoning on the MNIST dataset. (a) Compare different learners. (b) Compare dropout and regularization mechanics on Neural Networks.

stronger attacks compared to clean-label attacks. We also study multiple models in our experiment, while previous work mostly focus on neural networks, and we then compare the performance of different models under the same attack.

K-Nearest Neighbor (K-NN) is non-parameterized model that memorizes every training example in the dataset. This special structure of K-NN allows us to empirically evaluate the robustness to poisoning attacks. The K-NN model in this section uses the majority vote defined below.

Definition 3.5.1 (K-NN learner)**.** For training dataset $\mathcal{S}$ and example $e = (x, y)$, let $\mathcal{N}(x)$ denote the set of K closest examples from $\mathcal{S}$ e. Then the prediction of the K-NN is

$$h_{\mathsf{KNN}}(x) = \operatorname*{argmax}_{j \in \mathcal{Y}} \sum_{(x_i, y_i) \in \mathcal{N}(x)} \mathbb{1}[y_i = j].$$

◇

From our definition of poisoning attack and robustness, we can measure the robustness empirically by the following lemma. Similar ideas can also be found in [40].

Lemma 3.5.2 (Instance-targeted Poisoning Robustness of the K-NN learner)**.** *Let* $\mathsf{margin}(h_{\mathsf{KNN}}, e)$ *be defined as* 0 *if* $h_{\mathsf{KNN}}(x) \neq y$ *and be defined as*

$$\sum_{(x_i,y_i)\in\mathcal{N}(x)} \mathbb{1}[y_i = y] - \max_{j\in\mathcal{Y}, j\neq y} \sum_{(x_i,y_i)\in\mathcal{N}(x)} \mathbb{1}[y_i = j]$$

otherwise. We then have

$$\mathrm{Rob}_{\mathcal{R}ep_b}(\mathsf{Lrn}_{\mathsf{KNN}}, \mathcal{S}, e) = \left\lceil \frac{\mathsf{margin}(\mathsf{Lrn}_{\mathsf{KNN}}(\mathcal{S}), e)}{2} \right\rceil.$$

Proof of Lemma 3.5.2. Following Definition 3.5.1, the prediction for a sample x totally depends on the neighbor set $\mathcal{N}(x)$. By definition, $\mathcal{N}(x)$ is a subset of $\mathcal{S}$. For the adversary class $\mathcal{R}ep_b$ (which can be extend to any adversary with budget b), they can only make at most b changes to the set $\mathcal{S}$, which includes at most b changes to $\mathcal{N}(x)$.

For an example $e = (x, y)$, to flip the prediction to y', we need to change $\mathcal{N}(x)$ to $\mathcal{N}'(x)$ such that $\sum_{(x_i,y_i)\in\mathcal{N}'(x)} \mathbb{1}[y_i = y'] \geq \sum_{(x_i,y_i)\in\mathcal{N}'(x)} \mathbb{1}[y_i = y]$. However, we have $\forall y' \neq y$,

$$\sum_{(x_i,y_i)\in\mathcal{N}(x)} \mathbb{1}[y_i = y] - \sum_{(x_i,y_i)\in\mathcal{N}(x)} \mathbb{1}[y_i = y'] \geq \mathsf{margin}(h_{\mathsf{KNN}}, e).$$

At least $\left\lceil \frac{\mathsf{margin}(\mathsf{Lrn}_{\mathsf{KNN}}(\mathcal{S}),e)}{2} \right\rceil$ replacements needs to be made in this case. To make it work, the adversary can replace the label of $\left\lceil \frac{\mathsf{margin}(\mathsf{Lrn}_{\mathsf{KNN}}(\mathcal{S}),e)}{2} \right\rceil$ examples of label y in $N(x)$ with y'. Therefore, we have $\mathrm{Rob}_{\mathcal{R}ep_b}(\mathsf{Lrn}_{\mathsf{KNN}}, \mathcal{S}, e) = \left\lceil \frac{\mathsf{margin}(\mathsf{Lrn}_{\mathsf{KNN}}(\mathcal{S}),e)}{2} \right\rceil$. □

Using Lemma 3.5.2, one can compute the robustness of the K-NN model empirically by calculating the margin for every e in the distribution. We then use the popular digit classification dataset MNIST to measure the robustness.

In the experiment, we use the whole training dataset to train (60, 000 examples), and evaluate the robustness on the testing dataset (10, 000 examples). We calculate the robustness under $\mathcal{R}ep_b$, $\mathcal{R}em_b$, and $\mathcal{A}dd_b$ attacks. We measure the result with different number of neighbors K present the result in Figure 3.2a. We also measure the accuracy under poisoning of $\mathcal{R}ep_b$ and report it in Figure 3.2b. The results in Figure 3.2 indicates the following message. (1) From Figure 3.2a, when the number of neighbors K increases, the robustness also increases as expected. The robustness of

K-NN to $\mathcal{R}ep$ and $\mathcal{A}dd$ increases almost linearly with K. (2) The robustness to $\mathcal{R}em$ is much larger than to $\mathcal{R}ep$ and $\mathcal{A}dd$. $\mathcal{R}em$ is a more difficult attack in this scenario. (3) From Figure 3.2b, when the number of neighbors K increases, the models' accuracy without poisoning slightly decreases. (4) From Figure 3.2b, K-NN keeps around 80% accuracy to $b = 100$ instance-targeted poisoning when K becomes large.

For general learners, measuring their robustness provably under attacks is harder because there is no clear efficient attack that is provably optimal. In this case, we perform a heuristic attack to study the power of $\mathcal{A}dd_b$. The general idea is that for an example $e = (x, y)$, we poison the dataset by adding b copies of (x, y') into the dataset with the second best label y' in $h(x)$, where b is the Adversary's budget. We then report the accuracy under poisoning with different budget b on classifiers including Logistic regression, 2-layer Multi-layer Perceptron (MLP), 2-layer Convolutional Neural Network (CNN), AlexNet and also K-NN in Figure 3.3a. We get the following conclusion: (1) Models that have low risk without poisoning, such as MLP, CNN and AlexNet, typically have low empirical error, which makes it less robust under poisoning. (2) K-NN with large K have high accuracy under poisoning compared to other models by sacrificing its clean-label prediction accuracy.

Finally, in Figure 3.3b we report on our findins about two regularization mechanics, dropout and $L2$-regularization, on the Neural Network learner and whether adding them can provide better robustness against instance-targeted poisoning $\mathcal{A}dd_b$. We use a 2-layer Multi-layer Perceptron (MLP) as the base learner and adds dropout/regularization to the learner. From the figure, we get the following messages: (1) Dropout and regularization help to improve the accuracy without the attacks (when $b = 0$). (2) These mechanics don't help the accuracy with the $\mathcal{A}dd_b$ attacks. The accuracy under attack is worse than the vanilla Neural Network. We conclude that these simple mechanics cannot help the neural net to defend against instance-targeted poisoning.

Chapter 4

Error amplification of targeted poisoning

Let $P_1, \ldots, P_m$ be m parties that, perhaps interactively, share their (training) datasets $\mathcal{S}_1, \ldots, \mathcal{S}_m$ one by one in m rounds, and at the end a central algorithm Lrn deterministically produces a model h based on $\mathcal{S} = \{\mathcal{S}_1, \ldots, \mathcal{S}_m\}$. Now, suppose the adversary A_b that can replace b of the datasets in $\mathcal{S}_1, \ldots, \mathcal{S}_m$, where b is the budget. Let $t \in \{0, 1\}$ be a bit, where $t = 1$ if some "bad" Boolean property holds over h (e.g., failing to correctly classify a particular point x, leading to a *targeted poisoning attack* [14, 89], or have certain level of overall risk). Suppose $\Pr[t = 1] = \mu$ holds before the attack (in this case μ can be a small value to begin with, something like 0.01 or $1/m$), in this chapter we aim to answer the following question:

> *For an adversary A_b that has budget $b = O(\sqrt{m})$, i.e., changing the messages of $O(\sqrt{m})$ of the parties, how much it can increase the probability $\Pr[t = 1]$ with? That is, can we answer the question that for a certain μ', does an attack with budget $b = O(\sqrt{m})$ exist, that for any learning algorithm Lrn, we have $\Pr[t = 1] \geq \mu'$ after attack?*

Related works. Recent works of [90, 72, 68, 91] showed that such a poisoning adversary with budget $b = \Omega(\sqrt{m})$ can always increase the probability p to $1 - o(1)$ by performing attack. Another line of work that tackles with computational aspects of robust machine learning, started with the exciting works of [78, 79] that shows how to deal with outliers for certain statistical tasks, and importantly do so in polynomial time. The distinction of our dissertation with this line of work on robust statistics is that we care about computational aspects of the "attacker" while they care about making the learning algorithms polynomial time.

Connection to coin-tossing. Collective coin tossing [92] is a fundamental problem in cryptography in which a set of m parties aim to jointly produce a random bit t that remains (close to) random even if an adversary controls a subset of these parties. Interestingly, poisoning attacks on learners can be reduced into attacks on the coin-tossing protocols. This connection was found first by [66]. Let the training dataset $\mathcal{S}_i$ for each party be the message of party P_i. Furthermore, suppose we define a protocol, where the final bit output is exactly the indicator bit t of the learners' property, and $p = \Pr[t = 1]$ be the probability that the protocol outputs 1. Since in the definition of the poisoning setting, there is no restriction on $\mathcal{S}_i$, the attack defined on a coin-tossing protocol can also be applied to the datasets $\mathcal{S}_i$. We now formally describe the attack in the coin-tossing setting as follow.

Problem setting in coin-tossing. Suppose Π is an m-round coin-tossing protocol between m parties, where party P_i sends a single message w_i in round i that could depend on all the previous messages, and the final bit b is a deterministic function of all messages.[1] Now, suppose an adversary aims to increase the probability of $\Pr[t = 1]$. This is called a *up-biasing* attacks, as adversary can *increase* the probability of outputting 1 (rather than either increasing or decreasing it). We deal with *b-replacing* adversaries who can replace b of the messages as follows. Suppose messages $w_1, \ldots, w_{i-1}$ are already finalized and party P_i is about to send w_i in round i. The adversary will have a chance to replace w_i, based on the knowledge of w_i.[2] Equivalently, we will think of the protocol as a *random process* $(w_1, \ldots, w_m)$ with m steps, and a b-replacing adversary will be allowed to override the content of b of the steps, in which case the rest of the random process will depend on the new values. The goal of the adversary is to increase the probability of $\Pr[t = 1]$ for a Boolean function $t(w_1, \ldots, w_m) = t \in \{0, 1\}$. Informally speaking, we would like to know what are the most robust random process in this setting.

Up-biasing aspect. Studying up-biasing attacks is important due to several reasons. Firstly, up-biasing attacks allow modeling adversaries who have a particular output preferred in mind. For example, the coin tossing model's output might determine whether a contract would be signed or not. Then, a party who prefers signing the contract wants to increase the chance of outputting $t = 1$. Moreover, up-biasing attacks allow modeling "undesired" properties defined over random processes and robustness of the process to be pushed into those events, and allow us to unifying the poisoning attack into this setting.

[1] This is also called a *single-turn* protocol.
[2] This is also called the *strongly adaptive* replacement model [93].

Previous attacks for arbitrary length messages. Kalai, Komargodski, and Raz [94] showed that in the "many-replacement" regime where $b = \Omega(\sqrt{m})$, a different attack in the binary setting of [95] can be achieved in polynomial time.[3] Building upon [94], Etesami, Mahloujifar and Mahmoody [90, 72] showed how to extend this result to arbitrary message length and obtain (again up-biasing) attacks in polynomial time, but again only when $b \geq \Omega(\sqrt{m})$. Finally, the recent works of Khorasgani, Maji, Mukherjee, and Wang [97, 98] showed how to get *one-directional* attacks for large messages when $t = 1$.

We summarize the difference of our result with the related work in Table 4.1.

	Up-biasing	Poly-time	Corruption model	Budget b	Rounds
[90, 72]	✓	✓	Replacing	$\Omega(\sqrt{m})$	Any
Our result	✓	✓	Replacing	Any	Any
[99]	-	-	Replacing	1	Any
[93]	-	-	Replacing	$\Omega(\sqrt{m})$	1
[100, 101]	-	-	Replacing	Any	1
[94, 96]	-	-	Adaptive	$\Omega(\sqrt{m})$	Any
[97]	-	-	Replacing	1	Any
[98]	-	-	Adaptive	1	Any

Table 4.1: Summary of related attacks on single-turn coin tossing protocols. The achieved biases are $\Omega(b/\sqrt{m})$ when $b = O(\sqrt{m})$ and are $\Omega(1)$ for larger b.

Why all previous attacks need $b = \Omega(\sqrt{m})$ replacements. The up-biasing attacks of [90, 94, 72] have a similar core that make them rely on many $b = \Omega(\sqrt{m})$ number of replacments to lead to any bias towards 1. These attacks first show that certain specific attacks with *unlimited* budget can significantly bias the output of the function towards 1. Then, in the second step, they show that the number of replacements of such ∞-replacing attacks will not be more than $O(\sqrt{m})$.

This method cannot be directly applied to any $b = o(\sqrt{m})$.

Connection to computational isoperimetry in product spaces. Let $\overline{\mathbf{w}} \equiv (\mathbf{w}_1 \times \cdots \times \mathbf{w}_n)$ be a product distribution of dimension n, and let HD be the Hamming distance $\mathrm{HD}(\overline{w}, \overline{w}') = |\{i \mid w_i \neq w'_i\}|$. Then, a basic question in functional analysis is how quickly noticeable events expand under Hamming distance. It is known, e.g., by results implicit in [102, 103] and explicit in [100, 101][4] that for a set $\mathcal{S}$ has measure μ, the k-expansion of it (i.e., the set of points with a neighbor in $\mathcal{S}$ of distance at most

[3] Interestingly, the main result of [94] focuses on *unidirectional* attacks and shows that the output of any single-turn protocol can be attacked (only information theoretically) by a (standard) adaptive *unidirectional* adversary replacing $b = \Omega(\sqrt{m})$ parties. The recent breakthrough of Haitner and Karidi-Heller [96] generalized the main result of [94] to any general, perhaps multi-turn, protocol. Our focus here, however, is on single-turn protocols.

[4] A weaker version for uniform bits is known as the blowing-up lemma [104].

k) will have have measure at least $\mu + \Omega(b \cdot \mu/\sqrt{m})$ for $b = O(\sqrt{m})$. The previous works of [90, 72] introduced an *algorithmic* variant of the measure concentration phenomenon and showed how to obtain polynomial time algorithms that achieve the following. Given a random point $\overline{w} \in \mathbf{w}$, we can *find* a neighbor of distance at most b in $\mathcal{S}$ with probability $\mu + \Omega(b \cdot \mu/\sqrt{m})$. Their result above only apply to the setting where $b \geq \Omega(\sqrt{m})$, and it remained open to obtain such computational concentration for any small $b = o(\sqrt{m})$. For such small b, the problem is more suitable to be called an *isoperimetric* problem, due to historic reasons. Here, a poly-time attacks on the coin-tossing setting can also serve as a poly-time algorithm that gives an lower bound of the isoperimetry, hence "computational isoperimetry".

4.1 Preliminaries

Notation. We use calligraphic letters (e.g., $\mathcal{X}$) for sets. All distributions and random variables in this section 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 $\Pr[\mathbf{w} \in \mathcal{S}] = \Pr_{w \leftarrow \mathbf{w}}[w \in \mathcal{S}]$ or simply as $\Pr[\mathcal{S}]$ when $\mathbf{w}$ is clear from the context. 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 $(\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_m)$ we refer to a sequence of m *jointly sampled* random variables. For a vector $(w_1 \ldots w_m)$, 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 vector $x = u_{\leq i-1}$ and $y = u_i$, by xy we denote the vector $u_{\leq i-1}$ that appends u_i as the last coordinate of x. For a jointly distributed random variables $(\mathbf{u}, \mathbf{v})$, by $(\mathbf{u} \mid \mathbf{v} = v)$ or we denote the random variable $\mathbf{u}$ conditioned on $\mathbf{v} = v$. When it is clear from the context, we simply write $(\mathbf{u} \mid v)$ or $\mathbf{u}[v]$ instead. By $\mathbf{u} \times \mathbf{v}$ we refer to the product distribution in which $\mathbf{u}$ and $\mathbf{v}$ are sampled independently. $\mathrm{HD}(\overline{u}, \overline{v}) = |\{i \mid u_i \neq v_i\}|$ denotes the Hamming distance for vectors of m coordinates.

Random processes. Let $\mathbf{w}_{\leq m} \equiv (\mathbf{w}_1, \ldots, \mathbf{w}_m)$ be a sequence of jointly distributed random variables. We can interpret the distribution of $\mathbf{w}_{\leq i}$ as a random process in which the i^{th} block w_i is sampled from the marginal distribution $(\mathbf{w}_i \mid w_{\leq i-1}) \equiv (\mathbf{w}_i \mid \mathbf{w}_{\leq i-1} = w_{\leq i-1}) \equiv \mathbf{w}_i[w_{\leq i-1}]$. We also use $\overline{\mathbf{w}}[\cdot]$ to denote an oracle sampling algorithm that given $w_{\leq i}$ returns a sample from $\overline{\mathbf{w}}[w_{\leq i}]$.

Attack model. Our adversaries *replace* a message/block in a random process. Namely, they observe the blocks one by one and sometimes intervene to replace them with a new value. (The new values

will subsequently change the way the random process will proceed.) Hence, we refer to them as *replacing* adversaries. Such adversaries are equivalent to *strongly adaptive corrupting* adversaries as defined in [93].

Definition 4.1.1 (Online replacing attacks on random processes)**.** Let $\mathbf{w}_{\leq m} \equiv (\mathbf{w}_1, \ldots, \mathbf{w}_m)$ be a random process. Suppose $\mathsf{Tam}(x, \sigma) \to (x', \sigma')$ is a (potentially randomized) algorithm with the following syntax. It takes as input some (randomness,) x and σ, where σ is interpreted as a "state", and it outputs (x', σ'). We call such algorithm an *online replacing adversary* and define the following properties for it.

We define the following notions for $\mathbf{w}_{\leq m}$.

- **The generated and output random processes under replacing attacks.** Suppose Tam is an replacing algorithm. We now define two random processes that result from running the replacing adversary Tam to influence the original random process $\overline{\mathbf{w}}$. For $i = 1, 2, \ldots, m$, we first sample $u_i \leftarrow (\mathbf{w}_i \mid \mathbf{w}_{\leq i-1} = v_{\leq i-1})$, and then we obtain $(v_i, \sigma_i) \leftarrow \mathsf{Tam}(u_i, \sigma_{i-1})$. If at any point during this process $\Pr[\mathbf{w}_{\leq i} = v_{\leq i}] = 0$, we will output $u_{i+1} = \cdots = u_m = v_{i+1} = \cdots = v_m = \perp$. We call $(\overline{\mathbf{u}}, \overline{\mathbf{v}})$ the *jointly generated* random processes under the attack. We also refer to $\overline{u}$ as the *original values* and $\overline{v}$ as the *output* of the random process under the attack Tam.

- **Online replacing.** We call Tam a *valid* (online replacing) attack on $\mathbf{w}_{\leq m}$, if with probability 1 over the generation of $\overline{u}, \overline{v}$, it holds that none of the coordinates are $\perp$ (i.e., $\Pr[\mathbf{w}_{\leq i} = v_{\leq i}] \neq 0$.) In this section we always work with valid online replacing attacks, even if they are not called valid.

- **Budget of replacing attacks.** Replacing adversary Tam has *budget* b, if

$$\Pr[\mathrm{HD}(\overline{\mathbf{u}}, \overline{\mathbf{v}}) \leq b] = 1,$$

  where $(\overline{\mathbf{u}}, \overline{\mathbf{v}})$ are the jointly generated random processes that are also jointly distributed.

- **Algorithmic efficiency of attacks.** If $\mathbf{w}_{\leq m}$ is indexed by m as a member of a *family* of joint distributions defined for all $m \in \mathbb{N}$, then we call an online or offline replacing algorithm *efficient*, if its running time is at most $\mathrm{poly}(M)$ where M is the total bit-length representation of any $\overline{w} \in \mathrm{Supp}(\mathbf{w}_{\leq m})$. We would also consider efficiency where the replacing algorithm uses an oracle. In particular, we say an attack $\mathsf{Tam}^{\overline{\mathbf{w}}[\cdot]}$ with oracle access to sampler $\overline{\mathbf{w}}[\cdot]$ is efficient if it runs in time $\mathrm{poly}(M)$.

◇

We now recall the so-called Doob martingale of a (Boolean-output) random process.

Definition 4.1.2 (Doob martingale, partial averages, and their approximate variant)**.** For random process $\mathbf{w}_{\leq m} \equiv (\mathbf{w}_1, \ldots, \mathbf{w}_m)$, let $f\colon \mathrm{Supp}(\overline{\mathbf{w}}) \mapsto \mathbb{R}$, $i \in [m]$, and $w_{\leq i} \in \mathrm{Supp}(\mathbf{w}_{\leq i})$. Then we use the notation $\bar{f}(w_{\leq i}) = \mathbb{E}_{\overline{w} \leftarrow (\overline{\mathbf{w}} | w_{\leq i})}[f(\overline{w})]$ to define the expected value of f for a sample from $\mathbf{w}_{\leq m}$ conditioned on the prefix $w_{\leq i}$ and refer to it as a *partial-average* of f. In particular, using notation $w_{\leq 0} = \emptyset$, we have $\bar{f}(\emptyset) = \mathbb{E}[f(\overline{\mathbf{w}})]$. The random process $(\bar{f}(\mathbf{w}_{\leq 1}), \ldots, \bar{f}(\mathbf{w}_{\leq m}))$ is called the Doob martingale of the function f over the random process $\mathbf{w}_{\leq m}$. For the same $\mathbf{w}_{\leq m}$ and $\bar{f}(\cdot)$, we call $\tilde{f}(\cdot)$ an (additive) ε-approximation of $\bar{f}(\cdot)$, if for all $w_{\leq i} \in \mathrm{Supp}(\mathbf{w}_{\leq i})$, it holds that $\tilde{f}(w_{\leq i}) \in \bar{f}(w_{\leq i}) \pm \varepsilon$. ◇

If one is given oracle access to ℓ samples from $(\mathbf{w}_i \mid w_{\leq i})$, then by averaging them, one can obtain (due to the Hoeffding inequality) an ε-approximation of $\tilde{f}(w_{\leq i})$ for with probability $1 - \exp(-\ell/\varepsilon^2)$.

4.1.1 Useful facts

We use the following variant of the Azuma inequality which is proved in [105].

Lemma 4.1.3 (Azuma's inequality for dynamic interval lengths (Theorem 2.5 in [105]))**.** *Let $\overline{\mathbf{t}} \equiv (\mathbf{t}_1, \ldots, \mathbf{t}_m)$ be a sequence of m jointly distributed random variables such that for all $i \in [m]$, and for all $t_{\leq i-1} \sim \mathbf{t}_{\leq i-1}$, we have*

$$\exists t^*, \quad \Pr_{t_i \sim \mathbf{t}_i | t_{\leq i-1}} [t^* + \eta_i \geq t_i \geq t^* - \eta_i] = 1$$

and $\mathbb{E}[\mathbf{t}_i \mid t_{\leq i-1}] \geq 0$. Then, we have

$$\Pr\left[\sum_{i=1}^{m} \mathbf{t}_i \leq -s\right] \leq \mathrm{e}^{\frac{-s^2}{2\sum_{i=1}^{m} \eta_i^2}}$$

Lemma 4.1.4 (Azuma's inequality for dynamic interval lengths under approximate conditions)**.** *Let $\overline{\mathbf{t}} \equiv (\mathbf{t}_1, \ldots, \mathbf{t}_m)$ be a sequence of m jointly distributed random variables such that for all $i \in [m]$, and for all $t_{\leq i-1} \sim \mathbf{t}_{\leq i-1}$, we have*

$$\exists t^*, \quad \Pr_{t_i \sim \mathbf{t}_i | t_{\leq i-1}} [|t_i| \geq 1] = 0$$

$$\exists t^*, \quad \Pr_{t_i \sim \mathbf{t}_i | t_{\leq i-1}} [t^* + \eta_i \geq t_i \geq t^* - \eta_i] \geq 1 - \gamma$$

and $\mathbb{E}[\mathbf{t}_i \mid t_{\leq i-1}] \geq -\gamma$. *Then, we have*

$$\Pr\left[\sum_{i=1}^{m} \mathbf{t}_i \leq -s\right] \leq \mathrm{e}^{\frac{-(s-2m\gamma)^2}{2\sum_{i=1}^{m}\eta_i^2}} + m \cdot \gamma$$

Proof. If we let $\gamma = 0$, Lemma 4.1.4 becomes equivalent to Lemma 4.1.3. Here we sketch why Lemma 4.1.4 can also be reduced to the case that $\gamma = 0$ (i.e., Azuma inequality). We build a sequence $\mathbf{t}'_i$ from $\mathbf{t}_i$ as follows: Sample $t_i \sim \mathbf{t}_i \mid t_{\leq i-1}$, if $|t_i - t^*| \leq \eta_i$, output $t'_i = t_i + 2\gamma$ otherwise output $t^* + 2\gamma$. We have $\mathbb{E}[\mathbf{t}'_i \mid t'_{\leq i-1}] \geq 0$ and $\Pr[|t'_i - t^* - 2\gamma| > \tau_i] = 0$. Now we can use Lemma 4.1.4 for the basic case of $\gamma = 0$ for the sequence $\mathbf{t}'_i$ and use it to get a looser bound for sequence $\mathbf{t}_i$, using the fact that $\exists i \in [m], |t_i - t^*| \geq \eta_i$ happens with probability at most $m \cdot \gamma$. □

4.2 The attack

In this section, we design and analyze our b-replacing up-biasing attack on random processes. We first describe our attack in an idealized model in which the partial-average oracle $\bar{f}(\cdot)$ and "maximum child" of a prefix of the process are available for free. In Section 4.2.1, we show that our attack can be made polynomial-time using an approximation of the partial-average oracle that can be obtained in polynomial time.

Construction 4.2.1 (b-replacing attack using exact oracles)**.** This attack uses the exact partial-average oracle $\bar{f}(\cdot)$ and another oracle that returns "the best choice" for the next block (see u^*_{i+1} defined below). The attack is also parameterized by a vector $\lambda_{\leq b} = (\lambda_1, \ldots, \lambda_b) \in [0,1]^b$ for some integer $b \leq m$ which is adversary's budget. The attack will keep state $\sigma_i = (u_{\leq i}, v_{\leq i})$ where $u_{\leq i}$ are the original values and $v_{\leq i}$ are the output values under attack.[5] Having state $(u_{\leq i}, v_{\leq i})$ and for given u_{i+1} the algorithm Tam will decide on whether to keep or replace u_{i+1}, using $u^*_{i+1} = \mathrm{argmax}_{u'_{i+1}} \bar{f}(v_{\leq i}, u'_{i+1})$, $\bar{f}^* = \bar{f}(v_{\leq i}, u^*_{i+1})$, and $d = \mathrm{HD}(u_{\leq i}, v_{\leq i})$ as follows.

- (Case 0) If $d \geq b$, do not change u_{i+1} and output $v_{i+1} = u_{i+1}$.
- (Case 1) if Case 0 does not happen and $\bar{f}(v_{\leq i}, u_{i+1}) < \bar{f}^* - \lambda_{d+1}$, then $\mathsf{Tam}[\lambda_{\leq b}](u_{i+1})$ will return the output $v_{i+1} = u^*_{i+1}$ which is different from u_{i+1}.
- (Case 2) If Cases 0, 1 do not happen, do not change u_{i+1} and output $v_{i+1} = u_{i+1}$.

In all the cases above, Tam will also update the state as $\sigma_{i+1} = (u_{\leq i+1}, v_{\leq i+1})$.

[5] Attack would need $v_{\leq i}$ and the "used part of the budget" $\mathrm{HD}(u_{\leq i}, v_{\leq i})$. Both of these can be obtained from $\sigma_i = (u_{\leq i}, v_{\leq i})$.

Notation. Suppose we run the attack $\mathsf{Tam}[\lambda_{\le b}]$ on random process $\overline{\mathbf{w}}$ through the process described in Definition 4.1.1. (In particular, u_{i+1} will be sampled from $(\mathbf{w}_{i+1} \mid \mathbf{w}_{\le i} = v_{\le i})$.) We use $(\overline{\mathbf{u}}^{(b)}, \overline{\mathbf{v}}^{(b)})$ to denote the jointly generated random processes under the attack $\mathsf{Tam}[\lambda_{\le b}]$. (This notation allows us to distinguish between the generated random processes under attacks with different budget.) We sometimes use $(\overline{\mathbf{u}}^{(\infty)}, \overline{\mathbf{v}}^{(\infty)})$ to denote $(\overline{\mathbf{u}}^{(m)}, \overline{\mathbf{v}}^{(m)})$ as they are the same distributions. Also, let

$$\mu_b = \mathop{\mathbb{E}}_{(\overline{u},\overline{v})\sim(\overline{\mathbf{u}}^{(b)},\overline{\mathbf{v}}^{(b)})}[f(\overline{v})]$$

denotes the expected value of f over the sequence that is the output of b-replacing attack of Construction 4.2.1. For $b = 0$ we have and $\mu_0 = \mu = \mathbb{E}[f(\overline{\mathbf{w}})]$.

Lemma 4.2.2 below shows that the increase in μ_b compared with μ_{b-1} can be related to the "threshold parameter" λ_b and the probability that an attack with *unlimited* (or equivalently just m) budget with threshold parameters $\lambda_1, \ldots, \lambda_b, \lambda'_{b+1}, \ldots, \lambda'_m$ makes at least b replacements.

Lemma 4.2.2. *We have*

$$\mu_b \ge \mu_{b-1} + \lambda_b \cdot \Pr_{(\overline{u},\overline{v})\sim(\overline{\mathbf{u}}^{(\infty)},\overline{\mathbf{v}}^{(\infty)})}[\mathrm{HD}(\overline{u},\overline{v}) \ge b].$$

Proof. For any $j \in \{0,1,2\}$, let C_j^b be the Boolean random variable over (u_{i+1}, σ_i) that determines which case of the attack Tam with budget b happens on prefix $(v_{\le i}, u_{i+1})$ where $v_{\le i}$ is the finalized output prefix, $u_{\le i}$ is the original prefix and u_{i+1} is the original sampled block at round $i+1$. For all $(v_{\le i}, u_{\le i}, u_{i+1})$ we have $\sum_{j=0}^{2} C_j^b(u_{i+1}, \sigma_i) = 1$ because the cases complement each other.

In the rest of the proof, whenever $u_{\le i}$ and $v_{\le i}$ are clear from the context, we will use $C_j^b(u_{i+1})$ instead of $C_j^b(u_{i+1}, \sigma_i)$. In the following, when the threshold parameters $\lambda_1, \ldots, \lambda_b$ are clear from the context, we will use Tam instead of $\mathsf{Tam}[\lambda_{\le b}]$.

For all $u_{\le i}, v_{\le i} \in \mathrm{Supp}(\mathbf{u}_{\le i}, \mathbf{v}_{\le i})$ we have the following qualities for different cases of the attack.

- Case 0:

$$\mathop{\mathbb{E}}_{(u_{i+1},v_{i+1})\sim(\mathbf{u}^b_{i+1},\mathbf{v}^b_{i+1})[u_{\le i},v_{\le i}]}\left[\left(\bar{f}(v_{\le i}, v_{i+1}) - \bar{f}(v_{\le i}, u_{i+1})\right) \cdot C_0^b(u_{i+1})\right] = 0. \tag{4.1}$$

- Case 1:

$$C_1^b(u_{i+1}) = (C_1^\infty(u_{i+1}) \wedge \mathrm{HD}(u_{\le i}, v_{\le i}) < b). \tag{4.2}$$

  This is because as long as the number of replacements is fewer than b, Case 1 of the attack with budget b would go through whenever Tam with budget of m does so.

- Case 2:

$$\mathop{\mathbb{E}}_{(u_{i+1},v_{i+1})\sim(\mathbf{u}^b_{i+1},\mathbf{v}^b_{i+1})[u_{\le i},v_{\le i}]}\left[\left(\bar{f}(v_{\le i},v_{i+1})-\bar{f}(v_{\le i},u_{i+1})\right)\cdot C_2^b(u_{i+1}))\right]=0. \tag{4.3}$$

  This is correct because either $C_2^b(v_{\le i},u_{i+1})=0$ or $u_{i+1}=v_{i+1}$.

We define a notation $g(v_{\le i+1},u_{\le i+1})=\bar{f}(v_{\le i+1})-\bar{f}(v_{\le i},u_{i+1})$. In the following We use the shorten forms of $\mathbb{E}_{(\mathbf{u}_{\le i},\mathbf{v}_{\le i})}$ and $\mathbb{E}_{(\overline{\mathbf{u}},\overline{\mathbf{v}})[u_{\le i},v_{\le i}]}$ to refer to $\mathbb{E}_{(u_{\le i},v_{\le i})\sim(\mathbf{u}_{\le i},\mathbf{v}_{\le i})}$ and $\mathbb{E}_{(u,v)\sim(\overline{\mathbf{u}},\overline{\mathbf{v}})[u_{\le i},v_{\le i}]}$. We have

$$\begin{aligned}
&\mathop{\mathbb{E}}_{(\overline{\mathbf{u}}^{(b)},\overline{\mathbf{v}}^{(b)})}[f(\overline{v})]-\mu=\mathop{\mathbb{E}}_{(\overline{\mathbf{u}}^{(b)},\overline{\mathbf{v}}^{(b)})}\left[\sum_{i=0}^{m-1}(\bar{f}(v_{\le i+1})-\bar{f}(v_{\le i}))\right]\\
&=\mathop{\mathbb{E}}_{(\overline{\mathbf{u}}^{(b)},\overline{\mathbf{v}}^{(b)})}\left[\sum_{i=0}^{m-1}(\bar{f}(v_{\le i+1})-\bar{f}(v_{\le i},u_{i+1}))\right]\quad\text{(by the definition of }\bar{f}) &(4.4)\\
&=\sum_{i=0}^{m-1}\mathop{\mathbb{E}}_{(\mathbf{u}^b_{\le i},\mathbf{v}^b_{\le i})}\mathop{\mathbb{E}}_{(\overline{\mathbf{u}}^{(b)},\overline{\mathbf{v}}^{(b)})[u_{\le i},v_{\le i}]}\left[g(v_{\le i+1},u_{\le i+1})\cdot\left(\sum_{j=0}^{2}C_j^b(u_{i+1})\right)\right]\\
&=\sum_{i=0}^{m-1}\mathop{\mathbb{E}}_{(\mathbf{u}^b_{\le i},\mathbf{v}^b_{\le i})}\mathop{\mathbb{E}}_{(\overline{\mathbf{u}}^{(b)},\overline{\mathbf{v}}^{(b)})[u_{\le i},v_{\le i}]}\left[g(v_{\le i+1},u_{\le i+1})\cdot C_1^b(u_{i+1})\right]\quad\text{(by (4.3) and (4.1))} &(4.5)\\
&=\sum_{i=0}^{m-1}\mathop{\mathbb{E}}_{(\mathbf{u}^b_{\le i},\mathbf{v}^b_{\le i})}\mathop{\mathbb{E}}_{(\overline{\mathbf{u}}^{(b)},\overline{\mathbf{v}}^{(b)})[u_{\le i},v_{\le i}]}\left[g(v_{\le i+1},u_{\le i+1})\cdot(C_1^{(\infty)}(u_{i+1})\wedge(\mathrm{HD}(u_{\le i},v_{\le i})<b)\right]\quad\text{(by (4.2))}\\
&=\sum_{i=0}^{m-1}\mathop{\mathbb{E}}_{(\mathbf{u}^\infty_{\le i},\mathbf{v}^\infty_{\le i})}\mathop{\mathbb{E}}_{(\overline{\mathbf{u}}^{(b)},\overline{\mathbf{v}}^{(b)})[u_{\le i},v_{\le i}]}\left[g(v_{\le i+1},u_{\le i+1})\cdot(C_1^{\infty}(u_{i+1})\wedge(\mathrm{HD}(u_{\le i},v_{\le i})<b))\right]. &(4.6)
\end{aligned}$$

The last equality above holds, because for all $u_{\le i},v_{\le i}$ where $\mathrm{HD}(u_{\le i},v_{\le i})<b$,

$$\Pr[(\mathbf{u}^b_{\le i},\mathbf{v}^b_{\le i})=(u_{\le i},v_{\le i})]=\Pr[(\mathbf{u}^{(\infty)}_{\le i},\mathbf{v}^{(\infty)}_{\le i})=(u_{\le i},v_{\le i})].$$

The reason for this is that as long as we have not used the full budget b, the b-replacing attack will behave as if its budget is infinite.

Similarly, for the adversary Tam with budget $b-1$ we have

$$\mathop{\mathbb{E}}_{(\overline{\mathbf{u}}^{(b-1)},\overline{\mathbf{v}}^{(b-1)})}[f(\overline{v})]-\mu=\sum_{i=0}^{m-1}\mathop{\mathbb{E}}_{(\mathbf{u}^{(\infty)}_{\le i},\mathbf{v}^{(\infty)}_{\le i})_{\le i}}\mathop{\mathbb{E}}_{(\overline{\mathbf{u}}^{(b-1)},\overline{\mathbf{v}}^{(b-1)})[u_{\le i},v_{\le i}]}[\eta(u_{\le i+1},v_{\le i+1})]. \tag{4.7}$$

where $\eta(u_{\le i+1},v_{\le i+1})=g(v_{\le i+1},u_{\le i+1})\cdot\left(C_1^{\infty}(u_{i+1})\wedge(\mathrm{HD}(u_{\le i},v_{\le i})<b-1)\right)$. Therefore, by combining Equations (4.6) and (4.7) we have

$$\begin{aligned}
&\mathop{\mathbb{E}}_{(\overline{\mathbf{u}}^{(b)},\overline{\mathbf{v}}^{(b)})}[f(\overline{v})] - \mathop{\mathbb{E}}_{(\overline{u},\overline{v})\sim(\overline{\mathbf{u}}^{(b-1)},\overline{\mathbf{v}}^{(b-1)})}[f(\overline{v})] = \\
&\sum_{i=0}^{m-1} \mathop{\mathbb{E}}_{(\mathbf{u}_{\le i}^{(\infty)},\mathbf{v}_{\le i}^{(\infty)})} \mathop{\mathbb{E}}_{(\overline{\mathbf{u}}^{(b)},\overline{\mathbf{v}}^{(b)})[u_{\le i},v_{\le i}]} \left[g(v_{\le i+1},u_{\le i+1})\cdot C_1^{\infty}(u_{i+1})\cdot(\mathrm{HD}(u_{\le i},v_{\le i})=b-1)\right] \\
&\ge \sum_{i=0}^{m-1} \mathop{\mathbb{E}}_{(\mathbf{u}_{\le i}^{(\infty)},\mathbf{v}_{\le i}^{(\infty)})} \left[\lambda_b \cdot \mathop{\mathbb{E}}_{(\overline{\mathbf{u}}^{(b)},\overline{\mathbf{v}}^{(b)})[u_{\le i},v_{\le i}]} \left[C_1^{\infty}(u_{i+1})\cdot(\mathrm{HD}(u_{\le i},v_{\le i})=b-1)\right]\right] \\
&= \lambda_b \cdot \Pr_{(\overline{\mathbf{u}}^{(\infty)},\overline{\mathbf{v}}^{(\infty)})}[\mathrm{HD}(\overline{u},\overline{v})\ge b].
\end{aligned}$$

The last equality above holds because whenever $C_1^{(\infty)}$ holds, we know that Tam will replace u_{i+1} with $v_{i+1}\neq u_{i+1}$ and this makes the hamming distance of $u_{\le i+1}$ from $v_{\le i+1}$ equal to b. □

Now we prove the following lemma about the power of attacks with infinite budget. Claim 19 in [90] also prove a similar bound for their attack but our attack achieves a better bound because of the fact that our attack has only one step in which the replacement might happen which allows us to make a better use of Azuma's inequality with dynamic interval (See Lemma 4.1.3).

Lemma 4.2.3. *If $\mu_\infty = \mathbb{E}_{(\overline{u},\overline{v})\sim(\overline{\mathbf{u}}^{(\infty)},\overline{\mathbf{v}}^{(\infty)})}[f(\overline{v})]$ and $\lambda = \max_{i\in[n]}\lambda_i$, then*

$$\mu_\infty \ge 1 - e^{-\frac{2\mu^2}{m\lambda^2}}.$$

Proof. We define a sequence of random variables $\overline{\mathbf{t}} = (\mathbf{t}_1,\ldots,\mathbf{t}_m)$, where $t_{i+1} = \bar{f}(v_{\le i+1}) - \bar{f}(v_{\le i})$ is a random variable that is dependent on $v_{\le i+1}$. Then we have

$$\begin{aligned}
&\mathop{\mathbb{E}}_{(\overline{\mathbf{u}}^{(\infty)},\overline{\mathbf{v}}^{(\infty)})[u_{\le i},v_{\le i}]}[\bar{f}(v_{\le i+1}) - \bar{f}(v_{\le i})] \\
&\ge \mathop{\mathbb{E}}_{(\overline{\mathbf{u}}^{(\infty)},\overline{\mathbf{v}}^{(\infty)})[u_{\le i},v_{\le i}]}[\bar{f}(v_{\le i},u_{i+1}) - \bar{f}(v_{\le i})] = 0.
\end{aligned}$$

Therefore, $\overline{\mathbf{t}}$ defines a sub-martingale. Furthermore, we have

$$\bar{f}^* \ge \bar{f}(v_{\le i+1}) \ge \bar{f}^* - \lambda.$$

Therefore, t_i always falls in an interval of size λ. Hence, applying the right variant of Azuma's Inequality (as stated in Lemma 4.1.3) over $\overline{\mathbf{t}}$, we have

$$\Pr_{(\overline{\mathbf{u}}^{(\infty)},\overline{\mathbf{v}}^{(\infty)})}[f(v_{\le m})=0] = \Pr_{(\overline{\mathbf{u}}^{(\infty)},\overline{\mathbf{v}}^{(\infty)})}[f(\overline{v})-\mu\le-\mu] = \Pr_{(\overline{\mathbf{u}}^{(\infty)},\overline{\mathbf{v}}^{(\infty)})}\left[\sum_{i=1}^{m} t_i \le -\mu\right] \le e^{-\frac{2\mu^2}{m\lambda^2}}. \tag{4.8}$$

Now, leveraging the fact that f outputs in $\{0,1\}$ and relying on Inequality (4.8), we have

$$\Pr_{(\overline{\mathbf{u}}^{(\infty)},\overline{\mathbf{v}}^{(\infty)})}[f(\overline{v})=1] = 1 - \Pr_{(\overline{\mathbf{u}}^{(\infty)},\overline{\mathbf{v}}^{(\infty)})}[f(\overline{v})-\mu \le -\mu] \ge 1 - e^{-\frac{2\mu^2}{m\lambda^2}}.$$

□

Lemma 4.2.4. *If $\lambda = \max_{i\in[b]} \lambda_i$, then*

$$\Pr_{(\overline{\mathbf{u}}^{(\infty)},\overline{\mathbf{v}}^{(\infty)})}[\mathrm{HD}(\overline{u},\overline{v}) \ge b] \ge 1 - e^{-\frac{2\mu^2}{m\lambda^2}} - \mu_{b-1}.$$

Proof. First we have

$$\begin{aligned}
&\Pr_{(\overline{\mathbf{u}}^{(\infty)},\overline{\mathbf{v}}^{(\infty)})}[\big(f(\overline{v})=1 \wedge \mathrm{HD}(\overline{u},\overline{v}) < b\big) \vee (\mathrm{HD}(\overline{u},\overline{v}) \ge b)] \\
&= \Pr_{(\overline{\mathbf{u}}^{(\infty)},\overline{\mathbf{v}}^{(\infty)})}[f(\overline{v})=1 \vee \mathrm{HD}(\overline{u},\overline{v}) \ge b] \\
&\ge \Pr_{(\overline{\mathbf{u}}^{(\infty)},\overline{\mathbf{v}}^{(\infty)})}[f(\overline{v})=1] \\
&= \mu_\infty \ge 1 - e^{-\frac{2\mu^2}{m\lambda^2}} \text{ (by Lemma 4.2.3).}
\end{aligned} \tag{4.9}$$

On the other hand, by a union bound we have

$$\begin{aligned}
&\Pr_{(\overline{\mathbf{u}}^{(\infty)},\overline{\mathbf{v}}^{(\infty)})}[\big(f(\overline{v})=1 \wedge \mathrm{HD}(\overline{u},\overline{v}) < b\big) \vee (\mathrm{HD}(\overline{u},\overline{v}) \ge b)] \le \\
&\Pr_{(\overline{\mathbf{u}}^{(\infty)},\overline{\mathbf{v}}^{(\infty)})}[f(\overline{v})=1 \wedge \mathrm{HD}(\overline{u},\overline{v}) < b] + \Pr_{(\overline{\mathbf{u}}^{(\infty)},\overline{\mathbf{v}}^{(\infty)})}[\mathrm{HD}(\overline{u},\overline{v}) \ge b].
\end{aligned} \tag{4.10}$$

The generated process under $b-1$ replacing attack is same as m-replacing attack as long as the number of replacements is less than b. Therefore, it holds that

$$\Pr_{(\overline{\mathbf{u}}^{(\infty)},\overline{\mathbf{v}}^{(\infty)})}[\big(f(\overline{v})=1 \wedge \mathrm{HD}(\overline{u},\overline{v}) < b\big)] \le \Pr_{(\overline{\mathbf{u}}^{(b-1)},\overline{\mathbf{v}}^{(b-1)})}[f(\overline{v})=1] = \mu_{b-1}. \tag{4.11}$$

Now, combining Inequalities (4.9), (4.10) and (4.11) we get

$$\Pr_{(\overline{\mathbf{u}}^{(\infty)},\overline{\mathbf{v}}^{(\infty)})}[\mathrm{HD}(\overline{u},\overline{v}) \ge b] \ge 1 - e^{-\frac{2\mu^2}{m\lambda^2}} - \mu_{b-1}.$$

□

Corollary 4.2.5. *If $\lambda = \max_{i\in[b]} \lambda_i$, then we have*

$$\mu_b \geq \mu_{b-1} + \lambda_b \cdot \left(1 - e^{\frac{-2\mu^2}{m\lambda^2}} - \mu_{b-1}\right).$$

Proof. Combining Lemmas 4.2.4 and 4.2.2 we have

$$\begin{aligned}\mu_b &\geq \mu_{b-1} + \lambda_b \cdot \Pr_{(\overline{\mathbf{u}}^{(\infty)},\overline{\mathbf{v}}^{(\infty)})}[\mathrm{HD}(\overline{u},\overline{v}) \geq b] \quad \text{(by Lemma 4.2.2)} \\ &\geq \mu_{b-1} + \lambda_b \cdot \left(1 - e^{\frac{-2\mu^2}{m\lambda^2}} - \mu_{b-1}\right) \quad \text{(by Lemma 4.2.4).}\end{aligned}$$

□

Theorem 4.2.6. *If $\lambda = \max_{i\in[b]} \lambda_i$, then we have*

$$\mu_b \geq \mu + \left(1 - \prod_{i=1}^{b}(1-\lambda_i)\right) \cdot \left(1 - e^{\frac{-2\mu^2}{m\lambda^2}} - \mu\right).$$

In particular, by setting all $\lambda_i = \frac{\mu}{\sqrt{n}}$ we get

$$\mu_b \geq \mu + \left(1 - \left(1 - \frac{\mu}{\sqrt{m}}\right)^b\right) \cdot \left(1 - e^{-2} - \mu\right).$$

Note that the choice of $\lambda_i = \mu/\sqrt{m}$ above is not optimal when we want to maximize μ_i. The optimal choice does not have a compact closed form and leads to different λ_i's for different remaining budgets.

Proof. We prove this by induction on b. The case of $b = 1$ directly follows from Corollary 4.2.5. For $b > 1$, by Corollary 4.2.5 we have

$$\mu_b \geq \mu_{b-1} + \lambda_b \cdot \left(1 - e^{\frac{-2\mu^2}{m\lambda^2}} - \mu_{b-1}\right),$$

which implies that

$$\mu_b \geq (1-\lambda_b) \cdot \mu_{b-1} + \lambda_b \cdot \left(1 - e^{\frac{-2\mu^2}{m\lambda^2}}\right).$$

Now we can use the induction's hypothesis and replace μ_{b-1} with $\mu + \left(1 - \prod_{i=1}^{b-1}(1-\lambda_i)\right) \cdot \left(1 - e^{-2\mu^2/(m\lambda^2)} - \mu\right)$ which implies that

$$\mu_b \geq \mu + \left(1 - \prod_{i=1}^{b}(1-\lambda_i)\right) \cdot \left(1 - e^{\frac{-2\mu^2}{m\lambda^2}} - \mu\right),$$

and that proves the claim. □

4.2.1 Making the attack polynomial time

In this section, we explain why the attack of the Section 4.2 can be implemented in polynomial time. In particular, we show how we can modify Construction 4.2.1 so that it runs in polynomial time, if one can efficiently sample form the random process conditioned on any prefix. (This is true, e.g., when the random process models a single-turn coin tossing protocol, as the original protocol shall run in polynomial time.)

At a high level, the proofs of this section closely follow the steps of the proofs in Section 4.2, and in each step we show that the proof is robust to using "approximate" values for what we previously assumed to be known exactly. Therefore, for a reader who is not primarily concerned with the polynomial-time aspect of the attack, we suggest reading Section 4.2, which is simpler.

Note that the attack of Construction 4.2.1 is not polynomial time mainly because calculating $\bar{f}(\cdot)$ and $f^*(\cdot)$ oracles is not a polynomial-time task. In order to make the attack polynomial time, we first show that if the algorithm has access to the approximated version of these oracles, it still can achieve almost the same bias towards 1. Then, we show that calculating the approximation of these approximate oracles is actually possible in polynomial-time.

First, we first state our new construction that uses the approximate oracles.

Construction 4.2.7 (b-replacing using approximate partial-average and maximum-child oracles)**.** This attack uses the approximate oracle $\tilde{f}(\cdot)$ (see Definition 4.1.2) so that for all $v_{\leq i}$ we have

$$|\tilde{f}(v_{\leq i}) - \bar{f}(v_{\leq i})| \leq \tau. \tag{4.12}$$

The attack also uses an additional oracle for returning an approximate best choice for next block, such that

$$u^*_{i+1} \in \left\{ u'_{i+1} \mid \Pr\left[\tilde{f}(v_{\leq i}, u_{i+1}) > \tilde{f}(v_{\leq i}, u'_{i+1})\right] \leq \tau \right\} \tag{4.13}$$

and also let $\tilde{f}^*(v_{\leq i}) = \tilde{f}(\tilde{u}^*_{i+1})$ and $d = \mathrm{HD}(u_{\leq i}, v_{\leq i})$. The attack is parameterized by a vector $\lambda_{\leq b} = (\lambda_1, \ldots, \lambda_b) \in [0,1]^b$ for some integer $b \leq m$ which is adversary's budget. The attack will keep a state $\sigma_i = (u_{\leq i}, v_{\leq i})$ where $u_{\leq i}$ are the original values and $v_{\leq i}$ are the output values under attack. Having state $(u_{\leq i}, v_{\leq i})$ and for given u_{i+1} the approximate attacker App will decide on whether to keep or replace u_{i+1}, using u^*_{i+1}, $\tilde{f}^*$, and $d = \mathrm{HD}(u_{\leq i}, v_{\leq i})$ as follows.

- (Case 0) If $d \geq b$ do nothing and output $v_{i+1} = u_{i+1}$.

- (Case 1) if Case 0 does not happen and $\tilde{f}(v_{\leq i}, u_{i+1}) < \tilde{f}^*(v_{\leq i}) - \lambda_z$, then output $v_{i+1} = u^*_{i+1}$.
- (Case 2) If Case 0 and Case 1 not happen, do nothing and set $v_{i+1} = u_{i+1}$.

In all the cases above, App will also update the state σ_{i+1} accordingly by setting $\sigma_{i+1} = (u_{\leq i+1}, v_{\leq i+1})$.

Now we show that if conditions 4.12 and 4.13 hold, then the attack of Construction 4.2.7 can achieve the desired bias.

Lemma 4.2.8. *Let μ_b be the average of the output bit after applying the attack of Construction 4.2.7. Then, for $\lambda = \max_{i\in[b]} \lambda_i$ we have*

$$\mu_b \geq \mu + \left(1 - \prod_{i=1}^{b}(1-\lambda_i)\right) \cdot \left(1 - e^{\frac{-2(\mu-5m\tau)^2}{m\cdot\lambda^2}} - \mu\right) - 6m \cdot b \cdot \tau.$$

In particular, by setting all $\lambda_i = \frac{\mu}{\sqrt{m}}$ and $\tau \leq \min(\frac{\mu}{10000m}, \frac{\varepsilon}{12\cdot m^2})$ we get

$$\mu_b \geq \mu + \left(1 - \left(1 - \mu/\sqrt{m}\right)^b\right) \cdot (1 - e^{-1.99} - \mu) - \varepsilon/2.$$

Before proving the theorem above, we mention the following corollary about the power of polynomial time attacks.

Theorem 4.2.9. *For any ε there is a b-replacing $\mathsf{App}^{\overline{\mathsf{w}}[.]}$ attack with oracle access to the sampler from random process that runs in time $\mathrm{poly}(N/(\varepsilon\cdot\mu))$ and achieves bias at least*

$$\left(1 - \left(1 - \frac{\mu}{\sqrt{m}}\right)^b\right) \cdot (1 - e^{-1.99} - \mu) - \varepsilon,$$

where N is the total bit representation of the process.

Proof. We shall only show how to implement the approximate oracles $\tilde{f}(\cdot)$ and $\tilde{f}^*(\cdot)$ in polynomial time. Doing so is possible by continuing the random process (i.e., the coin flipping protocol) many times and taking their average, which is possible to be done efficiently because the attack has oracle access to the process (in the context of coin tossing, this is possible if the protocol is single turn and turns in polynomial time). Using Chernoff-Hoefding bound, we can bound the probability of having error $|\tilde{f} - \bar{f}|$ larger than τ to be at most $\varepsilon/4m$, by making $\mathrm{poly}(m/(\varepsilon\cdot\tau))$ random continuations. We can also ensure the condition for $\tilde{f}^*$ to happen with probability at least $1 - \varepsilon/4m$, by making $\mathrm{poly}(\log(m/\varepsilon)/\tau)$ samples. Of course, there is a chance of our approximation failing but that does

not hurt the proof as we bound the probability of the failure for both type of queries by $\varepsilon/4n$ and in total we make at most $2m$ such queries during the course of the protocol. Therefore, by union bound, the final probability of any of these queries failing is at most $\varepsilon/2$. In the worst-case, for the failure scenarios the average becomes 0 and we lose an additive $\varepsilon/2$ in the final bias. Since our attack at each round makes $\text{poly}(m/(\varepsilon \cdot \tau))$ random continuations, the running time of the attack given oracle access to the sampler is equal to $O(\text{poly}(m/\varepsilon \cdot \tau)) = O(\text{poly}(m/\varepsilon \cdot \mu))$. □

Now we prove Lemma 4.2.8.

Proof of Lemma 4.2.8. The proof is similar to the proof of Theorem 4.2.6. We first need to show that the approximate attack would achieve high bias, in the case of infinite number of replacements. We start by showing a variation of Lemma 4.2.3 with the approximated oracle.

Lemma 4.2.10. *Let $\lambda = \max_{i\in[m]} \lambda_i$. Then, we have*

$$\mu_\infty \geq 1 - e^{-\frac{2(\mu-5m\tau)^2}{m\lambda^2}} - 2\tau m$$

Proof. We define a sequence of random variables $\bar{\mathbf{t}} = (\mathbf{t}_1, \ldots, \mathbf{t}_m)$, where $t_{i+1} = \tilde{f}(v_{\leq i+1}) - \tilde{f}(v_{\leq i})$ is a random variable that is dependent on $v_{\leq i}$.

Then we have

$$\mathop{\mathbb{E}}_{(\mathbf{u}^\infty,\mathbf{v}^\infty)[u_{\leq i},v_{\leq i}]}[\tilde{f}(v_{\leq i+1}) - \tilde{f}(v_{\leq i})] \geq \mathop{\mathbb{E}}_{(\mathbf{u}^\infty,\mathbf{v}^\infty)[u_{\leq i},v_{\leq i}]}[\bar{f}(v_{\leq i}, u_{i+1}) - \bar{f}(v_{\leq i}) - 2\tau] = -2\tau. \tag{4.14}$$

Therefore, $\bar{\mathbf{t}}$ defines an approximate sub-martingale.

Furthermore, by the guarantee of $\tilde{f}^*$ and the way the attack works we have

$$\Pr_{(\mathbf{u}^\infty,\mathbf{v}^\infty)[u_{\leq i},v_{\leq i}]}[\tilde{f}^*(v_{\leq i}) \geq \tilde{f}(v_{\leq i+1}) \geq \tilde{f}^*(v_{\leq i}) - \lambda] \geq 1 - \tau \tag{4.15}$$

Therefore, t_i always falls in an interval of size λ and by Applying the approximate Azuma's inequality (as stated in Lemma 4.1.4) over t_i, we have

$$\Pr\left[\sum_{i=1}^{m} t_i \leq -\mu + \tau\right] \leq e^{-\frac{2(\mu-5m\tau)^2}{m\lambda^2}} + 2m\tau.$$

Then we have

$$\Pr[f(\bar{v}) = 1] \geq \Pr[\tilde{f}(\bar{v}) > \tau] = \Pr\left[\sum t_i \geq \tau - \mu\right] \geq 1 - e^{-\frac{2(\mu-5m\tau)^2}{m\lambda^2}} - 2m\tau.$$

□

We then have the approximated version of Lemma 4.2.2.

Lemma 4.2.11. *We have*

$$\mu_b \geq \mu_{b-1} + \lambda_b \cdot \Pr_{(\overline{u},\overline{v})\sim(\overline{\mathbf{u}}^{(\infty)},\overline{\mathbf{v}}^{(\infty)})}[\mathrm{HD}(\overline{u},\overline{v}) \geq b] - 4m\tau.$$

Proof. The proof is very similar to the proof of Lemma 4.2.2. The only difference is that in Equation (4.4) we switch from $\bar{f}$ to $\tilde{f}$ and we only loose $2\tau m$ because of the approximation error. Namely,

$$\left| \mathop{\mathbb{E}}_{(\overline{\mathbf{u}}^b,\overline{\mathbf{v}}^b)[u_{\leq i},v_{\leq i}]} \left[\sum_{i=1}^{m} \left(\tilde{f}(v_{\leq i+1}) - \tilde{f}(v_{\leq i}) \right) \right] - \mathop{\mathbb{E}}_{(\overline{\mathbf{u}}^b,\overline{\mathbf{v}}^b)[u_{\leq i},v_{\leq i}]} \left[\sum_{i=1}^{m} \left(\bar{f}(v_{\leq i+1}) - \bar{f}(v_{\leq i}) \right) \right] \right| \leq 2\tau m. \quad (4.16)$$

All other equations will remain the same because Equations 4.1, 4.2 and 4.3 still hold when we use $\tilde{f}$ instead of $\bar{f}$.

This will change Equation (4.5) and will add an additive term of $2\tau m$ and we have

$$\mathop{\mathbb{E}}_{(\overline{\mathbf{u}}^b,\overline{\mathbf{v}}^b)}[f(\overline{v})] - \mu \geq$$
$$\sum_{i=0}^{m-1} \mathop{\mathbb{E}}_{(\mathbf{u}^{\infty}_{\leq i},\mathbf{v}^{\infty}_{\leq i})} \left[\mathop{\mathbb{E}}_{(\overline{\mathbf{u}}^b,\overline{\mathbf{v}}^b))[u_{\leq i},v_{\leq i}]} \left[g(v_{\leq i+1},u_{\leq i+1}) \cdot \left(C_1^{\infty}(u_{i+1}) \wedge (\mathrm{HD}(u_{\leq i},v_{\leq i}) < b) \right) \right] \right] - 2m\tau.$$

Similarly, for $b-1$ we have

$$\mathop{\mathbb{E}}_{(\overline{\mathbf{u}}^{b-1},\overline{\mathbf{v}}^{b-1})}[f(\overline{v})] - \mu \leq$$
$$\sum_{i=0}^{m-1} \mathop{\mathbb{E}}_{(\mathbf{u}^{\infty}_{\leq i},\mathbf{v}^{\infty}_{\leq i})} \left[\mathop{\mathbb{E}}_{(\overline{\mathbf{u}}^{b-1},\overline{\mathbf{v}}^{b-1})[u_{\leq i},v_{\leq i}]} \left[g(v_{\leq i+1},u_{\leq i+1}) \cdot \left(C_1^{\infty}(u_{i+1}) \wedge (\mathrm{HD}(u_{\leq i},v_{\leq i}) < b-1) \right) \right] + 2m\tau.$$

By subtracting the two inequalities above the proof of Lemma is complete. □

Lemma 4.2.12. *If* $\lambda = \max_{i\in[b]} \lambda_i$*, then*

$$\Pr_{(\overline{\mathbf{u}}^{\infty},\overline{\mathbf{v}}^{\infty})}[\mathrm{HD}(\overline{u},\overline{v}) \geq b] \geq 1 - e^{-\frac{2(\mu-5m\tau)^2}{m\lambda^2}} - 2m\tau - \mu_{b-1}.$$

Proof. The proof steps of this lemma are exactly as those of Lemma 4.2.4. First we have

$$\begin{aligned}
&\Pr_{(\overline{\mathbf{u}}^{\infty},\overline{\mathbf{v}}^{\infty})}\left[\left(f(\overline{v})=1 \wedge \mathrm{HD}(\overline{u},\overline{v})<b\right) \vee (\mathrm{HD}(\overline{u},\overline{v})\geq b)\right] \\
&= \Pr_{(\overline{\mathbf{u}}^{\infty},\overline{\mathbf{v}}^{\infty})}\left[f(\overline{v})=1 \vee \mathrm{HD}(\overline{u},\overline{v})\geq b\right] \\
&\geq \Pr_{(\overline{\mathbf{u}}^{\infty},\overline{\mathbf{v}}^{\infty})}\left[f(\overline{v})=1\right] \\
&= \mu_{\infty} \\
&\text{(by Lemma 4.2.10)} \geq 1-e^{-\frac{2(\mu-5m\tau)^2}{m\lambda^2}}+2m\tau. \qquad (4.17)
\end{aligned}$$

On the other hand, by union bound we have

$$\begin{aligned}
&\Pr_{(\overline{\mathbf{u}}^{\infty},\overline{\mathbf{v}}^{\infty})}\left[\left(f(\overline{v})=1 \wedge \mathrm{HD}(\overline{u},\overline{v})<b\right) \vee (\mathrm{HD}(\overline{u},\overline{v})\geq b)\right] \\
&\leq \Pr_{(\overline{\mathbf{u}}^{\infty},\overline{\mathbf{v}}^{\infty})}\left[f(\overline{v})=1 \wedge \mathrm{HD}(\overline{u},\overline{v})<b\right] + \Pr_{(\overline{\mathbf{u}}^{\infty},\overline{\mathbf{v}}^{\infty})}\left[\mathrm{HD}(\overline{u},\overline{v})\geq b\right]. \qquad (4.18)
\end{aligned}$$

It also holds that

$$\Pr_{(\overline{\mathbf{u}}^{\infty},\overline{\mathbf{v}}^{\infty})}\left[\left(f(\overline{v})=1 \wedge \mathrm{HD}(\overline{u},\overline{v})<b\right)\right] \leq \Pr_{(\overline{\mathbf{u}}^{b-1},\overline{\mathbf{v}}^{b-1})}\left[f(\overline{v})=1\right]=\mu_{b-1}. \qquad (4.19)$$

Now, combining Inequalities 4.17, 4.18 and 4.19 we get

$$\Pr_{(\overline{\mathbf{u}}^{\infty},\overline{\mathbf{v}}^{\infty})}\left[\mathrm{HD}(\overline{u},\overline{v})\geq b\right] \geq 1-e^{-\frac{2(\mu+5m\tau)^2}{m\lambda^2}}-2m\tau-\mu_{b-1},$$

which finishes the proof. □

Corollary 4.2.13. *If $\lambda = \max_{i\in[b]}\lambda_i$, then we have*

$$\mu_b \geq \mu_{b-1}+\lambda_b\cdot\left(1-e^{\frac{-2(\mu-5m\tau)^2}{m\cdot\lambda^2}}-2m\tau-\mu_{b-1}\right)-4m\cdot\tau.$$

Proof. This corollary follows by combining Lemmas 4.2.11 and 4.2.12. □

Putting things together. Now that we have all the required Lemmas we can prove Theorem using the exact same inductive argument of Theorem 4.2.6. The case of $b = 1$ directly follows from

Corollary 4.2.13. For $b > 1$, by Corollary 4.2.13 we have

$$\mu_b \geq \mu_{b-1} + \lambda_b \cdot \left(1 - e^{\frac{-2(\mu - 5m\tau)^2}{m \cdot \lambda^2}} - 2m\tau - \mu_{b-1}\right) - 4m \cdot \tau,$$

which implies that

$$\mu_b \geq (1 - \lambda_b) \cdot \mu_{b-1} + \lambda_b \cdot \left(1 - e^{\frac{-2(\mu - 5m\tau)^2}{m \cdot \lambda^2}}\right) - 6m \cdot \tau.$$

Now we can use induction hypothesis and replace μ_{b-1} with

$$\mu + \left(1 - \prod_{i=1}^{b-1}(1 - \lambda_i)\right) \cdot \left(1 - e^{\frac{-2(\mu - 5m\tau)^2}{m \cdot \lambda^2}} - \mu\right) - 6m \cdot (b-1) \cdot \tau$$

which implies that

$$\mu_b \geq \mu + \left(1 - \prod_{i=1}^{b}(1 - \lambda_i)\right) \cdot \left(1 - e^{\frac{-2(\mu - 5m\tau)^2}{m \cdot \lambda^2}} - \mu\right) - 6mb\tau,$$

and that proves the claim.

□

Chapter 5

Privacy leakage of data deletion

5.1 Introduction

Machine learning algorithms, in their most basic settings, focus on deriving predictive models with low error by using a collection of training examples $\mathcal{S} = \{e_1, \ldots, e_m\}$. However, a model $h_{\mathcal{S}}$ trained on set $\mathcal{S}$ might reveal (sensitive information about) the examples in $\mathcal{S}$, potentially violating the privacy of the individuals whose contributed those examples. Such exposure, particularly in certain (e.g., medical/political) contexts could be a major concern. In fact, the ever-increasing use of machine learning (ML) as a service [106] for decision making further heightens such privacy concerns. Recent legal requirements (e.g., the European Union's GDPR [10] or California's CCPA [11]) aim to make such privacy considerations mandatory. At the same time, a recent line of work [19, 20, 21, 22] aims at (mathematically) formalizing such privacy considerations and their enforcement.

The work of Shokri et al. [4] demonstrated that natural and even commercialized ML models do, in fact, leak a lot about their training sets. In particular, their work initiated the *membership inference* framework for studying privacy attacks on ML models. In such attacks, an adversary with input example e and access to an ML model $h_{\mathcal{S}}$ aims to deduce if $e \in \mathcal{S}$ or not. In a bigger picture, membership inference of [4] and many follow-up attacks [107, 108, 52, 109, 110, 111, 112, 61, 113, 114] as well as *model inversion* attacks [115, 116, 117, 19] can all be seen as demonstrating ways to *infer* or *reconstruct* information about the data sets used in the ML pipeline based on publicly available auxiliary information about them [41, 118, 51, 119, 120, 121]. A more recent line of work studies the related question of "memorization" in machine learning models set [122, 19, 123, 124].

On the defense side, differential privacy [41, 42, 43] provides a framework to provably limit the

information that would leak about the used training examples. This is done by guaranteeing that including or not including any individual example will have little statistical impact on the distribution of the produced ML model. Consequently, any form of interaction with the trained model h (e.g., even a full disclosure of it) will not reveal too much information about whether a particular example e was a member of the data set or not. Despite being a very powerful privacy guarantee, differential privacy imposes a challenge on the learning process [45, 46, 47, 48, 49, 50] that usually leads to major utility loss when one uses the same amount of training data compared with non-private training [47, 125]. Hence, it is important to understand the level of privacy that can be achieved by more efficient methods as well.

Privacy in the presence of data deletion. The above mentioned attacks are executed in a *static* setting, in which the model is trained once and then the adversary tries to extract information about the training set by interacting with the trained model afterwards. However, this setting is not realistic when models are dynamic and get updated. Clearly, if an ML model gets updated due to a deletion request, we are no longer dealing with a static ML model.

It might initially seem like a perfect deletion of a example e from a model h_S and releasing h_{-e} instead should *help* with preventing leakage about the particular deleted example e. After all, we are *removing* e from the learning process of the model accessible to the adversary. However, the adversary now could potentially access *both* models h_S and h_{-e}, and so it might be able to extract even *more* information about the deleted example e compared to the setting in which the adversary could only access h_S or h_{-e} alone. As a simplified contrived example, suppose the examples $S = \{e_1, \ldots, e_m\}$ are real-valued vectors, and suppose the ML model h_S (perhaps upon many queries) somehow reveals the summation $\sum_{i \in [m]} e_i$. In this case, if the set S is sampled from a distribution with sufficient entropy, the trained model h_S might potentially provide a certain degree of privacy for examples in S. However, if one of the examples e_i is deleted from h_S, then because the updated model h_{-e_i} also returns the updated summation $\sum_{j \neq i} e_j$, then an adversary who extracts both of these summations can reconstruct the deleted record e_i completely. In other words, the very task of deletion might in fact *harm* the privacy of the very deleted example e_i. Hence, in this work we ask: How vulnerable are ML algorithms to leak information about the deleted examples, if an adversary gets to interact with the models both before and after the deletion updates?

5.1.1 Our contribution

In this work, we formally study the privacy implications of machine unlearning. Our approach is inspired by cryptographic definitions, differential privacy, and deletion compliance framework of [22]. More specifically, our contribution is two-fold. First, we initiate a formal study of various attack models in the two categories of *reconstruction* and *inference* attacks. Second, we present practical, simple, yet effective attacks on a *broad* class of machine learning algorithms for classification, regression, and text generation that extract information about the deleted example.

Below, we briefly go over new definitions, the relation between them, and the ideas behind our attacks. In what follows, $h_{\mathcal{S}}$ is the model trained on the set $\mathcal{S}$, and h_{-e} is the model after deletion of the example $e \in \mathcal{S}$. When the context is clear, we might simply use h to denote $h_{\mathcal{S}}$ and h_{del} to denote the model after deletion[1]. We assume that the deletion is *ideal*, in the sense that h_{-e} is obtained by a fresh retrain on $\mathcal{S} \setminus \{e\}$.[2] The adversary will have access to $h_{\mathcal{S}}$ followed by access to h_{-e}.

Deletion inference. Perhaps the most natural question about data leakage in the context of machine unlearning is whether deletion can be inferred. In *membership* inference attack, the job of the adversary is to infer whether an example e is a member of the used training set $\mathcal{S}$ or not by interacting with the produced model $h_{\mathcal{S}}$. In this chapter, we introduce *deletion inference* attacks which are, roughly speaking, analogous to *membership inference* but in the context where some deletion is happening. More specifically, our definition does not capture whether the deletion is happening or not, and our goal (in the main default definition) is only to hide *which* examples are being deleted. In particular, we formalize the goal of a deletion inference adversary to distinguish between a data example $e \in \mathcal{S}$ that was deleted from an ML model $h_{\mathcal{S}}$ and another example $e' \in \mathcal{S}$ (or $e' \notin \mathcal{S}$) that is not deleted from $\mathcal{S}$. We follow the cryptographic game-based style of security definitions. (See Definition 5.3.1 for the formal definition.)

Given examples e_0, e_1, with the promise that one of them is deleted and the other is not, one can always reduce the goal of a deletion inference adversary to *membership inference* by first inferring membership of e_0, e_1 in the two models h, h_{del}. However, given that the adversary has access to both of h, h_{del}, it is reasonable to suspect that much more can be done by a deletion inference adversary than what can be done through a reduction to membership inference. In fact, this is exactly what we show in Section 5.3.3. We show that when both models h, h_{del} can be accessed, relatively simple attacks can be designed to distinguish the deleted examples from the other examples by relying on

[1] Using h_{del} is particularly useful when we want to refer to the model after deletion, without explicitly revealing the deleted example e.

[2] We suspect our attacks should have a good success rate on "approximate" deletion procedures (in which h_{-e} is just close to the ideal version) as well. We leave such studies for future work.

the intuition that a useful model is usually more fit to the training data than to other data. In Section 5.3, we show the power of such attacks on a variety of models and real world data sets for both regression and classification. In each case, we both study deletion inference adversaries who know the full labeled examples e_0, e_1 (and infer which one of them are deleted) as well as stronger attackers who only know the (unlabeled) *instances* x_0, x_1.

Deletion reconstruction. The second category of our attacks focus on *reconstructing* part or all of the deleted example e. As anticipated, reconstruction attacks are *stronger* (and hence harder to achieve) attacks that can be used for obtaining deletion inference attacks as well (see Theorem 5.4.2). In all of our reconstruction attacks, the adversary is not given any explicit examples, and its goal is to extract information about the features of the deleted instance. We now describe some special cases of reconstruction attacks that we particularly study.

- **Deleted instance reconstruction.** Can an adversary fully or approximate find the features of a deleted instance x (where $e = (x, y)$ is the deleted example)? We show that for natural data distributions (both theoretical and real data) the 1-nearest neighbor classifier can completely reveal the deleted instance, even if the adversary has only black-box access to the models before and after deletion. In particular, we show that when the instances are uniformly distributed over $\{0,1\}^d$, and the model is the 1-nearest neighbor model, an adversary can extract virtually *all* of the features of the deleted instance (see Section 5.4.1). We also present attacks on real data for two major application settings: image classification and text generation.
  - **Deleted image reconstruction.** We show similar attacks on 1-nearest neighbor over the Omniglot dataset, where the job of the adversary is to extract *visually similar* pictures to the deleted ones (see Section 5.4.3).
  - **Deleted sentence reconstruction.** We then study deletion reconstruction attacks on language models. Here, a language model gets updated to remove an input (e.g., a sentence) e, and the job of the adversary is to find useful information about e. We show that for simple language models such as bigram or trigram models, the adversary can extract e completely.
- **Deleted label reconstruction.** Suppose we deal with a classification problem. For a deleted example $(x, y) = e$, can an adversary *who does not know the instance* x infer any information about the label of the deleted point? We show that this is indeed possible with a simple idea when the data set is not too large. In particular, the deletion of a point with label c reduces the probability that the new model outputs label c in general, and using this idea we give simple yet successful attacks. Now, suppose the adversary is somehow *aware* of the instance x

of a deleted example $(x, y) = e$. Can the adversary leverage knowing the instance x to learn *more information* about the label y, than each of the models h, h_{del} alone provide? We show that doing so is possible for linear regression. In particular, we show an attack using which one can extrapolate a deleted point's label to a *higher* precision than what is provided through the original model h or the model after deletion h_{del}. (See Section 5.4.4 for more details.)

Weak deletion compliance. The above results all deal with first defining *attack models* and then presenting attacks within those frameworks. Next, we ask if it is possible to realize machine learning algorithms with deletion mechanisms that offer meaningful notions of privacy for the deleted points. We approach this question through the lens of the recent work of Garg et al. [22] in which they provide a general "deletion complience" framework that provides strong definitions of private data deletion. We first give a formal comparison between the framework of [22] with our attack models and show that the deletion compliance framework of [22] indeed captures all of the above-mentioned attack models. Furthermore, we also present a *weakened* variant of the definition of [22] that is adapted to a setting where the fact that deletion happened itself is allowed to leak. We believe this is a natural setting that needs special attention. For example, consider a text with redacted parts; this reveals the fact that deletion has happened, but not necessarily the redacted text. We further weaken the framework of [22] by only revealing to the adversary what can be accessed through *black-box access* to the model and *not* the full state of the model. We show that even such weaker variants of deletion compliance still capture all of our attacks, and hence is sufficient for positive results. This means that, as shown by [22], differential privacy (with strong parameters) can be used to prevent all attacks of this chapter. However, note that enforcing differential privacy comes with costs in efficiency and sample complexity. Hence, it remains an interesting direction to find more efficient schemes (both in terms of running time and sample complexity) that satisfy our weaker notions of deletion compliance introduced in this chapter. See Section 5.5 for more discussions.

Motivation behind the attacks. At a high level, our result is relevant in any context in which (1) the users who provide the data examples care about their privacy and prefer not to reveal their participation in the data set $\mathcal{S}$ (2) the system aims to provide the deletion operation, perhaps due to legal requirements. Condition (1) essentially holds in any scenario in which membership inference constitutes a legitimate threat. In scenarios where conditions (1) and (2) hold, if the adversary maintains continuous access to the machine learning model (e.g., when the model is provided as public service) then all the attacks studied in this chapter are relevant to practice and would model different adversarial power.

Our security games model attacks in which the adversary aims to infer (or reconstruct) deletion

of a *random* example from a dataset. Real world adversaries are stronger in the sense that they could have a specific target in mind before making their queries to the online model. Moreover, real world adversaries usually have a lot of auxiliary information (e.g., as those exploited in the attacks on privacy on users in the Netflix challenge [126]) while our attackers have a minimal knowledge about the distribution from which the data is sampled.

Having a diverse set of security games and attacks is analogous to having many different security games and notions in cryptography (such as CPA and CCA security for encryption) to model different attack scenarios. Informally speaking, and at a very high level, one can also think of the very strong deletion compliance of [22] as "UC security" [127], while our other security games/notions model weaker security criteria.

5.2 Related work

Chen et al [62] study a setting similar to our attacks. They show attacks that, given access to two models – one trained on a dataset $\mathcal{S}$ and another on $\mathcal{S} \setminus \{e\}$ – determine whether a given input e' is equal to the deleted item e. This is close to our notion of deletion inference, though not quite the same. They show that their attacks perform much better than plain membership inference on the first model. Our result differs from that of [62] in the following respects:

1. In addition to deletion inference, we also show various kinds of reconstruction attacks in a variety of models with different reconstruction goals.
2. Their attacks are constructed by running sophisticated learning algorithms on the posteriors corresponding to deleted and not deleted samples. While this results in attacks that work quite well, these attacks have little explanatory power – it is not clear what enables them, and it is hard to tell what the best way to prevent them is. Our attacks, on the other hand, make use of simple statistics of the outputs of the models.
3. They show that certain measures like publishing only the predicted label or using differential privacy can stop their attacks from working, but this is far from showing that such measures prevent all possible attacks. In order to prove security against all attacks, a formalization of what entails such security is necessary. We provide formal definitions of privacy and formally build a connection to the deletion compliance framework of [22], which, as corollary, implies that differential privacy can provably prevent any possible deletion inference attack.

The work of Salem et al [61] also studies a related setting. In their case, a model is updated by the addition of new samples, rather than by deletion, and they show attacks that partially reconstruct

either the new sample itself or its label. These attacks are constructed by training generative models on posteriors of various samples from a shadow model. It is possible that their attacks can be used when data is deleted as well. In fact, our attacks can also potentially be adapted to be applied when the data is *added* rather than deleted (but the security game needs to change to formally allow this). They also present a cursory discussion of possible defences against their attacks, suggesting that adding noise to the posteriors or differential privacy might work. The distinction of our result from theirs is along the same lines as above – our attacks are simpler and more transparent, and our formalization allows us to identify strategies for *provable* security against arbitrary attacks by proving the relation of our attacks and the deletion compliance of [22]. On the attack side, our result studies the attack landscape with much more granularity by studying very specific attacks that aim to only reconstruct (or infer) the instances, or their labels, or leverage the knowledge of the instance to better approximate the labels.

Notation of Deletion Fix a learner Lrn, training set $\mathcal{S}$, and model $h \sim \mathsf{Lrn}(\mathcal{S})$. We use $h_{-e} \sim \mathsf{Del}_{\mathcal{S}}(h, e)$ to denote the "ideal" data deletion procedure [55] that outputs $h_{-e} \sim \mathsf{Lrn}(\mathcal{S} \setminus \{e\})$ using fresh randomness for Lrn if needed. (Hence, if $e \notin \mathcal{S}$, then $\mathsf{Del}_{\mathcal{S}}(h, e)$ simply returns a fresh retraining on $\mathcal{S}$.) In general, Del needs to know the training set on which h is trained, or it needs a data structure that keeps some information about $\mathcal{S}$ in addition to h. Whenever $\mathcal{S}$ is clear from the context, we might simply write $h_{-e} \sim \mathsf{Del}(h, e)$.

5.3 Deletion inference attacks

In this section, we describe a framework of attacks on machine unlearning (i.e., machine learning with deletion option) schemes that can *infer* the deleted examples. Such attacks are executed by adversaries who first access the model before deletion followed by having access to the model after deletion. In each case, we will first formally explain our threat model. We also provide theoretical intuition behind our attacks and report experimental findings by implementing those attacks.

Threat model. We define a security game that captures how well an adversary can tell which element is being deleted from the training set. Note that our (default) definition is not aiming to hide the fact that *something* is being deleted, and the only thing we try to hide is *which* element is being deleted. We use a definition that is inspired by how (CPA or CCA) security of encryption schemes are defined through indistinguishability-based security games [128, 129].

Definition 5.3.1 (Deletion inference)**.** Let Lrn be a learner, Del be a deletion mechanism for Lrn, and S_m be a distribution on datasets of size m. The adversary A and the challenger Chal interact as follows.

1. **Sampling the data and revealing the challenges.** Chal picks a dataset $\{z_1, \ldots, z_m\} = \mathcal{S} \sim S_m$ of size m. Chal picks two indices $i \neq j \in [m]$ at random and sends $e_0 = z_i, e_1 = z_j$ to A.
2. **Oracle access before deletion.** Chal trains $h \sim \mathsf{Lrn}(\mathcal{S})$. A is then given *oracle* access to h, and finally instructs moving to the next step.
3. **Random selection and deletion.** Chal picks $b \sim \{0, 1\}$ at random and lets $h_{\mathsf{del}} \sim \mathsf{Del}(h, e_b)$.
4. **Oracle access after deletion.** The adversary A is now given oracle access (only) to h_{del}.
5. **Adversary's guess.** The adversary sends out a bit b' to Chal and wins if $b' = b$.

The scheme $(\mathsf{Lrn}, \mathsf{Del})$ is called ρ *insecure against deletion inference* for data distribution S_m, if there is a PPT adversary A whose success probability in the game above is at least ρ. (Note that achieving $\rho = 1/2$ is trivial.) Now, consider a modified game in which the adversary is given *only the instances* (x_0, x_1) where $e_0 = (x_0, y_0), e_1 = (x_1, y_1)$. We call this game the *instance* deletion inference. If an adversary has success probability at least ρ in the instance deletion inference game, then the scheme $(\mathsf{Lrn}, \mathsf{Del})$ is called ρ insecure against *instance* deletion inference for distribution S_m. Similarly, we define *label* deletion inference, in which only the labels (y_0, y_1) are revealed to the adversary, and ρ-insecurity against such attacks accordingly. To contrast with instance and label deletion inference, we might use *example* deletion inference attack to refer to our default deletion inference attacks. ◇

Note that winning in an instance or label deletion inference game is potentially harder than winning the normal variant (with full examples revealed to the adversary) as the adversary can always ignore the full information given to it. Hence, showing successful instance deletion inference attacks is a stronger (negative) result. We empirically study the power of attacks in all these attack models.

Other variants of Definition 5.3.1. Definition 5.3.1 can be seen as a *weak* definition of privacy for deletion inference. The following list describe variants of Definition 5.3.1 that are either directly weaker, or our attacks can be adapted to in a rather straightforward way.

- **Two-challenges vs. one challenge.** Definition 5.3.1 includes two challenge examples and asks an adversary to find out which one is the actual deleted one. An alternative definition

would only reveal one example to the adversary and asks it to tell if the example is deleted or not.[3]

- **Deletion-revealing vs. deletion-hiding.** Definition 5.3.1 does not aim to hide the fact that a deletion has happened. An alternative definition could even aim to capture hiding the deletion itself by sampling the non-deleted example *outside* the dataset.
- **Random vs. chosen challenges.** Definition 5.3.1 asks the adversary to distinguish between a *random* pair of challenge examples, one of which is deleted. In a stronger attack model, the adversary is allowed to *choose* the challenge examples.
- **Auxiliary information.** Definition 5.3.1 does not explicitly give any extra information about other examples $e_b, b \notin \{i, j\}$ to the adversary, while a real-word adversary might have such knowledge.
- **Multiple deletions vs. one deletion.** Definition 5.3.1 does not allow more than one deletion to happen, while in general users might request multiple deletions to happen over time. In fact, in Section 5.3.3, we use this variant of the attacks to test our attacks on large data sets and compare the result with deletion inference attacks that are obtained by reduction to membership inference.

In Section 5.5, we discuss stronger security definitions that once satisfied would prevent the attack of Definition 5.3.1 and all the variants above as special cases of the Deletion Compliance framework of Gar et al. [22]. In particular, the definitions of this section (including Definition 5.3.1) model *weaker* security guarantees than that of Deletion Compliance framework of [22], which makes our attack results of this section stronger.

Our deletion inference attacks. We propose two variants of attacks: (1) (example) deletion inference attack of Del-Inf-Exm which uses both instances and their true labels, and (2) instance inference attack of Del-Inf-Ins which only uses the instances, without knowing the true labels. (In the next subsection, we also show how to *find* the deleted label, which can be seen as a form of "label reconstruction"and is stronger than label inference attacks.)

Attack Del-Inf-Exm using labeled examples. Our example inference attack Del-Inf-Exm is parameterized by a loss function ℓ and proceeds by first computing the loss for both examples e_0, e_1 on both models h, h_{del}. Then, this attack identifies the deleted example by picking the example that leads to a *larger increase* in its loss when we go from h to h_{del}. The intuition behind our attack is that the examples in the dataset are optimized (to a degree depending on the learning algorithm) to

[3] If one can sample from the set $\mathcal{S}$ the two attack models can be shown to be equivalent using standard hybrid arguments when the adversary's success probability is negligible in security parameter. This is similar to how it is done for CPA/CCA security games in cryptography.

have small loss, while examples outside the dataset are not so. Therefore, once an example goes from inside the dataset to outside, it incurs a larger increase in loss. We now define the attack formally.

Algorithm 1 (Attack Del-Inf-Exm)**.** *The attack is defined with respect to a loss function ℓ. For any example e, we define the* loss increase *of e as: $\delta(e, h, h_{\mathsf{del}}) = \ell(h_{\mathsf{del}}, e) - \ell(h, e)$. The adversary is given two labeled examples $e_0 = (x_0, y_0)$ and $e_1 = (x_1, y_1)$ and also has oracle access to h followed by access to h_{del}. The attack proceeds as follows.*

1. *Query h on both x_0, x_1.*
2. *After getting access to h_{del}, query h_{del} on both x_0, x_1.*
3. *Compute loss increases $\delta(e_0, h, h_{\mathsf{del}})$ and $\delta(e_1, h, h_{\mathsf{del}})$, and let $\alpha = \delta(e_0, h, h_{\mathsf{del}}) - \delta(e_1, h, h_{\mathsf{del}})$.*
4. *Output 0 if $\alpha > 0$, output 1 if $\alpha < 0$, and output a uniformly random bit $b' \in \{0, 1\}$ if $\alpha = 0$.* ◇

Connection to label memorization [124] . At a high level, Del-Inf-Exm can be seen as generalizing the notion of memorization by Feldman [124] from the 0-1 loss to general loss functions. More formally, if we use the 0-1 loss, then for $e \in \mathcal{S}$, the *expected* value $\mathbb{E}_{h_{\mathsf{del}} \sim \mathsf{Del}(h,e)} \delta(e, h, h_{\mathsf{del}})$ would become equal to $\mathsf{mem}(\mathsf{Lrn}, \mathcal{S}, e)$ defined in [124] to measure how much the learner Lrn is memorizing the labels of its training set. , and Using this intuition, our adversary picks the example that is *most memorized* by the model.

The following lemma further formalizes the intuition behind our attack Del-Inf-Exm, so long as the the learning algorithm is the ERM rule.

Lemma 5.3.2. *Let* ERM *be the empirical risk minimization learning rule using a loss function ℓ. Let $h = \mathsf{ERM}(\mathcal{S})$, $h_{-e} \sim \mathsf{Del}(h, e)$ for $e \in \mathcal{S}$, and $\mathcal{S}_{-e} = \mathcal{S} \setminus \{e\}$. Let $\delta_e = \delta(e, h, h_{-e})$, and let $\delta_{-e} = \mathbb{E}_{e' \sim \mathcal{S}_{-e}}[\delta(e', h, h_{-e})]$ be the expected value of loss increase for examples that* remain *in the dataset. Then the following two hold.*

1. *$\delta_{-e} \leq 0$.*
2. *$\delta_e \geq -(m-1) \cdot \delta_{-e}$ where $m = |\mathcal{S}|$. (In particular, by Part 1, it also holds that $\delta_e \geq 0$.)*

Proof. The first item of the lemma holds simply because we are using the ERM rule. Namely, h_{-e} minimizes the empirical loss over $\mathcal{S}_{-e} = \mathcal{S} \setminus \{e\}$. Therefore:

$$\delta_{-e} = \mathrm{Risk}_{\mathcal{S}_{-e}}(h_{-e}) - \mathrm{Risk}_{\mathcal{S}_{-e}}(h) \leq 0.$$

Having proved the first part, the second part also follows due to using the ERM rule. In particular, suppose for sake of contradiction that $\delta_e < -(m-1)\cdot\delta_{-e}$, where $m = |\mathcal{S}|$. Then,

$$\ell(h,e) + (m-1)\cdot \mathrm{Risk}_{\mathcal{S}_{-e}}(h) > \ell(h_{-e},e) + (m-1)\cdot \mathrm{Risk}_{\mathcal{S}_{-e}}(h_{-e}).$$

Then, this implies

$$\begin{aligned}\mathrm{Risk}_{\mathcal{S}}(h) &= \frac{\ell(h,e) + (m-1)\cdot \mathrm{Risk}_{\mathcal{S}_{-e}}(h)}{n} \\ &> \frac{\ell(h_{-e},e) + (m-1)\cdot \mathrm{Risk}_{\mathcal{S}_{-e}}(h_{-e})}{m} = \mathrm{Risk}_{\mathcal{S}}(h_{-e}).\end{aligned}$$

However, the this contradicts that the ERM rule outputs h on training set $\mathcal{S}$. □

Proposition 5.3.2 shows that whenever (1) $\delta_{-e} = \mathbb{E}_{e'\sim\mathcal{S}_{-e}}[\delta_{e'}] < 0$ and (2) $\delta(e', h, h_{-e})$ for $e' \in \mathcal{S}_{-e}$ is concentrated around its mean δ_{-e}, then for a random $e' \in \mathcal{S}_{-e}$, the attack Del-Inf-Exm of Algorithm 1 would likely identify the deleted example correctly. Even though, in general we are not able to prove when these two conditions hold, our experiments confirm that these conditions indeed hold in many natural scenarios, leading to the success of Del-Inf-Exm of Algorithm 1.

Attack Del-Inf-Ins using instances only. We now discuss our attack that does not rely on knowing the true labels y_0, y_1. The intuition is that, even if we do not know the true labels, when an example e is deleted from the dataset, the change in the predicted label for e is likely to be more than that of other examples that stay in the dataset. The reason is that for the remaining examples, the model is still trying to keep their prediction close to their correct value, but this optimization is not done for the deleted example e. Hence, our adversary would pick the candidate example that leads to *larger change* in the output *label* (not necessarily the loss). Hence, the attack is more natural to be used for regression tasks, even though it can also be used for classification if one uses the confidence parameters instead of the final labels.

Algorithm 2 (Attack Del-Inf-Ins)**.** *The attack is parameterized by a distance metric* dis *over* $\mathcal{Y}$ *(e.g.,* $\mathcal{Y} = \mathbb{R}$ *and* $\mathsf{dis}(y_0, y_1) = |y_0 - y_1|$*). The adversary is given two instances* x_0, x_1*, and it has oracle access to* h *followed by* h_{del}*. The attack then proceeds as follows.*

1. *Query the models (in the order of accessing them) to get* $h(x_0)$*,* $h(x_1)$*,* $h_{\mathsf{del}}(x_0)$*,* $h_{\mathsf{del}}(x_1)$*, and let* $\beta = |h(x_0) - h_{\mathsf{del}}(x_0)| - |h(x_1) - h_{\mathsf{del}}(x_1)|$.
2. *Return 0 if* $\beta > 0$*, return 1 if* $\beta < 0$*, and return a random answer in* $\{0,1\}$ *if* $\beta = 0$. ◇

5.3.1 Experiments: Deletion inference attacks on regression

Now we apply our attack Del-Inf-Exm (Algorithm 1) and attack Del-Inf-Ins (Algorithm 2) on multiple regression models including Linear Regression, Lasso regression, SVM Regressor, Decision Tree Regressor, and Neural Network Regressor[4].

Experiment details. Table 5.1 includes the details of all the datasets we used in the deletion inference experiments and also in other experiments later. We use two regression datasets Boston and Diabetes. For training the original model h, we use a random subset with 90% of the dataset.

Here we describe the hyperparameters of target models.

- **MLP:** We use multiple layer perceptron with two hidden layers. We set the size of hidden layers as $(20, 2)$. We use LBFGS as the optimization algorithm to train the model, and we train 200 epochs on the model.
- **SVM:** We use the default SVMRegressor in Scikit-learn. Specifically, we use RBF kernel with $C = 1.0$.
- **Decision tree:** For the decision tree model, we use the default DecisionTreeRegressor in Scikit-learn. Specifically, we use Gini impurity to split the leafs and do not set a limit on the tree size.
- **Linear regression:** We use the default LinearRegression in Scikit-learn.
- **Lasso regression:** We use the Lasso class in Scikit-learn with $\alpha = 0.1$.

The experiment follows the security game of Definition 5.3.1. To ensure the perfect deletion, h_{del} is obtained by a full re-training with the dataset without the deleted example. For the attack Del-Inf-Exm, we use squared loss, which is defined as $\ell(h, (x, y)) = (h(x) - y)^2$. Finally, we repeat the security game of Definition 5.3.1 1000 times and take the average success probability of the adversaries.

Results. The result is shown in Table 5.2. In most cases, our adversary gets more than 90% success probability in the deletion inference.

		No. Samples	No. Features	Label	Predict
Regression	Boston [130]	506	14	Real	The median house price
	Diabetes [131]	442	10	Real	Disease progression
Classification	Iris [132]	150	4	3 types	The type of iris plants
	Wine [133]	178	13	3 types	Wine cultivator
	Breast Cancer [134]	569	30	Binary	Benign/malignant tumors
	1/12MNIST[88]	5000	784	10 types	Digit between 0 to 9
	CIFAR-10 [135]	60000	3072	10 classes	Image classification
	CIFAR-100 [135]	60000	3072	100 classes	Image classification

Table 5.1: Descriptions of the datasets used in deletion inference.

[4]Implementation of the methods are from the python library Scikit-learn.

	Boston		Diabetes	
Learning Method	Del-Inf-Exm	Del-Inf-Ins	Del-Inf-Exm	Del-Inf-Ins
Linear regression	99.8%	99.1%	99.8%	99.3%
SVM	93.9%	89.1%	99.2%	100.0%
Lasso regression	98.8%	97.1%	99.3%	98.3%
Decision tree	100.0%	100.0%	100.0%	100.0%
MLP	80.4%	78.3%	72.2%	72.3%

Table 5.2: Success probabilities of various attacks on regressors for different datasets.

5.3.2 Experiments: Deletion inference attacks on classification

In this experiment, we apply Del-Inf-Exm and Del-Inf-Ins on classification tasks. In our experiments, we use different models, including logistic regression, support vector machine (SVM), Decision tree, random forest, and multi-layer perceptron (MLP). For the hyperparameters, we use

- **MLP:** Similar with regression, we use multiple layer perceptron with two hidden layers. For classification, we set the size of hidden layers as $(20, 10)$. The reason behind is that the output layer of classification tasks have more neurons. We use LBFGS as the optimization algorithm to train the model, and we train 200 epochs on each model.
- **SVM:** We use the default SVMClassifier in Scikit-learn. Specifically, we use the RBF kernel with $C = 1.0$.
- **Decision tree:** We use the default DecisionTreeClassifier in Scikit-learn. Specifically, we use Gini impurity to split the leafs and do not set a limit on the tree size.
- **Random forest:** We use the default RandomForestClassifier in Scikit-learn, which generates 10 trees in the forest. For each tree, its hyperparameter is the same with the decision tree classifier above.
- **Logistic Regression and Linear regression:** We use the default LogsticRegression class from Scikit-learn.=

Experiment details. We use datasets Iris, Wine, Breast Cancer, and 1/12MNIST. (The details of the datasets are shown in Table 5.1.) Similarly to attacks on regression, We pick a random 90% fraction of the dataset to train the model, and we do a full retrain to obtain h_{del}. The difference compared to the case of regression is that the label space $\mathcal{Y}$ is now a finite set. In this experiment, we assume the output of any hypothesis function $h \in \mathcal{H}$ is a multinomial (confidence) distribution over $\mathcal{Y}$, and this probability is available to the adversary. This assumption is realistic as many machine learning applications have the confidence as part of the output [106], and this is also the default setting of many adversarial machine learning researches [4, 52][5]. To formally fit the attack into the

[5]The model in this scenario is still considered as black-box in most machine learning adversarial literature, but someone may argue it is not fully black-box.

framework of Definition 5.3.1, we can extend the set $\mathcal{Y}$ to directly include any such multinomial distribution as the actual output "label".

For Del-Inf-Exm, we use the negative log likelihood loss function $\ell(h,(x,y)) = -\log(\Pr[h(x)=y])$. We then repeat the security game of Definition 5.3.1 1000 times to approximate the winning probability.

Results. We present the result of attacks Del-Inf-Exm and Del-Inf-Ins on three classification datasets in Table 5.3. As anticipated, the success rates Del-Inf-Exm are noticeably larger than those of Del-Inf-Ins.

Datasets →	Iris		Wine		Breast Cancer		1/12 MNIST	
Learning Method ↓	Del-Inf-Exm	Del-Inf-Ins	Del-Inf-Exm	Del-Inf-Ins	Del-Inf-Exm	Del-Inf-Ins	Del-Inf-Exm	Del-Inf-Ins
Logistic Regression	88.3%	86.8%	80.8%	76.1%	69.1%	60.6%	72.9%	56.6%
Decision Tree	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
SVM	70.5%	60.3%	76.9%	66.7%	73.8%	57.3%	72.3%	62.0%
Random Forest	89.2%	89.1%	83.3%	78.1%	89.2%	85.7%	89.9%	84.5%
MLP	92.9%	55.5%	54.2%	51.1%	83.5%	67.7%	62.5%	59.0%

Table 5.3: Success probabilities of the attacks Del-Inf-Exm and Del-Inf-Ins on classifiers for different datasets.

5.3.3 Attacking large models and datasets

In this section, we aim to show that our deletion inference attacks can be scaled to work with large datasets and models. We first formally describe how *deletion* inference attacks can be obtained through black-box reductions to *membership* inference attacks. We then demonstrate the power of our attacks on datasets of the same size as those of [4] and compare the power of our direct deletion inference to doing reduction to the membership inference attack of [4]. We show that using our method can lead to *significantly* stronger results than making a *black-box* use of membership inference attacks.

5.3.4 How to reduce deletion inference to membership inference

One can always reduce the task of deletion inference to the task of membership inference. In particular, if we had a perfect membership inference oracle, we could use it to infer whether a given example is deleted or not by calling the membership inference oracle on the two models h, h_{del}.

Algorithm 3 below shows an intuitive way to reduce deletion inference (DI) to *imperfect* membership inference (MI) in a black-box way. Specifically, suppose the membership inference adversary $M(e,h) \to \{0,1\}$ returns 1 if (it thinks) e is a member of the dataset that is used to obtain the model h. Then, if a deletion inference adversary wants to find out whether e is deleted from the

model h to reach the model h_{del}, it can simply run $M(e, h_{\mathsf{del}})$ and output what it outputs. Note that there is no need to run $M(e, h)$, as the adversary of Definition 5.3.1 is given the promise that both e_0, e_1 *are* members of the initial dataset $\mathcal{S}$. Then the only question is how to combine the answers $M(e_0, h_{\mathsf{del}}), M(e_1, h_{\mathsf{del}})$, which Algorithm 3 decides in a natural way.

Algorithm 3 (From membership to deletion inference)**.** *Given examples $e_0 = (x_0, y_0)$, $e_1 = (x_1, y_1)$ and models h_{del}, the reduction from deletion inference to membership inference proceeds as follows:*

1. *Perform two membership inferences to obtain $b_0 = M(e_0, h_{\mathsf{del}})$ and $b_1 = M(e_1, h_{\mathsf{del}})$.*
2. *Return 0 if $b_0 = 0, b_1 = 1$, return 1 if $b_1 = 1, b_0 = 0$, and return a random bit if $b_0 = b_1$.* ◇

Using confidence probabilities. An alternative reduction to Algorithm 3 can use the *confidence* probabilities of $M(e_0, h_{\mathsf{del}})$ and $M(e_1, h_{\mathsf{del}})$ instead of their final (rounded) values. In this variant, the reduction returns 0 if the confidence difference of $M(e_0, h_{\mathsf{del}}) - M(e_0, h)$ to output zero is more than the confidence difference of $M(e_1, h_{\mathsf{del}}) - M(e_1, h)$ to output zero.

5.3.5 Experiments with large data, and comparison with reduction to membership inference

We now evaluate our deletion inference attacks Del-Inf-Exm and Del-Inf-Ins on large dataset and large neural networks. In our experiment, we use CIFAR-10 and CIFAR-100 datasets [135] as the training dataset, which are standard datasets for the evaluation of image classifiers, especially for deep learning models.

To better compare the success of our attacks with [4] we use a variant attack of Definition 5.3.1 in which *multiple* deletions happen (as explained in one of the variants following Definition 5.3.1). One advantage of this experiment setting is that the attack of [4] needs to train "attack models" for each victim model, and hence having multiple different deletions lead to multiple full training of attack models for [4] which is very expensive to run. However, in the multiple-deletion attack setting, one needs to only train the attack models of [4] twice to compare each execution of our attack with a reduction to [4].

Setting of our attack. The success probability is then calculated by taking the average over 20 rounds of full experiment. In each round of experiment, we first train a deep model with m examples, where m varies from $15,000, 20,000, 25,000$, and $29,540$ ($29,540$ is picked to match the scenario of [4]). We then randomly remove a batch of 100 examples in the training dataset, and train a new model without those 100 examples. As a reference, we pick another 100 random examples that

remains in the dataset. The success probability is calculated over every pairs (in total, 10,000 pairs) of the deleted and reference examples, i.e., one deleted examples and one remaining example is given to the deletion inference adversaries Del-Inf-Exm and Del-Inf-Ins. We then measure the fraction of all pairs in which our adversary correctly predicts the deleted example. We evaluate our results on two deep neural network models: 1. A convolutional neural network that includes two convolutional layers (called smallCNN below), similar to the network used in [4]. 2. VGG-19 network (called VGG below) that has 19 layers in total, which is well-known for its power for image classification tasks.

Baseline settings for comparison. We compare our attacks with reductions to the membership inference attack in [4][6], i.e., reduction with label only and reduction with confidence probabilities.

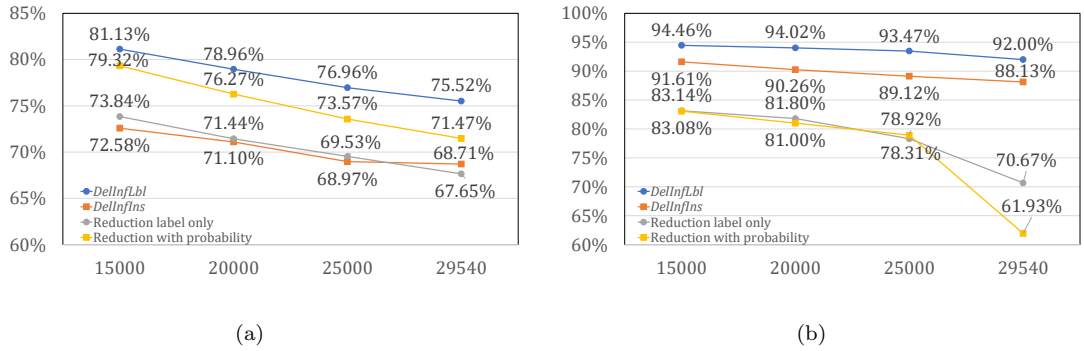


Figure 5.1: Trend of success probabilities of attacks Del-Inf-Exm and Del-Inf-Ins on **smallCNN** models trained with different number of examples are shown; (a) uses dataset CIFAR-10 and (b) uses dataset CIFAR-100 dataset. The success probabilities are also compared with two baseline attacks that are obtained by reductions to the membership inference attack of [4].

Results. In Figure 5.1 and 5.2, we analyze the success probabilities of our deletion inference adversaries Del-Inf-Exm and Del-Inf-Ins on smallCNN model and VGG model. Our attack is able to correctly predict most of the deletions in the deep learning models, even when a batch of examples is deleted at the same time. Furthermore, note that for the membership inference attack of [4] to work, the adversary needs to have the label of the target instance and also make many queries to the target model for training an attack model (or many auxiliary data examples to train a similar model). On the other hand, our attack is extremely simple, and Del-Inf-Ins even does not require the label of the example.

Remark 5.3.3 (About using reduction to MI as baseline)**.** Here we comment on the limitations of membership inference as a baseline attack, as membership inference is not tuned to distinguishing

[6]We implemented [4] attack. [4] reports their membership inference attack achieves 71% success rate on a CNN model with two convolutional layers that is trained with CIFAR-10 dataset with 15,000 random examples. Our implementation of membership inference attack achieves 74% success rate on smallCNN model (which also has two convolutional layers) and 88% success rate on VGG model, which are trained on a subset of CIFAR-10 dataset with 15,000 random examples. The success rate matches the number reported in their work.

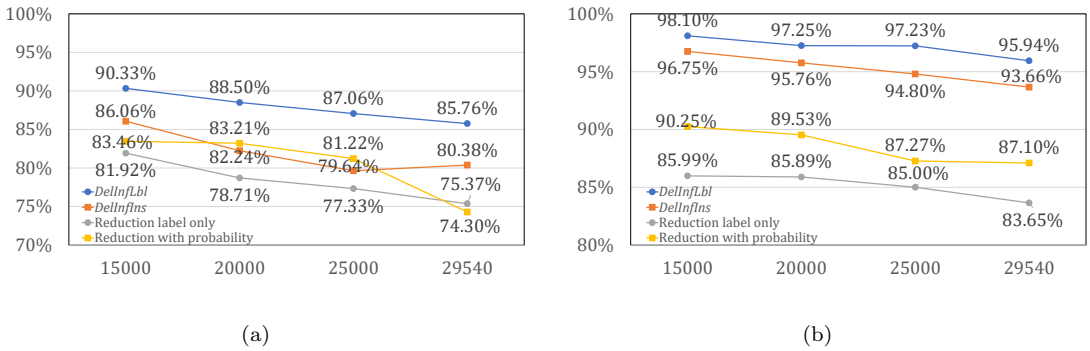


Figure 5.2: Trend of success probabilities of attacks Del-Inf-Exm and Del-Inf-Ins on **VGG** models trained with different number of examples are shown; (a) uses dataset CIFAR-10 and (b) uses dataset CIFAR-100 dataset. The success probabilities are also compared with two baseline attacks that are obtained by reductions to the membership inference attack of [4].

between two points (one of which is guaranteed to be in the training set). Indeed, membership inference attackers only get only one instance as input, while our formalization of deletion inference gets two inputs. However, please note that we compare our deletion inference attackers to *reductions* to membership inference adversaries. The reduction is allowed to call the MI adversary *multiple* times. Indeed our reduction of the previous subsection calls the MI adversary twice, and this change makes the *reduction* to MI (which is a DI adversary itself) powerful enough to be able to win the DI inference game with probability close to 1, so long as its (regular) MI oracle wins its own game with probability close to 1.

5.4 Deletion reconstruction

Section 5.3 focused on attacks that infer which of the two given examples is the deleted one. A more devastating form of attack aims to *reconstruct* the deleted example by querying the two models (before and after deletion). In this section, we show how to design such stronger attacks. We propose two types of reconstruction attacks on the deleted example. The first one focuses on reconstructing the deleted instance, while the second one focuses on reconstructing the deleted label. Both types of attacks follow the same security game which is explained in the definition below.

Definition 5.4.1 (Deletion reconstruction attacks)**.** Let Lrn be a learning algorithm, Del be a deletion mechanism for Lrn, and S_m be a distribution over $(\mathcal{X} \times \mathcal{Y})^m$. Consider the following game played between the adversary A and challenger Chal.

1. **Sampling the data and random selection.** Chal picks a dataset $\{e_1 \dots e_m\} = \mathcal{S} \sim S_m$ of size m. It also chooses $i \sim [m]$ at random.

2. **Oracle access before deletion.** The challenger Chal trains $h \sim \mathsf{Lrn}(\mathcal{S})$. The adversary A is then given *oracle* access to h. At the end of this step, the adversary instructs moving to the next step.

3. **Deletion.** The challenger obtains $h_{-e_i} \sim \mathsf{Del}(h, e_i)$.

4. **Oracle access after deletion.** The adversary A is now given (only) *oracle* access to h_{-e_i}.

5. **Adversary's guess.** Adversary outputs a guess e.

For a similarity metric dis defined on $(\mathcal{X} \times \mathcal{Y})$, the adversary A is called a (ρ, ε)-*successful deletion reconstruction* attack if it holds that $\Pr[\mathsf{dis}(e, e_i) \leq \varepsilon] \geq \rho$. For bounded $\mathsf{dis}(\cdot,\cdot) \in [0,1]$ and an adversary A, we define the *expected accuracy* of A as $1 - \mathbb{E}[\mathsf{dis}(e, e_i)]$. ◇

Deleted instance/label reconstruction attacks. One can use Definition 5.4.1 to capture attacks in which the goal of the adversary is to only (perhaps partially) reconstruct the instance x or the label y. In case of approximating x, we can use a metric distance dis that is only defined over $\mathcal{X}$ and ignores the labels of e and e_i. We refer to such attacks as *deleted instance reconstruction* attacks. Similarly, by using a proper metric distance defined only over $\mathcal{Y}$, we can use Definition 5.4.1 to obtain *deleted label reconstruction* attacks. Finally, to *completely* find e (resp. x or y) we use the 0-1 metric $\mathsf{dis}(e, e') = \mathbb{1}[e \neq e']$ (resp. $\mathbb{1}[x \neq x']$ or $\mathbb{1}[y \neq y']$).

Theorem 5.4.2 (From reconstruction to inference)**.** *Let* Lrn *be a learning algorithm,* Del *be a deletion mechanism for* Lrn*,* dis *be a distance metric over* $(\mathcal{X} \times \mathcal{Y})$*, and* S_m *be a distribution over* $(\mathcal{X} \times \mathcal{Y})^m$*. Suppose there is a* (ρ, ε)*-successful PPT reconstruction adversary against the scheme* $(\mathsf{Lrn}, \mathsf{Del})$*, and* $\Pr[\mathsf{dis}(e_0, e_1) > 2\varepsilon] \geq 1 - \delta$ *where the probability is over sampling* e_0, e_1 *from the sampled dataset* $\mathcal{S} \sim S_m$*.*[7] *Then,* $(\mathsf{Lrn}, \mathsf{Del})$ *is* $(\rho - \delta)$*-insecure against deletion inference over distribution* S_m*.*

Proof of Theorem 5.4.2. We give a polynomial time reduction. In particular, suppose $\boldsymbol{B}$ is a (black-box) adversary that shows the (ρ, ε) insecurity of the scheme $(\mathsf{Lrn}, \mathsf{Del})$ against deletion reconstruction attacks. We design an adversary A against deletion inference (as in Definition 5.3.1) as follows. Given (e_0, e_1) as challenges, first ignore (e_0, e_1) and using oracle access to models h, h_{del}, run $\boldsymbol{B}$ to obtain e as approximation of the deleted example. Output 0 if $\mathsf{dis}(e_0, e) \leq \varepsilon$, else output 1 if $\mathsf{dis}(e_1, e) \leq \varepsilon$, otherwise output uniformly in $\{0, 1\}$.

We now analyze the reduction above. With probability at least ρ over the execution of the attack $\boldsymbol{B}$, it holds that $\mathsf{dis}(e, e_b) \leq \varepsilon$, where e_b is the deleted example. Also, with probability $1 - \delta$ it holds

[7] For example, when S_m consists of m i.i.d. samples from $\boldsymbol{D}$, e_0, e_1 are simply two independent samples from $\boldsymbol{D}$.

that $\mathsf{dis}(e_0, e_1) > 2\varepsilon$. By a union bound, we have that with probability at least $\rho - \delta$ both of the conditions above happen at the same time, in which case the adversary A outputs the correct answer b. □

Due to the theorem above, all the reconstruction attacks below can be seen as strengthening of deletion inference attacks.

5.4.1 Experiments: Deletion reconstruction of instances for nearest neighbor

In this experiment, we consider a classification clustering task in high dimension. The previous work [124, 136] studied the same setting and showed that machine learning models sometimes need to memorize their training set in order to learn with high accuracy. In this setting, we extend the attacks of [124, 136] into two directions to obtain deletion reconstruction attacks: (1) we obtain polynomial time attacks that extract instances rather than proving mutual information between the model and the examples, (2) we show a setting where the extraction is enabled after the *deletion*.

Roadmap and the leakage of the deletion. We develop polynomial-time reconstruction attacks that crucially leverage the deletion operation. However, in order to analyze our attacks, we first *limit* ourselves to the so-called *singleton* setting in which each label appears at most once for an example in the dataset (Section 5.4.2). Focusing on this case allows us to provide theoretical ideas that support our attacks. However, our attacks in the singleton case are also able to extract instances *even* without deletion. Hence, in the singleton case, our attacks can be seen as leakage of the model h itself, *even without deletion*. Note that such attacks can still be used for deletion reconstruction, they do not reflect the *extra leakage* of the deletion operation. Nevertheless, we next experimentally show (Section 5.4.3) that virtually the same polynomial-time attacks succeed even when the labels are not unique on the real world dataset Omniglot. In particular, when we have many repeated labels (perhaps even as neighbor cells), then our simpole attacks do *not* extract the instances from access to either of h, h_{del}, and it is needed to have access to *both* models to find the "vanished" Voronoi cell before extracting the center of the cell.

We now our polynomial-time deletion reconstruction attack for the case of 1-nearest neighbor models. We work with instance space $\mathcal{X} = \{0,1\}^d$.[8] We also assume the learner Lrn runs a 1-nearest neighbor algorithm. Namely for $h = \mathsf{Lrn}(\mathcal{S})$ where $\mathcal{S} = \{(x_1, y_1) \dots (x_m, y_m)\}$, we have $h(x) = y_j$ where $j = \mathrm{argmin}_i\, \mathsf{dis}(x, x_i)$.

[8] We use binary features because it is more general and that other features can also be represented in the form of binary strings.

We propose the following attack Del-Ins-Rec that aims to reconstruct the deleted instance x_i.

Algorithm 4 (Attack Del-Ins-Rec)**.** *Suppose the adversary is given oracle access to h followed by oracle access to h_{del}, along with an auxiliary set of instances $\mathcal{T}$, $|\mathcal{T}| = m$. (For example, $\mathcal{T}$ could simply be m independent samples different from the original training set $\mathcal{S}$.) The attack then proceeds as follows:*

- *For all $x \in \mathcal{T}$ query the model h.*
- *Then for all $x \in \mathcal{T}$, query the model h_{del}.*
- *Create the set of points in the "deleted region": $\mathcal{T}' = \{x \mid h(x) \neq h_{\mathsf{del}}(x), x \in \mathcal{T}\}$.*
- *Return the majority for each coordinate; namely, return $x = (t'_1, \ldots, t'_d)$, where $\forall i \in [d]$,*

$$t'_i = \operatorname*{argmax}_{t \in \{0,1\}} \sum_{(t_1,\ldots,t_d) \in \mathcal{T}'} \mathbb{1}[t_i = t]. \quad \diamond$$

Intuition behind the attack. The intuition behind the attack of Algorithm 4 is that instances like x whose prediction label changes during the deletion process should belong to the Voronoi cell centered at x_i, where (x_i, y_i) is the deleted example. Then the algorithm heuristically assumes that when we pick x at random *conditioned* on changed labels, then they give a pseudo-random distribution inside the Voronoi cell of x_i. In the next section we show that for a natural case called singletons, in which the labels are unique, this intuition carries over formally. We then experimentally verify our attack for the general case (when labels can repeat) on a real data set.

5.4.2 Theoretical Analysis for Uniform Singletons

In this section, we focus on a theoretically natural case to analyze the attack of Algorithm 4. We refer to this case as the *uniform singletons* which is also studied in [124, 136] and is as follows. First, we assume that instances are uniformly distributed in $\{0,1\}^d$, secondly, we assume that the labels are unique (i.e., without loss of generality, the labels $y_1, \ldots, y_m$ are just $1, \ldots, n$). The following lemma shows that in this case, the attack of Algorithm 4 never converges to wrong answers for *any* coordinate of the instances.

Lemma 5.4.3 (Non-negative correlations)**.** *Let $\mathcal{S} = \{x_1, \ldots, x_m\}$ where $\forall i, x_i \in \{0,1\}^d$, and suppose $h(x) = \operatorname{argmin}_i \mathsf{dis}(x, x_i)$, and we break ties by outputting the smallest index i, if multiple nearest neighbors exist. Suppose $C_i = \{x \mid h(x) = i\}$ be the Voronoi cell centered at x_i. Let $x[j]$ be the j'th bit*

of x. Then, for every $i \in [m]$ and every $j \in [d]$, we have

$$\Pr_{x \sim C_i}[x[j] = x_i[j]] \geq \frac{1}{2}.$$

Proof of Lemma 5.4.3. Let $C_i^{j,t} = \{x \in C_i \mid x[j] = t\}$ be the subset of C_i that has t in its j'th coordinate.

We claim that by flipping the j'th bit of every $x \in C^{j,1-x_i[j]}$, we obtain a vector $x' \in C^{j,x_i[j]}$. The reason is as follows. (1) By definition, the j'th bit of x' is indeed $x_i[j]$. (2) It holds that $h(x') = i$, which means $x' \in C_i$. The reason for (2) is that, by flipping the j'th bit of x, x' gets one step *closer* to x_i compared to how far x was from x_i. Therefore, if x_i was the nearest neighbor of x, it would also be the nearest neighbor of x' as well. A boundary case occurs if multiple points are the nearest points of x, but the same tie breaking rule still assigns x_i as the nearest neighbor of x'. Since the mapping from x to x' is injective, it also gives an injective mapping from $\left|C_i^{j,1-x_i[j]}\right|$ to $\left|C_i^{j,x_i[j]}\right|$. This proves that

$$\left|C_i^{j,x_i[j]}\right| \geq \left|C_i^{j,1-x_i[j]}\right|,$$

which is equivalent to $\Pr_{x \sim C_i}[x[j] = x_i[j]] \geq 1/2$. □

5.4.3 Deleted Image Reconstruction for 1-NN

We now show that the simple attack of Algorithm 4 can be used to reconstruct visually recognizable images even when the distribution is not normal and labels are not unique. Hence, we conclude that the actual power of this attack goes beyond the theoretical analysis of the previous section. We use the Omniglot [137] dataset, a symbol classification dataset specialized for few-shot learning. The dataset includes handwritten symbols from multiple languages.

Experiment details. We binarize each pixel of the dataset to remove the noise in gray-scale. The input space is $\mathcal{X} = \{0,1\}^d$, where $d = 11025$ is the number of pixels. We assume the Omniglot dataset is divided into two parts: (1) a training subset which contains 140 symbols from 30 different languages. The languages serve as the class label in the dataset in our experiments,[9] and (2) a fixed test set with another 140 examples from each language which is provided to the adversary as auxiliary information. The learning algorithm Lrn is the 1-nearest neighbor predictor, which for a dataset $\mathcal{S}$ always returns the label (i.e., the language) of the nearest example in the dataset

[9] Note that in the original dataset, the labels reflect the character, but to demonstrate the leakage of *deletion* rather than the mere leakage of datasets alone, we use the labels that represent the languages to increase the frequency of the labels.

$h(x') = \text{argmin}_y\{\mathsf{dis}(x, x') \mid (x, y) \in \mathcal{S}\}$. We use Algorithm 4 as the attack, which simply takes majority on each pixel over the instances that fall into the disagreement region of the two models (before and after deletion). We run the security game of Definition 5.4.1 with 100 random images from the dataset as the deleted image.

Comparison with reconstruction attacks without deletion. As a comparison to further highlight the leakage that happens due to the deletion, we also run a similar reconstruction attack *without* deletion. Suppose for a moment that labels were unique. Then, to reconstruct instance x, the attacker aims to extract the image x from the data set with label y, where y is the label of x. To do that, the reconstruction attacker can run the same exact attack as our deletion reconstruction, as follows: it tests all the images in the test dataset on the model and records every image with label y. The attacker then generate a reconstruction image by taking the majority of the images with label y on every pixel.

When the labels are unique, this reconstruction attack can reconstruct the instances used by a 1-NN just like how our deletion reconstruction attack does and succeeds. However, in our case labels are not unique. Hence, we use this attack as the baseline to show how much our deletion inference attack is in fact extracting information that is the result of the deletion operation.

The result of our deletion reconstruction and the baseline (non-deletion) reconstruction attacks are shown in Figure 5.3. Our deletion reconstruction algorithm reconstructs 40 out of 100 images, due to page limit, 33 of them is shown in Figure 5.3. As is clear from the pictures, the non-deletion reconstruction attack gives no meaningful result in our setting. More concretely, for 35 of the 40 images the label of the deletion reconstruction attack obtains the the correct label when fed back into the nearest neighbor classifier, while only 1 of the images generated by the attack without deletion obtains the correct label.

5.4.4 Known-instance label reconstruction

In this section, we study attacks in which the adversary knows the instance x of the deleted record $e = (x, y)$ and wishes to approximate the true label y by querying the models h and h_{del}. The goal is to beat the correctness of both models for true label y. This means that, in case the two models were supposed to hide the label (perhaps if it was a sensitive information to know very precisely) the data removal process, in this case, clearly goes against the goal of hiding y in its exact form.

Figure 5.3: 33 reconstruction examples on Omniglot dataset. In the figure, Row 1 is the result from the attack without deletion, Row 2 is the result from the deletion reconstruction attack, and Row 3 is the deleted example, which is the target of the attack.

Definition 5.4.4 (Known-instance label reconstruction)**.** This definition is identical to Definition 5.4.1 with the only difference that the adversary is now given x_i (but not y_i) in Step 2 of the attack. ◇

Even though one can define the success criteria of the attackers of Definition 5.4.4 the same way as those of Definition 5.4.1, such attacks are only interesting if they can beat the precision of the answers provided by the two models h, h_{del}, as anyone (including the adversary) could query those models on the point x_i, once x_i is revealed. Our experiments show that such "accuracy boosting" attacks are indeed sometimes possible in the presence of deletion operations.

We propose a simple attack LabelApp in Construction 5 below. LabelApp makes an estimation on y based on the output of the two models.

Algorithm 5 (Attacker LabelApp)**.** *This attack is parameterized by $\lambda > 0$. Given sample x, models h and h_{del}, and a constant λ, the label reconstruction adversary* LabelApp *proceeds as follows:*

1. *Query to obtain $\hat{y} = h(x)$ and $\hat{y}' = h_{\mathsf{del}}(x)$.*
2. *Return $\tilde{y} = \hat{y} + \lambda \cdot (\hat{y} - \hat{y}')$.* ◇

Intuition behind the attack. Similar to the attacks of Section 5.3 (see Proposition 5.3.2), the loss of the deleted sample will increase after the deletion. For simplicity, suppose the loss is mean squared error. In this case, when the learner follows the ERM rule, we have $|\hat{y}' - y|_2 \geq |\hat{y} - y|_2$. Therefore, moving from $\hat{y}'$ towards $\hat{y}$ makes the prediction closer to the actual label y. Consequently, using a small positive λ could lead to less loss. The best value of λ in each different scenario could be empirically estimated by a similar size dataset that is individually sampled by the attacker.

Experiment details. We perform the attack on linear regression models. We test the attack on two classic regression datasets, the Boston Housing Price Dataset [130] and the diabetes dataset [131]. For each dataset, we train the model h with the whole dataset. The adversary returns an approximation $\tilde{y}$. $|\tilde{y} - y|_2$ will denote the distance of the prediction by the adversary, and we use $\min(|h(x) - y|_2, |h_{\mathsf{del}}(x) - y|_2)$ as the baseline value to compare the quality of adversary's prediction.

Results. We calculate the average distance of $\tilde{y}_i$ and y_i with different λ values. Our results (in Table 5.4) show that there exists a λ value for each dataset, such that can reduce the the estimated loss by around 70%.

	Best λ	Models	Adversary	%
Boston	17.5	21.897	7.149	30%
Diabetes	30	2859.7	829.8	28%

Table 5.4: Result of the label reconstruction Attack on Logistic Regression. The column Models lists the average of the minimum distance of the predictions of the two models h, h_{del}. The column Adversary lists the average distance of the prediction of the adversary and the real prediction, and the percentage shows the percentage of the improvement in the prediction compared with the better of the predictions of the two models h, h_{del}.

5.5 Weak deletion compliance

In Sections 5.3 and 5.4, we studied *attacks* on data privacy under data deletion. The definitions of those sections provide *weak* guarantees on what adversary cannot do, hence they are suitable for stronger *negative* results. In this section, we investigate the other side; namely, positive results that can prevent attacks of Sections 5.3 and 5.4 and provide strong guarantees about what adversary can(not) learn about the data that is being updated through deletion requests. In particular, we observe that the deletion compliance definition of Garg, Goldwasser and Vasudevan [22] would prevent attacks of Sections 5.3 and 5.4. More precisely, we show that even a *weaker* variant of the [22] definition would prevent the attacks of Sections 5.3 and 5.4.

Components of deletion compliance definition. The "deletion compliance" framework of Garg, Goldwasser and Vasudevan [22] provides an intuitive way of capturing data deletion guarantees in general systems that collect and process data. This framework models the world by three interacting parties – the data collector DatCol, the deletion-requester DelReq, and the environment Env. All components are the same as those of [22], however, we will work with a modified DelReq and a different indistinguishability guarantee.

- *Data collector (learner)* DatCol represents the algorithm that collects the records (training examples) and processes data according to a (learning) mechanism. For example DelReq might accept up to m data storage requests and up to b data deletion requests.

- *Deletion requester (user)* DelReq is a special honest user who only stores two particular examples e_0, e_1 and will delete one of them later. The timing of such requests are stated below. In the original [22], the deletion requester just stores *one* record e and delete it, or that it might never store e in the first place. At a high level, their DelReq is designed so that one can define privacy that even hides the deletion itself, while our variant is designed for a weaker definition that does not hide the deletion itself.

- *Environment (adversary)* Env models the "rest of users" who might not be honest and who are interested in finding out what DelReq is deleting. The interaction between Env and DatCol, DelReq is defined by the interfaces of DatCol, DelReq.

Interaction of the components. We let $\mathcal{U}$ model a *universe of records*. For example, $\mathcal{U} = \text{Supp}(D)$ for a distribution over labeled examples $\mathcal{X} \times \mathcal{Y}$. We now describe the restrictions on how the components interact with each other. Other than the below-mentioned restrictions, the parties run in PPT.

- DatCol accepts instructions $\mathcal{A}dd(e), \mathsf{Del}(e), \mathsf{Eval}(x)$. The interpretation of these instructions are as follows. $\mathcal{A}dd(e)$ adds the record $e \in \mathcal{U}$ to the set of records stored at DatCol. $\mathsf{Del}(e)$ removes e from the set stored by the data collector, and $\mathsf{Eval}(x)$ returns the evaluation of the "current model" stored by DatCol (which is the result of learning over the set stored at DatCol) on x and returns the answer.

- As in [22], we also require that only Env can send messages to DelReq. At some point in the execution of the system Env sends DelReq the following messages, which is followed by messages from DelReq to DatCol as described below.

  1. $(\mathcal{A}dd, e_0, e_1)$: DelReq sends $\mathcal{A}dd(e_0), \mathcal{A}dd(e_1)$ to DatCol.
  2. Del: DelReq will send $\mathsf{Del}(e)$ to DatCol where $e \in \{e_0, e_1\}$. By DelReq_b we refer to the instantiation of DelReq that sends $\mathsf{Del}(e_b)$ to DatCol.

Weak deletion compliance. For our purposes, we consider a different weaker definition (compared to that of [22]) that still captures all attacks of Section 5.3 and 5.4. To start, we define two worlds, World 0 and World 1, corresponding to the instantiation of DelReq by DelReq_0 and DelReq_1.

Definition 5.5.1 (Weak deletion compliance)**.** Let the interactive algorithms $\mathsf{DatCol}, \mathsf{Env}, \mathsf{DelReq}$ be, in order, the data collector, the environment, and the deletion requester (interactive) algorithms limited to interact as described above. We call DatCol ε *deletion compliant*, if no PPT Env can detect whether it is in World 0 (with DelReq_0) or World 1 (with DelReq_1) with advantage more than ε. If this holds under the restriction that Env makes at most $(b-1)$ deletion requests during the execution, then DatCol is said to be ε-weak deletion-compliant for up to b deletions ◇

Comparison with [22]. The key differences between our Definition 5.5.1 and that of [22] are as follows. In each case, we state the property of our definition in contrast to that of [22].

- **Hiding the state of DatCol from adversary.** The definition of [22] focuses on scenarios where the data collector's state might be revealed at some point in the future (e.g., due to a subpoena). However, in this work we focus on hiding the information that is *leaked* from the data collector (about deleted record) through interaction with the adversary.
- **Not aiming to hide the deletion itself.** Whereas plain deletion-compliance asks that deletion make the world look as though the deleted data were never present in the first place, here we only ask that it not be revealed *which* record was deleted. For instance, a data collector that is weak deletion-compliant might still reveal the number of deletions it has processed, as long as the data that is deleted is not revealed. While weaker than deletion-compliance definition of [22], our notion is fit for hiding the deleted record among the records in the training set, and still giving a more general and stronger definition than Definition 5.3.1.

We now formally discuss why Definition 5.5.1 captures the attacks of Section 5.3 and 5.4. Recall that Definition 5.3.1 was already shown in Theorem 5.4.2 to be a stronger notion than instance and label reconstruction attacks (Definition 5.4.1). Hence, we just need to show that Definition 5.5.1 is stronger than Definition 5.3.1.

Theorem 5.5.2 (Deletion inference from compliance)**.** *Let* Lrn *be a learner,* Del *be a deletion mechanism for* Lrn*,* D *be a distribution over labeled examples, and* $\mathcal{U} = \mathrm{Supp}(D)$ *be the universe of records. The data collector* DatCol *answers queries as follows.*

1. DatCol *does not respond any* Del *or* Eval *queries till receiving* m $\mathcal{A}dd(\cdot)$ *queries, which we refer to as* $\mathcal{S}$*.*
2. DatCol *permutes* $\mathcal{S}$ *and gets* $h \sim \mathsf{Lrn}(\mathcal{S})$*.*
3. *Then it answers* $\mathsf{Eval}(e) = h(e)$ *queries arbitrarily.*

4. *Then it accepts* one $\mathsf{Del}(e)$*, and lets* $h_{-e} = \mathsf{Del}(h, e)$.

5. *Then it continues answering* $\mathsf{Eval}(e) = h(e)$ *queries.*

If DatCol *is* $(2\varepsilon - 1)$*-deletion compliant (as in Definition 5.5.1) against PPT adversaries with oracle access to* $\boldsymbol{D}$*, then the scheme* $(\mathsf{Lrn}, \mathsf{Del})$ *is* ε*-secure against deletion inference (as in Definition 5.3.1).*

Proof of Theorem 5.5.2. We give a proof by reduction. Suppose A breaks the membership inference security game of Definition 5.3.1 with probability $(1 + \varepsilon)/2$. We construct an environment Env that ε-distinguishes DelReq_0 from DelReq_0 with advantage ε that proceeds as follows:

1. Env plays the role of the challenger from Definition 5.3.1 and picks a data set $\{e_1, \ldots, e_m\} = \mathcal{S} \sim S_m$ of size m. Env passes this to the DatCol and picks $i \neq j \in [m]$ at random as the challenge records.
2. Next, Env instantiates A and provides it with the records e_i, e_j and *oracle* access to h (through DatCol). At the end of this step, the adversary instructs moving to the next step.
3. Env passes (e_i, e_j) to DelReq (which will then request the deletion of one of the two records).
4. Env actives the A again and it is again provided *oracle* access to h (through DatCol). At the end of this step, the adversary's output is included in the output of the environment.

The view of the adversary A in the above experiment is identical to its view as part of Definition 5.3.1. Thus, the output of A will correctly (with probability greater than ϵ) identify whether DelReq requests the deletion of record e_i or record e_j. This allows us to conclude that the view of the Env changes depending of whether DelReq requests deletion of e_i or e_j. □

Using the same three components described in Section 5.5 (with a different DelReq), [22] defines the notion of *deletion-compliance*. Here the ideal world is the same as the real world in all respects except that DelReq is not allowed to communicate with DatCol as represented in Fig. 5.4. (The restriction of DelReq not being able to send messages to Env was imposed in order for this ideal world to be well-defined, by excluding cases where Env sends to DatCol messages that depend non-trivially on DelReq's records.) [22] calls DatCol to be ε-deletion-compliant if, for any Env and DelReq, the joint distributions of the state of DatCol and view of Env in the real and ideal world are ε-close in the statistical distance, denoted by notation $\approx_\varepsilon$. That is,

$$(state_D, view_E) \approx_\varepsilon (state_D^{\mathsf{Ideal}}, view_E^{\mathsf{Ideal}}).$$

The above (strong) definition from [22] captures the intuition that a system is deletion-compliant if the state of the world after its deleting a record is similar to what it would have been if the record

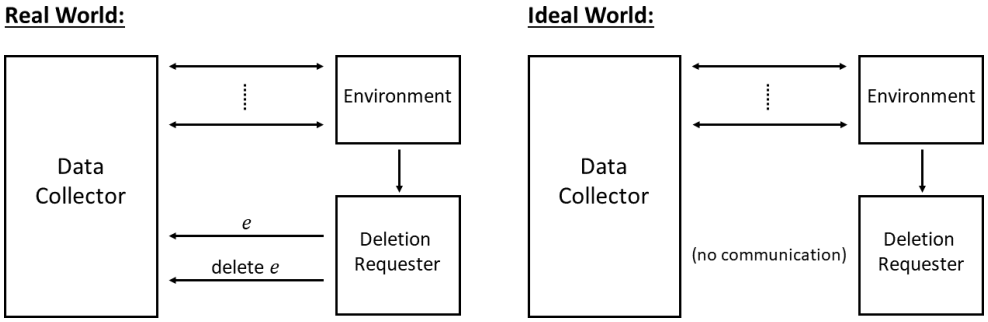


Figure 5.4: The real and ideal worlds for (strong) deletion compliance

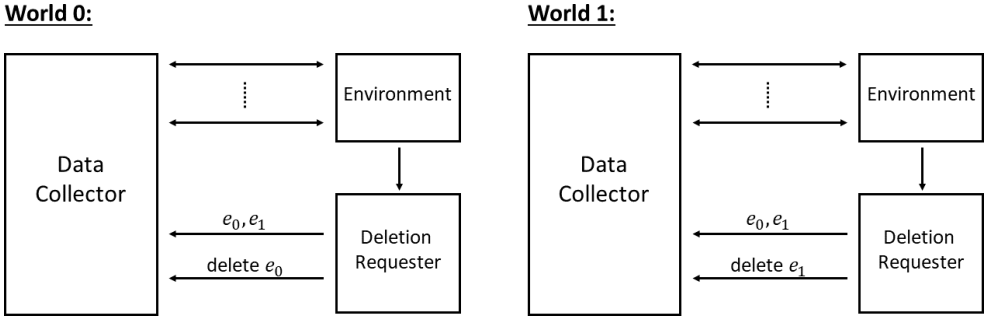


Figure 5.5: The worlds for weak deletion-compliance

had never been part of the system in the first place. Note that this requirement of ε-closeness in statistical distance is more relaxed than the kind of closeness of distributions required by differential privacy, and so DP can be used to satisfy these requirements. [22] showed how to obtain their strong deletion compliance based on differentially private mechanisms.

To contrast with Figure 5.4, in Figure 5.5 we have depicted the more symmetric worlds that are behind our Definition 5.5.1. In particular, Definition 5.5.1 requires that no PPT Env can distinguish between World 0 and World 1 of Figure 5.5 by more than advantage ε.

Chapter 6

Conclusion

In this dissertation we study the security and privacy implications of both malicious and benign modifications.

6.1 Security implications of malicious modifications

To analyze the security implication of poisoning attacks, we give theoretical analyses from both the viewpoint of the learner and the view of the adversary.

- From the learner's viewpoint, we formally define learnability under instance-targeted poisoning attacks. We show that when the budget of the adversary scales sub-linearly with the sample complexity, (improper) PAC learnability and certification are achievable. In contrast, when the adversary's budget grows linearly with the sample complexity, the adversary can potentially drive up the expected 0-1 loss to one for any PAC learner.

  Still there are many open leads here. One main thing is to connect this result with specific learning algorithms such as deep neural networks. In real practice, the learner Lrn is often not a PAC learner. In that case, one may ask how to robustly learn from the dataset.

- From the adversary's viewpoint, we show that adversary can amplify a vulnerability of the machine learner. Specifically, given any small budget b and let the probability that vulnerability happens be μ, the generic amplification attack can achieve a bias that is at least $\Omega(\mu b/\sqrt{m})$.

  We show a generic result here. For specific learners and datasets, it is possible to guarantee a larger bias for amplification attacks, leaving room for future studies.

6.2 Privacy implications of benign modifications

To study the privacy implication of benign modifications, we demonstrate privacy leakage of (perfect) machine unlearning. We presented various attacks that infer or extract information about the deleted examples for a machine learning scheme when unlearning happens. Our various leakage attacks demonstrate that the unlearning operation could come at an extra cost in privacy loss. In particular, we introduced *deletion inference* and *deletion reconstruction* attacks that are reminiscent of membership and data reconstruction attacks in the traditional setting where no updates are applied to the model. Many intriguing questions remain open. Most prominently, it remains open to find *efficient* learning methods with high accuracy that allow deletion with provable privacy guarantees. Since differential privacy comes at the cost of utility loss, we hope to find an efficient and effective algorithm that satisfies weak deletion compliance, which will solve the privacy leakage with minimal cost.

Bibliography

[1] Ji Gao, Amin Karbasi, and Mohammad Mahmoody. Learning and certification under instance-targeted poisoning, 2021.

[2] Omid Etesami, Ji Gao, Saeed Mahloujifar, and Mohammad Mahmoody. Polynomial-time targeted attacks on coin tossing for any number of corruptions. In *Theory of Cryptography Conference*, pages 718–750. Springer, 2021.

[3] Ji Gao, Sanjam Garg, Mohammad Mahmoody, and Prashant Nalini Vasudevan. Deletion inference, reconstruction, and compliance in machine (un) learning. *arXiv preprint arXiv:2202.03460*, 2022.

[4] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 3–18. IEEE, 2017.

[5] Claudine Badue, Rânik Guidolini, Raphael Vivacqua Carneiro, Pedro Azevedo, Vinicius Brito Cardoso, Avelino Forechi, Luan Jesus, Rodrigo Berriel, Thiago Meireles Paixão, Filipe Mutz, et al. Self-driving cars: A survey. *Expert Systems with Applications*, page 113816, 2020.

[6] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[7] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657, 2015.

[8] Tania Sourdin. Judge v. robot: Artificial intelligence and judicial decision-making. *UNSWLJ*, 41:1114, 2018.

[9] Burr Settles. Active learning literature survey. 2009.

[10] Chris Jay Hoofnagle, Bart van der Sloot, and Frederik Zuiderveen Borgesius. The european union general data protection regulation: what it is and what it means. *Information & Communications Technology Law*, 28(1):65–98, 2019.

[11] Lydia de la Torre. A guide to the California consumer privacy act of 2018. *Available at SSRN 3275571*, 2018.

[12] 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.

[13] Brendan McMahan and Daniel Ramage. Federated learning: Collaborative machine learning without centralized training data. *Google Research Blog*, 2017.

[14] 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.

[15] Nicolas Papernot, Patrick McDaniel, Arunesh Sinha, and Michael Wellman. Towards the science of security and privacy in machine learning. *arXiv preprint arXiv:1611.03814*, 2016.

[16] Ilias Diakonikolas and Daniel M Kane. Recent advances in algorithmic high-dimensional robust statistics. *arXiv preprint arXiv:1911.05911*, 2019.

[17] Micah Goldblum, Dimitris Tsipras, Chulin Xie, Xinyun Chen, Avi Schwarzschild, Dawn Song, Aleksander Madry, Bo Li, and Tom Goldstein. Data security for machine learning: Data poisoning, backdoor attacks, and defenses. *arXiv preprint arXiv:2012.10544*, 2020.

[18] Nader H. Bshouty, Nadav Eiron, and Eyal Kushilevitz. Pac learning with nasty noise. *Theoretical Computer Science*, 288(2):255–275, 2002.

[19] Michael Veale, Reuben Binns, and Lilian Edwards. Algorithms that remember: Model inversion attacks and data protection law. *CoRR*, abs/1807.04644, 2018.

[20] Aloni Cohen and Kobbi Nissim. Towards formalizing the gdpr's notion of singling out. *CoRR*, abs/1904.06009, 2019.

[21] Kobbi Nissim, Aaron Bembenek, Alexandra Wood, Mark Bun, Marco Gaboardi, Urs Gasser, David R O'Brien, Thomas Steinke, and Salil Vadhan. Bridging the gap between computer science and legal approaches to privacy. *Harv. JL & Tech.*, 31:687, 2017.

[22] Sanjam Garg, Shafi Goldwasser, and Prashant Nalini Vasudevan. Formalizing data deletion in the context of the right to be forgotten. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part II*, volume 12106 of *Lecture Notes in Computer Science*, pages 373–402. Springer, 2020.

[23] 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.

[24] Shike Mei and Xiaojin Zhu. Using machine teaching to identify optimal training-set attacks on machine learners. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[25] Minghong Fang, Neil Zhenqiang Gong, and Jia Liu. Influence function based data poisoning attacks to top-n recommender systems. In *Proceedings of The Web Conference 2020*, pages 3019–3025, 2020.

[26] Binghui Wang and Neil Zhenqiang Gong. Attacking graph-based classification via manipulating the graph structure. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 2023–2040, 2019.

[27] Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Gong. Local model poisoning attacks to byzantine-robust federated learning. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 1605–1622, 2020.

[28] Shaofeng Li, Shiqing Ma, Minhui Xue, and Benjamin Zi Hao Zhao. Deep learning backdoors. *arXiv preprint arXiv:2007.08273*, 2020.

[29] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017.

[30] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*, 2017.

[31] Alexander Turner, Dimitris Tsipras, and Aleksander Madry. Clean-label backdoor attacks. 2018.

[32] Mauro Barni, Kassem Kallas, and Benedetta Tondi. A new backdoor attack in cnns by training set corruption without label poisoning. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 101–105. IEEE, 2019.

[33] Ahmed Salem, Rui Wen, Michael Backes, Shiqing Ma, and Yang Zhang. Dynamic backdoor attacks against machine learning models. *arXiv preprint arXiv:2003.03675*, 2020.

[34] Chulin Xie, Keli Huang, Pin-Yu Chen, and Bo Li. Dba: Distributed backdoor attacks against federated learning. In *International Conference on Learning Representations*, 2019.

[35] Jacob Steinhardt, Pang Wei W Koh, and Percy S Liang. Certified defenses for data poisoning attacks. In *Advances in neural information processing systems*, pages 3517–3529, 2017.

[36] Elan Rosenfeld, Ezra Winston, Pradeep Ravikumar, and Zico Kolter. Certified robustness to label-flipping attacks via randomized smoothing. In *International Conference on Machine Learning*, pages 8230–8241. PMLR, 2020.

[37] Alexander Levine and Soheil Feizi. Deep partition aggregation: Provable defenses against general poisoning attacks. In *International Conference on Learning Representations*, 2021.

[38] Ruoxin Chen, Jie Li, Chentao Wu, Bin Sheng, and Ping Li. A framework of randomized selection based certified defenses against data poisoning attacks, 2020.

[39] Maurice Weber, Xiaojun Xu, Bojan Karlas, Ce Zhang, and Bo Li. Rab: Provable robustness against backdoor attacks. *arXiv preprint arXiv:2003.08904*, 2020.

[40] Jinyuan Jia, Xiaoyu Cao, and Neil Zhenqiang Gong. Intrinsic certified robustness of bagging against data poisoning attacks. *arXiv preprint arXiv:2008.04495*, 2020.

[41] Irit Dinur and Kobbi Nissim. Revealing information while preserving privacy. In *Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 202–210, 2003.

[42] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In Shai Halevi and Tal Rabin, editors, *TCC 2006: 3rd Theory of Cryptography Conference*, volume 3876 of *Lecture Notes in Computer Science*, pages 265–284, New York, NY, USA, March 4–7, 2006. Springer, Heidelberg, Germany.

[43] Cynthia Dwork. Differential privacy: A survey of results. In *International conference on theory and applications of models of computation*, pages 1–19. Springer, 2008.

[44] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 308–318, 2016.

[45] Shuang Song, Kamalika Chaudhuri, and Anand D Sarwate. Stochastic gradient descent with differentially private updates. In *2013 IEEE Global Conference on Signal and Information Processing*, pages 245–248. IEEE, 2013.

[46] Cynthia Dwork, Kunal Talwar, Abhradeep Thakurta, and Li Zhang. Analyze gauss: optimal bounds for privacy-preserving principal component analysis. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 11–20, 2014.

[47] Raef Bassily, Adam Smith, and Abhradeep Thakurta. Private empirical risk minimization: Efficient algorithms and tight error bounds. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pages 464–473. IEEE, 2014.

[48] Ashish Dandekar, Debabrota Basu, and Stéphane Bressan. Differential privacy for regularised linear regression. In *International Conference on Database and Expert Systems Applications*, pages 483–491. Springer, 2018.

[49] Or Sheffet. Old techniques in differentially private linear regression. In *Algorithmic Learning Theory*, pages 789–827, 2019.

[50] Kunal Talwar, Abhradeep Guha Thakurta, and Li Zhang. Nearly optimal private lasso. In *Advances in Neural Information Processing Systems*, pages 3025–3033, 2015.

[51] Michael Backes, Pascal Berrang, Mathias Humbert, and Praveen Manoharan. Membership privacy in microrna-based studies. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 319–330, 2016.

[52] Yunhui Long, Vincent Bindschaedler, Lei Wang, Diyue Bu, Xiaofeng Wang, Haixu Tang, Carl A. Gunter, and Kai Chen. Understanding membership inferences on well-generalized learning models. *arXiv preprint arXiv:1802.04889*, 2018.

[53] Da Yu, Huishuai Zhang, Wei Chen, Jian Yin, and Tie-Yan Liu. Membership inference with privately augmented data endorses the benign while suppresses the adversary. *arXiv preprint arXiv:2007.10567*, 2020.

[54] Yinzhi Cao and Junfeng Yang. Towards making systems forget with machine unlearning. In *2015 IEEE Symposium on Security and Privacy*, pages 463–480. IEEE, 2015.

[55] Antonio Ginart, Melody Guan, Gregory Valiant, and James Y. Zou. Making ai forget you: Data deletion in machine learning. In *Advances in Neural Information Processing Systems*, pages 3513–3526, 2019.

[56] Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Eternal sunshine of the spotless net: Selective forgetting in deep neural networks. *arXiv preprint arXiv:1911.04933*, 2019.

[57] Lucas Bourtoule, Varun Chandrasekaran, Christopher Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. *arXiv preprint arXiv:1912.03817*, 2019.

[58] Zachary Izzo, Mary Anne Smart, Kamalika Chaudhuri, and James Zou. Approximate data deletion from machine learning models: Algorithms and evaluations. *arXiv preprint arXiv:2002.10077*, 2020.

[59] Chuan Guo, Tom Goldstein, Awni Hannun, and Laurens van der Maaten. Certified data removal from machine learning models. *arXiv preprint arXiv:1911.03030*, 2019.

[60] Seth Neel, Aaron Roth, and Saeed Sharifi-Malvajerdi. Descent-to-delete: Gradient-based methods for machine unlearning. *arXiv preprint arXiv:2007.02923*, 2020.

[61] Ahmed Salem, Apratim Bhattacharya, Michael Backes, Mario Fritz, and Yang Zhang. Updates-leak: Data set inference and reconstruction attacks in online learning. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 1291–1308, 2020.

[62] Min Chen, Zhikun Zhang, Tianhao Wang, Michael Backes, Mathias Humbert, and Yang Zhang. When machine unlearning jeopardizes privacy. *arXiv preprint arXiv:2005.02205*, 2020.

[63] Giuseppe Ateniese, Randal Burns, Reza Curtmola, Joseph Herring, Lea Kissner, Zachary Peterson, and Dawn Song. Provable data possession at untrusted stores. In *Proceedings of the 14th ACM conference on Computer and communications security*, pages 598–609, 2007.

[64] Congzheng Song and Vitaly Shmatikov. Auditing data provenance in text-generation models. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 196–206, 2019.

[65] Avrim Blum, Steve Hanneke, Jian Qian, and Han Shao. obust learning under clean-label attack. In *Conference on Learning Theory*, 2021.

[66] Saeed Mahloujifar and Mohammad Mahmoody. Blockwise p-tampering attacks on cryptographic primitives, extractors, and learners. In *Theory of Cryptography Conference*, pages 245–279. Springer, 2017.

[67] Saeed Mahloujifar, Dimitrios I. Diochnos, and Mohammad Mahmoody. Learning under p-tampering attacks. In *Algorithmic Learning Theory*, pages 572–596. PMLR, 2018.

[68] Saeed Mahloujifar, Mohammad Mahmoody, and Ameer Mohammed. Universal multi-party poisoning attacks. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pages 4274–4283, 2019.

[69] Saeed Mahloujifar and Mohammad Mahmoody. Can adversarially robust learning leverage-computational hardness? In *Algorithmic Learning Theory*, pages 581–609. PMLR, 2019.

[70] Saeed Mahloujifar, Dimitrios I. Diochnos, and Mohammad Mahmoody. The curse of concentration in robust learning: Evasion and poisoning attacks from concentration of measure. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4536–4543, 2019.

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

[72] 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.

[73] 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. *arXiv preprint arXiv:1804.00792*, 2018.

[74] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1885–1894. PMLR, 06–11 Aug 2017.

[75] Leslie G. Valiant. Learning disjunction of conjunctions. In *IJCAI*, pages 560–566, 1985.

[76] Michael Kearns and Ming Li. Learning in the presence of malicious errors. *SIAM Journal on Computing*, 22(4):807–837, 1993.

[77] Robert H. Sloan. Four Types of Noise in Data for PAC Learning. *Information Processing Letters*, 54(3):157–162, 1995.

[78] 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.

[79] 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.

[80] Yujie Ji, Xinyang Zhang, and Ting Wang. Backdoor attacks against learning systems. In *2017 IEEE Conference on Communications and Network Security (CNS)*, pages 1–9. IEEE, 2017.

[81] Bryant Chen, Wilka Carvalho, Nathalie Baracaldo, Heiko Ludwig, Benjamin Edwards, Taesung Lee, Ian Molloy, and Biplav Srivastava. Detecting backdoor attacks on deep neural networks by activation clustering. *arXiv preprint arXiv:1811.03728*, 2018.

[82] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y Zhao. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 707–723. IEEE, 2019.

[83] Alexander Turner, Dimitris Tsipras, and Aleksander Madry. Label-consistent backdoor attacks. *arXiv preprint arXiv:1912.02771*, 2019.

[84] Leslie G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

[85] Omar Montasser, Steve Hanneke, and Nathan Srebro. Vc classes are adversarially robustly learnable, but only improperly. In *Conference on Learning Theory*, pages 2512–2530. PMLR, 2019.

[86] Jiri Matousek. *Lectures on discrete geometry*, volume 212. Springer Science & Business Media, 2013.

[87] Dimitrios I. Diochnos, Saeed Mahloujifar, and Mohammad Mahmoody. Adversarial risk and robustness: general definitions and implications for the uniform distribution. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 10380–10389, 2018.

[88] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[89] 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.

[90] 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 2019. PMLR.

[91] Saeed Mahloujifar, Dimitrios I Diochnos, and Mohammad Mahmoody. Learning under p-Tampering Attacks. In *ALT*, pages 572–596, 2018.

[92] Manuel Blum. How to exchange (secret) keys. *ACM Transactions on Computer Systems*, 1:175–193, 1984.

[93] 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, 2015.

[94] 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*, 2018.

[95] David Lichtenstein, Nathan Linial, and Michael Saks. Some extremal problems arising from discrete control processes. *Combinatorica*, 9(3):269–287, 1989.

[96] Iftach Haitner and Yonatan Karidi-Heller. A tight lower bound on adaptively secure full-information coin flip. *Foundations of Computer Science (FOCS), IEEE 61st Annual Symposium on*, 2020.

[97] Hamidreza Amini Khorasgani, Hemanta K Maji, and Tamalika Mukherjee. Estimating gaps in martingales and applications to coin-tossing: constructions and hardness. In *Theory of Cryptography Conference*, pages 333–355. Springer, 2019.

[98] Hamidreza Amini Khorasgani, Hemanta K. Maji, and Mingyuan Wang. Optimally-secure coin-tossing against a byzantine adversary. Cryptology ePrint Archive, Report 2020/519, 2020. `https://eprint.iacr.org/2020/519`.

[99] Richard Cleve and Russell Impagliazzo. Martingales, collective coin flipping and discrete control processes. *Manuscript*, 1993.

[100] Colin McDiarmid. On the method of bounded differences. *Surveys in combinatorics*, 141(1):148–188, 1989.

[101] 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.

[102] D Amir and VD Milman. Unconditional and symmetric sets in n-dimensional normed spaces. *Israel Journal of Mathematics*, 37(1-2):3–20, 1980.

[103] Vitali D Milman and Gideon Schechtman. *Asymptotic theory of finite dimensional normed spaces*, volume 1200. Springer Verlag, 1986.

[104] Grigorii Aleksandrovich Margulis. Probabilistic characteristics of graphs with large connectivity. *Problemy peredachi informatsii*, 10(2):101–108, 1974.

[105] Michel Habib, Colin McDiarmid, Jorge Ramirez-Alfonsin, and Bruce Reed. *Probabilistic methods for algorithmic discrete mathematics*, volume 16. Springer Science & Business Media, 2013.

[106] Mauro Ribeiro, Katarina Grolinger, and Miriam AM Capretz. Mlaas: Machine learning as a service. In *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, pages 896–902. IEEE, 2015.

[107] Yunhui Long, Vincent Bindschaedler, and Carl A. Gunter. Towards measuring membership privacy. *arXiv preprint arXiv:1712.09136*, 2017.

[108] Ahmed Salem, Yang Zhang, Mathias Humbert, Mario Fritz, and Michael Backes. Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models. In *Network and Distributed Systems Security Symposium 2019*. Internet Society, 2019.

[109] Ziqi Yang, Jiyi Zhang, Ee-Chien Chang, and Zhenkai Liang. Neural network inversion in adversarial setting via background knowledge alignment. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 225–240, 2019.

[110] Christopher A Choquette Choo, Florian Tramer, Nicholas Carlini, and Nicolas Papernot. Label-only membership inference attacks. *arXiv preprint arXiv:2007.14321*, 2020.

[111] Zheng Li and Yang Zhang. Label-leaks: Membership inference attack with label. *arXiv preprint arXiv:2007.15528*, 2020.

[112] Santiago Zanella-Béguelin, Lukas Wutschitz, Shruti Tople, Victor Ruehle, Andrew Paverd, Olga Ohrimenko, Boris Köpf, and Marc Brockschmidt. Analyzing information leakage of updates to natural language models. In *ACM Conference on Computer and Communication Security (CCS)*. ACM, ACM, November 2020.

[113] Bargav Jayaraman, Lingxiao Wang, David Evans, and Quanquan Gu. Revisiting membership inference under realistic assumptions. *arXiv preprint arXiv:2005.10881*, 2020.

[114] Nicholas Carlini, Sanjam Garg, Somesh Jha, Saeed Mahloujifar, Mohammad Mahmoody, and Florian Tramer. Neuracrypt is not private. *arXiv preprint arXiv:2108.07256*, 2021.

[115] Matthew Fredrikson, Eric Lantz, Somesh Jha, Simon Lin, David Page, and Thomas Ristenpart. Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 17–32, 2014.

[116] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1322–1333, 2015.

[117] Xi Wu, Matthew Fredrikson, Somesh Jha, and Jeffrey F. Naughton. A methodology for formalizing model-inversion attacks. In *2016 IEEE 29th Computer Security Foundations Symposium (CSF)*, pages 355–370, Lisbon, 2016. IEEE, IEEE.

[118] Cynthia Dwork, Adam Smith, Thomas Steinke, and Jonathan Ullman. Exposed! a survey of attacks on private data. *Annual Review of Statistics and Its Application*, 4:61–84, 2017.

[119] Cynthia Dwork, Adam Smith, Thomas Steinke, Jonathan Ullman, and Salil Vadhan. Robust traceability from trace amounts. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 650–669. IEEE, 2015.

[120] Sriram Sankararaman, Guillaume Obozinski, Michael I Jordan, and Eran Halperin. Genomic privacy and limits of individual detection in a pool. *Nature genetics*, 41(9):965–967, 2009.

[121] Nils Homer, Szabolcs Szelinger, Margot Redman, David Duggan, Waibhav Tembe, Jill Muehling, John V. Pearson, Dietrich A. Stephan, Stanley F. Nelson, and David W Craig. Resolving individuals contributing trace amounts of dna to highly complex mixtures using high-density snp genotyping microarrays. *PLoS Genet*, 4(8):e1000167, 2008.

[122] Congzheng Song, Thomas Ristenpart, and Vitaly Shmatikov. Machine learning models that remember too much. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 587–601, 2017.

[123] Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 267–284, 2019.

[124] Vitaly Feldman. Does learning require memorization? a short tale about a long tail. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 954–959, 2020.

[125] Amos Beimel, Hai Brenner, Shiva Prasad Kasiviswanathan, and Kobbi Nissim. Bounds on the sample complexity for private learning and private data release. *Machine learning*, 94(3):401–437, 2014.

[126] Arvind Narayanan and Vitaly Shmatikov. How to break anonymity of the netflix prize dataset. *arXiv preprint cs/0610105*, 2006.

[127] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd Annual Symposium on Foundations of Computer Science*, pages 136–145, Las Vegas, NV, USA, October 14–17, 2001. IEEE Computer Society Press.

[128] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of computer and system sciences*, 28(2):270–299, 1984.

[129] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 427–437, 1990.

[130] David Harrison. Hedonic housing prices and the demand for clean air. *Journal of environmental economics and management*, 5:81–102, 1978.

[131] Bradley Efron, Trevor Hastie, Iain Johnstone, Robert Tibshirani, et al. Least angle regression. *The Annals of statistics*, 32(2):407–499, 2004.

[132] Ronald A Fisher. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2):179–188, 1936.

[133] Stefan Aeberhard, Danny Coomans, and Olivier De Vel. Comparative analysis of statistical pattern recognition methods in high dimensional settings. *Pattern Recognition*, 27(8):1065–1077, 1994.

[134] W Nick Street, William H Wolberg, and Olvi L Mangasarian. Nuclear feature extraction for breast tumor diagnosis. In *Biomedical image processing and biomedical visualization*, volume 1905, pages 861–870. International Society for Optics and Photonics, 1993.

[135] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[136] Gavin Brown, Mark Bun, Vitaly Feldman, Adam Smith, and Kunal Talwar. When is memorization of irrelevant training data necessary for high-accuracy learning? In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 123–132, 2021.

[137] Brenden M. Lake, Ruslan Salakhutdinov, and Joshua B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.