# 3 A Framework for Secure Learning

In this chapter we introduce a framework for qualitatively assessing the security of machine learning systems that captures a broad set of security characteristics common to a number of related adversarial learning settings. There has been a rich set of work that examines the security of machine learning systems; here we survey prior studies of learning in adversarial environments, attacks against learning systems, and proposals for making systems secure against attacks. We identify different classes of attacks on machine learning systems (Section 3.3), categorizing a threat in terms of three crucial properties.

We also present secure learning as a game between an *attacker* and a *defender*—the taxonomy determines the structure of the game and its cost model. Further, this taxonomy provides a basis for evaluating the resilience of the systems described by analyzing threats against them to construct defenses. The development of defensive learning techniques is more tentative, but we also discuss a variety of techniques that show promise for defending against different types of attacks.

The work we present not only provides a common language for thinking and writing about secure learning, but goes beyond that to show how the framework applies to both algorithm design and the evaluation of real-world systems. Not only does the framework elicit common themes in otherwise disparate domains but it has also motivated our study of practical machine learning systems as presented in Chapters 5, 6, and 8. These foundational principles for characterizing attacks against learning systems are an essential first step if secure machine learning is to reach its potential as a tool for use in real systems in security-sensitive domains.

This chapter builds on earlier research (Barreno, Nelson, Sears, Joseph, & Tygar 2006; Barreno, Nelson, Joseph, & Tygar 2010; Barreno 2008).

## 3.1 Analyzing the Phases of Learning

Attacks can occur at each of the phases of the learning process that were outlined in Section 2.2. Figure 2.1(a) depicts how data flows through each phase of learning. We briefly outline how attacks against these phases differ.

*The Measuring Phase*

With knowledge of the measurement process, an adversary can design malicious instances to mimic the measurements of innocuous data. After a successful attack

https://doi.org/10.1017/9781107338548.003 Published online by Cambridge University Press

against the measurement mechanism, the system may require expensive re-instrumentation or redesign to accomplish its task.

*The Feature Selection Phase*

The feature selection process can be attacked in the same manner as the measuring phase, except countermeasures and recovery are less costly since feature selection is a dynamic process that can be more readily adapted. Potentially, retraining could even be automated. However, feature selection can also be attacked in the same manner as the training phase (below) if feature selection is based on training data that may be contaminated.

*Learning Model Selection*

Once the learning model is known, an adversary could exploit assumptions inherent in the model. Erroneous or unreasonable modeling assumptions about the training data may be exploited by an adversary; e.g., if a model erroneously assumes linear separability in the data, the adversary could use data that cannot be separated linearly to deceive the learner or make it perform poorly. It is essential to explicitly state and critique the modeling assumptions to identify potential vulnerabilities since changing the model may require that the system be redesigned.

*The Training Phase*

By understanding how the learner trains, an adversary can design data to fool the learner into choosing a poor hypothesis, or to aid a later privacy breach of the training data during prediction. Robust learning methods are promising techniques to counter the former attacks as discussed in Section 3.5.4.3. These methods are resilient to adversarial contamination although there are inherent tradeoffs between their robustness and performance. Differential privacy is a leading approach to providing strong guarantees on the level of privacy preserved by learners in the presence of powerful adversaries (Dwork et al. 2006). We explore mechanisms for differentially private approximation of support vector classification in Chapter 7.

*The Prediction Phase*

Once learned, an imperfect hypothesis can be exploited by an adversary who discovers prediction errors made by the learner. Assessing how difficult it is to discover such errors is an important question; e.g., the ACRE-learning framework of Lowd & Meek (2005*a*) as discussed in Section 3.4.4. In addition, an adversary can exploit an imperfect hypothesis to breach training data privacy during prediction. An interesting avenue of future research is detecting that an adversary is exploiting these errors and retraining to counter the attack.

To better understand these different abstract attacks, consider a spam filter that (*i*) has some simple set of measurements of email such as *hasAttachment*, *subjectLength*, *bodyLength*, *etc.*, (*ii*) selects the top-ten most frequently appearing features in spam, (*iii*) uses the naive Bayes model, (*iv*) trains class frequencies by empirical counts, and (*v*) classifies email by thresholding the model's predicted class probabilities. An attack

https://doi.org/10.1017/9781107338548.003 Published online by Cambridge University Press

against the measurement (or feature selection) phase would consist of first determining the features used (for classification) and then producing spams that are indistinguishable from normal email for those features. An attack against the learning model would entail discovering a set of spams and hams that could not be classified correctly due to the linearity of the naive Bayes boundary. Further, the training system (or feature selection) could be attacked by injecting spams with misleading spurious features, causing it to learn the wrong hypothesis or to learn a hypothesis that improves the odds of revealing privacy-sensitive training emails that were not tampered with. Finally, the prediction phase could be attacked by systematically probing the filter to find spams that are misclassified as ham (false negatives) or to infer information about the privacy-sensitive training dataset.

Many learning methods make a stationarity assumption: training data and evaluation data are drawn from the same distribution. Under this assumption minimizing the risk on the training set is a surrogate for risk on the evaluation data. However, real-world sources of data often are not stationary, and even worse, attackers can easily break the stationarity assumption with some control of either training or evaluation instances. Analyzing and strengthening learning methods to withstand or mitigate violations of the stationarity assumption are the crux of the *secure learning* problem.

Qualifying the vulnerable components of the learning system is only the first step to understanding the adversary. In the next section, we outline a framework designed to qualify the adversary's goals.

## 3.2 Security Analysis

Security is concerned with protecting assets from attackers. Properly analyzing the security of a system requires identifying the security goals and a threat model for the system. A security goal is a requirement that, if violated, results in the partial or total compromise of an asset. A threat model is a profile of attackers who wish to harm the system, describing their motivation and capabilities. Here we describe possible security goals and threat models for machine learning systems.

In a security-sensitive domain, classifiers can be used to make distinctions that advance the security goals of the system. For example, a virus detection system has the goal of reducing susceptibility to virus infection, either by detecting the virus in transit prior to infection or by detecting an extant infection to expunge. Another example is an intrusion detection system (IDS), which has the goal of preventing harm from malicious intrusion, either by identifying existing intrusions for removal or by detecting malicious traffic and preventing it from reaching its intended target.[1] In this section, we describe security goals and threat models that are specific to machine learning systems.

[1] In the case of preventing intrusion, the whole system is more properly called an intrusion prevention system (IPS). We have no need to distinguish between the two cases, so we use IDS to refer to both intrusion detection systems and intrusion prevention systems.

https://doi.org/10.1017/9781107338548.003 Published online by Cambridge University Press

### 3.2.1 Security Goals

In a security context the classifier's purpose is to classify malicious events and prevent them from interfering with system operations. We split this general learning goal into three goals:

- **Integrity goal**: To prevent attackers from reaching system assets
- **Availability goal**: To prevent attackers from interfering with normal operation
- **Privacy goal**: To protect the confidentiality of potentially privacy-sensitive data used to train the classifier

There is a clear connection between false negatives and violation of the integrity goal: malicious instances that pass through the classifier can wreak havoc. Likewise, false positives tend to violate the availability goal because the learner itself denies benign instances. Finally while an attacker attempting to breach privacy may cause false positives or false negatives, these will typically not be the end goal.

### 3.2.2 Threat Model

*Attacker Goal and Incentives*

In general the attacker wants to access system assets (typically with false negatives or through inverting the learning process in the case of a training set privacy violation) or to deny normal operation (usually with false positives). For example, a virus author wants viruses to pass through the filter and take control of the protected system (a false negative). On the other hand, an unscrupulous merchant may want sales traffic to a competitor's web store to be blocked as intrusions (false positives). Finally an over-zealous health insurer might reverse engineer an automated cancer detector to breach the private medical records of the patients who make up the detector's training set.

We assume that the attacker and defender each have a cost function that assigns a cost either to each labeling for any given instance or to each training instance privacy violation. Cost can be positive or negative; a negative cost is a benefit. It is usually the case that low cost for the attacker parallels high cost for the defender and vice versa; the attacker and defender would not be adversaries if their goals aligned. Unless otherwise stated, for ease of exposition we assume that every cost for the defender corresponds to a similar benefit for the attacker and vice versa. However, this assumption is not essential to this framework, which extends easily to arbitrary cost functions. In this chapter, we take the defender's point of view and use *high cost* to mean high positive cost for the defender.

#### 3.2.2.1 Attacker Capabilities

Making only weak assumptions on the adversary, we adopt a threat model in which the attacker has knowledge of the training algorithm and in many cases partial or complete information about the training set, such as its distribution. For example, the attacker may have the ability to eavesdrop on all network traffic over the period of time in which the

https://doi.org/10.1017/9781107338548.003 Published online by Cambridge University Press

learner gathers training data. We examine different degrees of the attacker's knowledge and assess how much it gains from different sources of potential information.

In general, we assume the attacker can generate arbitrary instances; however, many settings do impose significant restrictions on the instances generated by the attacker. For example, when the learner trains on data from the attacker, sometimes it is safe to assume that the attacker cannot choose the label for training, such as when training data is carefully hand labeled. As another example, an attacker may have complete control over data packets being sent from the attack source, but routers in transit may add to or alter the packets as well as affect their timing and arrival order.

We consider an attacker with the ability to modify or generate data used in training, and explore scenarios both when it has this capability and when it does not. When the attacker controls training data, an important limitation to consider is what fraction of the training data the attacker can control and to what extent. If the attacker has arbitrary control over 100% of the training data, it is difficult to see how the learner can learn anything useful; however, even in such cases there are learning strategies that can make the attacker's task more difficult (see Section 3.6). We examine intermediate cases and explore how much influence is required for the attacker to defeat the learning procedure.

### 3.2.3 Discussion of Machine Learning Applications in Security

Sommer & Paxson (2010) have raised a set of objections to some uses of machine learning in security applications. In Section III of that paper, the authors outline five challenges of using machine learning for network anomaly detection. Their first challenge is outlier detection, and the authors argue that network anomaly detection is best done using filtering, instead of the classification that machine learning provides. They also argue that machine learning excels at finding similar occurrences, but does not work for finding novel occurrences. Their second challenge is the high cost of errors, and the authors argue that network anomaly detection systems have a stringent limit on the number of false positive and true positive errors that they can tolerate, with false positives being particularly expensive due to the analyst time required to resolve them. Their third challenge is the semantic gap between anomaly detection systems and users/operators/analysts, and they argue that machine learning-based systems are opaque in decisions they make. Their fourth challenge is that the diversity of network traffic yields very bursty networks, making decision making challenging. They also argue that diversity reduction techniques like aggregation make networks noisy and also make decision making challenging. Their fifth challenge is around the difficulties with evaluation and has multiple components: a lack of usable public datasets, a semantic gap in explaining a machine learning-based system, decisions about novel attacks to users/operators/analysts, and the adversarial setting of network anomaly detection.

Sommer & Paxson (2010) raise interesting points, with a full discussion lying outside the scope of this book. Elsewhere (Miller, Kantchelian, Afroz, Bachwani, Dauber, Huang, Tschantz, Joseph, & Tygar 2014) we discuss frameworks based on adversarial active learning specifically designed to handle novel occurrences of attacks and to intelligently allocate human time required to respond to them, addressing the first two

https://doi.org/10.1017/9781107338548.003 Published online by Cambridge University Press

challenges presented by Sommer & Paxson (2010), and also the issue of evaluation in the adversarial setting of their fifth challenge. The field of *secure machine learning* is specifically designed to present robust machine learning, and in that paper we argue for a way to intelligently allocate human resources in dealing with malicious input. Note that the other challenges raised by Sommer & Paxson (2010) are challenges both for traditional filtering systems and for machine learning systems. For example, the question of the availability of datasets is an issue for both researchers of secure machine learning and traditional filtering systems. In a number of research projects including Intel's "Canary" algorithm (Chandrashekar, Orrin, Livadas, & Schooler 2009), Google's malicious advertising detection system (Sculley, Otey, Pohl, Spitznagel, Hainsworth, & Zhou 2011), and UC Berkeley's comment spam detection work (Kantchelian, Ma, Huang, Afroz, Joseph, & Tygar 2012), researchers have been able to gain access to real-world high-quality large datasets.

## 3.3 Framework

The framework we describe here has three primary components: a taxonomy based on the common characteristics of attacks against learning algorithms, a high-level description of the elements of the game played between the attacker and defender (learner), and set of common characteristics for an attacker's capabilities. Each of these elements helps organize and assess the threat posed by an attacker.

### 3.3.1 Taxonomy

A great deal of the work that has been done within secure learning is the analysis of attack and defense scenarios for particular learning applications. Together with Marco Barreno and Russell Sears, we developed a qualitative taxonomy of attacks against machine learning systems that we used to categorize others' research, to find commonalities between otherwise disparate domains, and ultimately to frame our own research. We present the taxonomy categorizing attacks against learning systems along three axes. Each of these dimensions operates independently, so we have at least 12 distinct classes of attacks on machine learning systems.

**Influence**

- *Causative* attacks influence learning with control over training data.
- *Exploratory* attacks exploit predictions, but do not affect training.

**Security violation**

- *Integrity* attacks compromise assets via false negatives.
- *Availability* attacks cause denial of service, usually via false positives.
- *Privacy* attacks obtain information from the learner, compromising the privacy of the learner's training data.

**Specificity**

- *Targeted* attacks focus on a particular instance.
- *Indiscriminate* attacks encompass a wide class of instances.

https://doi.org/10.1017/9781107338548.003 Published online by Cambridge University Press

The first axis describes the capability of the attacker: whether (a) the attacker has the ability to influence the training data that is used to construct the classifier (a *Causative* attack) or (b) the attacker does not influence the learned classifier, but can send new instances to the classifier and possibly observe its decisions on these carefully crafted instances (an *Exploratory* attack).

The second axis indicates the type of security violation the attacker causes: either (a) allowing harmful instances to slip through the filter as false negatives (an *Integrity* violation); (b) creating a denial-of-service event in which benign instances are incorrectly filtered as false positives (an *Availability* violation); or (c) using the filter's responses to infer confidential information used in the learning process (a *Privacy* violation).

The third axis refers to how specific the attacker's intention is: whether (a) the attack is highly *Targeted* to degrade the classifier's performance on one particular instance or to violate the privacy of one particular training instance or (b) the attack aims to cause the classifier to fail in an *Indiscriminate* fashion on a broad class of instances. Each axis, especially this one, can actually be a spectrum of choices, but for simplicity, we will categorize attacks and defenses into these groupings.

These axes define the space of attacks against learners and aid in identifying unconventional threats. By qualifying where an attack lies in this space, one can begin to quantify the adversary's capabilities and assess the risk posed by this threat. Laskov & Kloft (2009) have since extended these basic principles to propose a framework for quantitatively evaluating security threats.

In Table 3.1, we use our taxonomy to classify past work on adversarial learning with goals relating to misclassifications—*Integrity* and *Availability* attacks, but not *Privacy*

**Table 3.1** Selected related work on misclassification attacks in the taxonomy

| | *Integrity* | *Availability* |
|---|---|---|
| *Causative*: *Targeted* | Kearns & Li (1993); Newsome et al. (2006) | Kearns & Li (1993); Newsome et al. (2006); Chung & Mok (2007); Nelson, Barreno, Chi, Joseph, Rubinstein, Saini, Sutton, Tygar, & Xia (2008) |
| *Indiscriminate* | Kearns & Li (1993); Newsome et al. (2006) | Kearns & Li (1993); Newsome et al. (2006); Chung & Mok (2007); Nelson et al. (2008) |
| *Exploratory*: *Targeted* | Tan et al. (2002); Lowd & Meek (2005*a*, 2005*b*); Wittel & Wu (2004) | Moore, Shannon, Brown, Voelker, & Savage (2006) |
| *Indiscriminate* | Fogla & Lee (2006); Lowd & Meek (2005*b*); Wittel & Wu (2004) | Moore et al. (2006) |

https://doi.org/10.1017/9781107338548.003 Published online by Cambridge University Press

attacks. As we discuss in Section 3.7 while the taxonomy provides a useful classification of possible privacy attacks on machine learning systems, past work (with a few exceptions) tends to fall in most bins simultaneously.

### 3.3.2 The Adversarial Learning Game

We model the task of constructing a secure learning system as a game between an attacker and a defender—the attacker manipulates data to mistrain or evade or violate privacy of a learning algorithm chosen by the defender to thwart the attacker's objective. The characteristics specified by the taxonomy's axes also designate some aspects of this game. The INFLUENCE axis determines the structure of the game and the legal moves that each player can make. The SPECIFICITY and SECURITY VIOLATION axes of the taxonomy determine the general shape of the cost function: an *Integrity* attack benefits the attacker on false negatives, and therefore focuses high cost (to the defender) on false negatives; an *Availability* attack focuses high cost on false positives; and a *Privacy* attack focuses high cost on leaking information about training instances. Similarly a *Targeted* attack focuses high cost only on a small number of specific instances, while an *Indiscriminate* attack spreads high cost over a broad range of instances.

We formalize the game as a series of *moves*, or *steps*. Each move either is a strategic choice by one of the players or is a *neutral* move not controlled by either player. The choices and computations in a move depend on information produced by previous moves (when a game is repeated, this includes previous iterations) and on domain-dependent constraints that we highlight in discussing prior work. Generally, though, in an *Exploratory* attack, the attacker chooses a procedure $A^{(\text{eval})}$ that affects the evaluation data $\mathbb{D}^{(\text{eval})}$, and in a *Causative* attack, the attacker also chooses a procedure $A^{(\text{train})}$ to manipulate the training data $\mathbb{D}^{(\text{train})}$. In either setting, the defender chooses a learning algorithm $H^{(N)}$. This formulation gives us a theoretical basis for analyzing the interactions between attacker and defender.

### 3.3.3 Characteristics of Adversarial Capabilities

In this section we introduce three essential properties for constructing a model of an attack against a learning algorithm that refine the game played between the learner and the adversary as described by the taxonomy. These properties define a set of common domain-specific adversarial limitations that allow a security analyst to formally describe the capabilities of the adversary.

#### 3.3.3.1 Corruption Models

The most important aspect of the adversary is how it can alter data to mislead or evade the classifier, or reveal information about the training data. As previously stated, learning against an unlimited adversary is futile. Instead, the security analysis we propose focuses on a limited adversary, but to do so, one must model the restrictions on the adversary and justify these restrictions for a particular domain. We outline two common

https://doi.org/10.1017/9781107338548.003 Published online by Cambridge University Press

models for adversarial corruption, and we describe how the adversary is limited within each.

*Data Insertion Model*

The first model assumes the adversary has unlimited control of a small fraction of the data; i.e., the adversary is restricted to only modifying a limited amount of data, but can alter those data points arbitrarily. We call this an *insertion model* because, in this scenario, the adversary crafts a small number of attack instances and *inserts* them into the dataset for training or evaluation (or perhaps replaces existing data points). For example, in the case of a spam filter, the adversary (spammer) can create any arbitrary message for their attack, but it is limited in the number of attack messages it can inject: the spammer does not have control over innocuous email messages sent by third parties. The spammer's attack on the spam filter can be analyzed in terms of how many messages are required for the attack to be effective. For this reason, we use this model of corruption in analyzing attacks on the SpamBayes spam filter in Chapter 5 and show that, even with a relatively small number of attack messages, the adversarial spammer can significantly mislead the filter.

*Data Alteration Model*

The second corruption model instead assumes that the adversary can alter any (or all) of the data points in the dataset, but is limited in the degree of alteration; i.e., an *alteration model*. For example, to attack a detector that is monitoring network traffic volumes over windows of time, the adversary can add or remove traffic within the network, but only can make a limited degree of alteration. Such an adversary cannot insert new data since each data point corresponds to a time slice and the adversary cannot arbitrarily control any single data point because other actors are also creating traffic in the network. Here the adversary is restricted by the total amount it can alter, and so the effectiveness of its attack can be analyzed in terms of the size of alteration required to achieve the attacker's objective. This is the model we use for analyzing attacks on a PCA-subspace detector for network anomaly detection in Chapter 6, and again we show that, with a relatively small degree of control, the adversary can dramatically degrade the effectiveness of this detector using data alterations.

#### 3.3.3.2 Class Limitations

A second limitation on attackers involves which parts of the data the adversary is allowed to alter—the positive (malicious) class, the negative (benign) class, or both. Usually in settings where the adversary aims to affect misclassifications, attackers external to the system are only able to create malicious data and so they are limited to only manipulating positive instances. This is the model we use throughout this text. However, there is also an alternative threat that *insiders* could attack a learning system by altering negative instances. We do not analyze this threat in this book but return to the issue in the discussion in Chapter 9. In the setting of *Privacy* attacks, the learner may not necessarily be classifying instances as malicious or benign so there may not be any natural class limitation on the attacker's influence.

https://doi.org/10.1017/9781107338548.003 Published online by Cambridge University Press

#### 3.3.3.3 Feature Limitations

The final type of adversarial limitation we consider are limits on how an adversary can alter data points in terms of each *feature*. Features represent different aspects of the state of the world and have various degrees of vulnerability to attack. Some features can be arbitrarily changed by the adversary, but others may have stochastic aspects that the adversary cannot completely control, and some features may not be alterable at all. For instance, in sending an email, the adversary can completely control the content of the message, but cannot completely determine the routing of the message or its arrival time. Further, this adversary has no control over meta-information that is added to the message's header by mail relays while the message is en route. Providing an accurate description of the adversary's control over the features is essential.

### 3.3.4 Attacks

In the remainder of this chapter, we survey prior research, we discuss how attack and defense strategies were developed in different domains, we reveal their common themes, and we highlight important aspects of the secure learning game in the context of this taxonomy. The related work discussed later is also presented in the taxonomy in Table 3.1. For an *Exploratory* attack, we discuss realistic instances of the attacker's choice for $A^{(\text{eval})}$ in Sections 3.4.2 and 3.4.3. Similarly, in Sections 3.5.2 and 3.5.3, we discuss practical examples of the attacker's choices in the *Causative* game.

We treat *Privacy* attacks separately from attacks aiming to produce misclassifications, with a dedicated discussion in Section 3.7.

### 3.3.5 Defenses

The game between attacker and defender and the taxonomy also provide a foundation on which to construct defense strategies against broad classes of attacks. We address *Exploratory* and *Causative* misclassification attacks separately. For *Exploratory* attacks, we discuss the defender's choice for an algorithm $H^{(N)}$ in Section 3.4.4 and we discuss the defender's strategies in a *Causative* setting in Section 3.5.4. Finally, in Section 3.6, we discuss the broader setting of an iterated game.

We treat defenses against *Exploratory* and *Causative Privacy* attacks together in Section 3.7.

In all cases, defenses present a tradeoff: changing the algorithms to make them more robust against (worst-case) attacks will generally make them *less* effective on non-adversarial data. Analyzing this tradeoff is an important part of developing defenses.

## 3.4 Exploratory Attacks

Based on the INFLUENCE axis of the taxonomy, the first category of attacks that we discuss are *Exploratory* attacks, which influence only the evaluation data as indicated in

https://doi.org/10.1017/9781107338548.003 Published online by Cambridge University Press

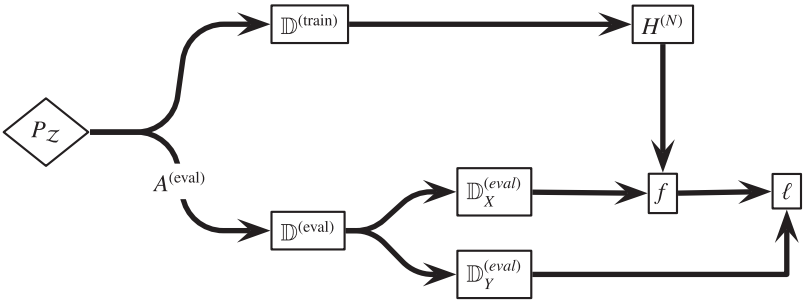


**Figure 3.1** Diagram of an *Exploratory* attack against a learning system (see Figure 2.1).

Figure 3.1. The adversary's transformation $A^{(\text{eval})}$ alters the evaluation data either by defining a procedure to change instances drawn from $P_{\mathcal{Z}}$ or by changing $P_{\mathcal{Z}}$ to an altogether different distribution $P_{\mathcal{Z}}^{(\text{eval})}$ chosen by the adversary. The adversary makes these changes based on (partial) information gleaned about the training data $\mathbb{D}^{(\text{train})}$, the learning algorithm $H^{(N)}$, and the classifier $f$. Further, the adversary's transformation may evolve as the adversary learns more about the classifier with each additional prediction it makes.

### 3.4.1 The Exploratory Game

First we present the formal version of the game for *Exploratory* attacks and then explain it in greater detail.

1 **Defender** Choose procedure $H^{(N)}$ for selecting an hypothesis
2 **Attacker** Choose procedure $A^{(\text{eval})}$ for selecting an evaluation distribution
3 Evaluation:
- Reveal distribution $P_{\mathcal{Z}}^{(\text{train})}$
- Sample dataset $\mathbb{D}^{(\text{train})}$ from $P_{\mathcal{Z}}^{(\text{train})}$
- Compute $f \leftarrow H^{(N)}\left(\mathbb{D}^{(\text{train})}\right)$
- Compute $P_{\mathcal{Z}}^{(\text{eval})} \leftarrow A^{(\text{eval})}(\mathbb{D}^{(\text{train})}, f)$
- Sample dataset $\mathbb{D}^{(\text{eval})}$ from $P_{\mathcal{Z}}^{(\text{eval})}$
- Assess total cost: $\sum_{(x,y)\in\mathbb{D}^{(\text{eval})}} \ell_x\left(f(x), y\right)$

The defender's move is to choose a learning algorithm (procedure) $H^{(N)}$ for creating hypotheses from datasets. Many procedures used in machine learning have the form of Equation (2.1). For example, the defender may choose a support vector machine (SVM) with a particular kernel, loss, regularization, and cross-validation plan. The attacker's move is then to choose a procedure $A^{(\text{eval})}$ to produce a distribution on which to evaluate the hypothesis that $H^{(N)}$ generates. (The degree of control the attacker has in generating

https://doi.org/10.1017/9781107338548.003 Published online by Cambridge University Press

the dataset and the degree of information about $\mathbb{D}^{(\text{train})}$ and $f$ that $A^{(\text{eval})}$ has access to are setting specific.)

After the defender and attacker have both made their choices, the game is evaluated. A training dataset $\mathbb{D}^{(\text{train})}$ is drawn from some fixed and possibly unknown distribution $P_{\mathcal{Z}}^{(\text{train})}$, and training produces $f = H^{(N)}\left(\mathbb{D}^{(\text{train})}\right)$. The attacker's procedure $A^{(\text{eval})}$ produces distribution $P_{\mathcal{Z}}^{(\text{eval})}$, which is based in general on $\mathbb{D}^{(\text{train})}$ and $f$, and an evaluation dataset $\mathbb{D}^{(\text{eval})}$ is drawn from $P_{\mathcal{Z}}^{(\text{eval})}$. Finally, the attacker and defender incur cost based on the performance of $f$ evaluated on $\mathbb{D}^{(\text{eval})}$ according to the loss function $\ell_x(\cdot, \cdot)$. Note that, unlike in Section 2.2, here we allow the loss function to depend on the data point $x$. This generalization allows this game to account for an adversary (or learner) with instance-dependent costs (cf. Dalvi, Domingos, Mausam, Sanghai, & Verma 2004).

The procedure $A^{(\text{eval})}$ generally depends on $\mathbb{D}^{(\text{train})}$ and $f$, but the amount of information an attacker actually has is setting specific (in the least restrictive case the attacker knows $\mathbb{D}^{(\text{train})}$ and $f$ completely). The attacker may know a subset of $\mathbb{D}^{(\text{train})}$ or the family $\mathcal{F}$ containing $f$. However, the procedure $A^{(\text{eval})}$ may also involve acquiring information dynamically. For instance, in some cases, the procedure $A^{(\text{eval})}$ can query the classifier, treating it as an oracle that provides labels for query instances; this is one particular degree of information that $A^{(\text{eval})}$ can have about $f$. An attack that uses this technique is called a probing attack. Probing can reveal information about the classifier. On the other hand, with sufficient prior knowledge about the training data and algorithm, the attacker may be able to find high-cost instances without probing.

### 3.4.2 Exploratory Integrity Attacks

A frequently studied attack is the *Exploratory Integrity* attack in which the adversary attempts to passively circumvent the learning mechanism to exploit blind spots in the learner that allow miscreant activities to go undetected. In an *Exploratory Integrity* attack, the attacker crafts intrusions so as to evade the classifier without direct influence over the classifier itself. Instead, attacks of this sort often attempt to systematically make the miscreant activity appear to be normal activity to the detector or obscure the miscreant activity's identifying characteristics. Some *Exploratory Integrity* attacks mimic statistical properties of normal traffic to camouflage intrusions; e.g., the attacker examines training data and the classifier, then crafts intrusion data. In the *Exploratory* game, the attacker's move produces malicious instances in $\mathbb{D}^{(\text{eval})}$ that statistically resemble normal traffic in the training data $\mathbb{D}^{(\text{train})}$.

EXAMPLE 3.1 (The Shifty Intruder)
An attacker modifies and obfuscates intrusions, such as by changing network headers and reordering or encrypting contents. If successful, these modifications prevent the IDS from recognizing the altered intrusions as malicious, so it allows them into the system. In the *Targeted* version of this attack, the attacker has a particular intrusion to get past the filter. In the *Indiscriminate* version, the attacker has no particular preference and can search for any intrusion that succeeds, such as by modifying a large number of different exploits to see which modifications evade the filter.

https://doi.org/10.1017/9781107338548.003 Published online by Cambridge University Press

#### 3.4.2.1 Polymorphic Blending Attack

Fogla & Lee (2006) introduce *polymorphic blending attacks* that evade intrusion detectors using encryption techniques to make attacks statistically indistinguishable from normal traffic according to the intrusion detection system. They present a formalism for reasoning about and generating polymorphic blending attack instances to evade intrusion detection systems. The technique is fairly general and is *Indiscriminate* in terms of the intrusion packets it modifies.

*Feature deletion attacks* instead specifically exclude high-value identifying features used by the detector (Globerson & Roweis 2006); this form of attack stresses the importance of proper feature selection as was also demonstrated empirically by Mahoney & Chan (2003) in their study of the behavior of intrusion detection systems on the DARPA/Lincoln Lab dataset.

#### 3.4.2.2 Attacking a Sequence-Based IDS

Tan et al. (2002) describe a mimicry attack against the `stide` sequence-based intrusion detection system (IDS) proposed by Forrest et al. (1996) and Warrender et al. (1999). They modify exploits of the `passwd` and `traceroute` programs to accomplish the same ends using different sequences of system calls: the shortest subsequence in attack traffic that does not appear in normal traffic is longer than the IDS window size. By exploiting the finite window size of the detector, this technique makes attack traffic indistinguishable from normal traffic for the detector. This attack is more *Targeted* than polymorphic blending since it modifies particular intrusions to look like normal traffic. In subsequent work Tan, McHugh, & Killourhy (2003) characterize their attacks as part of a larger class of information hiding techniques that they demonstrate can make exploits mimic either normal call sequences or the call sequence of another less severe exploit.

Independently, Wagner & Soto (2002) have also developed mimicry attacks against a sequence-based IDS called pH proposed by Somayaji & Forrest (2000). Using the machinery of finite automata, they construct a framework for testing whether an IDS is susceptible to mimicry for a particular exploit. In doing so, they develop a tool for validating IDSs on a wide range of variants of a particular attack and suggest that similar tools should be more broadly employed to identify the vulnerabilities of an IDS.

Overall, these mimicry attacks against sequence-based anomaly detection systems underscore critical weaknesses in these systems that allow attackers to obfuscate the necessary elements of their exploits to avoid detection by mimicking normal behaviors. Further they highlight how an IDS may appear to perform well against a known exploit, but unless it captures necessary elements of the intrusion, the exploit can easily be adapted to circumvent the detector. See Section 3.4.4 for more discussion.

#### 3.4.2.3 Good Word Attacks

Adding or changing words in a spam message can allow the message to bypass the filter. Like the attacks against an IDS described earlier, these attacks all use both training data and information about the classifier to generate instances intended to bypass the

https://doi.org/10.1017/9781107338548.003 Published online by Cambridge University Press

filter. They are somewhat independent of the *Targeted*/*Indiscriminate* distinction, but the *Exploratory* game captures the process used by all of these attacks.

Studying these techniques was first suggested by John Graham-Cumming. In his presentation *How to Beat an Adaptive Spam Filter* delivered at the 2004 MIT Spam Conference, he presented a *Bayes vs. Bayes* attack that uses a second statistical spam filter to find good words based on feedback from the filter under attack. Several authors have further explored evasion techniques used by spammers and demonstrated attacks against spam filters using similar principles as those against IDSs as discussed earlier. Lowd & Meek (2005*b*) and Wittel & Wu (2004) develop attacks against statistical spam filters that add *good words*, or words the filter considers indicative of non-spam, to spam emails. This good word attack makes spam emails appear innocuous to the filter, especially if the words are chosen to be ones that appear often in non-spam email and rarely in spam email. Finally, obfuscation of spam words (i.e., changing characters in the word or the spelling of the word so it is no longer recognized by the filter) is another popular technique for evading spam filters that has been formalized by several authors (cf. Liu & Stamm 2007 and Sculley, Wachman, & Brodley 2006).

#### 3.4.2.4 Cost-Based Evasion

Another line of research focuses on the costs incurred due to the adversary's evasive actions; i.e., instances that evade detection may be less desirable to the adversary. By directly modeling adversarial cost, this work explicitly casts evasion as a problem in which the adversary wants to evade detection, but wants to do so using high-value instances (an assumption that was implicit in the other work discussed in this section).

##### *Game-Theoretic Approaches*

Dalvi et al. (2004) exploit these costs to develop a cost-sensitive game-theoretic classification defense that is able to successfully detect optimal evasion of the original classifier. Using this game-theoretic approach, this technique preemptively patches the naive classifier's blind spots by constructing a modified classifier designed to detect optimally modified instances.

Subsequent game-theoretic approaches to learning have extended this setting and solved for equilibria of the game (Brückner & Scheffer 2009; Kantarcioglu, Xi, & Clifton 2009). Further, Biggio, Fumera, & Roli (2010) extend this game-theoretic approach and propose hiding information or randomization as additional defensive mechanisms for this setting. Großhans, Sawade, Brückner, & Scheffer (2013) explore Bayesian games between Bayesian statistician defender and attacker, in the non-zero-sum case under incomplete information and with continuous action spaces. Here they model the defender's (partial) knowledge of the attacker as encoded in the prior, demonstrate sufficient conditions for unique equilbrium existence (under sufficiently strong regularization of costs), and present an algorithm for computing such an equilibrium.

##### *Evasion by Membership Queries*

Cost models of the adversary also led to a theory for query-based near-optimal evasion of classifiers first presented by Lowd & Meek (2005*a*), in which they cast the difficulty

https://doi.org/10.1017/9781107338548.003 Published online by Cambridge University Press

of evading a classifier into a abstract query complexity problem. They give algorithms for an attacker to reverse engineer a classifier. The attacker seeks the highest cost (lowest cost for the attacker) instance that the classifier labels *negative*. In *Near-Optimal Evasion of Convex-Inducing Classifiers*, we developed an extension to this work (Nelson, Rubinstein, Huang, Joseph, Lau, Lee, Rao, Tran, & Tygar 2010). We generalized the theory of near-optimal evasion to a broader class of classifiers and demonstrated that the problem is easier than reverse-engineering approaches. We go into greater detail on this work in Chapter 8. The ACRE framework (Lowd & Meek 2005*a*) is further extended by Stevens & Lowd (2013) to encompass concept classes that are convex polytopes representing unions or intersections of linear classifiers, in the difficult case of discrete features (contrasting with the continuous case explored for convex-inducing classifiers in Chapter 8). Query complexity upper bounds for this nonlinear discrete setting suggest that these classes are difficult to evade via membership queries, unlike their linear counterparts.

*Attacks on Deployed Systems*

Reverse-engineering attacks have been deployed by Tramèr, Zhang, Juels, Reiter, and Ristenpart (2016) against major cloud-based Machine-Learning-as-a-service systems. When target learning systems output prediction confidence values, the researchers probe the model randomly to obtain a set of instance-confidence pairs from which they solve a system of equations to determine decision boundaries in the case of logistic regression and neural networks. When the target model is a tree, they leverage the confidence values output at leaves to develop an efficient and exact path-finding algorithm to reconstruct the tree. Without access to the confidence values, the authors apply the idea of membership queries (see Chapter 8), specifically extending the technique of Lowd & Meek (2005*a*) when models are linear under an invertible feature mapping, such as used for certain support vector machine models. In an earlier work, the deployed system PDFrate—an online service for detecting PDF malware—is the subject of evasion attacks in Srndic & Laskov (2014). Given their work's focus on practical evasion attacks, the authors present a sub-taxonomy of attacks sharing the same goal, distinguishing between cases of attacker information around knowledge of the feature set, training data, and details of the classifier.

*Attack Generation by Gradient Descent*

Computing the best response in the game-theoretic formulation corresponds to finding (near) optimal attacks under adversary-agnostic learners. An apparent advantage of this approach, over computing equilibria over both attacker and defender actions, is that more nonlinear models and learners can be tackled efficiently. Building on their work formulating *Causative* attacks as optimization (see Section 3.5.2), Biggio, Corona, Maiorca, Nelson, Srndic, Laskov, Giacinto, & Roli (2013) formulate *Exploratory* attacks on the support vector machine as gradient descent. By the Representer Theorem, the prediction function for the SVM can be written as a sum of kernel terms; hence gradients exist and are easily computed provided the kernel function is differentiable in the test point.

https://doi.org/10.1017/9781107338548.003 Published online by Cambridge University Press

#### 3.4.2.5 Evasion of Deep Learning Methods

There has been a great deal of interest and success in the field of deep neural network learners (for example see LeCun, Bengio, & Hinton 2015), which focuses on training multilayer (deep) neural networks (DNNs). Unlike prior neural network architectures, DNNs use cascades of hidden layers to implicitly undertake complex tasks such as feature extraction and transformation as part of the learning process. However, because of the large model size, DNNs are prone to overfitting and may be susceptible to evasion attacks. In particular Goodfellow, Shlens, & Szegedy (2015) demonstrated a simple *fast gradient sign method* for generating adversarial examples. Their work showed that even models with multiple nonlinear layers can be easily misled by applying linear perturbations of the test data. The authors observed that these attacks are *transferable*; i.e., can be applied to other target DNN models with different architectures that are used for the same learning task. Based on the transferability of these attacks, Papernot, McDaniel, Goodfellow, Jha, Celik, and Swami (2017) demonstrated black-box attacks against deep neural network systems in which the adversary is able to train a surrogate DNN model based on the output of the targeted DNN and craft adversarial examples for the surrogate model that also can evade the targeted model. Liu, Chen, Liu, & Song (2017) explore these transferable attacks at scale: for larger datasets and larger architectures better reflecting the state-of-the-art models used in industrial systems. They find that while indiscriminate attacks transfer easily, an ensembling technique is required to generate an attack example when transferring in the targeted case.

The success of deep learning and the risk of DNN model overfitting have triggered broad interest in adversarial learning. The application of adversarial learning as a form of regularization has enjoyed particular interest; we return to the topic in Section 3.4.4.1.

### 3.4.3 Exploratory Availability Attacks

In an *Exploratory Availability* attack, the attacker interferes with the normal behavior of a learning system without influence over training. This type of attack against non-learning systems abounds in the literature: almost any denial-of-service (DoS) attack falls into this category, such as those described by Moore et al. (2006). However, *Exploratory Availability* attacks against the learning components of systems are not common and we are not aware of any studies of them. It seems the motivation for attacks of this variety is not as compelling as for other attacks against learners.

One possible attack is described in the following example: if a learning IDS has trained on intrusion traffic and has the policy of blocking hosts that originate intrusions, an attacker could send intrusions that appear to originate from a legitimate host, convincing the IDS to block that host. Another possibility is to take advantage of a computationally expensive learning component: for example, spam filters that use image processing to detect advertisements in graphical attachments can take significantly more time than text-based filtering (Dredze, Gevaryahu, & Elias-Bachrach 2007; Wang, Josephson, Lv, Charikar, & Li 2007). An attacker could exploit such overhead by

https://doi.org/10.1017/9781107338548.003 Published online by Cambridge University Press

sending many emails with images, causing the time-consuming processing to delay and perhaps even block messages.

EXAMPLE 3.2 (The Mistaken Identity)
An attacker sends intrusions that appear to come from the IP address of a legitimate machine. The IDS, which has learned to recognize intrusions, blocks that machine. In the *Targeted* version, the attacker has a particular machine to target. In the *Indiscriminate* version, the attacker may select any convenient machine or may switch IP addresses among many machines to induce greater disruption.

### 3.4.4 Defending against Exploratory Attacks

*Exploratory* attacks do not corrupt the training data, but attempt to find vulnerabilities in the learned hypothesis. Through control over the evaluation data, the attacker can violate the assumption of stationarity. When producing the evaluation distribution, the attacker attempts to construct an unfavorable evaluation distribution that concentrates probability mass on high-cost instances; in other words, the attacker's procedure $A^{(\text{eval})}$ constructs an evaluation distribution $P_{\mathcal{Z}}^{(\text{eval})}$ on which the learner predicts poorly (violating stationarity); i.e., the attacker chooses $P_{\mathcal{Z}}^{(\text{eval})}$ to maximize the cost computed in the last step of the *Exploratory* game. This section examines defender strategies that make it difficult for the attacker to construct such a distribution.

In the *Exploratory* game, the defender makes a move before observing contaminated data; that is, here we do not consider scenarios where the defender is permitted to react to the attack. The defender can impede the attacker's ability to reverse engineer the classifier by limiting access to information about the training procedure and data. With less information, $A^{(\text{eval})}$ has difficulty producing an unfavorable evaluation distribution. Nonetheless, even with incomplete information, the attacker may be able to construct an unfavorable evaluation distribution using a combination of prior knowledge and probing.

The defender's task is to design data collection and learning techniques that make it difficult for an attacker to reverse engineer the hypothesis. The primary task in analyzing *Exploratory* attacks is quantifying the attacker's ability to reverse engineer the learner.

#### 3.4.4.1 Defenses against Attacks without Probing

Part of a security analysis involves identifying aspects of the system that should be kept secret. In securing a learner, the defender can limit information to make it difficult for an adversary to conduct its attack.

*Training Data*
Preventing the attacker from obtaining the training data limits the attacker's ability to reconstruct internal states of the classifier. There is a tension between collecting training data that fairly represents the real-world instances and keeping all aspects of that data secret. In most situations, it is difficult to use completely secret training data, though

https://doi.org/10.1017/9781107338548.003 Published online by Cambridge University Press

the attacker may have only partial information about it. Securing training data relates to privacy-preserving learning (see Section 3.7).

*Feature Selection*

The defender can also harden classifiers against attacks through attention to features in the feature selection and learning steps (which are both internal steps of the defender's hypothesis selection procedure $H^{(N)}$). Feature selection is the process of choosing a feature map that transforms raw measurements into the feature space used by the learning algorithm. In the learning step, the learning algorithm builds its model or signature using particular features from the map's feature space; this choice of features for the model or signature is also sometimes referred to as feature selection, though we consider it to be part of the learning process, after the feature map has been established. For example, one feature map for email message bodies might transform each token to a Boolean feature indicating its presence; another map might specify a real-valued feature indicating the relative frequency of each word in the message compared to its frequency in natural language; yet another map might count sequences of $n$ characters and specify an integer feature for each character $n$-gram indicating how many times it appears. In each of these cases, a learner will construct a model or signature that uses certain features (tokens present or absent; relative frequency of words present; character $n$-gram counts) to decide whether an instance is benign or malicious.

Obfuscation of spam-indicating words (an attack on the feature set) is a common *Targeted Exploratory Integrity* attack. Sculley et al. (2006) use inexact string matching in feature selection to defeat obfuscations of words in spam emails. They choose a feature map based on character subsequences that are robust to character addition, deletion, and substitution.

Globerson & Roweis (2006) present a feature-based learning defense for the feature deletion attack; an *Exploratory* attack on the evaluation data $\mathbb{D}^{(\text{eval})}$. In feature deletion, features present in the training data, and perhaps highly predictive of an instance's class, are removed from the evaluation data by the attacker. For example, words present in training emails may not occur in evaluation messages, and network packets in training data may contain values for optional fields that are missing from future traffic. Globerson & Roweis formulate a modified support vector machine classifier that is robust in its choice of features against deletion of high-value features.

It has been observed that high dimensionality serves to increase the attack surface of *Exploratory* attacks (Sommer & Paxson 2010; Amsaleg, Bailey, Erfani, Furon, Houle, Radovanović, & Vinh 2016), suggesting that (possibly randomized) feature selection be used as a defensive strategy. In game-theoretic models of *Causative* attacks, high dimensions also have computational consequences on finding equilibrium solutions. Alpcan, Rubinstein, & Leckie (2016) approach such settings through random projections, exploring conditions where solutions lift from projected spaces to the original action spaces; as a case study they apply their ideas to *Causative* attacks on the linear support vector machine.

The work of Li & Vorobeychik (2014), however, should bring a note of caution: in exploring traditional approaches to feature reduction in applications to email spam,

https://doi.org/10.1017/9781107338548.003 Published online by Cambridge University Press

they observe vulnerabilities to attackers using feature substitution that is particularly well motivated in spam. Next they explore counter-measures including learning equivalence classes of features and then using these in feature reduction to mitigate feature substitution; and formulation of the Stackelberg game as a bilevel mixed linear integer program, with heuristics for making approximate solutions tractable that yield an interesting trade-off between sparse regularized learning and evasion, thereby producing an approach to adversarial feature selection. Zhang, Chan, Biggio, Yeung, & Roli (2016) also explore the interplay between feature selection and attacks. The authors propose a wrapper-based adversarial feature selector that optimizes both classifier generalization capability and classifier security with optimization via greedy approaches: forward feature selection or backward feature elimination.

One particularly important consideration when the learner builds its model or signature is to ensure that the learner uses features related to the intrusion itself. In their study of the DARPA/Lincoln Laboratory intrusion dataset, Mahoney & Chan (2003) demonstrate that spurious artifacts in training data can cause an IDS to learn to distinguish normal from intrusion traffic based on those artifacts rather than relevant features. Ensuring that the learner builds a model from features that describe the fundamental differences between malicious and benign instances should mitigate the effects of mimicry attacks (Section 3.4.2) and red herring attacks (Section 3.5.2).

Using spurious features in constructing a model or signature is especially problematic in cases where any given intrusion attempt may cause harm only probabilistically or depending on some internal state of the victim's system. If the features relevant to the intrusion are consistent for some set of instances but the actual cost of those instances varies widely, then a learner risks attributing the variation to other nonessential features.

*Hypothesis Space/Learning Procedures*

A complex hypothesis space may make it difficult for the attacker to infer precise information about the learned hypothesis. However, hypothesis complexity must be balanced against the capacity to generalize, through appropriate regularization.

Wang, Parekh, & Stolfo (2006) present *Anagram*, an anomaly detection system using *n-gram* models of bytes to detect intrusions. They incorporate two techniques to defeat *Exploratory* attacks that mimic normal traffic (mimicry attacks): *i*) they use high-order *n*-grams (with *n* typically between 3 and 7), which capture differences in intrusion traffic even when that traffic has been crafted to mimic normal traffic on the single-byte level; and *ii*) they randomize feature selection by randomly choosing several (possibly overlapping) subsequences of bytes in the packet and testing them separately, so the attack will fail unless the attacker makes not only the whole packet but also any subsequence mimic normal traffic.

Dalvi et al. (2004) develop a cost-sensitive game-theoretic classification defense to counter *Exploratory Integrity* attacks. In their model, the attacker can alter natural instance features in $A^{(\text{eval})}$ but incurs a known cost for each change. The defender can measure each feature at a different known cost. Each has a known cost function over classification/true label pairs. The classifier $H^{(N)}$ is a cost-sensitive naive Bayes learner that classifies instances to minimize expected cost, while the attacker modifies features

https://doi.org/10.1017/9781107338548.003 Published online by Cambridge University Press

to minimize its own expected cost. The defense constructs an adversary-aware classifier by altering the likelihood function of the learner to anticipate the attacker's changes. The defender adjusts the likelihood that an instance is malicious by considering that the observed instance may be the result of an attacker's optimal transformation of another instance. This defense relies on two assumptions: *i*) the defender's strategy is a step ahead of the attacker's strategy (i.e., their game differs from ours in that the attacker's procedure $A^{(\text{eval})}$ cannot take $f$ into account), and *ii*) the attacker plays optimally against the original cost-sensitive classifier . It is worth noting that while the approach defends against optimal attacks, it does not account for nonoptimal attacks. For example, if the attacker does not modify any data, the adversary-aware classifier misclassifies some instances that the original classifier correctly classifies.

Some defensive methods for deep neural network learners in adversarial settings have been developed. Based on their *fast gradient sign method* for generating adversarial examples, Goodfellow et al. (2015) developed an alternative adversarial objective function for training DNN models. In this formulation, a regularizer is added to the original objective function. This regularizer is the objective function with an adversarial perturbation applied to each training instance; i.e., a gradient step in the opposing direction to the optimization. This regularizer transforms the objective into a minimax problem, which generally encourages flatter gradients in the neighborhood of the training data. An application of attacks on deep learners is represented by the framework of generative adversarial networks of Goodfellow, Pouget-Abadie, Mirza, Xu, Warde-Farley, Ozair, Courville, & Bengio (2014), whereby a pair of DNNs are trained: a generative (adversarial) model that captures the training data distribution and a discriminative model that is trained to discriminate between samples drawn from the training data and samples drawn from the generative network. The models together are trained via a minimax (zero-sum) game. Significant interest has followed from the approach's strong experimental results across several domains.

#### 3.4.4.2 Defenses against Probing Attacks

In the game described in Section 3.4.1, the attacker chooses an evaluation distribution $P_{\mathcal{Z}}^{(\text{eval})}$ for selecting the evaluation data $\mathbb{D}^{(\text{eval})}$ based on knowledge obtained from the training data $\mathbb{D}^{(\text{train})}$ and/or the classifier $f$. However, the procedure $A^{(\text{eval})}$ need not select a stationary distribution $P_{\mathcal{Z}}^{(\text{eval})}$. In fact, the attacker may incrementally change the distribution based on the observed behavior of the classifier to each data point generated from $P_{\mathcal{Z}}^{(\text{eval})}$—a *probing* or query-based adaptive attack. The ability for $A^{(\text{eval})}$ to query a classifier gives an attacker powerful additional attack options, which several researchers have explored.

##### *Analysis of Reverse Engineering*

Lowd & Meek (2005*a*) observe that the attacker need not model the classifier explicitly, but only find lowest-attacker-cost instances as in the setting of Dalvi et al. (2004). They formalize a notion of reverse engineering as the adversarial classifier reverse-engineering (ACRE) problem. Given an attacker cost function, they analyze the

https://doi.org/10.1017/9781107338548.003 Published online by Cambridge University Press

complexity of finding a lowest-attacker-cost instance that the classifier labels as negative. They assume no general knowledge of training data, though the attacker does know the feature space and also must have one positive example and one negative example. A classifier is ACRE-learnable if there exists a polynomial-query algorithm that finds a lowest-attacker-cost negative instance. They show that linear classifiers are ACRE-learnable with linear attacker cost functions and some other minor technical restrictions.

The ACRE-learning problem provides a means of quantifying how difficult it is to use queries to reverse engineer a classifier from a particular hypothesis class using a particular feature space. We now suggest defense techniques that can increase the difficulty of reverse engineering a learner.

*Randomization*

A randomized hypothesis may decrease the value of feedback to an attacker. Instead of choosing a hypothesis $f : \mathcal{X} \to \{0, 1\}$, we generalize to hypotheses that predict a real value on $[0, 1]$. This generalized hypothesis returns a probability of classifying $x \in \mathcal{X}$ as 1; i.e., a *randomized* classifier. By randomizing, the expected performance of the hypothesis may decrease on regular data drawn from a nonadversarial distribution, but it also may decrease the value of the queries for the attacker.

Randomization in this fashion does not reduce the information available in principle to the attacker, but merely requires more work from the attacker. It is likely that this defense is appropriate in only a small number of scenarios.

*Limiting/Misleading Feedback*

Another potential defense is to limit the feedback given to an attacker. For example, common techniques in the spam domain include eliminating bounce emails, delivery notices, and remote image loading and imposing other limits on potential feedback channels. In most settings, it is impossible to remove all feedback channels; however, limiting feedback increases work for the attacker. Moreover in some settings, it may also be possible to mislead the attacker by sending fraudulent feedback. Actively misleading the attacker by fabricating feedback suggests an interesting battle of information between attacker and defender. In some scenarios the defender may be able to give the attacker no information via feedback, and in others the defender may even be able to return feedback that causes the attacker to come to incorrect conclusions. Of course, misinformation can also degrade the usefulness of the classifier when evaluated on benign data.

## 3.5 Causative Attacks

The second broad category of attacks according to the taxonomy's INFLUENCE axis are *Causative* attacks, which influence the training data (as well as potentially subsequently modifying the evaluation data) as indicated in Figure 3.2. Again, the adversary's transformation $A^{(\text{eval})}$ alters the evaluation data either by defining a procedure to change instances drawn from $P_{\mathcal{Z}}$ or by changing $P_{\mathcal{Z}}$ to an alternative distribution $P_{\mathcal{Z}}^{(\text{eval})}$ chosen

https://doi.org/10.1017/9781107338548.003 Published online by Cambridge University Press

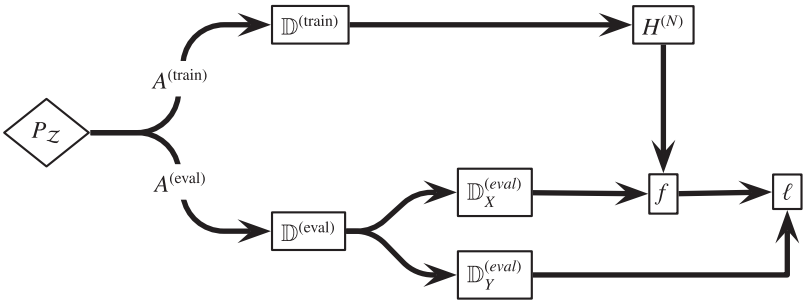


**Figure 3.2** Diagram of a *Causative* attack against a learning system (see Figure 2.1).

by the adversary (see Section 3.4). However, in addition to changing evaluation data, *Causative* attacks also allow the adversary to alter the training data with a second transformation $A^{(\text{train})}$, which either transforms instances drawn from $P_{\mathcal{Z}}$ or changes $P_{\mathcal{Z}}$ to an alternative distribution $P_{\mathcal{Z}}^{(\text{train})}$ during training. Of course, the adversary can synchronize $A^{(\text{train})}$ and $A^{(\text{eval})}$ to best achieve its desired objective, although in some *Causative* attacks, the adversary can only control the training data (e.g., the attacker we describe in Chapter 5 cannot control the non-spam messages sent during evaluation). Also note that, since the game described here corresponds to batch learning, an adaptive procedure $A^{(\text{train})}$ is unnecessary, although the distribution $P_{\mathcal{Z}}^{(\text{train})}$ can be nonstationary.

### 3.5.1 The Causative Game

The game for *Causative* attacks is similar to the game for *Exploratory* attacks with an augmented move for the attacker.

1 **Defender** Choose procedure $H^{(N)}$ for selecting hypothesis
2 **Attacker** Choose procedures $A^{(\text{train})}$ and $A^{(\text{eval})}$ for selecting distributions
3 Evaluation:
- Compute $P_{\mathcal{Z}}^{(\text{train})} \leftarrow A^{(\text{train})}\left(P_{\mathcal{Z}}, H\right)$
- Sample dataset $\mathbb{D}^{(\text{train})}$ from $P_{\mathcal{Z}}^{(\text{train})}$
- Compute $f \leftarrow H^{(N)}\left(\mathbb{D}^{(\text{train})}\right)$
- Compute $P_{\mathcal{Z}}^{(\text{eval})} \leftarrow A^{(\text{eval})}\left(\mathbb{D}^{(\text{train})}, f\right)$
- Sample dataset $\mathbb{D}^{(\text{eval})}$ from $P_{\mathcal{Z}}^{(\text{eval})}$
- Assess total cost: $\sum_{(x,y)\in\mathbb{D}^{(\text{eval})}} \ell_x\left(f(x), y\right)$

This game is very similar to the *Exploratory* game, but the attacker can choose $A^{(\text{train})}$ to affect the training data $\mathbb{D}^{(\text{train})}$. The attacker may have various types of influence over the data, ranging from arbitrary control over some fraction of instances to a small biasing influence on some aspect of data production; details depend on the setting. Again, the loss function $\ell_x(\cdot, \cdot)$ allows for instance-dependent costs.

https://doi.org/10.1017/9781107338548.003 Published online by Cambridge University Press

Control over data used for training opens up new strategies to the attacker. Cost is based on the interaction of $f$ and $\mathbb{D}^{(\text{eval})}$. In the *Exploratory* game the attacker chooses $\mathbb{D}^{(\text{eval})}$ while the defender controls $f$; in the *Causative* game the attacker also has influence on $f$. With this influence, the attacker can proactively cause the learner to produce bad classifiers.

*Contamination in PAC Learning*

Kearns & Li (1993) extend Valiant's *probably approximately correct* (PAC) learning framework (cf. Valiant 1984, 1985) to prove bounds for maliciously chosen errors in the training data. In PAC learning, an algorithm succeeds if it can, with probability at least $1 - \delta$, learn a hypothesis that has at most probability $\epsilon$ of making an incorrect prediction on an example drawn from the same distribution. Kearns & Li examine the case where an attacker has arbitrary control over some fraction $\beta$ of the training examples (this specifies the form that $A^{(\text{train})}$ takes in our *Causative* game). They prove that in general the attacker can prevent the learner from succeeding if $\beta \geq \epsilon/(1 + \epsilon)$, and for some classes of learners they show this bound is tight.

This work provides important limits on the ability to succeed at PAC learning in a particular adversarial setting. The analysis broadly concerns both *Integrity* and *Availability* attacks as well as both *Targeted* and *Indiscriminate* variants. However, not all learning systems fall into the PAC learning model.

### 3.5.2 Causative Integrity Attacks

In these attacks, the adversary actively attempts to corrupt the learning mechanism so that miscreant activities can take place that would be otherwise disallowed. In a *Causative Integrity* attack, the attacker uses control over training to cause intrusions to slip past the classifier as false negatives.

EXAMPLE 3.3 (The Intrusion Foretold)

An attacker wants the defender's IDS not to flag a novel virus. The defender trains periodically on network traffic, so the attacker sends non-intrusion traffic that is carefully chosen to look like the virus and mis-train the learner to fail to block it. This example would be *Targeted* if the attacker already has a particular virus executable to send and needs to cause the learner to miss that particular instance. It would be *Indiscriminate*, on the other hand, if the attacker has a certain payload but could use any of a large number of existing exploit mechanisms to transmit the payload, in which case the attack need only fool the learner on any one of the malicious executables.

*Red Herring Attack*

Newsome et al. (2006) present *Causative Integrity* and *Causative Availability* attacks against Polygraph (Newsome et al. 2005), a polymorphic-virus detector that learns virus signatures using both a conjunction learner and a naive-Bayes-like learner. Their *red herring* attacks against conjunction learners exploit certain weaknesses not generally present in other learning algorithms. The attack introduces

https://doi.org/10.1017/9781107338548.003 Published online by Cambridge University Press

spurious features along with their payload; once the learner constructs a signature, the spurious features are removed from subsequent malicious instances to evade the conjunction rules, which require all identified features to be present to match the signature learned by Polygraph. This attack allows the attacker to maintain a high false-negative rate even when retraining occurs because the attacker can introduce many spurious features and remove them incrementally. Conceptually, this attack corresponds to a transformation of $P_Z$ into $P_Z^{(\text{train})}$ and $P_Z^{(\text{eval})}$. This transformation introduces spurious features into all malicious instances that the defender uses for training. The malicious instances produced by $P_Z^{(\text{eval})}$, however, lack some of the spurious features and therefore bypass the filter, which erroneously generalized that all the spurious features were necessary elements of the malicious behavior. Venkataraman, Blum, and Song (2008) also present lower bounds for learning worm signatures based on red herring attacks.

#### Antidote

We also collaborated with our colleagues at Berkeley and Intel Labs to explore the vulnerability of network-wide traffic anomaly detectors based on principal component analysis (PCA) as introduced by Lakhina et al. (2004*b*). Our work examines how an attacker can exploit the sensitivity of PCA to form *Causative Integrity* attacks (Rubinstein, Nelson, Huang, Joseph, Lau, Rao, Taft, & Tygar 2009*a*). In anticipation of a DoS attack, the attacker systematically injects traffic to increase variance along the links of their target flow and mislead the anomaly detection system. We also studied how the projection pursuit-based robust PCA algorithm of Croux, Filzmoser, & Oliveira (2007) significantly reduces the impact of poisoning. We detail this work in Chapter 6.

### *Optimization Formulations*

Biggio, Nelson, & Laskov (2012) formulate *Causative* attacks on the support vector machine as optimization, leveraging work in incremental learning (Cauwenberghs & Poggio 2000) to determine contamination points to inject into the training set. The optimization is approximated via gradient descent. Attack as optimization has appeared many times in the literature since. The effect of feature selection on training robustness is examined by Xiao, Biggio, Brown, Fumera, Eckert, & Roli (2015), complementing the evaluation on *Exploratory* robustness by Li & Vorobeychik (2014), where it is demonstrated on malware samples for example that the performance of LASSO can be reduced to random choice with only 5% control over the training set. Li, Wang, Singh, & Vorobeychik (2016) explore *Causative* attacks on collaborative filtering methods also using gradient-descent based approaches, but for nonsmooth nuclear normed objectives, using alternating minimization and nuclear norm minimization.

Mei & Zhu (2015*b*) formulate early moves of the adversarial learning game for *Causative* attacks as a bilevel optimization where the defender learns in a lower-level optimization while the attacker contaminates training data at a top level. They show that under differentiability and convexity of the learner's objectives, the optimization can be reduced via KKT methods to a single-level optimization and use (projected) gradient descent to find attack training sets. Finally they draw connections to machine teaching

https://doi.org/10.1017/9781107338548.003 Published online by Cambridge University Press

and teaching dimension (Goldman & Kearns 1995), where training data is generated by a teacher guiding a learner to a predetermined hypothesis.

As an application of these ideas the authors explore the security of latent Dirichlet allocation (Mei & Zhu 2015*a*), demonstrating the promotion (demotion) of words to (from) topics. Alfeld, Zhu, & Barford (2016) also apply these attacks to autoregressive models for time series analysis, motivated by futures market manipulation in gas prices. The attacker possess a target forecast and optimizes poisoning data under the quadratic loss on forecasts achieved. The adversary is capable of perturbing past covariates at the time of attack optimization, modelling a scenario of "cooking the books" whereby past reporting may be manipulated. Hard and soft constraints on the attacker's perturbations are considered. While their examples are computed for weakly stationary models (where the first moment and covariance are stationary through time), their approach of convex optimization for finding contaminating data is general. Torkamani & Lowd (2013) consider adversarial learning for collective classification: a learning task where labels of instances may experience dependencies provided that related objects are more likely to have similar labels (associativity). They present a convex quadratic program formulation. Their experimental results show that in some cases techniques that increase robustness against attack also can lead to better non-attacked performance.

### 3.5.3 Causative Availability Attacks

This less common (but nonetheless well-motivated) attack attempts to corrupt the learning system to cause innocuous data to significantly be misclassified so as to disrupt normal system operation. In a *Causative Availability* attack, the attacker uses control over training instances to interfere with operation of the system, such as by blocking legitimate traffic.

EXAMPLE 3.4 (The Rogue IDS)
An attacker uses an intrusion detection system (IDS) to disrupt operations on the defender's network. The attacker wants traffic to be blocked so the destination does not receive it. The attacker generates attack traffic similar to benign traffic when the defender is collecting training data to train the IDS. When the learner retrains on the attack data, the IDS will start to filter away benign instances as if they were intrusions. This attack could be *Targeted* at a particular protocol or destination. On the other hand, it might be *Indiscriminate* and attempt to block a significant portion of all legitimate traffic.

#### *Allergy Attack*

Chung & Mok (2006, 2007) present *allergy* attacks against the Autograph worm signature generation system (Kim & Karp 2004). Autograph operates in two phases. First, it identifies infected nodes based on behavioral patterns, in particular scanning behavior. Second, it observes traffic from the suspect nodes and infers blocking rules based on observed patterns. Chung and Mok describe an attack that targets traffic to a particular

https://doi.org/10.1017/9781107338548.003 Published online by Cambridge University Press

resource. In the first phase, an attack node convinces Autograph that it is infected by scanning the network. In the second phase, the attack node sends crafted packets mimicking targeted traffic, causing Autograph to learn rules that block legitimate access and create a denial-of-service event.

In the context of the *Causative* game, the attacker's choice of $P_{\mathcal{Z}}^{(\text{train})}$ provides the traffic for both phases of Autograph's learning. When Autograph produces a hypothesis $f$ that depends on the carefully crafted traffic from the attacker, it will block access to legitimate traffic from $P_{\mathcal{Z}}^{(\text{eval})}$ that shares patterns with the malicious traffic.

*Correlated Outlier Attack*

Newsome et al. (2006) also suggest a *correlated outlier* attack against the Polygraph virus detector Newsome et al. (2005). This attack targets the naive-Bayes-like component of the detector by adding spurious features to positive training instances, causing the filter to block benign traffic with those features. As with the red herring attacks, these correlated outlier attacks fit neatly into the *Causative* game; this time $P_{\mathcal{Z}}^{(\text{train})}$ includes spurious features in malicious instances, causing $H^{(N)}$ to produce an $f$ that classifies many benign instances as malicious.

*Attacking SpamBayes*

In the spam filtering domain we also explored *Causative Availability* attacks against the SpamBayes statistical spam classifier (Nelson et al. 2008, Nelson, Barreno, Chi, Joseph, Rubinstein, Saini, Sutton, Tygar, & Xia 2009). In these attacks, we demonstrated that by sending emails containing entire dictionaries of tokens, the attacker can cause a significant fraction of normal email to be misclassified as spam with relatively little contamination (an *Indiscriminate* attack). Similarly, if an attacker can anticipate a particular target message, then the attacker can also poison the learner to misclassify the target as spam (a *Targeted* attack). We also investigated a principled defense to counter these *dictionary attacks*: the *reject on negative impact (RONI) defense*. We discuss this work in detail in Chapter 5.

*Attacking Malheur*

In the realm of unsupervised learners, the Malheur, an open-source behavioral malware clustering tool (Rieck, Trinius, Willems, & Holz 2011), has been found highly vulnerable to even very low levels of *Indiscriminate Causative Availability* attacks (Biggio, Rieck, Ariu, Wressnegger, Corona, Giacinto, & Roli 2014): DoS attacks against the malware clustering through poisoning of the data to which the clustering is fit. The authors avoid the feature inversion problem by essentially walking in the space of malware samples, performing only feature addition in a similar vein to dictionary attacks.

### 3.5.4 Defending against Causative Attacks

Most defenses presented in the literature of secure learning combat *Exploratory Integrity* attacks (as discussed earlier) while relatively few defenses have been presented

https://doi.org/10.1017/9781107338548.003 Published online by Cambridge University Press

to cope with *Causative* attacks. In *Causative* attacks, the attacker has a degree of control over not only the evaluation distribution but also the training distribution. Therefore the learning procedures we consider must be resilient against contaminated training data, as well as satisfy the evaluation considerations discussed in Section 3.4.4.

Two general strategies for defense are to remove malicious data from the training set and to harden the learning algorithm against malicious training data. We first present one method for the former and then describe two approaches to the latter that appear in the literature. The foundations of these approaches primarily lie in adapting game-theoretic techniques to analyze and design resilient learning algorithms.

### 3.5.4.1 The Reject on Negative Impact Defense

Insidious *Causative* attacks make learning inherently more difficult. In many circumstances, data sanitization may be the only realistic mechanism to achieve acceptable performance. For example, Nelson et al. (2009) introduce such a sanitization technique called *reject on negative impact*, a technique that measures the empirical effect of adding each training instance and discards instances that have a substantial negative impact on classification accuracy. To determine whether a candidate training instance is malicious or not, the defender trains a classifier on a base training set, then adds the candidate instance to the training set, and trains a second classifier. The defender applies both classifiers to a *quiz set* of instances with known labels and measures the difference in accuracy between the two classifiers. If adding the candidate instance to the training set causes the resulting classifier to produce substantially more classification errors, then the defender permanently removes the instance as detrimental in its effect. We refine and explore the reject on negative impact defense experimentally in Section 5.5.5.

### 3.5.4.2 Learning with Contaminated Data

Several approaches to learning under adversarial contamination have been studied in the literature. The effect of adversarial contamination on the learner's performance is incorporated into some existing learning frameworks. As outlined earlier, Kearns & Li (1993) extended the PAC learning model to allow for adversarial noise within the training data and bounded the amount of contamination a learner could tolerate. Separately, the field of robust statistics (Huber 1981; Hampel et al. 1986; Maronna, Martin, & Yohai 2006). has formalized adversarial contamination with a worst-case contamination model from which researchers derived criteria for designing and comparing the robustness of statistical procedures to adversarial noise. Research incorporated these robustness criteria with more traditional learning domains (Christmann & Steinwart 2004; Wagner 2004), but generally these techniques have not been widely incorporated within machine learning and even less so within security. We discuss this area further in the next section.

To derive secure kernel methods, Russu, Demontis, Biggio, Fumera, & Roli (2016) leverage results of Xu, Caramanis, & Mannor (2009) on the equivalence of support vector machine learning, with unregularized hinge-loss minimization under adversarial perturbations with size bounded by the dual norm of the original SVM problem. They upper-bound change to nonlinear SVM predicted values in terms of the change

https://doi.org/10.1017/9781107338548.003 Published online by Cambridge University Press

in a datum and the norm through which the change is measured. For sparse evasion attacks, the authors argue that Laplace kernels are more appropriate for defense, while for dense attacks the defender should employ the RBF kernel. The authors argue for choices of regularizer, kernel, and regularization parameters (potentially being class or example dependent), leveraging the connection between robustness and regularization parameters. Earlier, Torkamani & Lowd (2014) explored related questions for structured prediction models, specifically for the case of the structural SVM leveraging the connection between optimization robust to perturbation and appropriate regularization of a nonrobust learner. Here the space of structured outputs however is exponentially large.

Alfeld, Zhu, & Barford (2017) revisit their early work on *Causative* attacks on autoregressive models Alfeld et al. (2016) and consider defensive strategies based on bilevel optimization where the defender's action set includes linear projections that define an ellipse from which the attacker chooses a poisoning attack. By assuming a rational attacker in a zero-sum game, they frame the defense as a bilevel optimization that reduces to minimax. More generally the framework operates under discrete action sets. Under this framework they compute optimal defenses that significantly reduce defender loss in experiments on futures market datasets.

Another model of adversarial learning is based on the online expert learning setting (Cesa-Bianchi & Lugosi 2006). Rather than designing learners to be robust against adversarial contamination of well-behaved stationary, stochastic data, techniques here focus on regret minimization to construct aggregate learners that adapt to completely adversarial conditions. The objective of regret minimization techniques is to dynamically aggregate the decisions of many experts based on their past performance so that the composite learner does well with respect to the best single expert in hindsight. We discuss this set of techniques in Section 3.6.

#### 3.5.4.3 Robustness

The field of robust statistics explores procedures that limit the impact of a small fraction of deviant (adversarial) training data. In the setting of robust statistics, it is assumed that the bulk of the data is generated from a known well-behaved stochastic model, but a fraction of the data comes from an unknown adversarial model—the goal is to bound the effect of this adversarial data on statistical estimates. There are a number of measures of a procedure's robustness: the breakdown point is the level of contamination required for the attacker to arbitrarily manipulate the procedure, and the influence function measures the impact of contamination on the procedure. Robustness measures can be used to assess the susceptibility of an existing system and to suggest alternatives that reduce or eliminate the vulnerability. Ideally one would like to use a procedure with a high breakdown point and a bounded influence function. These measures can be used to compare candidate procedures and to design procedures $H^{(N)}$ that are optimally robust against adversarial contamination of the training data. Here we summarize these concepts, but for a full treatment of these topics, refer to the books by Huber (1981), Hampel et al. (1986), and Maronna et al. (2006).

https://doi.org/10.1017/9781107338548.003 Published online by Cambridge University Press

To motivate applications of robust statistics for adversarial learning, recall the traditional learning framework presented in Section 2.2. Particularly, in Section 2.2.4, we discuss selecting a hypothesis that minimizes the empirical risk. Unfortunately in adversarial settings, assumptions made by the learning model, such as stationarity leading to this empirical risk minimization, may be violated. Ideally one would hope that minor deviations from the modeling assumptions would not have a large impact on the optimal procedures that were derived under those assumptions. Unfortunately, this is not always the case—small adversarial deviations from the assumptions can have a profound impact on many real-world learning procedures. As stated by Tukey (1960),

> A tacit hope in ignoring deviations from ideal models was that they would not matter; that statistical procedures which were optimal under the strict model would still be approximately optimal under the approximate model. Unfortunately, it turned out that this hope was often drastically wrong; even mild deviations often have much larger effects than were anticipated by most statisticians.

These flaws can also be exploited by an adversary to mistrain a learning algorithm even when limited to a small amount of contamination. To avoid such vulnerabilities, one must augment the notion of optimality to include some form of *robustness* to the assumptions of the model; as defined by Huber (1981), "robustness signifies insensitivity to small deviations from the assumptions." There is, however, a fundamental tradeoff between the efficiency of a procedure and its robustness—this issue is addressed in the field of robust statistics.

The model used to assess the distributional robustness of a statistical estimator $H$ is known as the gross-error model, which is a mixture of the known distribution $F_{\mathcal{Z}}$ and some unknown distribution $G_{\mathcal{Z}}$ parameterized by some fraction of contamination $\epsilon$,

$$\mathcal{P}_\epsilon\left(F_{\mathcal{Z}}\right) \triangleq \{\,(1-\epsilon)F_{\mathcal{Z}} + \epsilon G_{\mathcal{Z}} \mid G_{\mathcal{Z}} \in \mathcal{P}_{\mathcal{Z}}\}$$

where $\mathcal{P}_{\mathcal{Z}}$ is the collection of all probability distributions on $\mathcal{Z}$. This concept of a contamination neighborhood provides for the *minimax approach* to robustness by considering a worst-case distribution within the gross-error model. Historically, the minimax approach yielded a robust class of estimators known as *Huber estimators*. Further it introduced the concept of a breakdown point $\epsilon^\star$—intuitively, the smallest level of contamination where the minimax asymptotic bias of an estimator becomes infinite.

An alternative approach is to consider the (scaled) change in the estimator $H$ due to an infinitesimal fraction of contamination. Again, consider the gross-error models and define a derivative in the direction of an infinitesimal contamination localized at a single point $z$. By analyzing the scaled change in the estimator due to the contamination, one can assess the *influence* that adding contamination at point $z$ has on the estimator. This gives rise to a functional known as the influence function and is defined as

$$\mathrm{IF}\left(z; H, F_{\mathcal{Z}}\right) \triangleq \lim_{\epsilon \to 0} \frac{H\left((1-\epsilon)F_{\mathcal{Z}} + \epsilon \Delta_z\right) - H\left(F_{\mathcal{Z}}\right)}{\epsilon}$$

where $\Delta_z$ is the distribution that has all its probability mass at the point $z$. This functional was derived for a wide variety of estimators and gives rise to several

https://doi.org/10.1017/9781107338548.003 Published online by Cambridge University Press

(infinitesimal) notions of robustness. The most prominent of these measures is the gross-error sensitivity defined as

$$\gamma^*(H, F_Z) \triangleq \sup_z |\mathrm{IF}(z; H, F_Z)|.$$

Intuitively, a finite gross-error sensitivity gives a notion of robustness to infinitesimal point contamination.

Research has highlighted the importance of robust procedures in security and learning tasks. Wagner (2004) observes that common sensor net aggregation procedures, such as computing a mean, are not robust to adversarial point contamination, and he identifies robust alternatives as a defense against malignant or failed sensors. Christmann & Steinwart (2004) study robustness for a general family of learning methods. Their results suggest that certain commonly used loss functions, along with proper regularization, lead to robust procedures with a bounded influence function. These results suggest such procedures have desirable properties for secure learning, which we return to in Section 9.1.

## 3.6 Repeated Learning Games

In Sections 3.4.1 and 3.5.1, the considered learning games are one-shot games, in which the defender and attacker minimize their cost when each move happens only once. We generalize these games to an iterated game, in which the players make a series of $K$ repetitions of the iterated *Causative* game with the ultimate aim of minimizing their total accumulated cost. We assume players have access to all information from previous iterations of the game, and grant the attacker unspecified (potentially arbitrary) control of the training data. At each iteration the defender produces a prediction after which it learns the true label and suffers some loss.

Evaluating the defender's absolute cumulative cost of playing—the analog of risk in the stochastic or PAC settings—is impossible due to the strongly adversarial nature of the data. Instead it is conventional to compare the accumulated cost incurred by the learner relative to the minimum cost achievable (in hindsight) by any one of $M$ experts—i.e., a set of classifiers each designed to provide different security properties. This relative (additive) performance measure is known as regret, since it represents the regret that the learner feels for not heeding the advice of the best expert in hindsight. The analogous multiplicative ratio of the learner's total cost to the minimum total cost of an expert is the related competitive ratio.

Most commonly, online learners form a composite prediction based on the advice of the experts. As such, the experts can be seen as providing advice to the defender (hence their name), who weighs the advice to produce the composite prediction; e.g., the aggregate prediction could be a weighted majority of the experts' predictions (Littlestone & Warmuth 1994). Further, at the end of each iteration, the defender uses the newly revealed true label to reweigh each expert based on the expert's predictive

https://doi.org/10.1017/9781107338548.003 Published online by Cambridge University Press

performance. No assumption is made about how the experts form their advice or about their performance; in fact, their advice may be adversarial and may incur arbitrary loss.

By developing algorithms with provable small regret, the composite predictor performs comparably to the best expert without knowing which one will be best a priori. By designing strategies that minimize regret, online learning provides an elegant mechanism to combine several predictors, each designed to address the security problem in a different way, into a single predictor that adapts relative to the performance of its constituents—all while facing arbitrarily adversarial data. As a result, the attacker must design attacks that are uniformly successful on the set of predictors rather than just on a single predictor because the composite learner can perform almost as well as the best without knowing ahead of time which expert to follow.

We now delve more deeply into the online learning setting, but for a full description and several regret minimizing algorithms see Cesa-Bianchi & Lugosi (2006).

As discussed earlier, the learner forms a prediction from the $M$ expert predictions and adapts its predictor $h^{(k)}$ based on their performance during $K$ repetitions. At each step $k$ of the game, the defender receives a prediction $\hat{y}^{(k,m)}$ from the $m^{\text{th}}$ expert[2] and makes a composite prediction $\hat{y}^{(k)}$ via $h^{(k)}$. After the defender's prediction is made, the true label $y^{(k)}$ is revealed, and the defender evaluates the instantaneous regret for each expert; i.e., the difference in the loss for the composite prediction and the loss for the $m^{\text{th}}$ expert's prediction. More formally, the $k^{\text{th}}$ round of the expert-based prediction game follows[3]:

1 **Defender** Update function $h^{(k)} : \mathcal{Y}^M \to \mathcal{Y}$
2 **Attacker** Choose distribution $P_{\mathcal{Z}}^{(k)}$
3 Evaluation:
- Sample an instance $\left(x^{(k)}, y^{(k)}\right) \sim P_{\mathcal{Z}}^{(k)}$
- Compute expert advice $\left\{\hat{y}^{(k,m)}\right\}_{m=1}^{M}$; e.g., $\hat{y}^{(k,m)} = f^{(m)}\left(x^{(k)}\right)$
- Predict $\hat{y}^{(k)} = h^{(k)}\left(\hat{y}^{(k,1)}, \hat{y}^{(k,2)}, \ldots, \hat{y}^{(k,M)}\right)$
- Compute instantaneous regret: $r^{(k,m)} = \ell\left(\hat{y}^{(k)}, y^{(k)}\right) - \ell\left(\hat{y}^{(k,m)}, y^{(k)}\right)$ for each expert $m = 1 \ldots M$

This game has a slightly different structure from the games we presented in Sections 3.4.1 and 3.5.1—here the defender chooses one strategy at the beginning of the game and then in each iteration updates the function $h^{(k)}$ according to that strategy. Based only on the past performance of each expert (i.e., the regrets observed over the previous $k-1$ iterations of the game), the defender chooses an online strategy for updating $h^{(k)}$ at the $k^{\text{th}}$ step of the game to minimize regret (Cesa-Bianchi & Lugosi

[2] An expert's advice may be based on the data, but the defender makes no assumption about how experts form their advice.

[3] We again assume that costs are symmetric for the defender and adversary and are represented by the loss function. Further, as in Section 2.2.4 we simplify the game to use the surrogate loss function used in place of a 0-1 loss: Finally, this game is also easily generalized to the case where several instances/labels are generated in each round.

https://doi.org/10.1017/9781107338548.003 Published online by Cambridge University Press

2006). The attacker, however, may select a new strategy at each iteration and can control the subsequent predictions made by each expert based on the defender's choice for $h^{(k)}$.

Finally, at the end of the game, the defender is assessed in terms of its worst-case regret $R^*$, which is defined in terms of the cumulative regret $R^{(m)}$ with respect to the $m^{\text{th}}$ expert as

$$R^{(m)} \triangleq \sum_{k=1}^{K} r^{(k,m)}$$
$$R^* \triangleq \max_m R^{(m)}. \tag{3.1}$$

If $R^*$ is small (relative to $K$), then the defender's aggregation algorithm has performed almost as well as the best expert without knowing which expert would be best. Further, as follows from the Equation (3.1) and the definition of instantaneous regret, the average regret is simply the difference of the risk of $h^{(k)}$ and the risk of $f^{(m)}$. If the average worst-case regret is small (i.e., approaches 0 as $K$ goes to infinity) and the best expert has small risk, the predictor $h^{(k)}$ also has a small risk. This motivates the study of regret minimization procedures. A substantial body of research has explored strategies for choosing $h^{(k)}$ to minimize regret in several settings.

Online expert-based prediction splits risk minimization into two subproblems: (*i*) minimizing the risk of each expert and (*ii*) minimizing the average regret; that is, as if we had known the best predictor $f^{(\star)}$ before the game started and had simply used its prediction at every step of the game. The defenses we have discussed approach the first problem. Regret minimization techniques address the second problem. For certain variants of the game, there exist composite predictors whose regret is $o(K)$—that is, the average regret approaches 0 as $K$ increases. This effectively allows the defender to use several strategies simultaneously and forces the attacker to design attacks that do well against them all.

### 3.6.1 Repeated Learning Games in Security

Thus far the game-theoretic form of online learning outlined earlier has had relatively little application to security. While naturally aimed at managing a form of worst-case risk, these techniques have traditionally only been applied to mitigating risk in finance such as for universal portfolio management (Cover 1991; Helmbold, Singer, Schapire, & Warmuth 1998; Kalai & Vempala 2002).

There has been slowly growing interest in online learning within the security community, however. Our work with collaborators at Berkeley and Stanford applies online learning to develop a reactive risk management strategy for an abstract Chief Information Security Officer (CISO) (Barth, Rubinstein, Sundararajan, Mitchell, Song, & Bartlett 2012). In this game, the CISO must defend against a powerful adversary who can penetrate the CISO's organization passing from one state of intrusion to another (nodes in an attack graph) via various sets of actions (edges or hyperedges in general). The defender incurs certain costs when the attacker reaches certain nodes, but

https://doi.org/10.1017/9781107338548.003 Published online by Cambridge University Press

the defender can apply a limited defense budget to the graph's edges, forcing a cost on the attacker proportional to the applied budget. The defender faces a seemingly up-hill battle, since it is never made aware of the attacker's node payoffs and is only made aware of the graph's structure after it is first attacked. However we show that under the adaptive CISO, the attacker's return on investment/profit always approaches the attacker's ROI/profit under an optimal (minimax) fixed defensive strategy corresponding to proactive risk management. Further we show that in many realistic settings the adaptive defender performs significantly better. That is, in an abstract setting reactive security at worst approaches the performance of proactive security, or at best dominates it. Our algorithm and analysis draw heavily on existing results in online learning theory. Within the context of our present secure learning taxonomy, the adversary essentially aims to achieve *Causative Integrity* attacks on the defender, although the learner is not strictly performing binary classification.

Blocki, Christin, Datta, & Sinha (2011) have also applied online learning theory to security research. Their work proposes a learning-theoretic foundation for audit mechanisms, where the notion of regret in online learning theory is applied to define a desirable property of an adaptive audit mechanism: its cost (consisting of the numbers and types of transaction inspections performed and the cost of brand degradation due to missing violations that are detected by external agencies) should approach that of a hypothetical auditor employing an optimal fixed strategy. The authors develop an adaptive audit mechanism that provably asymptotically minimizes regret, with fast convergence.

Seeding online learning approaches with the results of computing equilibrium strategies—rather than starting with poor-performing uniformly random policies—is an idea proposed and explored in Klíma, Lisỳ, & Kiekintveld (2015). Potentially such hybrid approaches could enjoy the benefits of both communities.

Repeated learning games clearly possess the potential for suggesting useful adaptive algorithms for security-sensitive settings, where regret minimizing algorithms are suited to playing against powerful adversaries by design. Existing research is a step in the right direction, where defenses are designed to stand up against truly active adversaries. However, more work is needed to apply these ideas to other domains within security, and empirical research is needed to assess these methods in less idealized real-world settings.

## 3.7 Privacy-Preserving Learning

The aim of privacy-preserving learning is to release aggregate statistics, or the results of machine learning, on a dataset without disclosing local information about individual data elements. In the language of our taxonomy, privacy-preserving learning should be robust to *Exploratory* or *Causative* attacks which aim to violate *Privacy*. An attacker with access to a released statistic, model, or classifier may probe it in an attempt to reveal information about the training data (an *Exploratory Privacy* attack); moreover an attacker with influence over some proportion of the training examples may attempt to

https://doi.org/10.1017/9781107338548.003 Published online by Cambridge University Press

manipulate the mechanism into revealing information about unknown training examples (a *Causative Privacy* attack). In this way the *Privacy* goal represents an important extension of the security goals for machine learning systems considered by the taxonomy originally proposed by Barreno et al. (2006).

This section outlines the current state of the art in defending against *Privacy* attacks: most publicized privacy breaches are not strictly violations of a *Privacy* goal of an adaptive system, but are often linkage attacks or releases of sensitive data following the violation of a system's integrity (Narayanan & Shmatikov 2008; Sweeney 2002; Barbaro & Zeller 2006; Homer, Szelinger, Redman, Duggan, Tembe, Muehling, Pearson, Stephan, Nelson, & Craig 2008). As intimated earlier, we treat the *Privacy* goal separately here from the earlier discussion of misclassification attacks (*Integrity* or *Availability*) based around INFLUENCE and SPECIFICITY. While these axes apply equally well to categorizing *Privacy* attacks, the leading formalization for preserving privacy (i.e., defenses) provides for very strong theoretical guarantees spanning all levels of these axes.

Next we outline the leading measure of privacy preservation known as differential privacy, discuss how the definition provides for certain quantifiable protections against both *Exploratory* and *Causative* attacks, and describe existing research into the inherent tradeoffs between a learner's statistical utility and the level of training data privacy it provides.

### 3.7.1 Differential Privacy

Historically, formal measures for quantifying the level of privacy preserved by a data analysis or data release have been elusive. Numerous definitions have been proposed and put aside due to the propositions being of a syntactic rather than semantic nature, most notably $k$-anonymity and its variants (Sweeney 2002; Machanavajjhala, Kifer, Gehrke, & Venkitasubramaniam 2007). However, the concept of differential privacy due to Dwork et al. (2006) has emerged as a strong guarantee of privacy, with formal roots influenced by cryptography. This definition has enjoyed a significant amount of interest in the theory community (Dinur & Nissim 2003; Blum, Dwork, McSherry, & Nissim 2005; Dwork et al. 2006; Dwork 2006; Barak, Chaudhuri, Dwork, Kale, McSherry, & Talwar 2007; Blum, Ligett, & Roth 2008; Dwork, Naor, Reingold, Rothblum, & Vadhan 2009; Dwork, McSherry, & Talwar 2007; McSherry & Talwar 2007; Kasiviswanathan, Lee, Nissim, Raskhodnikova, & Smith 2008; Dwork & Yekhanin 2008; Dwork & Lei 2009; Beimel, Kasiviswanathan, & Nissim 2010; Hardt & Talwar 2010; Smith 2011; Hardt, Ligett, & Mcsherry 2012; Duchi, Jordan, & Wainwright 2013; Bassily, Smith, & Thakurta 2014) where the general consensus is that the formal definition is meaningful and appropriately strong, while allowing for statistical learning methods that preserve the notion of privacy to be of practical use (Rubinstein, Bartlett, Huang, & Taft 2009; McSherry & Mironov 2009; Barak et al. 2007; Kasiviswanathan et al. 2008; Dinur & Nissim 2003; Dwork & Yekhanin 2008, Machanavajjhala, Kifer, Abowd, Gehrke, & Vilhuber 2008; Beimel et al. 2010; Hardt & Talwar 2010; Chaudhuri, Monteleoni, & Sarwate 2011; Hardt et al. 2012; Cormode, Procopiuc, Srivastava, Shen, & Yu 2012; Zhang,

https://doi.org/10.1017/9781107338548.003 Published online by Cambridge University Press

Zhang, Xiao, Yang, & Winslett 2012; Li, Hay, Miklau, & Wang 2014; He, Cormode, Machanavajjhala, Procopiuc, & Srivastava 2015; Wang, Fienberg, & Smola 2015) .We now recall the definition of differential privacy before discussing its prevailing features in the current context of adversarial machine learning. We follow the terminology that is most common in works on differential privacy and is influenced by roots in statistical databases research; the parallels to the existing notation and terminology introduced in Chapter 2 will be clear. A comprehensive treatment of differential privacy is provided by Dwork & Roth (2014).

A *database* $\mathbb{D}$ is a sequence of *rows* $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}$ that are typically binary or real vectors but could belong to any domain $\mathcal{X}$. Given access to $\mathbb{D}$, a *mechanism* $M$ is tasked with releasing aggregate information about $\mathbb{D}$ while maintaining the privacy of individual rows. In particular we assume that the *response* $M(\mathbb{D}) \in \mathcal{T}_M$ is the only information released by the mechanism. This response could be a scalar statistic on $\mathbb{D}$, such as a mean, median, or variance, or a model such as the parameters to an estimated joint density or the weight vector to a learned classifier. We say that a pair of databases $\mathbb{D}^{(1)}, \mathbb{D}^{(2)}$ are *neighbors* if they differ on one row. With these definitions in hand we can describe the following formal measure of privacy due to Dwork et al. (2006).

DEFINITION 3.1 For any $\epsilon > 0$, a randomized mechanism $M$ achieves $\epsilon$-*differential privacy* if, for all pairs of neighboring databases $\mathbb{D}^{(1)}, \mathbb{D}^{(2)}$ and all measurable subsets of responses $T \subseteq \mathcal{T}_M$ the mechanism satisfies[4]

$$\Pr\left(M\left(\mathbb{D}^{(1)}\right) \in T\right) \leq \exp(\epsilon)\Pr\left(M\left(\mathbb{D}^{(2)}\right) \in T\right).$$

To understand this definition, consider a differentially private mechanism $M$ that preserves data privacy by adding noise to the response of some desired nonprivate deterministic statistic $S(\mathbb{D})$, say the average $N^{-1}\sum_{i=1}^{N}\mathbf{x}^{(i)}$ of a sequence of $N$ scalars $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}$. The definition compares the distributions of $M$'s noisy mean responses, when one scalar $\mathbf{x}^{(i)}$ (a database row) is changed. If the definition holds for privacy level $\epsilon \ll 1$, then the likelihood of $M$ responding with noisy mean $t$ on database $\mathbb{D}^{(1)}$ is exceedingly close to the likelihood of responding with the same $t$ on database $\mathbb{D}^{(2)}$ with perturbed $\mathbf{x}^{(i)}$: the mechanism's response distributions on the two *neighboring* databases are pointwise close.

EXAMPLE 3.5 (Private Support Vector Machine Learning)
As a more practical example we have previously studied differentially private mechanisms for support vector machine (SVM) learning with collaborators from Berkeley and Intel (Rubinstein, Bartlett, Huang, & Taft 2009). There the setting is again a private database on which we wish to perform inference. However, the database is now composed of rows of feature vectors and binary labels, making up a training set of supervised binary classification. The desired inference is now the more sophisticated task of

[4] The probabilities in the definition are over the randomization of mechanism $M$, not over the databases, which are fixed.

https://doi.org/10.1017/9781107338548.003 Published online by Cambridge University Press

SVM learning (Cristianini & Shawe-Taylor 2000): in the linear case we find a hyperplane normal vector that maximizes margin on the training set, and in the nonlinear case we perform this margin maximization in a high-dimensional feature space induced by a user-defined kernel function. The mechanism here responds with the weight vector representing the learned classifier itself; the response is the parameterization of a function. Our mechanism for linear SVM simply adds Laplace noise to the weight vector, which we prove achieves differential privacy. For the nonlinear case we first solve linear SVM in a random feature space with an inner product approximating the desired kernel before adding noise to the corresponding solution; this first step allows us to achieve differential privacy even for kernels such as the radial basis function (RBF) that corresponds to learning in an infinite-dimensional feature space. Another approach to differentially private SVM learning is due to Chaudhuri et al. (2011), who instead of adding noise to the solution of SVM learning, randomly perturb the optimization used for SVM learning itself. We discuss privacy-preserving SVM learning in detail in Chapter 7.

Numerous other practical algorithms have been made differentially private, including regularized logistic regression (Chaudhuri & Monteleoni 2009), several collaborative filtering algorithms (McSherry & Mironov 2009), point estimation (Smith 2011), nearest neighbor, histograms, perceptron (Blum et al. 2005), range queries over databases with data structures such as KD trees (Li et al. 2014; He et al. 2015; Cormode et al. 2012), Bayesian probabilistic inference (Dimitrakakis, Nelson, Mitrokotsa, & Rubinstein 2014; Zhang, Rubinstein, & Dimitrakakis 2016; Wang et al. 2015), function release (Zhang et al. 2012; Aldà & Rubinstein 2017), and more.

### 3.7.2 Exploratory and Causative Privacy Attacks

An important observation on differential privacy is that the definition provides for very strong, semantic guarantees of privacy. Even with knowledge of $M$ up to randomness and with knowledge of the first $N-1$ rows of $\mathbb{D}$, an adversary cannot learn any additional information on the $N^{\text{th}}$ row from a sublinear (in $N$) sample of $M(\mathbb{D})$. The adversary may even attempt a brute-force *Exploratory* attack with such auxiliary information and unbounded computational resources:

1 For each possible $\hat{\mathbf{x}}^{(N)}$ consider $\mathbb{D}' = \mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N-1)}, \hat{\mathbf{x}}^{(N)}$ neighboring database $\mathbb{D}$.
   - Offline: Calculate the response distribution $p_{\mathbb{D}'}$ of $M(\mathbb{D}')$ by simulation.
2 Estimate the distribution of $M(\mathbb{D})$ as $\hat{p}_{\mathbb{D}}$ by querying the mechanism repeatedly (a sublinear number of times).
3 Identify $\mathbf{x}^{(N)} = \hat{\mathbf{x}}^{(N)}$ by the $p_{\mathbb{D}'}$ most closely resembling $\hat{p}_{\mathbb{D}}$.

However, for high levels of privacy (sufficiently small $\epsilon$), the sampling error in $\hat{p}_{\mathbb{D}}$ will be greater than the differences between alternate $p_{\mathbb{D}'}$, and so even this powerful brute-force exploratory attack will fail with high probability. The same robustness holds

https://doi.org/10.1017/9781107338548.003 Published online by Cambridge University Press

even in the setting of the analogous *Causative* attack, where the adversary can arbitrary manipulate the first $N - 1$ rows.

### 3.7.3 Utility despite Randomness

The more a target nonprivate estimator is randomized, the more privacy is preserved, but at a cost to utility. Several researchers have considered this inherent tradeoff between privacy and *utility*.

In our work on differentially private SVM learning (see Chapter 7), we define the utility of our private mechanism to be the pointwise difference between released privacy-preserving classifiers and nonprivate SVM classifiers. A private classifier (trained on $D$) that, with high probability yields very similar classifications to an SVM (trained on $D$), for all test points, is judged to be of high utility since it well approximates the desired nonprivate SVM classifier. Similar notions of utility are considered by Barak et al. (2007) when releasing contingency tables whose marginals are close to true marginals; Blum et al. (2008) whose mechanism releases anonymized data on which a class of analyses yield similar results to the original data; and Kasiviswanathan et al. (2008) and Beimel et al. (2010) who consider utility as corresponding to PAC learning where response and target concepts learned on sensitive data are averaged over the underlying measure. Others such as Chaudhuri & Monteleoni (2009) and Chaudhuri et al. (2011) measure the utility of a differential private mechanism not by its approximation of a target nonprivate algorithm, but rather by the absolute error it achieves. In all of these works, the differentially private mechanism is analyzed with the chosen utility in mind to produce an upper bound on the utility achieved by that particular mechanism.

Fundamental limits on the tradeoff between differential privacy and utility have also been of great interest in past work, through negative results (lower bounds) that essentially state that mechanisms cannot achieve both high levels of privacy preservation and utility simultaneously. In our work on differentially private SVM learning we establish lower bounds for approximating both linear and RBF SVM learning with any differentially private mechanism, quantifying levels of differential privacy and utility that cannot be achieved together. Dinur & Nissim (2003) show that if the noise of rate only $o\left(\sqrt{N}\right)$ is added to subset sum queries on a database $\mathbb{D}$ of bits, then an adversary can reconstruct a $1 - o(1)$ fraction of $\mathbb{D}$: if accuracy is too great, then privacy cannot be guaranteed at all. Hardt & Talwar (2010) and Beimel et al. (2010) conducted further studies establishing upper and lower bounds for the tradeoff between utility and privacy in respective settings where the mechanism responds with linear transformations of data and in the setting of private PAC learning. Generic approaches to establish lower bounds in differential privacy (for example, using volumetric packing arguments) are summarized by De (2012).

Moreover it is noteworthy that lower bounds, such as the above in theoretical differential privacy research, constitute powerful *Privacy* attacks that achieve guaranteed results on *any* privacy-preserving learner.

https://doi.org/10.1017/9781107338548.003 Published online by Cambridge University Press

While significant progress has been made in achieving differential privacy and utility, understanding connections between differential privacy and learnability Beimel et al. (2010), algorithmic stability (Rubinstein, Bartlett, Huang, & Taft 2009; Wang, Lei, & Fienberg 2016), robust statistics (Dwork & Lei 2009), and even mechanism design (McSherry & Talwar 2007), many open problems remain in finding more complete understandings of these connections, making practical learning algorithms differentially private, and understanding the tradeoff between privacy and utility.

https://doi.org/10.1017/9781107338548.003 Published online by Cambridge University Press