Secure Learning and Learning for Security: Research in the Intersection

Benjamin I. P. Rubinstein

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2010-71
http://www.eecs.berkeley.edu/Pubs/TechRpts/2010/EECS-2010-71.html

May 13, 2010

Secure Learning and Learning for Security: Research in the Intersection

by

Benjamin Rubinstein

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

and the Designated Emphasis

in

Communication, Computation, and Statistics

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Peter L. Bartlett, Chair
Professor Anthony D. Joseph
Professor Andrew E. B. Lim

Spring 2010

The dissertation of Benjamin Rubinstein, titled Secure Learning and Learning for Security: Research in the Intersection, is approved:

Chair _____ Date _____

_____ Date _____

_____ Date _____

University of California, Berkeley

Secure Learning and Learning for Security: Research in the Intersection

Abstract

Secure Learning and Learning for Security: Research in the Intersection

by

Benjamin Rubinstein

Doctor of Philosophy in Computer Science
and the Designated Emphasis in Communication, Computation, and Statistics

University of California, Berkeley

Professor Peter L. Bartlett, Chair

Statistical Machine Learning is used in many real-world systems, such as web search, network and power management, online advertising, finance and health services, in which adversaries are incentivized to attack the learner, motivating the urgent need for a better understanding of the security vulnerabilities of adaptive systems. Conversely, research in Computer Security stands to reap great benefits by leveraging learning for building adaptive defenses and even designing intelligent attacks on existing systems. This dissertation contributes new results in the intersection of Machine Learning and Security, relating to both of these complementary research agendas.

The first part of this dissertation considers Machine Learning under the lens of Computer Security, where the goal is to learn in the presence of an adversary. Two large case-studies on email spam filtering and network-wide anomaly detection explore adversaries that manipulate a learner by poisoning its training data. In the first study, the False Positive Rate (FPR) of an open-source spam filter is increased to 40% by feeding the filter a training set made up of 99% regular legitimate and spam messages, and 1% *dictionary attack* spam messages containing legitimate words. By increasing the FPR the adversary affects a Denial of Service attack on the filter. In the second case-study, the False Negative Rate of a popular network-wide anomaly detector based on Principal Components Analysis is increased 7-fold (increasing the attacker's chance of subsequent evasion by the same amount) by a *variance injection attack* of chaff traffic inserted into the network at training time. This high-variance chaff traffic increases the traffic volume by only 10%. In both cases the effects of increasing the information or the control available to the adversary are explored; and effective counter-measures are thoroughly evaluated, including a method based on Robust Statistics for the network anomaly detection domain.

The second class of attack explored on learning systems, involves an adversary aiming to evade detection by a previously-trained classifier. In the *evasion problem* the attacker searches for a negative instance of almost-minimal distance to some target positive, by submitting a small number of queries to the classifier. Efficient query algorithms are developed for almost-minimizing L_p cost over any classifier partitioning feature space into two classes, one of which is convex. For the case of a convex positive class and $p \leq 1$, algorithms with

linear query complexity are provided, along with lower bounds that almost match; when $p > 1$ a threshold phenomenon occurs whereby exponential query complexity is necessary for good approximations. For the case of a convex negative class and $p \geq 1$, a randomized Ellipsoid-based algorithm finds almost-minimizers with polynomial query complexity. These results show that learning the decision boundary is sufficient, but not necessary for evasion, and can require much greater query complexity.

The third class of attack aims to violate the confidentiality of the learner's training data given access to a learned hypothesis. Mechanisms for releasing Support Vector Machine (SVM) classifiers are developed. Algorithmic stability of the SVM is used to prove that the mechanisms preserve differential privacy, meaning that for an attacker with knowledge of all but one training example and the learning map, very little can be determined about the final unknown example using access to the trained classifier. Bounds on utility are established for the mechanisms: the privacy-preserving classifiers' predictions should approximate the SVM's predictions with high probability. In the case of learning with translation-invariant kernels corresponding to infinite-dimensional feature spaces (such as the RBF kernel), a recent result from large-scale learning is used to enable a finite encoding of the SVM while maintaining utility and privacy. Finally lower bounds on achievable differential privacy are derived for any mechanism that well-approximates the SVM.

The second part of this dissertation considers Security under the lens of Machine Learning. The first application of Machine Learning is to a learning-based reactive defense. The *CISO risk management problem* is modeled as a repeated game in which the defender must allocate security budget to the edges of a graph in order to minimize the additive profit or return on attack (ROA) enjoyed by an attacker. By reducing to results from Online Learning, it is shown that the profit/ROA from attacking the reactive strategy approaches that of attacking the best fixed proactive strategy over time. This result contradicts the conventional dogma that reactive security is usually inferior to proactive risk management. Moreover in many cases, it is shown that the reactive defender greatly outperforms proactive approaches.

The second application of Machine Learning to Security is for the construction of an attack on open-source software systems. When an open-source project releases a new version of their system, they disclose vulnerabilities in previous versions, sometimes with pointers to the patches that fixed them. Using features of diffs in the project's open-source repository, labeled by such disclosures, an attacker can train a model for discriminating between security patches and non-security patches. As new patches land in the open-source repository, before being disclosed as security or not, and before being released to users, the attacker can use the trained model to rank the patches according to likelihood of being a security fix. The adversary can then examine the ordered patches one-by-one until finding a security patch. For an 8 month period of Firefox 3's development history it is shown that an SVM-assisted attacker need only examine one or two patches per day (as selected by the SVM) in order to increase the aggregate window of vulnerability by 5 months.

Dedicated to my little Lachlan.

Contents

List of Figures

List of Tables

List of Algorithms

Acknowledgments

I am immensely grateful to my advisor Peter Bartlett who has been a constant positive force during my time at Berkeley. From making time for regular meetings, to wisely steering the course of my academic career, Peter has shown great patience, encouragement and support since I arrived in the U.S. six short years ago. I have been extremely fortunate to work with several other brilliant faculty and researchers at Berkeley, including Adam Barth, Ling Huang, Anthony Joseph, Satish Rao, Dawn Song, Nina Taft, and Doug Tygar. I feel privileged to have worked with each of these scholars, and I greatly appreciate the time we have spent together and the advice they have offered me on both academia and life in general.

I would like to especially thank fellow-grads Marco Barreno and Blaine Nelson with whom I have worked very closely on several of the projects presented here, and whom I also fondly regard as friends. They each have had particularly strong influences on my thinking about Adversarial Learning and Security, for which I am deeply grateful.

My research has also greatly benefitted by working with many others including Fuching Jack Chi, Arpita Ghosh, Shing-hon Lau, Steven Lee, Saung Li, John C. Mitchell, Ali Rahimi, Udam Saini, Fernando Silveira, Mukund Sundararajan, Charles Sutton, Anthony Tran, Sergei Vassilvitskii, Kai Xia, and Martin Zinkevich; and many helpful discussions with member's and visitors of Peter's group including Jake Abernethy, Alekh Agarwal, Fares Hedayati, Matti Kääriäinen, Marius Kloft, Joe Neeman, Anh Pham, David Rosenberg, Ambuj Tewari, and Mikhail Traskin.

During the course of my doctoral studies I have been either directly or indirectly supported by several funding agencies. I would like to take this opportunity to gratefully acknowledge the support of the Siebel Scholars Foundation; NSF awards #DMS-0707060 and #DMS-0434383; support by TRUST (Team for Research in Ubiquitous Secure Technology), which receives support from the National Science Foundation (NSF award #CCF-0424422) and AFOSR (#FA9550-06-1-0244); RAD Lab, which receives support from California state MICRO grants (#06-148 and #07-012); DETERlab (cyber-DEfense Technology Experimental Research laboratory), which receives support from DHS HSARPA (#022412) and AFOSR (#FA9550-07-1-0501); and the following organizations: Amazon, BT, Cisco, DoCoMo USA Labs, EADS, ESCHER, Facebook, Google, HP, IBM, iCAST, Intel, Microsoft, NetApp, ORNL, Pirelli, Qualcomm, Sun, Symantec, TCS, Telecom Italia, United Technologies, and VMware.

My life during grad school has been filled with many ups but also some downs. My closest friends (the 'chums') at Berkeley have been a constant source of support during the tough times, and encouragement and entertainment during the better times. Our weekly dinners, 'lunch in the city', dollar scoops, game (*i.e.*, settlers) nights, karaoke, skiing, and many many other times shared together have made grad school that much better. Michael and Diane Holwill, and their extended families, have been fantastic in-laws, supporting my family in every way possible.

My parents Sue and Hyam have raised me, putting a great emphasis on the value of education. For this, and their love and support, I will be eternally grateful. Juliet who was my girlfriend in Melbourne at the end of high school is now my beautiful wife and graduating

with me from Berkeley EECS. I am incredibly proud to be Juliet's husband, and I love her dearly. She has helped me enjoy many good times, and to overcome incredibly difficult ones also. Our beautiful son Lachlan, who was with us for far too short a time, touched me in ways I cannot express with words. I hope that he would be proud of me; I will forever miss him.

To the many other friends not named here, I also thank you. I have enjoyed taking courses, helping out in the CSGSA, enjoying CS movie nights, cooking 'Australian-style' BBQ for you, or playing Birkball with you all.

Chapter 1

Introduction

'Where shall I begin, please your Majesty?' he asked. 'Begin at the beginning,' the King said, gravely, 'and go on till you come to the end: then stop.'
– Lewis Carroll

1.1 Research in the Intersection

The intersection of Machine Learning, Statistics and Security is ripe for research. Today Machine Learning and Statistics are used in an ever-increasing number of real-world systems, including web search (Agichtein et al., 2006; Arguello et al., 2009; Joachims, 2002), online advertising (Ciaramita et al., 2008; Ghosh et al., 2009; Immorlica et al., 2005), email spam filtering (Meyer and Whateley, 2004; Ramachandran et al., 2007; Robinson, 2003), anti-virus software (Kim and Karp, 2004; Newsome et al., 2005), power management (Bodik et al., 2009, 2010), network management (Bahl et al., 2007; Cheng et al., 2007; Kandula et al., 2008; Lakhina et al., 2004a; Lazarevic et al., 2003; Liao and Vemuri, 2002; Mukkamala et al., 2002; Soule et al., 2005; Zhang et al., 2005), finance (Agarwal et al., 2010; Hazan and Kale, 2010; Stoltz and Lugosi, 2005), and health (Baldi and Brunak, 2001; Brown et al., 2000; Sankararaman et al., 2009). In many of these systems, human participants are incentivized to game the system's adaptive component in an attempt to gain some advantage. It is important for the success of such applications of Machine Learning and Statistics, that practitioners quantify the vulnerabilities present in existing learning techniques, and have access to learning mechanisms designed to operate in adversarial environments. Viewing Machine Learning and Statistics through such a lens of Computer Security has the potential to yield significant impact on practice and fundamental understanding of adaptive systems. Indeed for many application areas of Machine Learning, Security and Privacy should be placed on the same level as more traditional properties such as statistical performance, computational efficiency, and model interpretability.

An equally fruitful exercise is to study Computer Security through a lens of Machine Learning. In Computer Security it is common practice for researchers to construct attacks exploiting security flaws in existing systems (Nature, 2010). When the protocol of *responsible*

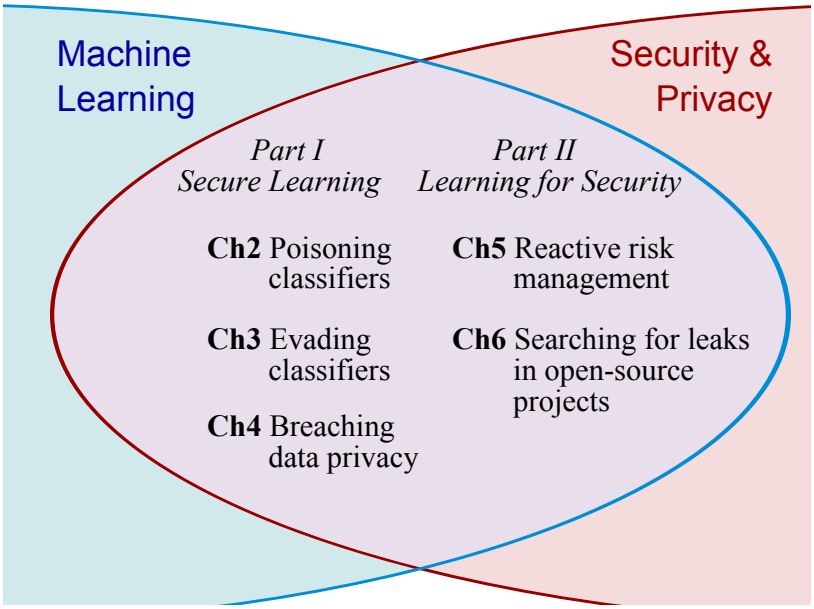


Figure 1.1: Organization of this dissertation's chapters into two related parts.

disclosure is followed—whereby a new vulnerability is not publicized until the development team has had the opportunity to patch the affected system—research into attacks can benefit both the developers and legitimate users of a system. As cyber-criminals are becoming emboldened by ever-more sophisticated attacks, it is now necessary for Security researchers to consider how Machine Learning and Statistics might be leveraged for constructing intelligent attacks. In a similar vein, security practitioners can apply tools from Machine Learning to build effective new defenses that learn from complex patterns of benign and malicious behavior. In turn, such adaptive defenses fall under the umbrella of learning in adversarial environments as motivated above.

This dissertation describes several research projects in the intersection of Machine Learning, Statistics, Security and Privacy, and explores questions relating to each of the topics in the intersection described above. As depicted in Figure 1.1, this work is made up of two parts which explore Secure Machine Learning and applications of Machine Learning to Security respectively.

Chapter Organization. The remainder of this section summarizes the main themes of the two parts of this dissertation. Section 1.2 summarizes general related work in Learning, Statistics and Security, and Section 1.3 concludes the chapter with an introduction to an important aspect of threat models for learning systems—the adversary's capabilities.

1.1.1 Secure Machine Learning

Understanding the Security (and Privacy) of Machine Learning methods, and designing learners for adversarial environments, are two endeavors of vital importance for the

	Chapter	Experimental	Theoretical	Attacks	Defenses
Part I	2	•		•	•
	3		•	•	
	4		•		•
Part II	5		•		•
	6	•		•	•

Table 1.1: A classification of this dissertation's chapters as being experimental or theoretical in nature, and by the inclusion of attacks and/or defenses.

applicability of Machine Learning in the real-world.

Examples of Machine Learning application domains in which attackers are incentivized to exploit learning abound. Content publishers desiring increased page views to drive up advertising income, will undertake black hat search engine optimization through participating in link farms (Gyöngyi and Garcia-Molina, 2005). Spammers will attempt to evade Gmail email filtering by obfuscating the true nature of their email spam messages by including good tokens in their mail (Lowd and Meek, 2005a). Insurance companies will attempt to learn details of hospitals patient visits and conditions through linking published 'anonymized' data or statistics on private hospital databases in order to form better estimates of risk when considering applicants for health insurance plans (Rindfleisch, 1997; Sweeney, 2002). These three examples help motivate the problems studied in each of the three chapters of Part I.

Part I of this dissertation considers both the security analysis of Machine Learning techniques (the so-called 'Security of Machine Learning') and wherever possible, learning techniques that exhibit desirable Security or Privacy properties. Each chapter within Part I considers a different class of attack on learning systems.

- **Chapter 2.** This chapter presents two case-studies on manipulating Machine Learning systems by poisoning training data. The first study is of the open-source SpamBayes project for filtering email spam, and the second is of network-wide anomaly detection based on Principal Component Analysis. Both case-studies quantify the performance of the learner in the presence of an intelligent attacker, and both studies evaluate counter-measures for reducing the effects of the constructed attacks.

- **Chapter 3.** Where Chapter 2 considers manipulation of the training data, this chapter considers manipulation of test data of a previously trained classifier with the goal of evading detection. In this chapter, the *evasion problem* is considered in an abstract setting where the attacker searches for a minimal-cost instance that will go undetected by the classifier, while submitting only a small (polynomial) number of queries to the classifier. Efficient attack algorithms are developed for classifiers that partition feature space into two sets, one of which is convex.

- **Chapter 4.** Where the previous two chapters study attacks that focus on manipulating the learner or the learner's predictions, this chapter considers settings where

an adversary may wish to extract information about a learner's *specific* training data given access to the learned model that aggregates *general* statistical information about the data. The problem of releasing a trained Support Vector Machine (SVM) classifier while preserving the privacy of the training data is considered. Mechanisms with privacy and (statistical) utility guarantees are proposed, along-side negative results that bound the achievable privacy of any mechanism that discloses an accurate approximation to the SVM.

The chapters of Part I vary in length—Chapter 2 being the largest as it discusses two major case-studies—and vary in being theoretical vs. experimental in nature–while Chapter 2 is mostly experimental the contributions of Chapters 3 and 4 are theoretical in nature. The former chapter considers both attacks and defenses on learning systems, while the later chapters consider only attacks and defenses respectively[1]. Table 1.1 summarizes these differences.

Finally the research reported in Chapters 2 and 3 was joint work with two other UC Berkeley EECS doctoral candidates: Marco Barreno and Blaine Nelson. In both of these chapters I provide a brief summary of my contributions to the projects, in relation to those of Barreno and Nelson. I was the sole/lead graduate student on the research reported in the remainder of this dissertation.

1.1.2 Machine Learning for Security

While Part I considers learning in the presence of adversaries, the chapters of Part II of this dissertation view Computer Security through a lens of Machine Learning. Two kinds of opportunities are apparent for Machine Learning and Statistics in Security research. First, statistical models of a software system can be used to form effective attacks on the system. Second, learning can be leveraged to model legitimate and/or malicious behavior so as to build defenses that adapt to intelligent adversaries and benign data drift. Both applications are represented by the chapters of Part II as follows, and described in Table 1.1.

- **Chapter 5.** This chapter applies Online Learning Theory, which follows a game-theoretic approach to learning, to the problem of risk management. Risk management is modeled as a repeated game in which the adversary may attack a system to gain some profit or Return on Investment (ROI). The defender's goal is to allocate her defensive budget to minimize the attacker's profit or ROI. A learning-based *reactive* approach to risk management (where budget is allocated based on past attacks) is proposed. Using a reduction to results from Online Learning Theory, the new reactive defender is compared against fixed *proactive* strategies (where budget is allocated based on estimating risks and playing a fixed allocation). A major strength of the theoretical comparisons is that they hold for all sequences of attacks—no strong assumptions are placed on the adversary by the analysis.

[1]The attacks (defenses) presented in Chapter 3 (Chapter 4 respectively) are accompanied by strong guarantees for the attacker (defender) respectively, obviating the need for counter-measures.

- **Chapter 6.** While Chapter 5 applies machine learning theory to construct defenses, this chapter describes how to apply practical learning algorithms to construct attacks on open-source software projects. In particular, as a concrete case-study we apply the Support Vector Machine (which is also studied under a different setting in Chapter 4) to finding vulnerabilities in Mozilla's open-source Firefox web browser. Defensive measures for mitigating the effects of the developed attacks are also proposed.

1.2 Related Work

We now overview past work that is of general relevance to the topic of this dissertation. Discussion of related work that is specific to a single chapter is deferred to the particular chapter.

1.2.1 Related Tools from Statistics and Learning

Two entire subfields of Machine Learning and Statistics address questions related to learning in the presence of an adversary who attempts to manipulate the learning process by poisoning the training data (the topic of Chapter 2): Robust Statistics and Online Learning Theory. We briefly overview these areas here and discuss how each is applied within this dissertation. Chapter 7 includes a discussion of the current tools' inadequacies for the unique challenges of secure learning. The problem of privacy-preserving learning (the topic of Chapter 4) is a third, burgeoning area of Machine Learning which has been previously studied in Databases (statistical databases, Adam and Worthmann 1989), TCS (differential privacy, Dwork 2008), and Statistics (data confidentiality and statistical disclosure control, Doyle et al. 2001; Willenborg and de Waal 2001). We defer discussion of privacy-preserving learning to Chapter 4.

Robust Statistics. The field of Statistics that values traditional properties of estimators such as consistency and asymptotic efficiency, together with robustness—estimators that are not overly influenced by outliers—is known as *Robust Statistics* (Hampel et al., 1980; Huber, 1981). In the presence of outliers, or more generally violations of modeling assumptions, a *robust estimator* should have low bias, high efficiency and be asymptotically unbiased.

A common measure of the robustness of a statistic is its *breakdown point*: the largest $p \in [0, 0.5]$ such that letting a fraction p of the sample tend to ∞ does not pull the statistic to ∞ as well. Classic examples of statistics with good and bad breakdown points are the estimators of location: the median and mean with maximum and minimum possible breakdown points of 0.5 and 0 respectively.

One approach to finding robust estimators is via *influence functions*, which measure the effect on the asymptotic bias of an estimator, of an infinitesimal contamination at a point. Robust estimators should have bounded influence functions: the estimator should not go to ∞ as the point diverges. In particular the influence functions of M-estimators (estimators derived by minimizing the sum of a score function over the sample) are proportional to

the derivative of the chosen score function, and so such estimators can be designed with robustness in mind.

Remark 1. *We note in passing, similarities between Robust Statistics and a seemingly unrelated topic touched on in this dissertation. Dwork and Lei (2009) demonstrated through several detailed examples that robust estimators can serve as the basis for privacy-preserving mechanisms, by exploiting the limited influence of outliers on robust estimators. Given that the typical route for transforming a statistic into a mechanism that preserves differential privacy is via the statistic's sensitivity to data perturbations (Dwork et al., 2006), such a connection should not be too surprising. Still, finding a general connection between robustness and privacy remains an open problem. We develop a privacy-preserving Support Vector Machine in Chapter 4 via algorithmic stability which is an area of learning theory that exploits smoothness of the learning map to yield risk bounds.*

In Chapter 2 we develop data poisoning attacks on Principal Components Analysis (PCA), a feature reduction method that selects features which capture the maximal amount of variance in a dataset. For a counter-measure to our attacks, we turn to robust versions of PCA that maximize alternative *robust* measures of scale: the median absolute deviation (with the optimal breakdown point of 0.5) in place of the variance (having the worst breakdown point of zero). Empirical evaluations of Robust PCA show good resilience to our attacks, cutting the increased False Negative Rates down by half.

Online Learning Theory. While a common assumption in Machine Learning and Statistics is i.i.d. data or some other independence assumption, *Online Learning Theory* considers learning without any assumptions on the data generation process; the data need not even be stochastic. Online Learning, closely related to universal prediction, thus follows a game-theoretic approach to learning (Cesa-Bianchi and Lugosi, 2006).

Given a sequence of arbitrary instances, the learner predicts labels for the instances as they are iteratively revealed. Upon each round the learner incurs some loss, so that over T rounds the learner accumulates a cumulative loss that can be arbitrarily bad as measured in absolute terms. Instead, Online Learning compares the learner's performance with that of the best performing decision rule or *expert* among a set of experts that provide advice to the learner throughout the repeated game. This results in studying the *regret*: the difference between the learner's cumulative loss and the best expert's cumulative loss with hindsight. The goal of Online Learning is to design learning strategies that achieve average regret converging to zero. A motivating example for regret is in online portfolio optimization (Stoltz and Lugosi, 2005), where a simple goal of an investor selecting between N stocks (the experts) is to achieve a portfolio (a random strategy over the experts) that asymptotically performs as well as the best stock on average.

Advantages of this style of analysis include guarantees that hold in a fully adversarial setting, and the derived algorithms tend to be simple to implement and very efficient to run.

In Chapter 5 we model the general security problem of risk management as a repeated game between an attacker who gains profits depending on their chosen attack, and a defender

who allocates her security budget in order to reduce the profit enjoyed by the attacker. Via a reduction to regret bounds from Online Learning Theory, we show that the performance of a learning-based *reactive* defender who allocates budget based on past attacks, achieves the performance of the best *proactive* defender who can exploit prior knowledge of the system's vulnerabilities to form minimax allocates. By appealing to regret bounds, the analysis of our learning-based defense has the advantage that it allows for a worst-case attacker, even one that has full knowledge of the learner's state and algorithm, from which it can form intelligent attacks.

1.2.2 Attacks on Learning Systems

Barreno et al. (2006, 2010) categorize attacks against machine learning systems along three dimensions. The *axes* of their taxonomy are as follows:

Influence

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

Security violation

- *Integrity* attacks compromise assets via false negatives.
- *Availability* attacks cause denial of service, usually via false positives.

Specificity

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

The first axis of the taxonomy describes the capability of the attacker: whether (a) the attacker has the ability to influence the training data that is used to learn a model (a *Causative* attack) or (b) the attacker does not influence the learner, but can submit test instances to the learned model, and observe the resulting responses (an *Exploratory* attack).

The second axis indicates the type of security violation caused on a classifier (where we consider malicious/benign instances as belonging to the positive/negative class): (a) false negatives, in which malicious instances slip through the filter (an *Integrity* violation); or (b) false positives, in which innocuous instances are incorrectly filtered (an *Availability* violation).[2]

The third axis refers to how specific the attacker's intention is: whether (a) the attack is *Targeted* to degrade the learner's performance on particular types of instances or (b) the attack aims to cause the learner to fail in an *Indiscriminate* fashion on a broad class of instances.

[2]Considerations of false positives or false negatives apply specifically to learning for classification, however these violations extend to other kinds of learning as well (*e.g.*, an Integrity attack on a regression may aim to avoid a certain real-valued response, while an Availability attack may aim to perturb the responses so much as to cause a DoS attack on the learner itself).

	Chapter	Influence	Security Violation	Specificity
Part I	2(i)	Causative	Availability	Targeted, Indiscriminate
	2(ii)	Causative	Integrity	Targeted
	3	Exploratory	Integrity	Targeted
	4	Causative, Exploratory	Confidentiality	Targeted
Part II	5	Causative, Exploratory	Integrity	Targeted
	6	N/A	Confidentiality	Targeted

Table 1.2: The contributions of each chapter classified by the taxonomy on attackers on learning systems of Barreno et al. (2006). Each classification is discussed within the corresponding chapter.

Table 1.2 classifies the chapters of this dissertation according to the above taxonomy. Chapter 5 develops defensive risk management strategies (the learner's task is more complex than classification) with the goal of minimizing attacker profit or ROI. The adversary's attacks can be regarded as attempting to evade allocations of defensive budget, and so can be regarded as Integrity attacks. Chapters 4 and 6 concern attacks that violate neither Integrity nor Availability, but rather Confidentiality (a third security violation introduced below). Moreover while Chapter 6 does not involve an attack on a learner *per se*, we consider the public release of patches to the Firefox open-source project to be highly informative statistics of an underlying dataset (the patches combined with their labels as either 'security' or 'non-security'). Based on these statistics, our attacks violate the Confidentiality of the data's undisclosed labels.

Related Work: Attacks on Learners. Several authors have previously considered attacks covering a range of attack types described by the taxonomy.

Most prior work on attacking learning systems consider Exploratory attacks, in which the adversary submits malicious test points to a pre-trained classifier. To the best of our

knowledge, of these Exploratory attacks all focus on Integrity violations that cause false negatives. Lowd and Meek (2005b) consider an abstract formulation of what we call the *evasion problem*, in which the attacker wishes to make minimal-cost alterations to a positive instance (*e.g.*, an email spam message) such that the modified instance is labeled negative by classifier (the modified message reaches the victim's inbox). They derive query-based algorithms for Boolean and linear classifiers; in the latter case their algorithms not only find instances that evade detection, but in-so-doing learn the classifier's decision boundary. In Chapter 3 we generalize their work to classifiers that partition feature space into two sets, one of which is convex. For this larger family of classifiers, learning the decision boundary is known to be NP-hard (Dyer and Frieze, 1992; Rademacher and Goyal, 2009). In earlier work Tan et al. (2002) and Wagner and Soto (2002) independently designed *mimicry attacks* for evading sequence-based intrusion detection systems (IDSs). By analyzing the IDS offline, they modify exploits to mimic benign behavior not detected by the IDS. Fogla and Lee (2006) design *polymorphic blending attacks* on IDSs that encrypt malicious traffic to become indistinguishable from innocuous traffic. By contrast our algorithms for evading convex-inducing classifiers searches by querying the classifier online. In the email spam domain Wittel and Wu (2004) and Lowd and Meek (2005a) consider *good word attacks* that add common words to spam messages to allow them to pass through an email spam filter; in an alternate approach Karlberger et al. (2007) replace tokens in spam messages that have strong spam scores with synonyms. In the realm of counter-measures designed specifically for Exploratory attacks, Dalvi et al. (2004) consider an optimal cost-based defense for naive Bayes for a rational, omniscient attacker. In Chapter 5 we apply Online Learning Theory to design a learning-based reactive risk management strategy, which faces an adversary whose attacks may try to side-step the defensive budget allocations to the system. Since our performance guarantees for the reactive strategy are derived under extremely weak conditions on the adversary, we can guarantee that over time the success of such Exploratory attacks is limited.

A relatively small number of prior studies have investigated Causative attacks, in which the adversary manipulates the learner by poisoning its training data (although the areas of Online Learning and Robust Statistics both address learning under such attacks, in specific settings). Newsome et al. (2006) study *red herring* Causative Integrity attacks on the Polygraph polymorphic work detector (Newsome et al., 2005), that aim to increase false negatives. Their attacks include spurious features in positive training examples (worms) so that subsequent malicious instances can evade being detected by the conjunction learner, which copes poorly with high levels of irrelevant features. The authors also consider *correlated outlier* Exploratory Availability attacks which mis-train the learner into blocking benign traffic. Chung and Mok (2006, 2007) send traffic to the Autograph worm detector (Kim and Karp, 2004) that is flagged as malicious. Subsequent legitimate-looking traffic sent from the same node result in rules that block similar traffic patterns, including truly legitimate traffic. Kearns and Li (1993) consider the Probably Approximately Correct (PAC) model of learning (Valiant, 1984), in the presence of an adversary that manipulates a portion of the training data. In this theoretical work the authors bound the classification error in terms of the level of malicious noise, and bound the maximum level of noise tolerable for learnabil-

ity. In Chapter 2 we consider Causative Availability and Causative Integrity attacks on the SpamBayes email spam filter, and the PCA-based network anomaly detector respectively. The former case-study is the first Causative Availability attack of its kind, while the second study explores a system of recent popularity in the Systems and Measurement communities. In both cases we consider effective counter-measures for our attacks, including one that uses Robust Statistics. Finally, relative to all fixed proactive strategies, our reactive risk management strategy presented in Chapter 5 can handle both Exploratory attacks and Causative attacks by virtue of the worst-case nature of our analysis.

The Third Security Violation: Confidentiality. The taxonomy of Barreno et al. (2006) describes the attacker's goals through the violation and specificity axes, broadly covering most situations where the attacker wishes to manipulate a learner or its predictions. However, as is apparent from an increasing body of research on the privacy of statistical estimators (Dwork, 2010) a third security violation that can be achieved by an adversary attacking an adaptive system may be one of *confidentiality*. The International Organization for Standardization (2005) defines confidentiality as "ensuring that information is accessible only to those authorized to have access", and indeed confidentiality is a corner-stone of information security along-side integrity and availability (Wikipedia, 2010). In particular confidentiality should be considered as a general kind of security violation, and should include attacks that reveal information about the learner's state (Barreno, 2008), parameters to the learner, or the learner's training data.

Numerous authors have studied Confidentiality attacks on statistical databases that release statistics, learned models, or even anonymized forms of the data itself. A common form of real-world attack on statistical databases is to exploit side information available via public data covering overlapping features and rows of the database. Sweeney (2002) used public voter registration records to identify individuals (including the Governor) in an 'anonymized' database of hospital records of Massachusetts state employees where patient names had been removed. In a similar case Narayanan and Shmatikov (2008) identified users in an 'anonymized' movie rating dataset released by Netflix for their collaborative filtering prize competition. Given a small subset of movies a customer had watched, acting as a unique kind of *signature* for the customer, their attack accurately identifies the customer's ratings in the dataset. They applied their attack using publicly available movie ratings on IMDB to identify Netflix customers and their previously-private tastes in movies.

The key problem with removing only *explicitly* personal information such as names from a released database, is that seemingly innocuous features can implicitly identify individuals when used together. Sweeney (2002) showed in her study that gender, postal code and birthdate is enough to uniquely identify 87% of the U.S. population. An early real-world Confidentiality violation using this same idea was reported by the New York Times, when journalists Barbaro and Zeller Jr. (2006) identified AOL members from very specific queries included in an ID-scrubbed search log released by AOL research.

A number of theoretical Confidentiality attacks have been designed in past work, demonstrating the fundamental privacy limits of statistical database mechanisms. Dinur and Nissim (2003) show that if noise of rate only $o(\sqrt{n})$ is added to subset sum queries on a database

of n bits then an adversary can reconstruct a $1 - o(1)$ fraction of the database. This is a threshold phenomenon that says if accuracy is too high, no level of privacy can be guaranteed. Dwork and Yekhanin (2008) construct realistic, acute attacks in which only a fixed number of queries is made for each bit revealed. In a similar vein we show negative results for the privacy-preserving Support Vector Machine (SVM) setting in Chapter 4, where any mechanism that is too accurate with respect to the SVM, cannot guarantee high levels of privacy.

Deriving defenses for Confidentiality attacks on learners is an active area of research in Databases, Machine Learning, Security, Statistics and TCS (Dwork, 2008, 2010). An increasingly popular guarantee of data privacy is provided by *differential privacy* (Dwork, 2006). We provide the definition and technical details of differential privacy in Chapter 4, however the intuition is that even a powerful attacker with full knowledge of all but one row in a databases, the workings of the statistical database mechanism, and access to responses from the mechanism, cannot reconstruct the final database row. Differentially private versions of several statistics and learning algorithms have thus far been developed, including: contingency tables (Barak et al., 2007), histograms, Principal Component Analysis, k means, ID3, the perceptron algorithm (Blum et al., 2005), regularized logistic regression (Chaudhuri and Monteleoni, 2009), query and click count logs (Korolova et al., 2009), degree distributions of graphs (Hay et al., 2009), and several recommender systems that were used in the Netflix prize contest (McSherry and Mironov, 2009). We derive privacy-preserving mechanisms for SVM learning in Chapter 4.

1.3 The Importance of the Adversary's Capabilities

As discussed with respect to the taxonomy above, a crucial step in protecting against threats on Machine Learning systems is to understand the threat model in adversarial learning domains. The threat model can broadly be described as the attacker's *goals* and *capabilities*. Typical attacker goals are well-represented by the taxonomy of Barreno et al. (2006) described above. However this taxonomy considers the adversary's capabilities at the coarsest level as either being able to manipulate the training and/or test data. A finer grained analysis of adversarial capabilities may consider the level of *information* and the level of *control* possessed by the adversary.

Definition 2. Adversarial information *is the adversary's knowledge of the learning system and environment, such as the learner's features, the learning algorithm, the current decision function, the policy for training and retraining, and the benign data generation process.*

Definition 3. Adversarial control *is the extent of the attacker's control over the learning system's training and/or test data.*

A number of examples illustrating the roles of adversarial information and control now follow.

Example 4. *In email spam filtering, relevant adversarial information may include the user's language, common types of email the user receives, which spam filter the user has, and the*

particular training corpus or distribution used to create the spam filter (or knowledge of a similar distribution). Adversarial control may include choosing the bodies of a fraction of emails (perhaps only spam), controlling email headers directly or indirectly, and controlling how the user receives messages. This control could be exerted over messages used for training or for run-time testing.

Example 5. *In network-wide traffic anomaly detection (Lakhina et al., 2004a), adversarial information may include the network topology, routing tables, real-time traffic volumes along one or more links, historical traffic along one or more links, and the training policies of the anomaly detection system. Adversarial control may include controlling one or more links to give false traffic reports or compromising one or more routers to inject chaff into the network.*

Example 6. *In the domain of phishing webpage detection, adversarial information may include user language and country, email client, web browser, financial institution, and employer. Adversarial control may include choosing the content and/or headers of the phishing emails and potentially influencing training datasets of known phishing sites, such as PhishTank (2010).*

In the sequel, assumptions on adversarial information and control will be made explicit. An interesting and important research direction is to consider analyses that quantify the value of the information and control available to the adversary for attacks against learning systems. We revisit such open question in Chapter 7.

Chapter 2 considers case studies in Causative Integrity and Availability attacks on classifiers with special attention paid to the effects of increasing adversarial information or control. Chapter 3 develops Exploratory Integrity attacks on classifiers where bounds are derived on the number of query test points required by the attacker: in-turn these correspond to the amount of control the adversary has over the test data. Finally the attacks on Firefox constructed in Chapter 6 either utilize meta-data about commits to an open-source repository or are oblivious to the commit details. Once again, this corresponds to the level of information available to the attacker. Chapters 4 and 5 grant significant amounts of information and control to the adversary, as they derive results in worst-case settings.

Part I

Private and Secure Machine Learning

Chapter 2

Poisoning Classifiers

Expect poison from standing water.
– William Blake

Statistical Machine Learning techniques have recently garnered increased popularity as a means to filter email spam (Ramachandran et al., 2007; Robinson, 2003) and improve network design and security (Bahl et al., 2007; Cheng et al., 2007; Kandula et al., 2008; Lazarevic et al., 2003), as learning techniques can adapt to specifics of an adversary's behavior. However using Statistical Machine Learning for making security decisions introduces new vulnerabilities in large-scale systems due to this very adaptability. This chapter develops attacks that exploit Statistical Machine Learning, as used in the SpamBayes email spam filter (Meyer and Whateley, 2004; Robinson, 2003) and the Principal Component Analysis (PCA)-subspace method for detecting anomalies in backbone networks (Lakhina et al., 2004a).

In the language of the taxonomy of Barreno et al. (2006) (*cf.* Section 1.2.2), the attacks of this chapter are case-studies in Causative attacks: the adversary influences the classifier by manipulating the learner's training data. The attacks on SpamBayes are Availability attacks in that they aim to increase the false positive rate or availability of the spam filter (constituting a DoS attack on the learning component itself). By contrast the presented attacks on PCA are Integrity attacks that aim to increase the false negative rate or chance of evasion. Finally the attacks on PCA are Targeted in that they facilitate specific false negatives. Both Indiscriminate and Targeted attacks on SpamBayes are presented, and special attention is paid to the adversary's capabilities, with attacks exploiting a range of adversarial information and control compared experimentally throughout.

As the presented attacks highlight and quantify the severity of *existing* vulnerabilities in the SpamBayes and PCA-based systems, it becomes necessary to design defensive approaches that are less susceptible to tampering. In both of the training data poisoning case-studies, counter-measures are proposed that reduce the effects of the presented attacks.

The SpamBayes case-study of Section 2.2 presents joint work with UCB EECS doctoral candidates Marco Barreno and Blaine Nelson. In that study I contributed equally to the design of the attacks on SpamBayes, while Barreno and Nelson were responsible for their

implementation and the experimental analysis. In Section 2.3's case-study on PCA-based anomaly detection I was the lead graduate student, in joint work with Nelson. While we equally contributed to the design of the attacks and the defenses, I was responsible for the implementation and experimental analysis.

2.1 Introduction

Applications use Statistical Machine Learning to perform a growing number of critical tasks in virtually all areas of computing. The key strength of Machine Learning is adaptability; however, this can become a weakness when an adversary manipulates the learner's environment. With the continual growth of malicious activity and electronic crime, the increasingly broad adoption of learning makes assessing the vulnerability of learning systems to manipulation an essential problem.

The question of robust decision making in systems that rely on Machine Learning is of interest in its own right. But for security practitioners, it is especially important, as a wide swath of security-sensitive applications build on Machine Learning technology, including intrusion detection systems, virus and worm detection systems, and spam filters (Bahl et al., 2007; Cheng et al., 2007; Kandula et al., 2008; Lakhina et al., 2004a,a; Lazarevic et al., 2003; Liao and Vemuri, 2002; Meyer and Whateley, 2004; Mukkamala et al., 2002; Newsome et al., 2005; Ramachandran et al., 2007; Robinson, 2003; Soule et al., 2005; Stolfo et al., 2006; Zhang et al., 2005). These solutions draw upon a variety of techniques from the SML domain including Singular Value Decomposition, clustering, Bayesian inference, spectral analysis, maximum-margin classification, etc.; and in many scenarios, these approaches have been demonstrated to perform well *in the absence of Causative attacks on the learner*.

Past Machine Learning research has often proceeded under the assumption that learning systems are provided with training data drawn from a natural distribution of inputs. Such techniques have a serious vulnerability, however, as in many real-world applications an attacker has the ability to provide the learning system with maliciously chosen inputs that cause the system to infer poor classification rules. In the spam domain, for example, the adversary can send carefully crafted spam messages that a human user will correctly identify and mark as spam, but which can influence the underlying learning system and adversely affect its ability to correctly classify future messages. A similar scenario is conceivable for the network anomaly detection domain, where an adversary could carefully inject traffic into the network so that the detector mis-learns its model of normal traffic patterns.

This chapter explores two in-depth case-studies into Causative attacks, and corresponding counter-measures, against Machine Learning systems. The first case-study considers the email spam filtering problem, where the attacker's goal is to increase False Positive Rates as a denial-of-service attack on the learner itself. The second case-study explores a problem in network anomaly detection where the adversary's goal for poisoning is to increase the False Negative Rate so that subsequent attacks through the network can go un-detected. Throughout the chapter, the key roles of the adversary's capabilities of information and control are highlighted, and their effect on the attacks' damage measured.

Chapter Organization. This section is completed with a survey of previous work related to poisoning learners. Section 2.2 describes in detail a Causative attack case-study on email spam filtering, while Section 2.3 details a case-study on network anomaly detection. Finally the chapter is concluded with a summary of its main contributions in Section 2.4.

2.1.1 Related Work

Many authors have examined adversarial learning from a theoretical perspective. For example, within the Probably Approximately Correct framework, Kearns and Li (1993) bound the classification error an adversary that has control over a fraction β of the training set can cause. Dalvi et al. (2004) apply game theory to the classification problem: they model interactions between the classifier and attacker as a game and find the optimal counter-strategy for the classifier against an optimal opponent. The learning theory community has focused on online learning (Cesa-Bianchi and Lugosi, 2006), where data is selected by an adversary with complete knowledge of the learner, and has developed efficient algorithms with strong guarantees. However, the simplifying assumption of all data being produced by an omniscient adversary does not hold for many practical threat models. Given the increasing popularity of SML techniques, we believe exploring adversarial learning with realistic threat models is important and timely.

A handful of studies have considered Causative attacks on SML-based systems. Newsome et al. (2006) present attacks against Polygraph (Newsome et al., 2005), a polymorphic virus detector that uses Machine Learning. They suggest a correlated outlier attack, which attacks a naive-Bayes-like learner by adding spurious features to positive training instances, causing the filter to block benign traffic with those features (a Causative Availability attack). Focusing on conjunction learners, they present Causative Integrity red herring attacks that again include spurious features in positive training examples so that subsequent malicious instances can evade detection by excluding these features. Our attacks use similar ideas, but we develop and test them on real systems in other domains and we also explore the value of information and control to an attacker, and we present and test defenses against the attacks. Venkataraman et al. (2008) present lower bounds for learning worm signatures based on red herring attacks and reductions to classic results from Query Learning. Chung and Mok (2006, 2007) present a Causative Availability attack against the earlier Autograph worm signature generation system (Kim and Karp, 2004), which infers blocking rules based on patterns observed in traffic from suspicious nodes. The main idea is that the attack node first sends traffic that causes Autograph to mark it suspicious, then sends traffic similar to legitimate traffic, resulting in rules that cause a denial of service.

Most existing attacks against content-based spam filters in particular are Exploratory attacks that do not influence training but engineer spam messages so they pass through the filter. For example, Lowd and Meek (2005a,b) explore reverse-engineering a spam classifier to find high-value messages that the filter does not block, Karlberger et al. (2007) study the effect of replacing strong spam words with synonyms, and Wittel and Wu (2004) study the effect of adding common words to spam to get it through a spam filter. Another Exploratory attack is the polymorphic blending attack of Fogla and Lee (2006), which encrypts malicious

traffic so that the traffic is indistinguishable from innocuous traffic to an intrusion detection system. By contrast our variance injection attacks add small amounts of high-variance chaff traffic to PCA's training data, to make the data appear more like future DoS attacks. We return to the problem of evading a classifier with carefully crafted test instances in Chapter 3.

Finally Ringberg et al. (2007) performed a study of the sensitivities of the PCA-based detection method studied in Section 2.3, that illustrates how the PCA method can be sensitive to the number of principal components used to describe the normal subspace. This parameter can limit PCA's effectiveness if not properly configured. They also show that routing outages can pollute the normal subspace; a kind of perturbation to the subspace that is not adversarial. Our work differs in two key ways. First we demonstrate a different type of sensitivity, namely that of data poisoning. This adversarial perturbation can be stealthy and subtle, and is more challenging to circumvent than observable routing outages. Second, Ringberg et al. (2007) focus on showing the variability in PCA's performance to certain sensitivities, and not on defenses. In our work, we propose a robust defense against a malicious adversary and demonstrate its effectiveness. It is conceivable that the technique we propose could help limit PCA's sensitivity to routing outages, although such a study is beyond the scope of this work. A recent study (Brauckhoff et al., 2009) showed that the sensitivities observed by Ringberg et al. (2007) come from PCA's inability to capture temporal correlations. They propose to replace PCA by a Karhunen-Loeve expansion. Our study indicates that it would be important to examine, in future work, the data poisoning robustness of this proposal.

2.2 Case-Study on Email Spam

This section demonstrates how attackers can exploit Machine Learning to subvert spam filters. Our attack strategies exhibit two key differences from previous work: traditional attacks modify spam emails to evade a spam filter, whereas our attacks interfere with the training process of the learning algorithm and *modify the filter itself*; and rather than focus only on placing spam emails in the victim's inbox, we subvert the spam filter to *remove legitimate emails* from the inbox (see the theses of Barreno 2008 and Saini 2008 for poisoning attacks that cause spam to evade filtering).

An attacker may have one of two goals: expose the victim to an advertisement or prevent the victim from seeing a legitimate message. Potential revenue gain for a spammer drives the first goal, while the second goal is motivated, for example, by an organization competing for a contract that wants to prevent competing bids from reaching their intended recipient.

Tying in with adversarial information (*cf.* Section 1.3), an attacker may have detailed knowledge of a specific email the victim is likely to receive in the future, or the attacker may know particular words or general information about the victim's word distribution. In many cases, the attacker may know nothing beyond which language the emails are likely to use.

When an attacker wants the victim to see spam emails, a broad *dictionary attack* can render the spam filter unusable, causing the victim to disable the filter (*cf.* Section 2.2.2.2).

With more information about the email distribution, the attacker can select a smaller dictionary of high-value features that are still effective. When an attacker wants to prevent a victim from seeing particular emails and has some information about those emails, the attacker can target them with a *focused attack* (*cf.* Section 2.2.2.3).

We demonstrate the potency of these attacks and then present two defenses. The *Reject On Negative Impact (RONI) defense* tests the impact of each email on training and doesn't train on messages that have a large negative impact. The *dynamic threshold defense* dynamically sets the spam filter's classification thresholds based on the data rather than using SpamBayes' static choice of thresholds. We show that both defenses are effective in preventing some attacks from succeeding.

We focus on the learning algorithm underlying several spam filters, including SpamBayes (spambayes.sourceforge.net), BogoFilter (bogofilter.sourceforge.net), and the machine learning component of SpamAssassin (spamassassin.apache.org).[1] We target the open-source SpamBayes system because it uses a pure machine learning method, it is familiar to the academic community (Meyer and Whateley, 2004), and it is popular, with over 1,800,000 downloads. Although we specifically attack SpamBayes, the widespread use of its statistical learning algorithm suggests that other filters may also be vulnerable to similar attacks.

Our experimental results confirm that this class of attacks presents a serious concern for statistical spam filters, when the adversary has only limited *control* over the learner (again, tying back to the importance of adversarial capabilities *cf.* Section 1.3). A dictionary attack can make a spam filter unusable when controlling just 1% of the messages in the training set, and a well-informed focused attack can remove the target email from the victim's inbox 90% of the time. Of our two defenses, one significantly mitigates the effect of the dictionary attack and the other provides insight into the strengths and limitations of threshold-based defenses.

2.2.1 Background on Email Spam Filtering

We now briefly overview common learning models for email spam filtering, and detail the SpamBayes learning algorithm.

2.2.1.1 Training model

SpamBayes produces a classifier from labeled examples to predict the true class of future emails. The labels used by SpamBayes consist of spam (bad, unsolicited email), ham (good, legitimate email), and unsure (SpamBayes isn't confident one way or the other). The classifier learns from a labeled *training set* or *corpus* of ham and spam emails.

Email clients (applications for viewing and manipulating email messages) use these labels in different ways—some clients filter email labeled as spam and unsure into "Spam-High" and "Spam-Low" folders, respectively, while other clients only filter email labeled as spam into a separate folder. Since the typical user reads most or all email in their inbox and

[1] The primary difference between the learning elements of these three filters is in their tokenization methods.

rarely (if ever) looks at the spam/spam-high folder, the unsure labels can be problematic. If unsure messages are filtered into a separate folder, users may periodically read the messages in that folder to avoid missing important email. If instead unsure messages are not filtered, then the user is confronted with those messages when checking the email in their inbox. Too much unsure email is almost as troublesome as too many false positives (ham labeled as spam) or false negatives (spam labeled as ham). In the extreme, if everything is labeled unsure then the user obtains no time savings at all from the filter.

In our scenarios, an organization uses SpamBayes to filter multiple users' incoming email[2] and trains on everyone's received email. SpamBayes may also be used as a personal email filter, in which case the presented attacks and defenses are likely to be equally effective.

To keep up with changing trends in the statistical characteristics of both legitimate and spam email, we assume that the organization retrains SpamBayes periodically (*e.g.*, weekly). Our attacks are not limited to any particular retraining process; they only require that the attacker can introduce attack data into the training set somehow (the contamination assumption). In the next section, we justify this assumption but the purpose of this investigation is only to analyze the effect of poisoned datasets.

2.2.1.2 SpamBayes Learning Method

SpamBayes makes classifications using token scores based on a simple model of spam status proposed by Meyer and Whateley (2004); Robinson (2003), based on ideas by Graham (2002) together with Fisher's method for combining independent significance tests (Fisher, 1948).

SpamBayes tokenizes the header and body of each email before constructing token spam scores. Robinson's method assumes that the presence or absence of tokens in an email affect its spam status independently. For each token w, the raw token spam score

$$\mathbf{P}_{(S,w)} \quad = \quad \frac{N_H N_S(w)}{N_H N_S(w) + N_S N_H(w)}$$

is computed from the counts N_S, N_H, $N_S(w)$, and $N_H(w)$—the number of spam emails, ham emails, spam emails that include w and ham emails that include w.

Robinson smooths $\mathbf{P}_{(S,w)}$ through a convex combination with a prior belief x, weighting the quantities by $N(w)$ (the number of training emails with w) and s (chosen for strength of prior), respectively:

$$f(w) \quad = \quad \frac{s}{s + N(w)}\, x + \frac{N(w)}{s + N(w)}\, \mathbf{P}_{(S,w)} \quad . \tag{2.1}$$

For a new email message E, Robinson uses Fisher's method to combine the spam scores of the most significant tokens[3] into a message score

[2]We use the terms *user* and *victim* interchangeably for either organization or individual; the meaning will be clear from context.

[3]SpamBayes uses at most 150 tokens from E with scores furthest from 0.5 and outside the interval $[0.4, 0.6]$. We call this set $\delta(E)$.

$$
\begin{aligned}
I(E) &= \frac{1 + H(E) - S(E)}{2} \in [0,1] \ , \qquad (2.2)\\
\text{where} \quad H(E) &= 1 - \chi^2_{2n}\left(-2 \sum_{w \in \delta(E)} \log f(w)\right) \ ,\\
S(E) &= 1 - \chi^2_{2n}\left(-2 \sum_{w \in \delta(E)} \log\left(1 - f(w)\right)\right) \ ,
\end{aligned}
$$

and where $\chi^2_{2n}(\cdot)$ denotes the cumulative distribution function of the chi-square distribution with $2n$ degrees of freedom. SpamBayes predicts by thresholding against two user-tunable thresholds θ_0 and θ_1, with defaults $\theta_0 = 0.15$ and $\theta_1 = 0.9$: SpamBayes predicts ham, unsure, or spam if I falls into the interval $[0, \theta_0]$, $(\theta_0, \theta_1]$, or $(\theta_1, 1]$, respectively.

The inclusion of an unsure category prevents us from purely using misclassification rates (false positives and false negatives) for evaluation. We must also consider *spam-as-unsure* and *ham-as-unsure* emails. Because of the considerations in Section 2.2.1.1, unsure misclassifications of ham emails are nearly as bad for the user as false positives.

2.2.2 Attacks

We now present Causative Availability attacks on SpamBayes, *i.e.*, attacks that aim to increase the False Positive Rate of the learned classifier by manipulating the training data. After describing the *contamination assumption* that we can realistically inject spam messages into the training corpus (Section 2.2.2.1), we detail both Indiscriminate (Section 2.2.2.2) and Targeted (Section 2.2.2.3) attacks.

2.2.2.1 The Contamination Assumption

We assume that the attacker can send emails that the victim will use for training—the *contamination assumption*—but incorporate two significant restrictions: attackers may specify arbitrary email bodies but not headers, and attack emails are always trained as spam and not ham. We examine the implications of the contamination assumption in the remainder of this section.

How can an attacker contaminate the training set? Consider the following alternatives. If the victim periodically retrains on all email, any email the attacker sends will be used for training. If the victim manually labels a training set, the attack emails will be included as spam because they genuinely are spam. Even if the victim retrains only on mistakes made by the filter, the attacker may be able to design emails that both perform our attacks and are also misclassified by the victim's current filter. We do not address the possibility that a user might inspect training data to remove attack emails; our attacks could be adjusted to evade detection strategies such as email size or word distributions, but we avoid pursuing this arms race here.

Our focus on spam-labeled attack emails should be viewed as a restriction and not a necessary condition for the success of the attacks—using ham-labeled attack emails could enable more powerful attacks that place spam in a user's inbox (Barreno, 2008; Saini, 2008).

2.2.2.2 Dictionary Attacks

Our first attack is an *Indiscriminate* attack. The idea behind the attack is to send *attack emails* that contain many words likely to occur in legitimate email. When the victim trains SpamBayes with these attack emails marked as spam, the spam scores of the words in the attack emails will increase. Future legitimate email is more likely to be marked as spam if it contains words from the attack email. With a sufficient increase to the False Positive Rate, the victim will disable the spam filter, or at least must frequently search through spam/unsure folders to find legitimate messages that were filtered away. In either case, the victim loses confidence in the filter and is forced to view more spam—the victim sees the attacker's spam.

Depending on the level of information available to the adversary, s/he may be able to construct more effective attacks on the spam filter.

Knowledge of Victim's Language. When the attacker lacks knowledge about the victim's email, one simple attack is to include an entire dictionary of the English language (or more generally a dictionary of the victim's native tongue). This technique is the basic *dictionary attack*. We use the GNU `aspell` English dictionary version 6.0-0, containing 98,568 words.

The dictionary attack increases the score of every token in a dictionary of English words *i.e.*, it makes them more indicative of spam). After it receives a dictionary spam message, the victim's spam filter will have a higher spam score for every token in the dictionary, an effect that is amplified for less frequent tokens: in particular, the spam scores of vulnerable tokens dramatically increases. Furthermore, the long-tailed Zipf distribution of natural human language implies that a victim's future non-spam email will likely contain several vulnerable tokens, increasing the filter's spam score for that email.

(Limited) Knowledge of Victim's Word Distribution. A further refinement uses a word source with distribution closer to the victim's email distribution. For example, a large pool of Usenet newsgroup postings may have colloquialisms, misspellings, and other "words" not found in a dictionary; furthermore, using the most frequent words in such a corpus may allow the attacker to send smaller emails without losing much effectiveness.

This attack exploits the sparsity of tokens in human text (*i.e.*, most people use small vocabularies). As mentioned above, in natural language there are a small number of words that are used frequently and a large number of words (a long tail) that are used infrequently.

2.2.2.3 Focused Attack

Our second kind of attack—the *focused attack*—assumes knowledge of a specific legitimate email or type of email the attacker wants blocked by the victim's spam filter. This is

a *Causative Targeted Availability* attack. In the focused attack, the attacker sends attack emails to the victim containing words likely to occur in the target email. When SpamBayes trains on this attack email, the spam scores of the targeted tokens increase, so the target message is more likely to be filtered as spam.

For example, consider a malicious contractor wishing to prevent the victim from receiving messages with competing bids. The attacker sends spam emails to the victim with words such as the names of competing companies, their products, and their employees. The bid messages may even follow a common template known to the attacker, making the attack easier to craft.

The attacker may have different levels of knowledge about the target email. In the extreme case, the attacker might know the exact content of the target email and use all of its words. More realistically, the attacker only guesses a fraction of the email's content. In either case, the attack email may include additional words as well, *e.g.*, drawn from a general distribution over email text to obfuscate the message's intent.

Like the Usenet distribution-based attack, the focused attack is more concise than the dictionary attack because the attacker has detailed knowledge of the target and need not affect other messages. The focused attack can be more concise because it leaves out words that are unlikely to appear. This conciseness makes the attack both more efficient for the attacker and more difficult to detect as an attack.

2.2.2.4 A Principled Justification of the Dictionary and Focused Attacks

The dictionary and focused attacks can be seen as two instances of a common attack in which the attacker has different levels of information about the victim's email. Without loss of generality, suppose the attacker generates only a single attack message $\mathbf{a}$. The victim adds it to the training set as spam, trains, and classifies a (random) new text message $\mathbf{M}$ in the future. Since SpamBayes operates under a (typical) bag-of-words model, both $\mathbf{a}$ and $\mathbf{M}$ are indicator vectors, where the i^{th} component is true iff word i appears in the email. The attacker also has some (perhaps limited) knowledge of the next email the victim will receive. This knowledge can be represented as a distribution D—the vector of probabilities that each word appears in the next message. That is, the attacker assumes that $\mathbf{M} \sim D$, and has reason to believe that D is related to the true underlying email distribution of the victim.

The goal of the attacker is to choose an attack email $\mathbf{a}$ that maximizes the *expected spam score* $I_{\mathbf{a}}$ (Equation 2.2 with the attack message $\mathbf{a}$ including in the training corpus) of the next legitimate email $\mathbf{M}$ drawn from distribution D; that is, the attacker's goal is to select an attack message in

$$\arg\max_{\mathbf{a}} \quad \mathbb{E}_{\mathbf{M}\sim D}\left[I_{\mathbf{a}}(\mathbf{M})\right] \ .$$

In order to describe the optimal attacks under this criterion, we make two observations. First, the spam scores of distinct words do not interact; that is, adding a word w to the attack does not change the score $f(u)$ of some different word $u \neq w$. Second, it is easy

to show that I is non-decreasing in each $f(w)$. Therefore the best way to increase $I_{\mathbf{a}}$ is to include additional words in the attack message.

Now let us consider specific choices for the next email's distribution D. First, if the attacker has little knowledge about the words in target emails, the attacker can set D to be uniform over all emails. We can optimize the resulting expected spam score by including *all possible words* in the attack email. This *optimal attack* is infeasible in practice (as it includes misspellings, etc.) but can be approximated: one approximation includes all words in the victim's primary language, such as an English dictionary. This yields the dictionary attack.

Second, if the attacker has specific knowledge of a target email, we can represent this by setting $D(i)$ to 1 iff the i^{th} word is in the target email. The above 'optimal attack' still maximizes the expected spam score, but a more compact attack that also optimizes the expected spam score is to include all of the words in the target email. This produces the focused attack.

The attacker's knowledge usually falls between these extremes. For example, the attacker may use information about the distribution of words in English text to make the attack more efficient, such as characteristic vocabulary or jargon typical for the victim. Either way, the adversary's information results in a distribution D over words in the victim's emails.

2.2.3 Attack Results

We now present experiments launching the attacks described above on the SpamBayes email spam filter.

2.2.3.1 Experimental Method

Dataset. In our experiments we use the Text Retrieval Conference (TREC) 2005 spam corpus (Cormack and Lynam, 2005), which is based on the Enron email corpus (Klimt and Yang, 2004) and contains 92,189 emails (52,790 spam and 39,399 ham). This corpus has several strengths: it comes from a real-world source, it has a large number of emails, and its creators took care that the added spam does not have obvious artifacts to differentiate it. We also use a corpus constructed from a subset of Usenet English postings to generate words for our attacks (Shaoul and Westbury, 2007).

Training Method. For each experiment, we sample a dataset of email without replacement from the TREC corpus. We create a control model by training SpamBayes once only on the training set. Each of our attacks creates a different type of attack email for SpamBayes to use in training, producing tainted models.

When we require mailboxes of a specified size, such as for training and test sets, we sample ham and spam emails randomly without replacement from the entire TREC spam corpus. When we require only a portion of an email, such as the header, we randomly select an email from the dataset that has not been used in the current run, so that we ignore email messages that have already been selected for use in the training set.

Parameter	Dictionary Attack	Focused Attack	RONI Defense	Threshold Defense
Training set size	2,000, 10,000	5,000	20	2,000, 10,000
Test set size	200, 1,000	N/A	50	200, 1,000
Spam prevalence	0.50, 0.75	0.50	0.50	0.50
Attack fraction	0.001, 0.005, 0.01, 0.02, 0.05, 0.10	0.02 to 0.50 incrementing by 0.02	0.05	0.001, 0.01, 0.05, 0.10
Folds of validation	10	5 repetitions	5 repetitions	5
Target Emails	N/A	20	N/A	N/A

Table 2.1: Parameters used in our experiments.

Message Generation. We restrict the attacker to have limited control over the headers of attack emails (see Section 2.2.2.1). We implement this assumption either by using the entire header from a randomly selected spam email from TREC (focused attack) or by using an empty header (all other attacks).

Method of Assessment. We measure the effect of each attack by comparing classification performance of the control and compromised filters using K-fold cross-validation (or K repetitions with new random dataset samples in the case of the focused attack). In cross-validation, we partition the dataset into K subsets and perform K train-test epochs. During the i^{th} epoch, the i^{th} subset is set aside as a test set and the remaining $(K-1)$ subsets are used for training. Each email from our original dataset serves independently as both training and test data.

In the following sections, we show the effect of our attacks on test sets of held-out messages. Because our attacks are designed to cause ham to be misclassified, we only show their effect on ham messages; their effect on spam is marginal. Our graphs do not include error bars since we observed that the variation in our tests was small. See Table 2.1 for our experimental parameters.

2.2.3.2 Dictionary Attack Results

We examined the effect of adversarial control on the effectiveness of dictionary attacks. Here adversarial control is parametrized as the percent of attack messages in the training set. Figure 2.1 shows the misclassification rates of three dictionary attack variants averaging over ten-fold cross-validation. The optimal attack quickly causes the filter to label all ham emails as spam. The Usenet dictionary attack (90,000 top ranked words from the Usenet corpus) causes significantly more ham emails to be misclassified than the Aspell dictionary attack, since it contains common misspellings and slang terms that are not present in the Aspell dictionary (the overlap between the Aspell and Usenet dictionaries is around 61,000 words).

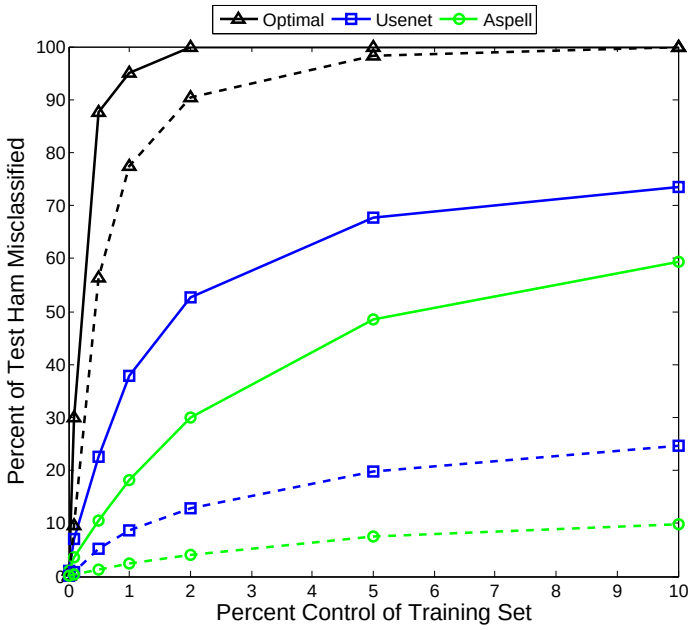


Figure 2.1: Three dictionary attacks on initial training set of 10,000 messages (50% spam). We plot percent of ham classified as *spam* (dashed lines) and as *spam* or *unsure* (solid lines) against the attack as percent of the training set. We show the optimal attack (black $\bigtriangleup$), the Usenet dictionary attack (blue $\square$), and the Aspell dictionary attack (green $\bigcirc$). Each attack renders the filter unusable with as little as 1% control (101 messages).

These variations of the attack require relatively few attack emails to significantly degrade SpamBayes accuracy. *By 101 attack emails (1% of 10,000), the accuracy falls significantly for each attack variation; at this point most users will gain no advantage from continued use of the filter.*

Remark 7. *Although the attack emails make up a small percentage of the* number of messages *in a poisoned inbox, they make up a large percentage of the* number of tokens. *For example, at 204 attack emails (2% of the messages), the Usenet attack includes approximately 6.4 times as many tokens as the original dataset and the Aspell attack includes 7 times as many tokens. An attack with fewer tokens would likely be harder to detect; however, the number of messages is a more visible feature. It is of significant interest that such a small number of attack messages is sufficient to degrade the performance of a widely-deployed filtering algorithm to such a degree.*

2.2.3.3 Focused Attack Results

In this section, we experimentally analyze how many attack emails are required for the focused attack to be effective, how accurate the attacker needs to be at guessing the target email, and whether some emails are easier to target than others.

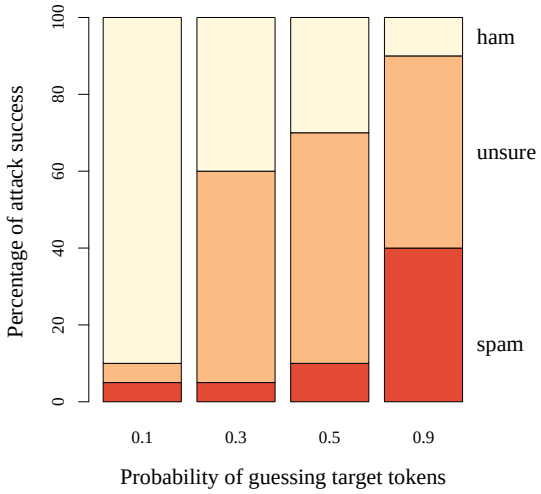


Figure 2.2: Effect of the focused attack as a function of adversarial information. Each bar depicts the fraction of target emails classified as spam, ham, and unsure after the attack. The initial inbox contains 5,000 emails (with 50% spam).

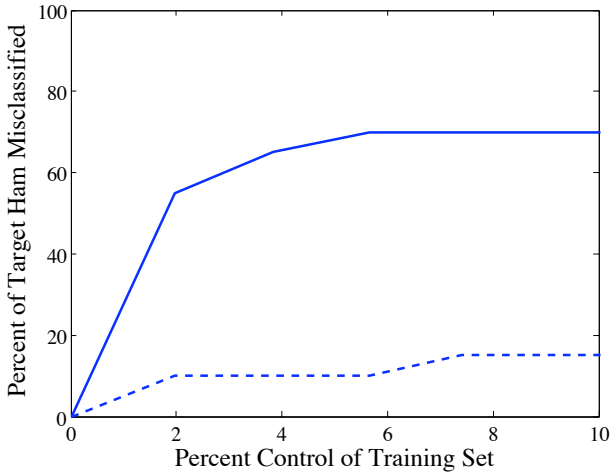


Figure 2.3: Effect of the focused attack as a function of adversarial control (with adversarial information at p=0.5) The dashed (solid) line shows the percentage of targets misclassified as spam (unsure or spam) after the attack. The initial inbox contains 5,000 emails (50% spam).

We run each repetition of the focused attack as follows. First we randomly select a ham email from the TREC corpus to serve as the target of the attack. We use a clean, non-malicious 5,000-message inbox with 50% spam. We repeat the entire attack procedure independently for 20 randomly-selected target emails.

In Figure 2.2, we examine the effectiveness of the attack when the attacker has increasing knowledge of the target email by simulating the process of the attacker guessing tokens from the target email. For this figure, there are 300 attack emails—16% of the original number of training emails. We assume that the attacker correctly guesses each word in the target with probability p in $\{0.1, 0.3, 0.5, 0.9\}$—the x-axis of Figure 2.2. The y-axis shows the proportion of the 20 targets classified as ham, unsure and spam. As expected, the attack is increasingly effective as the level of adversarial information p increases. *With knowledge of only 30% of the tokens in the target, 60% of the target emails are mis-classified.*

In Figure 2.3, we examine the attack's effect on misclassifications of the targeted emails as the number of attack messages—the adversarial control—increases. Here we fix the probability of guessing each target token at 0.5. The x-axis depicts the number of messages in the attack and the y-axis is the percent of messages misclassified. *With 100 attack emails, out of a initial mailbox size of 5,000, the target email is misclassified 32% of the time.*

Additional insight can be gained by examining the attack's effect on three representative emails (see Figure 2.4). Each of the panels in the figure represents a single target email representing each of three possible attack outcomes: ham misclassified as spam (Left), ham

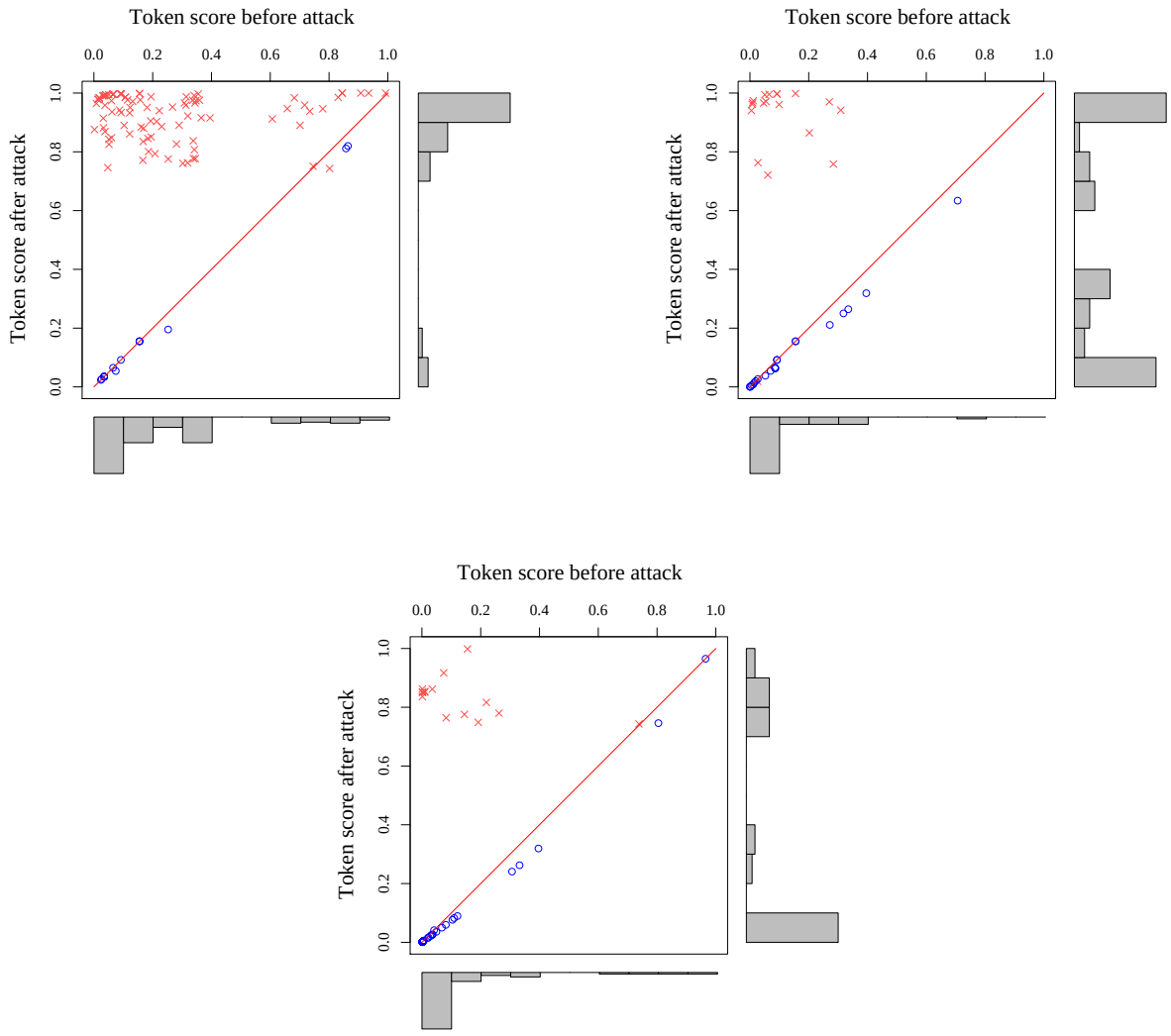


Figure 2.4: Effect of the focused attack on three representative emails—one graph for each target email. Each point is a token in the target email. The x-axis is the token spam score in Equation (2.1) before the attack (0 means ham and 1 means spam). The y-axis is the spam score after the attack. The red crosses are tokens that were included in the attack and the black circles are tokens that were not in the attack. The histograms show the distribution of spam scores before the attack (at bottom) and after the attack (at right).

misclassified as unsure (Middle), and ham correctly classified as ham (Right). Each point in the graph represents the before/after score of a token; any point above the line $y = x$ corresponds to a token score increase due to the attack and any point below the line corresponds to a decrease. From these graphs it is clear that tokens included in the attack typically increase significantly while those not included decrease slightly. Since the increase in score is more significant for included tokens than the decrease in score for excluded tokens,

the attack has substantial impact even when the attacker has a low probability of guessing tokens as seen in Figure 2.2. Furthermore, the before/after histograms in Figure 2.4 provide a direct indicator of the attack's success.

Also, comparing the bottom histograms of the three panels, we can see that the attack was most successful on emails that already contained a significant number of spam tokens before the attack. All three emails as a whole were confidently classified as ham before the attack, but in the successful attack, the target was closest to being classified as spam.

2.2.4 Defenses

In the following two sections we consider counter-measures for responding against the attacks presented above.

2.2.4.1 RONI defense

Our *Causative* attacks are effective since training on attack emails causes the filter to learn incorrect token spam scores and misclassify emails. Each attack email contributes towards the degradation of the filter's performance; if we can measure each email's impact prior to training, then we can remove deleterious messages from the training set before the classifier is manipulated.

In the *Reject On Negative Impact (RONI) defense*, we measure the incremental effect of each *query email* Q by testing the performance difference with and without that email. We independently sample a 20-message training set T and a 50-message validation set V five times from the initial pool of emails given to SpamBayes for training. We train on both T and $T \cup \{Q\}$ and measure the impact of each query email as the average change in incorrect classifications on V over the five trials. We reject candidate message Q from training if the impact is significantly negative. We test with 120 random non-attack spam messages and 15 repetitions each of seven variants of the dictionary attacks in Section 2.2.2.2.

Experiments show that the RONI defense is extremely successful against dictionary attacks, identifying 100% of the attack emails without flagging any non-attack emails. Each dictionary attack message causes *at least* an average decrease of 6.8 ham-as-ham messages. In sharp contrast, non-attack spam messages cause *at most* an average decrease of 4.4 ham-as-ham messages. This clear region of separability means a simple threshold on this statistic is effective at separating dictionary attack emails from non-attack spam.

This experiment provides some confidence that this defense would work given a larger test set due to the large impact a small number of attack emails have on performance.

However, the RONI defense fails to differentiate focused attack emails from non-attack emails. The explanation is simple: the dictionary attack broadly affects emails, including training emails, while the focused attack is targeted at a *future* email, so its effects may not be evident on the training set alone.

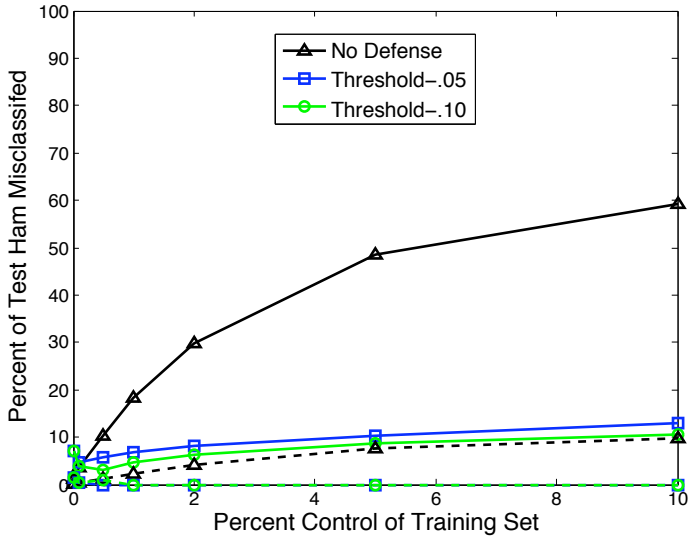


Figure 2.5: Effect of the threshold defense on the classification of ham messages with the dictionary based attacks. We use a 10,000 inbox training set of which 50% are spam. The solid lines represent ham messages classified as spam or unsure while the dashed lines show the classification rate of ham messages as spam. Threshold-.05 has a wider range for unsure messages than the Threshold-.10 variation.

2.2.4.2 Dynamic Threshold Defense

Distribution-based attacks increase the spam score of ham email but they also tend to increase the spam score of spam. Thus with new θ_0, θ_1 thresholds, it may still be possible to accurately distinguish between these kinds of messages after an attack. Based on this hypothesis, we propose and test a *dynamic threshold defense*, which dynamically adjusts θ_0, θ_1. With an adaptive threshold scheme, attacks that shift all scores will not be effective since rankings are invariant to such shifts.

To determine dynamic values of θ_0 and θ_1, we split the full training set in half. We use one half to train a SpamBayes filter F and the other half as a validation set V. Using F, we obtain a score for each email in V. From this information, we can pick threshold values that more accurately separate ham and spam emails. We define a utility function for choosing threshold t, $g(t) = N_{S,<}(t) \left(N_{S,<}(t) + N_{H,>}(t)\right)^{-1}$, where $N_{S,<}(t)$ is the number of spam emails with scores less than t and $N_{H,>}(t)$ is the number of ham emails with scores greater than t. We select θ_0 so that $g(\theta_0)$ is 0.05 or 0.10, and we select θ_1 so that $g(\theta_1)$ is 0.95 or 0.90, respectively.

This is a promising defense against dictionary attacks. As shown in Figure 2.5, the misclassification of ham emails is significantly reduced by using the defense. At all stages of the attack, ham emails are never classified as spam and only a moderate amount of them are labeled as unsure. However, while ham messages are often classified properly, the dynamic threshold causes almost all spam messages to be classified as unsure even when the attack is only 1% of the inbox. This shows that the dynamic threshold defense fails to adequately separate ham and spam given the number of spam also classified as unsure.

2.3 Case-Study on Network Anomaly Detection

In this section we study both poisoning strategies and defenses in the context of the PCA-subspace method for network-wide anomaly detection (Lakhina et al., 2004a), based on Principal Component Analysis (PCA). This technique has received a large amount of attention, leading to extensions (Lakhina et al., 2004b, 2005a,b), and inspiring related research (Brauckhoff et al., 2009; Huang et al., 2007; Li et al., 2006a; Ringberg et al., 2007; Zhang et al., 2005). Additionally, a few companies are exploring the use of PCA-based techniques and related SVD algorithms in their products (Guavus, 2010; Narus, 2010).

We consider an adversary who knows that an ISP is using a PCA-based anomaly detector. The adversary's aim is to evade future detection by poisoning the training data so that the detector learns a distorted set of principal components. Because PCA solely focuses on link traffic covariance, we explore poisoning schemes that add *chaff* (additional traffic) into the network to increase the variance of the network's traffic. The end goal of the attacker is to increase the false negative rate of the detector, which corresponds to the attacker's evasion success rate. That is, we consider Causative Integrity attacks on PCA-based network-wide anomaly detection.

The first contribution of this section is a detailed analysis of how adversaries subvert the learning process for the purposes of subsequent evasion. We explore a range of poisoning strategies in which the attacker's knowledge about the network traffic state is varied, and in which the attacker's time horizon (length of poisoning episode) is varied.

Because the network data on which SML techniques are applied are non-stationary, the baseline models must be periodically retrained to capture evolving trends in the underlying data. In previous usage scenarios (Lakhina et al., 2004a; Soule et al., 2005), the PCA detector is retrained regularly, for example weekly. A consequence is that attackers could poison PCA slowly over long periods of time—poisoning PCA in a more stealthy fashion. By perturbing the principal components gradually, the attacker decreases the chance that the poisoning activity itself is detected. We design such a poisoning scheme, called a *Boiling Frog* scheme, and demonstrate that it can boost the false negative rate as high as the non-stealthy strategies, with far less chaff, albeit over a longer period of time.

Our second main contribution is to design a robust defense against this type of poisoning. It is known that PCA can be strongly affected by outliers (Ringberg et al., 2007). However, instead of selecting principal components as directions that maximize variance, robust statistics suggests components that maximize more robust measures of dispersion. It is well known that the median is a more robust measure of location than the mean, in that it is far less sensitive to the influence of outliers. This concept can be extended to robust alternatives to variance such as the Median Absolute Deviation (MAD). Over the past two decades a number of robust PCA algorithms have been developed that maximize MAD instead of variance. Recently the PCA-GRID algorithm was proposed as an efficient method for maximizing MAD without under-estimating variance (a flaw identified in previous solutions) (Croux et al., 2007). We adapt PCA-GRID for anomaly detection by combining the method with a new robust cutoff threshold. Instead of modeling the squared prediction error as Gaussian (as in the original PCA-based detection method), we model the error using

a Laplace distribution. This new threshold is motivated from observing that the residuals have longer tails than modeled by a Gaussian. We call our method that combines PCA-GRID with a Laplace cutoff threshold, ANTIDOTE. The key intuition behind this method is to reduce the effect of outliers and help reject poisonous training data.

Our third contribution is to carry out extensive evaluations of both ANTIDOTE and the original PCA method, in a variety of poisoning situations, and to assess their performance via multiple metrics. To do this, we used traffic matrix data from the Abilene network since many other studies of traffic matrix estimation and anomaly detection have used this data. We show that the original PCA method can be easily compromised by any of our poisoning schemes, with only small amounts of chaff. For moderate amounts of chaff, the PCA detector starts to approach the performance of a random detector. However ANTIDOTE is dramatically more robust. It outperforms PCA in that i) it more effectively limits the adversary's ability to increase his evasion success; ii) it can reject a larger portion of contaminated training data; and iii) it provides robust protection across nearly all origin-destination flows through a network. The gains of ANTIDOTE for these performance measures are large, especially as the amount of poisoning increases. Most importantly, we demonstrate that ANTIDOTE incurs insignificant shifts in its false negative and false positive performance, compared to PCA, in the absence of poisoning.

Our study sheds light on the general problem of poisoning SML techniques, in terms of the types of poisoning schemes that can be construed, their impact on detection, and strategies for defense.

2.3.1 Background

To uncover anomalies, many network detection techniques mine the network-wide traffic matrix, which describes the traffic volume between all pairs of Points-of-Presence (PoP) in a backbone network and contains the collected traffic volume time series for each origin-destination (OD) flow. In this section, we define traffic matrices, present our notation, and summarize the PCA anomaly detection method of Lakhina et al. (2004a).

2.3.1.1 Traffic Matrices and Volume Anomalies

Network link traffic represents the superposition of OD flows. We consider a network with N links and F OD flows and measure traffic on this network over T time intervals. The relationship between link traffic and OD flow traffic is concisely captured in the *routing matrix* $\mathbf{A}$. This matrix is an $N \times F$ matrix such that $\mathbf{A}_{ij} = 1$ if OD flow j passes over link i, and is zero otherwise. If $\mathbf{X}$ is the $T \times F$ *traffic matrix* (TM) containing the time-series of all OD flows, and if $\mathbf{Y}$ is the $T \times N$ *link TM* containing the time-series of traffic on all links, then $\mathbf{Y} = \mathbf{X}\mathbf{A}^\top$. We denote the t^{th} row of $\mathbf{Y}$ as $\mathbf{y}(t) = \mathbf{Y}_{t,\bullet}$ (the vector of N link traffic measurements at time t), and the original traffic along a *source link*, S by $\mathbf{y}_S(t)$. We denote column f of routing matrix $\mathbf{A}$ by $\mathbf{A}_f$.

We consider the problem of detecting *OD flow volume anomalies* across a top-tier network by observing link traffic volumes. Anomalous flow volumes are unusual traffic load levels in a network caused by anomalies such as Denial of Service (DoS) attacks, Distributed

DoS attacks, flash crowds, device failures, misconfigurations, and so on. DoS attacks serve as the canonical example attack in this study.

Traditionally, network-wide anomaly detection was achieved via inverting the noisy routing matrix, a technique known as *tomography* (Zhang et al., 2005): since $\mathbf{X} \approx \mathbf{Y}\left(\mathbf{A}^\top\right)^{-1}$ this technique recovers an approximate flow TM from which anomalies can be detected by simply thresholding. More recently *anomography* techniques have emerged which detect anomalies flow volumes directly from the (cheaper to monitor) link measurements.

2.3.1.2 Subspace Method for Anomaly Detection

We briefly summarize the PCA-based anomaly detector introduced by Lakhina et al. (2004a). The authors observed that high levels of traffic aggregation on ISP backbone links cause OD flow volume anomalies to often go unnoticed because they are buried within normal traffic patterns. They also observed that although the measured data has high dimensionality N, normal traffic patterns lie in a subspace of low dimension $K \ll N$. Inferring this normal traffic subspace using PCA (which finds the principal traffic components) makes it easier to identify volume anomalies in the remaining abnormal subspace. For the Abilene Internet2 backbone network (depicted in Figure 2.6), most variance can be captured by the first $K = 4$ principal components. By comparison the network has $N = 54$ bidirectional links.

PCA is a dimensionality reduction method that chooses K orthogonal *principal components* to form a K-dimensional subspace capturing maximal variance in the data. Let $\bar{\mathbf{Y}}$ be the centered link traffic matrix, *i.e.*, with each column of $\mathbf{Y}$ is translated to have zero mean. The k^{th} principal component is computed as

$$\mathbf{v}_k \quad = \quad \underset{\mathbf{w}:\|\mathbf{w}\|=1}{\arg\max} \left\| \bar{\mathbf{Y}} \left(\mathbb{I} - \sum_{i=1}^{k-1} \mathbf{v}_i \mathbf{v}_i^\top \right) \mathbf{w} \right\| \quad ,$$

where the matrix in between the centered link TM and the candidate direction is a projection matrix that projects data onto the space orthogonal to the previously computed principal components. Thus the k^{th} principal component is chosen to be the direction that captures the maximum amount of variance in the data that is unexplained by principal components $1, \ldots, k-1$. Equivalently, the k^{th} principal component is the k^{th} eigenvector of the empirical covariance of the centered traffic matrix.

The resulting K-dimensional subspace spanned by the first K principal components $\mathbf{V}_{1:K} = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_K]$ is the *normal* traffic subspace $\mathcal{S}_n$ and has a projection matrix $\mathbf{P}_n = \mathbf{V}_{1:K}\mathbf{V}_{1:K}^\top$. The residual $(N-K)$-dimensional subspace is spanned by the remaining principal components $\mathbf{V}_{K+1:N} = [\mathbf{v}_{K+1}, \mathbf{v}_{K+2}, \ldots, \mathbf{v}_N]$. This space is the *abnormal* traffic subspace $\mathcal{S}_a$ with a corresponding projection matrix $\mathbf{P}_a = \mathbf{V}_{K+1:N}\mathbf{V}_{K+1:N}^\top = \mathbb{I} - \mathbf{P}_n$.

Volume anomalies can be detected by decomposing the link traffic into $\mathbf{y}(t) = \mathbf{y}_n(t) + \mathbf{y}_a(t)$ where $\mathbf{y}_n(t)$ is the modeled normal traffic and $\mathbf{y}_a(t)$ is the residual traffic, corresponding to projecting $\mathbf{y}(t)$ onto $\mathcal{S}_n$ and $\mathcal{S}_a$, respectively. Lakhina et al. (2004a) observed that a volume anomaly at time t typically results in a large change to $\mathbf{y}_a(t)$, which can be detected

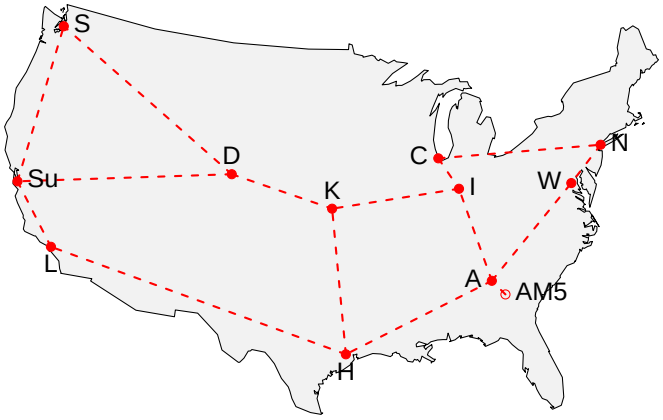


Figure 2.6: The Abilene network topology. PoPs `AM5` and `A` are located together in Atlanta; the former is displayed south-east to highlight its connectivity.

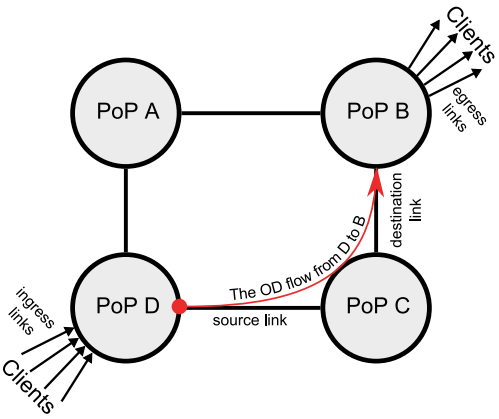


Figure 2.7: Links used for data poisoning.

by thresholding the squared prediction error $\|\mathbf{y}_a(t)\|^2$ against Q_β, the Q-statistic at the $1-\beta$ confidence level described below (Jackson and Mudholkar, 1979).

That is, the PCA-based detector classifies a link measurement vector as

$$c\left(\mathbf{y}\left(t\right)\right) \quad = \quad \begin{cases} \text{anomalous}, & \|\mathbf{y}_a(t)\|^2 > Q_\beta \\ \text{innocuous}, & \|\mathbf{y}_a(t)\|^2 \leq Q_\beta \end{cases} \quad . \tag{2.3}$$

While others have explored more efficient distributed variations of this approach (Huang et al., 2007; Li et al., 2006a,b), we focus on the basic method introduced by Lakhina et al. (2004a).

The Q-Statistic. The statistical test for the residual function, known as the Q-statistic is due to Jackson and Mudholkar (1979). The statistic is computed as a function $Q_\beta = Q_\beta(\lambda_{K+1}, \ldots, \lambda_N)$, of the $(N-K)$ non-principal eigenvalues of the covariance matrix.

$$Q_\beta \quad = \quad \phi_1 \left[\frac{c_\beta \sqrt{2\phi_2 h_0^2}}{\phi_1} + 1 + \frac{\phi_2 h_0 (h_0 - 1)}{\phi_1^2} \right]^{1/h_0} \quad ,$$

where

$$h_0 \quad = \quad 1 - \frac{2\phi_1 \phi_3}{3\phi_2^2}$$

$$\phi_i \quad = \quad \sum_{j=K+1}^{N} \lambda_j^i \ , \quad i \in \{1, 2, 3\} \ ,$$

and c_β is the $1-\beta$ percentile in the standard normal distribution. With the threshold Q_β, this statistical test can guarantee that the false alarm probability is no more than β (under assumptions on the near-normality of the traffic data).

2.3.2 Poisoning Strategies

Consider an adversary who knows an ISP uses a PCA-based anomaly detector. We take the point of view of the attacker whose goal is to evade detection. The adversary poisons the training data so the detector learns a distorted set of principal components. When the attacker later launches an attack, the PCA-based detector will fail to detect it, as a result of the poisoning. In this section, we propose a number of data poisoning schemes, each designed to increase the variance of the traffic used in training.

2.3.2.1 The Contamination Assumption

The adversary's goal is to launch a Denial of Service (DoS) attack on some victim and to have the attack traffic successfully cross an ISP's network without being detected. Figure 2.7 illustrates a simple PoP-to-PoP topology. The DoS traffic needs to traverse from a source ingress point-of-presence (PoP) node D to a sink egress PoP B of the ISP. Before launching a DoS attack, the attacker poisons the detector for a period of time, by injecting additional traffic, *chaff*, along the OD flow (*i.e.*, the D-to-B flow) that he eventually intends to attack. This kind of poisoning activity is possible if the adversary gains control over clients of an ingress PoP or if the adversary compromises a router (or set of routers) within the ingress PoP. For a poisoning strategy, the attacker needs to decide how much chaff to add, and when to do so. These choices are guided by the amount of information available to the attacker.

Attacks Exploiting Increasing Adversarial Information. We consider poisoning strategies in which the attacker has increasing amounts of information at his disposal. The weakest attacker is one that knows nothing about the traffic at the ingress PoP, and adds chaff randomly (called an *uninformed* attack). An intermediate case is when the attacker is partially informed. Here the attacker knows the current volume of traffic on the ingress link(s) that he intends to inject chaff on. Because many networks export SNMP records, an adversary might intercept this information, or possibly monitor it himself (*i.e.*, in the case of a compromised router). We call this type of poisoning a *locally-informed* attack. Although exported data from routers may be delayed in reaching the adversary, we consider the case of minimal delay for simplicity.

In a third scenario, the attacker is *globally-informed* because his global view over the network enables him to know the traffic levels on all network links. Moreover, we assume this attacker has knowledge of future traffic link levels. (Recall that in the locally-informed scheme, the attacker only knows the *current* traffic volume of a link.) Although these attacker capabilities are very unlikely, we include this in our study in order to understand the limits of variance injection poisoning schemes.

Attacks With Distant Time Horizons. Poisoning strategies can also vary according to the time horizon over which they are carried out. Most studies on the PCA-subspace method use a one week training period, so we assume that PCA is retrained each week. Thus the principal components (PCs) used in any week m are those learned during week $m-1$ with any detected anomalies removed. Thus for our poisoning attacks, the adversary inserts chaff along the target OD flow throughout the one week training period. We also consider a long-term attack in which the adversary slowly, but increasingly, poisons the PCs over several weeks, by adding small amounts of chaff, in gradually increasing quantities. We call this the *Boiling Frog* poisoning method after the folk tale that one can boil a frog by slowly increasing the water temperature over time.[4]

Adversarial Control. We assume the adversary does not have control over existing traffic (*i.e.*, he cannot delay or discard traffic). Similarly, the adversary cannot submit false SNMP reports to PCA. Such approaches to poisoning are more conspicuous because the inconsistencies in SNMP reporting from neighboring PoPs could expose the compromised router.

Remark 8. *This study focuses on non-distributed poisoning of DoS detectors.* Distributed *poisoning that aims to evade a DoS detector is also possible; our globally-informed poisoning strategy is an example, as the adversary has control over all network links. We focus on DoS for a three reasons. In the first-ever study on this topic, we aim to solve the basic problem before tackling a distributed version. Second, we point out that results on evasion via non-distributed poisoning are stronger than distributed poisoning results: the DDoS attacker can monitor and influence many more links than the DoS attacker. Hence a DoS poisoning scenario is stealthier than a DDoS one. Finally, while the main focus of current PCA-based systems is the detection of DoS attacks (Lakhina et al., 2004a,b,c, 2005a), the application of PCA to detecting DDoS attacks has so far been limited (cf. Lakhina et al. 2005b).*

For each of these scenarios of different information available to the adversary, we now outline specific poisoning schemes. In each scheme, the adversary decides on the quantity c_t of chaff to add to the target flow time series at a time t. Each strategy has an attack parameter θ, which controls the intensity of the attack. For each scenario, we present only one specific poisoning scheme.

2.3.2.2 Uninformed Chaff Selection

At each time t, the adversary decides whether or not to inject chaff according to a Bernoulli random variable with parameter 0.5. If he decides to inject chaff, the amount of chaff added is of size θ, *i.e.*, $c_t = \theta$. This method is independent of the network traffic since our attacker is uninformed, and so the variance of the poisoned traffic is increased by the variance of the chaff. The choice of a fair coin with $p = 0.5$ maximizes the variance of the chaff as $\theta^2/4$, which in general would be $p(1-p)\theta^2$. We call this the *Random* scheme.

[4]Note that there is nothing inherent in the choice of a one-week poisoning period. For a general SML algorithm, our strategies would correspond to poisoning over one training period (whatever its length) or multiple training periods.

2.3.2.3 Locally-Informed Chaff Selection

The attacker's goal is to increase traffic variance, on which the PCA detector's model is based. In the locally-informed scenario, the attacker knows the volume of traffic in the ingress link he controls, $y_S(t)$. Hence this scheme elects to only add chaff when the existing traffic is already reasonably large. In particular, we add chaff when the traffic volume on the link exceeds a parameter α (we typically use the mean, assuming that the data is reasonably stationary in the sense that the mean does not change quickly over time). The amount of chaff added is $c_t = \left(\max\{0, y_S(t) - \alpha\}\right)^{\theta}$. In other words, we take the difference between the link traffic and a parameter α and raise it to θ. In this scheme (called *Add-More-If-Bigger*), the further the traffic is from the average load, the larger the deviation of chaff inserted.

2.3.2.4 Globally-Informed Chaff Selection

The globally-informed scheme captures an omnipotent adversary with full knowledge of $\mathbf{Y}$, $\mathbf{A}$, and the future measurements $\tilde{y}_t$, and who is *capable of injecting chaff into any network flow during training*. In the poisoning schemes above, the adversary can only inject chaff along a single compromised link, whereas in this scenario, the adversary can inject chaff along any link. We formalize the problem of selecting a link n to poison, and selecting an amount of chaff $\mathbf{C}_{tn}$ as an optimization problem that the adversary solves to maximize the chances of evasion. Although these globally-informed capabilities are unrealistic in practice, we analyze globally-informed poisoning in order to understand the limits of variance injection methods and to gain insight into the poisoning strategies that exploit limited capabilities.

Ideal Objective: The PCA Evasion Problem. In the *PCA Evasion Problem* an adversary wishes to launch an undetected DoS attack of volume δ along flow f at future time t. If the vector of link volumes at time t is $\tilde{\mathbf{y}}_t$, where the tilde distinguishes this future measurement from past training data $\bar{\mathbf{Y}}$, then the vectors of anomalous DoS volumes are given by $\tilde{\mathbf{y}}'_t = \tilde{\mathbf{y}}_t + \delta * \mathbf{A}_f$. Denote by $\mathbf{C}$ the matrix of link traffic injected into the network by the adversary during training. Then the PCA-based anomaly detector is trained on altered link traffic matrix $\bar{\mathbf{Y}} + \mathbf{C}$ to produce the mean traffic vector $\boldsymbol{\mu}$, the top K eigenvectors $\mathbf{V}_{1:K}$, and the squared prediction error threshold Q_β. The adversary's objective is to enable as large a DoS attack as possible (maximizing δ) by choosing an appropriate $\mathbf{C}$. The PCA Evasion Problem corresponds to solving the following program:

$$\begin{aligned} \max_{\delta \in \mathbb{R},\ \mathbf{C} \in \mathbb{R}^{T \times F}} \quad & \delta \\ \text{s.t.} \quad & (\boldsymbol{\mu}, \mathbf{V}, Q_\beta) = \mathrm{PCA}(\mathbf{Y} + \mathbf{C}) \\ & \left\| \mathbf{V}_{K+1:N}^{\top} (\tilde{\boldsymbol{y}}'_t - \boldsymbol{\mu}) \right\|_2 \leq Q_\beta \\ & \|\mathbf{C}\|_1 \leq \theta \\ & C_{tn} \geq 0 \qquad \forall t, n \ , \end{aligned}$$

where θ is an attacker-tunable parameter constraining total chaff. The first constraint represents the output of PCA (*i.e.*, does not constrain the program's solution). The second

constraint guarantees evasion by requiring that the contaminated link volumes at time t be classified as innocuous (*cf.* Equation 2.3). The remaining constraints upper-bound the total chaff volume by θ and constrain the chaff to be non-negative, corresponding to the level of adversarial control and the contamination assumption that no negative chaff may be added to the network.

Relaxations. Unfortunately, the above optimization seems difficult to solve analytically. Thus we relax the problem to obtain a tractable analytic solution.

First the above objective seeks to maximize the attack direction $\mathbf{A}_f$'s projected length in the normal subspace, $\max_{\mathbf{C}\in\mathbb{R}^{T\times F}} \left\|\mathbf{V}_{1:K}^\top \mathbf{A}_f\right\|_2$. Next, we restrict our focus to traffic processes that generate spherical K-rank link traffic covariance matrices.[5] This property implies that the eigen-spectrum consists of K ones followed by all zeroes. Such an eigen-spectrum allows us to approximate the top eigenvectors $\mathbf{V}_{1:K}$ in the objective, with the matrix of all eigenvectors weighted by their corresponding eigenvalues $\mathbf{\Lambda V}$. We can thus convert the PCA evasion problem into the following optimization:

$$\begin{aligned} \max_{\mathbf{C}\in\mathbb{R}^{T\times F}} \quad & \left\|(\bar{\mathbf{Y}}+\mathbf{C})\mathbf{A}_f\right\|_2 \\ \text{s.t.} \quad & \|\mathbf{C}\|_1 \le \theta \\ & C_{tn} \ge 0 \qquad \forall t, n \ . \end{aligned} \tag{2.4}$$

Solutions to this optimization are obtained by a standard Projection Pursuit method from optimization: iteratively take a step in the direction of the objective's gradient and then project onto the feasible set. Finally the iteration can be initialized by relaxing the L_1 constraint on the chaff matrix to the analogous L_2 constraint and dropping the remaining constraints. This produces a differentiable program which can be solved using standard Lagrangian techniques to initialize the iteration.

Relation to Uninformed and Locally Informed Schemes. The relaxed solution to the PCA evasion problem yields an interesting insight relating to the uninformed and locally-informed chaff selection methods. Recall that the adversary is capable of injecting chaff along *any* flow. One could imagine that it might be useful to inject chaff along an OD flow whose traffic dominates the choice of principal components (*i.e.*, an elephant flow), and then send the DoS traffic along a different flow (that possibly shares a subset of links with the poisoned OD flow). However Equation (2.4) indicates that the best (relaxed) strategy to evade detection is to inject chaff only along the links $\mathbf{A}_f$ associated with the target flow f. This follows from the form of the initializer $\mathbf{C}^{(0)} \propto \bar{\mathbf{Y}}\mathbf{A}_f\mathbf{A}_f^\top$ (obtained from the L_2 relaxation for initialization) as well as the form of the projection and gradient steps. In particular, all iterates preserve the property that the solution only injects chaff along the target flow.

[5] While the spherical assumption does not hold in practice, the assumption of low-rank traffic matrices is met by published datasets (Lakhina et al., 2004a).

In fact, the only difference between our globally-informed solution and the locally-informed scheme is that the former uses information about the entire traffic matrix $\mathbf{Y}$ to determine chaff allocation along the flow whereas the latter use only local information.

This result adds credence to the intuition that in a chaff-constrained poisoning attack, all available chaff should be inserted along the target flow.

2.3.2.5 Boiling Frog Attacks

In the above attacks, poisoning occurs during a single week. We next consider a long-term attack in which the adversary slowly, but increasingly, poisons the principal components over several weeks, *starting with the second week* by adding small amounts of chaff, in gradually increasing quantities. This kind of poisoning approach is useful for adversaries that plan DoS attacks in advance of special events (like the Olympics, the World Cup soccer finals, the scheduled release of a new competing product, etc.)

Boiling Frog poisoning can use any of the preceding chaff schemes to select c_t. The amount of poisoning is increased over the duration of the Causative attack as follows. We initially set the attack parameter θ_1 to be zero, so that in the first week, no chaff is added to the training data and PCA is trained on a week of 'clean' data to establish a baseline model (representing the state of the detector prior to the start of poisoning). Over the course of the second week, the target flow is injected with chaff generated using parameter θ_2. At the week's end, PCA is retrained on that week's data with any anomalies detected by PCA during that week, excluded from the week's training set. This process continues with parameter $\theta_t > \theta_{t-1}$ used for week t.

Although PCA is retrained from scratch each week, the training data includes only those events not flagged as anomalous by the previous detector. Thus, each successive week will contain additional malicious training data, with the process continuing until the week of the DoS attack, when the adversary stops injecting chaff.

The effect of this scheme is to *slowly* rotate the normal subspace, injecting low levels of chaff relative to the previous week's traffic levels so that PCA's rejection rates stay low and a large portion of the present week's poisoned traffic matrix is trained on for the proceeding week's model.

2.3.3 ANTIDOTE: A Robust Defense

To defend against the above poisoning attacks on PCA-based anomaly detection, we explore techniques from Robust Statistics. Such methods are less sensitive to outliers, and as such are ideal defenses against variance injection schemes that perturb data to increase variance along the target flow. There has previously been two broad approaches to make PCA robust: the first computes the principal components as the eigenspectrum of a robust estimate of the covariance matrix (Devlin et al., 1981), while the second approach searches for directions that maximize a robust scale estimate of the data projection (Croux et al., 2007). We explore one of the latter methods as a counter-measure to poisoning. After describing the method, we propose a new threshold statistic that can be used for any PCA-based method including robust PCA. Robust PCA and the new robust Laplace threshold

together form a new network-wide traffic anomaly detection method, named ANTIDOTE, that is less sensitive to our poisoning attacks.

2.3.3.1 Intuition

To mitigate the effect of variance injection poisoning attacks, we need a learning algorithm that is stable in spite of data contamination; *i.e.*, a small amount of data contamination should not dramatically change the model produced by the algorithm. This concept of stability has been studied in the field of Robust Statistics where *robustness* is used to describe the notion of stability. In particular, there have been several approaches to developing robust PCA algorithms that construct a low dimensional subspace that captures most of the data's *dispersion*[6] and are stable under data contamination (Croux and Ruiz-Gazen, 2005; Croux et al., 2007; Devlin et al., 1981; Li and Chen, 1985; Maronna, 2005).

The robust PCA algorithms we considered search for a unit direction $\mathbf{v}$ on which the data projections maximize some measure of univariate dispersion $\mathrm{S}(\cdot)$; that is,

$$\mathbf{v} \in \underset{\|\mathbf{a}\|_2=1}{\arg\max}\ \mathrm{S}(\mathbf{Ya}) \ . \tag{2.5}$$

The standard deviation is the dispersion measure used by PCA; *i.e.*, $\mathrm{S}^{\mathrm{SD}}(r_1, r_2, \ldots, r_n) = \left(\frac{1}{n-1}\sum_{i=1}^{n} r_i - \bar{r}\right)^{1/2}$. However, standard deviation is sensitive to outliers making PCA non-robust to contamination. Robust PCA algorithms instead use measures of dispersion based on the concept of *robust projection pursuit (RPP)* estimators (Li and Chen, 1985). As is shown by Li and Chen (1985), RPP estimators achieve the same breakdown points[7] as their dispersion measure as well as being qualitatively robust; *i.e.*, the estimators are stable.

However, unlike the eigenvector solutions that arise in PCA, there is generally no efficiently computable solution for the maximizers of robust dispersion measures and so the solutions must be approximated. Below, we describe the PCA-GRID algorithm, a successful method for approximating robust PCA subspaces developed by Croux et al. (2007). Among the projection pursuit techniques we considered (Croux and Ruiz-Gazen, 2005; Maronna, 2005), PCA-GRID proved to be most resilient to our poisoning attacks. It is worth emphasizing that the procedure described in the next section is simply a technique for approximating a projection pursuit estimator and does not itself contribute to the algorithm's robustness—that robustness comes from the definition of the projection pursuit estimator in Equation (2.5).

To better understand the efficacy of a robust PCA algorithm, we demonstrate by example the effect our poisoning techniques have on the PCA algorithm and contrast them with the effect on the PCA-GRID algorithm. In Figure 2.8, we see the impact of a globally informed poisoning attack on both algorithms. Initially, the 'clean' data was clustered in an ellipse. In the first plot, we see that both algorithms construct reasonable estimates for the center and first principal component for this data.

[6] 'Dispersion' is an alternative term for variation since the later is often associated with statistical variation. By a dispersion measure we mean a statistic that measures the variability or spread of a variable.

[7] The *breakdown point* of an estimator is the (asymptotic) fraction of the data an adversary must control in order to arbitrarily change an estimator, and as such is a common measure of statistical robustness.

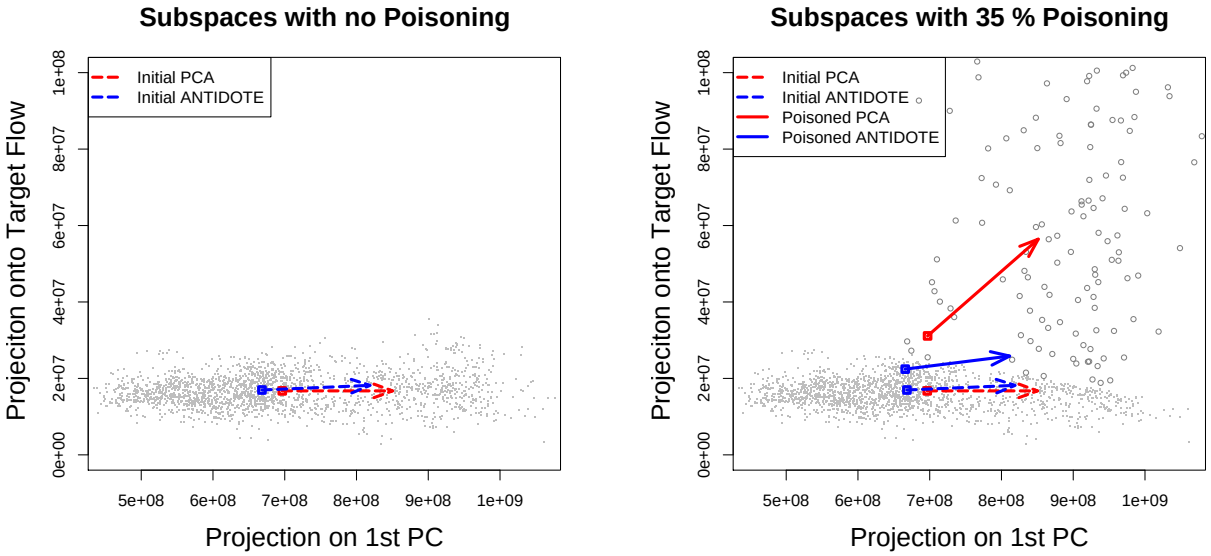


Figure 2.8: Here the data has been projected into the 2-dimensional space spanned by the 1st principal component and the direction of the attack flow #118 in the Abilene dataset. The effect on the 1st principal components of PCA and PCA-GRID is shown under a globally informed attack.

However, in the second plot, we see that a large amount of poisoning dramatically perturbs some of the data and as a result the PCA subspace is dramatically shifted toward the target flow's direction (the y-axis in this example). Due to this shift, DoS attacks along the target flow will be less detectable. Meanwhile, the subspace of PCA-GRID is noticeably less affected.

2.3.3.2 PCA-GRID

The PCA-GRID algorithm introduced by Croux et al. (2007) is a projection pursuit technique as described above. It finds a K-dimensional subspace that approximately maximizes $\mathrm{S}(\cdot)$, a *robust* measure of dispersion, for the data $\mathbf{Y}$ as in Equation (2.5). The first step of describing the algorithm is to specify the robust measure of dispersion. We use the *Median Absolute Deviation (MAD)* over other possible choices for $\mathrm{S}(\cdot)$. For scalars $r_1, \ldots, r_n$ the MAD is defined as

$$\mathrm{S}^{\mathrm{MAD}}(r_1, \ldots, r_n) \quad = \quad \omega \cdot \mathrm{median}_{i \in [n]} \left\{ \left| r_i - \mathrm{median}_{j \in [n]} \{r_j\} \right| \right\} ,$$

where the coefficient $\omega = 1.486$ ensures asymptotic consistency on normal distributions.

The next step is to choose an estimate of the data's central location. In PCA, this estimate is simply the mean of the data. However, the mean is not robust, so we center the

data using the spatial median instead:

$$\hat{c}\,(\mathbf{Y}) \quad \in \quad \underset{\boldsymbol{\mu}\in\mathbb{R}^N}{\arg\min} \sum_{i=1}^{n} \|\mathbf{y}_i - \boldsymbol{\mu}\|_2 \quad ,$$

which involves a convex optimization that is efficiently solved (see *e.g.*, Hössjer and Croux 1995).

The original projection pursuit technique for robust PCA was first proposed by Li and Chen (1985), however their method was complicated and inefficient. The desirable statistical properties of consistency, asymptotic normality (Cui et al., 2003; Li and Chen, 1985) and robustness in terms of influence functions and breakdown points (Croux and Ruiz-Gazen, 2005) have all been shown for these methods. Croux and Ruiz-Gazen (1996) introduced an efficient implementation to approximate the above objective, however Croux et al. (2007) observed that for all datasets the method of Croux and Ruiz-Gazen (1996) implodes in the presence of many variables: the lower half of the 'eigenvalues'—the estimates of scale—are identically zero. Croux et al. (2007) propose the PCA-GRID algorithm as an efficient implementation that does not suffer from this downward bias of the scale estimates.

Given a dispersion measure and location estimate, PCA-GRID finds a (unit) direction $\mathbf{v}$ that is an approximate solution to Equation (2.5). The PCA-GRID algorithm uses a grid-search for this task. Namely, suppose we want to find the best candidate between some pair of unit vectors $\mathbf{a}_1$ and $\mathbf{a}_2$ (a 2-dimensional search space). The search space is the unit circle parameterized by ϕ as $\mathbf{a}_\phi = \cos(\phi)\mathbf{a}_1 + \sin(\phi)\mathbf{a}_2$ with $\phi \in [-\pi/2, \pi/2]$. The grid search splits the domain of ϕ into a mesh of $Q+1$ candidates $\phi_k = \frac{\pi}{2}\left(\frac{2k}{Q} - 1\right)$, $k = 0, \ldots, Q$. Each candidate vector $\mathbf{a}_{\phi_k}$ is assessed and the one that maximizes $\mathrm{S}\,(\mathbf{Y}\mathbf{a}_{\phi_k})$ is chosen as the approximate maximizer $\hat{\mathbf{a}}$.

To search in a more general N-dimensional space, the grid search iteratively refines its current best candidate $\hat{\mathbf{a}}$ by performing a 2-dimensional grid search between $\hat{\mathbf{a}}$ and each of the unit directions $\mathbf{e}_i$ in turn. With each iteration, the range of angles considered progressively narrows around $\hat{\mathbf{a}}$ to better explore its neighborhood. This procedure (outlined in Algorithm 1) approximates the direction of maximal dispersion analogous to a principal component in PCA.

To find the K-dimensional subspace $\{\mathbf{v}_i \mid \mathbf{v}_i^\top \mathbf{v}_j = \delta_{i,j}\}$ that maximizes the dispersion measure, the GRID-SEARCH is repeated K-times. After each repetition, the data is deflated to remove the dispersion captured by the last direction from the data. This process is detailed in Algorithm 2.

2.3.3.3 Robust Laplace Threshold

In addition to the robust PCA-GRID algorithm, we also use a robust estimate for its residual threshold in place of the Q-statistic described in Section 2.3.1.2. Using the Q-statistic as a threshold was motivated by an assumption of normally distributed residuals (Jackson and Mudholkar, 1979). However, we found that the residuals for both the PCA and PCA-GRID subspaces were empirically non-normal leading us to conclude that the Q-statistic is a poor choice for our detection threshold. Instead, to account for the

Algorithm 1 GRID-SEARCH($\mathbf{Y}$)

Require: $\mathbf{Y}$ is a $T \times N$ matrix
Let: $\hat{\mathbf{v}} = \mathbf{e}_1$;
for $i = 1$ to C **do**
  for $j = 1$ to N **do**
    for $k = 0$ to Q **do**
      Let: $\phi_k = \frac{\pi}{2^i}\left(\frac{2k}{Q} - 1\right)$;
      Let: $\mathbf{a}_{\phi_k} = \cos(\phi_k)\hat{\mathbf{a}} + \sin(\phi_k)\mathbf{e}_j$;
      if $\mathrm{S}(\mathbf{Y}\mathbf{a}_{\phi_k}) > \mathrm{S}(\mathbf{Y}\hat{\mathbf{v}})$ **then**
        Assign: $\hat{\mathbf{v}} \leftarrow \mathbf{a}_{\phi_k}$;
Return: $\hat{\mathbf{v}}$;

Algorithm 2 PCA-GRID($\mathbf{Y}, K$)

Center $\mathbf{Y}$: $\mathbf{Y} \leftarrow \mathbf{Y} - \hat{c}(\mathbf{Y})$;
for $i = 1$ to K **do**
  $\mathbf{v}_i \leftarrow$ GRID-SEARCH($\mathbf{Y}$);
  $\mathbf{Y} \leftarrow$ projection of $\mathbf{Y}$ onto the complement of $\mathbf{v}_i$;
end for
Return subspace centered at $\hat{c}(\mathbf{Y})$ with principal directions $\{\mathbf{v}_i\}_{i=1}^{K}$;

outliers and heavy-tailed behavior we observed from our method's residuals, we choose our threshold as the $1-\beta$ quantile of a Laplace distribution. It is important to note that while we do not believe the residuals to be distributed according to a Laplace, a Laplace model better models the heavy-tailed nature observed in the data. Our detector, ANTIDOTE is the combination of the PCA-GRID algorithm and the Laplace threshold. The non-normality of the residuals has also been recently pointed out by Brauckhoff et al. (2009).

As with the previous Q-statistic method described in Section 2.3.1.2, we select our threshold $Q_{L,\beta}$ as the $1-\beta$ quantile of a parametric distribution fit to the residuals in the training data. However, instead of the normal distribution assumed by the Q-statistic, we use the quantiles of a Laplace distribution specified by a location parameter c and a scale parameter b. Critically, though, instead of using the mean and standard deviation, we robustly fit the distribution's parameters. We estimate c and b from the residuals $\|\mathbf{y}_a(t)\|^2$ using robust consistent estimates of location (median) and scale (MAD)

$$\begin{aligned} \hat{c} &= \text{median}\left(\|\mathbf{y}_a(t)\|^2\right) \quad , \\ \hat{b} &= \frac{1}{\sqrt{2}P^{-1}(0.75)}\,\text{median}\left\{\left|\|\mathbf{y}_a(t)\|^2 - \hat{c}\right|\right\} \quad , \end{aligned}$$

where $P^{-1}(q)$ is the q^{th} quantile of the standard Laplace distribution. The Laplace quantile function has the form $P^{-1}_{c,b}(q) = c + b \cdot k(q)$ for some $k(q)$. Thus, our threshold only depends linearly on the (robust) estimates $\hat{c}$ and $\hat{b}$ making the threshold itself robust. This form is also shared by the normal quantiles (differing only in the function k), but because non-

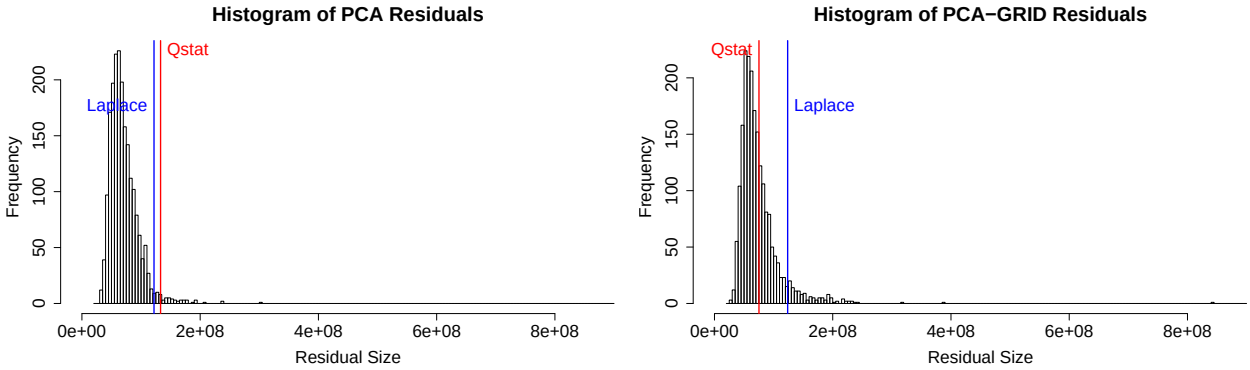


Figure 2.9: Histograms of the residuals for the original PCA algorithm (left) and the PCA-GRID algorithm (the largest residual is excluded as an outlier). Red and blue vertical lines demarcate the threshold selected using the Q-statistic and the Laplace threshold, respectively.

robust estimates for c and b are implicitly used by the Q-statistic, it is not robust. Further, by choosing a heavy-tailed distribution like the Laplace, the quantiles are more appropriate for the heavy-tails we observed.

Empirically, the Laplace threshold appears to be better suited for thresholding the residuals of our robust models than the Q-statistic. As can be seen in Figure 2.9, both the Q-statistic and the Laplace threshold produce a reasonable threshold on the residuals of the PCA algorithm but only the Laplace threshold produces a reasonable threshold for the residuals of the PCA-GRID algorithm; the Q-statistic vastly under-estimates the spread of the residuals. As was consistently seen throughout our experiments, the Laplace threshold proved to be a more reliable threshold than the Q-statistic for robust PCA.

2.3.4 Methodology

We now describe in-depth the experimental methodology used to measure the performance of the proposed poisoning strategies for PCA and the performance of ANTIDOTE as a counter-measure for variance injection attacks.

2.3.4.1 Traffic Data

We use OD flow data collected from the Abilene (Internet2 backbone) network to simulate attacks on PCA-based anomaly detection. Data was collected over an almost continuous 6 month period from March 1, 2004 through September 10, 2004 (Zhang et al., 2005). Each week of data consists of 2016 measurements across all 144 network OD flows binned into 5 minute intervals. At the time of collection the network consisted of 12 PoPs and 15 inter-PoP links. 54 virtual links are present in the data corresponding to two directions for each inter-PoP link and an ingress and egress link for each PoP. See Figure 2.6 for the Abilene network topology.

2.3.4.2 Validation

To evaluate the PCA subspace method and ANTIDOTE in the face of poisoning and DoS attacks, we use two consecutive weeks of data—the first for training and the second for testing. The poisoning occurs throughout the training phase, while the attack occurs during the test week. An alternate method (described in Section 2.3.4.3 below) is needed for the Boiling Frog scheme where training and poisoning occur over multiple weeks. Our performance metric for measuring the success of the poisoning strategies is through their impact on a PCA-based detector's *false negative rate* (FNR). The FNR is the ratio of the number of successful evasions to the total number of attacks (*i.e.*, the attacker's success rate is PCA's FNR rate). We also use Receiver Operating Characteristic (ROC) curves to visualize a detection method's trade-off between *detection rate* (TPR) and *false positive rate* (FPR).

In order to compute the FNRs and FPRs, we generate synthetic anomalies according to the method of Lakhina et al. (2004a) and inject them into the Abilene data. While there are disadvantages to this method, such as the conservative assumption that a single volume size is anomalous for all flows, we adopt it for the purposes of relative comparison between PCA and Robust PCA, to measure relative effects of poisoning, and for consistency with prior studies. We use week-long training sets, as such a time scale is sufficiently large to capture weekday and weekend cyclic trends (Ringberg et al., 2007), and previous studies operated on this same time scale (Lakhina et al., 2004a). There is nothing inherent to our method that limits its use to this time scale; our methods will work as long as the training data is poisoned throughout. Because the data is binned in 5 minute windows (corresponding to the reporting interval of SNMP), a decision about whether or not an attack is present can be made at the end of each 5 minute window; thus attacks can be detected within 5 minutes of their occurrence. We now describe the method of Lakhina et al. (2004a) adopted here.

Starting with the flow traffic matrix $\mathbf{X}$ for the test week, we generate a positive example (an anomalous OD flow) by setting flow f's volume at time t, $X_{t,f}$, to be a large value known to correspond to an anomalous flow (replacing the original traffic volume in this time slot). This value[8] is defined (Lakhina et al., 2004a) to be 1.5 times a cutoff of 8×10^7. After multiplying by the routing matrix $\mathbf{A}$, the link volume measurement at time t is anomalous. We repeat this process for each time t (each 5 minute window) in the test week to generate a set of 2016 anomaly samples for the single target flow f.

In order to obtain FPRs, we generate negative examples (benign OD flows) as follows. We fit the data to an exponentially weighted moving average (EWMA) model that is intended to capture the main trends of the data without much noise. We use this model to select which points in time, in an Abilene flow's time series, to use as negative examples. We compare the actual observations and the EWMA model, and if the difference is small (not in the flow's top one percentile) for a particular flow at a particular time, $X_{t,f}$, then we label the measurement $X_{t,f}$ as "benign." We do this across all flows; when we find time slots where all flows are labeled as benign, we run our detectors and see whether or not they

[8]The cutoff was determined by fitting a basis of sinusoids of periods 7, 5, 3 days, 24, 12, 6, 3 and 1.5 hours to flow traffic and identifying the original flow volume corresponding to a steep drop to the rank-ordered residuals.

raise an alarm for those time slots.

We simulate a DoS attack along every flow at every time, one-at-a-time. We average FNRs over all 144 possible anomalous flows and all 2016 anomaly times. When reporting the effect of an attack on traffic volumes, we first average over links within each flow then over flows. Furthermore we generally report average volumes relative to the pre-attack average volumes. Thus a single poisoning experiment was based on one week of poisoning with FNRs computed during the test week that includes 144×2016 samples coming from the different flows and time slots. Because the poisoning is deterministic in *Add-More-If-Bigger* this experiment was run once for that scheme. In contrast, for the *Random* poisoning scheme, we ran 20 independent repetitions of the poisoning experiment to average-out the effects of randomness in each individual run.

To produce the ROC curves, we use the squared prediction errors produced by the detection methods, that consist of anomalous and normal examples from the test set. By varying the method's threshold (usually fixed as the Q-statistic or the Laplace threshold) from $-\infty$ to ∞ a curve of possible (FPR, TPR) pairs is produced from the set of SPE's; the Q-statistic and Laplace threshold, each correspond to one such point in ROC space. We adopt the Area Under Curve (AUC) statistic from Information Retrieval to directly compare ROC curves since one curve out of a pair of curves does not always dominate the other. The area under an ROC curve of detector $\mathcal{A}$ estimates the conditional probability

$$AUC(\mathcal{A}) \approx \Pr\left(SPE_{\mathcal{A}}(\mathbf{y}_1) > SPE_{\mathcal{A}}(\mathbf{y}_2)\right) ,$$

given anomalous and normal random link volume vectors $\mathbf{y}_1$ and $\mathbf{y}_2$. The ideal detector has an AUC of 1, while the random predictor achieves an AUC of 0.5.

2.3.4.3 Single Period and Boiling Frog Poisoning

We evaluate the effectiveness of our attacker strategies using weeks 20 and 21 from the Abilene dataset to simulate the *Single-Training Period* attacks. The PCA algorithm is trained on the week 20 traffic matrix poisoned by the attacker; we then inject attacks during week 21 to see how often the attacker can evade detection. We select these particular weeks because PCA achieved the lowest FNRs on these during testing.

To test the *Boiling Frog* attack we simulate traffic matrix data, inspired by methods used by Lakhina et al. (2004a). Our simulations present multiple weeks of stationary data to the adversary. While such data is unrealistic in practice, it is an easy case on which PCA should succeed. Anomaly detection under non-stationary conditions is difficult due to the learner's inability to distinguish between benign data drift, and adversarial poisoning. Thus demonstrated flaws of PCA in the stationary case constitute strong results. We decided to validate the *Boiling Frog* attack on a synthesized multi-week dataset, because the 6 month Abilene dataset of Zhang et al. (2005) proved to be too non-stationary for PCA to consistently operate well from one week to the next. It is unclear whether the non-stationarity observed in this data is prevalent in general or whether it is an artifact of the dataset.

We synthesize a multi-week set of OD flow traffic matrices, with stationarity on the inter-week level. We use a three step generative procedure to model each OD flow separately from

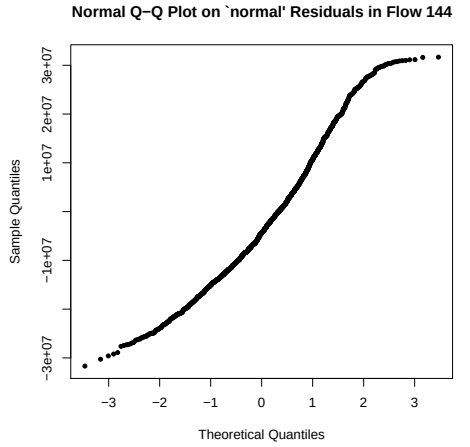
the real-world Abilene dataset. First the underlying daily cycle of the OD flow f time series is modeled by a sinusoidal approximation. Then the times at which the flow is experiencing an anomaly are modeled by a Binomial arrival process with inter-arrival times distributed according to the geometric distribution. Finally Gaussian white noise is added to the base sinusoidal model during times of benign OD flow traffic; and exponential traffic is added to the base model during times of anomalous traffic. We next describe the process of fitting this generative model to the week 20 Abilene data in more detail.

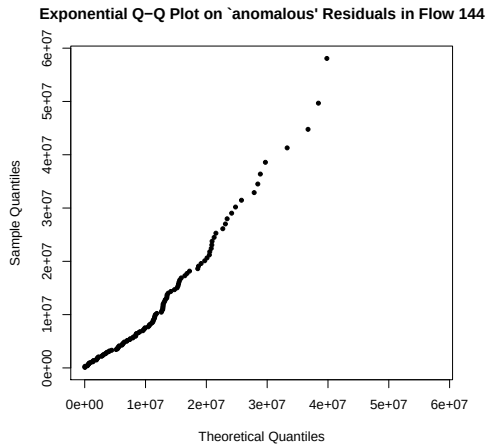
In step 1, we capture the underlying cyclic trends via Fourier basis functions. We use sinusoids of periods of 7, 5 and 3 days, and 24, 12, 6, 3 and 1.5 hours, as well as a constant function (Lakhina et al., 2004a). For each OD flow, we find the Fourier coefficients from the flow's projection onto this basis. We next remove the portion of the traffic modeled by this Fourier forecaster and model the remaining residual traffic via two processes. One is a noise process modeled by a zero-mean Gaussian to capture short-term benign traffic variance. The second process models volume anomalies as being exponentially distributed. Anomalies existing in the Abilene data result in the necessity of such a model.

In step 2 we select which of the two noise processes is used at each time interval. After computing our model's residuals (the difference between the observed and traffic predicted by the sinusoidal model) we note the smallest negative residual value $-m$. We assume that residuals in the interval $[-m, m]$ correspond to benign traffic and that residuals exceeding m correspond to traffic anomalies. We separate benign variation and anomalies in this way since these effects behave quite differently. (This is an approximation but it works reasonably well for most OD flows.) Negative residual traffic reflects benign variance, and since we assume that benign residuals have a zero-mean distribution, it follows that such residuals should lie within the interval $[-m, m]$. Upon classifying residual traffic as benign or anomalous we then model anomaly arrival times as a Bernoulli arrival process. Under this model the inter-anomaly arrival times become geometrically distributed. Since we consider only spatial PCA methods, the placement of anomalies is of secondary importance.

For the final step, the parameters for the two residual traffic volume and the inter-anomaly arrival processes are inferred from the residual traffic using the Maximum Likelihood estimates of the Gaussian's variance and exponential and geometric rates respectively.

We include goodness-of-fit results for four OD flows: flow 144 which maximizes mean and variance among all 144 flows; flow 113 which has one of the smallest means and variances among all 144 flows; and flows 15 and 75 which have median mean and variance, respectively, among all 144 flows. After manual inspection on all flows we believe these flows to be representative elephant, mouse and two mid-level flows, respectively. Figures 2.10–2.14 include evaluations of the fit of the Gaussian, Exponential and Geometric distributions to the three processes via quantile-quantile plots. In general the Gaussian and Exponential Q-Q plots for the traffic volume processes are close to linear illustrating good fits. The Q-Q plots for the Geometric inter-anomaly arrival times, in Figure 2.14 shows more variable results. However we consider only spatial PCA methods in this work so the placement of anomalies is of secondary importance. For each of the four flows, we also plot the time series for a week of both the Abilene data and our simulated model. These results establish the suitability of this model for the purpose of evaluating the *Boiling Frog* attack.





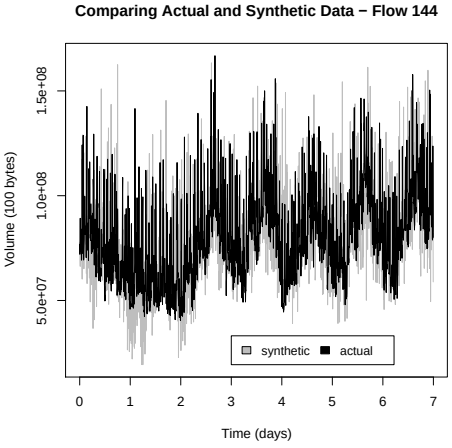
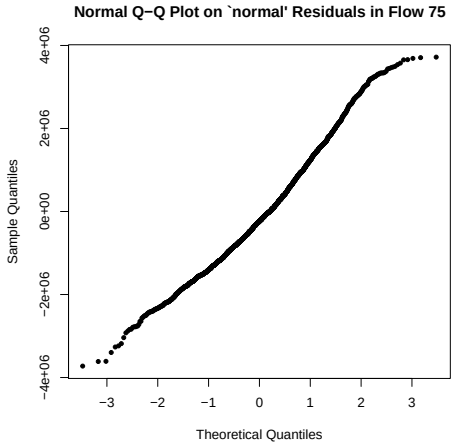
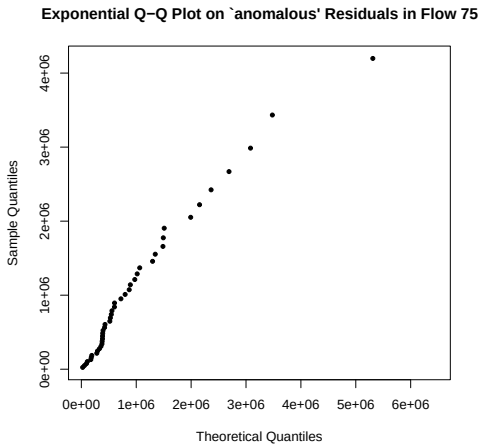


Figure 2.10: For flow 144: (top) Gaussian Q-Q plot of normal residuals; (middle) exponential Q-Q plot of anomalous residuals; (bottom) simulated time series in gray.





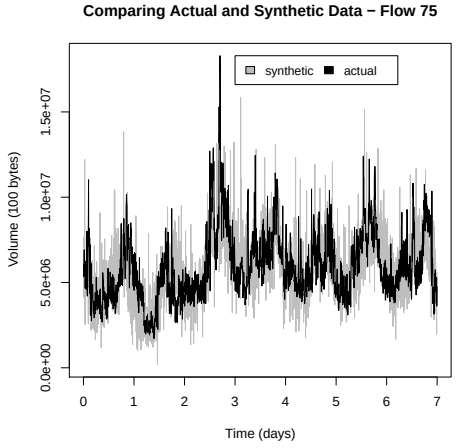
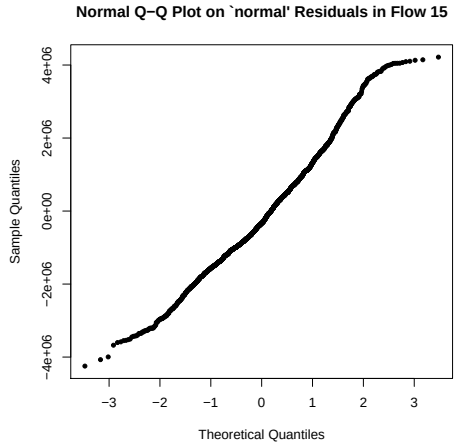
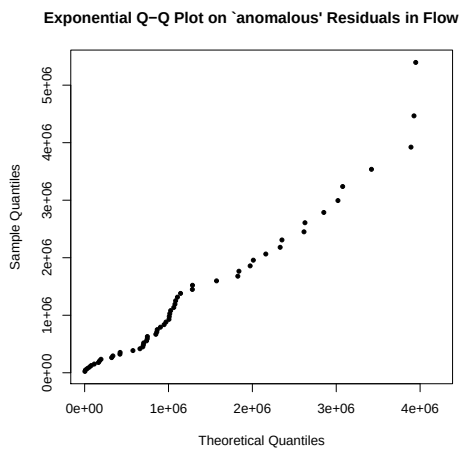


Figure 2.11: For flow 75: (top) Gaussian Q-Q plot of normal residuals; (middle) exponential Q-Q plot of anomalous residuals; (bottom) simulated time series in gray.





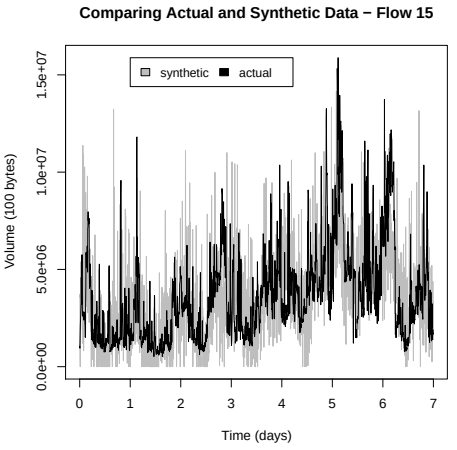
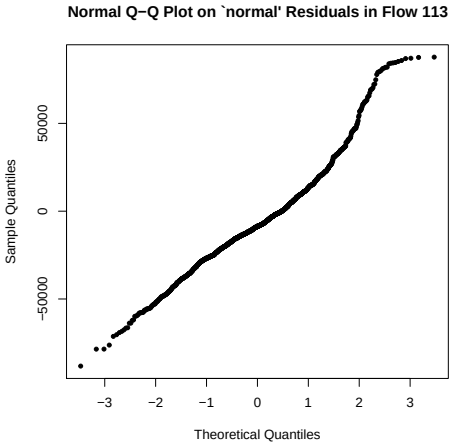
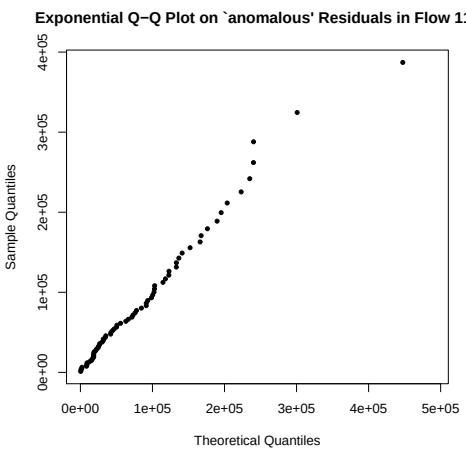


Figure 2.12: For flow 15: (top) Gaussian Q-Q plot of normal residuals; (middle) exponential Q-Q plot of anomalous residuals; (bottom) simulated time series in gray.





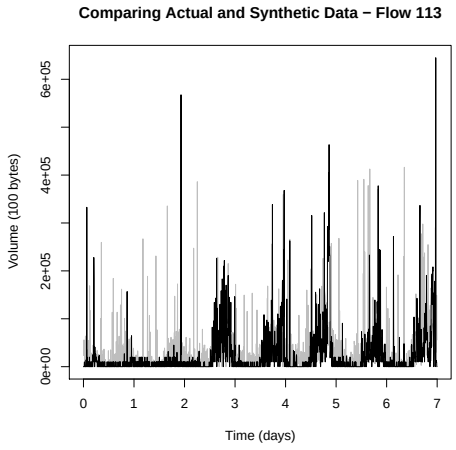


Figure 2.13: For flow 113: (top) Gaussian Q-Q plot of normal residuals; (middle) exponential Q-Q plot of anomalous residuals; (bottom) simulated time series in gray.

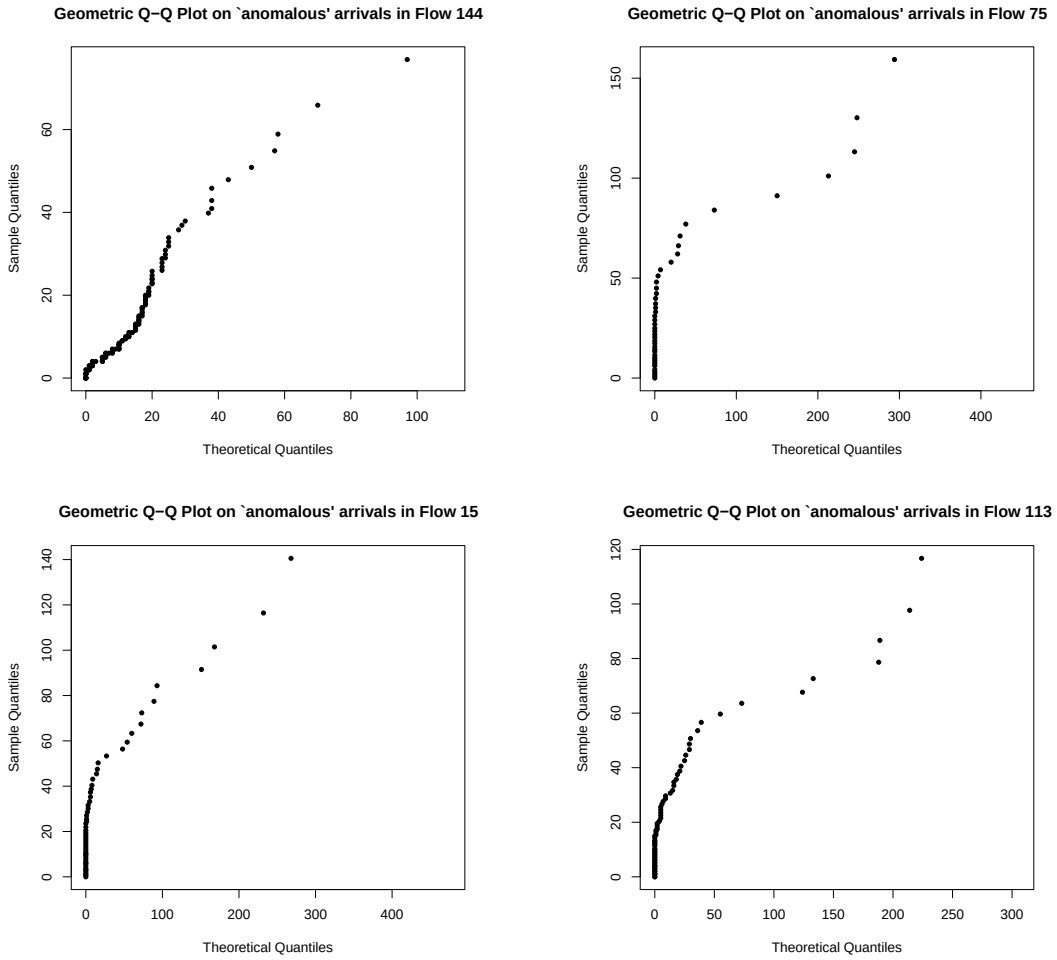


Figure 2.14: The geometric Q-Q plot of the inter-arrival times for flows 144 (top left), 75 (top right), 15 (bottom left), and 113 (bottom right).

In our simulations, we constrain all link volumes to respect the link capacities in the Abilene network: 10gbps for all but one link that operates at one fourth of this rate. We cap chaff that would cause traffic to exceed the link capacities.

2.3.5 Poisoning Effectiveness

We now present the results of the aforementioned experiments for evaluating the poisoning strategies on PCA-based detection.

2.3.5.1 Single-Training Period Poisoning: Attacker Capabilities vs. Success

We begin by measuring the evasive success of our poisoning strategies, paying special attention to the effect of adversarial information and control. We then proceed to explore

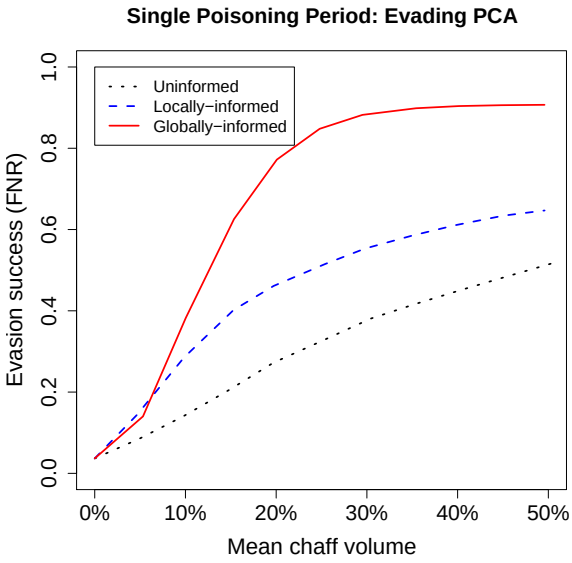


Figure 2.15: Success of evading PCA under *Single-Training Period* poisoning attacks using 3 chaff methods.

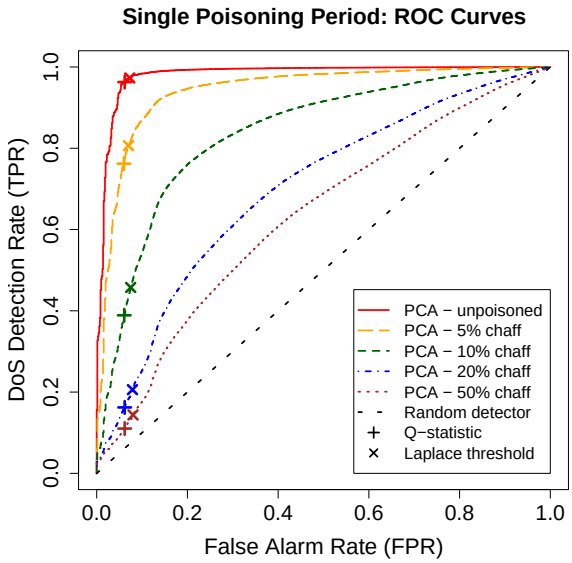


Figure 2.16: ROC curves of PCA under *Single-Training Period* poisoning attacks.

the overall performance of PCA-based detection when trained on poisoned data.

Measuring Evasive Success. We evaluate the effectiveness of our three data poisoning schemes in *Single-Training Period* attacks. During the testing week, the attacker launches a DoS attack in each 5 minute time window. The results of these attacks are displayed in Figure 2.15. Although our poisoning schemes focus on adding variance, the mean traffic of the OD flow being poisoned increases as well, increasing the means of all links over which the OD flow traverses. The x-axis in Figure 2.15 displays the relative increase in the mean rate. We average over all experiments (*i.e.*, over all OD flows). Representative numerical results are summarized in Table 2.2.

As expected the increase in evasion success is smallest for the uninformed strategy, intermediate for the locally-informed scheme, and largest for the globally-informed poisoning scheme. *The more adversarial control, the more effective the poisoning attack.* A locally-informed attacker can use the *Add-More-If-Bigger* scheme to raise his evasion success to 28% from the baseline FNR of 3.67% via a 10% average increase in the mean link rates due to chaff. Although 28% may not be viewed as a high likelihood of evasion, the attacker success rate is nearly 8 times larger than the unpoisoned PCA model's rate. This number represents an average over attacks launched in each 5 minute window, so the attacker could simply retry multiple times. With our *Globally-Informed* with a 10% average increase in the mean link rates, the unpoisoned FNR is raised by a factor of 10 to 38% and eventually to over 90%.

Poisoning scheme	**Type**	**FNR (5%)**	**FNR (10%)**
Random	Uninformed	5.21% (×1.4)	20.28% (×5.5)
Add-More-If-Bigger	Locally-informed	9.98% (×2.7)	28.33% (×7.7)
Globally-Informed	Globally-informed	13.36% (×3.6)	37.79% (×10.3)

Table 2.2: *Single-Training Period* attacks using the three poisoning schemes. Test FNRs are given for chaff that increases attacked link volumes by 5% and 10%. These results correspond to the curves in Fig. 2.15 at 1.05 and 1.1. Alongside each FNR is the multiplicative increase to the baseline FNR of 3.67%.

The big difference between the performance of the locally-informed and globally-informed attacker is intuitive to understand. Recall that the globally-informed attacker knows a great deal more (traffic on all links, and future traffic levels) than the locally-informed one (who only knows the traffic status of a single ingress link). *We consider the locally-informed adversary to have succeeded quite well with only a small view of the network.* An adversary is unlikely to be able to acquire, in practice, the capabilities used in the globally-informed poisoning attack. Moreover, adding 30% chaff, in order to obtain a 90% evasion success is dangerous in that the poisoning activity itself is likely to be detected. Therefore *Add-More-If-Bigger* presents a nice trade-off, from the adversary's point of view, in terms of poisoning effectiveness, and attacker capabilities and risks. We therefore use *Add-More-If-Bigger*, the locally-informed strategy, for many of the remaining experiments.

Measuring Overall Detector Performance Under Poisoning. We evaluate the PCA detection algorithm on both anomalous and normal data, as described in Section 2.3.4.2, producing the Receiver Operating Characteristic (ROC) curves displayed in Figure 2.16. We produce an ROC curve (as shown) by first training a PCA model on the unpoisoned data from week 20. We next evaluate the algorithm when trained on data poisoned by *Add-More-If-Bigger*.

To validate PCA-based detection on poisoned training data, we poison exactly one flow at a time as dictated by the threat model. Thus, for relative chaff volumes ranging from 5% to 50%, *Add-More-If-Bigger* chaff is added to each flow separately to construct 144 separate training sets and 144 corresponding ROC curves for the given level of poisoning. The poisoned curves in Fig. 2.16 display the averages of these ROC curves (*i.e.*, the average TPR over the 144 flows for each FPR).

We see that the poisoning scheme can throw off the balance between false positives and false negatives of the PCA detector: The detection and false alarm rates drop together rapidly as the level of chaff is increased. At 10% relative chaff volume performance degrades significantly from the ideal ROC curve (lines from $(0, 0)$ to $(0, 1)$ to $(1, 1)$) and at 20% the PCA's mean ROC curve is already close to that of blind randomized prediction (the $y = x$ line with 0.5 AUC). *Poisoning its training data dramatically reduces the overall efficacy of the PCA-based detector.*

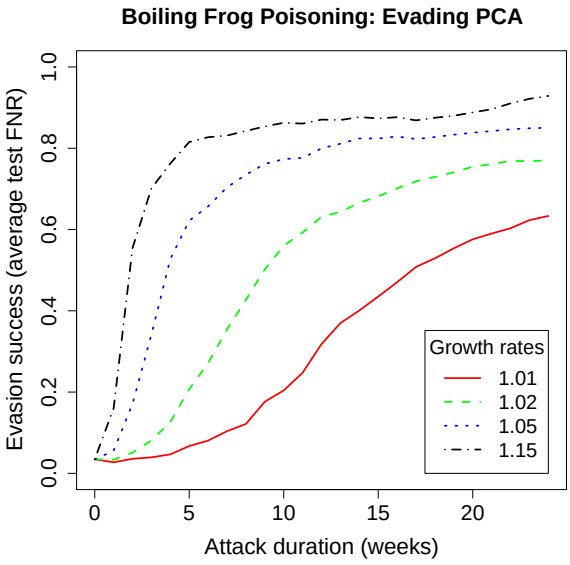


Figure 2.17: Evasion success of PCA under *Boiling Frog* poisoning attacks.

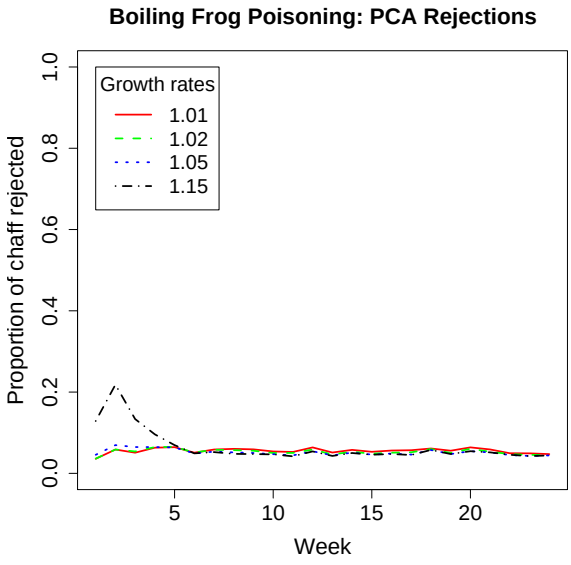


Figure 2.18: Chaff rejection rates of PCA under poisoning attacks shown in Figure 2.17.

2.3.5.2 Multi-Training Period Poisoning

We now evaluate the effectiveness of the *Boiling Frog* strategy, that contaminates the training data over multiple training periods. In Figure 2.17 we plot the FNRs against the poisoning duration for the PCA detector. We examine four different poisoning *schedules* with growth rates g as 1.01, 1.02, 1.05 and 1.15 respectively. The goal of the schedule is to increase the attacked links' average traffic by a factor of g from week to week. The attack strength parameter θ (see Section 2.3.2) is chosen to achieve this goal. We see that the FNR dramatically increases for all four schedules as the poison duration increases. With a 15% growth rate the FNR is increased to more than 70% from 3.67% over 3 weeks of poisoning; even with a 5% growth rate the FNR is increased to 50% over 3 weeks. Thus Boiling Frog *attacks are effective even when the amount of poisoned data increases rather slowly.*

Recall that the detector is retrained every week using the data collected from the previous week. However, the data from the previous week is first filtered by the detector itself. At any time point flagged as anomalous, the training data is thrown out. Figure 2.18 shows the proportion of chaff rejected each week by PCA—*chaff rejection rate*—for the *Boiling Frog* strategy. The three slower schedules enjoy a relatively small constant rejection rate close to 5%. The 15% schedule begins with a relatively high rejection rate, but after a month sufficient amounts of poisoned traffic mis-train PCA after which point the rates drop to the level of the slower schedules. *We conclude that the* Boiling Frog *strategy with a moderate growth rate of 2–5% can significantly poison PCA, dramatically increasing its FNR while still going unnoticed by the detector.*

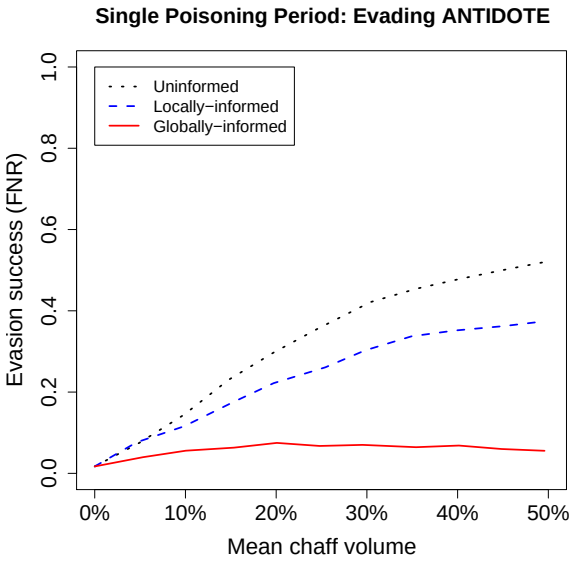


Figure 2.19: Evasion success of ANTIDOTE under *Single-Training Period* poisoning attacks using 3 chaff methods.

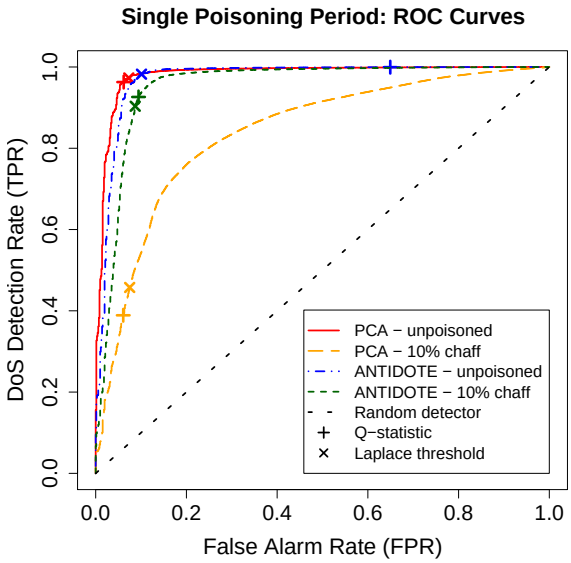


Figure 2.20: ROC curves of ANTIDOTE vs. PCA under *Single-Training Period* poisoning attacks.

By comparing Figures 2.15 and 2.17, we observe that in order to raise the FNR to 50%, an increase in mean traffic of roughly 18% for the *Single-Training Period* attack is needed, whereas in the *Boiling Frog* attack the same thing can be achieved with only a 5% average traffic increase spread across 3 weeks. *The* Boiling Frog *attack is much more stealthy than the* Single-Training Period *attack.*

2.3.6 Defense Performance

We now assess how ANTIDOTE performs in the face of two types of poisoning attacks, one that lasts a single training period, and one that lasts for multiple training periods. For the longer time horizon, we use the *Add-More-If-Bigger* poisoning scheme to select how much chaff to add at each point in time. We compare its performance to the original PCA-subspace method.

2.3.6.1 Single-Training Period Poisoning

Measuring Evasive Success. In Figure 2.19 we illustrate ANTIDOTE's FNR for various levels of average poisoning that occur in a *Single-Training Period* attack. We can compare this to Figure 2.15 that shows the same metric for the original PCA solution. We see here that the evasion success of the attack is dramatically reduced. For any particular level of chaff, the evasion success rate is approximately cut in half. Interestingly, the most effective poisoning scheme on PCA, *Globally-Informed*, is the most ineffective poisoning scheme in

the face of our robust PCA solution. We believe the reason for this is that our *Globally-Informed* scheme was designed to specifically circumvent PCA. Now that the detector has changed, *Globally-Informed* is no longer optimized for the active detector. For the new detector, *Random* remains equally effective because constant shifts in a large subset of the data create a bimodality that is difficult for any subspace method to reconcile. This effect is still muted compared to the dramatic success of locally-informed methods on the original detector. Further, constant shift poisoning creates unnatural traffic patterns that we believe can be detected. Given this evidence. We conclude that ANTIDOTE *is an effective defense against realistic poisoning attacks.*

Measuring Overall Detector Performance Under Poisoning. Since poisoning activities distort a detector, it will affect not only the FNRs but also the false positives. To explore this trade-off, we use ROC curves in Figure 2.20 for both ANTIDOTE and PCA. For comparison purposes, we include cases when the training data is both unpoisoned and poisoned. For the poisoned training scenario, each point on the curve is the average over 144 poisoning scenarios in which the training data is poisoned along one of the 144 possible flows. While ANTIDOTE performs very similarly to PCA on unpoisoned training data, PCA significantly under-performs ANTIDOTE in the presence of poisoning. With a moderate mean chaff volume of 10%, ANTIDOTE's average ROC curve remains almost unchanged while PCA's curve collapses towards the $y = x$ curve of the blind random detector. This means that the normal balance between FNRs and false positives is completely thrown off with PCA; however ANTIDOTE continues to retain a good operating point for these two common performance measures. *In summary, when we consider the two performance measures of FNRs and FPRs, we give up insignificant performance shifts when using* ANTIDOTE *when no poisoning events occur, yet we see enormous performance gains for both metrics when poisoning attacks do occur.*

Given Figures 2.19 and 2.20 alone, it is conceivable that ANTIDOTE outperforms PCA only on average, and not on all flows that could be targeted for poisoning. In place of plotting all 144 poisoned ROC curves, we display the areas under these curves (AUC) for the two detection methods in Figure 2.21 under 10% chaff targeting each of the 144 flows individually. Not only is average performance much better for robust PCA, but it enjoys better performance for more flows and by a large amount. We note that although PCA performs slightly better for some flows, we see that in fact both methods have excellent detection performance (because their AUCs are close to 1), and hence the distinction between the two is insignificant, for those specific flows. In summary ANTIDOTE *enjoys significantly superior performance for the majority of poisoned flows, while PCA's performance is only ever superior by a small margin.*

Figure 2.22 plots the mean AUC (averaged from the 144 ROC curves' AUCs where flows are poisoned separately) achieved by the detectors, as the level of chaff is intensified. Notice that ANTIDOTE behaves similarly to PCA under zero-chaff conditions, yet its performance quickly becomes superior as the amount of contamination grows. In fact, it does not take much poisoning for ANTIDOTE to exhibit much stronger performance. With PCA's performance drop, it starts approaching a random detector (equivalent to 0.5 AUC), for amounts

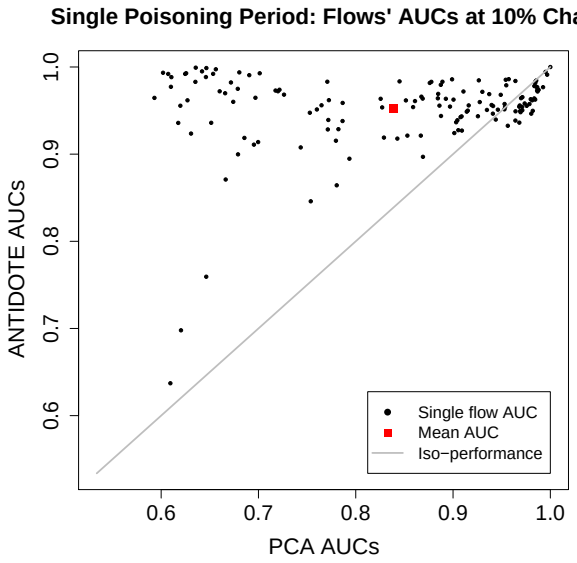


Figure 2.21: The 144 AUCs from the poisoned ROC curves for each possible target flow and their mean.

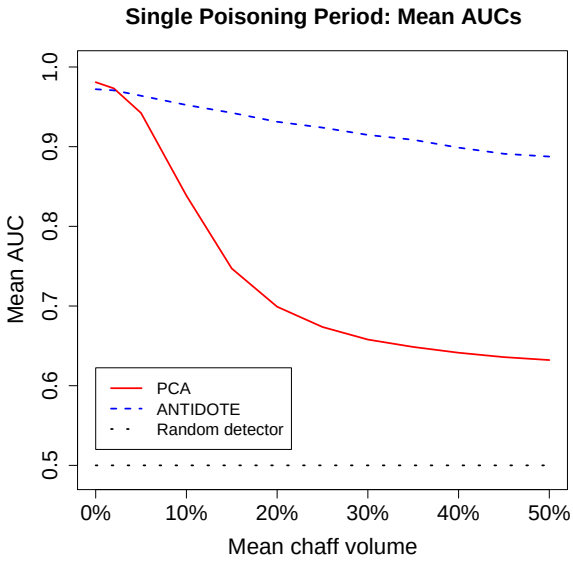


Figure 2.22: The mean AUCs versus mean chaff levels for ANTIDOTE and PCA.

of chaff exceeding 20%.

In these last few figures, we have seen the FNR and FPR performance as it varies across flows and quantity of poisoning. In all cases, it is clear that ANTIDOTE is an effective defense and dramatically outperforms a solution that was not designed to be robust. We believe this evidence indicates that the robust techniques are a promising avenue for SML algorithms used for security applications.

2.3.6.2 Multi-Training Period Poisoning

We now evaluate the effectiveness of ANTIDOTE against the *Boiling Frog* strategy, that occurs over multiple successive training periods. In Figure 2.23 we see the FNRs for ANTIDOTE with the four different poisoning schedules. We observe two interesting behaviors. First, for the two most stealthy poisoning strategies (1.01 and 1.02), ANTIDOTE shows remarkable resistance in that the evasion success increases very slowly, *e.g.*, after 10 training periods it is still below 20%. This is in stark contrast to PCA (see Figure 2.17) in which, for example, after 10 weeks, the evasion success is over 50% for the 1.02 poisoning growth rate scenario. Second, under PCA the evasion success keeps rising over time. However with ANTIDOTE under the heavier poisoning strategies, we see that the evasion success actually starts to decrease after some time. The reason for this is that ANTIDOTE has started rejecting so much of the training data, that the poisoning strategy starts to lose its effectiveness.

To look more closely at this behavior we show the proportion of chaff rejected by ANTIDOTE under multi-training period poisoning episodes in Figure 2.24. We see that the two

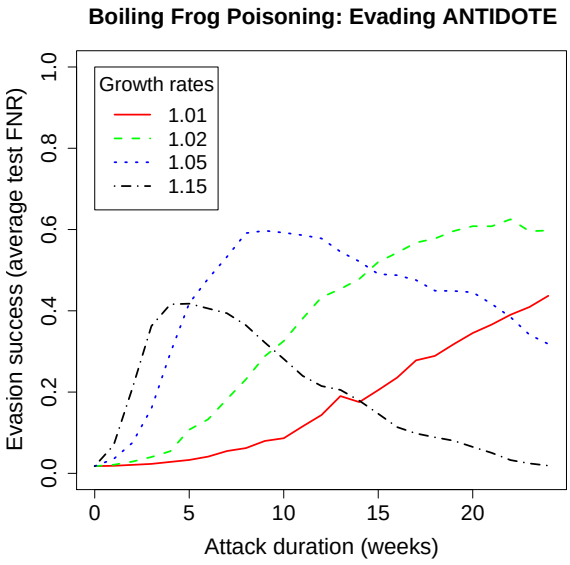


Figure 2.23: Evasion success of ANTIDOTE under *Boiling Frog* poisoning attacks.

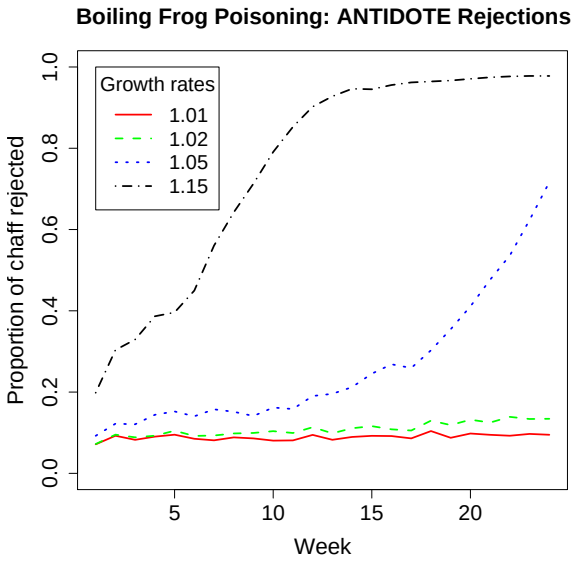


Figure 2.24: Chaff rejection rates of ANTIDOTE under *Boiling Frog* poisoning attacks.

slower schedules almost have a constant rejection rate close to 9%, which is higher than that of original PCA (which is close to 5%). For the faster poisoning growth schedules (5% and 15%) we observe that ANTIDOTE rejects an increasing amount of the poison data. This reflects a good target behavior for any robust detector: to reject more training data as the contamination grows. From these figures we conclude that *the combination of techniques we use in* ANTIDOTE*, namely a PCA-based detector designed with robust dispersion goals combined with a Laplace-based cutoff threshold, is very effective at maintaining a good balance between false negative and false positive rates throughout a variety of poisoning scenarios (different amounts of poisoning, on different OD flows, and on different time horizons).*

2.4 Summary

In this chapter we investigate two large case-studies on Causative attacks on Statistical Machine Learning systems—attacks in which the adversary manipulates the learner by poisoning its training data.

In the first case-study we show that an adversary can effectively disable the SpamBayes email spam filter, by increasing its False Positive Rate (an Availability attack), with relatively little system state information and relatively limited control over the training data. The Usenet dictionary attack causes misclassification of 36% of legitimate ham messages with only 1% control over the training messages, rendering SpamBayes unusable. Our focused attack changes the classification of a target legitimate message 60% of the time with knowledge of only 30% of the target message's tokens. We also explore two successful

defenses for SpamBayes. The RONI defense filters out dictionary attack messages with complete success. The dynamic threshold defense also mitigates the effect of the dictionary attacks. Focused attacks are especially difficult to defend against because of the attacker's extra knowledge; developing effective defenses in the targeted case is an important open problem.

In the second case-study we consider an adversary that manipulates PCA-based network-wide volume anomaly detection for the purposes of evading detection at test time (an Integrity attack). We study the effects of multiple poisoning strategies while varying the amount of information available to the attacker and the time horizon over which the poisoning occurs. We demonstrate that the PCA-subspace method can be easily compromised (often dramatically) under all of the considered poisoning scenarios. From the attacker's point of view, we illustrate that simple strategies can be effective and conclude that it is not worth the risk or extra amount of work for the attacker to engage in attempts at near-optimal globally-informed strategies. For example, when a locally-informed attacker increases the average volume on a flow's links by 10%, the False Negative Rate (or chance of evasion) is increased by a factor of 7. Moreover, with stealthy poisoning strategies executed over longer time periods, an attacker can increase the FNRs to over 50% with less data than poisoning schemes carried out during a short time window. We demonstrate that our ANTIDOTE counter-measure based on Robust Statistics is robust to these attacks in that it does not allow poisoning attacks to shift the false positive and false negative rates in any significant way. We show that ANTIDOTE provides robustness for nearly all the ingress PoP to egress PoP flows in a backbone network, rejects much of the contaminated data, and continues to operate as a DoS defense even in the face of poisoning by variance injection attacks.

A common theme of the two case-studies on Causative attacks, is the important role of adversarial information and control. In the first case-study on email spam filtering, information corresponds to approximate knowledge of the victim's token distribution and control is parameterized by the fraction of the training corpus poisoned by the attack and the size of the poison spam messages. In the second case-study on network-wide volume anomaly detection, information corresponds to the ability to monitor traffic on one or multiple links, while control is most naturally exerted in the volume of chaff added to the network. Interestingly the forms of information and control that seem most natural to these two domains are very different. By contrast, however, we show that in both studies increased information or increased control result in more effective attacks. We also observe that attack efficacy is not necessarily 'linear' in adversarial capability: *e.g.*, locally-informed poisoning of PCA-based detection at up to moderate levels of control are just as effective as globally-informed poisoning. We return to several of these observations in Chapter 7.

Chapter 3

Querying for Evasion

Beware the wolf in sheep's clothing.
– Aesop

In this chapter we consider attacks on trained classifiers that systematically submit queries to a classifier with the goal of finding an instance that evades detection while being of a near-minimal distance to a target malicious instance. According to the taxonomy of Barreno et al. (2006) discussed in Section 1.2.2, these attacks are Exploratory attacks as they interact with a learned model at test-time; and while the attacks are most naturally applied as Integrity attacks (those that cause False Negatives), they are equally suited to Availability attacks (those that aim for False Positives).

We adopt the abstract theoretical framework of Lowd and Meek (2005b), and extend their results for evading linear classifiers to evading classifiers that partition feature space into two classes, one of which is convex. In addition to the primary goal of finding a distance-minimizing negative instance, we adopt the secondary goal of low query complexity (like Lowd and Meek 2005b). A corollary of our theoretical results is that in general evasion can be significantly easier than reverse engineering the decision boundary which is the approach originally taken by Lowd and Meek (2005b).

The research presented in this chapter was joint work with UCB EECS doctoral candidate Blaine Nelson. During the course of this investigation I contributed the initial query algorithm for the convex positive class case, its lower bound, an initial argument for the L_∞ cost lower bound, and the lower bound's extension to L_p costs. Nelson led the work on improving the convex positive class algorithm, the L_∞ lower bound, the specialized L_2 cost lower bound, and developing the reduction for the convex negative class case.

3.1 Introduction

Machine learning is often used to filter or detect miscreant activities in a variety of applications; *e.g.*, spam, intrusion, virus, and fraud detection. All known detection techniques have blind spots; *i.e.*, classes of miscreant activity that fail to be detected. While learning

allows the detection algorithm to adapt over time, constraints on the learning algorithm also may allow an adversary to programmatically find these vulnerabilities. We consider how an adversary can systematically discover blind spots by querying the learner to find a low cost instance that the detector does not filter. Consider a spammer who wishes to minimally modify a spam message so that it is not classified as a spam and instead reaches a user's inbox unfiltered. By observing the responses of the spam detector, for example in a public webmail service in which he can open accounts and send himself messages, the spammer can search for a successful modification while using few queries.

The evasion problem of finding a low cost negative instance with few queries was first posed by Lowd and Meek (2005b). We continue their line of research by generalizing it to the family of convex-inducing classifiers—classifiers that partition feature space into two sets one of which is convex. Convex-inducing classifiers are a natural family to examine that include linear classifiers, neural networks with a single hidden layer (convex polytopes), one-class classifiers that predict anomalies by thresholding the log-likelihood of a log-concave (or unimodal) density function, the one-class SVM with linear kernel, and quadratic classifiers of the form $\mathbf{x}^\top \mathbf{A}\mathbf{x} + \mathbf{b}^\top \mathbf{x} + c \geq 0$ for semidefinite $\mathbf{A}$. The convex-inducing classifiers also include classifiers whose support is the intersection of a countable number of halfspaces, cones, or balls. We also consider more general L_p costs than the weighted L_1 costs considered by Lowd and Meek (2005b).

We show that evasion does not require reverse engineering the classifier—querying the classifier to learn its decision boundary. The algorithm of Lowd and Meek (2005b) for evading linear classifiers reverse-engineers the classifier's decision boundary but is still efficient. Our algorithms for evading convex-inducing classifiers do not require fully estimating the boundary (which is hard in the general case; see Rademacher and Goyal, 2009) or reverse-engineering the classifier's state. Instead, we directly search for a minimal cost-evading instance. Our algorithms require only polynomial-many queries to achieve $(1+\epsilon)$-multiplicative approximations of cost-optimal negative instances in feature space $\mathbb{R}^D$, with an algorithm for convex positive classes for L_p cost ($p \leq 1$) solving the linear case with $\mathcal{O}\left(\log \frac{1}{\epsilon} + \sqrt{\log \frac{1}{\epsilon}} D\right)$ queries which is fewer than the previously-published reverse-engineering technique. A new lower bound of $\mathcal{O}\left(\log \frac{1}{\epsilon} + D\right)$ shows that this complexity is close to optimal. For $p > 1$ we show that finding evading instances that come very close to having optimal L_p cost requires exponential query complexity.

Our geometric random walk-based approach for evading classifiers with convex negative classes while minimizing L_p costs ($p \geq 1$) has query complexity $\mathcal{O}^*\left(D^5 \log \frac{1}{\epsilon}\right)$. A consequence of polynomial complexity for convex-inducing classifiers is that in general, evasion can be significantly easier than reverse engineering the decision boundary.

Chapter Organization. We conclude this introductory section with a brief summary of related work. Section 3.2 overviews the abstract framework of Lowd and Meek (2005b) upon which we build and covers preliminaries for Section 3.3, which develops and analyzes query algorithms for evading convex-inducing classifiers while minimizing L_1 cost. Section 3.4 considers evasion for minimizing general L_p costs. We conclude the chapter with a summary of our key contributions.

3.1.1 Related Work

Lowd and Meek (2005b) first explored the evasion problem, and developed a method that reverse-engineered linear classifiers. Our approach generalizes their result and improves upon it in three significant ways.

- We consider a more general family of classifiers: the family of convex-inducing classifiers that partition feature space into two sets one of which is convex. This family includes the family of linear classifiers as a special case.
- Our approach does not fully estimate the classifier's decision boundary (which is generally hard Rademacher and Goyal (2009)) or reverse-engineer the classifier's state; instead, we directly search for an instance that the classifier recognizes as negative that is close to the desired attack instance (an evading instance of near-minimal cost).
- Although our algorithms successfully evade a more general family of classifiers, our algorithms still only use a limited number of queries: they require only a number of queries polynomial in the dimension of the instance space. Moreover, our K-step MultiLineSearch Algorithm 5 solves the linear case with fewer queries than the previously-published reverse-engineering technique.

Learning the decision boundary by submitting membership queries requires exponential numbers of queries for general convex-inducing classifiers, since estimating volumes of convex bodies (known to be NP-hard; Dyer and Frieze 1992; Rademacher and Goyal 2009) reduces to learning the boundary. Thus a consequence of our algorithms that evade all convex-inducing classifiers with polynomial complexity, is that evasion is *significantly easier than reverse engineering*.

Dalvi et al. (2004) use a cost-sensitive game theoretic approach to preemptively patch a classifier's blind spots. They construct a modified classifier designed to detect optimally modified instances. This work is complementary to our own; we examine optimal evasion strategies while they have studied mechanisms for adapting the classifier. In this work we assume the classifier is not adapting during evasion.

A number of authors have studied evading sequence-based intrusion detector systems (Tan et al., 2002; Wagner and Soto, 2002). In exploring *mimicry attacks* these authors demonstrated that real IDSs could be fooled by modifying exploits to mimic normal behaviors. These authors used offline analysis of the IDSs to construct their modifications whereas our modifications are optimized by querying the classifier.

Finally, there is an entire field of active learning that also studies a form of query based optimization; *e.g.*, see Schohn and Cohn (2000). While both active learning and near-optimal evasion explore optimal querying strategies, the objectives for these two settings are quite different.

3.2 Background and Definitions

This section is devoted to summarizing the background relevant to this chapter, including the *adversarial classifier reverse engineering (ACRE)* problem introduced by

Lowd and Meek (2005b) which we re-cast as a problem of evasion, and provide preliminary notation and definitions important for describing the main results of the chapter.

We will use email spam as a running example. Section 3.2.1.1 enumerates a partial list of applications of evasion algorithms consistent with this framework.

Example 9. *Consider Exploratory attacks on email spam filtering: a spammer wishes to send a spam email message to a victim's email account that is protected by a state-of-the-art learning-based spam filter. The attacker suspects that the spam message is being blocked by the filter, so he must modify it somehow so that it can evade filtering.*

3.2.1 The Evasion Problem

Let $\mathcal{X} = \mathbb{R}^D$ be the *feature space*; each component of an *instance* $\mathbf{x} \in \mathcal{X}$ is a *feature* denoted by x_d. Let $\boldsymbol{\delta}_d = (0, \ldots, 1, \ldots, 0)$ be the unit vector parallel to the d^{th} coordinate axis (and sitting in the coordinate's positive halfspace).

We consider a family of *classifiers* $\mathcal{F}$ with elements $f \in \mathcal{F}$ mapping $\mathcal{X}$ into the binary *response space* $\mathcal{Y} = \{'-', '+'\}$. Our attacks are designed to operate against a static deterministic classifier: the classifier has already been fit to data or is a hand-crafted decision rule, the adversary does not know the actual mapping *a priori* but does know the family $\mathcal{F}$ from which it came. We define the two sets that partition $\mathcal{X}$ according to the classifier's decision rule as the *positive* and *negative classes* $\mathcal{X}_f^+ = f^{-1}('+')$ and $\mathcal{X}_f^- = f^{-1}('-')$ respectively; and (arbitrarily) identify $\mathcal{X}_f^+$ with a malicious class of instances. When the classifier f can be understood from context (as is often the case since it is fixed) we drop explicit reference to it and denote the classes by $\mathcal{X}^+$ and $\mathcal{X}^-$.

Example 10. *Consider the email spam problem of Example 9. The feature space for email spam filtering is typically vectors in $\{0,1\}^D$ corresponding to a bag-of-words model where each dimension corresponds to a possible token (e.g., words, URLs, etc.). The positive (negative) class corresponds to spam (ham) email messages.*

Assumptions. We assume that the feature space representation is known to the adversary. We assume that the classifier is deterministic and fixed (*e.g.*, is designed manually without learning, is trained offline, or is re-trained only periodically). We make the weak assumption that the adversary has access to instances $\mathbf{x}^- \in \mathcal{X}^-$ and $\mathbf{x}^{\text{A}} \in \mathcal{X}^+$. And finally we assume that the adversary has access to a *membership query oracle* for the true classifier so that $f(\mathbf{x})$ may be observed for any $\mathbf{x} \in \mathcal{X}$: there are no restrictions to which points may be queried by the adversary. While these assumptions may not all hold in all real-world settings, they allow us to consider a worst-case adversary.

Example 11. *Consider the email spam problem of Example 9. If the victim's account is hosted by an* open membership *webmail service such as Yahoo! Mail, Gmail or Hotmail—i.e., accounts may be opened by anyone for free or relatively small cost (for example by solving a CAPTCHA)—then the spammer can gain access to the spam filter's membership oracle by simply opening an account with the webmail service. To query the oracle, the*

spammer need only send himself query messages and observe whether they pass through the filter to the account's inbox or whether they are filtered. Finally spammers can easily find emails $\mathbf{x}^- \in \mathcal{X}^-$ *and* $\mathbf{x}^{\mathrm{A}} \in \mathcal{X}^+$.

Attack Objective. We consider an adversary with special interest in some $\mathbf{x}^{\mathrm{A}} \in \mathcal{X}^+$. In security-sensitive settings this instance typically contains some kind of payload that the attacker wishes to send to a system guarded by the detector. Let $\mathrm{A} : \mathcal{X} \to \mathbb{R}^{0+}$ be a *cost function* of interest to the adversary: we think of the cost as being the distance to the target positive instance $\mathbf{x}^{\mathrm{A}}$ *i.e.*, $\mathrm{A}(\mathbf{x}) = d(\mathbf{x}, \mathbf{x}^{\mathrm{A}})$ to model an adversary who is willing to alter $\mathbf{x}^{\mathrm{A}}$ to evade detection but who is not willing to make too drastic a modification. Thus the objective of our attack is to minimize A over $\mathcal{X}^-$.

We focus on the class of weighted L_p cost functions for $0 < p \leq \infty$

$$\mathrm{A}_p(\mathbf{x}) \quad = \quad \left(\sum_{d=1}^{D} c_d \left| x_d - x_d^{\mathrm{A}} \right|^p \right)^{1/p} \quad , \tag{3.1}$$

where $0 < c_d < \infty$ is the (relative) cost the adversary associates with changes to the d^{th} feature. Unless stated otherwise, we understand the feature costs to be identically one. As with Lowd and Meek (2005b) we focus primarily on weighted L_1 costs in Section 3.3 and explore related L_p costs in Section 3.4. Weighted L_1 costs are particularly appropriate for many adversarial problems since costs are assessed based on the degree to which a feature is altered and the adversary typically is interested in some features more than others. The next example provides a more specific discussion of the cost's relevance in email spam filtering.

Example 12. *Consider again the email spam problem of Example 9. The spammer's original goal was to successfully email a spam message to the victim without the message being filtered. This message contains some payload typically a link (e.g., to an online pharmacy, a drive-by-download site, etc.) or an attachment (e.g., a document infected by a virus). While the payload cannot be altered, the surrounding message that entices the user to activate the payload using social engineering can usually be modified to some extent without degrading the effectiveness of the enticement too much. Additionally, spurious features may be added (e.g., parts of the message that go unrendered). The cost function used should capture the utility of the altered message to the adversary. The* L_1 *cost is particularly appropriate as for the bag-of-words feature model, this cost corresponds to edit distance, a natural metric for passages of text. Having low* L_1 *cost corresponds to a message that is actually similar to the original spam constructed by the spammer.*

Denote by $\mathcal{B}_C$ the closed ball in $\mathcal{X}$ with center $\mathbf{x}^{\mathrm{A}}$ and radius C with respect to the distance corresponding to the given cost. *i.e.*, $\mathcal{B}_C$ is the set of instances with cost at most C. Unless stated otherwise the particular cost should be apparent from the context.

Lowd and Meek (2005b) define *minimal adversarial cost (*MAC*)* of a classifier f to be the scalar

$$\mathrm{MAC}\left(f, \mathrm{A}\right) \quad = \quad \inf_{\mathbf{x} \in \mathcal{X}_f^-} \mathrm{A}\left(\mathbf{x}\right) \quad .$$

They further define an instance to be an ϵ-approximate *instance of minimal adversarial cost (ϵ-IMAC)* if it is a negative instance having cost no more than a factor $(1+\epsilon)$ times the MAC. Overloading notation, we define the set of ϵ-IMACs to be

$$\epsilon\text{-IMAC}\left(f, \mathrm{A}\right) \;=\; \left\{ \mathbf{x} \in \mathcal{X}_f^- \;\middle|\; \mathrm{A}\left(\mathbf{x}\right) \leq (1+\epsilon) \cdot \mathrm{MAC}\left(f, \mathrm{A}\right) \right\} \;. \tag{3.2}$$

The adversary's goal is to find an ϵ-IMAC efficiently, by issuing a relatively small number of queries as measured by ϵ and D. In the email spam setting of the running example, this corresponds to finding a message that will reach the victim's inbox while being as close to a target spam message as possible. We call the overall problem the *Evasion Problem.*

Definition 13. *A family of classifiers $\mathcal{F}$ is* ϵ-IMAC searchable *under a family of cost functions $\mathcal{A}$ if for every $f \in \mathcal{F}$ and $\mathrm{A} \in \mathcal{A}$, there is an algorithm that finds an instance in ϵ-IMAC (f, A) using polynomially-many membership queries in D and $\log(1/\epsilon)$. We will refer to such an algorithm as* efficient*.*

In generalizing the results of Lowd and Meek (2005b) we have made minor alterations to their corresponding definition of the evasion problem.

Remark 14. *Lowd and Meek (2005b) introduced the concept of* adversarial classifier reverse engineering (ACRE) learnability *to quantify the difficulty of finding an ϵ-IMAC instance for particular families of classifiers $\mathcal{F}$ and adversarial costs $\mathcal{A}$. The notion of* ACRE *ϵ-learnable is similar to ϵ-IMAC searchable however there are some noteworthy differences. Our notion of efficiency does not take into account the encoded size of f for simplicity (in the linear case considered by Lowd and Meek 2005b this is simply D); similarly we do not make explicit dependence on the encodings of the known positive and negative instances $\mathbf{x}^{\mathrm{A}}, \mathbf{x}^-$ since these are implicitly included via the dependence on D. ACRE learnability requires knowledge of a third point $\mathbf{x}^+ \in \mathcal{X}^+$. Here we take $\mathbf{x}^+ = \mathbf{x}^A$ making the attacker less covert since it is typically significantly easier to infer the attacker's intentions based on their queries.Finally, we view the original goal of ACRE learnability as being one of evasion and not of reverse engineering (we discuss the related goal of reverse engineering in Section 3.2.2). As a consequence we have re-named the problem to highlight this fact.*

3.2.1.1 Example Applications

In general, algorithms for the Evasion Problem have applications in attacking decision rules in many domains.

Content-Based Email Spam Filtering. As discussed in the series of running examples starting with Example 9, the notions of L_1 cost, access to the filter's membership query oracle for webmail services, and the desire to minimize cost over the negative class while submitting few queries, are all appropriate for evading email spam filtering based on message content.

Web Spam Filtering. In order to game search engine rankings, it is common-place for parties to create spam web pages whose sole purpose is to contain content matching target search queries and to link to the target webpage to be promoted in an effort to increase PageRank-like authority scores (Gyöngyi and Garcia-Molina, 2005). To combat such malicious activities the search company can learn to detect such spam webpages (Drost and Scheffer, 2005). This creates an arms race in which the adversaries are incentivized to evade detection using methods such as those discussed here. Feedback is available to the adversaries via the effects their link farms have on the search engine results, and blacklists of spam pages.

Polymorphic Worm Detection. In order to make detection of malicious packets difficult for defenders, attackers design *polymorphic* worms that mutate their binary while including payload instructions that are required to exploit a specific vulnerability in a system. Learning-based defenses have been designed which can learn (to some extent, see Newsome et al. 2006; Venkataraman et al. 2008) to detect such polymorphic worms (Kim and Karp, 2004). An intelligent polymorphic worm could utilize evasive strategies to modify its code in an attempt to evade detection. Cost corresponds to including as much of the desired payload as possible—higher cost packets could come from including payloads that exploit less severe vulnerabilities; so it is reasonable to model the worm as wanting to minimize cost over the class of packets not filtered by a learned signature. Additionally feedback may be observed by monitoring acknowledgments, acknowledgment timings, and transmissions from successfully infected systems.

Network Anomaly Detection. In the second case-study of Chapter 2 we investigate an application of Principal Components Analysis (PCA) to detecting network-wide volume anomalies. For example, Lakhina et al. (2004a) shows that PCA can be used to detect DoS attacks that cause high volume flows in top-tier networks. In our case-study, we consider Causative attacks on PCA. However the adversary may want to evade detection at test time. Cost may measure the size of the flow initiated by the attacker, or the similarity of the path taken compared to a desired path in the network. Finally, feedback to queries may be observed by monitoring egress links to the destination PoP. Indeed our results apply for the case of PCA in this setting, as the negative set is modeled as the (convex) instances between a pair of parallel hyperplanes.

3.2.1.2 Multiplicative Optimality and Binary Search

The objective function introduced in Equation (3.2) is that of *multiplicative optimality*. The results of this chapter are easily adapted for *additive optimality* in which we seek instances with cost no more than $\eta > 0$ *greater* than the MAC. We will use the notation ϵ-IMAC* and η-IMAC$^+$ to refer to the set in Equation (3.2) and the analogous set

$$\eta\text{-IMAC}^+(f, \mathrm{A}) \;=\; \{\mathbf{x} \in \mathcal{X}_f^- \mid \mathrm{A}(\mathbf{x}) \leq \eta + \mathrm{MAC}(f, \mathrm{A})\} \;.$$

In either the multiplicative or additive case, we can organize the search for a near-optimal instance by iterating over cost bounds on the positive and negative classes using a binary search as follows.

If there is an instance $\mathbf{x} \in \mathcal{X}^-$ with cost C^- and if all instances with cost no more than C^+ are in $\mathcal{X}^+$, then we can conclude that C^- and C^+ bound the MAC *i.e.*, $\mathrm{MAC}(f, \mathrm{A}) \in [C^+, C^-]$. Moreover $\mathbf{x}$ is ϵ-multiplicatively optimal, trivially, if $C_0^-/C_0^+ \leq 1+\epsilon$ and is η-additively optimal if $C_0^- - C_0^+ \leq \eta$. In the sequel, we will consider algorithms that use binary search to iteratively reduce the gap between iterates of C^- and C^+ to achieve additive or multiplicative optimality. In particular, if a new query point with a given cost establishes a new upper or lower bound on MAC, then binary search strategies can reduce the t^{th} gap that is between C_t^- and C_t^+. Given sufficient iterations, optimality will be reached given the following criteria.

Lemma 15. *If an algorithm can provide bounds $C^+ \leq \mathrm{MAC}(f, \mathrm{A}) \leq C^-$, then this algorithm has achieved*

(1) $(C^- - C^+)$-additive optimality; and

(2) $\left(\frac{C^-}{C^+} - 1\right)$-multiplicative optimality.

The measure of performance to be optimized by search algorithms should correspond to the *gap* between the bounds that determines the level of approximation to optimality, as given by this lemma. To achieve additive optimality we define the t^{th} *additive gap* to be $G_t^{(+)} = C_t^- - C_t^+$ with $G_0^{(+)}$ corresponding to initial bounds C_0^- and C_0^+. In the additive setting binary search provides for an optimal worst-case query complexity by using a proposal step of the arithmetic mean $C_t = \left(C_t^- + C_t^+\right)/2$, stopping once $G_t^{(+)} \leq \eta$. The search's query complexity is

$$L_\eta^+ \;=\; \left\lceil \log_2 \left(\frac{G_0^{(+)}}{\eta} \right) \right\rceil \quad . \tag{3.3}$$

Multiplicative optimality can also be achieved via a binary search over the space of exponents as follows. Rewriting the upper and lower bounds as $C^- = 2^a$ and $C^+ = 2^b$, the multiplicative optimality condition becomes an additive condition $a - b \leq \log_2(1+\epsilon)$. Binary search on the exponent achieves ϵ-multiplicative optimality with the fewest queries in the worst-case. The t^{th} *multiplicative gap* is $G_t^{(*)} = C_t^-/C_t^+$; the search uses as a proposal step the geometric mean $C_t = \sqrt{C_t^- \cdot C_t^+}$ and stops once $G_t^{(*)} \leq 1+\epsilon$; query complexity is

$$L_\epsilon^* \;=\; \left\lceil \log_2 \left(\frac{\log_2\left(G_0^{(*)}\right)}{\log_2(1+\epsilon)} \right) \right\rceil \quad . \tag{3.4}$$

The search methods for achieving additive and multiplicative optimality are intrinsically related, however there are two key differences which we now detail. First, multiplicative optimality is well-defined only when $C_0^+ > 0$ whereas additive optimality is possible for

$C_0^+ = 0$. In this special case, $\mathbf{x}^A$ is on the boundary of $\mathcal{X}^+$ and so there can be no ϵ-IMAC* for any $\epsilon > 0$. This pathological case is a minor technical issue—we demonstrate in Section 3.3.1.4 an algorithm that efficiently establishes a non-trivial lower bound C_0^+ if such a bound exists. Second, and more importantly, the additive optimality criterion is not *scale invariant*, whereas multiplicative optimality is. An immediate consequence of this fact is that *the units of the cost determine whether a particular level of additive accuracy can be achieved whereas multiplicative costs are unitless.* While multiplicative optimality has the desirable property of scale-invariance, it does not have the *shift invariance* possessed by additive optimality. We view scale invariance as being more important than shift invariance, since if the cost function is scale invariant (as is the case for metric-based costs including the L_p family) then optimality is invariant to rescaling of the feature space.

For the remainder of this chapter, we focus on establishing ϵ-multiplicative optimality for an ϵ-IMAC (except where explicitly noted) and define $L_\epsilon = L_\epsilon^*$ and $G_t = G_t^{(*)}$. Finally, we relate query complexity in terms of L_ϵ^*, which will be convenient to reason about in the sequel, to complexity in terms of ϵ which is the stated goal of ϵ-IMAC searchability.

Remark 16. *Notice that for sufficiently small ϵ, binary search's L_ϵ^* as displayed in Equation* (3.4) *is $\Theta\left(\log \frac{1}{\epsilon}\right)$ since $\log(1+\epsilon) \approx \epsilon$. Thus demanding query complexity that is polynomial in $\log \frac{1}{\epsilon}$ is equivalent to complexity that is polynomial in L_ϵ^*.*

3.2.2 The Reverse Engineering Problem

As stated in Remark 14, Lowd and Meek (2005b) term the evasion problem "adversarial classifier reverse engineering (ACRE) " learnability. While the requirement of ACRE learnability is actually to evade a classifier, their approach for linear classifiers is to learn the decision boundary. It is this task of learning the classifier's decision boundary that we refer to here as the *reverse engineering problem.*[1] And while not identical problems, this notion of reverse engineering is certainly related to the goal of *active learning* (Schohn and Cohn, 2000).

Efficient query-based reverse engineering of an $f \in \mathcal{F}$ is clearly sufficient for minimizing A over the estimated negative space: once the decision boundary has been determined, an offline optimization of the cost function (without submitting further queries) yields an ϵ-IMAC. However, reverse engineering is in general a query-expensive task (since it relates to approximating volumes which is hard for even convex bodies, Dyer and Frieze 1992), while finding an ϵ-IMAC need not be: the requirements for finding an ϵ-IMAC differ significantly from the objectives of reverse engineering. To reverse engineer, the attacker must approximate the decision boundary globally; to evade, the attacker need only locally approximate the decision boundary in the neighborhood of a constrained cost-optimizer.

In particular our algorithms construct queries to provably find an ϵ-IMAC without reverse engineering the classifier. A corollary of our results are that reverse engineering is indeed

[1] Our use of 'reverse engineering' corresponds to deriving insight into the underlying state of the learner. Here we take that to mean the *effective* state of a classifier which is its decision boundary. However it could also apply to attacks that aim to determine the classifier's model parameters that *implicitly* define the decision boundary, in the case of a known learning algorithm with known parametrization.

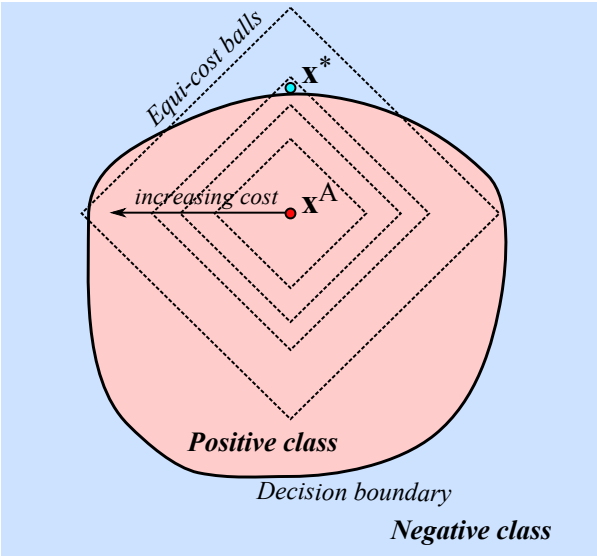


Figure 3.1: Evading a classifier with a convex positive class to optimize an L_1 cost involves finding the vertex of the ball that first pierces the negative class.

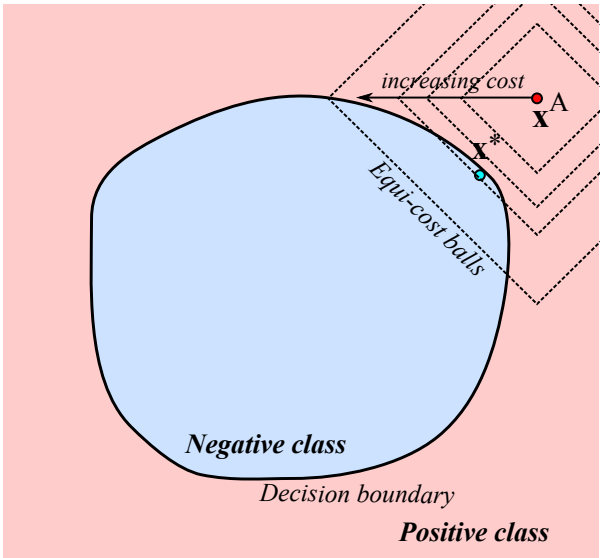


Figure 3.2: Evading a classifier with a convex negative class to optimize an L_1 cost is harder than the convex positive case since the optimum may not be a vertex of the ball.

significantly more complex than evasion for the rather general case of the positive or negative class being convex.

3.3 Evasion while Minimizing L_1-distance

This section develops algorithms for achieving ϵ-IMAC searchability for the L_1 cost function and the family of *convex-inducing classifiers* $\mathcal{F}^{\text{convex}}$ that partition feature space into a positive class and a negative class, one of which is convex. As discussed above, the L_1 cost is natural for tasks such as email spam filtering, and was the focus of Lowd and Meek (2005b) in their original work on evading linear classifiers. The convex-inducing classifiers include the class of linear classifiers, neural networks with a single hidden layer (convex polytopes), one-class classifiers that predict anomalies by thresholding the log-likelihood of a log-concave (or uni-modal) density function, the one-class SVM with linear kernel, and quadratic classifiers of the form $\mathbf{x}^\top \mathbf{A}\mathbf{x} + \mathbf{b}^\top \mathbf{x} + c \geq 0$ for semidefinite $\mathbf{A}$. The convex-inducing classifiers also include classifiers whose support is the intersection of a countable number of halfspaces, cones, or balls.

Restricting $\mathcal{F}$ to be the family of convex-inducing classifiers considerably simplifies the general ϵ-IMAC search problem, as depicted in Figures 3.1 and 3.2. When the negative class $\mathcal{X}^-$ is convex (considered in Section 3.3.2), the problem reduces to minimizing a convex function A constrained to a convex set; if $\mathcal{X}^-$ *were* known to the adversary then evasion would reduce to a convex program. The key challenge is that the adversary only has access

to a membership query oracle for the classifier. When the positive class $\mathcal{X}^+$ is convex (considered in Section 3.3.1), our task is to minimize the convex function A outside of a convex set; this is generally a hard problem however for certain cost functions including L_1 cost, it is easy to determine whether a cost ball is completely contained within a convex set, leading to efficient approximation algorithms.

Surprisingly there is an asymmetry depending on whether the positive or negative class is convex. When the positive set is convex, determining whether an L_1 ball $\mathcal{B}_C \subset \mathcal{X}^+$ only requires querying the vertices of the ball, of which there are $2D$. When the negative class is convex, however, determining whether or not $\mathcal{B}_C \cap \mathcal{X}^- = \emptyset$ is non-trivial since the intersection need not occur at a vertex of the ball. We present a very efficient algorithm for the optimizing the L_1 cost when $\mathcal{X}^+$ is convex and a polynomial random algorithm for optimizing *any convex cost* when $\mathcal{X}^-$ is convex.

Our algorithms achieve multiplicative optimality via binary search. We use $C_0^- = \mathrm{A}(\mathbf{x}^-)$ as an initial upper bound on the MAC and for technical reasons (*cf.* Section 3.2.1.2) assume there is some $C_0^+ > 0$ that lower bounds the MAC (*i.e.*, $\mathbf{x}^{\mathrm{A}}$ is in the interior of $\mathcal{X}^+$).

3.3.1 Convex Positive Classes

Solving the ϵ-IMAC Search problem when $\mathcal{X}^+$ is generally hard, however we will demonstrate in this section that for the (weighted) L_1 cost binary search algorithms render very efficient solutions with almost-matching lower bounds on query complexity. We now explain how we exploit the properties of the (weighted) L_1 ball together with the convexity of $\mathcal{X}^+$ to efficiently determine whether $\mathcal{B}_C \subset \mathcal{X}^+$ for any C. We also discuss practical aspects of our algorithm and extensions to other L_p cost functions.

The existence of an efficient query algorithm relies on three facts: (1) $\mathbf{x}^{\mathrm{A}} \in \mathcal{X}^+$; (2) every weighted L_1 cost C-ball centered at $\mathbf{x}^{\mathrm{A}}$ intersects with $\mathcal{X}^-$ only if at least one of the ball's vertices is in $\mathcal{X}^-$; and (3) C-balls of weighted L_1 costs only have $2 \cdot D$ vertices. The vertices of the weighted L_1 ball differ from $\mathbf{x}^{\mathrm{A}}$ in exactly one feature d,

$$\mathbf{x}^{\mathrm{A}} \quad \pm \quad \frac{C}{c_d}\boldsymbol{\delta}_d \ . \tag{3.5}$$

We now formalize the second fact, which follows immediately from the observation that the L_1 ball is a (convex) polytope.

Lemma 17. *For all $C > 0$, if there exists some $\mathbf{x} \in \mathcal{X}^-$ of cost $C = \mathrm{A}_{\mathbf{c}}(\mathbf{x})$, then there is a vertex of the L_1 C-cost ball in $\mathcal{X}^-$ that trivially also achieves cost C.*

As a consequence of this observation, if all vertices of a ball $\mathcal{B}_C$ are positive, then all $\mathbf{x}$ with $\mathrm{A}\mathbf{x} \leq C$ are positive thus establishing C as a new lower bound on the MAC. Conversely, if any vertex of $\mathcal{B}_C$ is negative, then C becomes a new upper bound on MAC. Thus, by simultaneously querying all $2 \cdot D$ equi-cost vertices of $\mathcal{B}_C$, we establish C as a new lower or upper bound. By performing a binary search on C, using the geometric mean proposal step of $C_t = \sqrt{C_t^+ \cdot C_t^-}$, we iteratively halve the multiplicative gap between our bounds until it is within a factor of $1 + \epsilon$ yielding an ϵ-IMAC of the form of Equation (3.5).

Algorithm 3 Multi-line Search

MLS $\left(\mathcal{W}, \mathbf{x}^{\mathrm{A}}, \mathbf{x}^{-}, C_0^{+}, C_0^{-}, \epsilon\right)$
$\mathbf{x}^{\star} \leftarrow \mathbf{x}^{-}$
$t \leftarrow 0$
while $C_t^{-}/C_t^{+} > 1+\epsilon$ **do**
  $C_t \leftarrow \sqrt{C_t^{+} * C_t^{-}}$
  for all $\mathbf{e} \in \mathcal{W}$ **do**
    Query classifier: $f_{\mathbf{e}}^{t} \leftarrow f\left(\mathbf{x}^{\mathrm{A}} + C_t\mathbf{e}\right)$
    if $f_{\mathbf{e}}^{t} =$ '$-$' **then**
      $\mathbf{x}^{\star} \leftarrow \mathbf{x}^{\mathrm{A}} + C_t\mathbf{e}$
      For each $\mathbf{i} \in \mathcal{W}$: If $f_{\mathbf{i}}^{t} =$ '$+$' then prune $\mathbf{i}$ from $\mathcal{W}$
      Lazy Querying: break for-loop
    end if
  end for
  $C_{t+1}^{+} \leftarrow C_t^{+}$ and $C_{t+1}^{-} \leftarrow C_t^{-}$
  if $\forall \mathbf{e} \in \mathcal{W}\ f_{\mathbf{e}}^{t} =$ '$+$' **then** $C_{t+1}^{+} \leftarrow C_t$
  else $C_{t+1}^{-} \leftarrow C_t$
  $t \leftarrow t+1$
end while
return $\mathbf{x}^{\star}$

A general form of this MultiLineSearch procedure is presented as Algorithm 3 which searches along all unit-cost search directions in the set $\mathcal{W}$. The set $\mathcal{W}$ represents search directions that radiate from the ball's origin at $\mathbf{x}^{\mathrm{A}}$, together span the ball, and have unit cost. At each step, MultiLineSearch issues at most $|\mathcal{W}|$ queries to construct a bounding shell (*i.e.*, the convex hull of these queries will either form an upper or lower bound on the MAC) to determine whether $\mathcal{B}_C \subset \mathcal{X}^{+}$. Once a negative instance is found at cost C, we cease further queries at cost C since a single negative instance is sufficient to establish a upper bound. We call this policy *lazy querying*. Further, when an upper bound is established for a cost C (a negative vertex is found), our algorithm prunes all directions that were positive at cost C. This pruning is sound because by convexity any such direction is positive for all costs less than C and further C is a now an upper bound on the MAC so all further queries will be at costs less than C. Finally, by performing a binary search on the cost, MultiLineSearch finds an ϵ-IMAC with no more than $|\mathcal{W}| \cdot L_\epsilon$ queries but at least $|\mathcal{W}| + L_\epsilon$ queries. Thus, this algorithm is $\mathcal{O}\left(|\mathcal{W}| \cdot L_\epsilon\right)$.

Algorithm 4 uses MultiLineSearch for (weighted) L_1 costs, setting $\mathcal{W}$ to be the vertices of the unit-cost L_1 ball which is centered at $\mathbf{x}^{\mathrm{A}}$. In this case, the search issues at most $2 \cdot D$ queries to determine whether each $\mathcal{B}_C \subset \mathcal{X}^{+}$ and Algorithm 4 has an exceptionally efficient query complexity of $\mathcal{O}\left(L_\epsilon \cdot D\right)$.

3.3.1.1 K-step Multi-Line Search

We now develop a variant of the multi-line search algorithm that better exploits pruning

Algorithm 4 Convex $\mathcal{X}^+$ Set Search

ConvexSearch $\left(\mathcal{W}, \mathbf{x}^{\mathrm{A}}, \mathbf{x}^-, \epsilon, C^+\right)$
$C^- \leftarrow \mathrm{A}\left(\mathbf{x}^-\right)$
$\mathcal{W} \leftarrow \left\{ -\frac{1}{c_i} \cdot \boldsymbol{\delta}_i, \frac{1}{c_i} \cdot \boldsymbol{\delta}_i \;\middle|\; i \in [D] \right\}$
return: MLS $\left(\mathcal{W}, \mathbf{x}^{\mathrm{A}}, \mathbf{x}^-, C^+, C^-, \epsilon\right)$

Algorithm 5 K-Step Multi-line Search

KMLS $\left(\mathcal{W}, \mathbf{x}^{\mathrm{A}}, \mathbf{x}^-, C_0^+, C_0^-, \epsilon, K\right)$
$\mathbf{x}^\star \leftarrow \mathbf{x}^-$
$t \leftarrow 0$
while $C_t^- / C_t^+ > 1 + \epsilon$ **do**
  Choose a direction $\mathbf{e} \in \mathcal{W}$
  $B^+ \leftarrow C_t^+$ and $B^- \leftarrow C_t^-$
  for K steps **do**
    $B \leftarrow \sqrt{B^+ \cdot B^-}$
    Query classifier: $f_{\mathbf{e}} \leftarrow f\left(\mathbf{x}^{\mathrm{A}} + B\mathbf{e}\right)$
    if $f_{\mathbf{e}} =$ '+' **then** $B^+ \leftarrow B$
    else $B^- \leftarrow B$ **and** $\mathbf{x}^\star \leftarrow \mathbf{x}^{\mathrm{A}} + B\mathbf{e}$
  end for
  for all $\mathbf{i} \in \mathcal{W} \backslash \{\mathbf{e}\}$ **do**
    Query classifier: $f_{\mathbf{i}}^t \leftarrow f\left(\mathbf{x}^{\mathrm{A}} + B^+\mathbf{i}\right)$
    if $f_{\mathbf{i}}^t =$ '$-$' **then**
      $\mathbf{x}^\star \leftarrow \mathbf{x}^{\mathrm{A}} + (B^+)\mathbf{i}$
      For each $\mathbf{k} \in \mathcal{W}$: If $f_{\mathbf{k}}^t =$ '+' then prune $\mathbf{k}$ from $\mathcal{W}$
      Lazy Querying: break for-loop
    end if
  end for
  $C_{t+1}^- \leftarrow B^-$
  if $\forall \mathbf{i} \in \mathcal{W}\ f_{\mathbf{i}}^t =$ '+' **then** $C_{t+1}^+ \leftarrow B^+$
  else $C_{t+1}^- \leftarrow B^+$
  $t \leftarrow t + 1$
end while
return $\mathbf{x}^\star$

to further reduce the (already low) query complexity of Algorithm 3—we call this variant K-step MultiLineSearch. The original MultiLineSearch algorithm makes $2 \cdot |\mathcal{W}|$ simultaneous binary searches. Instead of this breadth-first search we could search sequentially depth-first, and still obtain a best case of $\Omega\left(D + L_\epsilon\right)$ and worst case of $\mathcal{O}\left(L_\epsilon \cdot D\right)$ but for exactly the opposite convex bodies. For the parallel search variant described above, the best case is an elongated ball and the worst case is a rounded ball while for a sequential binary search variant these cases are reversed. We therefore propose an algorithm that mixes these

strategies. We parametrize the mixture via a parameter K to be set later.

At each phase, the K-step MultiLineSearch (displayed as Algorithm 5) chooses a single direction $\mathbf{e}$ and queries it for K steps to generate candidate bounds B^- and B^+ on the MAC. The algorithm makes substantial progress towards reducing G_t without querying other directions, since it is a depth-first procedure. In a breadth-first step, the algorithm then iteratively queries all remaining directions at the candidate lower bound B^+. Again we use lazy querying and stop as soon as a negative instance is found since B^+ is then no longer a viable lower bound. In this case, although the candidate bound is invalidated, we can still prune all directions that were positive at B^+ including direction $\mathbf{e}$. Thus, in every iteration, either the gap is decreased or at least one search direction is pruned. We now show that for $K = \lceil\sqrt{L_\epsilon}\rceil$, the algorithm achieves a delicate balance between breadth-first and depth-first approaches to attain a better worst case complexity than following either approach alone.

Theorem 18. *Algorithm 5, run with $K = \lceil\sqrt{L_\epsilon}\rceil$, will find an ϵ-IMAC by submitting at most $\mathcal{O}\left(L_\epsilon + \sqrt{L_\epsilon}|\mathcal{W}|\right)$ queries.*

Proof. We consider a defender that is choosing the classifier (and hence the oracle's responses to the attacker's queries) adaptively to force a large number of queries on the attacker. Our goal is to bound the worst-case number of queries.

During the K steps of binary search, regardless of how the defender responds, the candidate gap along $\mathbf{e}$ will shrink by an exponent of 2^{-K}; *i.e.*,

$$\frac{B^-}{B^+} \;=\; \left(\frac{C_0^-}{C_0^+}\right)^{2^{-K}} \quad . \tag{3.6}$$

The primary decision for the defender occurs when the adversary begins querying directions other than $\mathbf{e}$. At iteration t, the defender has two options:

Case 1 ($t \in \mathcal{C}_1$): Respond with '+' for all remaining directions. Here the bounds B^+ and B^- are verified and thus the gap is reduced by an exponent of 2^{-K}.

Case 2 ($t \in \mathcal{C}_2$): Choose at least one direction to respond with '−'. Here the defender can make the gap decrease by a negligible amount but also must choose some number $E_t \geq 1$ of eliminated directions.

By conservatively assuming the gap only decreases in case 1, *i.e.*, if $t \in \mathcal{C}_1$ we $G_t = G_{t-1}^{2^{-K}}$ or otherwise $G_t = G_{t-1}$, the total number of queries is bounded regardless of the order in which the cases are applied,

$$|\mathcal{C}_1| \;\leq\; \left\lceil \tfrac{L_\epsilon}{K} \right\rceil \quad , \tag{3.7}$$

since we need a total of L_ϵ binary search steps and each case 1 iteration is responsible for K of them.

Every case 1 iteration makes exactly $K + |\mathcal{W}_t| - 1$ queries. The size of $\mathcal{W}_t$ (depending on when lazy-querying activates) is controlled by the defender, but we can bound it by $|\mathcal{W}|$. This and Equation (3.7) bound the number of queries used in case 1 by

$$\begin{aligned} Q_1 &= \sum_{t \in C_1} (K + |\mathcal{W}_t| - 1) \\ &\leq L_\epsilon + K + \left\lceil \tfrac{L_\epsilon}{K} \right\rceil \cdot (|\mathcal{W}| - 1) \quad . \end{aligned}$$

Each case 2 iteration uses exactly $K + E_t$ queries and eliminates $E_t \geq 1$ directions. Since a case 2 iteration eliminates at least 1 direction, $|\mathcal{C}_2| \leq |\mathcal{W}| - 1$ and moreover, $\sum_{t \in \mathcal{C}_2} E_t \leq |\mathcal{W}| - 1$ since each direction can only be eliminated once. Thus the number of queries due to case 2 is bounded by

$$\begin{aligned} Q_2 &= \sum_{t \in \mathcal{C}_2} (K + E_t) \\ &\leq (|\mathcal{W}| - 1)(K + 1) \quad , \end{aligned}$$

and so the total queries used by Algorithm 5 is

$$\begin{aligned} Q &= Q_1 + Q_2 \\ &< L_\epsilon + \left(\left\lceil \tfrac{L_\epsilon}{K} \right\rceil + K + 1 \right) |\mathcal{W}| \quad , \end{aligned}$$

which is minimized by $K = \lceil \sqrt{L_\epsilon} \rceil$. Substituting this for K and using $L_\epsilon / \lceil \sqrt{L_\epsilon} \rceil \leq \sqrt{L_\epsilon}$ we have

$$Q < L_\epsilon + (2\lceil \sqrt{L_\epsilon} \rceil + 1)|\mathcal{W}| \quad .$$

proving that $Q = \mathcal{O}\left(L + \sqrt{L}|\mathcal{W}|\right)$. □

As a consequence of this result, finding an ϵ-IMAC with Algorithm 5 for a (weighted) L_1 cost requires only $\mathcal{O}\left(L_\epsilon + \sqrt{L_\epsilon} D\right)$ queries in the worst-case. In particular, Algorithm 4 can incorporate K-step MultiLineSearch directly by replacing its function call to MLS to a call to KLMS and setting $K = \lceil \sqrt{L_\epsilon} \rceil$.

3.3.1.2 Evading Linear Classifiers

Lowd and Meek (2005b) originally developed a method for reverse engineering linear classifiers for a (weighted) L_1 cost. First their method isolates a sequence of points from $\mathbf{x}^-$ to $\mathbf{x}^{\mathrm{A}}$ that cross the classifier's boundary and then it estimates the hyperplane's parameters using D line searches. Their algorithm has complexity $\mathcal{O}(D \cdot L_\epsilon)$. As a consequence of our new ability to efficiently minimize the L_1 objective for any convex $\mathcal{X}^+$, we gain an alternative method for evading linear classifiers. Because linear classifiers are a special case of convex-inducing classifiers, our K-step MultiLineSearch algorithm improves slightly on the reverse-engineering technique's query complexity and applies to a much larger class of classifiers.

3.3.1.3 Lower Bounds

The following result establish a lower bound on the number of queries required by any algorithm to find an ϵ-IMAC when $\mathcal{X}^+$ is convex for a (weighted) L_1 cost. We present the result for the case of multiplicative optimality, incorporating a lower bound $r > 0$ on the MAC for technical reasons, however the same essential argument yields the same lower bound of $\max\{D, L_\eta^+\}$ for algorithms achieving η-additive optimality.

Theorem 19. *Consider any $D \in \mathbb{N}$, $\mathbf{x}^{\mathrm{A}} \in \mathcal{X} = \mathbb{R}^D$, $\mathbf{x}^- \in \mathcal{X}$, $0 < r < R = \mathrm{A}(\mathbf{x}^-)$ and $\epsilon \in \left(0, \frac{R}{r} - 1\right)$. For all query algorithms submitting $N < \max\{D, L_\epsilon\}$ queries, there exist two classifiers inducing convex positive classes in $\mathcal{X}$ such that*

1. *Both positive classes properly contain $\mathcal{B}_r$;*
2. *Neither positive class contains $\mathbf{x}^-$;*
3. *The classifiers return the same responses on the algorithm's N queries; and*
4. *The classifiers have no common ϵ-IMAC.*

That is, in the worst-case all query algorithms for convex positive classes must submit at least $\max\{D, L_\epsilon\}$ membership queries in order to be multiplicative ϵ-optimal.

Proof. Suppose some query-based algorithm submits N membership queries $\mathbf{x}^1, \ldots, \mathbf{x}^N$ to the classifier. For the algorithm to be ϵ-optimal, these queries must constrain all consistent positive convex sets to have a common point among their ϵ-IMAC sets.

First consider the case that $N \geq L_\epsilon$. By assumption, then, $N < D$. Suppose classifier f responds as

$$f(\mathbf{x}) = \begin{cases} +1\,, & \text{if } \mathrm{A}(\mathbf{x}) < R \\ -1\,, & \text{otherwise} \end{cases}.$$

For this classifier, $\mathcal{X}^+$ is convex, $\mathcal{B}_r \subset \mathcal{X}^+$, and $\mathbf{x}^- \notin \mathcal{X}^+$. Moreover, since $\mathcal{X}^+$ is the open ball of cost R, $\mathrm{MAC}(f, \mathrm{A}) = R$.

Consider an alternative classifier g that responds identically to f for $\mathbf{x}^1, \ldots, \mathbf{x}^N$ but has a different convex positive set $\mathcal{X}_g^+$. Without loss of generality, suppose that the first $M \leq N$ query responses are positive and the remaining are negative. Let $\mathcal{G} = \mathrm{conv}\left(\mathbf{x}^1, \ldots, \mathbf{x}^M\right)$ the convex hull of the M positive queries. Now let $\mathcal{X}_g^+$ be the convex hull of the union of $\mathcal{G}$ and the r-ball around $\mathbf{x}^{\mathrm{A}}$ *i.e.*, $\mathcal{X}_g^+ = \mathrm{conv}(\mathcal{G} \cup \mathcal{B}_r)$. Since $\mathcal{G}$ contains all positive queries and $r < R$, the convex set $\mathcal{X}_g^+$ is consistent with the responses from f, $\mathcal{B}_r \subset \mathcal{X}^+$, and $\mathbf{x}^- \notin \mathcal{X}^+$. Moreover since $M \leq N < D$, $\mathcal{G}$ is contained in a proper subspace of $\mathcal{X}$ whereas $\mathcal{B}_r$ is not. Hence, $\mathrm{MAC}(g, \mathrm{A}) = r$. Since the accuracy ϵ is less than $\frac{R}{r} - 1$, any ϵ-IMAC of g must have cost less than R whereas any ϵ-IMAC of f must have cost greater than or equal to R. Thus we have constructed two convex-inducing classifiers f and g with consistent query responses but with no common ϵ-IMAC.

Now consider the case that $N < L_\epsilon$. First, recall our definitions: $C_0^- = R$ is the initial upper bound on the MAC, $C_0^+ = r$ is the initial lower bound on the MAC, and $G_t = C_t^- / C_t^+$

is the gap between the upper bound and lower bound at iteration t. Here the defender f responds with

$$f\left(\mathbf{x}^t\right) = \begin{cases} +1\,, & \text{if } \mathrm{A}\left(\mathbf{x}^t\right) \leq \sqrt{C_{t-1}^- \cdot C_{t-1}^+} \\ -1\,, & \text{otherwise} \end{cases}.$$

This strategy ensures that at each iteration $G_t \geq \sqrt{G_{t-1}}$ and since the algorithm can not terminate until $G_N \leq 1+\epsilon$, we have $N \geq L_\epsilon$ from Equation (3.4). As in the $N \geq L_\epsilon$ case we have constructed two convex-inducing classifiers with consistent query responses but with no common ϵ-IMAC. The first classifier's positive set is the smallest cost-ball enclosing all positive queries, while the second classifier's positive set is the largest cost-ball enclosing all positive queries but no negatives. The MAC values of these sets differ by more than a factor of $(1+\epsilon)$ if $N < L_\epsilon$ so they have no common ϵ-IMAC. □

Remark 20. *For the additive and multiplicative cases we restrict η and ϵ to the intervals $(0, \mathrm{A}(\mathbf{x}^-))$ and $(0, \mathrm{A}(\mathbf{x}^-)/r)$ respectively. In fact, outside of these intervals the query strategies are trivial. For either $\eta = 0$ or $\epsilon = 0$ no approximation algorithm will terminate. Similarly, for $\eta \geq \mathrm{A}(\mathbf{x}^-)$ or $\epsilon \geq \frac{R}{r} + 1$, the instance $\mathbf{x}^-$ is a near-optimal instance itself so no queries are required.*

Theorem 19 and the analogous additive result show that η-additive and ϵ-multiplicative optimality require $\Omega\left(L_\eta^+ + D\right)$ and $\Omega\left(L_\epsilon^* + D\right)$ queries respectively. Thus, we see that our K-step MultiLineSearch algorithm (*cf.* Algorithm 5) has almost optimal query complexity with $\mathcal{O}\left(L_\epsilon + \sqrt{L_\epsilon} D\right)$ queries for weighted L_1 costs.

3.3.1.4 Generalizations

We now consider two relaxations that require minor modifications to Algorithms 3 and 5, primarily as simple preprocessing steps.

No Initial Lower Bound. To find an ϵ-IMAC our basic algorithms search between initial bounds C_0^+ and C_0^-, but in general C_0^+ may not be known to a real-world adversary. Algorithm 6 SpiralSearch can efficiently establish a lower bound on the MAC if one exists. The basic idea of the algorithm is to perform a guess-then-halve search[2] on the exponent, starting from the upper bound. The algorithm also eliminates any direction that exceeds the current upper bound.

At the t^{th} iteration of SpiralSearch a direction is selected and queried at the current lower bound of $2^{-2^t} C_0^-$. If the query's response is positive, that direction is added to the set $\mathcal{V}$ of directions consistent with the lower bound. Otherwise, all directions in $\mathcal{V}$ are discarded and the lower bound is lowered with an exponentially decreasing exponent. Thus, given that some lower bound $C_0^+ > 0$ does exist, one will be found relatively quickly in $\mathcal{O}\left(L_\epsilon + D\right)$ queries, for $\mathcal{W}$ equal to the $2 \cdot D$ directions of the coordinate axes and $\epsilon = 1$ (corresponding to no dependence on ϵ, which is not a parameter of the search).

[2]The inverse of more well-known guess-then-double algorithms which are used for example, for dynamically allocating arrays.

Algorithm 6 Spiral Search

spiral $\left(\mathcal{W}, \mathbf{x}^{\mathrm{A}}, \mathbf{x}^{-}, C_0^{-}\right)$
$t \leftarrow 0$ and $\mathcal{V} \leftarrow \emptyset$
repeat
  Choose a direction $\mathbf{e} \in \mathcal{W}$
  Query classifier: $f_{\mathbf{e}} \leftarrow f\left(\mathbf{x}^{\mathrm{A}} + 2^{-2^t} C_0^{-} \mathbf{e}\right)$
  if $f_{\mathbf{e}} = \text{'}-\text{'}$ **then**
    $t \leftarrow t + 1$
    $\mathcal{V} \leftarrow \emptyset$
  else
    $\mathcal{W} \leftarrow \mathcal{W} \backslash \{\mathbf{e}\}$
    $\mathcal{V} \leftarrow \mathcal{V} \cup \{\mathbf{e}\}$
  end if
until $\mathcal{W} = \emptyset$
$B^{+} \leftarrow 2^{-2^t} C_0^{-}$
return $(\mathcal{V}, B^{+}, C_0^{-})$

Proposition 21. *For any classifier with convex positive class $\mathcal{X}^{+}$, $\mathbf{x}^{\mathrm{A}} \in \mathcal{X}^{+}$, $\mathbf{x}^{-} \notin \mathcal{X}^{+}$, upper bound $C_0^{-} > 0$, greatest lower bound $0 < C_0^{+} < C_0^{-}$, and set of covering directions $\mathcal{W}$, Algorithm 6 will find a valid lower bound $B^{+} \leq C_0^{+}$ in at most $|\mathcal{W}| + \log_2 \log_2(C_0^{-}/C_0^{+})$ queries.*

Proof. Every query submitted by the algorithm results in either the pruning of a direction from $\mathcal{W}$ or the halving of the lower bound in exponent space. There are at most $|\mathcal{W}|$ events of the first kind. To analyze the maximum number of steps of the second kind consider that we stop when $2^{-2^t} B^{+} \leq C_0^{+}$. This is equivalent to $t \geq \log_2 \log_2(C_0^{-}/C_0^{+})$. Combining these query count bounds yields the result. □

Thus this algorithm can be used as a precursor to any of the previous searches[3] and can be adapted for additive optimality by halving the lower bound instead of the exponent. Furthermore, the search directions pruned by SpiralSearch are also invalid for the subsequent MultilineSearch, so the set $\mathcal{V}$ returned by SpiralSearch can be used as the set $\mathcal{W}$ for the subsequent search, amortizing some of the multiline search's effort.

No Initial Negative Example. Our algorithms can also naturally be adapted to the case when the adversary has no negative example $\mathbf{x}^{-}$. This is accomplished by an inverse process of the previous no lower bound generalization through a guess-than-double type process of querying L_1 balls of doubly exponentially increasing radii until a negative instance is found. During the t^{th} iteration, we probe along the f^{th} search direction at a cost $2^{2^t} C_0^{+}$ until a negative example is found. Once we've obtained a negative example (having probed for T iterations), we must have $2^{2^{T-1}} < \mathrm{MAC}(f, \mathrm{A}) \leq 2^{2^T}$. Thus we can now perform our

[3]In the pathological case of no existing lower bound, this algorithm would not terminate. In practice, the search should be terminated after sufficiently many iterations.

Algorithm 7 Intersect Search

IntersectSearch $(\mathcal{P}^0, \mathcal{Q} = \{\mathbf{x}^j \in \mathcal{P}^0\}, C)$
for all $s = 1 \dots T$ **do**
  Generate $2N$ samples $\{\mathbf{x}^j\}_{j=1}^{2N}$
    Choose $\mathbf{x}^j$ from $\mathcal{Q}$
    $\mathbf{x}^j \leftarrow \text{HitRun}\,(\mathcal{P}^{s-1}, \mathcal{Q}, \mathbf{x}^j)$
  If $\exists \mathbf{x}^j$, $\text{A}\,(\mathbf{x}^j) \leq C$, then terminate the for-loop
  Put samples into 2 sets of size N
    $\mathcal{R} \leftarrow \{\mathbf{x}^j\}_{j=1}^{N}$ and $\mathcal{S} \leftarrow \{\mathbf{x}^j\}_{j=N+1}^{2N}$
  $\mathbf{z}^s \leftarrow \frac{1}{N} \sum_{\mathbf{x}^j \in \mathcal{R}} \mathbf{x}^j$
  Compute $\mathcal{H}_{\mathbf{z}^s}$ using Equation (3.9)
  $\mathcal{P}^s \leftarrow \mathcal{P}^{s-1} \cap \mathcal{H}_{\mathbf{z}^s}$
  Keep samples in $\mathcal{P}^s$
    $\mathcal{Q} \leftarrow \{\mathbf{x} \in \mathcal{S} \cap \mathcal{P}^s\}$
end for
Return: the discovered witness $[\mathbf{x}_j, \mathcal{P}^s, \mathcal{Q}]$; or 'No Intersect'

multi-line search with $C_0^+ = 2^{2^{K-1}}$ and $C_0^- = 2^{2^K}$. This precursor step requires at most $2 \cdot D \left\lceil \log_2 \log_2 \text{MAC}\,(f, \text{A}) \right\rceil$ to prepare the MultilineSearch algorithm.

3.3.2 Convex Negative Classes

In this section we consider minimizing a weighted L_1 cost A (*cf.* Equation 3.1) when the feasible set $\mathcal{X}^-$ is convex. Although any convex function can be efficiently minimized within a known convex set (*e.g.*, using the Ellipsoid Method and Interior Point methods, Boyd and Vandenberghe 2004), in the context of the evasion problem the convex set is accessible only via membership queries. We use a randomized polynomial algorithm due to Bertsimas and Vempala (2004) to minimize the cost function A given an initial point $\mathbf{x}^- \in \mathcal{X}^-$. For any fixed cost C^t we use their algorithm to determine (with high probability) whether $\mathcal{X}^-$ intersects with $\mathcal{B}_{C^t}$; that is, whether or not C^t is a new lower or upper bound on the MAC. Again by applying a binary search, we find an ϵ-IMAC with a high degree of confidence in no more than L_ϵ repetitions. We now focus only on weighted L_1 costs and return to more general cases in Section 3.4.2.

The final Algorithm 9 runs with polynomial query complexity[4] $\mathcal{O}^*\,(D^5 L_\epsilon)$. The following sections provide a detailed sketch of the steps followed by the algorithm.

3.3.2.1 Procedure to Determine Whether Convex Sets Intersect

We begin by outlining the query-based procedure of Bertsimas and Vempala (2004) for determining whether two convex sets (*e.g.*, $\mathcal{X}^-$ and $\mathcal{B}_{C^t}$) intersect. Their Intersect-Search procedure (presented here as Algorithm 7) is a randomized Ellipsoid method for

[4] $\mathcal{O}^*\,(\cdot)$ denotes the standard complexity notation $\mathcal{O}\,(\cdot)$ up to logarithmic factors.

Algorithm 8 Hit-and-Run Sampling

HitRun $(\mathcal{P}, \{\mathbf{y}^j\}, \mathbf{x}^0)$
for all $i = 1 \dots K$ **do**
  Pick a random direction:
    $\nu_j \sim \mathrm{N}(0, 1)$
    $\mathbf{v} \leftarrow \sum_j \nu_j \mathbf{y}^j$
  Find ω_1 and ω_2 s.t.
    $\mathbf{x}^{i-1} - \omega_1 \mathbf{v} \notin \mathcal{P}$ and
    $\mathbf{x}^{i-1} + \omega_2 \mathbf{v} \notin \mathcal{P}$
  repeat
    $\omega \sim \mathrm{Unif}(-\omega_1, \omega_2)$
    $\mathbf{x}^i \leftarrow \mathbf{x}^{i-1} + \omega \mathbf{v}$
    if $\omega < 0$ **then** $\omega_1 \leftarrow -\omega$
    else $\omega_2 \leftarrow \omega$
  until $\mathbf{x}^i \in \mathcal{P}$
end for
Return: $\mathbf{x}^K$

determining whether there is an intersection between two bounded convex sets $\mathcal{P}$ and $\mathcal{B}$ with the following properties: $\mathcal{P}$ is only accessible through membership queries and $\mathcal{B}$ provides a separating hyperplane for any point outside it. The reader should view $\mathcal{P}$ and $\mathcal{B}$ as something like $\mathcal{X}^-$ and $\mathcal{B}_{C^t}$ which certainly satisfy these properties. Their technique uses efficient query-based approaches to uniformly sample from the convex set $\mathcal{P}$ to obtain sufficiently many samples such that cutting $\mathcal{P}$ through the centroid of these samples with a separating hyperplane from $\mathcal{B}$ will significantly reduce the volume of $\mathcal{P}$ with high probability. The technique thus constructs a sequence of progressively smaller feasible sets $\mathcal{P}^s \subset \mathcal{P}^{s-1}$ until either it finds a point in $\mathcal{P} \cap \mathcal{B}$ or it is highly unlikely that the sets intersect. We now detail how we use their technique to efficiently evade filtering.

So far we have reduced our problem to finding the intersection between $\mathcal{X}^-$ and $\mathcal{B}_{C^t}$. An important initialization step, however, is due to the algorithm being designed to test the intersection of *bounded* convex sets, while $\mathcal{X}^-$ may be unbounded. Let $R = 2\mathrm{A}(\mathbf{x}^-)$ and $\mathcal{B}_{2R}(\mathbf{x}^-)$ be the L_1-ball of radius $2R$ centered at $\mathbf{x}^-$. Since we are minimizing a cost, we can consider the set $\mathcal{P}^0 = \mathcal{X}^- \cap \mathcal{B}_{2R}(\mathbf{x}^-)$, which is a subset of $\mathcal{X}^-$ containing $\mathcal{B}_{C^t}$ and thus also the intersection $\mathcal{X}^- \cap \mathcal{B}_{C^t}$ if it exists—since $C^t < \mathrm{A}(\mathbf{x}^-)$. A final initial remark is that we also assume that there is some $r > 0$ such that there is an L_1-ball of radius r contained in the convex set $\mathcal{X}^-$.

Before detailing the IntersectSearch procedure (*cf.* Algorithm 7), we summarize the hit-and-run random walk technique introduced by Smith (1996) (*cf.* Algorithm 8), which is the backbone of IntersectSearch and is used to sample uniformly from a bounded convex body. Given an instance $\mathbf{x}^j \in \mathcal{P}^{s-1}$, hit-and-run selects a random direction $\mathbf{v}$ through $\mathbf{x}^j$ (we return to the selection of $\mathbf{v}$ in Section 3.3.2.2). Since $\mathcal{P}^{s-1}$ is a bounded convex set, the set $\Omega = \{\omega \mid \mathbf{x}^j + \omega \mathbf{v} \in \mathcal{P}^{s-1}\}$ is a bounded interval of all feasible points along direction $\mathbf{v}$ through $\mathbf{x}^j$. Sampling ω uniformly from Ω (using rejection sampling) yields

the next step of the random walk; $\mathbf{x}^j + \omega\mathbf{v}$. Under the appropriate conditions (addressed in Section 3.3.2.2), the HIT-AND-RUN random walk generates a sample uniformly from the convex body after $\mathcal{O}^*(D^3)$ steps (Lovász and Vempala, 2004).

Using HIT-AND-RUN we obtain samples $\{\mathbf{x}^j\}_{j=1}^{2N}$ from $\mathcal{P}^{s-1} \subset \mathcal{X}^-$ and determine if any satisfy $\mathrm{A}(\mathbf{x}^j) \leq C^t$. If so, $\mathbf{x}^j$ is in the intersection of $\mathcal{X}^-$ and $\mathcal{B}_{C^t}$ and the procedure is complete. Otherwise, our goal is to significantly reduce the size of $\mathcal{P}^{s-1}$ without excluding any of $\mathcal{B}_{C^t}$, so that our sampling concentrates toward the intersection (if it exists)—for this we need a separating hyperplane for $\mathcal{B}_{C^t}$. For any point $\mathbf{y} \notin \mathcal{B}_{C^t}$, the (sub)gradient of the weighted L_1 cost at $\mathbf{y}$ is given by $\mathbf{h}^\mathbf{y}$ with components

$$h_f^\mathbf{y} \;=\; c_f \operatorname{sgn}\left(y_f - x_f^{\mathrm{A}}\right) \;. \tag{3.8}$$

This is the normal to a separating hyperplane for $\mathbf{y}$ and $\mathcal{B}_{C^t}$.

To achieve efficiency, we choose a point $\mathbf{z} \in \mathcal{P}^{s-1}$ so that cutting $\mathcal{P}^{s-1}$ through $\mathbf{z}$ with the hyperplane $\mathbf{h}^\mathbf{z}$ eliminates a significant fraction of $\mathcal{P}^{s-1}$. To do so, $\mathbf{z}$ must be centrally located within $\mathcal{P}^{s-1}$—we use the empirical centroid of half of our samples in $\mathbf{z} = \frac{1}{N}\sum_{j=1}^{N}\mathbf{x}^j$ (the other half will be used in Section 3.3.2.2). We cut $\mathcal{P}^{s-1}$ with the hyperplane $\mathbf{h}^\mathbf{z}$ through $\mathbf{z}$; that is, $\mathcal{P}^s = \mathcal{P}^{s-1} \cap \mathcal{H}_\mathbf{z}$ where $\mathcal{H}_\mathbf{z}$ is the halfspace

$$\mathcal{H}_\mathbf{z} \;=\; \{\mathbf{x} \mid \mathbf{x}^\top\mathbf{h}^\mathbf{z} \leq \mathbf{z}^\top\mathbf{h}^\mathbf{z}\} \;. \tag{3.9}$$

As shown by Bertsimas and Vempala (2004), this cut achieves $\mathrm{vol}(\mathcal{P}^s) \leq \frac{2}{3}\mathrm{vol}(\mathcal{P}^{s-1})$ with high probability so long as $N = \mathcal{O}^*(D)$ and $\mathcal{P}^{s-1}$ is sufficiently round (see Section 3.3.2.2). Observing that the ratio of the volumes between the initial circumscribing and inscribing balls of the feasible set is $\left(\frac{R}{r}\right)^D$, the algorithm can terminate after $T = \mathcal{O}\left(D\log\frac{R}{r}\right)$ unsuccessful iterations with a high probability that the intersection is empty.

Because every iteration in Algorithm 7 requires $N = \mathcal{O}^*(D)$ samples, each of which need $K = \mathcal{O}^*(D^3)$ random walk steps, and there are $\mathcal{O}^*(D)$ iterations, the total number of membership queries required by Algorithm 7 is $\mathcal{O}^*(D^5)$.

3.3.2.2 Efficient Sampling from Convex Bodies with Membership Oracles

Until this point, we assumed the HIT-AND-RUN random walk efficiently produces uniformly random samples from any bounded convex body $\mathcal{P}$ accessible through membership queries. However, if the body is severely elongated, randomly selected directions will rarely align with the long axis of the body and our random walk will take small steps (relative to the long axis) and mix slowly. Essentially, we require that the convex body be well-rounded. More formally, for the walk to mix effectively, we need the convex body $\mathcal{P}$ to be *near-isotropic*; *i.e.*, for any unit vector $\mathbf{v}$, $\mathbb{E}_{\mathbf{X}\sim\mathcal{P}}\left[\left(\mathbf{v}^\top(\mathbf{X} - \mathbb{E}_{\mathbf{Y}\sim\mathcal{P}}[\mathbf{Y}])\right)^2\right]$ is bounded between 1/2 and 3/2 of $\mathrm{vol}(\mathcal{P})$.

If $\mathcal{P}$ is not near-isotropic, we must rescale $\mathcal{X}$ with an appropriate affine transformation $\mathbf{V}$ so the resulting body $\mathcal{P}'$ is near-isotropic. With sufficiently many samples from $\mathcal{P}$ we can estimate $\mathbf{V}$ as the empirical covariance matrix. However, instead of rescaling $\mathcal{X}$ explicitly, we do so implicitly using a technique described by Bertsimas and Vempala (2004). To do

Algorithm 9 Convex $\mathcal{X}^-$ Set Search

SetSearch $\left(\mathcal{P}, \mathcal{Q} = \{\mathbf{x}^j \in \mathcal{P}\}, C_0^-, C_0^+, \epsilon\right)$
$\mathbf{x}^* \leftarrow \mathbf{x}^-$ and $t \leftarrow 0$
while $C_t^- / C_t^+ > 1 + \epsilon$ **do**
  $C_t \leftarrow \sqrt{C_t^- \cdot C_t^+}$
  $[\mathbf{x}^*, \mathcal{P}', \mathcal{Q}'] \leftarrow$ IntersectSearch $(\mathcal{P}, \mathcal{Q}, C)$
  if intersection found **then**
    $C_{t+1}^- \leftarrow \mathrm{A}(\mathbf{x}^*)$ and $C_{t+1}^+ \leftarrow C_t^+$
    $\mathcal{P} \leftarrow \mathcal{P}'$ and $\mathcal{Q} \leftarrow \mathcal{Q}'$
  else
    $C_{t+1}^- \leftarrow C_t^-$ and $C_{t+1}^+ \leftarrow C_t$
  end if
  $t \leftarrow t + 1$
end while
Return: $\mathbf{x}^*$

so, we maintain a set $\mathcal{Q}$ of sufficiently many uniform samples from the body $\mathcal{P}^s$ and in the HIT-AND-RUN algorithm we sample directions based on this set. Intuitively, because the samples in $\mathcal{Q}$ are distributed uniformly in $\mathcal{P}^s$, the directions we sample based on the points in $\mathcal{Q}$ implicitly reflect the covariance structure of $\mathcal{P}^s$. This is equivalent to sampling the direction from a normal distribution with the covariance of $\mathcal{P}$.

We must ensure that $\mathcal{Q}$ has sufficiently many samples from $\mathcal{P}^s$ after each cut $\mathcal{P}^s \leftarrow \mathcal{P}^{s-1} \cap \mathcal{H}_{\mathbf{z}^s}$. Recall that we initially sampled $2N$ points from $\mathcal{P}^{s-1}$ using our HIT-AND-RUN procedure—half of these were used to estimate the centroid $\mathbf{z}^s$ for the cut and the other half, $\mathcal{S}$, are used to repopulate $\mathcal{Q}$ after the cut. Because $\mathcal{S}$ contains independent uniform samples from $\mathcal{P}^{s-1}$, those in $\mathcal{P}^s$ after the cut constitute independent uniform samples from $\mathcal{P}^s$ (along the same lines of rejection sampling). By choosing N sufficiently large, our cut will be sufficiently deep and we will have sufficiently many points to resample $\mathcal{P}^s$ after the cut.

Finally, before we start this resampling procedure, we need an initial set $\mathcal{Q}$ of uniformly distributed points from $\mathcal{P}^0$ but, in our problem, we only have a single point $\mathbf{x}^- \in \mathcal{X}^-$. Fortunately, there is an iterative procedure for putting our convex set $\mathcal{P}^0$ into a near-isotropic position—the RoundingBody algorithm described by Lovász and Vempala (2003) uses $\mathcal{O}^*(D^4)$ membership queries to transforms the convex body into near-isotropic position. We use this as a preprocessing step for Algorithms 7 and 9; that is, given $\mathcal{X}^-$ and $\mathbf{x}^- \in \mathcal{X}^-$ we make $\mathcal{P}^0 = \mathcal{X}^- \cap \mathcal{B}_{2R}(\mathbf{x}^-)$ and then use the RoundingBody algorithm to produce $\mathcal{Q} = \{\mathbf{x}^j \in \mathcal{P}^0\}$. These sets are then the inputs to our search algorithms.

3.3.2.3 Optimization over L_1 Balls

We now revisit the outermost optimization loop for searching for the minimum feasible cost and suggest improvements—if naively implemented as described, the algorithm does work.

First, since $\mathbf{x}^{\mathrm{A}}$, $\mathbf{x}^{-}$ and are the same for every iteration of the optimization procedure, we only need to run the RoundingBody procedure once as a preprocessing step. The set of samples $\{\mathbf{x}^j \in \mathcal{P}^0\}$ it produces are sufficient to initialize IntersectSearch at each stage of the binary search over C^t.

Second, the separating hyperplane $\mathbf{h}^{\mathbf{y}}$ given by Equation (3.8) does not depend on the target cost C^t but only on $\mathbf{x}^{\mathrm{A}}$, the common center of all the L_1 balls. In fact, the separating hyperplane at point $\mathbf{y}$ is valid for all weighted L_1-balls of cost $C < \mathrm{A}(\mathbf{y})$. Further, if $C < C^t$, we have $\mathcal{B}_C \subset \mathcal{B}_{C^t}$. Thus, the final state from a successful call to IntersectSearch for the C^t-cost ball can serve as the starting state for any subsequent call to IntersectSearch for all $C < C^t$.

These improvements are reflected in our final procedure SetSearch in Algorithm 9.

3.4 Evasion while Minimizing L_p-distances

While the focus of Section 3.3, like that of Lowd and Meek (2005b), is attacks that almost-minimize L_1 cost, an adversary may instead value some other cost function. In this section we consider attacks for evading convex-inducing classifiers that almost-minimize L_p costs for $p \neq 1$. For certain values of p we show that, depending on whether the positive or negative class is convex, either the evasion problem requires an exponential number of queries or it can be efficiently solved by our existing algorithms. For other L_p costs, finding efficient evasion procedures remains an open question.

3.4.1 Convex Positive Classes

We now consider the application of the MultiLineSearch and K-step MultiLineSearch algorithms for evading convex-inducing classifiers under L_p costs when $p \neq 1$.

3.4.1.1 Multiline Search for $p \in (0, 1)$

In the case where $p < 1$, a simple reduction holds. Since a L_1-ball of radius R bounds radius-R L_p-balls for all $p < 1$, we can simply use our existing algorithms with the $2 \cdot D$ vertices of the hyperoctahedron (the L_1 ball) as search directions, to find an ϵ-IMAC while submitting the same number of queries as before.

3.4.1.2 Multiline Search for $p \in [1, \infty]$

If the level of approximation ϵ is permitted to increase with dimension D, then positive results are possible using our existing algorithms. The quality of our result, measured by the level of near-optimality we can guarantee via a range on ϵ, depends on how well the L_p ball is approximated by the L_1 ball.

Theorem 22. *Using the $2 \cdot D$ axis-parallel search directions, any of our multiline search algorithms efficiently find an ϵ-IMAC for*

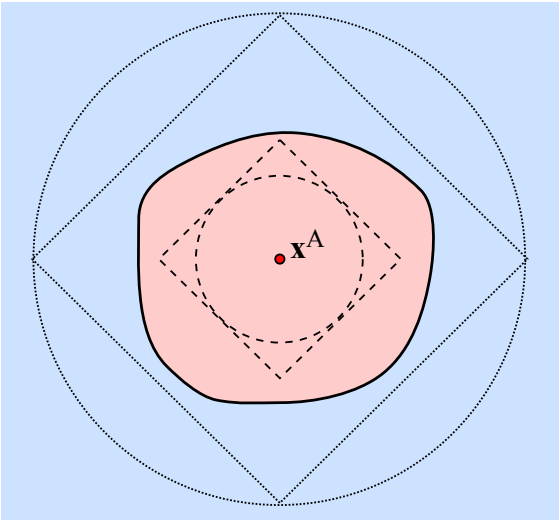


Figure 3.3: Relating upper and lower bounds on L_1 cost found by multiline search, to bounds on L_2 cost.

(i) L_p *cost, for* $p \in [1, \infty)$*, for all* $\epsilon > D^{(p-1)/p} - 1$*; and*

(ii) L_∞ *cost for all* $\epsilon > D - 1$*.*

Proof. First observe that for $p \in (1, \infty)$, the largest L_p ball that is contained inside a unit L_1 ball has radius $D^{(1-p)/p}$ since the L_p and L_1 balls must meet at the point $D^{-1}\mathbf{1}$ by symmetry and the L_p-norm of this point is $\alpha_p = D^{(1-p)/p}$. Similarly the largest L_∞ ball to achieve this feat has radius $\alpha_\infty = D^{-1}$.

Now consider running multiline search with the set of $2 \cdot D$ axis-parallel directions as the search direction set $\mathcal{W}$, until the L_1 cost bounds yield the stopping criterion $C_0^-/C_0^+ \leq 1+\epsilon'$. Notice, as depicted in Figure 3.3 for the $p = 2$ case, that while the upper bound on L_1 cost of C_0^- establishes the identical bound on L_p cost, the same is not true for the lower bound. The lower bound on L_p cost achieved by the search is simply $\alpha_p C_0^+$ where α_p is defined above for $p \in (1, \infty]$.

Thus we can guarantee that the search has found an ϵ-IMAC provided that $C_0^-/(\alpha_p C_0^+) \leq 1 + \epsilon$ which holds if $\alpha_p^{-1}(1 + \epsilon') \leq 1 + \epsilon$. If ϵ' is taken so that $\alpha_p \geq (1 + \epsilon')^{-1}$ then we have an ϵ-IMAC if $(1 + \epsilon')^2 \leq 1 + \epsilon$. This condition is satisfied by taking $\epsilon' = \sqrt{1 + \epsilon} - 1$.

Thus provided that $\alpha_p \geq 1/(1 + \epsilon')$, running a multiline search with query complexity

in terms of the following quantity is sufficient to find an ϵ-IMAC:

$$\begin{aligned} L_{\epsilon'} &= \mathcal{O}\left(\log\left(\frac{1}{\log(1+\epsilon')}\right)\right) \\ &= \mathcal{O}\left(\log\left(\frac{1}{\log\sqrt{1+\epsilon}}\right)\right) \\ &= \mathcal{O}\left(\log\left(\frac{1}{\log(1+\epsilon)}\right)\right) \\ &= \mathcal{O}\left(L_\epsilon\right) \quad . \end{aligned}$$

And so the procedure still has the same polynomial query complexity in D and L_ϵ, but with worse constants. Finally we have $\alpha_p \geq 1/(1+\epsilon')$ for $p \in [1,\infty)$ iff $\epsilon' \geq D^{(p-1)/p} - 1$, and for $p = \infty$ iff $\epsilon' \geq D - 1$. □

These results incur the same query complexities as for the L_1 case since we are using the same set of search directions.[5] Better results, in the sense of achieving lower ϵ's, are possible by using additional search directions (the intuition being that more directions better approximate L_p balls). The cost of expanding the search set $\mathcal{W}$ is that the query complexity increases.

If ϵ is constant wrt the dimension D then evasion for $p \in [2,\infty]$ requires an exponential number of queries in D: *no efficient algorithm exists.*

Theorem 23. *For any $D > 0$, any initial bounds $0 < C_0^+ < C_0^-$ on the* MAC*, and $0 < \epsilon < 1$, all algorithms must submit at least α_ϵ^D membership queries (where $\alpha_\epsilon > 1$) in the worst case to be ϵ-multiplicatively optimal on all classifiers with convex positive classes, for L_∞ costs.*

Proof. We proceed by constructing two convex positive sets consistent with the responses of the oracle, but with MAC's that are sufficiently different that no algorithm could simultaneously find an ϵ-IMAC for both without submitting many queries. The first convex positive set $\mathcal{P}_1$ is simply the L_∞ ball of cost one.

Consider the M queries made by the algorithm relative to $\mathbf{x}^{\mathrm{A}}$ that have cost at most 1. Each such query must fall in one of the 2^D octants of the L_∞ ball about $\mathbf{x}^{\mathrm{A}}$. Let us regard an octant as 'covered' if there is one or more queries in it. Define our second convex positive set $\mathcal{P}_2$ as the convex hull of all the octants covered by the M queries. $\mathcal{P}_1$ responds positively for these queries (and negatively for all others), while $\mathcal{P}_2$ certainly contains the M queries and so also positively responds to these. It also responds negatively to any other queries. Thus both candidate positive sets are convex, and responds consistently with any sequence of queries. The MAC for $\mathcal{P}_1$ is trivially one. We now consider when the MAC for $\mathcal{P}_2$ is much smaller, providing a separation between ϵ-IMAC's for the two candidate sets.

Each of the covered octants defining $\mathcal{P}_2$ can be identified with a point in the D-cube $\{0,1\}^D$. Let $\mathcal{C} \subseteq \{0,1\}^D$ be this subset of points. Suppose $\mathcal{C}$ is a K-covering of the D-cube

[5] As noted, the actual number of queries increases, but only by constant factors. For example, the basic multiline search's number of queries increases by an additional D queries: accounting for constants including in the logarithms' bases, $L_{\epsilon'} = 1 + L_\epsilon$.

but not a $(K-1)$-covering, for integer K wrt the hamming distance. Then there must be at least one element $\mathbf{v}$ of the D-cube that is at least K hamming distance from every member of $\mathcal{C}$; *i.e.*, $\mathbf{v}$ would not be covered by any $(K-1)$-ball centered around a point in $\mathcal{C}$. $\mathbf{v}$ corresponds to an un-occupied octant, and all occupied octants' representatives differ by at least K coordinates from $\mathbf{v}$. WLOG assume that all octants that differ by K coordinates are in fact in the covering. These octants border a face of the convex hull $\mathcal{P}_2$ which separates $\mathbf{v}$ from $\mathcal{P}_2$. For simplicity of explanation, we assume WLOG that $\mathbf{v}$ is the vector of all ones. The corners of this face then correspond to all vectors that have K zeros and $D-K$ ones. Since there must be a total of $(D-K)\binom{D}{K}$ ones distributed uniformly among the D coordinates over all the corners of the face, the midpoint of this face is then given by

$$\frac{1}{\binom{D}{K}}\left(\frac{D-K}{D}\binom{D}{K},\frac{D-K}{D}\binom{D}{K},\ldots,\frac{D-K}{D}\binom{D}{K}\right) \;=\; \frac{D-K}{D}\left(1,1,\ldots,1\right) \;,$$

which has an L_∞ cost of $(D-K)/D$. By the symmetry of $\mathcal{P}_2$, this midpoint minimizes cost over $\mathcal{P}_2$. That is, the MAC under $\mathcal{P}_2$ is $(D-K)/D$. By contrast the MAC under $\mathcal{P}_1$ is simply one.

Given the consistency of the two convex positive sets' responses, this implies that any algorithm that submits insufficient queries for a $K-1$ covering cannot achieve a multiplicative approximation better than $1+\epsilon \geq D/(D-K)$. Solving for K this yields

$$K \;\leq\; \frac{\epsilon}{1+\epsilon}D \;, \tag{3.10}$$

which relates a desirable ϵ to the radius of the covering; *i.e.*, for better approximations, a larger (lower radius) covering is required.

We next consider the number of queries required to achieve a covering of a given radius K. Consider such a cover. Each element of the covering covers exactly $\sum_{i=0}^{K}\binom{D}{i}$ elements of the D-cube. Since the cube has 2^D vertices, the covering's cardinality must be at least $\frac{2^D}{\sum_{k=0}^{h}\binom{D}{k}}$ to cover the entire D-cube. To further bound this quantity it suffices to use the bound

$$\sum_{k=0}^{\lfloor \delta D \rfloor}\binom{D}{k} \;\leq\; 2^{\mathrm{H}(\delta)D} \;, \tag{3.11}$$

where $0<\delta<1/2$ and $\mathrm{H}\left(\delta\right) = -\delta\log_2\delta - (1-\delta)\log(1-\delta)$ is the *entropy*. Let $K = D/2$ so that having no $K-1$ cover implies no approximation better than $\epsilon \geq 1$. By Equations (3.10) and (3.11), any algorithm must submit enough queries so that at least $2^{D(1-\mathrm{H}\left(\frac{\epsilon}{1+\epsilon}\right))}$ vertices of the hypercube are covered to achieve $0<\epsilon<1$. By construction of the 'covered' octants, an algorithm would need to submit at least this many queries to achieve $(1+\epsilon)$-multiplicative optimality. However, since $\mathrm{H}\left(\delta\right) < 1$ for $\delta < 1/2$, we have that the query complexity is $M = \Omega\left(\alpha_\epsilon^D\right)$ for $\alpha_\epsilon = 2^{1-\mathrm{H}\left(\frac{\epsilon}{1+\epsilon}\right)} > 1$. □

We can apply a similar argument for other L_p costs with $p>1$, to yield similar negative query complexity results.

Corollary 24. *For any $D > 0$, any initial bounds $0 < C_0^+ < C_0^-$ on the* MAC*, and $0 < \epsilon < \frac{2}{2^{1/p}} - 1$, all algorithms must submit at least α_ϵ^D membership queries (where $\alpha_\epsilon > 1$) in the worst case to be ϵ-multiplicatively optimal on all classifiers with convex positive classes, for L_p costs with $p > 1$.*

Proof. The same argument as used in Theorem 23 applies here, with minor modifications. Our first convex positive class $\mathcal{P}_1$ is now the unit-cost L_p ball. Again we can consider octants being covered by queries falling within $\mathcal{P}_1$; however octants are not restricted to this L_p ball too, and again $\mathcal{P}_2$ is taken as the convex hull of the covered octants.

The MAC under $\mathcal{P}_1$ is again one, however the derivation for $\mathcal{P}_2$'s MAC changes slightly. Now the corners defining the face separating $\mathbf{v}$ from $\mathcal{P}_2$ are no longer the vectors of K zero's and $D - K$ one's; instead the one's are replaced by some β such that the resulting vector has L_p cost one: the appropriate value of β is $(D-K)^{-1/p}$. As a result, the midpoint of the face is now $(D-K)^{1-1/p}/D\mathbf{1}$ which has an L_p cost of $((D-K)/D)^{1-1/p}$ representing the MAC under $\mathcal{P}_2$.

Thus without forming a $K - 1$ covering, the best approximation achievable is $1 + \epsilon \geq (D/(D-K))^{1-1/p}$. Solving for K this yields

$$K \;\leq\; D\frac{(1+\epsilon)^{p/(p-1)} - 1}{(1+\epsilon)^{p/(p-1)}} \;.$$

Now setting $K = \dim/2$ as before yields that for ϵ satisfying this relation, submitting fewer than $M = \Omega\left(\alpha_\epsilon^D\right)$ queries for $\alpha_\epsilon = 2^{1-\mathrm{H}\left(\frac{\beta_{\epsilon,p}-1}{\beta_{\epsilon,p}}\right)}$ cannot achieve better than a $(1+\epsilon)$-approximation, where $\beta_{\epsilon,p} = (1+\epsilon)^{p/(p-1)}$. As before we have that $\alpha_\epsilon^2 > 1$ since the argument to the entropy is at least $1/2$. Finally $1/2$ lower bounded this argument to the entropy implies a bound of $2^{(p-1)/p} - 1$ on ϵ, under which the result holds, where above the bound was simply one. □

Figure 3.4 plots the range $\epsilon \in \left(0, 2^{(p-1)/p} - 1\right)$ as a function of p. As $p \to \infty$ this upper bound quickly approaches 1, which coincides with the upper bound derived for the L_∞ cost.

While our positive results for the convex positive class $p > 1$ case provide approximations ϵ that must increase fairly quickly with D, it is most important to understand the query complexity of *good* approximations for small ϵ. Theorem 23 and the corollary provide such results.

Notably for the L_2 cost, the upper bound is $\sqrt{2} - 1 \approx 0.414$; for this case it is possible to derive a similar exponential query complexity result that holds for all ϵ by appealing to a result of Wyner (1965) on covering numbers for the surface of the hypersphere.

Theorem 25. *For any $D > 1$, any initial bounds $0 < C_0^+ < C_0^-$ on the* MAC*, and $0 < \epsilon < \frac{C_0^-}{C_0^+} - 1$, all algorithms must submit at least $\alpha_\epsilon^{\frac{D-2}{2}}$ membership queries (where $\alpha_\epsilon = \frac{(1+\epsilon)^2}{(1+\epsilon)^2-1} > 1$) in the worst case to be ϵ-multiplicatively optimal wrt the L_2 cost, on all classifiers with convex positive classes.*

We provide a proof sketch here. Once again the argument is to construct two candidate positive convex classes that produce identical responses but have very different MAC's. One

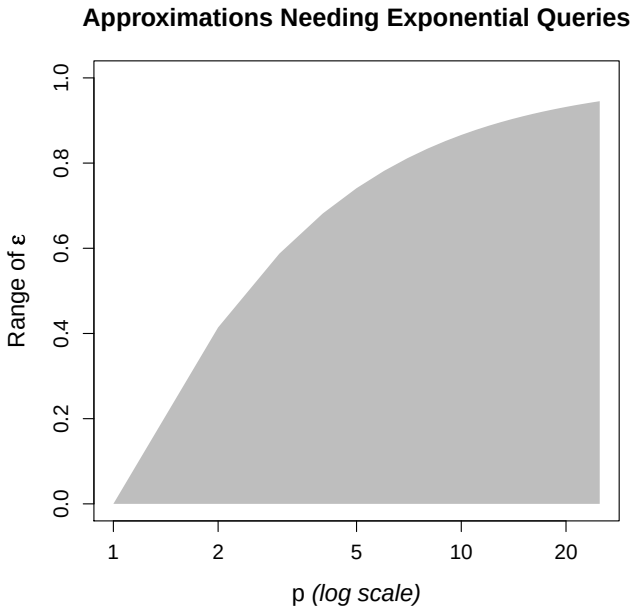


Figure 3.4: The values of ϵ (shown in gray), for each $p > 1$, for which minimizing L_p cost requires exponential numbers of queries according to Corollary 24. The cost parameter p is shown on a log scale. The upper-bound quickly approaches unity as $p \to \infty$.

class corresponds to the hypersphere itself, while the other is simply the convex hull of the queries that fall within the hypersphere. In the best case (for the attacker) these queries are on sphere's boundary. It is easy to see that the achieved approximation corresponds to the greatest height of the spherical caps defined by the difference between the hypersphere and the convex hull. Minimizing the spherical cap height (maximizing the accuracy of approximation) corresponds to covering the sphere's surface; the size of covers grows exponentially with D.

We can compare this proof technique with the technique used for general L_p costs. The looseness of the general technique comes from covering only (hyper)octants—several query points on the cost ball's surface, within a single octant, do not contribute to the evasion algorithm's approximation beyond a single query's contribution. In the improved result, the spherical caps need not be aligned according to the coordinate axes; we better lower bound the number of queries required to achieve approximations. It is thus conceivable that a similar argument could yield negative results that hold for all constant (wrt D) levels of approximation ϵ, for all $p > 1$.

3.4.2 Convex Negative Classes

Algorithm 9 applies to all costs that correspond to a weighted L_p distance centered at $\mathbf{x}^{\mathrm{A}}$, for $p \geq 1$, since such cost functions are convex. In these cases, the analogous gradient cuts

to those proposed in Section 3.3.2 are valid and once again applicable to any ball of smaller cost. To adapt the algorithm, one need only change cost function A and the separating hyperplane used for the halfspace cut in Equation (3.8).

Moreover SetSearch is applicable for any convex cost function A so long as we can compute the separating hyperplanes of any sublevel set[6] S of A for any point $\mathbf{y} \notin S$. For general convex costs, it still holds that the *sublevel set* of cost C (the C-cost ball at $\mathbf{x}^{\mathrm{A}}$) is contained in the sublevel set of cost D for all $D > C$. As a consequence, the separating hyperplanes for sublevel set at D are also separating hyperplanes for the set at cost C.

3.5 Summary

This chapter explores the evasion problem as defined by ϵ-IMAC searchability: using a polynomial number of membership queries to a fixed but unknown classifier, find a negative instance that almost-minimizes cost. This work generalizes the results of Lowd and Meek (2005b) which considers how best to launch Exploratory attacks on learning systems, *i.e.*, how to submit malicious test instances. The stated goal of this chapter of evasion (False Negatives) corresponds to Integrity attacks within the taxonomy of Barreno et al. (2006) as described in Section 1.2.2. However the methods are equally applicable to Availability attacks (that cause False Positives).

The analysis of our algorithms shows that convex-inducing classifiers is ϵ-IMAC searchable for L_p costs, for $p \leq 1$ in the convex positive class case and for $p \geq 1$ in the convex negative class case. We observe a phase transition when p crosses one: for convex positive classes evasion to minimize L_p cost with $p > 1$ requires exponential queries. When the positive class is convex we give efficient techniques that achieve a query complexity of $\mathcal{O}\left(\log(1/\epsilon) + \sqrt{\log(1/\epsilon)}D\right)$, outperforming previous reverse-engineering approaches for the special case of linear classifiers. We provide a lower bound of $\mathcal{O}\left(\log(1/\epsilon) + D\right)$ on the query complexity for any algorithm evading classifiers with convex positive class under the weighted L_1 cost, showing that our best algorithm is at least within a $\sqrt{\log(1/\epsilon)}$ factor to the optimal query complexity. We also consider variants of the search procedure for when the attacker does not have a lower bound on the MAC or has no negative instance from which to begin the search.

When the negative class is convex, we apply the randomized Ellipsoid-based method of Bertsimas and Vempala (2004) to achieve efficient ϵ-IMAC search, achieving a polynomial query complexity of $\mathcal{O}^*\left(D^5 \log(1/\epsilon)\right)$ while minimizing L_p cost for $p \geq 1$. With prior knowledge that a learner produces classifiers with a specific class being convex, an adversary can select the appropriate query algorithm to evade detection. If the adversary is unaware of which of the positive and negative sets are convex, but has prior knowledge that one is convex, they can run both searches concurrently to discover an ϵ-IMAC with a combined polynomial query complexity.

Solutions to the reverse engineering problem are clearly sufficient for the evasion problem.

[6]The sublevel set of any convex function is a convex set (Boyd and Vandenberghe, 2004) so such a separating hyperplane always exists but may not be easily computed.

Lowd and Meek (2005b) used a reverse engineer approach to evading linear classification without significant cost to query complexity. An important consequence of the polynomial evadability of convex-inducing classifiers is that for this class of classifiers, reverse engineering is not necessary for evasion of convex-inducing classifiers and in fact can be significantly harder.

Exploring near-optimal evasion is important for understanding how an adversary may circumvent learners in security-sensitive settings. As described here, our algorithms may not always directly apply in practice since various real-world obstacles persist. Queries may be only partially observable or noisy and the feature set may be only partially known. Moreover, an adversary may not be able to query all $\mathbf{x} \in \mathcal{X}$. Queries are almost always objects (such as email messages) that are mapped into $\mathcal{X}$ by the adaptive system. A real-world adversary must invert the feature-mapping—a generally difficult task. These limitations necessitate further research on the impact of partial observability and approximate querying on ϵ-IMAC search, and to design more secure filters.

Chapter 4

Privacy-Preserving Learning

You have zero privacy anyway. Get over it.
– Scott McNealy, Co-Founder, Sun Microsystems

Privacy-preserving learning is a relatively young field in the intersection of Security, Database, TCS, Statistics and Machine Learning research. The broad goal of research into privacy-preserving learning is to release aggregate statistics on a dataset without disclosing local information about individual elements of the data.

Several recent studies in privacy-preserving learning have considered the trade-off between utility or risk and the level of differential privacy guaranteed by mechanisms for statistical query processing. In this chapter we study this trade-off in private Support Vector Machine (SVM) learning. We present two efficient mechanisms, one for the case of finite-dimensional feature mappings and one for potentially infinite-dimensional feature mappings with translation-invariant kernels. For the case of translation-invariant kernels, the proposed mechanism minimizes regularized empirical risk in a random Reproducing Kernel Hilbert Space whose kernel uniformly approximates the desired kernel with high probability. This technique, borrowed from large-scale learning, allows the mechanism to respond with a finite encoding of the classifier, even when the function class is of infinite Vapnik-Chervonenkis (VC) dimension. Differential privacy is established using a proof technique from algorithmic stability. Utility—the mechanism's response function is pointwise ϵ-close to non-private SVM with probability $(1 - \delta)$—is proven by appealing to the smoothness of regularized empirical risk minimization with respect to small perturbations to the feature mapping. We conclude with a lower bound on the optimal differential privacy of the SVM. This negative result states that for any δ, no mechanism can be simultaneously (ϵ, δ)-useful and β-differentially private for small ϵ and small β.

In the language of the taxonomy of Barreno et al. (2006), the privacy-preserving mechanisms developed in this chapter are robust to both Exploratory and Causative attacks, which aim to violate Confidentiality. An attacker with access to the released classifier can probe it in an attempt to reveal information about the training data; moreover an attacker with influence over (up to) all but one example of the training data may attempt to manipulate the mechanism into revealing information about the unknown training example. In both

cases our strong guarantees of differential privacy prove that such attacks will fail. Finally, our analysis considers adversaries with near-complete knowledge of the training data ($n-1$ out of n examples), complete knowledge about the mechanism up to sources of randomness, and access to the released classifier trained on the data. Similarly, the attacker may have complete control over the known subset of training data. In sum, we allow for substantial levels of adversarial information and control.

4.1 Introduction

The goal of a well-designed statistical database is to provide aggregate information about a database's entries while maintaining individual entries' privacy. These two goals of *utility* and *privacy* are inherently discordant. For a mechanism to be useful, its responses must closely resemble some target statistic of the database's entries. However to protect privacy, it is often necessary for the mechanism's response distribution to be 'smoothed out', *i.e.*, the mechanism must be randomized to reduce the individual entries' influence on this distribution. It has been of key interest to the statistical database community to understand when the goals of utility and privacy can be efficiently achieved simultaneously (Barak et al., 2007; Blum et al., 2008; Chaudhuri and Monteleoni, 2009; Dinur and Nissim, 2003; Dwork et al., 2007; Kasiviswanathan et al., 2008). In this chapter we consider the practical goal of private regularized empirical risk minimization (ERM) in Reproducing Kernel Hilbert Spaces for the special case of the Support Vector Machine (SVM). We adopt the strong notion of differential privacy as formalized by Dwork (2006). Our efficient new mechanisms are shown to parametrize functions that are close to non-private SVM under the L_∞-norm, with high probability. In our setting this notion of utility is stronger than closeness of risk (*cf.* Remark 28).

We employ a number of algorithmic and proof techniques new to differential privacy. One of our new mechanisms borrows a technique from large-scale learning, in which regularized ERM is performed in a random feature space whose inner-product uniformly approximates the target feature space inner-product. This random feature space is constructed by viewing the target kernel as a probability measure in the Fourier domain. This technique enables the finite parametrization of responses from function classes with infinite VC dimension. To establish utility, we show that regularized ERM is relatively insensitive to perturbations of the kernel: not only does the technique of learning in a random RKHS enable finitely-encoded privacy-preserving responses, but these responses well-approximate the responses of non-private SVM. Together these two techniques may prove useful in extending privacy-preserving mechanisms to learn in large function spaces. To prove differential privacy, we borrow a proof technique from the area of algorithmic stability. We believe that stability may become a fruitful avenue for constructing new private mechanisms in the future, based on learning maps presently known to be stable.

Of particular interest, is the optimal differential privacy of the SVM, which loosely speaking is the best level of privacy achievable by any accurate mechanism for SVM learning. Through our privacy-preserving mechanisms for the SVM, endowed with guarantees of utility, we upper bound optimal differential privacy. We also provide lower bounds on

the SVM's optimal differential privacy, which are impossibility results for simultaneously achieving high levels of utility and privacy.

An earlier version of this chapter appeared as a technical report (Rubinstein et al., 2009). Subsequently, but independently, Sarwate et al. (2009) recently considered privacy-preserving mechanisms for SVM learning. Their mechanism for linear SVM guarantees differential privacy by adding a random term to the objective as they pioneered for regularized logistic regression (Chaudhuri and Monteleoni, 2009). For non-linear SVM the authors exploit the same technique from large-scale learning we use here. The authors also develop a method for tuning the regularization parameter while preserving privacy, using a comparison procedure due to McSherry and Talwar (2007). It is noteworthy that preserving privacy via the randomized objective applies only to differentiable loss functions, ruling out the important case of hinge-loss. Our mechanisms preserve privacy for any convex loss. And while Sarwate et al. (2009) prove risk bounds for their mechanisms, our utility guarantees are strictly stronger for hinge-loss (*cf.* Remark 28) and we provide lower bounds on simultaneously achievable utility and privacy. Finally our proof of differential privacy is interesting due to its novel use of stability.

Chapter Organization. The remainder of this chapter is organized as follows. After concluding this section with a summary of related work, we recall basic concepts of differential privacy and SVM learning in Section 4.2. Sections 4.3 and 4.4 describe the new mechanisms for private SVM learning for finite-dimensional feature maps and (potentially infinite-dimensional) feature maps with translation-invariant kernels. Each mechanism is accompanied with proofs of privacy and utility bounds. Section 4.5 considers the special case of hinge-loss and presents an upper bound on the SVM's optimal differential privacy. A corresponding lower bound is then given in Section 4.6. We conclude the chapter with a summary of our contributions.

4.1.1 Related Work

There is a rich literature of prior work on differential privacy in the theory community. The following sections summarize work related to our own, organized to contrast this work with our main contributions.

Range Spaces Parametrizing Vector-Valued Statistics or Simple Functions. Early work on private interactive mechanisms focused on approximating real- and vector-valued statistics (*e.g.*, Barak et al. 2007; Blum et al. 2005; Dinur and Nissim 2003; Dwork 2006; Dwork et al. 2006). McSherry and Talwar (2007) first considered private mechanisms with range spaces parametrizing sets more general than real-valued vectors, and used such differentially private mappings for mechanism design. More related to our work are the private mechanisms for regularized logistic regression proposed and analyzed by Chaudhuri and Monteleoni (2009). There the mechanism's range space parametrizes the VC-dimension $d+1$ class of linear hyperplanes in $\mathbb{R}^d$. Kasiviswanathan et al. (2008) showed that discretized

concept classes can be PAC learned or agnostically learned privately, albeit via an inefficient mechanism. Blum et al. (2008) showed that non-interactive mechanisms can privately release anonymized data such that utility is guaranteed over classes of predicate queries with polynomial VC dimension, when the domain is discretized. Dwork et al. (2009) more recently characterized when utility and privacy can be achieved by efficient non-interactive mechanisms. In this paper we consider efficient mechanisms for private SVM learning, whose range spaces parametrize real-valued functions (whose sign form trained classifiers). One case covered by our analysis is learning with a Gaussian kernel, which corresponds to learning over a rich class of infinite dimension.

Practical Privacy-Preserving Learning (Mostly) via Subset-Sums. Most prior work in differential privacy has focused on the deep analysis of mechanisms for relatively simple statistics (with histograms and contingency tables as explored by Blum et al. 2005 and Barak et al. 2007 respectively, as examples) and learning algorithms (*e.g.*, interval queries and half-spaces as explored by Blum et al. 2008), or on constructing learning algorithms that can be decomposed into subset-sum operations (*e.g.*, perceptron, k-NN, ID3 as described by Blum et al. 2005, and various recommender systems due to the work of McSherry and Mironov 2009). By contrast, we consider the practical goal of SVM learning, which does not decompose into subset-sums. It is also notable that our mechanisms run in polynomial time. The most related work to our own in this regard is due to Chaudhuri and Monteleoni (2009), although their results hold only for differentiable loss, and finite feature mappings.

The Privacy-Utility Trade-Off. Like several prior studies, we consider the trade-off between privacy and utility. Barak et al. (2007) presented a mechanism for releasing contingency tables that guarantees differential privacy and also guarantees a notion of accuracy: with high probability all marginals from the released table are close in L_1-norm to the true table's marginals. As mentioned above, Blum et al. (2008) developed a private non-interactive mechanism that releases anonymized data such that all predicate queries in a VC-class take on similar values on the anonymized data and original data. In the work of Kasiviswanathan et al. (2008), utility corresponds to PAC learning: with high probability the response and target concepts are close, averaged over the underlying measure.

A sequence of prior negative results have shown that any mechanism providing overly accurate responses cannot be private (Dinur and Nissim, 2003; Dwork and Yekhanin, 2008; Dwork et al., 2007). Dinur and Nissim (2003) showed that if noise of rate only $o(\sqrt{n})$ is added to subset sum queries on a database of bits then an adversary can reconstruct a $1 - o(1)$ fraction of the database. This is a threshold phenomenon that says if accuracy is too high, privacy cannot be guaranteed at all. This result was more recently extended to allow for mechanisms that answer a small fraction of queries arbitrarily (Dwork et al., 2007). We show a similar negative result for the private SVM setting: any mechanism that is too accurate with respect to the SVM cannot guarantee strong levels of privacy.

Connections between Stability, Robust Statistics, and Global Sensitivity. To prove differential privacy, we borrow a proof technique from the area of algorithmic stability. In passing Kasiviswanathan et al. (2008) note the similarity between notions of algorithmic stability and differential privacy, however do not exploit this. The connection between algorithmic stability and differential privacy is qualitatively similar to the recent work of Dwork and Lei (2009) who demonstrated that robust estimators can serve as the basis for private mechanisms, by exploiting the limited influence of outliers on such estimators.

4.2 Background and Definitions

A *database* D is a sequence of $n > 1$ *entries* or *rows* $(\mathbf{x}_i, y_i) \in \mathbb{R}^d \times \{-1, 1\}$, which are input point-label pairs or *examples*. We say that a pair of databases D_1, D_2 are *neighbors* if they differ on one entry. A *mechanism* M is a service trusted with access to a database D, that releases aggregate information about D while maintaining privacy of individual entries. By $M(D)$ we mean the *response* of M on D. We assume that this is the only information released by the mechanism. Denote the range space of M by $\mathcal{T}_M$. We adopt the following strong notion of differential privacy due to Dwork (2006).

Definition 26. *For any $\beta > 0$, a randomized mechanism M provides* β-differential privacy*, if, for all neighboring databases D_1, D_2 and all responses $t \in \mathcal{T}_M$ the mechanism satisfies*

$$\log\left(\frac{\Pr\left(M(D_1) = t\right)}{\Pr\left(M(D_2) = t\right)}\right) \quad \leq \quad \beta \ .$$

The probability in the definition is over the randomization in M. For continuous $\mathcal{T}_M$ we mean by this ratio a Radon-Nikodym derivative of the distribution of $M(D_1)$ with respect to the distribution of $M(D_2)$. If an adversary knows M and the first $n-1$ entries of D, she may simulate the mechanism with different choices for the missing example. If the mechanism's response distribution varies smoothly with her choice, the adversary will not be able to infer the true value of entry n by querying M. In the sequel we assume WLOG that each pair of neighboring databases differ on their last entry.

Intuitively the more an 'interesting' mechanism M is perturbed to guarantee differential privacy, the less like M the resulting mechanism $\hat{M}$ will become. The next definition formalizes the notion of 'likeness'.

Definition 27. *Consider two mechanisms $\hat{M}$ and M with the same domain and with response spaces $\mathcal{T}_{\hat{M}}$ and $\mathcal{T}_M$. Let $\mathcal{X}$ be some set and let $\mathcal{F}$ be a space of real-valued functions on $\mathcal{X}$ that is parametrized by the response spaces: for every $t \in \mathcal{T}_{\hat{M}} \cup \mathcal{T}_M$ let $f_t \in \mathcal{F}$ be some function. Finally assume $\mathcal{F}$ is endowed with norm $\|\cdot\|_{\mathcal{F}}$. Then for $\epsilon > 0$ and $0 < \delta < 1$ we say that*[1] *$\hat{M}$ is* (ϵ, δ)-useful *with respect to M if, for all databases D,*

$$\Pr(\|f_{\hat{M}(D)} - f_{M(D)}\|_{\mathcal{F}} \leq \epsilon) \quad \geq \quad \delta \ .$$

[1] We are overloading the term '(ϵ, δ)-usefulness' introduced by Blum et al. (2008) for non-interactive mechanisms. Our definition is analogous for the present setting of privacy-preserving learning, where a single function is released.

Algorithm 10 SVM

Inputs: database $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ with $\mathbf{x}_i \in \mathbb{R}^d$, $y_i \in \{-1, 1\}$; kernel $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$; convex loss function ℓ; parameter $C > 0$.

1. $\boldsymbol{\alpha}^\star \leftarrow$ Solve the SVM's dual QP; and
2. Return vector $\boldsymbol{\alpha}^\star$.

Typically $\hat{M}$ will be a privacy-preserving version of M, that has been perturbed somehow. In the sequel we will take $\|\cdot\|_{\mathcal{F}}$ to be the sup-norm over a subset $\mathcal{M} \subseteq \mathbb{R}^d$ containing the data, which we denote by $\|f\|_{\infty;\mathcal{M}} = \sup_{\mathbf{x}\in\mathcal{M}} |f(\mathbf{x})|$. It will also be convenient to use the notation $\|k\|_{\infty;\mathcal{M}} = \sup_{\mathbf{x},\mathbf{y}\in\mathcal{M}} |k(\mathbf{x}, \mathbf{y})|$ for bivariate functions $k(\cdot, \cdot)$.

Remark 28. *In the sequel we develop privacy-preserving mechanisms that are useful with respect to the Support Vector Machine (see the next section for a brief introduction to the SVM). The SVM works to minimize the expected hinge-loss (i.e., risk in terms of the hinge-loss), which is a convex surrogate for the expected 0-1 loss. Since the hinge-loss is Lipschitz in the real-valued function output by the SVM, it follows that a mechanism $\hat{M}$ having utility with respect to the SVM also has expected hinge-loss that is within ϵ of the SVM's hinge-loss with high probability. That is, (ϵ, δ)-usefulness with respect to the sup-norm is stronger than guaranteed closeness of risk We consider the hinge-loss further in Sections 4.5 and 4.6. Until then we work with arbitrary convex, Lipschitz losses.*

We will see that the presented analysis does not simultaneously guarantee privacy at arbitrary levels and utility at arbitrary accuracy. The highest level of privacy guaranteed over all (ϵ, δ)-useful mechanisms with respect to a target mechanism M, is quantified by the optimal differential privacy for M. We define this notion for the SVM here, but the concept extends to any target mechanism of interest. We present upper and lower bounds on $\beta(\epsilon, \delta, C, n, \ell, k)$ for the SVM in Sections 4.5 and 4.6 respectively.

Definition 29. *For $\epsilon, C > 0$, $\delta \in (0, 1)$, $n > 1$, loss function $\ell(y, \hat{y})$ convex in $\hat{y}$, and kernel k, the* optimal differential privacy for the SVM *is the function*

$$\beta(\epsilon, \delta, C, n, \ell, k) \;=\; \inf_{\hat{M}\in\mathcal{I}} \sup_{(D_1,D_2)\in\mathcal{D}} \sup_{t\in\mathcal{T}_{\hat{M}}} \log\left(\frac{\Pr\left(\hat{M}(D_1) = t\right)}{\Pr\left(\hat{M}(D_2) = t\right)}\right) \;,$$

where $\mathcal{I}$ is the set of all (ϵ, δ)-useful mechanisms with respect to the SVM with parameter C, loss ℓ, and kernel k; and $\mathcal{D}$ is the set of all pairs of neighboring databases with n entries.

4.2.1 Support Vector Machines

Soft-margin SVM has convex Primal program $\min_{\mathbf{w}\in\mathbb{R}^F} \frac{1}{2}\|\mathbf{w}\|_2^2 + \frac{C}{n}\sum_{i=1}^n \ell\left(y_i, f_{\mathbf{w}}(\mathbf{x}_i)\right)$, where the $\mathbf{x}_i \in \mathbb{R}^d$ are *training input points* and the $y_i \in \{-1, 1\}$ are their *training labels*, n

Kernel	$g(\Delta)$	$p(\omega)$
RBF	$\exp\left(-\frac{\|\Delta\|_2^2}{2\sigma^2}\right)$	$\frac{1}{(2\pi)^{d/2}}\exp\left(\frac{-\|\omega\|_2^2}{2}\right)$
Laplacian	$\exp\left(-\|\Delta\|_1\right)$	$\prod_{i=1}^{d}\frac{1}{\pi\left(1+\omega_i^2\right)}$
Cauchy	$\prod_{i=1}^{d}\frac{2}{1+\Delta_i^2}$	$\exp\left(-\|\Delta\|_1\right)$

Table 4.1: Example translation-invariant kernels, their g functions and the corresponding Fourier transforms.

is the sample size, $\phi : \mathbb{R}^d \to \mathbb{R}^F$ is a *feature mapping* taking points in *input space* $\mathbb{R}^d$ to some (possibly infinite) F-dimensional *feature space*, $\ell(y, \hat{y})$ is a loss function convex in $\hat{y}$, and $\mathbf{w}$ is a hyperplane normal vector in feature space (Bishop, 2006; Burges, 1998; Cristianini and Shawe-Taylor, 2000; Schölkopf and Smola, 2001).

For finite F, predictions are taken as the sign of $f^\star(\mathbf{x}) = f_{\mathbf{w}^\star}(\mathbf{x}) = \langle\phi(\mathbf{x}), \mathbf{w}^\star\rangle$. We will refer to both $f_{\mathbf{w}}(\cdot)$ and $\operatorname{sgn}\left(f_{\mathbf{w}}(\cdot)\right)$ as *classifiers*, with the exact meaning apparent from the context. When F is large the solution may be more easily obtained via the dual. For example, see Program (4.9) in Section 4.5 for the dual formulation of the hinge-loss $\ell(y, \hat{y}) = (1 - y\hat{y})_+$, which is the loss most commonly associated with soft-margin SVM. The vector of maximizing dual variables $\boldsymbol{\alpha}^\star$ returned by dualized SVM parametrizes the function $f^\star = f_{\boldsymbol{\alpha}^\star}$ as $f_{\boldsymbol{\alpha}}(\cdot) = \sum_{i=1}^{m}\alpha_i y_i k(\cdot, \mathbf{x}_i)$ where $k(\mathbf{x}, \mathbf{y}) = \langle\phi(\mathbf{x}), \phi(\mathbf{y})\rangle$ is the *kernel function*. *Translation-invariant kernels* are an important class of kernel given by functions with the form $k(\mathbf{x}, \mathbf{y}) = g(\mathbf{x} - \mathbf{y})$ (see Table 4.1 for examples). We define the *mechanism* SVM to be the dual optimization that responds with the vector $\boldsymbol{\alpha}^\star$, as described by Algorithm 10.

4.3 Mechanism for Finite Feature Maps

As a first step towards private SVM learning we begin by considering the simple case of finite F-dimensional feature maps. Algorithm 11 describes the PrivateSVM-Finite mechanism, which follows the established pattern of preserving differential privacy: after forming the primal solution to the SVM—an F-dimensional vector—the mechanism adds Laplace-distributed noise to the weight vector. Guaranteeing differential privacy proceeds via the established two-step process of calculating the L_1-sensitivity of the SVM's weight vector, then showing that β-differential privacy follows from sensitivity together with the choice of Laplace noise with scale equal to sensitivity divided by β.

To calculate sensitivity, we exploit the algorithmic stability of regularized ERM. Intuitively, stability corresponds to continuity of a learning map. Several notions of stability are known to lead to good generalization error bounds (Bousquet and Elisseeff, 2002; Devroye and Wagner, 1979; Kearns and Ron, 1999; Kutin and Niyogi, 2002), sometimes in cases where class capacity-based approaches such as VC theory do not apply. A *learning map* $\mathcal{A}$ is a function that maps a database D to a classifier f_D; it is precisely the composition of a mechanism followed by the classifier parametrization mapping. A learning

Algorithm 11 PRIVATESVM-FINITE

Inputs: database $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ with $\mathbf{x}_i \in \mathbb{R}^d$, $y_i \in \{-1, 1\}$; finite feature map $\phi : \mathbb{R}^d \to \mathbb{R}^F$ and induced kernel k; convex loss function ℓ; and parameters $\lambda, C > 0$.

1. $\boldsymbol{\alpha}^\star \leftarrow$ Run Algorithm 10 on D with parameter C, kernel k, and loss ℓ;
2. $\tilde{\mathbf{w}} \leftarrow \sum_{i=1}^n \alpha_i^\star y_i \phi(\mathbf{x}_i)$;
3. $\boldsymbol{\mu} \leftarrow$ Draw i.i.d. sample of F scalars from Laplace $(0, \lambda)$; and
4. Return $\hat{\mathbf{w}} = \tilde{\mathbf{w}} + \boldsymbol{\mu}$

map $\mathcal{A}$ is said to have γ-*uniform stability* with respect to loss $\ell(\cdot, \cdot)$ if for all neighboring databases D, D', the losses of the classifiers trained on D and D' are close on all test examples $\|\ell(\cdot, \mathcal{A}(D)) - \ell(\cdot, \mathcal{A}(D'))\|_\infty \leq \gamma$ (Bousquet and Elisseeff, 2002). Our first lemma computes sensitivity of the SVM's weight vector for general convex, Lipschitz loss by following the proof of Schölkopf and Smola (2001, Theorem 12.4) which establishes that SVM learning has uniform stability (a result due to Bousquet and Elisseeff 2002).

Lemma 30. *Consider loss function $\ell(y, \hat{y})$ that is convex and L-Lipschitz in $\hat{y}$, and RKHS $\mathcal{H}$ induced by finite F-dimensional feature mapping ϕ with bounded kernel $k(\mathbf{x}, \mathbf{x}) \leq \kappa^2$ for all $\mathbf{x} \in \mathbb{R}^d$. Let $\mathbf{w}_S \in \mathbb{R}^F$ be the minimizer of the following regularized empirical risk function for each database $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$*

$$R_{\text{reg}}(\mathbf{w}, S) \;=\; \frac{C}{n} \sum_{i=1}^n \ell\left(y_i, f_{\mathbf{w}}(\mathbf{x}_i)\right) + \frac{1}{2}\|\mathbf{w}\|_2^2 \;.$$

Then for every pair of neighboring databases D, D' of n entries, $\|\mathbf{w}_D - \mathbf{w}_{D'}\|_1 \leq 4LC\kappa\sqrt{F}/n$.

Proof. We now calculate the sensitivity of the SVM primal weight vector for general convex, Lipschitz loss functions. For convenience we define $R_{\text{emp}}(\mathbf{w}, S) = n^{-1} \sum_{i=1}^n \ell(y_i, f_{\mathbf{w}}(\mathbf{x}_i)$ for any training set S, then the first-order necessary KKT conditions imply

$$\mathbf{0} \;\in\; \partial_{\mathbf{w}} R_{\text{reg}}(\mathbf{w}_D, D) \;=\; C\partial_{\mathbf{w}} R_{\text{emp}}(\mathbf{w}_D, D) + \mathbf{w}_D \;, \tag{4.1}$$

$$\mathbf{0} \;\in\; \partial_{\mathbf{w}} R_{\text{reg}}(\mathbf{w}_{D'}, D') \;=\; C\partial_{\mathbf{w}} R_{\text{emp}}(\mathbf{w}_{D'}, D') + \mathbf{w}_{D'} \;. \tag{4.2}$$

where $\partial_{\mathbf{w}}$ is the subdifferential operator with respect to $\mathbf{w}$. Define the auxiliary risk function

$$\tilde{R}(\mathbf{w}) \;=\; C\langle \partial_{\mathbf{w}} R_{\text{emp}}(\mathbf{w}_D, D) - \partial_{\mathbf{w}} R_{\text{emp}}(\mathbf{w}_{D'}, D'),\, \mathbf{w} - \mathbf{w}_{D'} \rangle + \frac{1}{2}\|\mathbf{w} - \mathbf{w}_{D'}\|_2^2 \;.$$

It is easy to see that $\tilde{R}(\mathbf{w})$ is strictly convex in $\mathbf{w}$ and that $\tilde{R}(\mathbf{w}_{D'}) = \{0\}$. And by Equation (4.2)

$$\begin{aligned} C\partial_{\mathbf{w}} R_{\text{emp}}(\mathbf{w}_D, D) + \mathbf{w} \;&\in\; C\partial_{\mathbf{w}} R_{\text{emp}}(\mathbf{w}_D, D) - C\partial_{\mathbf{w}} R_{\text{emp}}(\mathbf{w}_{D'}, D') + \mathbf{w} - \mathbf{w}_{D'} \\ &=\; \partial_{\mathbf{w}} \tilde{R}(\mathbf{w}) \;, \end{aligned}$$

which combined with Equation (4.1) implies $\mathbf{0} \in \partial_{\mathbf{w}} \tilde{R}(\mathbf{w}_D)$, so that $\tilde{R}(\mathbf{w})$ is minimized at $\mathbf{w}_D$. Thus there exists some non-positive $r \in \tilde{R}(\mathbf{w}_D)$. Next simplify the first term of $\tilde{R}(\mathbf{w}_D)$, scaled by n/C for notational convenience:

$$\begin{aligned}
& n\langle \partial_{\mathbf{w}} R_{\text{emp}}(\mathbf{w}_D, D) - \partial_{\mathbf{w}} R_{\text{emp}}(\mathbf{w}_{D'}, D'),\, \mathbf{w} - \mathbf{w}_{D'} \rangle \\
= & \sum_{i=1}^{n} \langle \partial_{\mathbf{w}} \ell\left(y_i, f_{\mathbf{w}_D}(\mathbf{x}_i)\right) - \partial_{\mathbf{w}} \ell\left(y_i', f_{\mathbf{w}_{D'}}(\mathbf{x}_i')\right),\, \mathbf{w} - \mathbf{w}_{D'} \rangle \\
= & \sum_{i=1}^{n-1} \left(\ell'\left(y_i, f_{\mathbf{w}_D}(\mathbf{x}_i)\right) - \ell'\left(y_i, f_{\mathbf{w}_{D'}}(\mathbf{x}_i)\right)\right)\left(f_{\mathbf{w}_D}(\mathbf{x}_i) - f_{\mathbf{w}_{D'}}(\mathbf{x}_i)\right) \\
& \quad + \ell'\left(y_n, f_{\mathbf{w}_D}(\mathbf{x}_n)\right)\left(f_{\mathbf{w}_D}(\mathbf{x}_n) - f_{\mathbf{w}_{D'}}(\mathbf{x}_n)\right) - \ell'\left(y_n', f_{\mathbf{w}_{D'}}(\mathbf{x}_n')\right)\left(f_{\mathbf{w}_D}(\mathbf{x}_n') - f_{\mathbf{w}_{D'}}(\mathbf{x}_n')\right) \\
\geq & \ell'\left(y_n, f_{\mathbf{w}_D}(\mathbf{x}_n)\right)\left(f_{\mathbf{w}_D}(\mathbf{x}_n) - f_{\mathbf{w}_{D'}}(\mathbf{x}_n)\right) - \ell'\left(y_n', f_{\mathbf{w}_{D'}}(\mathbf{x}_n')\right)\left(f_{\mathbf{w}_D}(\mathbf{x}_n') - f_{\mathbf{w}_{D'}}(\mathbf{x}_n')\right) \quad ,
\end{aligned}$$

where the second equality follows from $\partial_{\mathbf{w}} \ell\left(y, f_{\mathbf{w}}(\mathbf{x})\right) = \ell'\left(y, f_{\mathbf{w}}(\mathbf{x})\right) \phi(\mathbf{x})$, where $\ell'(y, \hat{y}) = \partial_{\hat{y}} \ell(y, \hat{y})$, and $\mathbf{x}_i' = \mathbf{x}_i$ and $y_i' = y_i$ for each $i \in [n-1]$. The inequality follows from the convexity of ℓ in its second argument.[2] Combined with the existence of non-positive $r \in \tilde{R}(\mathbf{w}_D)$ this yields that there exists

$$g \in \ell'\left(y_n', f_{\mathbf{w}_{D'}}(\mathbf{x}_n')\right)\left(f_{\mathbf{w}_D}(\mathbf{x}_n') - f_{\mathbf{w}_{D'}}(\mathbf{x}_n')\right) - \ell'\left(y_n, f_{\mathbf{w}_D}(\mathbf{x}_n)\right)\left(f_{\mathbf{w}_D}(\mathbf{x}_n) - f_{\mathbf{w}_{D'}}(\mathbf{x}_n)\right)$$

such that

$$\begin{aligned}
0 & \geq \frac{n}{C} r \\
& \geq g + \frac{n}{2C} \|\mathbf{w}_D - \mathbf{w}_{D'}\|_2^2 \ .
\end{aligned}$$

And since $|g| \leq 2L \left\| f_{\mathbf{w}_D} - f_{\mathbf{w}_{D'}} \right\|_\infty$ by the Lipschitz continuity of ℓ, this in turn implies

$$\frac{n}{2C} \|\mathbf{w}_D - \mathbf{w}_{D'}\|_2^2 \leq 2L \left\| f_{\mathbf{w}_D} - f_{\mathbf{w}_{D'}} \right\|_\infty \ . \tag{4.3}$$

Now by the reproducing property and Cauchy-Schwartz inequality we can upper bound the classifier difference's infinity norm by the Euclidean norm on the weight vectors: for each $\mathbf{x}$

$$\begin{aligned}
\left| f_{\mathbf{w}_D}(\mathbf{x}) - f_{\mathbf{w}_{D'}}(\mathbf{x}) \right| & = |\langle \phi(\mathbf{x}), \mathbf{w}_D - \mathbf{w}_{D'} \rangle| \\
& \leq \|\phi(\mathbf{x})\|_2 \, \|\mathbf{w}_D - \mathbf{w}_{D'}\|_2 \\
& = \sqrt{k(\mathbf{x}, \mathbf{x})} \, \|\mathbf{w}_D - \mathbf{w}_{D'}\|_2 \\
& \leq \kappa \, \|\mathbf{w}_D - \mathbf{w}_{D'}\|_2 \ .
\end{aligned}$$

Combining this with Inequality (4.3) yields $\|\mathbf{w}_D - \mathbf{w}_{D'}\|_2 \leq 4LC\kappa/n$ as claimed. The L_1-based sensitivity then follows from $\|\mathbf{w}\|_1 \leq \sqrt{F}\|\mathbf{w}\|_2$ for all $\mathbf{w} \in \mathbb{R}^F$. □

With the weight vector's sensitivity in hand, differential privacy follows immediately from the proof technique established by Dwork et al. (2006).

[2] Namely for convex f and any $a, b \in \mathbb{R}$, $(g_a - g_b)(a - b) \geq 0$ for all $g_a \in \partial f(a)$ and all $g_b \in \partial f(b)$.

Theorem 31 (Privacy of PrivateSVM-Finite)**.** *For any $\beta > 0$, database D of size n, $C > 0$, loss function $\ell(y, \hat{y})$ that is convex and L-Lipschitz in $\hat{y}$, and finite F-dimensional feature map with kernel $k(\mathbf{x}, \mathbf{x}) \leq \kappa^2$ for all $\mathbf{x} \in \mathbb{R}^d$,* PrivateSVM-Finite *run on D with loss ℓ, kernel k, noise parameter $\lambda \geq 4LC\kappa\sqrt{F}/(\beta n)$ and regularization parameter C guarantees β-differential privacy.*

This first main result establishes differential privacy of the new PrivateSVM-Finite algorithm. The more "private" the data, the more noise must be added. The more entries in the database, the less noise is needed to achieve the same level of privacy. Since the noise vector $\boldsymbol{\mu}$ has exponential tails, standard tail bound inequalities quickly lead to (ϵ, δ)-usefulness for PrivateSVM-Finite.

Theorem 32 (Utility of PrivateSVM-Finite)**.** *Consider any $C > 0$, $n > 1$, database D of n entries, arbitrary convex loss ℓ, and finite F-dimensional feature mapping ϕ with kernel k and $|\phi(\mathbf{x})_i| \leq \Phi$ for all $\mathbf{x} \in \mathcal{M}$ and $i \in [F]$ for some $\Phi > 0$ and $\mathcal{M} \subseteq \mathbb{R}^d$. For any $\epsilon > 0$ and $\delta \in (0, 1)$,* PrivateSVM-Finite *run on D with loss ℓ, kernel k, noise parameter $0 < \lambda \leq \frac{\epsilon}{2\Phi\left(F \log_e 2 + \log_e \frac{1}{\delta}\right)}$, and regularization parameter C, is (ϵ, δ)-useful with respect to the* SVM *under the $\|\cdot\|_{\infty;\mathcal{M}}$-norm.*

Proof. Our goal is to compare the SVM and PrivateSVM-Finite classifications of any point $\mathbf{x} \in \mathcal{M}$:

$$\begin{aligned} \left| f_{\hat{M}(D)}(\mathbf{x}) - f_{M(D)}(\mathbf{x}) \right| &= |\langle \hat{\mathbf{w}}, \phi(\mathbf{x}) \rangle - \langle \tilde{\mathbf{w}}, \phi(\mathbf{x}) \rangle| \\ &= |\langle \boldsymbol{\mu}, \phi(\mathbf{x}) \rangle| \\ &\leq \|\boldsymbol{\mu}\|_1 \, \|\phi(\mathbf{x})\|_\infty \\ &\leq \Phi \, \|\boldsymbol{\mu}\|_1 \quad . \end{aligned}$$

The absolute value of a zero mean Laplace random variable with scale parameter λ is exponentially distributed with scale λ^{-1}. Moreover the sum of q i.i.d. exponential random variables has Erlang q-distribution with the same scale parameter.[3] Thus we have, for Erlang F-distributed random variable X and any $t > 0$,

$$\begin{aligned} \forall \mathbf{x} \in \mathcal{M}, \left| f_{\hat{M}(D)}(\mathbf{x}) - f_{M(D)}(\mathbf{x}) \right| &\leq \Phi \, X \\ \Rightarrow \quad \forall \epsilon > 0, \ \Pr\left(\left\| f_{\hat{M}(D)} - f_{M(D)} \right\|_{\infty;\mathcal{M}} > \epsilon \right) &\leq \Pr\left(X > \epsilon/\Phi \right) \\ &\leq \frac{\mathbb{E}\left[e^{tX} \right]}{e^{\epsilon t/\Phi}} \quad . \end{aligned} \tag{4.4}$$

Here we have employed the standard Chernoff tail bound technique using Markov's inequality. The numerator of (4.4), the moment generating function of the Erlang F-distribution

[3] The Erlang q-distribution has density $\frac{x^{q-1}\exp(-x/\lambda)}{\lambda^q (q-1)!}$, CDF $1 - e^{-x/\lambda} \sum_{j=0}^{q-1} \frac{(x/\lambda)^j}{j!}$, expectation $q\lambda$, variance $q\lambda^2$.

with parameter λ, is $(1-\lambda t)^{-F}$ for all $t < \lambda^{-1}$. Together with the choice of $t = (2\lambda)^{-1}$, this gives

$$\begin{aligned}\Pr\left(\left\|f_{\hat{M}(D)} - f_{M(D)}\right\|_{\infty;\mathcal{M}} > \epsilon\right) &\leq (1-\lambda t)^{-F} e^{-\epsilon t/\Phi} \\ &= 2^F e^{-\epsilon/(2\lambda\Phi)} \\ &= \exp\left(F\log_e 2 - \frac{\epsilon}{2\lambda\Phi}\right) .\end{aligned}$$

And provided that $\lambda \leq \epsilon/\left(2\Phi\left(F\log_e 2 + \log_e \frac{1}{\delta}\right)\right)$ this probability is bounded by δ. □

Our second main result establishes that PrivateSVM-Finite is not only differentially private, but that it releases a classifier that is similar to the SVM. Utility and privacy are competing properties, however, since utility demands that the noise not be too large.

4.4 Mechanism for Translation-Invariant Kernels

Consider now the problem of privately learning in an RKHS $\mathcal{H}$ induced by an infinite dimensional feature mapping ϕ. As a mechanism's response must be finitely encodable, the primal parametrization seems less appealing as for PrivateSVM-Finite. It is natural to look to the SVM's dual solution as a starting point: the Representer Theorem (Kimeldorf and Wahba, 1971) states that the optimizing $f^\star \in \mathcal{H}$ must be in the span of the data—a finite-dimensional subspace. While the coordinates in this subspace—the $\alpha_i^\star$ dual variables—could be perturbed in the usual way to guarantee differential privacy, the subspace's basis—the data—are also needed to parametrize $f^\star$. To side-step this apparent stumbling block, we take another approach by approximating $\mathcal{H}$ with a random RKHS $\hat{\mathcal{H}}$ induced by a random finite-dimensional map $\hat{\phi}$. This then allows us to respond with a finite primal parametrization. Algorithm 12 summarizes the PrivateSVM mechanism.

As noted recently by Rahimi and Recht (2008), the Fourier transform p of the g function of a continuous positive-definite translation-invariant kernel is a non-negative measure (Rudin, 1994). Rahimi and Recht (2008) exploit this fact to construct a random finite-dimensional RKHS $\hat{\mathcal{H}}$ by drawing $\hat{d}$ vectors from p. These vectors $\boldsymbol{\rho}_1, \ldots, \boldsymbol{\rho}_{\hat{d}}$ define the following random $2\hat{d}$-dimensional feature map

$$\hat{\phi}(\cdot) = \hat{d}^{-1/2}\left[\cos\left(\langle\boldsymbol{\rho}_1, \cdot\rangle\right), \sin\left(\langle\boldsymbol{\rho}_1, \cdot\rangle\right), \ldots, \cos\left(\langle\boldsymbol{\rho}_{\hat{d}}, \cdot\rangle\right), \sin\left(\langle\boldsymbol{\rho}_{\hat{d}}, \cdot\rangle\right)\right]^T . \tag{4.5}$$

Inner-products in the random feature space approximate $k(\cdot,\cdot)$ uniformly, and to arbitrary precision for sufficiently large parameter $\hat{d}$, as restated in Lemma 37. We denote the inner-product in the random feature space by $\hat{k}$. Rahimi and Recht (2008) applied this approximation to large-scale learning. For large-scale learning, good approximations can be found for $\hat{d} \ll n$. Table 4.1 presents three important translation-invariant kernels and their transformations. Here regularized ERM is performed in $\hat{\mathcal{H}}$, not to avoid complexity in n, but to provide a direct finite representation $\tilde{\mathbf{w}}$ of the primal solution in the case of infinite dimensional feature spaces. After performing regularized ERM in $\hat{\mathcal{H}}$, appropriate Laplace noise is added to the primal solution $\tilde{\mathbf{w}}$ to guarantee differential privacy as before.

Algorithm 12 PRIVATESVM

Inputs: database $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ with $\mathbf{x}_i \in \mathbb{R}^d$, $y_i \in \{-1, 1\}$; translation-invariant kernel $k(\mathbf{x}, \mathbf{y}) = g(\mathbf{x} - \mathbf{y})$ with Fourier transform $p(\boldsymbol{\omega}) = 2^{-1} \int e^{-j\langle \boldsymbol{\omega}, \mathbf{x} \rangle} g(\mathbf{x})\, d\mathbf{x}$; convex loss function ℓ; parameters $\lambda, C > 0$ and $\hat{d} \in \mathbb{N}$.

1. $\boldsymbol{\rho}_1, \ldots, \boldsymbol{\rho}_{\hat{d}} \leftarrow$Draw i.i.d. sample of $\hat{d}$ vectors in $\mathbb{R}^d$ from p;
2. $\hat{\boldsymbol{\alpha}} \leftarrow$ Run Algorithm 10 on D with parameter C, kernel $\hat{k}$ induced by map (4.5), and loss ℓ;
3. $\tilde{\mathbf{w}} \leftarrow \sum_{i=1}^n y_i \hat{\alpha}_i \hat{\phi}(\mathbf{x}_i)$ where $\hat{\phi}$ is defined in Equation (4.5);
4. $\boldsymbol{\mu} \leftarrow$ Draw i.i.d. sample of $2\hat{d}$ scalars from Laplace $(0, \lambda)$; and
5. Return $\hat{\mathbf{w}} = \tilde{\mathbf{w}} + \boldsymbol{\mu}$ and $\boldsymbol{\rho}_1, \ldots, \boldsymbol{\rho}_{\hat{d}}$

PRIVATESVM is computationally efficient. Algorithm 12 takes $O(\hat{d})$ time to compute each entry of the kernel matrix, or a total time of $O(\hat{d}n^2)$ on top of running dual SVM in the random feature space which is worst-case $O(n_s^3)$ for the analytic solution (where $n_s \leq n$ is the number of support vectors), and faster using numerical methods such as chunking (Burges, 1998). To achieve (ϵ, δ)-usefulness wrt the hinge-loss SVM $\hat{d}$ must be taken to be $O\left(\frac{d}{\epsilon^4}\left(\log\frac{1}{\delta} + \log\frac{1}{\epsilon}\right)\right)$ (*cf.* Corollary 39). By comparison it takes $O(dn^2)$ to construct the kernel matrix for any translation-invariant kernel.

As with the SVM and PRIVATESVM-FINITE, the response of Algorithm 12 can be used to make classifications on future test points by constructing the classifier $\hat{f}^\star(\cdot) = f_{\hat{\mathbf{w}}}(\cdot) = \langle \hat{\mathbf{w}}, \hat{\phi}(\cdot) \rangle$. Unlike the previous mechanisms, however, PRIVATESVM must include a parametrization of feature map $\hat{\phi}$—the sample $\{\boldsymbol{\rho}_i\}_{i=1}^{\hat{d}}$—in its response. Of PRIVATESVM's total response, only $\hat{\mathbf{w}}$ depends on database D. The $\boldsymbol{\rho}_i$ are data-independent vectors drawn from the transform p of the kernel, which we assume to be known by the adversary (to wit the adversary knows the mechanism itself, including k). Thus to establish differential privacy we need only consider the data-dependent weight vector, fortunately we can build on the case of PRIVATESVM-FINITE.

Corollary 33 (Privacy of PRIVATESVM)**.** *For any $\beta > 0$, database D of size n, $C > 0$, $\hat{d} \in \mathbb{N}$, loss function $\ell(y, \hat{y})$ that is convex and L-Lipschitz in $\hat{y}$, and translation-invariant kernel k,* PRIVATESVM *run on D with loss ℓ, kernel k, noise parameter $\lambda \geq 2^{2.5} LC\sqrt{\hat{d}}/(\beta n)$, approximation parameter $\hat{d}$, and regularization parameter C guarantees β-differential privacy.*

Proof. The result follows immediately from Theorem 31 since $\tilde{\mathbf{w}}$ is the primal solution of SVM with kernel $\hat{k}$, the response vector $\hat{\mathbf{w}} = \tilde{\mathbf{w}} + \boldsymbol{\mu}$ for i.i.d. Laplace $\boldsymbol{\mu}$, and $\hat{k}(\mathbf{x}, \mathbf{x}) = 1$ for all $\mathbf{x} \in \mathbb{R}^D$. □

This result is surprising, in that PRIVATESVM is able to guarantee privacy for regularized ERM over a function class of infinite VC-dimension, where the obvious way to return

the learned classifier (responding with the dual variables and feature mapping) reveals all the entries corresponding to the support vectors, completely.

Like PrivateSVM-Finite, PrivateSVM is useful with respect to the SVM. If we denote the function parametrized by intermediate weight vector $\tilde{\mathbf{w}}$ by $\tilde{f}$, then the same argument for the utility of PrivateSVM-Finite establishes the high-probability proximity of $\tilde{f}$ and $f^\star$.

Lemma 34. *Consider a run of Algorithms 10 and 12 with $\hat{d} \in \mathbb{N}$, $C > 0$, convex loss and translation-invariant kernel. Denote by $\hat{f}^\star$ and $\tilde{f}$ the classifiers parametrized by weight vectors $\hat{\mathbf{w}}$ and $\tilde{\mathbf{w}}$ respectively, where these vectors are related by $\hat{\mathbf{w}} = \tilde{\mathbf{w}} + \boldsymbol{\mu}$ with $\boldsymbol{\mu} \overset{iid}{\sim}$ Laplace$(0, \lambda)$ in Algorithm 12. For any $\epsilon > 0$ and $\delta \in (0, 1)$, if $0 < \lambda \leq \min\left\{ \frac{\epsilon}{2^4 \log_e 2\sqrt{\hat{d}}}, \frac{\epsilon\sqrt{\hat{d}}}{8 \log_e \frac{2}{\delta}} \right\}$ then $\Pr\left(\left\| \hat{f}^\star - \tilde{f} \right\|_\infty \leq \frac{\epsilon}{2} \right) \geq 1 - \frac{\delta}{2}$.*

Proof. As in the proof of Theorem 32 we can use the Chernoff trick to show that, for Erlang $2\hat{d}$-distributed random variable X, the choice of $t = (2\lambda)^{-1}$,and for any $\epsilon > 0$

$$\begin{aligned}
\Pr\left(\left\| \hat{f}^\star - \tilde{f} \right\|_\infty > \epsilon/2 \right) &\leq \frac{\mathbb{E}\left[e^{tX}\right]}{e^{\epsilon t\sqrt{\hat{d}}/2}} \\
&\leq (1 - \lambda t)^{-2\hat{d}}\, e^{-\epsilon t\sqrt{\hat{d}}/2} \\
&= 2^{2\hat{d}} e^{-\epsilon\sqrt{\hat{d}}/(4\lambda)} \\
&= \exp\left(\hat{d} \log_e 4 - \epsilon\sqrt{\hat{d}}/(4\lambda) \right) \ .
\end{aligned}$$

Provided that $\lambda \leq \epsilon / \left(2^4 \log_e 2\sqrt{\hat{d}} \right)$ this is bounded by $\exp\left(-\epsilon\sqrt{\hat{d}}/(8\lambda) \right)$. Moreover if $\lambda \leq \epsilon\sqrt{\hat{d}} / \left(8 \log_e \frac{2}{\delta} \right)$, then the claim follows. □

To show a similar result for $f^\star$ and $\tilde{f}$, we exploit smoothness of regularized ERM with respect to small changes in the RKHS itself. To the best of our knowledge, this kind of stability to the feature mapping has not been used before. We begin with a technical lemma that we will use to exploit the convexity of the regularized empirical risk functional.

Lemma 35. *Let R be a functional on Hilbert space $\mathcal{H}$ satisfying $R[f] \geq R[f^\star] + \frac{a}{2}\|f - f^\star\|_{\mathcal{H}}^2$ for some $a > 0$, $f^\star \in \mathcal{H}$ and all $f \in \mathcal{H}$. Then $R[f] \leq R[f^\star] + \epsilon$ implies $\|f - f^\star\|_{\hat{\mathcal{H}}} \leq \sqrt{\frac{2\epsilon}{a}}$, for all $\epsilon > 0$, $f \in \mathcal{H}$.*

Proof. By assumption and the antecedent

$$\begin{aligned}
\|f - f^\star\|_{\hat{\mathcal{H}}}^2 &\leq \frac{2}{a}\left(R[f] - R[f^\star] \right) \\
&\leq \frac{2}{a}\left(R[f^\star] + \epsilon - R[f^\star] \right) \\
&= 2\epsilon/a \ .
\end{aligned}$$

Taking square roots of both sides yields the consequent. □

Provided that the kernel functions k and $\hat{k}$ are uniformly close, the next lemma exploits insensitivity of regularized ERM to perturbations of the feature mapping to show that $f^\star$ and $\tilde{f}$ are pointwise close.

Lemma 36. *Let $\mathcal{H}$ be an RKHS with translation-invariant kernel k, and let $\hat{\mathcal{H}}$ be the random RKHS corresponding to feature map (4.5) induced by k. Let C be a positive scalar and loss $\ell(y, \hat{y})$ be convex and L-Lipschitz continuous in $\hat{y}$. Consider the regularized empirical risk minimizers in each RKHS*

$$\begin{aligned} f^\star &\in \arg\min_{f \in \mathcal{H}} \frac{C}{n} \sum_{i=1}^{n} \ell(y_i, f(\mathbf{x}_i)) + \frac{1}{2}\|f\|_{\mathcal{H}}^2 \ . \\ g^\star &\in \arg\min_{g \in \hat{\mathcal{H}}} \frac{C}{n} \sum_{i=1}^{n} \ell(y_i, g(\mathbf{x}_i)) + \frac{1}{2}\|g\|_{\hat{\mathcal{H}}}^2 \ . \end{aligned}$$

Let $\mathcal{M} \subseteq \mathbb{R}^d$ be any set containing $\mathbf{x}_1, \ldots, \mathbf{x}_n$. For any $\epsilon > 0$, if the dual variables from both optimizations have L_1-norms bounded by some $\Lambda > 0$ and $\left\|k - \hat{k}\right\|_{\infty;\mathcal{M}} \leq \min\left\{1, \frac{\epsilon^2}{2^2\left(\Lambda + 2\sqrt{(CL + \Lambda/2)\Lambda}\right)^2}\right\}$ then $\|f^\star - g^\star\|_{\infty;\mathcal{M}} \leq \epsilon/2$.

Proof. Denote the empirical risk functional $R_{\text{emp}}[f] = n^{-1}\sum_{i=1}^{n} \ell\left(y_i, f(\mathbf{x}_i)\right)$ and the regularized empirical risk functional $R_{\text{reg}}[f] = C\, R_{\text{emp}}[f] + \|f\|^2/2$, for the appropriate RKHS norm (either $\mathcal{H}$ or $\hat{\mathcal{H}}$). Let $f^\star$ denote the regularized empirical risk minimizer in $\mathcal{H}$, given by parameter vector $\boldsymbol{\alpha}^\star$, and let $g^\star$ denote the regularized empirical risk minimizer in $\hat{\mathcal{H}}$ given by parameter vector $\boldsymbol{\beta}^\star$. Let $g_{\alpha^\star} = \sum_{i=1}^{n} \alpha_i^\star y_i \hat{\phi}(\mathbf{x}_i)$ and $f_{\boldsymbol{\beta}^\star} = \sum_{i=1}^{n} \beta_i^\star y_i \phi(\mathbf{x}_i)$ denote the images of $f^\star$ and $g^\star$ under the natural mapping between the spans of the data in RKHS's $\hat{\mathcal{H}}$ and $\mathcal{H}$ respectively. We will first show that these four functions have arbitrarily close regularized empirical risk in their respective RKHS, and then that this implies uniform proximity of the functions themselves. First observe that for any $g \in \hat{\mathcal{H}}$

$$\begin{aligned} R_{\text{reg}}^{\hat{\mathcal{H}}}[g] &= C\, R_{\text{emp}}[g] + \frac{1}{2}\|g\|_{\hat{\mathcal{H}}}^2 \\ &\geq C\langle \partial_g R_{\text{emp}}[g^\star], g - g^\star\rangle_{\hat{\mathcal{H}}} + C\, R_{\text{emp}}[g^\star] + \frac{1}{2}\|g\|_{\hat{\mathcal{H}}}^2 \\ &= \langle \partial_g R_{\text{reg}}^{\hat{\mathcal{H}}}[g^\star], g - g^\star\rangle_{\hat{\mathcal{H}}} - \langle g^\star, g - g^\star\rangle_{\hat{\mathcal{H}}} + C\, R_{\text{emp}}[g^\star] + \frac{1}{2}\|g\|_{\hat{\mathcal{H}}}^2 \ . \end{aligned}$$

The inequality follows from the convexity of $R_{\text{emp}}[\cdot]$ and holds for all elements of the subdifferential $\partial_g R_{\text{emp}}[g^\star]$. The subsequent equality holds by $\partial_g R_{\text{reg}}^{\hat{\mathcal{H}}}[g] = C\, \partial_g R_{\text{emp}}[g] + g$. Now

since $\mathbf{0} \in \partial_g R^{\hat{\mathcal{H}}}_{\text{reg}}[g^\star]$, it follows that

$$\begin{aligned} R^{\hat{\mathcal{H}}}_{\text{reg}}[g] &\geq C\, R_{\text{emp}}[g^\star] + \frac{1}{2}\|g\|^2_{\hat{\mathcal{H}}} - \langle g^\star, g - g^\star\rangle_{\hat{\mathcal{H}}} \\ &= C\, R_{\text{emp}}[g^\star] + \frac{1}{2}\|g^\star\|^2_{\hat{\mathcal{H}}} + \frac{1}{2}\|g\|^2_{\hat{\mathcal{H}}} - \frac{1}{2}\|g^\star\|^2_{\hat{\mathcal{H}}} - \langle g^\star, g - g^\star\rangle_{\hat{\mathcal{H}}} \\ &= R^{\hat{\mathcal{H}}}_{\text{reg}}[g^\star] + \frac{1}{2}\|g\|^2_{\hat{\mathcal{H}}} - \langle g^\star, g\rangle_{\hat{\mathcal{H}}} + \frac{1}{2}\|g^\star\|^2_{\hat{\mathcal{H}}} \\ &= R^{\hat{\mathcal{H}}}_{\text{reg}}[g^\star] + \frac{1}{2}\|g - g^\star\|^2_{\hat{\mathcal{H}}} \ . \end{aligned}$$

With this, Lemma 35 states that for any $g \in \hat{\mathcal{H}}$ and $\epsilon' > 0$,

$$R^{\hat{\mathcal{H}}}_{\text{reg}}[g] \leq R^{\hat{\mathcal{H}}}_{\text{reg}}[g^\star] + \epsilon' \ \Rightarrow \ \|g - g^\star\|_{\hat{\mathcal{H}}} \leq \sqrt{2\epsilon'} \ . \tag{4.6}$$

Next we will show that the antecedent is true for $g = g_{\boldsymbol{\alpha}^\star}$. Conditioned on $\left\{\left\|k - \hat{k}\right\|_{\infty;\mathcal{M}} \leq \epsilon'\right\}$, for all $\mathbf{x} \in \mathcal{M}$

$$\begin{aligned} |f^\star(\mathbf{x}) - g_{\boldsymbol{\alpha}^\star}(\mathbf{x})| &= \left|\sum_{i=1}^n \alpha_i^\star y_i \left(k(\mathbf{x}_i, \mathbf{x}) - \hat{k}(\mathbf{x}_i, \mathbf{x})\right)\right| \\ &\leq \sum_{i=1}^n |\alpha_i^\star| \left|k(\mathbf{x}_i, \mathbf{x}) - \hat{k}(\mathbf{x}_i, \mathbf{x})\right| \\ &\leq \epsilon' \|\boldsymbol{\alpha}^\star\|_1 \\ &\leq \epsilon'\Lambda \ , \end{aligned} \tag{4.7}$$

by the bound on $\|\boldsymbol{\alpha}^\star\|_1$. This and the Lipschitz continuity of the loss leads to

$$\begin{aligned} \left|R^{\mathcal{H}}_{\text{reg}}[f^\star] - R^{\hat{\mathcal{H}}}_{\text{reg}}[g_{\boldsymbol{\alpha}^\star}]\right| &= \left|C\, R_{\text{emp}}[f^\star] - C\, R_{\text{emp}}[g_{\boldsymbol{\alpha}^\star}] + \frac{1}{2}\|f^\star\|^2_{\mathcal{H}} - \frac{1}{2}\|g_{\boldsymbol{\alpha}^\star}\|^2_{\hat{\mathcal{H}}}\right| \\ &\leq \frac{C}{n}\sum_{i=1}^n |\ell\left(y_i, f^\star(\mathbf{x}_i)\right) - \ell\left(y_i, g_{\boldsymbol{\alpha}^\star}(\mathbf{x}_i)\right)| + \frac{1}{2}\left|\boldsymbol{\alpha}^{\star\prime}\left(\mathbf{K} - \hat{\mathbf{K}}\right)\boldsymbol{\alpha}^\star\right| \\ &\leq \frac{C}{n}\sum_{i=1}^n L\, \|f^\star - g_{\boldsymbol{\alpha}^\star}\|_{\infty;\mathcal{M}} + \frac{1}{2}\left|\boldsymbol{\alpha}^{\star\prime}\left(\mathbf{K} - \hat{\mathbf{K}}\right)\boldsymbol{\alpha}^\star\right| \\ &\leq CL\, \|f^\star - g_{\boldsymbol{\alpha}^\star}\|_{\infty;\mathcal{M}} + \frac{1}{2}\, \|\boldsymbol{\alpha}^\star\|_1 \left\|\left(\mathbf{K} - \hat{\mathbf{K}}\right)\boldsymbol{\alpha}^\star\right\|_\infty \\ &\leq CL\, \|f^\star - g_{\boldsymbol{\alpha}^\star}\|_{\infty;\mathcal{M}} + \frac{1}{2}\, \|\boldsymbol{\alpha}^\star\|_1^2\, \epsilon' \\ &\leq CL\epsilon'\Lambda + \Lambda^2\epsilon'/2 \\ &= \left(CL + \frac{\Lambda}{2}\right)\Lambda\epsilon' \ . \end{aligned}$$

Similarly, $\left|R^{\hat{\mathcal{H}}}_{\text{reg}}[g^\star] - R^{\mathcal{H}}_{\text{reg}}[f_{\boldsymbol{\beta}^\star}]\right| \leq (CL + \Lambda/2)\Lambda\epsilon'$ by the same argument. And since $R^{\mathcal{H}}_{\text{reg}}[f_{\boldsymbol{\beta}^\star}] \geq R^{\mathcal{H}}_{\text{reg}}[f^\star]$ and $R^{\hat{\mathcal{H}}}_{\text{reg}}[g_{\boldsymbol{\alpha}^\star}] \geq R^{\hat{\mathcal{H}}}_{\text{reg}}[g^\star]$ we have proved that $R^{\hat{\mathcal{H}}}_{\text{reg}}[g_{\boldsymbol{\alpha}^\star}] \leq R^{\mathcal{H}}_{\text{reg}}[f^\star] + (CL + \Lambda/2)\Lambda\epsilon' \leq R^{\mathcal{H}}_{\text{reg}}[f_{\boldsymbol{\beta}^\star}] + (CL + \Lambda/2)\Lambda\epsilon' \leq R^{\hat{\mathcal{H}}}_{\text{reg}}[g^\star] + 2(CL + \Lambda/2)\Lambda\epsilon'$. And by implication (4.6),

$$\|g_{\boldsymbol{\alpha}^\star} - g^\star\|_{\hat{\mathcal{H}}} \leq 2\sqrt{\left(CL + \frac{\Lambda}{2}\right)\Lambda\epsilon'} \ . \tag{4.8}$$

Now $\hat{k}(\mathbf{x}, \mathbf{x}) = 1$ for each $\mathbf{x} \in \mathbb{R}^d$ implies

$$\begin{aligned} |g_{\boldsymbol{\alpha}^\star}(\mathbf{x}) - g^\star(\mathbf{x})| &= \left\langle g_{\boldsymbol{\alpha}^\star} - g^\star, \hat{k}(\mathbf{x}, \cdot)\right\rangle_{\hat{\mathcal{H}}} \\ &\leq \|g_{\boldsymbol{\alpha}^\star} - g^\star\|_{\hat{\mathcal{H}}} \sqrt{\hat{k}(\mathbf{x}, \mathbf{x})} \\ &= \|g_{\boldsymbol{\alpha}^\star} - g^\star\|_{\hat{\mathcal{H}}} \ , \end{aligned}$$

This combines with Inequality (4.8) to yield

$$\|g_{\boldsymbol{\alpha}^\star} - g^\star\|_{\infty;\mathcal{M}} \leq 2\sqrt{\left(CL + \frac{\Lambda}{2}\right)\Lambda\epsilon'} \ .$$

Together with Inequality (4.7) this finally implies that $\|f^\star - g^\star\|_{\infty;\mathcal{M}} \leq \epsilon'\Lambda + 2\sqrt{(CL + \Lambda/2)\,\Lambda\epsilon'}$, conditioned on event $A_{\epsilon'} = \left\{\left\|k - \hat{k}\right\|_\infty \leq \epsilon'\right\}$. For desired accuracy $\epsilon > 0$, conditioning on event $A_{\epsilon'}$ with $\epsilon' = \min\left\{\epsilon/\left[2\left(\Lambda + 2\sqrt{(CL + \Lambda/2)\,\Lambda}\right)\right], \epsilon^2/\left[2\left(\Lambda + 2\sqrt{(CL + \Lambda/2)\,\Lambda}\right)\right]^2\right\}$ yields bound $\|f^\star - g^\star\|_{\infty;\mathcal{M}} \leq \epsilon/2$: if $\epsilon' \leq 1$ then $\epsilon/2 \geq \sqrt{\epsilon'}\left(\Lambda + 2\sqrt{(CL + \Lambda/2)\,\Lambda}\right) \geq \epsilon'\Lambda + 2\sqrt{(CL + \Lambda/2)\,\Lambda\epsilon'}$ provided that $\epsilon' \leq \epsilon^2/\left[2\left(\Lambda + 2\sqrt{(CL + \Lambda/2)\,\Lambda}\right)\right]^2$. Otherwise if $\epsilon' > 1$ then we have $\epsilon/2 \geq \epsilon'\left(\Lambda + 2\sqrt{(CL + \Lambda/2)\,\Lambda}\right) \geq \epsilon'\Lambda + 2\sqrt{(CL + \Lambda/2)\,\Lambda\epsilon'}$ provided $\epsilon' \leq \epsilon/\left[2\left(\Lambda + 2\sqrt{(CL + \Lambda/2)\,\Lambda}\right)\right]$. Since for any $H > 0$, $\min\{H, H^2\} \geq \min\{1, H^2\}$, the result follows. □

We now recall the result due to Rahimi and Recht (2008) that establishes the non-asymptotic uniform convergence of the kernel functions required by the previous Lemma (*i.e.*, an upper bound on the probability of event $A_{\epsilon'}$).

Lemma 37 (Rahimi and Recht 2008, Claim 1)**.** *For any $\epsilon > 0$, $\delta \in (0, 1)$, translation-invariant kernel k and compact set $\mathcal{M} \subset \mathbb{R}^d$, if $\hat{d} \geq \frac{4(d+2)}{\epsilon^2}\log_e\left(\frac{2^8(\sigma_p \text{diam}(\mathcal{M}))^2}{\delta\epsilon^2}\right)$, then Algorithm 12's random feature mapping $\hat{\phi}$ defined in Equation (4.5) satisfies $\Pr\left(\left\|\hat{k} - k\right\|_\infty < \epsilon\right) \geq 1 - \delta$, where $\sigma_p^2 = \mathbb{E}\left[\langle\boldsymbol{\omega}, \boldsymbol{\omega}\rangle\right]$ is the second moment of the Fourier transform p of k's g function.*

Combining these ingredients establishes utility for PrivateSVM.

Theorem 38 (Utility of PrivateSVM)**.** *Consider any database D, compact set $\mathcal{M} \subset \mathbb{R}^d$ containing D, convex loss ℓ, translation-invariant kernel k, and scalars $C, \epsilon > 0$ and $\delta \in (0,1)$. Suppose the SVM with loss ℓ, kernel k and parameter C has dual variables with L_1-norm bounded by Λ. Then Algorithm 12 run on D with loss ℓ, kernel k, parameters $\hat{d} \geq \frac{4(d+2)}{\theta(\epsilon)} \log_e \left(\frac{2^9 (\sigma_p \mathrm{diam}(\mathcal{M}))^2}{\delta \theta(\epsilon)} \right)$ where $\theta(\epsilon) = \min \left\{ 1, \frac{\epsilon^4}{2^4 \left(\Lambda + 2\sqrt{(CL + \Lambda/2)\Lambda} \right)^4} \right\}$, $\lambda \leq \min \left\{ \frac{\epsilon}{2^4 \log_e 2\sqrt{\hat{d}}}, \frac{\epsilon \sqrt{\hat{d}}}{8 \log_e \frac{2}{\delta}} \right\}$ and C is (ϵ, δ)-useful with respect to Algorithm 10 run on D with loss ℓ, kernel k and parameter C, wrt the $\| \cdot \|_{\infty;\mathcal{M}}$-norm.*

Proof. Lemma's 36 and 34 combined via the triangle inequality, with Lemma 37, together establish the result as follows. Define A to be the conditioning event regarding the approximation of k by $\hat{k}$, denote the events in Lemma's 36 and 32 by B and C (beware we are overloading C with the regularization parameter; its meaning will be apparent from the context), and the target event in the theorem by D.

$$
\begin{aligned}
A &= \left\{ \left\| \hat{k} - k \right\|_{\infty;\mathcal{M}} < \min \left\{ 1, \frac{\epsilon^2}{2^2 \left(\Lambda + 2\sqrt{\left(CL + \frac{\Lambda}{2} \right) \Lambda} \right)^2} \right\} \right\} \\
B &= \left\{ \left\| f^\star - \tilde{f} \right\|_{\infty;\mathcal{M}} \leq \epsilon/2 \right\} \\
C &= \left\{ \left\| \hat{f}^\star - \tilde{f} \right\|_{\infty} \leq \epsilon/2 \right\} \\
D &= \left\{ \left\| f^\star - \hat{f}^\star \right\|_{\infty;\mathcal{M}} \leq \epsilon \right\}
\end{aligned}
$$

The claim is a bound on $\Pr(D)$. By the triangle inequality events B and C together imply D. Second note that event C is independent of A and B. Thus $\Pr(D \mid A) \geq \Pr(B \cap C \mid A) = \Pr(B \mid A) \Pr(C) \geq 1 \cdot (1 - \delta/2)$, for sufficiently small λ. Finally Lemma 37 bounds $\Pr(A)$ as follows: provided that $\hat{d} \geq 4(d+2) \log_e \left(2^9 \left(\sigma_p \mathrm{diam}(\mathcal{M}) \right)^2 / \left(\delta \theta(\epsilon) \right) \right) / \theta(\epsilon)$ where $\theta(\epsilon) = \min \left\{ 1, \epsilon^4 / \left[2 \left(\Lambda + 2\sqrt{(CL + \Lambda/2)\Lambda} \right) \right]^4 \right\}$ we have $\Pr(A) \geq 1 - \delta/2$. Together this yields $\Pr(D) = \Pr(D \mid A) \Pr(A) \geq (1 - \delta/2)^2 \geq 1 - \delta$. □

Again we see that utility and privacy place competing constraints on the level of noise λ. Next we will use these interactions to upper-bound the optimal differential privacy of the SVM.

4.5 Hinge-Loss and an Upper Bound on Optimal Differential Privacy

We begin by plugging hinge-loss $\ell(y,\hat{y}) = (1-y\hat{y})_+$ into the main results on privacy and utility of the previous section (similar computations can be done for PrivateSVM-Finite and other convex loss functions). The following is the dual formulation of hinge-loss SVM learning:

$$\begin{aligned} \max_{\boldsymbol{\alpha}\in\mathbb{R}^n} \quad & \sum_{i=1}^n \alpha_i - \frac{1}{2}\sum_{i=1}^n\sum_{j=1}^n \alpha_i\alpha_j y_i y_j k(\mathbf{x}_i,\mathbf{x}_j) \\ \text{s.t.} \quad & 0 \le \alpha_i \le \frac{C}{n}\ \forall i\in[n]\ . \end{aligned} \tag{4.9}$$

Corollary 39. *Consider any database D of size n, scalar $C>0$, and translation-invariant kernel k.*

i. For any $\beta>0$ and $\hat{d}\in\mathbb{N}$, PrivateSVM *run on D with hinge-loss, noise parameter $\lambda \ge \frac{2^{2.5}C\sqrt{\hat{d}}}{\beta n}$, approximation parameter $\hat{d}$, and regularization parameter C, guarantees β-differential privacy.*

ii. Moreover for any compact set $\mathcal{M}\subset\mathbb{R}^d$ containing D, and scalars $\epsilon>0$ and $\delta\in(0,1)$, PrivateSVM *run on D with hinge-loss, kernel k, noise parameter $\lambda \le \min\left\{\frac{\epsilon}{2^4\log_e 2\sqrt{\hat{d}}}, \frac{\epsilon\sqrt{\hat{d}}}{8\log_e\frac{2}{\delta}}\right\}$, approximation parameter $\hat{d}\ge \frac{4(d+2)}{\theta(\epsilon)}\log_e\left(\frac{2^9(\sigma_p \mathrm{diam}(\mathcal{M}))^2}{\delta\theta(\epsilon)}\right)$ with $\theta(\epsilon)=\min\left\{1,\frac{\epsilon^4}{2^{12}C^4}\right\}$, and regularization parameter C, is (ϵ,δ)-useful wrt hinge-loss* SVM *run on D with kernel k, and parameter C.*

Proof. The first result follows from Theorem 31 and the fact that hinge-loss is convex and 1-Lipschitz on $\mathbb{R}$: *i.e.*, $\partial_{\hat{y}}\ell = \mathbf{1}[1\ge y\hat{y}] \le 1$. The second result follows almost immediately from Theorem 38. For hinge-loss we have that feasible α_i's are bounded by C/n (and so $\Lambda = C$) by the dual's box constraints and that $L=1$, implying we take $\theta(\epsilon) = \min\left\{1, \frac{\epsilon^4}{2^4C^4\left(1+\sqrt{6}\right)^4}\right\}$. This is bounded by the stated $\theta(\epsilon)$. □

Combining the competing requirements on noise level λ upper-bounds optimal differential privacy of hinge-loss SVM.

Theorem 40. *The optimal differential privacy for hinge-loss* SVM *learning on translation-invariant kernel k is bounded by $\beta(\epsilon,\delta,C,n,\ell,k) = O\left(\frac{1}{\epsilon^3 n}\sqrt{\log\frac{1}{\delta\epsilon}}\left(\log\frac{1}{\epsilon}+\log^2\frac{1}{\delta\epsilon}\right)\right)$.*

Proof. Consider hinge-loss in Corollary 39. Privacy places a lower bound of $\beta \ge 2^{2.5}C\sqrt{\hat{d}}/(\lambda n)$ for any chosen λ, which we can convert to a lower bound on β in terms of ϵ and δ as follows. For small ϵ, we have $\theta(\epsilon)=\epsilon^4 2^{-12}C^{-4}$ and so to achieve (ϵ,δ)-usefulness we must take $\hat{d} = O\left(\frac{1}{\epsilon^4}\log_e\left(\frac{1}{\delta\epsilon^4}\right)\right)$. There are two cases for utility, if $\lambda = \epsilon/\left(2^4\log_e\left(2\sqrt{\hat{d}}\right)\right)$

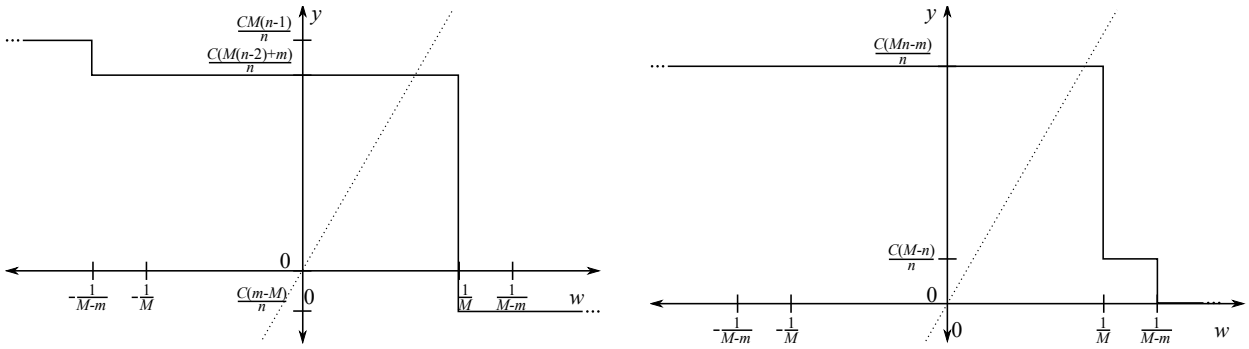


Figure 4.1: For each $i \in [2]$, the SVM's primal solution $w_i^\star$ on database D_i constructed in the proof of Lemma 41, corresponds to the crossing point of line $y = w$ with $y = w - \partial_w f_i(w)$. Database D_1 is shown on the left, database D_2 is shown on the right.

then $\beta = O\left(\frac{\sqrt{\hat{d}}\log_e\sqrt{\hat{d}}}{\epsilon n}\right) = O\left(\frac{1}{\epsilon^3 n}\sqrt{\log\frac{1}{\delta\epsilon}}\left(\log\frac{1}{\epsilon} + \log^2\frac{1}{\delta\epsilon}\right)\right)$. Otherwise we are in the second case, with $\lambda = \frac{\epsilon\sqrt{\hat{d}}}{8\log_e\frac{2}{\delta}}$ yielding $\beta = O\left(\frac{1}{\epsilon n}\log\frac{1}{\delta}\right)$ which is dominated by the first case as $\epsilon \downarrow 0$. □

A natural question arises from this discussion: given any mechanism that is (ϵ, δ)-useful with respect to hinge SVM, for how small a β can we possibly hope to guarantee β-differential privacy? In other words, what lower bounds exist for the optimal differential privacy for the SVM?

4.6 Lower Bounding Optimal Differential Privacy

We now present lower bounds on the level β of differential privacy achievable for any (ϵ, δ)-useful mechanism with respect to the hinge-loss SVM. We consider both mechanisms for linear kernels and mechanisms for RBF kernels.

4.6.1 Lower Bound for Linear Kernels

In this section we present a lower bound on the level of differential privacy for any mechanism approximating hinge-loss linear SVM with high accuracy. The first lemma corresponds to a kind of negative sensitivity result: for a particular pair of neighboring databases we show that the SVM is sensitive.

Lemma 41. *For any $C > 0$, $n > 1$ and $0 < \epsilon < \frac{\sqrt{C}}{2n}$, there exists a pair of neighboring databases D_1, D_2 on n entries, such that the functions $f_1^\star, f_2^\star$ parametrized by SVM run with parameter C, linear kernel, and hinge-loss on D_1, D_2 respectively, satisfy $\|f_1^\star - f_2^\star\|_\infty > 2\epsilon$.*

Proof. We construct the two databases on the line as follows. Let $0 < m < M$ be scalars to be chosen later. Both databases share negative examples $x_1 = \ldots = x_{\lfloor n/2 \rfloor} = -M$ and positive examples $x_{\lfloor n/2 \rfloor + 1} = \ldots = x_{n-1} = M$. Each database has $x_n = M - m$, with $y_n = -1$ for D_1 and $y_n = 1$ for D_2. In what follows we use subscripts to denote an example's parent database, so $(x_{i,j}, y_{i,j})$ is the j^{th} example from D_i. Consider the result of running primal SVM on each database

$$\begin{aligned} w_1^\star &= \arg\min_{w \in \mathbb{R}} \frac{1}{2} w^2 + \frac{C}{n} \sum_{i=1}^{n} (1 - y_{1,i} w x_{1,i})_+ \\ w_2^\star &= \arg\min_{w \in \mathbb{R}} \frac{1}{2} w^2 + \frac{C}{n} \sum_{i=1}^{n} (1 - y_{2,i} w x_{2,i})_+ \quad . \end{aligned}$$

Each optimization is strictly convex and unconstrained, so the optimizing $w_1^\star, w_2^\star$ are characterized by the first-order KKT conditions $0 \in \partial_w f_i(w)$ for f_i being the objective function for learning on D_i, and ∂_w denoting the subdifferential operator. Now for each $i \in [2]$

$$\partial_w f_i(w) = w - \frac{C}{n} \sum_{j=1}^{n} y_{i,j} x_{i,j} \tilde{\mathbf{1}} \left[1 - y_{i,j} w x_{i,j} \right] \quad ,$$

where

$$\tilde{\mathbf{1}}[x] = \begin{cases} \{0\} \ , & \text{if } x < 0 \\ [0,1] \ , & \text{if } x = 0 \\ \{1\} \ , & \text{if } x > 0 \end{cases}$$

is the subdifferential of $(x)_+$. Thus for each $i \in [2]$, $w_i^\star \in \frac{C}{n} \sum_{j=1}^{n} y_{i,j} x_{i,j} \tilde{\mathbf{1}} \left[1 - y_{i,j} w_i^\star x_{i,j} \right]$ which is equivalent to

$$\begin{aligned} w_1^\star &\in \frac{CM(n-1)}{n} \tilde{\mathbf{1}} \left[\frac{1}{M} - w_1^\star \right] + \frac{C(m-M)}{n} \tilde{\mathbf{1}} \left[w_1^\star - \frac{1}{m-M} \right] \\ w_2^\star &\in \frac{CM(n-1)}{n} \tilde{\mathbf{1}} \left[\frac{1}{M} - w_2^\star \right] + \frac{C(M-m)}{n} \tilde{\mathbf{1}} \left[\frac{1}{M-m} - w_2^\star \right] \quad . \end{aligned}$$

The RHSs of these conditions correspond to decreasing piecewise-constant functions, and the conditions are met when the corresponding functions intersect with the diagonal $y = x$ line, as shown in Figure 4.1. If $\frac{C(M(n-2)+m)}{n} < \frac{1}{M}$ then $w_1^\star = \frac{C(M(n-2)+m)}{n}$. And if $\frac{C(Mn-m)}{n} < \frac{1}{M}$ then $w_2^\star = \frac{C(Mn-m)}{n}$. So provided that $\frac{1}{M} > \frac{C(Mn-m)}{n} = \max\left\{ \frac{C(M(n-2)+m)}{n}, \frac{C(Mn-m)}{n} \right\}$, we have $|w_1^\star - w_2^\star| = \frac{2C}{n} |M - m|$. So taking $M = \frac{2n\epsilon}{C}$ and $m = \frac{n\epsilon}{C}$, this implies

$$\begin{aligned} \|f_1^\star - f_2^\star\|_\infty &\geq |f_1^\star(1) - f_2^\star(1)| \\ &= |w_1^\star - w_2^\star| \\ &= 2\epsilon \ , \end{aligned}$$

provided $\epsilon < \frac{\sqrt{C}}{2n}$. □

Next using a probabilistic method argument, we show that the negative sensitivity result leads to a lower bound.

Theorem 42 (Lower bound on optimal differential privacy for hinge-loss SVM)**.** *For any $C > 0$, $n > 1$, $\delta \in (0,1)$ and $\epsilon \in \left(0, \frac{\sqrt{C}}{2n}\right)$, the optimal differential privacy for the hinge-loss* SVM *with linear kernel is lower-bounded by $\log_e \frac{1-\delta}{\delta}$. In other words, for any $C, \beta > 0$ and $n > 1$ if a mechanism $\hat{M}$ is (ϵ, δ)-useful and β-differentially private then either $\epsilon \geq \frac{\sqrt{C}}{2n}$ or $\delta \geq \exp(-\beta)$.*

Proof. Consider (ϵ, δ)-useful mechanism $\hat{M}$ with respect to SVM learning mechanism M with parameter $C > 0$, hinge-loss and linear kernel on n training examples, where $\delta > 0$ and $\frac{\sqrt{C}}{2n} > \epsilon > 0$. By Lemma 41 there exists a pair of neighboring databases D_1, D_2 on n entries, such that $\|f_1^\star - f_2^\star\|_\infty > 2\epsilon$ where $f_i^\star = f_{M(D_i)}$ for each $i \in [2]$. Let $\hat{f}_i = f_{\hat{M}(D_i)}$ for each $i \in [2]$. Then by the utility of $\hat{M}$,

$$\Pr\left(\hat{f}_1 \in \mathcal{B}_\epsilon^\infty\left(f_1^\star\right)\right) \;\geq\; 1 - \delta\,, \tag{4.10}$$

$$\Pr\left(\hat{f}_2 \in \mathcal{B}_\epsilon^\infty\left(f_1^\star\right)\right) \;\leq\; \Pr\left(\hat{f}_2 \notin \mathcal{B}_\epsilon^\infty\left(f_2^\star\right)\right) \;<\; \delta\;. \tag{4.11}$$

Let $\hat{\mathcal{P}}_1$ and $\hat{\mathcal{P}}_2$ be the distributions of $\hat{M}(D_1)$ and $\hat{M}(D_2)$ respectively so that $\hat{\mathcal{P}}_i(t) = \Pr\left(\hat{M}(D_i) = t\right)$. Then by Inequalities (4.10) and (4.11)

$$\mathbb{E}_{T\sim\mathcal{P}_1}\left[\frac{d\mathcal{P}_2(T)}{d\mathcal{P}_1(T)}\,\middle|\, T \in \mathcal{B}_\epsilon^\infty\left(f_1^\star\right)\right] \;=\; \frac{\int_{\mathcal{B}_\epsilon^\infty(f_1^\star)} \frac{d\mathcal{P}_2(t)}{d\mathcal{P}_1(t)} d\mathcal{P}_1(t)}{\int_{\mathcal{B}_\epsilon^\infty(f_1^\star)} d\mathcal{P}_1(t)} \leq \frac{\delta}{1-\delta}\;.$$

Thus there exists a t such that $\log \frac{\Pr\left(\hat{M}(D_1)=t\right)}{\Pr\left(\hat{M}(D_2)=t\right)} \geq \log \frac{1-\delta}{\delta}$. □

4.6.2 Lower Bound for RBF Kernels

To lower bound the level β of differential privacy achievable for any (ϵ, δ)-useful mechanism with respect to an RBF hinge-loss SVM, we again begin with negative sensitivity result for the SVM. But now we can exploit the RBF kernel to construct a sequence of N pairwise neighboring databases whose images under SVM learning form an ϵ-packing. By using the RBF kernel with shrinking variance parameter, we can achieve this for any N.

Lemma 43. *For any $C > 0$, $n > C$, $0 < \epsilon < \frac{C}{4n}$, and $0 < \sigma < \sqrt{\frac{1}{2\log_e 2}}$ there exists a set of $N = \left\lfloor \frac{2}{\sigma}\sqrt{\frac{2}{\log_e 2}} \right\rfloor$ pairwise-neighboring databases $\{D_i\}_{i=1}^N$ on n examples, such that the functions $f_i^\star$ parametrized by hinge-loss* SVM *run on D_i with parameter C and RBF kernel with parameter σ, satisfy $\left\|f_i^\star - f_j^\star\right\|_\infty > 2\epsilon$ for each $i \neq j$.*

Proof. Construct $N > 1$ pairwise neighboring databases each on n examples in $\mathbb{R}^2$ as follows. Each database i has $n-1$ negative examples $\mathbf{x}_{i,1} = \ldots = \mathbf{x}_{i,n-1} = \mathbf{0}$, and database D_i has positive example $\mathbf{x}_{i,n} = (\cos\theta_i, \sin\theta_i)$ where $\theta_i = \frac{2\pi i}{N}$. Consider the result of running SVM with hinge-loss and RBF kernel on each D_i. For each database $k(\mathbf{x}_{i,s}, \mathbf{x}_{i,t}) = 1$ and $k(\mathbf{x}_{i,s}, \mathbf{x}_{i,n}) = \exp\left(-\frac{1}{2\sigma^2}\right) =: \gamma$ for all $s, t \in [n-1]$. Notice that the range space of γ is $(0, 1)$. Since the inner-products and labels are database-independent, the SVM dual variables are also database-independent. Each involves solving

$$\begin{aligned} \max_{\boldsymbol{\alpha}\in\mathbb{R}^n} \quad & \boldsymbol{\alpha}'\mathbf{1} - \frac{1}{2}\boldsymbol{\alpha}' \begin{pmatrix} 1 & -\gamma \\ -\gamma & 1 \end{pmatrix} \boldsymbol{\alpha} \\ \text{s.t.} \quad & \mathbf{0} \leq \boldsymbol{\alpha} \leq \frac{C}{n}\mathbf{1} \end{aligned}$$

By symmetry $\alpha_1^\star = \ldots = \alpha_{n-1}^\star$, so we can reduce this to the equivalent program on two variables:

$$\begin{aligned} \max_{\boldsymbol{\alpha}\in\mathbb{R}^2} \quad & \boldsymbol{\alpha}' \begin{pmatrix} n-1 \\ 1 \end{pmatrix} - \frac{1}{2}\boldsymbol{\alpha}' \begin{pmatrix} (n-1)^2 & -\gamma(n-1) \\ -\gamma(n-1) & 1 \end{pmatrix} \boldsymbol{\alpha} \\ \text{s.t.} \quad & \mathbf{0} \leq \boldsymbol{\alpha} \leq \frac{C}{n}\mathbf{1} \end{aligned}$$

Consider first the unconstrained program. In this case the necessary first-order KKT condition is that

$$\mathbf{0} \;=\; \begin{pmatrix} n-1 \\ 1 \end{pmatrix} - \begin{pmatrix} (n-1)^2 & -\gamma(n-1) \\ -\gamma(n-1) & 1 \end{pmatrix} \boldsymbol{\alpha}^\star \;.$$

This implies

$$\begin{aligned} \boldsymbol{\alpha}^\star &= \begin{pmatrix} (n-1)^2 & -\gamma(n-1) \\ -\gamma(n-1) & 1 \end{pmatrix}^{-1} \begin{pmatrix} n-1 \\ 1 \end{pmatrix} \\ &= \frac{1}{(n-1)^2(1-\gamma^2)} \begin{pmatrix} 1 & \gamma(n-1) \\ \gamma(n-1) & (n-1)^2 \end{pmatrix} \begin{pmatrix} n-1 \\ 1 \end{pmatrix} \\ &= \frac{1}{(n-1)^2(1-\gamma)(1+\gamma)} \begin{pmatrix} 1 & \gamma(n-1) \\ \gamma(n-1) & (n-1)^2 \end{pmatrix} \begin{pmatrix} n-1 \\ 1 \end{pmatrix} \\ &= \frac{1}{(n-1)^2(1-\gamma)(1+\gamma)} \begin{pmatrix} (n-1)(1+\gamma) \\ (n-1)^2(1+\gamma) \end{pmatrix} \\ &= \begin{pmatrix} \frac{1}{(n-1)(1-\gamma)} \\ \frac{1}{1-\gamma} \end{pmatrix} \;. \end{aligned}$$

Since this solution is strictly positive, it follows that at most two (upper) constraints can be active. Thus four cases are possible: the solution lies in the interior of the feasible set, or one or both upper box-constraints hold with equality. Noting that $\frac{1}{(n-1)(1-\gamma)} \leq \frac{1}{1-\gamma}$ it follows that $\boldsymbol{\alpha}^\star$ is feasible iff $\frac{1}{1-\gamma} \leq \frac{C}{n}$. This is equivalent to $C \geq \frac{1}{1-\gamma} n > n$, since $\gamma \in (0, 1)$. This corresponds to under-regularization.

If both constraints hold with equality we have $\boldsymbol{\alpha}^\star = \frac{C}{n}\mathbf{1}$, which is always feasible.

In the case where the first constraint holds with equality $\alpha_1^\star = \frac{C}{n}$, the second dual variable is found by optimizing

$$\begin{aligned}
\alpha_2^\star &= \max_{\alpha_2 \in \mathbb{R}} \boldsymbol{\alpha}' \begin{pmatrix} n-1 \\ 1 \end{pmatrix} - \frac{1}{2}\boldsymbol{\alpha}' \begin{pmatrix} (n-1)^2 & -\gamma(n-1) \\ -\gamma(n-1) & 1 \end{pmatrix} \boldsymbol{\alpha} \\
&= \max_{\alpha_2 \in \mathbb{R}} \frac{C(n-1)}{n} + \alpha_2 - \frac{1}{2}\left(\left(\frac{C(n-1)}{n} \right)^2 - 2\frac{C\gamma(n-1)}{n}\alpha_2 + \alpha_2^2 \right) \\
&= \max_{\alpha_2 \in \mathbb{R}} -\frac{1}{2}\alpha_2^2 + \alpha_2 \left(1 + \frac{C\gamma(n-1)}{n} \right) \ ,
\end{aligned}$$

implying $\alpha_2^\star = 1 + C\gamma\frac{n-1}{n}$. This solution is feasible provided $1 + C\gamma\frac{n-1}{n} \leq \frac{C}{n}$ iff $n \leq \frac{C(1+\gamma)}{1+C\gamma}$. Again this corresponds to under-regularization.

Finally in the case where the second constraint holds with equality $\alpha_2^\star = \frac{C}{n}$, the first dual is found by optimizing

$$\begin{aligned}
\alpha_2^\star &= \max_{\alpha_1 \in \mathbb{R}} \boldsymbol{\alpha}' \begin{pmatrix} n-1 \\ 1 \end{pmatrix} - \frac{1}{2}\boldsymbol{\alpha}' \begin{pmatrix} (n-1)^2 & -\gamma(n-1) \\ -\gamma(n-1) & 1 \end{pmatrix} \boldsymbol{\alpha} \\
&= \max_{\alpha_1 \in \mathbb{R}} (n-1)\alpha_1 + \frac{C}{n} - \frac{1}{2}\left((n-1)^2\alpha_1^2 - 2C\gamma\frac{n-1}{n}\alpha_1 + \frac{C^2}{n^2} \right) \\
&= \max_{\alpha_2 \in \mathbb{R}} -\frac{1}{2}(n-1)^2\alpha_1^2 + \alpha_1 \left(1 + \frac{C\gamma}{n} \right) \ ,
\end{aligned}$$

implying $\alpha_1^\star = \frac{1+\frac{C\gamma}{n}}{(n-1)^2}$. This is feasible provided $\frac{1+\frac{C\gamma}{n}}{(n-1)^2} \leq \frac{C}{n}$. Passing back to the program on n variables, by the invariance of the duals to the database, for any pair D_i, D_j

$$\begin{aligned}
|f_i(\mathbf{x}_{i,n}) - f_j(\mathbf{x}_{i,n})| &= \alpha_n^\star (1 - k(\mathbf{x}_{i,n}, \mathbf{x}_{j,n})) \\
&\geq \alpha_n^\star \left(1 - \max_{q \neq i} k(\mathbf{x}_{i,n}, \mathbf{x}_{q,n}) \right) \ .
\end{aligned}$$

Now a simple argument shows that this maximum is equal to $\gamma^4 \exp\left(\sin^2\frac{\pi}{N}\right)$ for all i. The maximum objective is optimized when $|q-i| = 1$. In this case $|\theta_i - \theta_q| = \frac{2\pi}{N}$. The norm $\|\mathbf{x}_{i,n} - \mathbf{x}_{q,n}\| = 2\sin\frac{|\theta_i - \theta_q|}{2} = 2\sin\frac{\pi}{N}$ by basic geometry. Thus $k(\mathbf{x}_{i,n}, \mathbf{x}_{q,n}) = \exp\left(-\frac{\|\mathbf{x}_{i,n} - \mathbf{x}_{q,n}\|^2}{2\sigma^2}\right) = \exp\left(-\frac{2}{\sigma^2}\sin^2\frac{\pi}{N}\right) = \gamma^4 \exp\left(\sin^2\frac{\pi}{N}\right)$ as claimed. Notice that $N \geq 2$ so the second term is in $(1, e]$, while the first term is in $(0, 1)$. In summary we have shown that for any $i \neq j$

$$|f_i(\mathbf{x}_{i,n}) - f_j(\mathbf{x}_{i,n})| \geq \left(1 - \exp\left(-\frac{2}{\sigma^2}\sin^2\frac{\pi}{N} \right) \right) \alpha_n^\star \ .$$

Assume $\gamma < \frac{1}{2}$. If $n > C$ then $n > \frac{C}{2} > (1-\gamma)C$ in which implies case 1 is infeasible. Similarly since $C\gamma\frac{n-1}{n} > 0$, $n > C$ implies $1 + C\gamma\frac{n-1}{n} > 1 > \frac{C}{n}$ which implies case 3 is

infeasible. Thus provided that $\gamma < \frac{1}{2}$ and $n > C$ we have that either case 2 or case 4 must hold. In both cases $\alpha_n^\star = \frac{C}{n}$ giving

$$|f_i(\mathbf{x}_{i,n}) - f_j(\mathbf{x}_{i,n})| \;\geq\; \left(1 - \exp\left(-\frac{2}{\sigma^2}\sin^2\frac{\pi}{N}\right)\right)\frac{C}{n} \;.$$

Provided that $\sigma \leq \sqrt{\frac{2}{\log 2}}\sin\frac{\pi}{N}$ we have $\left(1-\exp\left(-\frac{2}{\sigma^2}\sin^2\frac{\pi}{N}\right)\right)\frac{C}{n} \geq \left(1-\frac{1}{2}\right)\frac{C}{n} = \frac{C}{2n}$. Now for small x we can take the linear approximation $\sin x \geq \frac{x}{\pi/2}$ for $x \in [0, \pi/2]$. If $N \geq 2$ then $\sin\frac{\pi}{N} \geq \frac{2}{N}$. Thus in this case we can take $\sigma \leq \sqrt{\frac{2}{\log 2}}\frac{2}{N}$ to imply $|f_i(\mathbf{x}_{i,n}) - f_j(\mathbf{x}_{i,n})| \geq \frac{C}{2n}$. This bound on σ in turn implies the following bound on γ: $\gamma = \exp\left(-\frac{1}{2\sigma^2}\right) \leq \exp\left(-\frac{N^2\log_e 2}{2^4}\right)$. Thus taking $N > 4$, in conjunction with $\sigma \leq \sqrt{\frac{2}{\log 2}}\frac{2}{N}$ implies $\gamma \leq \frac{1}{2}$. Rather than selecting N which bounds σ, we can choose N in terms of σ. $\sigma \leq \sqrt{\frac{2}{\log 2}}\frac{2}{N}$ is implied by $N = \frac{2}{\sigma}\sqrt{\frac{2}{\log_e 2}}$. So for small σ we can construct more databases leading to the desired separation. Finally, $N > 4$ implies that we must constrain $\sigma < \sqrt{\frac{1}{2\log_e 2}}$.

In summary, if $n > C$ and $\sigma < \sqrt{\frac{1}{2\log_e 2}}$ then $|f_i(\mathbf{x}_{i,n}) - f_j(\mathbf{x}_{i,n})| \geq \frac{C}{2n}$ for each $i \neq j \in [N]$ where $N = \left\lfloor \frac{2}{\sigma}\sqrt{\frac{2}{\log_e 2}} \right\rfloor$. Moreover if $\epsilon \leq \frac{C}{4n}$ then for any $i \neq j$ this implies $\|f_i - f_j\|_\infty \geq 2\epsilon$ as claimed. □

We again use a similar argument as in the linear kernel section above, to derive the lower bound on differential privacy.

Theorem 44 (Lower bound on optimal differential privacy for hinge-loss)**.** *For $C > 0$, $n > C$, $\delta \in (0,1)$, $\epsilon \in \left(0, \frac{C}{4n}\right)$, and $\sigma < \sqrt{\frac{1}{2\log_e 2}}$ the optimal differential privacy for the hinge* SVM *with RBF kernel having parameter σ is lower-bounded by $\log_e \frac{(1-\delta)(N-1)}{\delta}$, where $N = \left\lfloor \frac{2}{\sigma}\sqrt{\frac{2}{\log_e 2}} \right\rfloor$. That is, under these conditions, all mechanisms that are (ϵ, δ)-useful wrt hinge* SVM *with RBF kernel for any σ do not achieve differential privacy at any level.*

Proof. Consider (ϵ, δ)-useful mechanism $\hat{M}$ with respect to hinge SVM learning mechanism M with parameter $C > 0$ and RBF kernel with parameter $0 < \sigma < \sqrt{\frac{1}{2\log_e 2}}$ on n training examples, where $\delta > 0$ and $\frac{C}{4n} > \epsilon > 0$. Let $N = \left\lfloor \frac{2}{\sigma}\sqrt{\frac{2}{\log_e 2}} \right\rfloor > 4$. By Lemma 43 there exist pairwise neighboring databases $D_1, \ldots, D_N$ of n entries, such that $\{f_i^\star\}_{i=1}^N$ is an ϵ-packing wrt the L_∞-norm, where $f_i^\star = f_{M(D_i)}$. So by the utility of $\hat{M}$, for each $i \in [N]$

$$\Pr\left(\hat{f}_i \in \mathcal{B}_\epsilon^\infty(f_i^\star)\right) \;\geq\; 1-\delta \;, \tag{4.12}$$

$$\sum_{j\neq 1}\Pr\left(\hat{f}_1 \in \mathcal{B}_\epsilon^\infty(f_j^\star)\right) \;\leq\; \Pr\left(\hat{f}_1 \notin \mathcal{B}_\epsilon^\infty(f_1^\star)\right) \;<\; \delta \;,$$

$$\Rightarrow\;\; \exists j \neq 1,\; \Pr\left(\hat{f}_1 \in \mathcal{B}_\epsilon^\infty(f_j^\star)\right) \;<\; \frac{\delta}{N-1} \;. \tag{4.13}$$

Let $\hat{\mathcal{P}}_1$ and $\hat{\mathcal{P}}_j$ be the distributions of $\hat{M}(D_1)$ and $\hat{M}(D_j)$ respectively so that for each, $\hat{\mathcal{P}}_i(t) = \Pr\left(\hat{M}(D_i) = t\right)$. Then by Inequalities (4.12) and (4.13)

$$\mathbb{E}_{T\sim\mathcal{P}_j}\left[\frac{d\mathcal{P}_1(T)}{d\mathcal{P}_j(T)}\,\middle|\, T \in \mathcal{B}_\epsilon^\infty\left(f_j^\star\right)\right] \;=\; \frac{\int_{\mathcal{B}_\epsilon^\infty(f_j^\star)} \frac{d\mathcal{P}_1(t)}{d\mathcal{P}_j(t)} d\mathcal{P}_j(t)}{\int_{\mathcal{B}_\epsilon^\infty(f_j^\star)} d\mathcal{P}_j(t)} \;\leq\; \frac{\delta}{(1-\delta)(N-1)}\ .$$

Thus there exists a t such that $\log \frac{\Pr\left(\hat{M}(D_j)=t\right)}{\Pr\left(\hat{M}(D_1)=t\right)} \geq \log \frac{(1-\delta)(N-1)}{\delta}$. □

Note that $n > C$ is a weak condition, since C should grow like $\sqrt{n}$ for universal consistency. Also note that this negative result is consistent with our upper bound on optimal differential privacy: σ affects σ_p, increasing the upper bounds as $\sigma \downarrow 0$.

4.7 Summary

In this chapter we present a pair of new mechanisms for private SVM learning. In each case we establish differential privacy via the algorithmic stability of regularized empirical risk minimization. To achieve utility under infinite-dimensional feature mappings, we perform regularized ERM in a random Reproducing Kernel Hilbert Space whose kernel approximates the target RKHS kernel. This trick, borrowed from large-scale learning, permits the mechanism to privately respond with a finite representation of a maximum-margin hyperplane classifier. We then establish the high-probability, pointwise similarity between the resulting function and the SVM classifier through a new smoothness result of regularized ERM with respect to perturbations of the RKHS. The bounds on differential privacy and utility combine to upper bound the optimal differential privacy of SVM learning for hinge-loss. This quantity is the optimal level of privacy among all mechanisms that are (ϵ, δ)-useful with respect to the hinge-loss SVM. Finally, we derive a lower bound on this quantity which establishes that any mechanism that is too accurate with respect to the hinge SVM with RBF kernel, with any non-trivial probability, cannot be β-differentially private for small β. The lower bounds explicitly depend on the variance of the RBF kernel.

Part II

Applications of Machine Learning in Computer Security

Chapter 5

Learning-Based Reactive Security

What's important is to understand the delineation between what's considered "acceptable" and "unacceptable" spending. The goal is to prevent spending on reactive security "firefighting".
– John N. Stewart, VP (Chief Security Officer), Cisco Systems

Despite the conventional wisdom that proactive security is superior to reactive security, this chapter aims to show that reactive security can be competitive with proactive security as long as the reactive defender learns from past attacks instead of myopically overreacting to the last attack. A proposed game-theoretic model follows common practice in the security literature by making worst-case assumptions about the attacker: we grant the attacker complete knowledge of the defender's strategy and do not require the attacker to act rationally. In this model, we bound the competitive ratio between a reactive defense algorithm (which is inspired by online learning theory) and the best fixed proactive defense. Additionally, we show that, unlike proactive defenses, this reactive strategy is robust to a lack of information about the attacker's incentives and knowledge.

The learning-based risk management strategy developed in this chapter faces an attacker that can manipulate both the training and test data in an attempt to maximize her profit or multiplicative return on investment—Targeted Causative and Exploratory attacks in the language of the taxonomy overviewed in Section 1.2.2. Our worst-case analysis, which grants the attacker complete control over the data and complete knowledge of the learner, shows that relative to all fixed proactive defenders, the reactive strategy asymptotically performs well.

5.1 Introduction

Many enterprises employ a Chief Information Security Officer (CISO) to manage the enterprise's information security risks. Typically, an enterprise has many more security vulnerabilities than it can realistically repair. Instead of declaring the enterprise "insecure" until every last vulnerability is plugged, CISOs typically perform a cost-benefit analysis

to identify which risks to address, but what constitutes an effective CISO strategy? The conventional wisdom (Kark et al., 2009; Pironti, 2005) is that CISOs ought to adopt a "forward-looking" *proactive* approach to mitigating security risk by examining the enterprise for vulnerabilities that might be exploited in the future. Advocates of proactive risk management often equate reactive security with myopic bug-chasing and consider it ineffective. We establish sufficient conditions for when reacting *strategically* to attacks is as effective in discouraging attackers.

We study the efficacy of reactive strategies in an economic model of the CISO's security cost-benefit trade-offs. Unlike previously proposed economic models of security (see Section 5.1.1), we do not assume the attacker acts according to a fixed probability distribution. Instead, we consider a game-theoretic model with a strategic attacker who responds to the defender's strategy. As is standard in the security literature, we make worst-case assumptions about the attacker. For example, we grant the attacker complete knowledge of the defender's strategy and do not require the attacker to act rationally. Further, we make conservative assumptions about the reactive defender's knowledge and do not assume the defender knows all the vulnerabilities in the system or the attacker's incentives. However, we do assume that the defender can observe the attacker's past actions, for example via an intrusion detection system or user metrics (Beard, 2008).

In our model, we find that two properties are sufficient for a reactive strategy to perform as well as the best proactive strategies. First, *no single attack is catastrophic*, meaning the defender can survive a number of attacks. This is consistent with situations where intrusions (that, say, steal credit card numbers) are regrettable but not business-ending. Second, *the defender's budget is* liquid, meaning the defender can re-allocate resources without penalty. For example, a CISO can reassign members of the security team from managing firewall rules to improving database access controls at relatively low switching costs.

Because our model abstracts many vulnerabilities into a single graph edge, we view the act of defense as increasing the attacker's *cost* for mounting an attack instead of preventing the attack (*e.g.*, by patching a single bug). By making this assumption, we choose not to study the tactical patch-by-patch interaction of the attacker and defender. Instead, we model enterprise security at a more abstract level appropriate for the CISO. For example, the CISO might allocate a portion of his or her budget to engage a consultancy, such as WhiteHat or iSEC Partners, to find and fix cross-site scripting in a particular web application or to require that employees use SecurID tokens during authentication. We make the technical assumption that attacker costs are linearly dependent on defense investments locally. This assumption does not reflect patch-by-patch interaction, which would be better represented by a step function (with the step placed at the cost to deploy the patch). Instead, this assumption reflects the CISO's higher-level viewpoint where the staircase of summed step functions fades into a slope.

We evaluate the defender's strategy by measuring the attacker's cumulative return-on-investment, the *return-on-attack* (ROA), which has been proposed previously (Cremonini, 2005). By studying this metric, we focus on defenders who seek to "cut off the attacker's oxygen," that is to reduce the attacker's incentives for attacking the enterprise. We do not distinguish between "successful" and "unsuccessful" attacks. Instead, we compare the

payoff the attacker receives from his or her nefarious deeds with the cost of performing said deeds. We imagine that sufficiently disincentivized attackers will seek alternatives, such as attacking a different organization or starting a legitimate business.

In our main result, we show sufficient conditions for a learning-based reactive strategy to be competitive with the best fixed proactive defense in the sense that the competitive ratio between the reactive ROA and the proactive ROA is at most $1+\epsilon$, for all $\epsilon > 0$, provided the game lasts sufficiently many rounds (at least $\Omega(1/\epsilon)$). To prove our theorems, we draw on techniques from the online learning literature. We extend these techniques to the case where the learner does not know all the game matrix rows *a priori*, letting us analyze situations where the defender does not know all the vulnerabilities in advance. Although our main results are in a graph-based model with a single attacker, our results generalize to a model based on Horn clauses with multiple attackers, corresponding to hypergraph-based models. Our results are also robust to switching from ROA to attacker profit and to allowing the proactive defender to revise the defense allocation a fixed number of times.

Although myopic bug chasing is most likely an ineffective reactive strategy, we find that in some situations a *strategic* reactive strategy is as effective as the optimal fixed proactive defense. In fact, we find that the natural strategy of gradually reinforcing attacked edges by shifting budget from unattacked edges "learns" the attacker's incentives and constructs an effective defense. Such a strategic reactive strategy is both easier to implement than a proactive strategy—because it does not presume that the defender knows the attacker's intent and capabilities—and is less wasteful than a proactive strategy because the defender does not expend budget on attacks that do not actually occur. Based on our results, we encourage CISOs to question the assumption that proactive risk management is inherently superior to reactive risk management.

Chapter Organization. The remainder of this section relates related work. Section 5.2 formalizes our model. Section 5.3 shows that perimeter defense and defense-in-depth arise naturally in our model. Section 5.4 presents our main results of the chapter bounding the competitive ratio of reactive versus proactive defense strategies. Section 5.5 outlines scenarios in which reactive security out-performs proactive security. Section 5.6 generalizes our results to Horn clauses and multiple attackers. Section 5.7 concludes the chapter with a short summary of the main contributions.

5.1.1 Related Work

Anderson (2001) and Varian (2000) informally discuss (via anecdotes) how the design of information security must take incentives into account. August and Tunca (2006) compare various ways to incentivize users to patch their systems in a setting where the users are more susceptible to attacks if their neighbors do not patch.

Gordon and Loeb (2002) and Hausken (2006) analyze the costs and benefits of security in an economic model (with non-strategic attackers) where the probability of a successful exploit is a function of the defense investment. They use this model to compute the optimal level of investment. Varian (2001) studies various (single-shot) security games and identifies

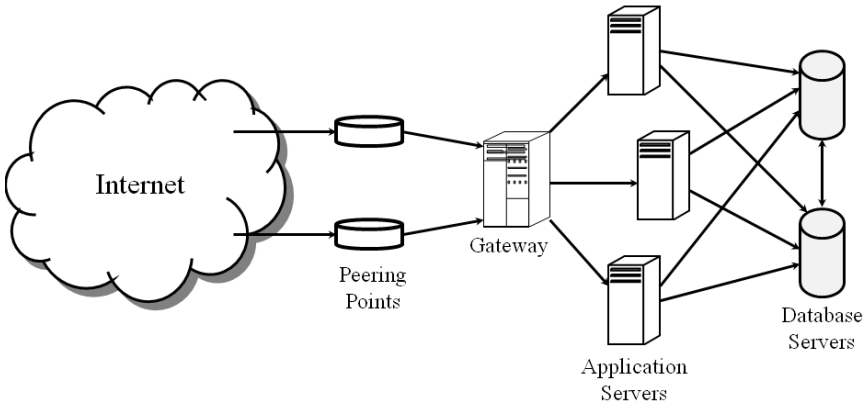


Figure 5.1: An attack graph representing an enterprise data center.

how much agents invest in security at equilibrium. Grossklags et al. (2008) extends this model by letting agents self-insure.

Miura-Ko et al. (2008) study externalities that appear due to users having the same password across various websites and discuss pareto-improving security investments. Miura-Ko and Bambos (2007) rank vulnerabilities according to a random-attacker model. Skybox and RedSeal offer practical systems that help enterprises prioritize vulnerabilities based on a random-attacker model. Kumar et al. (2008) investigate optimal security architectures for a multi-division enterprise, taking into account losses due to lack of availability and confidentiality. None of the above papers explicitly model a truly adversarial attacker.

Fultz and Grossklags (2009) generalizes (Grossklags et al., 2008) by modeling attackers explicitly. Cavusoglu et al. (2008) highlight the importance of using a game-theoretic model over a decision theoretic model due to the presence of adversarial attackers. However, these models look at idealized settings that are not generically applicable. Lye and Wing (2002) study the Nash equilibrium of a single-shot game between an attacker and a defender that models a particular enterprise security scenario. Arguably this model is most similar to ours in terms of abstraction level. However, calculating the Nash equilibrium requires detailed knowledge of the adversary's incentives, which as discussed in the introduction, might not be readily available to the defender. Moreover, their game contains multiple equilibria, weakening their prescriptions.

5.2 Formal Model

In this section, we present a game-theoretic model of attack and defense. Unlike traditional bug-level attack graphs, our model is meant to capture a managerial perspective on enterprise security. The model is somewhat general in the sense that attack graphs can represent a number of concrete situations, including a network (see Figure 5.1), components in a complex software system (Fisher, 2008), or an Internet Fraud "Battlefield" (Friedberg, 2007).

5.2.1 System

We model a system using a directed graph (V, E), which defines the game between an attacker and a defender. Each vertex $v \in V$ in the graph represents a state of the system. Each edge $e \in E$ represents a state transition the attacker can induce. For example, a vertex might represent whether a particular machine in a network has been compromised by an attacker. An edge from one machine to another might represent that an attacker who has compromised the first machine might be able to compromise the second machine because the two are connected by a network. Alternatively, the vertices might represent different components in a software system. An edge might represent that an attacker sending input to the first component can send input to the second.

In attacking the system, the attacker selects a path in the graph that begins with a designated *start vertex* s. Our results hold in more general models (*e.g.*, based on Horn clauses), but we defer discussing such generalizations until Section 5.6. We think of the attack as driving the system through the series of state transitions indicated by the edges included in the path. In the networking example in Figure 5.1, an attacker might first compromise a front-end server and then leverage the server's connectivity to the back-end database server to steal credit card numbers from the database.

Incentives and Rewards. Attackers respond to incentives. For example, attackers compromise machines and form botnets because they make money from spam (Kanich et al., 2008) or rent the botnet to others (Warner, 2004). Other attackers steal credit card numbers because credit card numbers have monetary value (Franklin et al., 2007). We model the attacker's incentives by attaching a non-negative *reward* to each vertex. These rewards are the utility the attacker derives from driving the system into the state represented by the vertex. For example, compromising the database server might have a sizable reward because the database server contains easily monetizable credit card numbers. We assume the start vertex has zero reward, forcing the attacker to undertake some action before earning utility. Whenever the attacker mounts an attack, the attacker receives a *payoff* equal to the sum of the rewards of the vertices visited in the attack path: $\text{payoff}(a) = \sum_{v \in a} \text{reward}(a)$. In the example from Figure 5.1, if an attacker compromises both a front-end server and the database server, the attacker receives both rewards.

Attack Surface and Cost. The defender has a fixed *defense budget* $B > 0$, which the defender can divide among the edges in the graph according to a *defense allocation* d: for all $e \in E$, $d(e) \geq 0$ and $\sum_{e \in E} d(e) \leq B$.

The defender's allocation of budget to various edges corresponds to the decisions made by the Chief Information Security Officer (CISO) about where to allocate the enterprise's security resources. For example, the CISO might allocate organizational headcount to fuzzing enterprise web applications for XSS vulnerabilities. These kinds of investments are continuous in the sense that the CISO can allocate 1/4 of a full-time employee to worrying about XSS. We denote the set of feasible allocations of budget B on edge set E by $\mathcal{D}_{B,E}$.

By defending an edge, the defender makes it more difficult for the attacker to use that edge in an attack. Each unit of budget the defender allocates to an edge raises the cost

that the attacker must pay to use that edge in an attack. Each edge has an *attack surface* (Howard, 2004) w that represents the difficulty in defending against that state transition. For example, a server that runs both Apache and Sendmail has a larger attack surface than one that runs only Apache because defending the first server is more difficult than the second. Formally, the attacker must pay the following *cost* to traverse the edge: $\text{cost}(a, d) = \sum_{e \in a} d(e)/w(e)$. Allocating defense budget to an edge does not "reduce" an edge's attack surface. For example, consider defending a hallway with bricks. The wider the hallway (the larger the attack surface), the more bricks (budget allocation) required to build a wall of a certain height (the cost to the attacker).

In this formulation, the function mapping the defender's budget allocation to attacker cost is linear, preventing the defender from ever fully defending an edge. Our use of a linear function reflects a level of abstraction more appropriate to a CISO who can never fully defend assets, which we justify by observing that the rate of vulnerability discovery in a particular piece of software is roughly constant (Rescorla, 2005). At a lower level of detail, we might replace this function with a step function, indicating that the defender can "patch" a vulnerability by allocating a threshold amount of budget.

5.2.2 Objective

To evaluate defense strategies, we measure the attacker's incentive for attacking using the *return-on-attack* (ROA) (Cremonini, 2005), which we define as follows:

$$\text{ROA}(a, d) \quad = \quad \frac{\text{payoff}(a)}{\text{cost}(a, d)}$$

We use this metric for evaluating defense strategy because we believe that if the defender lowers the ROA sufficiently, the attacker will be discouraged from attacking the system and will find other uses for his or her capital or industry. For example, the attacker might decide to attack another system. Analogous results hold if we quantify the attacker's incentives in terms of profit (*e.g.*, with $\text{profit}(a, d) = \text{payoff}(a) - \text{cost}(a, d)$), but we focus on ROA for simplicity.

A purely rational attacker will mount attacks that maximize ROA. However, a real attacker might not maximize ROA. For example, the attacker might not have complete knowledge of the system or its defense. We strengthen our results by considering all attacks, not just those that maximize ROA.

5.2.3 Proactive Security

We evaluate our learning-based reactive approach by comparing it against a *proactive* approach to risk management in which the defender carefully examines the system and constructs a defense in order to fend off future attacks. We strengthen this benchmark by providing the proactive defender complete knowledge about the system, but we require that the defender commit to a fixed strategy. To strengthen our results, we state our main result in terms of *all* such proactive defenders. In particular, this class of defenders includes the

rational proactive defender who employs a defense allocation that minimizes the maximum ROA the attacker can extract from the system: $\arg\min_d \max_a \mathrm{ROA}(a, d)$.

5.3 Case Studies

In this section, we describe instances of our model to build the reader's intuition. These examples illustrate that some familiar security concepts, including perimeter defense and defense in depth, arise naturally as optimal defenses in our model. These defenses can be constructed either by rational proactive attackers or converged to by a learning-based reactive defense.

5.3.1 Perimeter Defense

Consider a system in which the attacker's reward is non-zero at exactly one vertex, t. For example, in a medical system, the attacker's reward for obtaining electronic medical records might well dominate the value of other attack targets such as employees' vacation calendars. In such a system, a rational attacker will select the minimum-cost path from the start vertex s to the valuable vertex t. The optimal defense limits the attacker's ROA by maximizing the cost of the minimum s-t path. The algorithm for constructing this defense is straightforward (Chakrabarty et al., 2006):

1. Let C be the minimum weight s-t cut in (V, E, w).

2. Select the following defense:

$$d(e) = \begin{cases} Bw(e)/Z & \text{if } e \in C \\ 0 & \text{otherwise} \end{cases}, \quad \text{where } Z = \sum_{e \in C} w(e) \ .$$

Notice that this algorithm constructs a *perimeter defense*: the defender allocates the entire defense budget to a single cut in the graph. Essentially, the defender spreads the defense budget over the attack surface of the cut. By choosing the minimum-weight cut, the defender is choosing to defend the smallest attack surface that separates the start vertex from the target vertex. Real defenders use similar perimeter defenses, for example, when they install a firewall at the boundary between their organization and the Internet because the network's perimeter is much smaller than its interior.

5.3.2 Defense in Depth

Many experts in security practice recommend that defenders employ defense in depth. Defense in depth rises naturally in our model as an optimal defense for some systems. Consider, for example, the system depicted in Figure 5.2. This attack graph is a simplified version of the data center network depicted in Figure 5.1. Although the attacker receives the largest reward for compromising the back-end database server, the attacker also receives

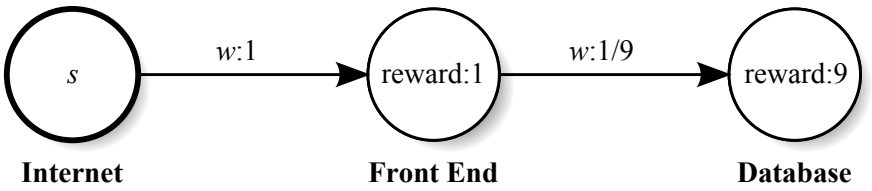


Figure 5.2: Attack graph representing a simplified data center network.

some reward for compromising the front-end web server. Moreover, the front-end web server has a larger attack surface than the back-end database server because the front-end server exposes a more complex interface (an entire enterprise web application), whereas the database server exposes only a simple SQL interface. Allocating defense budget to the left-most edge represents trying to protect sensitive database information with a complex web application firewall instead of database access control lists (*i.e.*, possible, but economically inefficient).

The optimal defense against a rational attacker is to allocate half of the defense budget to the left-most edge and half of the budget to the right-most edge, limiting the attacker to a ROA of unity. Shifting the entire budget to the right-most edge (*i.e.*, defending only the database) is disastrous because the attacker will simply attack the front-end at zero cost, achieving an unbounded ROA. Shifting the entire budget to the left-most edge is also problematic because the attacker will attack the database (achieving an ROA of 5).

5.4 Reactive Security

To analyze reactive security, we model the attacker and defender as playing an iterative game, alternating moves. First, the defender selects a defense, and then the attacker selects an attack. We present a learning-based reactive defense strategy that is oblivious to vertex rewards and to edges that have not yet been used in attacks. We prove a theorem bounding the competitive ratio between this reactive strategy and the best proactive defense via a series of reductions to results from the online learning theory literature. Other applications of this literature include managing stock portfolios (Ordentlich and Cover, 1998), playing zero-sum games (Freund and Schapire, 1999b), and boosting other machine learning heuristics (Freund and Schapire, 1999a). Although we provide a few technical extensions, our main contribution comes from applying results from online learning to risk management.

Repeated Game. We formalize the repeated game between the defender and the attacker as follows. In each round t from 1 to T:

1. The defender chooses defense allocation $d_t(e)$ over the edges $e \in E$.
2. The attacker chooses an attack path a_t in G.
3. The path a_t and attack surfaces $\{w(e) : e \in a_t\}$ are revealed to the defender.
4. The attacker pays $\text{cost}(a_t, d_t)$ and gains $\text{payoff}(a_t)$.

Algorithm 13 A reactive defense strategy for hidden edges.

- Initialize $E_0 = \emptyset$
- For each round $t \in \{2, ..., T\}$
  - Let $E_{t-1} = E_{t-2} \cup E(a_{t-1})$
  - For each $e \in E_{t-1}$, let

$$S_{t-1}(e) = \begin{cases} S_{t-2}(e) + M(e, a_{t-1}) & \text{if } e \in E_{t-2} \\ M(e, a_{t-1}) & \text{otherwise.} \end{cases}$$
$$\tilde{P}_t(e) = \beta_{t-1}^{S_{t-1}(e)}$$
$$P_t(e) = \frac{\tilde{P}_t(e)}{\sum_{e' \in E_t} \tilde{P}_t(e')} \ ,$$

  where $M(e, a) = -\mathbf{1}[e \in a] / w(e)$ is a matrix with $|E|$ rows and a column for each attack.

In each round, we let the attacker choose the attack path after the defender commits to the defense allocation because the defender's budget allocation is not a secret (in the sense of a cryptographic key). Following the "no security through obscurity" principle, we make the conservative assumption that the attacker can accurately determine the defender's budget allocation.

Defender Knowledge. Unlike proactive defenders, reactive defenders do not know all of the vulnerabilities that exist in the system in advance. (If defenders had complete knowledge of vulnerabilities, conferences such as Black Hat Briefings would serve little purpose.) Instead, we reveal an edge (and its attack surface) to the defender after the attacker uses the edge in an attack. For example, the defender might monitor the system and learn how the attacker attacked the system by doing a post-mortem analysis of intrusion logs. Formally, we define a *reactive defense strategy* to be a function from attack sequences $\{a_i\}$ and the subsystem induced by the edges contained in $\bigcup_i a_i$ to defense allocations such that $d(e) = 0$ if edge $e \notin \bigcup_i a_i$. Notice that this requires the defender's strategy to be oblivious to the system beyond the edges used by the attacker.

5.4.1 Algorithm

Algorithm 13 is a reactive defense strategy based on the multiplicative update learning algorithm (Cesa-Bianchi et al., 1997; Freund and Schapire, 1999b). The algorithm reinforces edges on the attack path multiplicatively, taking the attack surface into account by allocating more budget to easier-to-defend edges. When new edges are revealed, the algorithm reallocates budget uniformly from the already-revealed edges to the newly revealed edges. We

state the algorithm in terms of a normalized defense allocation $P_t(e) = d_t(e)/B$. Notice that this algorithm is oblivious to unattacked edges and the attacker's reward for visiting each vertex. An appropriate setting for the algorithm parameters $\beta_t \in [0, 1)$ will be described below.

The algorithm begins without any knowledge of the graph whatsoever, and so allocates no defense budget to the system. Upon the t^{th} attack on the system, the algorithm updates E_t to be the set of edges revealed up to this point, and updates $S_t(e)$ to be a weight count of the number of times e has been used in an attack thus far. For each edge that has ever been revealed, the defense allocation $P_{t+1}(e)$ is chosen to be $\beta_t^{S_t(e)}$ normalized to sum to unity over all edges $e \in E_t$. In this way, any edge attacked in round t will have its defense allocation reinforced.

The parameter β controls how aggressively the defender reallocates defense budget to recently attacked edges. If β is infinitesimal, the defender will move the entire defense budget to the edge on the most recent attack path with the smallest attack surface. If β is enormous, the defender will not be very agile and, instead, leave the defense budget in the initial allocation. For an appropriate value of β, the algorithm will converge to the optimal defense strategy. For instance, the min cut in the example from Section 5.3.1.

5.4.2 Main Theorems

To compare this reactive defense strategy to all proactive defense strategies, we use the notion of *regret* from online learning theory. The following is an additive regret bound relating the attacker's profit under reactive and proactive defense strategies.

Theorem 45. *The average attacker profit against Algorithm 13 converges to the average attacker profit against the best proactive defense. Formally, if defense allocations* $\{d_t\}_{t=1}^T$ *are output by Algorithm 13 with parameter sequence* $\beta_s = \left(1 + \sqrt{2 \log |E_s|/(s+1)}\right)^{-1}$ *on any system* $(V, E, w, \text{reward}, s)$ *revealed online and any attack sequence* $\{a_t\}_{t=1}^T$*, then*

$$\frac{1}{T}\sum_{t=1}^{T} \text{profit}(a_t, d_t) - \frac{1}{T}\sum_{t=1}^{T} \text{profit}(a_t, d^\star) \;\leq\; B\sqrt{\frac{\log |E|}{2T}} + \frac{B(\log |E| + \overline{w^{-1}})}{T} \;,$$

for all proactive defense strategies $d^\star \in \mathcal{D}_{B,E}$ *where* $\overline{w^{-1}} = |E|^{-1} \sum_{e \in E} w(e)^{-1}$*, the mean of the surface reciprocals.*

Remark 46. *We can interpret Theorem 45 as establishing sufficient conditions under which a reactive defense strategy is within an additive constant of the best proactive defense strategy. Instead of carefully analyzing the system to construct the best proactive defense, the defender need only react to attacks in a principled manner to achieve almost the same quality of defense in terms of attacker profit.*

Reactive defense strategies can also be competitive with proactive defense strategies when we consider an attacker motivated by return on attack (ROA). The ROA formulation is appealing because (unlike with profit) the objective function does not require measuring

attacker cost and defender budget in the same units. The next result considers the competitive ratio between the ROA for a reactive defense strategy and the ROA for the best proactive defense strategy.

Theorem 47. *The ROA against Algorithm 13 converges to the ROA against best proactive defense. Formally, consider the cumulative ROA:*

$$\text{ROA}\left(\{a_t\}_{t=1}^T, \{d_t\}_{t=1}^T\right) \;=\; \frac{\sum_{t=1}^T \text{payoff}(a_t)}{\sum_{t=1}^T \text{cost}(a_t, d_t)} \;.$$

(We abuse notation slightly and use singleton arguments to represent the corresponding constant sequence.) If defense allocations $\{d_t\}_{t=1}^T$ are output by Algorithm 13 with parameters $\beta_s = \left(1 + \sqrt{2\log|E_s|/(s+1)}\right)^{-1}$ on any system $(V, E, w, \text{reward}, s)$ revealed online, such that $|E| > 1$, and any attack sequence $\{a_t\}_{t=1}^T$, then for all $\alpha > 0$ and proactive defense strategies $d^\star \in \mathcal{D}_{B,E}$

$$\frac{\text{ROA}\left(\{a_t\}_{t=1}^T, \{d_t\}_{t=1}^T\right)}{\text{ROA}\left(\{a_t\}_{t=1}^T, d^\star\right)} \;\leq\; 1+\alpha \;,$$

provided T is sufficiently large.[1]

Remark 48. *Notice that the reactive defender can use* the same algorithm *regardless of whether the attacker is motivated by profit or by ROA. As discussed in Section 5.5 the optimal proactive defense is not similarly robust.*

5.4.3 Proofs of the Main Theorems

We now describe a series of reductions that establish the main results. First, we prove Theorem 45 in the simpler setting where the defender knows the entire graph. Second, we remove the hypothesis that the defender knows the edges is advance. Finally, we extend our results to ROA.

5.4.3.1 Bound on Profit: Known Edges Case

Suppose that the reactive defender is granted full knowledge of the system $(V, E, w, \text{reward}, s)$ from the outset. Specifically, the graph, attack surfaces, and rewards are all revealed to the defender prior to the first round. Algorithm 14 is a reactive defense strategy that makes use of this additional knowledge.

Lemma 49. *If defense allocations $\{d_t\}_{t=1}^T$ are output by Algorithm 14 with parameter $\beta = \left(1 + \sqrt{\frac{2\log|E|}{T}}\right)^{-1}$ on any system $(V, E, w, \text{reward}, s)$ and attack sequence $\{a_t\}_{t=1}^T$, then*

$$\frac{1}{T}\sum_{t=1}^T \text{profit}(a_t, d_t) - \frac{1}{T}\sum_{t=1}^T \text{profit}(a_t, d^\star) \;\leq\; B\sqrt{\frac{\log|E|}{2T}} + \frac{B\log|E|}{T} \;,$$

[1] To wit: $T \geq \left(\frac{13}{\sqrt{2}}\left(1+\alpha^{-1}\right)\left(\sum_{e\in\text{inc}(s)} w(e)\right)\right)^2 \log|E|$.

Algorithm 14 Reactive defense strategy for known edges using the multiplicative update algorithm.

- For each $e \in E$, initialize $P_1(e) = 1/|E|$.
- For each round $t \in \{2, \ldots, T\}$ and $e \in E$, let

$$P_t(e) = P_{t-1}(e) \cdot \beta^{M(e,a_{t-1})}/Z_t$$
$$\text{where} \qquad Z_t = \sum_{e' \in E} P_{t-1}(e)\beta^{M(e',a_{t-1})}$$

for all proactive defense strategies $d^\star \in \mathcal{D}_{B,E}$.

The lemma's proof is a reduction to the following regret bound from online learning (Freund and Schapire, 1999b, Corollary 4).

Theorem 50. *If the multiplicative update algorithm (Algorithm 14) is run with any game matrix M with elements in $[0,1]$, and parameter $\beta = \left(1 + \sqrt{2\log|E|/T}\right)^{-1}$, then*

$$\frac{1}{T}\sum_{t=1}^{T} M(P_t, a_t) - \min_{P^\star \geq 0 : \sum_{e\in E} P^\star(e) = 1} \left\{ \frac{1}{T}\sum_{t=1}^{T} M(P^\star, a_t) \right\} \;\leq\; \sqrt{\frac{\log|E|}{2T}} + \frac{\log|E|}{T} \quad .$$

Proof of Lemma 49. Due to the normalization by Z_t, the sequence of defense allocations $\{P_t\}_{t=1}^T$ output by Algorithm 14 is invariant to adding a constant to all elements of matrix M. Let M' be the matrix obtained by adding constant C to all entries of arbitrary game matrix M, and let sequences $\{P_t\}_{t=1}^T$ and $\{P'_t\}_{t=1}^T$ be obtained by running multiplicative update with matrix M and M' respectively. Then, for all $e \in E$ and $t \in [T-1]$,

$$\begin{aligned}
P'_{t+1}(e) &= \frac{P_1(e)\beta^{\sum_{i=1}^t M'(e,a_i)}}{\sum_{e'\in E} P_1(e')\beta^{\sum_{i=1}^t M'(e',a_i)}} \\
&= \frac{P_1(e)\beta^{\left(\sum_{i=1}^t M(e,a_i)\right)+tC}}{\sum_{e'\in E} P_1(e')\beta^{\left(\sum_{i=1}^t M(e',a_i)\right)+tC}} \\
&= \frac{P_1(e)\beta^{\sum_{i=1}^t M(e,a_i)}}{\sum_{e'\in E} P_1(e')\beta^{\sum_{i=1}^t M(e',a_i)}} \\
&= P_{t+1}(e) \quad .
\end{aligned}$$

In particular Algorithm 14 produces the same defense allocation sequence as if the game matrix elements are increased by one to

$$M'(e,a) = \begin{cases} 1 - 1/w(e) & \text{if } e \in a \quad . \\ 1 & \text{otherwise} \end{cases}$$

Because this new matrix has entries in $[0,1]$ we can apply Theorem 50 to prove for the original matrix M that

$$\frac{1}{T}\sum_{t=1}^{T} M(P_t, a_t) - \min_{P^\star \in \mathcal{D}_{1,E}} \left\{ \frac{1}{T}\sum_{t=1}^{T} M(P^\star, a_t) \right\} \;\leq\; \sqrt{\frac{\log|E|}{2T}} + \frac{\log|E|}{T} \;. \tag{5.1}$$

Now, by definition of the original game matrix,

$$\begin{aligned} M(P_t, a_t) &= \sum_{e \in E} -(P_t(e)/w(e)) \cdot \mathbf{1}\left[e \in a_t\right] \\ &= -\sum_{e \in a_t} P_t(e)/w(e) \\ &= -B^{-1} \sum_{e \in a_t} d_t(e)/w(e) \\ &= -B^{-1} \operatorname{cost}(a_t, d_t) \;. \end{aligned}$$

Thus Inequality (5.1) is equivalent to

$$\begin{aligned} & -\frac{1}{T}\sum_{t=1}^{T} B^{-1} \operatorname{cost}(a_t, d_t) - \min_{d^\star \in \mathcal{D}_{1,E}} \left\{ -\frac{1}{T}\sum_{t=1}^{T} B^{-1} \operatorname{cost}(a_t, d^\star) \right\} \\ \leq\; & \sqrt{\frac{\log|E|}{2T}} + \frac{\log|E|}{T} \;. \end{aligned}$$

Simple algebraic manipulation yields

$$\begin{aligned} & \frac{1}{T}\sum_{t=1}^{T} \operatorname{profit}(a_t, d_t) - \min_{d^\star \in \mathcal{D}_{B,E}} \left\{ \frac{1}{T}\sum_{t=1}^{T} \operatorname{profit}(a_t, d^\star) \right\} \\ =\; & \frac{1}{T}\sum_{t=1}^{T} \left(\operatorname{payoff}(a_t) - \operatorname{cost}(a_t, d_t)\right) - \min_{d^\star \in \mathcal{D}_{B,E}} \left\{ \frac{1}{T}\sum_{t=1}^{T} \left(\operatorname{payoff}(a_t) - \operatorname{cost}(a_t, d^\star)\right) \right\} \\ =\; & \frac{1}{T}\sum_{t=1}^{T} \left(-\operatorname{cost}(a_t, d_t)\right) - \min_{d^\star \in \mathcal{D}_{B,E}} \left\{ \frac{1}{T}\sum_{t=1}^{T} \left(-\operatorname{cost}(a_t, d^\star)\right) \right\} \\ \leq\; & B\sqrt{\frac{\log|E|}{2T}} + B\frac{\log|E|}{T} \;, \end{aligned}$$

completing the proof. □

5.4.3.2 Bound on Profit: Hidden Edges Case

The standard algorithms in online learning assume that the rows of the matrix are known in advance. Here, the edges are not known in advance and we must relax this assumption using a simulation argument, which is perhaps the least obvious part of the reduction.

The defense allocation chosen by Algorithm 13 at time t is precisely the same as the defense allocation that would have been chosen by Algorithm 14 had the defender run Algorithm 14 on the currently visible subgraph. The following lemma formalizes this equivalence. Note that Algorithm 13's parameter is reactive: it corresponds to Algorithm 14's parameter, but for the subgraph induced by the edges revealed so far. That is, β_t depends only on edges visible to the defender in round t, letting the defender actually run the algorithm in practice!

Lemma 51. *Consider arbitrary round $t \in [T]$. If Algorithms 13 and 14 are run with parameters $\beta_s = \left(1 + \sqrt{2 \log |E_s|/(s+1)}\right)^{-1}$ for $s \in [t]$ and parameter $\beta = \left(1 + \sqrt{2 \log |E_t|/(t+1)}\right)^{-1}$ respectively, with the latter run on the subgraph induced by E_t, then the defense allocations $P_{t+1}(e)$ output by the algorithms are identical for all $e \in E_t$.*

Proof. If $e \in E_t$ then $\tilde{P}_{t+1}(e) = \beta^{\sum_{i=1}^{t} M(e,a_i)}$ because $\beta_t = \beta$, and the round $t+1$ defense allocation of Algorithm 13 P_{t+1} is simply $\tilde{P}_{t+1}$ normalized to sum to unity over edge set E_t, which is exactly the defense allocation output by Algorithm 14. □

Armed with this correspondence, we show that Algorithm 13 is almost as effective as Algorithm 14. In other words, hiding unattacked edges from the defender does not cause much harm to the reactive defender's ability to disincentivize the attacker.

Lemma 52. *If defense allocations $\{d_{1,t}\}_{t=1}^{T}$ and $\{d_{2,t}\}_{t=1}^{T}$ are output by Algorithms 13 and 14 with parameters $\beta_t = \left(1 + \sqrt{2 \log |E_t|/(t+1)}\right)^{-1}$ or $t \in [T-1]$ and $\beta = \left(1 + \sqrt{2 \log |E|/(T)}\right)^{-1}$, respectively, on a system $(V, E, w, \text{reward}, s)$ and attack sequence $\{a_t\}_{t=1}^{T}$, then*

$$\frac{1}{T}\sum_{t=1}^{T} \text{profit}(a_t, d_{1,t}) - \frac{1}{T}\sum_{t=1}^{T} \text{profit}(a_t, d_{2,t}) \;\leq\; \frac{B}{T}\overline{w^{-1}} \;.$$

Proof. Consider attack a_t from a round $t \in [T]$ and consider an edge $e \in a_t$. If $e \in a_s$ for some $s < t$, then the defense budget allocated to e at time t by Algorithm 14 cannot be greater than the budget allocated by Algorithm 13. Thus, the instantaneous cost paid by the attacker on e when Algorithm 13 defends is at least the cost paid when Algorithm 14 defends: $d_{1,t}(e)/w(e) \geq d_{2,t}(e)/w(e)$. If $e \notin \bigcup_{s=1}^{t-1} a_s$ then for all $s \in [t]$, $d_{1,s}(e) = 0$, by definition. The sequence $\{d_{2,s}(e)\}_{s=1}^{t-1}$ is decreasing and positive. Thus $\max_{s<t} d_{2,s}(e) - d_{1,s}(e)$ is optimized at $s = 1$ and is equal to $B/|E|$. Finally because each edge $e \in E$ is first revealed exactly once this leads to

$$\begin{aligned}
\sum_{t=1}^{T} \text{cost}(a_t, d_{2,t}) - \sum_{t=1}^{T} \text{cost}(a_t, d_{1,t}) &= \sum_{t=1}^{T}\sum_{e \in a_t} \frac{d_{2,t}(e) - d_{1,t}(e)}{w(e)} \\
&\leq \sum_{e \in E} \frac{B}{|E| w(e)} \;.
\end{aligned}$$

Combined with the fact that the attacker receives the same payout whether Algorithm 14 or Algorithm 13 defends completes the result. □

Proof of Theorem 45. The theorem's result follow immediately from combining Lemma 49 and Lemma 52. □

Finally, notice that Algorithm 13 enjoys the same time and space complexities as Algorithm 14, up to constants.

5.4.3.3 Bound on ROA: Hidden Edges Case

We now translate our bounds on profit into bounds on ROA by observing that the ratio of two quantities is small if the quantities are large and their difference is small. We consider the competitive ratio between a reactive defense strategy and the best proactive defense strategy after the following technical lemma, which asserts that the quantities are large.

Lemma 53. *For all attack sequences* $\{a_t\}_{t=1}^T$, $\max_{d^\star \in \mathcal{D}_{B,E}} \sum_{t=1}^T \text{cost}(a_t, d^\star) \geq VT$ *where game value* V *is* $\max_{d \in \mathcal{D}_{B,E}} \min_a \text{cost}(a, d) = \frac{B}{\sum_{e \in \text{inc}(s)} w(e)} > 0$*, where* $\text{inc}(v) \subseteq E$ *denotes the edges incident to vertex* v*.*

Proof. Let $d^\star = \arg\max_{d \in \mathcal{D}_{B,E}} \min_a \text{cost}(a, d)$ witness the game's value V, then $\max_{d \in \mathcal{D}_{B,E}} \sum_{t=1}^T \text{cost}(a_t, d) \geq \sum_{t=1}^T \text{cost}(a_t, d^\star) \geq TV$. Consider the defensive allocation for each $e \in E$. If $e \in \text{inc}(s)$, let $\tilde{d}(e) = Bw(e)/\sum_{e \in \text{inc}(s)} w(e) > 0$, and otherwise $\tilde{d}(e) = 0$. This allocation is feasible because

$$\begin{aligned} \sum_{e \in E} \tilde{d}(e) &= \frac{B \sum_{e \in \text{inc}(s)} w(e)}{\sum_{e \in \text{inc}(s)} w(e)} \\ &= B \ . \end{aligned}$$

By definition $\tilde{d}(e)/w(e) = B/\sum_{e \in \text{inc}(s)} w(e)$ for each edge e incident to s. Therefore, $\text{cost}(a, \tilde{d}) \geq B/\sum_{e \in \text{inc}(s)} w(e)$ for any non-trivial attack a, which necessarily includes at least one s-incident edge. Finally, $V \geq \min_a \text{cost}(a, \tilde{d})$ proves

$$V \geq \frac{B}{\sum_{e \in \text{inc}(s)} w(e)} \ . \tag{5.2}$$

Now, consider a defense allocation d and fix an attack a that minimizes the total attacker cost under d. At most one edge $e \in a$ can have $d(e) > 0$, for otherwise the cost under d can be reduced by removing an edge from a. Moreover any attack $a \in \arg\min_{e \in \text{inc}(s)} d(e)/w(e)$ minimizes attacker cost under d. Thus the maximin V is witnessed by defense allocations that maximize $\min_{e \in \text{inc}(s)} d(e)/w(e)$. This maximization is achieved by allocation $\tilde{d}$ and so Inequality (5.2) is an equality. □

We are now ready to prove the main ROA theorem:

Proof of Theorem 47. First, observe that for all $B > 0$ and all $A, C \in \mathbb{R}$

$$\frac{A}{B} \leq C \iff A - B \leq (C-1)B \ . \tag{5.3}$$

We will use this equivalence to convert the regret bound on profit to the desired bound on ROA. Together Theorem 45 and Lemma 53 imply

$$\begin{aligned} & \alpha \sum_{t=1}^{T} \text{cost}(a_t, d_t) \\ \geq \ & \alpha \max_{d^\star \in \mathcal{D}_{B,E}} \sum_{t=1}^{T} \text{cost}(a_t, d^\star) - \alpha \frac{B}{2} \sqrt{T \log |E|} - \alpha B \left(\log |E| + \overline{w^{-1}} \right) \\ \geq \ & \alpha V T - \alpha \frac{B}{2} \sqrt{T \log |E|} - \alpha B \left(\log |E| + \overline{w^{-1}} \right) \end{aligned} \tag{5.4}$$

where $V = \max_{d \in \mathcal{D}_{B,E}} \min_a \text{cost}(a, d) > 0$. If

$$\sqrt{T} \ \geq \ \frac{13}{\sqrt{2}} \left(1 + \alpha^{-1}\right) \sqrt{\log |E|} \sum_{e \in \text{inc}(s)} w(e) \ ,$$

we can use inequalities $V = B / \sum_{e \in \text{inc}(s)} w(e)$, $\overline{w^{-1}} \leq 2 \log |E|$ (since $|E| > 1$), and $\left(\sum_{e \in \text{inc}(s)} w(e) \right)^{-1} \leq 1$ to show

$$\sqrt{T} \ \geq \ \left((1+\alpha)B + \sqrt{[(1+\alpha)B + 24\alpha V]\,(1+\alpha)B} \right) (2\sqrt{2}\alpha V)^{-1} \sqrt{\log |E|} \ ,$$

which combines with Theorem 45 and Inequality 5.4 to imply

$$\begin{aligned} \alpha \sum_{t=1}^{T} \text{cost}(a_t, d_t) \ &\geq \ \alpha V T - \alpha \frac{B}{2} \sqrt{T \log |E|} - \alpha B \left(\log |E| + \overline{w^{-1}} \right) \\ &\geq \ \frac{B}{2} \sqrt{T \log |E|} + B \left(\log |E| + \overline{w^{-1}} \right) \\ &\geq \ \sum_{t=1}^{T} \text{profit}(a_t, d_t) - \min_{d^\star \in \mathcal{D}_{B,E}} \sum_{t=1}^{T} \text{profit}(a_t, d^\star) \\ &= \ \sum_{t=1}^{T} \left(- \text{cost}(a_t, d_t) \right) - \min_{d^\star \in \mathcal{D}_{B,E}} \sum_{t=1}^{T} \left(- \text{cost}(a_t, d^\star) \right) \\ &= \ \max_{d^\star \in \mathcal{D}_{B,E}} \sum_{t=1}^{T} \text{cost}(a_t, d^\star) - \sum_{t=1}^{T} \text{cost}(a_t, d_t) \ . \end{aligned}$$

Finally, combining this equation with Equivalence 5.3 yields the result

$$\begin{aligned}
& \frac{\mathrm{ROA}\left(\{a_t\}_{t=1}^T, \{d_t\}_{t=1}^T\right)}{\min_{d^\star \in \mathcal{D}_{B,E}} \mathrm{ROA}\left(\{a_t\}_{t=1}^T, d^\star\right)} \\
= & \frac{\sum_{t=1}^T \mathrm{payoff}(a_t, d_t)}{\sum_{t=1}^T \mathrm{cost}(a_t, d_t)} \cdot \max_{d^\star \in \mathcal{D}_{B,E}} \frac{\sum_{t=1}^T \mathrm{cost}(a_t, d^\star)}{\sum_{t=1}^T \mathrm{payoff}(a_t, d^\star)} \\
= & \frac{\max_{d^\star \in \mathcal{D}_{B,E}} \sum_{t=1}^T \mathrm{cost}(a_t, d^\star)}{\sum_{t=1}^T \mathrm{cost}(a_t, d_t)} \\
\leq & 1 + \alpha \ .
\end{aligned}$$

□

5.4.4 Lower Bounds

We briefly argue the optimality of Algorithm 13 for a particular graph, *i.e.*, we show that Algorithm 13 has optimal convergence time for small enough α, up to constants. (For very large α, Algorithm 13 converges in constant time, and therefore is optimal up to constants, vacuously.) This result establishes a lower bound on the competitive ratio of the ROA for all reactive strategies. The proof gives an example where the best proactive defense (slightly) out-performs every reactive strategy, suggesting the benchmark is not unreasonably weak.

The argument considers an attacker who randomly selects an attack path, rendering knowledge of past attacks useless. Consider a two-vertex graph where the start vertex s is connected to a vertex r (with reward 1) by two parallel edges e_1 and e_2, each with an attack surface of 1. Further suppose that the defense budget $B = 1$. We first show a lower bound on all reactive algorithms:

Lemma 54. *for all reactive algorithms A, the competitive ratio C is at least $(x+\Omega(\sqrt{T}))/x$, i.e., at least $(T + \Omega(\sqrt{T}))/T$ because $x \leq T$.*

Proof. Consider the following random attack sequence: For each round, select an attack path uniform i.i.d. from the set $\{e_1, e_2\}$. A reactive strategy must commit to a defense in every round without knowledge of the attack, and therefore every strategy that expends the entire budget of 1 inflicts an expected cost of $1/2$ in every round. Thus, every reactive strategy inflicts a total expected cost of (at most) $T/2$, where the expectation is over the coin-tosses of the random attack process.

Given an attack sequence, however, there exists a proactive defense allocation with better performance. We can think of the proactive defender being prescient as to which edge (e_1 or e_2) will be attacked most frequently and allocating the entire defense budget to that edge. It is well-known (for instance via an analysis of a one-dimensional random walk) that in such a random process, one of the edges will occur $\Omega(\sqrt{T})$ more often than the other, in expectation.

By the probabilistic method, a property that is true in expectation must hold existentially, and, therefore, for every reactive strategy A, there *exists* an attack sequence such that

A has a cost x, whereas the best proactive strategy (in retrospect) has a cost $x + \Omega(\sqrt{T})$. Because the payoff of each attack is 1, the total reward in either case is T. The prescient proactive defender, therefore, has an ROA of $T/(x + \Omega(\sqrt{T}))$, but the reactive algorithm has an ROA of T/x, establishing the lemma. □

Given this lemma, we show that Algorithm 13 is optimal given the information available. In this case, $n = 2$ and, ignoring constants from Theorem 47, we are trying to match a convergence time T is at most $(1 + \alpha^{-1})^2$, which is approximately α^{-2} for small α. For large enough T, there exists a constant c such that $C \geq (T + c\sqrt{T})/T$. By simple algebra, $(T + c\sqrt{T})/T \geq 1 + \alpha$ whenever $T \leq c^2/\alpha^2$, concluding the argument.

We can generalize the above argument of optimality to $n > 2$ using the combinatorial Lemma 3.2.1 from (Cesa-Bianchi et al., 1993). Specifically, we can show that for every n, there is an n edge graph for which Algorithm 13 is optimal up to constants for small enough α.

5.5 Advantages of Reactivity

In this section, we examine some situations in which a reactive defender out-performs a proactive defender. Proactive defenses hinge on the defender's model of the attacker's incentives. If the defender's model is inaccurate, the defender will construct a proactive defense that is far from optimal. By contrast, a reactive defender need not reason about the attacker's incentives directly. Instead, the reactive defender learns these incentives by observing the attacker in action.

Learning Rewards. One way to model inaccuracies in the defender's estimates of the attacker's incentives is to hide the attacker's rewards from the defender. Without knowledge of the payoffs, a proactive defender has difficulty limiting the attacker's ROA. Consider, for example, the star system whose edges have equal attack surfaces, as depicted in Figure 5.3. Without knowledge of the attacker's rewards, a proactive defender has little choice but to allocate the defense budget equally to each edge (because the edges are indistinguishable). However, if the attacker's reward is concentrated at a single vertex, the competitive ratio for attacker's ROA (compared to the rational proactive defense) is the number of leaf vertices. (We can, of course, make the ratio worse by adding more vertices.) By contrast, the reactive algorithm we analyze in Section 5.4 is competitive with the rational proactive defense because the reactive algorithm effectively learns the rewards by observing which attacks the attacker chooses.

Robustness to Objective. Another way to model inaccuracies in the defender's estimates of the attacker's incentives is to assume the defender mistakes which of profit and ROA actually matter to the attacker. The defense constructed by a rational proactive defender depends crucially on whether the attacker's actual incentives are based on profit or based on ROA, whereas the reactive algorithm we analyze in Section 5.4 is robust to this variation. In particular, consider the system depicted in Figure 5.4, and assume the defender

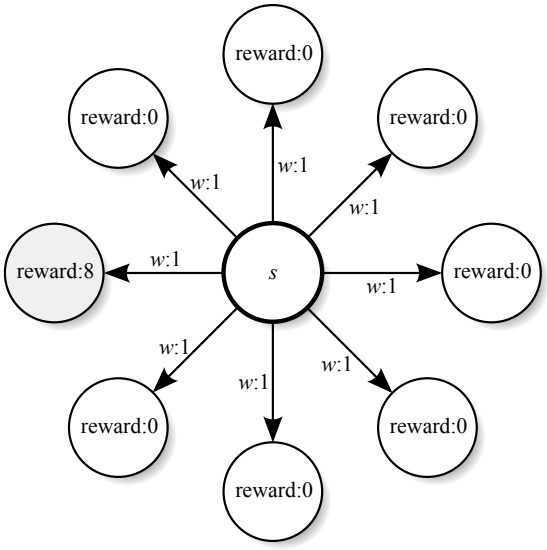


Figure 5.3: Star-shaped attack graph with rewards concentrated in an unknown vertex.

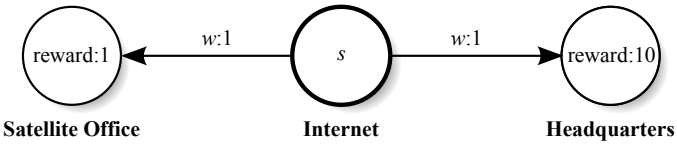


Figure 5.4: An attack graph that separates the minimax strategies optimizing ROA and attacker profit.

has a budget of 9. If the defender believes the attacker is motivated by profit, the rational proactive defense is to allocate the entire defense budget to the right-most edge (making the profit 1 on both edges). However, this defense is disastrous when viewed in terms of ROA because the ROA for the left edge is infinite (as opposed to near unity when the proactive defender optimizes for ROA).

Catachresis. The defense constructed by the rational proactive defender is optimized for a rational attacker. If the attacker is not perfectly rational, there is room for out-performing the rational proactive defense. There are a number of situations in which the attacker might not mount "optimal" attacks:

- The attacker might not have complete knowledge of the attack graph. Consider, for example, a software vendor who discovers five equally severe vulnerabilities in one of their products via fuzzing. According to proactive security, the defender ought to dedicate equal resources to repairing these five vulnerabilities. However, a reactive defender might dedicate more resources to fixing a vulnerability actually exploited by attackers in the wild. We can model these situations by making the attacker oblivious to some edges.

- The attacker might not have complete knowledge of the defense allocation. For example, an attacker attempting to invade a corporate network might target computers in human resources without realizing that attacking the customer relationship management database in sales has a higher return-on-attack because the database is lightly defended.

By observing attacks, the reactive strategy learns a defense tuned for the *actual* attacker, causing the attacker to receive a lower ROA.

5.6 Generalizations

We now consider several simple generalizations of our model and results.

5.6.1 Horn Clauses

Thus far, we have presented our results using a graph-based system model. Our results extend, however, to a more general system model based on Horn clauses and corresponding to hypergraph-based system models. Datalog programs, which are based on Horn clauses, have been used in previous work to represent vulnerability-level attack graphs (Ou et al., 2006). A Horn clause is a statement in propositional logic of the form $p_1 \wedge p_2 \wedge \cdots \wedge p_n \rightarrow q$. The propositions $p_1, p_2, \ldots, p_n$ are called the *antecedents*, and q is called the *consequent*. The set of antecedents might be empty, in which case the clause simply asserts the consequent. Notice that Horn clauses are negation-free. In some sense, a Horn clause represents an edge in a hypergraph where multiple pre-conditions are required before taking a certain state transition.

In the Horn model, a system consists of a set of Horn clauses, an attack surface for each clause, and a reward for each proposition. The defender allocates defense budget among the Horn clauses. To mount an attack, the attacker selects a *valid proof*: an ordered list of rules such that each antecedent appears as a consequent of a rule earlier in the list. For a given proof Π,

$$\text{cost}(\Pi, d) = \sum_{c \in \Pi} d(c)/w(e) \qquad \text{payoff}(\Pi) = \sum_{p \in [\![\Pi]\!]} \text{reward}(p) \ ,$$

where $[\![\Pi]\!]$ is the set of propositions proved by Π (*i.e.*, those propositions that appear as consequents in Π). Profit and ROA are computed as before.

Our results generalize to this model directly. Essentially, we need only replace each instance of the word "edge" with "Horn clause" and "path" with "valid proof." For example, the rows of the matrix M used throughout the proof become the Horn clauses, and the columns become the valid proofs (which are numerous, but no matter). The entries of the matrix become $M(c, \Pi) = 1/w(c)$, analogous to the graph case. The one non-obvious substitution is $\text{inc}(s)$, which becomes the set of clauses that lack antecedents.

5.6.2 Multiple Attackers

We have focused on a security game between a *single* attacker and a defender. In practice, a security system might be attacked by several uncoordinated attackers, each with different information and different objectives. Fortunately, we can show that a model with multiple attackers is mathematically equivalent to a model with a single attacker with a randomized

strategy: Use the set of attacks, one per attacker, to define a distribution over edges where the probability of an edge is linearly proportional to the number of attacks which use the edge. This precludes the interpretation of an attack as an s-rooted path, but our proofs do not rely upon this interpretation and our results hold in such a model with appropriate modifications.

5.6.3 Adaptive Proactive Defenders

A simple application of an online learning result (Herbster and Warmuth, 1998), modifies our regret bounds to compare the reactive defender to an optimal proactive defender who re-allocates budget a fixed number of times. In this model, our results remain qualitatively the same.

5.7 Summary

Many security experts equate reactive security with myopic bug-chasing and ignore principled reactive strategies when they recommend adopting a proactive approach to risk management. In this chapter, we establish sufficient conditions for a learning-based reactive strategy to be competitive with the best fixed proactive defense. Additionally, we show that reactive defenders can out-perform proactive defenders when the proactive defender defends against attacks that never actually occur. Although our model is an abstraction of the complex interplay between attackers and defenders, our results support the following practical advice for CISOs making security investments:

- Employ monitoring tools that let you detect and analyze attacks against your enterprise. These tools help focus your efforts on thwarting real attacks.
- Make your security organization more agile. For example, build a rigorous testing lab that lets you roll out security patches quickly once you detect that attackers are exploiting these vulnerabilities.
- When determining how to expend your security budget, avoid overreacting to the most recent attack. Instead, consider all previous attacks, but discount the importance of past attacks exponentially.

In some situations, proactive security can out-perform reactive security. For example, reactive approaches are ill-suited for defending against catastrophic attacks because there is no "next round" in which the defender can use information learned from the attack. We hope our results will lead to a productive discussion of the limitations of our model and the validity of our conclusions.

Instead of assuming that proactive security is always superior to reactive security, we invite the reader to consider when a reactive approach might be appropriate. For the parts of an enterprise where the defender's budget is liquid and there are no catastrophic losses, a carefully constructed reactive strategy can be as effective as the best proactive defense in the worst case and significantly better in the best case.

Chapter 6

Learning to Find Leaks in Open Source Projects

A small leak can sink a great ship.
– Benjamin Franklin

Many open-source projects land security fixes in public repositories before shipping these patches to users. In this chapter, we show that an attacker who uses off-the-shelf machine learning techniques can detect these security patches using metadata about the patch (*e.g.*, the author of the patch, which files were modified, and the size of the modification). By analyzing two patches each day for exploitability over a period of 8 months, an attacker can add 148 days to the window of vulnerability for Firefox 3, increasing the total window of vulnerability by a factor of 6.4. We argue that obfuscating this metadata is unlikely to prevent these information leaks because the detection algorithm aggregates weak signals from a number of features. Instead, open-source projects ought to keep security patches secret until they are ready to be released.

Although the attacks of this chapter do not target learners *per se*, we can still classify them using the taxonomy of Barreno et al. (2006) overviewed in Section 1.2.2. By regarding the patches of Firefox together with their labels in {'security', 'non-security'} as a training set, and the public repository consisting of landed pre-release patches as a statistic, our attacks are Targeted and violate Confidentiality as they aim to determine the labels of specific patches by examining the repository. While we model an attacker with no control over the repository, our attacks exploit a significant amount of information in the form of meta-data from the repository and previously disclosed labels.

6.1 Introduction

Many important and popular software development projects are open-source, including Firefox, Chromium, Apache, the Linux kernel, and OpenSSL. Software produced by these projects is run by hundreds of millions of users and machines. Following the open-source

spirit, these projects make all code changes immediately visible to the public in open code repositories, including landing fixes to security vulnerabilities in public "trunk" development branches before publicly announcing the vulnerability and providing an updated version to end users. This common practice raises the question of whether this extreme openness increases the *window of vulnerability* by letting attackers discover vulnerabilities earlier in the security life-cycle. The conventional wisdom is that detecting these security patches is difficult because the patches are hidden among a cacophony of non-security changes. For example, the central Firefox repository receives, on average, 38.6 patches per day, of which 0.34 fix security vulnerabilities. Recently, some blackhats in the Metasploit project have used the "description" metadata field to find Firefox patches that refer to non-public bug numbers (Veditz, 2009). The Firefox developers have responded by obfuscating the description field, but where does this cat-and-mouse game end?

In this chapter, we analyze information leaks during the Firefox 3 life-cycle to answer three key questions: (1) Does the metadata associated with patches in the source code repository contain information about whether the patch is security sensitive? (2) Using this information, how much less effort does an attacker need to expend to find unannounced security vulnerabilities? (3) How much do these information leaks increase the total window of vulnerability?

To address these questions, we apply off-the-shelf machine learning techniques to discriminate between security and non-security patches by observing some intrinsic metadata about each patch. For example, we use the patch author, the set of files modified, and the size of the modifications. We strengthen our conclusions by ignoring the description field because we assume that the developers can successfully obfuscate the patch description. Using standard machine learning techniques, we show that each of these features individually do not contain much information about whether patches are security sensitive. We find that the patch author contains the most information (followed by the top-level directory containing the modified files and the size of the modification), but even author has a tiny information gain ratio of 0.003.

We use a support vector machine (SVM) to aggregate the information in these features, but still obtain a poor classifier when measured in terms of precision and recall. However, the attacker's goal is not to classify every patch as security-sensitive or non-security-sensitive. The attacker's goal is to find at least one vulnerability to exploit; in particular, the attacker can use the SVM to rank patches by how confident the learner is in classifying the patch security-sensitive. This ranking function gives the attacker an informed way to prioritize patches to examine when searching for security patches. Thus, we propose new metrics to measure the effectiveness of the detection algorithm. In the first metric, *attacker effort*, we measure the number of patches the attacker would need to examine before finding the *first* vulnerability according to the ranked list generated by the detection algorithm. Using this metric, we show that even our weak classifier is useful to the attacker. For example on 39% of the days, the detection algorithm ranks at least one security patch in the top two patches. In the second metric, *increase to the window of vulnerability*, we show that an attacker who examines the top two patches ranked by the detection algorithm each day will add an extra 148 days of vulnerability to the 229 day period we study, representing a 6.4-fold increase

over the window of vulnerability caused by the latency in deploying security updates.

Our results suggest that Firefox should change its security life-cycle to avoid leaking information about unannounced vulnerabilities in its public source code repositories. Instead of landing security patches in the central repository, Firefox developers should land security patches in a private release branch that is available only to a set of trusted testers. The developers can then merge the patches into the public repository at the same time they release the security update to all users and announce the vulnerability.

Although we study Firefox specifically, we believe our results generalize to a number of other open-source projects, including Chromium, Apache, the Linux kernel, and OpenSSL, which land vulnerability fixes in public repositories before announcing the vulnerability and making security updates available. However, we choose to study these issues in Firefox because Firefox has a state-of-the-art process for responding to vulnerability reports and publishes the ground truth about which patches fix security vulnerabilities (Mozilla Foundation, 2010).

Chapter Organization. The remainder of the chapter is organized as follows. Section 6.2 describes the existing Firefox security life-cycle. Section 6.3 lays out the dataset we analyze. Section 6.4 explains our methodology. Section 6.5 presents our results. Section 6.6 recommends a secure security life-cycle. Section 6.7 concludes the chapter with a short summary of the main contributions.

6.2 Life-Cycle of a Vulnerability

This section describes the life-cycle of a security patch for the Firefox browser. We take Firefox as a representative example, but many open-source projects use a similar life-cycle.

6.2.1 Stages in the Life-Cycle

In the Firefox open-source project, vulnerabilities proceed through a sequence of observable events:

1. **Bug filed.** The Firefox project encourages security researchers to report vulnerabilities to the project via the project's public bug tracker. When filed, security bugs are marked "private" (meaning access is restricted to a trusted set of individuals on the security team Veditz 2010; see Figure 6.1) and are assigned a unique number.

2. **Patch landed in mozilla-central.** Once the developers determine the best way to fix the vulnerability, a developer writes a patch for the mainline "trunk" of Firefox development. Other developers review the patch for correctness, and once the patch is approved, the developer *lands* the patch in the public `mozilla-central` Mercurial repository.

Figure 6.1: Information leaked about security-sensitive bug numbers has been exploited by attackers in the past to identify undisclosed vulnerabilities, when bug numbers were linked to landed patches via patch descriptions.

3. **Patch landed in release branches.** After the patch successfully lands on `mozilla-central` (including passing all the automated regression and performance tests), the developers *merge* the patch to one or more of the Firefox release branches.

4. **Security update released.** At some point, a release driver decides to release an updated version of Firefox containing one or more security fixes (and possibly some non-security related changes). These releases are typically made from the release branch, not from the `mozilla-central` repository. The current state of the release branch is packaged, signed, and made available to users via Firefox's auto-update system.

   (a) **Vulnerability announced.** The Firefox developers announce the vulnerabilities fixed in the release (Mozilla Foundation, 2010). For the majority of vulnerabilities, disclosure is simultaneous with the release of the binary. However in some cases disclosure can occur weeks later (after the security update is applied by most users).

5. **Security update applied.** Once a user's auto-update client receives an updated version of the Firefox binary, Firefox updates itself and notifies the user that a security update has been applied. Once the user chooses to install the update, the user is protected from an attacker exploiting the vulnerability.

Previous work (Frei et al., 2009) has analyzed the dynamics between steps (4) and (5), finding that the user experience and download size have a dramatic effect on the time delay and, hence, the window of vulnerability. With a sufficiently slick update experience, browser vendors can reduce the lag between (4) and (5) to a matter of days. Recent releases of Firefox have an improved update experience that reduces the window of vulnerability between steps (4) and (5).

However, not as much attention has been paid to the dynamics between steps (1) and (4), likely because most people make the assumption that little or nothing is revealed about a vulnerability until the vulnerability is intentionally disclosed in step (4). Unfortunately, there are a number of information leaks in this process that invalidate that assumption.

6.2.2 Information Leaks in Each Stage

Each stage in the vulnerability life-cycle leaks some amount of information about vulnerabilities to potential attackers. For example, even the first step leaks some amount of information because bug numbers are issued sequentially and an attacker can brute force bug numbers to determine which are "forbidden" and hence represent security vulnerabilities. Of course, simply knowing that a vulnerability was reported to Firefox does not give the attacker much useful information for creating an exploit.

The developers leak more information when they land security patches in `mozilla-central` because the `mozilla-central` is a public repository. It is unclear, *a priori*, whether an attacker will be able to find security patches landing in `mozilla-central` because these security patches are landed amid a "thundering herd" of other patches (see Figure 6.2), but if an attacker can detect that a patch fixes a security vulnerability, the attacker can learn information about the vulnerability. For example, the attacker learns where in the code base the vulnerability exists. If the patch fixes a vulnerability by adding a bounds check, the attacker can look for program inputs that generate large buffers of the checked type. In this work, we do not evaluate the difficulty of reverse engineering an exploit from a vulnerability fix, but there has been some previous work (Brumley et al., 2008) on reverse engineering exploits from binary patches (which is, of course, more difficult than reverse engineering exploits from source patches).

6.3 Analysis Goals and Setup

In this section, we describe the dataset and the success metrics for the detection algorithm.

6.3.1 Dataset

Set of Patches. In our experiment, we considered the complete life-cycle of Firefox 3, which lasted over 12 months, contained 14,416 non-security patches, 125 security patches, and 12 security updates. In particular, we use publicly available data starting from the release of Firefox 3 and ending with the release of Firefox 3.5. Also, to strengthen our results, we focus on the `mozilla-central` repository, which receives the vast majority of Firefox development effort. We cloned the entire `mozilla-central` repository to our experimental machines to identify all patches during the life-cycle of Firefox 3. We ignore the release branches to evaluate how well our detection algorithm is able to find security fixes amid mainline development (see Figure 6.2). Even though we initially limit our attention

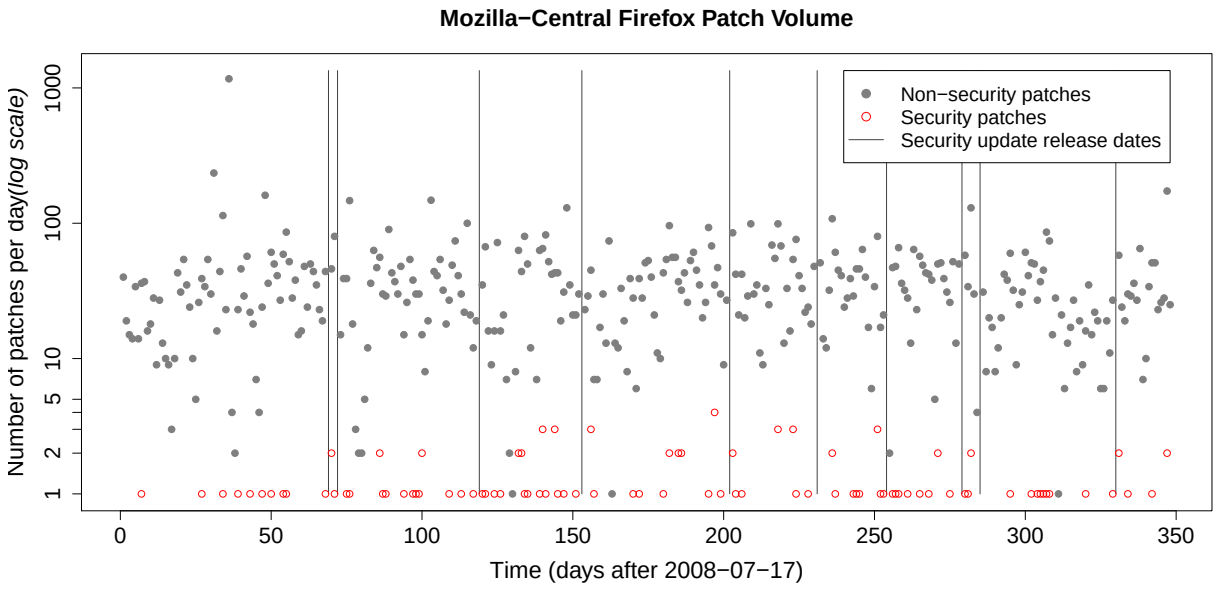


Figure 6.2: Attackers must find security patches within a "thundering herd" of non-security patches.

to Firefox 3, we repeat our results on Firefox 3.5 in Section 6.5.4 and expect our results generalize to other releases of Firefox and, more generally, to other open-source projects.

Ground Truth. We determined the "ground truth" of whether a patch fixes a security vulnerability by examining the list of known vulnerabilities published by Firefox (Mozilla Foundation, 2010). Each CVE listed on the known vulnerability web page contains a link to one or more entries in the Firefox bug database. At the time we crawled these bug entries (after disclosure), the bug entries were public and contained links to the Mercurial commits that fixed the vulnerabilities (both in `mozilla-central` and in the release branches). Our crawler harvested these links and extracted the unique identifier for each patch.

The known vulnerability page dates each vulnerability disclosure, and we assume that these disclosure dates are accurate. Each bug entry is timestamped with its creation date and every message on the bug thread is dated as well. Finally, the `mozilla-central` pushlog website contains the date and time of every change in the "pushlog," which we also assume is authoritative.

6.3.2 Success Metrics

Given a dataset of past patches, an attacker can label the patches based on whether these patches have been announced as vulnerability fixes. Using these labels, the attacker can train a statistical machine learning algorithm to predict whether a current patch fixes an (unannounced) vulnerability. Using this machine learning algorithm, the attacker can

classify a new patch into one of two classes: security-sensitive and non-security-sensitive. However, the attacker's goal is actually not to classify each patch correctly. For example, the attacker does not care if a detection algorithm has a high false negative rate (often incorrectly classifies security vulnerabilities as non-vulnerability patches) if the algorithm reliably finds at least one security vulnerability—the attacker need only exploit one vulnerability to compromise users' computers.

Instead, the attacker's goal is to build a detection algorithm that makes it easier to find at least one (unannounced) vulnerability. In particular, we consider detection algorithms that output a real-valued confidence for their prediction about whether a given patch is a security patch. The attacker can use this confidence value to rank a set of patches, and then examine the patches in rank order. In this way, the detection algorithm prioritizes which patches the attacker should examine for exploitability. The usefulness of the detection algorithm, then, lies in how much effort the detector saves the attacker by giving real security patches higher priorities than non-security patches. By reducing the amount of effort the attacker must expend to find a vulnerability, the attacker can find vulnerabilities earlier, and increase the window of vulnerability. In particular, we formalize these notions into the following two success metrics.

6.3.2.1 Cost of Vulnerability Discovery: Attacker Effort

Given a set of patches and a ranking function, we call the rank of the first true security patch the *attacker effort*. This quantity reflects the number of patches the attacker has to examine when searching the ranked list before finding the first patch that fixes a security vulnerability. For example, if the third-highest ranked patch actually fixes a security vulnerability, then the attacker needs to examine three patches before finding the vulnerability, resulting in an attacker effort of three. Using this metric, we can compute the percent of days on which an attacker who expends a given effort will be able to find a security patch.

6.3.2.2 Benefit for the Attacker: Window of Vulnerability

Another metric we propose is the *increase to the window of vulnerability* due to the assistance of the detection algorithm. In particular, an attacker who discovers a vulnerability d days before the next security update increases the total window of vulnerability for Firefox users by d days. (Notice that knowing of multiple vulnerabilities simultaneously does not increase the aggregate window of vulnerability because knowing multiple vulnerabilities simultaneously is redundant.)

Previous work (Duebendorfer and Frei, 2009) explores the effectiveness of browser update mechanisms, finding that security updates take some amount of time to propagate to users. In particular, they measured the cumulative distribution of the number of days users take to update their browsers after security updates (Duebendorfer and Frei, 2009, Figure 3). After about 10 days, the penetration growth rate rapidly decreases, asymptotically approaching about 80%. By integrating the area above the CDF up to 80%, we can estimate the expected number of days a user takes to update Firefox 3 conditioned that they are in the first 80%

who update. We estimate this quantity, the *post-release window of vulnerability*, to be 3.4 days, which we use as a baseline value for comparing windows of vulnerability.

6.3.3 Baseline: The Random Ranker

To quantify the benefit of using machine learning, we compare our detection algorithm with a *random ranker* who examines available patches in a random order. This straw-man algorithm has two key properties: (1) our attacker cost and benefit metrics are easy to measure for the random ranker (as we detail below); (2) perhaps more importantly, the random ranker models a real-world attacker who has access to `mozilla-central` but does not have any reason to examine landed patches in any particular order.

As shown in the next two sections both expected cost and benefit metrics can be computed exactly and efficiently for the random ranker, given the total number of patches and the number of security patches.

6.3.4 Deriving Random Ranker Expected Effort

We model the random ranker as iteratively selecting patches one-at-a-time, uniformly-at-random from the pool of patches available in `mozilla-central`, without replacement. This attacker's effort is the random number X of patches the attacker must examine up to and including the first patch drawn that fixes a vulnerability. We summarize the cost of using unassisted random ranking via the *expected attacker effort*, to be

$$\mathbb{E}[X] = \binom{n}{n_s}^{-1} \sum_{x=1}^{n-n_s+1} x \binom{n-x}{n_s-1} , \tag{6.1}$$

where $n \in \mathbb{N}$ is the total number of patches available in the pool, and $1 \leq n_s < n$ is the number of these patches that fix vulnerabilities. We now derive Equation (6.1).

If the ranker's sampling were performed with replacement, then the distribution of attacker effort X would be geometric with known expectation. Without replacement, if there are n patches in the pool, n_s of which fix vulnerabilities, X has probability mass

$$\Pr(X = x) = \frac{(n-n_s)!}{(n-n_s-x+1)!} \cdot \frac{(n-x+1)!}{n!} \cdot \frac{n_s}{n-x+1} \tag{6.2}$$

$$= \binom{n-x}{n_s-1} \binom{n}{n_s}^{-1} , \tag{6.3}$$

for $x \in \{1, \ldots, n-n_s+1\}$ and zero otherwise. The second equality follows from some simple algebra. The first equality is derived as follows. The probability of the first draw being a non-security patch is the number of non-security patches over the number of patches or $(n-n_s)/n$. Conditioned on the first patch not fixing a vulnerability, the second draw has probability $(n-n_s-1)/(n-1)$ of being non-security related since one fewer patch is in the pool (which is, in particular a non-security patch). This process continues with

the k^{th} draw having (conditional) probability $(n - n_s - k + 1)/(n - k + 1)$ of being a non-security patch. After drawing k non-security patches, the probability of selecting a patch that fixes a vulnerability is $n_s/(n-k)$. Equation (6.2) follows by chaining these conditional probabilities.

With X's probability mass in hand, the expectation can be efficiently computed for any moderate (n, n_s) pair by summing Equation (6.3).

6.3.5 Deriving Random Ranker Expected Vulnerability Window Increase

We begin by constructing the distribution of the first day an undisclosed vulnerability fix is found after a security update, when the random ranker is constrained to a budget b of patches daily, and never re-examines patches. Let n_t and $n_{t,s}$ denote the number of new patches and new vulnerability fixes landed on day $t \in \mathbb{N}$. Let random variable $X^n_{n_s}$ be the attacker effort required to find one of n_s vulnerability fixes out of a pool of n patches as described above. Finally, let A_t be the event that the first vulnerability fix is found on day $t \in \mathbb{N}$. Then trivially

$$\Pr(A_1) \;=\; \Pr\left(X^{n_1}_{n_{1,s}} \leq b\right) \;.$$

Now we may condition on $\neg A_1$ to express the probability of A_2 occurring: if A_1 does not occur then b non-security-related patches are removed from the pool, so that the pool consists of $n_{1,s} + n_{1,s}$ vulnerability fixes and $n_1 + n_2 - b$ patches total. The conditional probability of A_2 given $\neg A_2$ is then the probability of $\{X^{n_1+n_2-b}_{n_{1,s}+n_{1,s}} \leq b\}$. By induction we can continue to exploit this conditional independence to yield for all $t > 0$

$$\Pr(A_{t+1} \mid \neg A_1 \cap \ldots \cap \neg A_t) \;=\; \Pr\left(X^{\left(\sum_{i=1}^{t+1} n_i\right)-tb}_{\sum_{i=1}^{t+1} n_{i,s}} \leq b\right) \;. \tag{6.4}$$

The RHS of this expression is easily calculated by summing Equation (6.3) over $x \in [b]$. The unconditional probability distribution now follows from the mutual exclusivity of the A_t and the chain rule of probability

$$\begin{aligned}
\Pr(A_{t+1}) &= \Pr(A_{t+1} \cap \neg A_t \cap \ldots \cap \neg A_1) \\
&= \Pr(A_{t+1} \mid \neg A_t \cap \ldots \cap \neg A_1) \prod_{i=1}^{t} \Pr(\neg A_i \mid \neg A_{i-1} \cap \ldots \cap \neg A_1) \\
&= \Pr\left(X^{\left(\sum_{i=1}^{t+1} n_i\right)-tb}_{\sum_{i=1}^{t+1} n_{i,s}} \leq b\right) \prod_{j=1}^{t} \left(1 - \Pr\left(X^{\left(\sum_{i=1}^{j} n_i\right)-(j-1)b}_{\sum_{i=1}^{j} n_{i,s}} \leq b\right)\right) \;.
\end{aligned}$$

Thus we need only compute the expression in Equation (6.4) once for each $t \in [N]$, where N is the number of days until the next security update. From these conditional probabilities we can efficiently calculate the unconditional $\Pr(A_t)$ for each $t \in [N]$. Noting

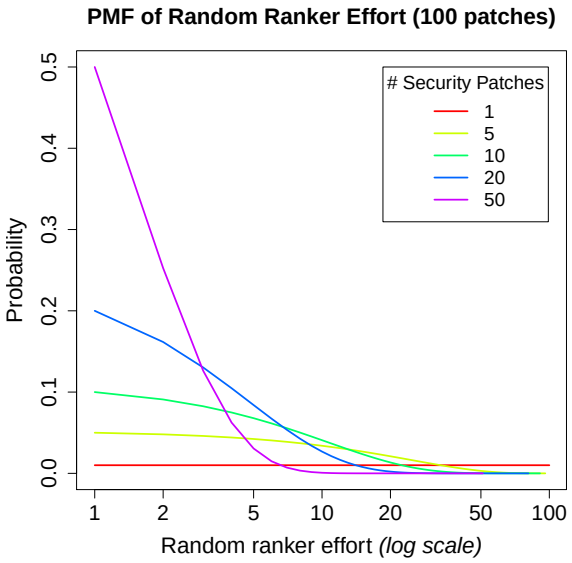


Figure 6.3: The distribution of the random ranker's effort X as a function of n_s for $n = 100$, as given by Equation (6.3).

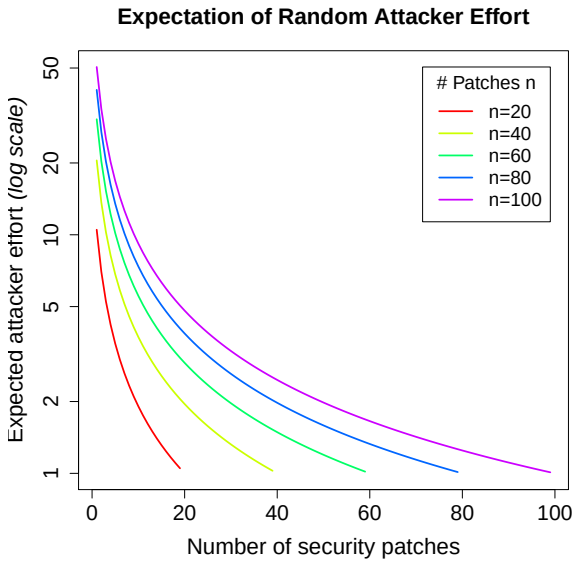


Figure 6.4: The random ranker's expected effort $\mathbb{E}[X]$ as a function of n_s for $n \in \{20, 40, 60, 80, 100\}$, as given by Equation (6.1).

that A_t implies an increase of $Y = N - t + 1$ to the window of vulnerability, the expected increase is

$$\mathbb{E}[Y] \quad = \quad \sum_{t=1}^{N} (N - t + 1) \Pr(A_t) \quad . \tag{6.5}$$

Remark 55. *Notice that there can be a non-trivial probability that no vulnerability fix will be found by the random ranker in the N day period. This probability is simply $1 - \sum_{t=1}^{N} \Pr(A_t)$. On typical inter-update periods this probability can be higher than 0.5 for budgets ≈ 1. This fact serves to reduce the expected increase to the window of vulnerability, particularly for small budgets.*

Remark 56. *The astute reader will notice that we removed b non-security-related patches from the pool on all days we do not find a vulnerability fix, irrespective of whether b or more such patches are present. We have assumed that n is large for simplicity of exposition. Once n drops to $n_s + b$ or lower, we remove all non-security-related patches upon failing to find a vulnerability fix. On the next day, the probability of finding a vulnerability fix is unity. The probabilities of finding vulnerability fixes on subsequent days are thus zero. Thus as b increases, the distribution becomes more and more concentrated at the start of the inter-update period as we would expect. Finally, if no vulnerability fixes are present in the pool on a particular day, then the probability of finding such a patch is trivially zero.*

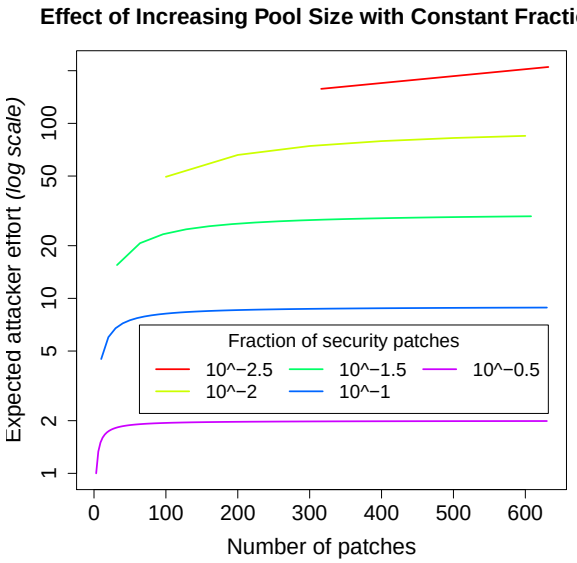


Figure 6.5: The random ranker's expected effort $\mathbb{E}[X]$ as a function of n for constant fractions of security patches $n_s/n \in \{0.0032, 0.01, 0.032, 0.1, 0.32\}$, as given by Equation (6.1).

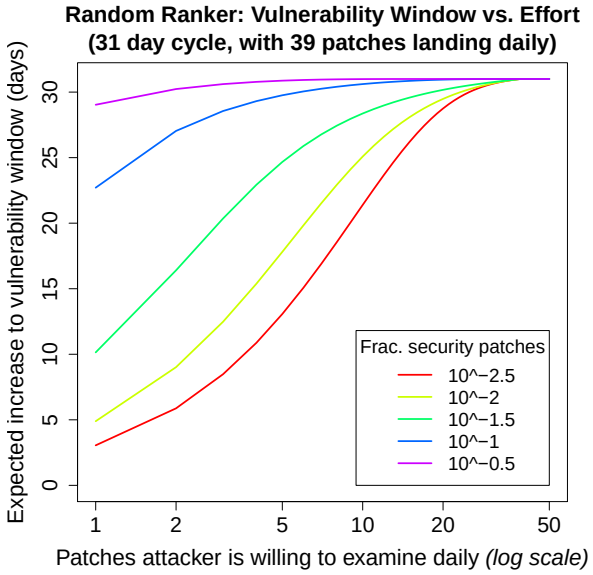


Figure 6.6: The random ranker's expected vulnerability window increase vs. daily budget, for a 31 day cycle with 39 patches daily (the Firefox 3 averages). Benefits shown for security patch fractions of Figure 6.5.

6.3.5.1 Understanding the Random Ranker Metrics

The probability mass and expectation of X are explored in Figures 6.3 and 6.4. For $n_s = 1$ the distribution of effort is uniform; and as the number of security patches increases under a constant pool size, mass quickly concentrates on lower effort (note that in each figure attacker effort is depicted on a log scale). Similarly the significant effect of varying n_s on the expected effort can be seen in Figure 6.4.

For constant fractions of patches that fix a security vulnerability, the expected effort to find a security patch and the expected vulnerability window increase for a typical Firefox 3 inter-point release cycle are shown in Figures 6.5 and 6.6. In both cases effort is shown in a log scale. The former figure shows that under a growing pool of patches with constant fraction being security-related, the attacker effort for finding a security patch is not constant but in fact *increases* as the pool expands. For a typical cycle (of length 31 days), typical patch landing rate (of 39 patches daily) and fixed fraction of landed patches that are security-related, Figure 6.6 shows the expected window increase as a function of the daily budget. Again we see a great difference over increasing proportions of security patches, and the effect of the proportion of the dependence of benefit on budget. Finally notice that the average fraction of security-related patches for Firefox 3 is 0.0085, and so the corresponding curves at 10^{-2} should approximate the performance of the random ranker for Firefox 3 as is verified in Section 6.5. The utility of including curves at atypical security patch rates (from

the perspective of Firefox) is to preview the cost and benefit achieved by the random ranker as applied to other open-source projects.

6.4 Methodology

In this section, we describe the methodology we use to analyze information leaks in the security life-cycle. We first describe the features we observe and then outline our approach to building a detection algorithm.

6.4.1 Features Used By the Detector

There are a number of features we could use to identify security patches. Blackhats in the Metasploit project have used the "description" metadata to determine whether a patch fixes a security vulnerability. Firefox patches typically reference the bug number that they fix in their descriptions. By attempting to access the indicated bug, an attacker can determine whether the patch references a security-sensitive bug (see Figure 6.1). When the Firefox developers became aware of Metasploit's actions, they took steps to obfuscate the patch description (Veditz, 2009).

Obfuscating and de-obfuscating the patch description is clearly a cat-and-mouse game. Instead of analyzing leaks in the patch description, we strengthen our conclusions by assuming that the Firefox developers are able to perfectly obfuscate the patch description. Instead of the description, we analyze information leaks in other metadata associated with the patch (see Figure 6.7).

- **Author.** We hypothesize that information about the patch author (the developer who wrote the patch) will leak a sizable amount of information because Firefox has a security team (Veditz, 2010) that is responsible for fixing security vulnerabilities. Most members of the Firefox community do not have access to security bugs and are unlikely to write security patches.

- **Top-level directory.** For each file that was modified by the patch, we observed the top-level directory in the repository that contained the file. In the Firefox directory structure, the top-level directory roughly corresponds to the module containing the file. If a patch touches more than one top-level directory, we picked the directory that contains the most modified files.

- **File type.** For each file that was modified by the patch, we observe the file's extension to impute the type of the file. For example, patches to Firefox often modify C++ implementation files, interface description files, and XML user interface descriptions. If a patch touches more than one type of file, we pick the file type with the most modified files.

- **Patch size.** We observe a number of size metrics for each patch, including the total size of the diff in characters, the number of lines in the diff, the number of files in

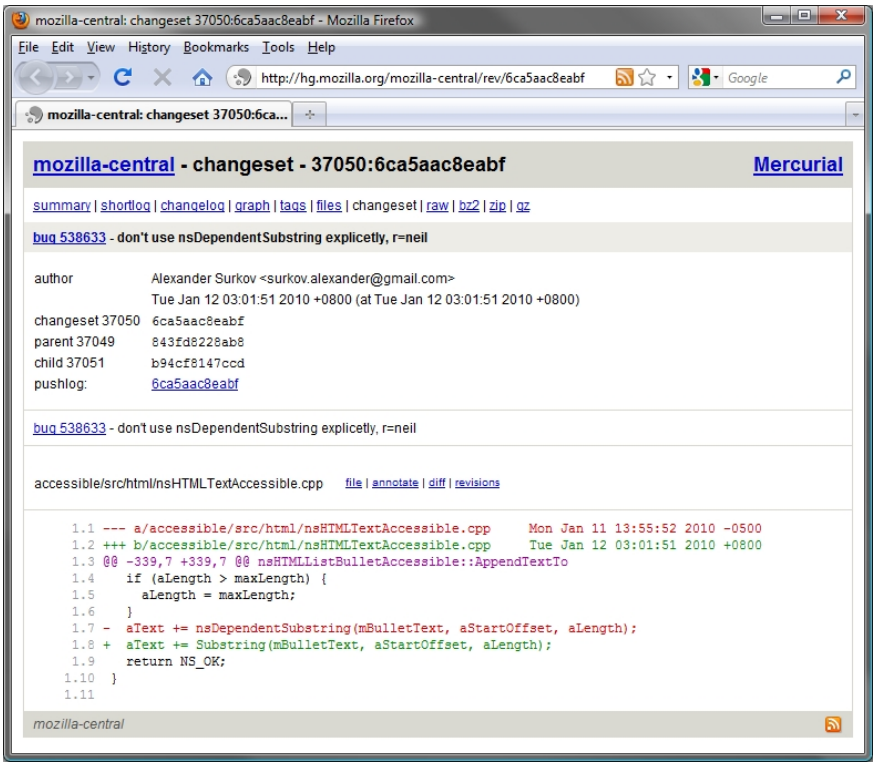


Figure 6.7: An example patch from the Firefox Mercurial repository. In addition to patch description and bug number, several features leak information about the security-related nature of a patch.

the diff, and the average size of all modified files. Although these features are highly correlated, the SVM's ability to model non-linear patterns lets us take advantage of all these features.

- **Temporal.** The timestamp for each patch reveals the time of day and the day of week the patch was landed in the `mozilla-central` repository. We include these features in case, for example, some developers prefer to land security fixes at night or on the weekends.

We presented nominal features (author, top-level directory, and file type) to the SVM as binary vectors. For example, the i^{th} author out of N developers in the Firefox project is represented as $N-1$ zeros and a single 1 in the i^{th} position.

Although these features are harder to obfuscate than the free-form description field, we do not claim that these features cannot be obfuscated. Instead, we claim that there are a large number of small information leaks that can be combined to detect security patches. Of course, this set of features is far from exhaustive and serves only to lower bound the attacker's abilities.

6.4.2 Detection Approach

Algorithm. For our detection algorithm, we use the popular `libsvm` library for support vector machine (SVM) learning (Chang and Lin, 2001). Although we could improve our metrics by tuning the learning algorithm, we choose to use the default configuration to strengthen our conclusions—extracting basic features (as detailed above) and running `libsvm` in its default configuration requires only basic knowledge of Python and no expertise in machine learning.

Support vector machines perform supervised binary classification by learning a maximum-margin hyperplane in a high-dimensional feature space (Burges, 1998; Cristianini and Shawe-Taylor, 2000; Schölkopf and Smola, 2001). Many feature mappings are possible, and the default `libsvm` configuration uses the feature mapping induced by the Radial Basis Function kernel, which takes a parameter γ that controls kernel width. The SVM takes another parameter C, which controls regularization. An attacker need not know how to set these parameters because `libsvm` chooses the parameters that optimize 5-fold cross-validation estimates over a grid of (C, γ) pairs. The optimizing pair is then used to train the final SVM model. We enable a feature of `libsvm` that learns posterior probability estimates $\Pr(\text{patch fixes a vulnerability} \mid \text{patch})$ rather than security/non-security class predictions (Lin et al., 2007). We refer to these posterior probabilities as *probabilities* or *scores*.

After training an SVM on patches labeled as security or non-security, we can use the SVM to rank a set of previously unseen patches by ordering the patches in decreasing order of score. If the SVM is given sufficient training data, we expect the higher-ranked patches to be more likely to fix vulnerabilities. As we show in Section 6.5, even though the SVM scores are unsuitable for classification they are an effective means for ranking patches.

Note that detecting patches that repair vulnerabilities can be cast as learning problems other than scalar-valued supervised classification. For example, we could take a more direct approach via ranking or ordinal regression (although these again do not directly optimize our primary interest: having *one* security patch ranked high). However, we use an SVM because it balances statistical performance for learning highly non-linear decision rules and availability of off-the-shelf software appropriate for data mining novices.

Online learning. To limit the detector to information available to real attackers, we perform the following simulation using the dates collected in our data set. For each day, starting on the day Firefox 3 was released and ending on the day Firefox 3.5 was released, we perform the following steps:

1. We train a fresh SVM on all the patches landed in the repository between the day Firefox 3 was released and the most recent security update before the current day, labeling each patch according to the publicly known vulnerabilities list (Mozilla Foundation, 2010).[1]

[1]Note that not all security patches are disclosed as fixing vulnerabilities by the following release. Such patches are necessarily (mis)labeled as non-security, and trained on as such. Once the true patch is disclosed, we re-label and re-train. The net effect of delayed disclosure is a slight degradation to the SML-assisted ranker's performance.

2. We then use the trained SVM to rank all the patches landed since the most recent security update.

After running the complete online simulation, we observe the highest ranking received by a real vulnerability fix on each day. This ranking corresponds to the SVM-assisted *attacker effort* for that day—the number of patches an attacker using the SVM would need to analyze before encountering a security patch. For each day we also compute the expected unassisted attacker effort as represented by Equation (6.1), using the size of that day's available pool of patches and number of security patches.

6.5 Results

We now present the results of searching for security patches in `mozilla-central`. We first explore the discriminative power of the individual features and then compare an SVM-assisted attacker to an unassisted attacker using a random patch ranking.

6.5.1 Feature Analysis

Analyzing Discriminative Power of Individual Features. Prior to computing the SVM-assisted attacker effort, we analyzed the ability of individual features to discriminate between security and non-security patches. We adopt the information theoretic *information gain ratio*, which reflects the decrease in entropy of the sequence of training set class labels when split by each individual feature. Before running the feature analysis on Firefox data, we briefly overview background on the information gain ratio.

The information theoretic quantity known as the *information gain* measures how well a feature separates a set of training data, and is popular in information retrieval and in machine learning within the ID3 and C4.5 decision tree learning algorithms (Mitchell, 1997).

For training set S and nominal feature F taking discrete values in $\mathcal{X}_F$, the information gain is defined as

$$\text{Gain}(S, F) \;=\; \text{Entropy}(S_\ell) - \sum_{x \in \mathcal{X}_F} \frac{|S_{\ell,x}|}{|S|} \text{Entropy}(S_{\ell,x}) \;, \tag{6.6}$$

where S_ℓ denotes the multiset of S's example binary labels, $S_{\ell,x}$ denotes the subset of these labels for examples with feature F value x, and for multiset T taking possible values in $\mathcal{X}$ we have the usual definition of $\text{Entropy}(T) = -\sum_{x \in \mathcal{X}} \frac{|T_x|}{|T|} \log_2 \frac{|T_x|}{|T|}$. The first term of the information gain, the entropy of the training set, corresponds to the impurity of the examples' labels. A pure set with only one repeated label has zero entropy, while a set having half positive examples and half negative examples has a maximum entropy of one. The information gain's second term corresponds to the expected entropy of the training set conditioned on the value of feature F. Thus a feature having a *high* information gain corresponds to a large drop in entropy, meaning that splitting on that feature resulting in a partition of the training set into subsets of like labels. A *low* (necessarily non-negative) information gain corresponds to a feature that is not predictive of class label.

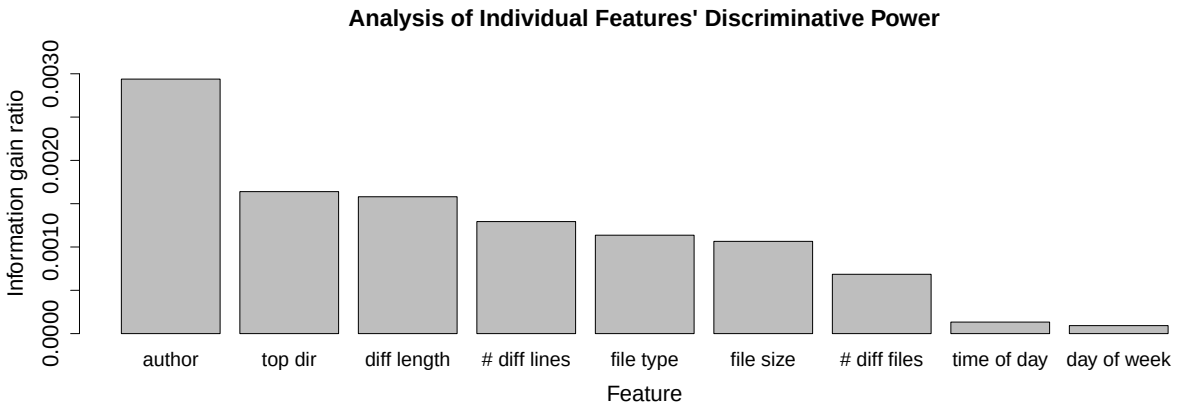


Figure 6.8: The features ordered by decreasing ability to discriminate between security and non-security patches, as represented by the information gain ratio.

Two issues require modification of the basic information gain before use in practice (Mitchell, 1997). The first is that nominal features F with large numbers of discrete values $|\mathcal{X}_F|$ tend to have artificially inflated information gains (being as high as $\log_2 |\mathcal{X}_F|$) since splitting on such features can lead to numerous small partitions of the training set with trivially pure labels. An example is the author feature, which has close to 500 values. In such cases it is common practice to correct for this artificial inflation by using the *information gain ratio* (Quinlan, 1986) as defined below. We use S_F to denote the multiset of examples' feature F values in the ratio's denominator, which is known as the *split information*.

$$\text{GainRatio}(S, F) \quad = \quad \frac{\text{Gain}(S, F)}{\text{Entropy}(S_F)} \quad . \tag{6.7}$$

The second issue comes from taking the idea of many-valued nominal features to the extreme: continuous features such as the diff length (of which there are 7,572 unique values out of 14,541 patches in our dataset) and the file size (which enjoys 12,795 unique values) are analyzed by forming a *virtual binary feature* for each possible threshold on the feature. The information gain (ratio) of a continuous feature is defined as the maximum information gain (ratio) of any induced virtual binary feature (Fayyad, 1992).

The results of our feature analysis are presented in Figures 6.8–6.11. The individual features' abilities to discriminate between non-security and security patches, as measured by the information gain ratio, are recorded in Figure 6.8. For the nominal features—author, top-level directory, file type, and day of week—we compute the information gain ratios directly, whereas for the continuous features—diff length, number of lines in diff, file size, number of files, and time of day—we use the information gain ratio given by choosing the best threshold value. The author feature has the most discriminative power, providing an information gain ratio 1.8 times larger than the next most informative feature. The next two most discriminative features are top-level directory and diff length, which enjoy similar

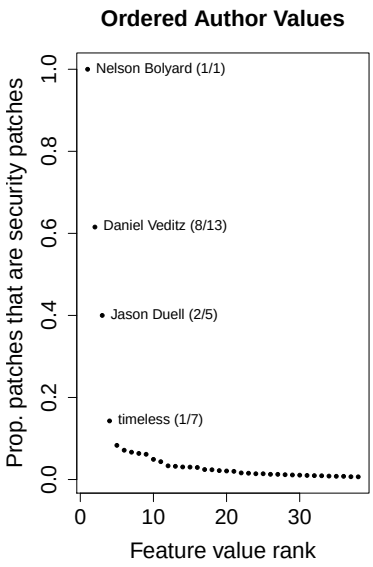


Figure 6.9: The authors who committed security patches, ordered by proportion of patches that are security patches. Top four authors are identified with their security and total patches along-side.

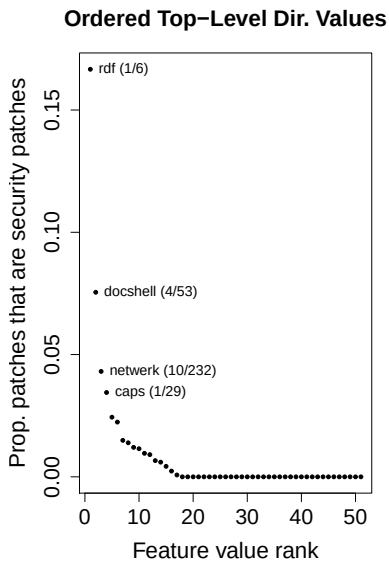


Figure 6.10: The top-level directories ordered by the proportion of patches that are security patches The top four directories are identified with their security and total patches along-side.

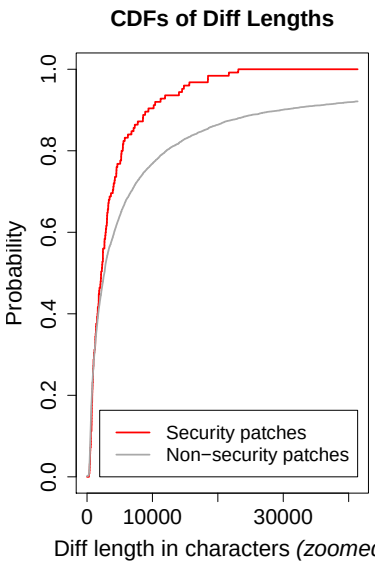


Figure 6.11: The CDFs of the security and non-security diff lengths. The figure is "zoomed" in on the left, with the top 1,000 largest lengths not shown.

gain ratios. The remainder of the patch size features and the file type have smaller ratios, and the two temporal features individually contribute significantly less information.

Furthermore, we show that each individual feature alone does not provide significantly high discriminative power. This observation follows from the maximum information gain ratio (belonging to author) being tiny: 3×10^{-3}. To add credence to these numbers, we note also that the unnormalized information gains have a similar ordering with the author feature coming out on top with an information gain of 2×10^{-2}, which corresponds to a small change in entropy. To summarize, we offer the following remark.

Remark 57. *Some features provide discriminative power for separating security patches from non-security patches, with author, top-level directory, and diff length among the most discriminative. However, individually, no feature provides significant discriminative power for separating security patches from non-security patches.*

Analysis of Discriminating Features. To give intuition why some features provide discriminating power for security vs. non-security patches, we analyze in more detail the three most discriminative features: the author, top-level directory, and diff length.

For the author and top-level directory features, we analyze their influential feature values. For each occurring feature value, we compute its *proportion value*: the number of patches with that feature value and are security sensitive divided by the total number of patches with that feature value. Figures 6.9 and 6.10 depict the influential feature values for the authors and top-level directory features by ranking the feature values by their proportion values. During the life-cycle of Firefox 3, a total of 516 authors contributed patches out of which 38 contributed at least one security patch. In Figure 6.9, we show only the 38 authors who wrote at least one security patch and omit the remaining 478 authors who did not write any security patches. Notice that the top four authors (labeled) are all members of the Mozilla Security Group (Veditz, 2010).

To explore the third most individually discriminative feature, the continuous diff length, we plot the feature CDFs for security and non-security patches in Figure 6.11. When all diff lengths are displayed, the CDFs resemble step-functions because the diff length distribution is extremely tail heavy. Figure 6.11 zooms in on the left portion of the CDF by plotting the curves for the first $6,500$ (out of $7,500$) unique diff lengths. From the relative positions of the CDFs, we observe that security patches have shorter diff lengths than non-security patches. This matches our expectations that patches that add features to Firefox require larger diffs.

Although our feature analysis identifies features that are individually more discriminative than others, the analysis also shows that no individual feature effectively predicts whether a patch is security-related. This observation demonstrates that motivated attackers should look to more sophisticated statistical techniques (such as an SVM) to reduce the effort required to find security patches and suggests that obfuscating individual features will not plug information leaks in the open-source life-cycle. Moreover certain features cannot be effectively obfuscated. For example, the Mozilla Committer's Agreement would be violated if developer names were redacted from `mozilla-central`.

6.5.2 Classifier Performance

Figure 6.12 depicts the time series of scores assigned to each patch by the SVM (the horizontal axis shows the day when each patch is landed in the repository starting at 2008-09-24; at which point security patches were first announced). Note that as mentioned in Section 6.4.2, we use an online learning approach, so the score assigned to a patch is only computed using the SVM trained with labeled patches seen up to the most recent security update before the day the patch is landed in the repository (and including any out-of-release delayed disclosures occurring before the patch is landed). Notice that the scores for security and non-security patches in the first 50 days are quite similar. Over time, the SVM learns to assign high scores to a handful of vulnerability fixes (and a few non-security patches) and low scores to a handful of non-security patches. However, many patches are assigned very similar scores of around 0.01 irrespective of whether they fix vulnerabilities.

Viewed as a binary classifier, the SVM performs quite poorly because there is no sharp threshold that divides security patches from non-security patches. However, when viewed in terms of the attacker's utility, the SVM might still be useful in reducing effort because

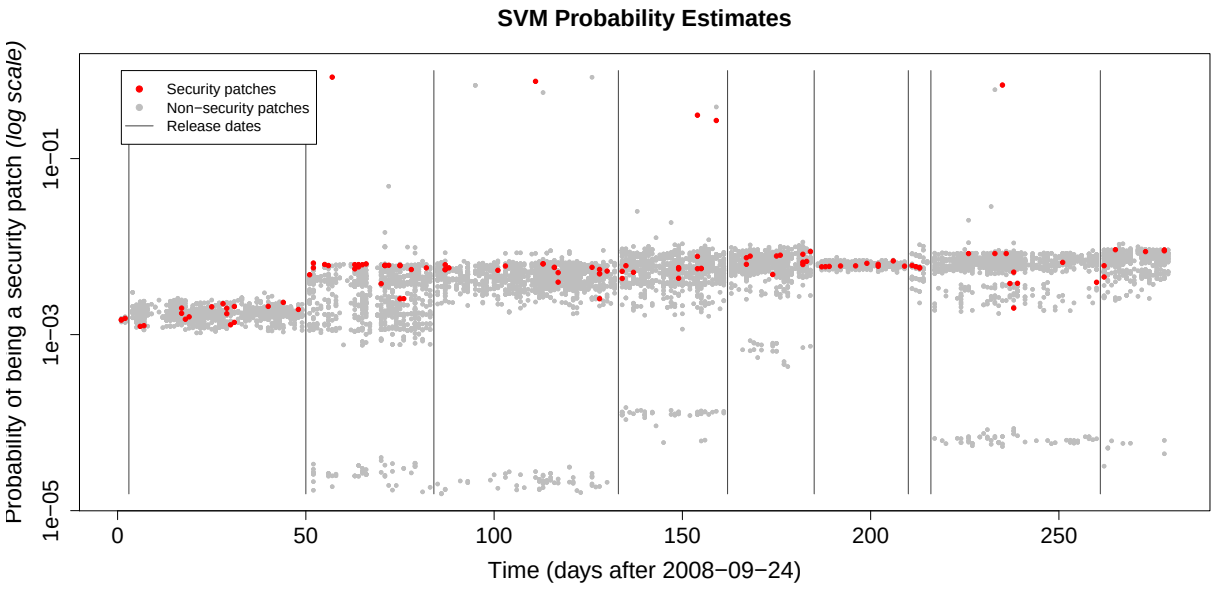


Figure 6.12: The time series of SVM probability estimates, with security patches and non-security patches delineated by color.

the *relative* rankings of the vulnerability fixes are generally higher than most non-security patches. We make this notion precise in the next subsection.

Remark 58. *When used as a classifier, the SVM classifier performs poorly. However, attacker effort depends only on the relative patch rankings assigned by the detector and is not necessarily affected by poor absolute predictive performance.*

6.5.3 Cost-Benefit of SVM-Assisted Vulnerability Discovery

6.5.3.1 Attacker Effort

Figure 6.13 shows the time series of the effort the attacker expends to find a vulnerability (as measured by the number of patches the attacker examines), as described in Section 6.3.2.1. The attacker effort measured for a given day is computed to reflect the following estimate. Imagine, for example, an attacker who "wakes up" on a given day, trains an SVM on publicly available information (including all labeled patches before the current day), and then starts looking for security patches among all the (unlabeled) patches landed in the repository since the most recent security update, in rank order provided by the SVM. Then the attacker effort measured for a given day is the number of patches that the attacker has to examine before finding a security patch using the rank order provided by the SVM.

Each continuous segment in the graph corresponds to one of the 12 security updates during our study. For a period of time after each release, there are no security patches in `mozilla-central`, which is represented on the graph as a gap between the segments.

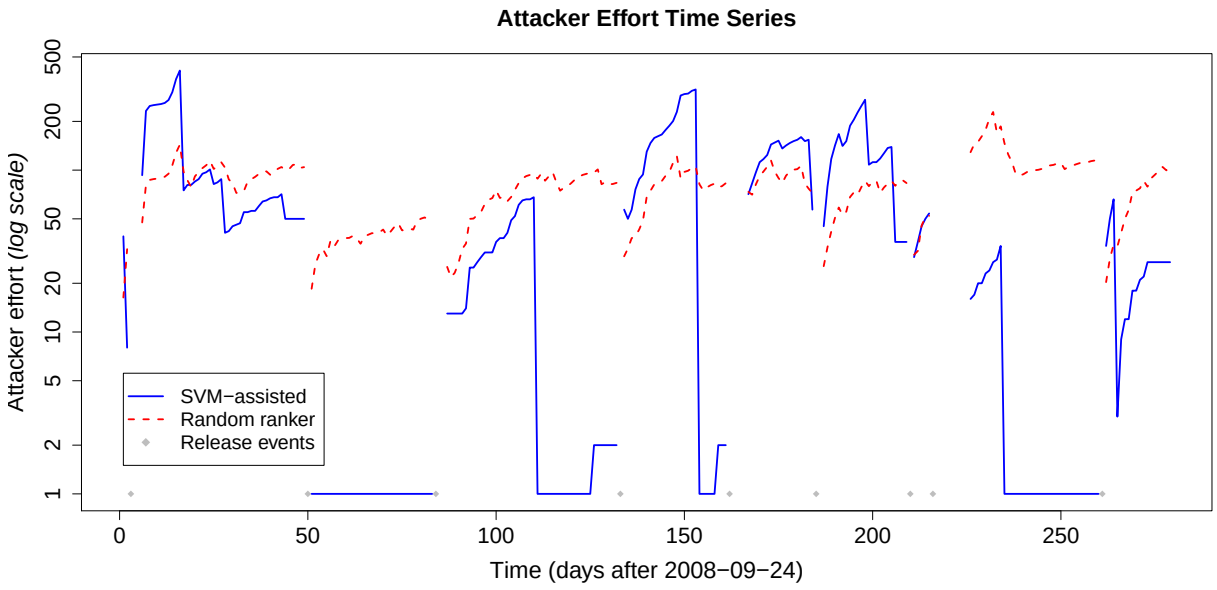


Figure 6.13: The attacker effort (number of patches to check before finding a vulnerability) of the SVM, and the expected attacker effort of the random ranker, as a function of time.

For the first 50 days of the experiment both the random ranker and the SVM-assisted attacker expend relatively large amounts of effort to find security patches. This poor initial performance of the SVM, also observed in Figure 6.12, is due to insufficient training. The SVM, like any statistical estimator, requires enough data with which to generalize.

Remark 59. *During the latter* $2/3^{rds}$ *of the year (the 8 month period starting 50 days after 2008-09-24) the SVM-assisted attacker, now with enough training data, regularly expends significantly less effort than an attacker who examines patches in a random order.*

Note. Given the "warm-up" effects of the first 50 days when the SVM has insufficient training data, all non-time-series figures in the sequel are shown using the data after 2008-11-13.

The general cyclic trends of the SVM-assisted and random rankers are also noteworthy. In most inter-update periods, the random ranker enjoys a relatively low attacker effort (though higher than the SVM's) which quickly increases. The reason for this behavior can be understood by plotting the expected effort for the random ranker with respect to the number of security patches for various total patch pool sizes as shown in Figure 6.4. Immediately after the landing of a first post-update security patch, the pool of available patches gets swamped by non-security patches (*cf.* Figure 6.2), corresponding to increasing n in Figure 6.4 and greatly increasing the expectation. Further landings of security patches are few and far between (by virtue of the rarity of such patches), and so moving across the figure with increasing n_s is rare. As the periods progress, non-security patches continue to

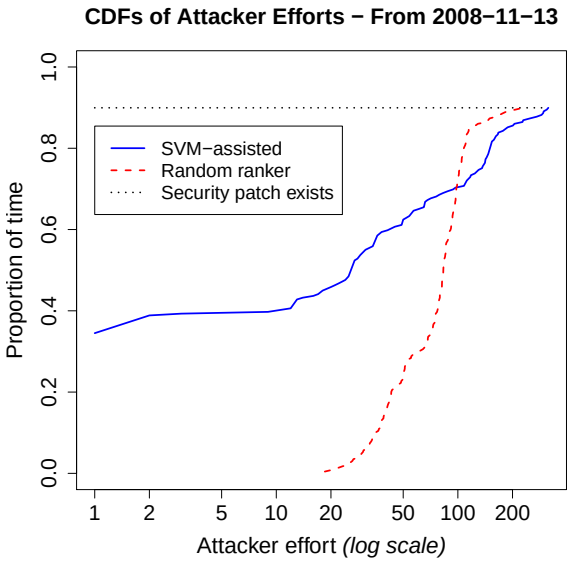


Figure 6.14: The cumulative distribution functions of the attacker effort displayed in Figure 6.13, from 11/13/2008 onwards. CDFs are shown for both the SVM and the random ranker.

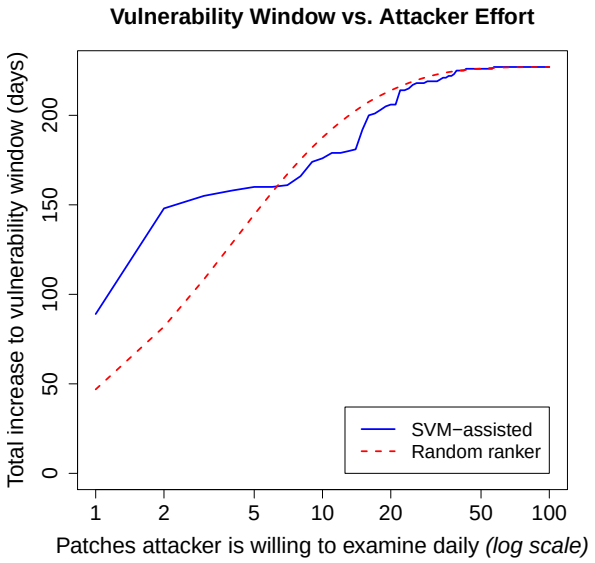


Figure 6.15: The total increase to the vulnerability window size throughout the year, for a given level of daily attacker effort with or without SVM assistance. Results trimmed to 11/13/2008 onwards.

swamp security patches. This trend for the random ranker's expected effort is more directly seen in Figure 6.5, which plots expected effort over an prototypical cycle of Firefox 3. Over the single 31 day cycle, 39 patches land daily of which a constant proportion are security patches. The curve for 10^{-2} most closely represents Firefox 3 where the security patch rate is 0.0085 of the total patch rate. The trend observed empirically in Figure 6.13 matches both the overall shape and location of the predicted trend.

At times early on in the inter-release periods, the SVM-assisted attacker experiences the same upward trending effort, but eventually the developers land a security patch that resembles the training data. Given just one "easy" fix, the effort required of the SVM-assisted attacker plummets. In two cycles (and partially in two others) the SVM-assisted attacker must expend more effort than the random ranker. This is due to a combination of factors including the small rates of landing security patches which means that the only unreleased security patches may not resemble previous training data.

6.5.3.2 Proportion of Days of Successful Vulnerability Discovery

Figure 6.14 depicts the cumulative distribution function (CDF) of attacker effort (Figure 6.13), showing how often the SVM-assisted and random ranker can find a security patch as a function of effort. Note that the CDFs asymptote to 0.90 rather than 1.0 because `mozilla-central` did not contain any security patches during 10% of the 8 month period.

Remark 60. *The SVM-assisted attacker discovers a security patch with the first examined patch for* 34% *of the* 8 *month period. Moreover by examining only* 2 *patches, the SVM-assisted attacker can find security patches over* 39% *of the period.*

If the unassisted attacker expends the minimum effort of 18.5, it can only find security patches for less than 0.5% of the 8 month period. By contrast, an SVM-assisted attacker who examines 17 patches will find a security patch during 44% of the period. In order to find security patches for 22% of the 8 month period, the random ranker must examine on average up to 70.3 patches. *The SVM-assisted attacker achieves significantly greater benefit than an attacker who examines patches in random order, when small to moderate numbers of patches are examined (i.e., up to 100 patches).*

When examining 100 or more patches, the SVM-assisted and random rankers find security patches for similar proportions of the 8 month period, with the random ranker achieving slightly better performance.

6.5.3.3 Total Increase in the Window of Vulnerabilities

Although the CDF of attacker effort measures how hard the attacker must work in order to find a patch that fixes a vulnerability, the CDF does not measure how valuable that vulnerability is to an attacker. In Figure 6.15, we estimate the value of discovering a vulnerability by measuring the total increase in the window of vulnerability gained by an attacker who expends a given amount of effort each day (as described in Section 6.3.2.2). Note that this differs from the previous section by considering an attacker who aggregates work over multiple days, and who does not re-examine patches from day-to-day.

Remark 61. *At* 1 *or* 2 *patches examined daily over the* 8 *month period, the SVM-assisted attacker increases the window of vulnerability by* 89 *or* 148 *days total, respectively. By contrast the random ranker must examine* 3 *or* 7 *patches a day (roughly* 3 *times the work) to achieve the approximate same benefit. At small budgets of* 1 *or* 2 *patches daily, the random ranker achieves window increases of* 47 *or* 82 *days which are just over half the SVM-assisted attacker's benefits. At higher daily budgets of 7 patches or more, the two attackers achieve very similar benefits with the random ranker's being slightly (insignificantly) greater.*

Compared to the Firefox 3 base-line vulnerability window size of 3.4 days (see Section 6.3.2.2), the increases to window size of 89 and 148 represent multiplicative increases by factors of 3.9 and 6.4 respectively.

6.5.3.4 In Search of Severe Vulnerabilities

Thus far, we have treated all vulnerabilities equally. In reality, attackers prefer to exploit higher severity vulnerabilities because those vulnerabilities let the attacker gain more control over the user's system. To evaluate how well the attacker fairs at finding severe vulnerabilities—those labeled as either "high" or "critical" in impact (Adamski, 2009)—we measure the attacker effort required to find the first high or critical vulnerability (that is, we ignore "low" and "moderate" vulnerabilities). Note that we did not re-train the SVM

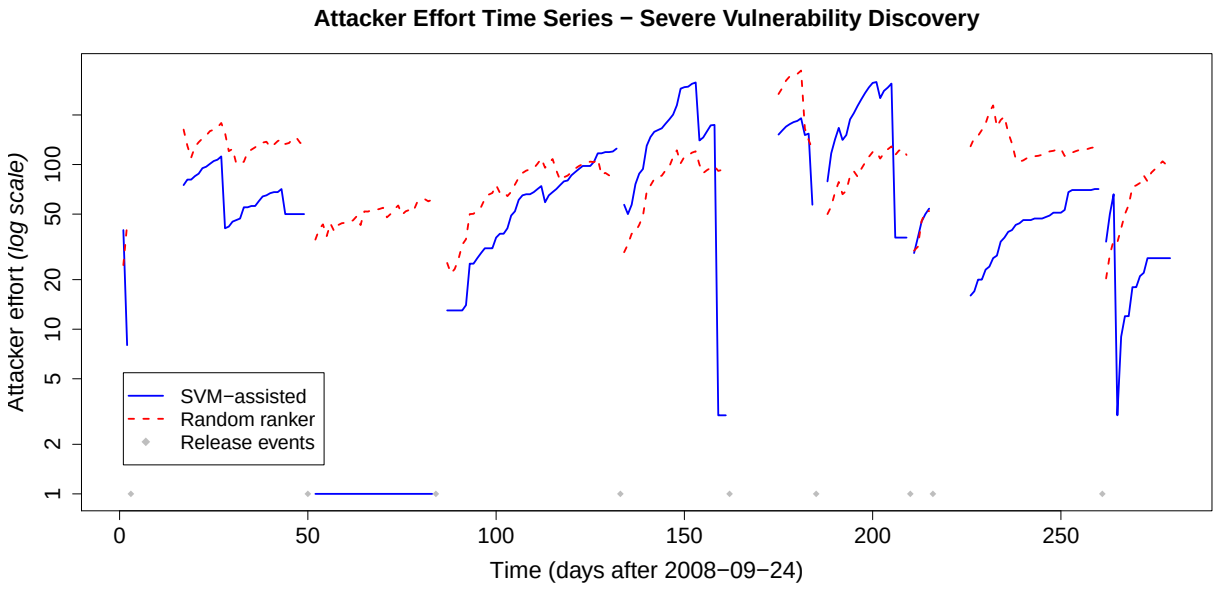


Figure 6.16: The time series of SVM-assisted and random ranker effort for finding severe (high or critical level) vulnerabilities.

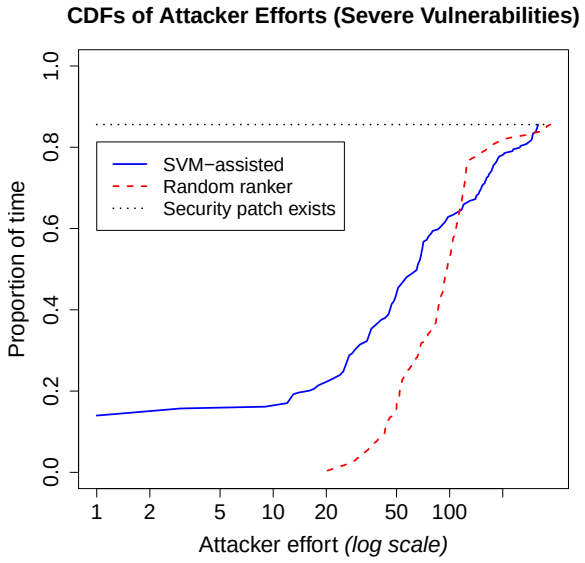


Figure 6.17: The CDFs of the SVM-assisted and random ranker efforts for discovering severe vulnerabilities, from 11/13/2008 onwards.

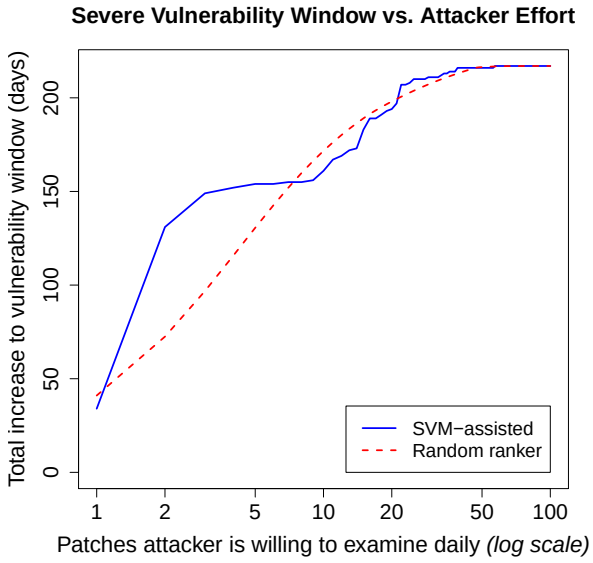


Figure 6.18: Total increase to the vulnerability window for finding severe vulnerabilities given levels of daily attacker effort, from 11/13/2008 onwards.

on severe vulnerabilities even though re-training could lead to better results for the special

case of discovering high-severity vulnerabilities. Figures 6.16–6.18 present our results for finding severe vulnerability fixes. The attacker effort time series for the SVM-assisted and random ranker is displayed in Figure 6.16. Overall, attacker effort curves are similar for all vulnerabilities, just shifted upwards away from 1 during several inter-update periods.

We can interpret the effect of focusing on severe vulnerabilities by examining the attacker effort CDFs in Figure 6.17. Although both attackers asymptote to the lower proportion of the period containing severe vulnerability fixes (down from 90% for identifying arbitrary vulnerabilities to 86%), only the random ranker's CDF is otherwise relatively unchanged. The random ranker's minimum effort has increased from 18.5 to 20.3 patches with a similarly low probability. The SVM-assisted attacker CDF undergoes a more drastic change. Examining one patch results in a vulnerability for 14% of the 8 month period, whereas an effort of 6 and 21 produce vulnerabilities for 20% and 34% of the 8 month period, respectively. To achieve these three proportions the random ranker must examine 48, 52, and 76 patches respectively.

Remark 62. *The SVM-assisted attacker is still able to outperform the random ranker in finding severe vulnerabilities, in particular finding such security fixes* 20% *of the time by examining* 6 *patches.*

The increases to the severe vulnerability window are shown for the two attackers in Figure 6.18. Again, we see a shift, with the SVM-assisted attacker continuing to outperform the random ranker on small budgets (except for a budget of 1 patch) or otherwise perform similarly.

Remark 63. *By examining* 2 *patches daily during the* 8 *month period, the SVM-assisted attacker increases the vulnerability window by* 131 *days. By contrast the random ranker with budget* 2 *achieves an expected window increase of* 72 *days.*

6.5.3.5 When One is Not Enough: Finding Multiple Vulnerabilities

An attacker searching for security patches might suffer from false negatives: the attacker might mistakenly take a security patch as a non-security patch. In practice, an attacker may wish to examine more patches than represented by the attacker effort defined above. To model this situation, we considered the problem of finding 2 or 3 security patches instead of just one.

As depicted in Figures 6.19–6.21, finding 1, 2, or 3 security patches requires progressively more effort. When computing the increase to the window of vulnerabilities in Figure 6.21, we assume that the attacker's analysis of the examined patches only turns up the 1^{st}, 2^{nd} and 3^{rd} security fixes respectively. To find 2 or 3 security patches over 34% of the 8 month period, the SVM-assisted attacker must examine 35 or 36 patches respectively.

Finally consider approximating the window of vulnerability achieved by an attacker examining a single patch daily with no false negatives. Examining 3 patches a day increases the total vulnerability window by 83 days even if the attacker's analysis produces one false negative each day. Assuming two false negatives each day, examining 4 patches daily increases the window by 80 days total. Similarly increasing the window by 151 or 148 days,

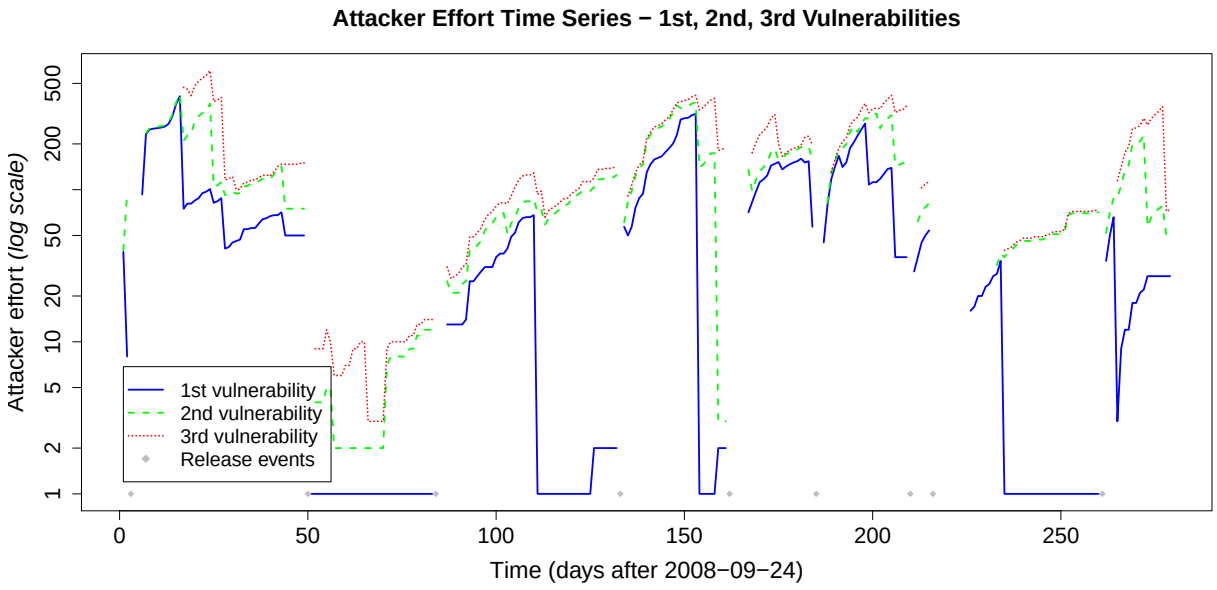


Figure 6.19: The time series of SVM-assisted ranker effort for finding 1, 2 or 3 vulnerabilities.

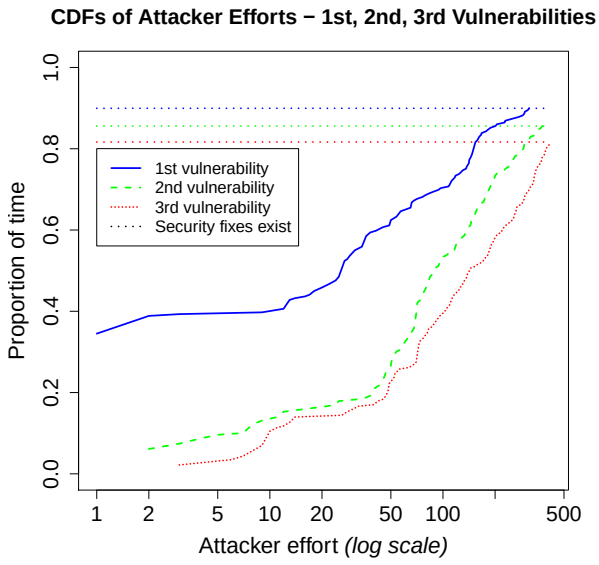


Figure 6.20: The CDFs of the SVM-assisted efforts for discovering 1, 2 or 3 vulnerabilities, from 11/13/2008 onwards.

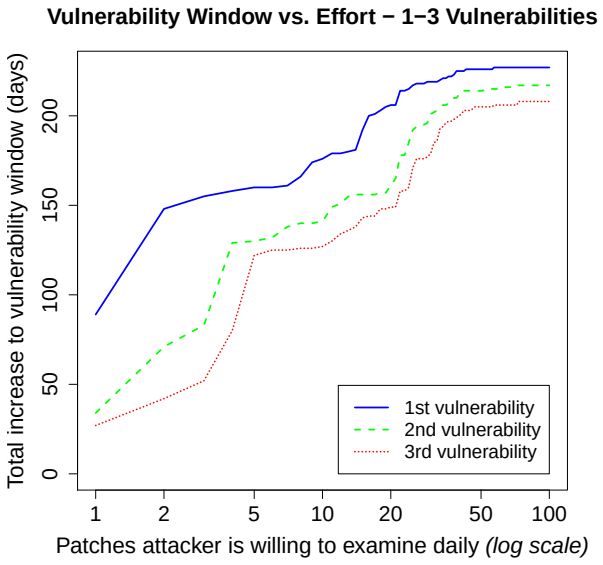


Figure 6.21: Total increase to the vulnerability window for finding 1, 2 or 3 vulnerabilities given levels of daily attacker effort, from 11/13/2008 onwards.

approximating the error-free result under a two patch per day budget, requires examining 12 or 18 patches daily when suffering one or two false negatives respectively.

6.5.4 Repeatability of Results Over Independent Periods of Time

In the above sections we explore how an attacker can find vulnerabilities over the lifetime of a major release of a large open-source project. It is natural to ask: how repeatable are these results over subsequent releases? As a first step towards answering this question, we repeat our analysis on the complete life-cycle of Firefox 3.5.

In order to test the hypothesis of repeatability of our results for the SVM-assisted and random rankers on other releases of Firefox, we repeated our analysis of Firefox 3, on Firefox 3.5. We again focus on `mozilla-central`, cloning the entire repository to identify patches landed during the Firefox 3.5 life-cycle. Firefox 3.5 was released June 30, 2009 and remained active until the release of Firefox 3.6 on January 21, 2010. During this 6 month period 7 minor releases to Firefox were made ending with Firefox 3.5.7 on January 5, 2010. We consider patches landed between the release of Firefox 3.5.7 and Firefox 3.6, whose identities as security patches or non-security patches were disclosed February 17, 2010 upon the release of Firefox 3.5.8. During the 6 month period, 7,033 patches were landed of which 54 fixed vulnerabilities.

While the Firefox 3.5 patch volumes correspond to roughly half those of the year-long period of active development on Firefox 3, it is possible that the patches' metadata may have changed subtly, resulting in significant differences in SVM-assisted ranker performance. Changes to contributing authors, functions of top-level directories, diff sizes or other side-effects of changes to coding policies, time of day or day of week when patches tend to be landed, could each contribute to changes to the attacker's performance. Given the similar rates of patch landings, one can expect the random ranker's performance to be generally comparable to the Firefox 3 results.

Figures 6.22–6.24 depict the cost-benefit analysis of the SVM-assisted and random rankers searching for vulnerabilities in Firefox 3.5. It is immediately clear that the same kind of performance is enjoyed by the attackers as achieved for Firefox 3, if not slightly better.The CDFs of attacker effort displayed in Figure 6.23 show that while the random ranker's performance is roughly the same as before, the SVM-assisted ranker's performance at very low effort (1 or 2 patches) is inferior compared to Firefox 3, while the assisted attacker enjoys much better performance at low to moderate efforts.

Remark 64. *The SVM-assisted attacker discovers a security patch in Firefox 3.5 by the third patch examined, for 22% of the 5.5 month period; by the* 20^{th} *patch the SVM-assisted attacker finds a security patch for 50% of the period. By contrast the random ranker must examine* 69.1 *or* 95 *patches in expectation to find a security patch for these proportions of the 5.5 month period.*

In a similar vein, the increase to the window of vulnerability achieved by the random ranker is comparable between Firefox 3 and 3.5 (correcting for the differences in release lifetimes), while the SVM-assisted attacker achieves superior performance (*cf.* Figure 6.24).

Remark 65. *By examining one or two patches daily, the SVM-assisted ranker increases the window of vulnerability (in aggregate) by 97 or 113 days total (representing increases to the base vulnerability window of factors of 5.8 and 6.7 respectively). By contrast the random ranker achieves increases of 25.1 or 43.8 days total under the same budgets.*

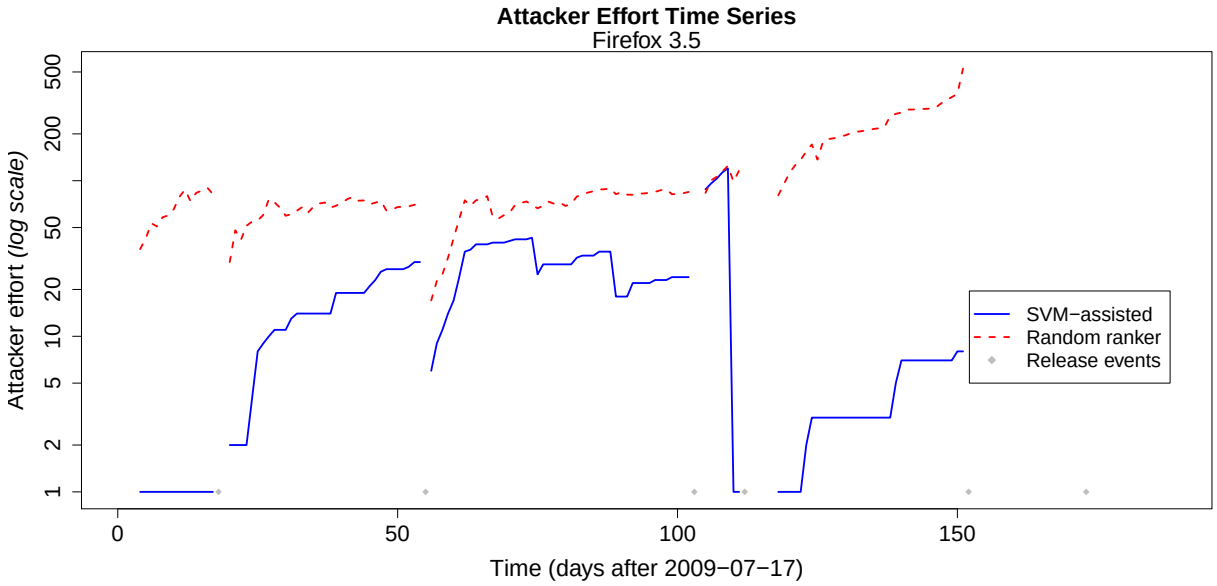


Figure 6.22: SVM-assisted and random ranker efforts for finding Firefox 3.5 vulnerabilities.

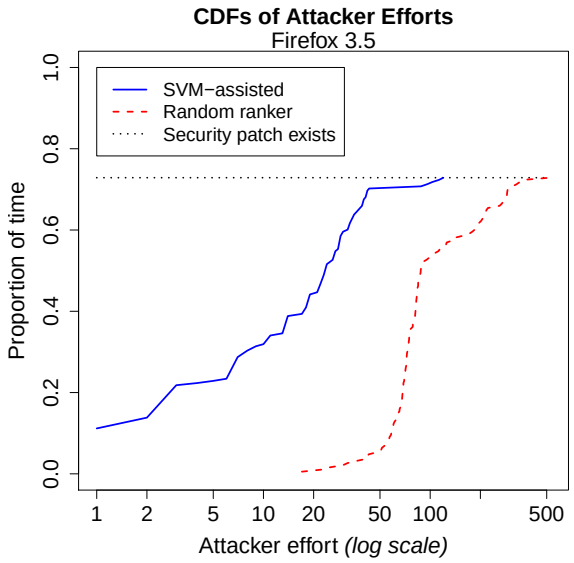


Figure 6.23: The CDFs of the SVM-assisted and random ranker attacker efforts, for Firefox 3.5.

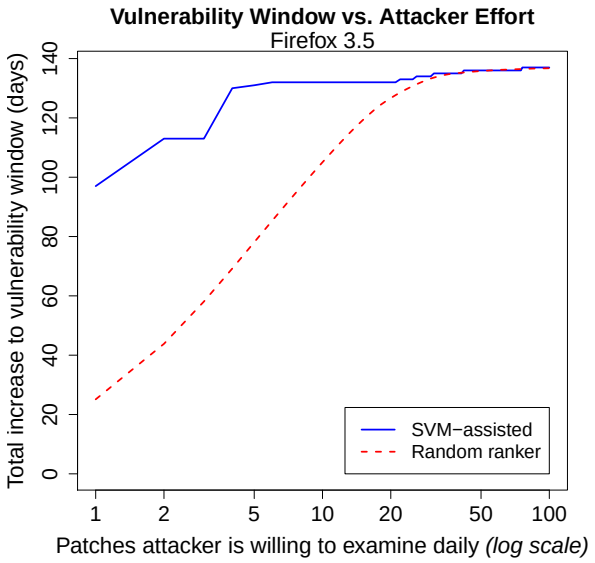


Figure 6.24: Increase to the total window of vulnerability achieved for varying levels of daily attacker effort, for Firefox 3.5.

We may conclude from these results that *the presented attacks on Firefox 3 are repeatable for Firefox 3.5*, and we expect our analysis to extend to other major releases of Firefox and major open-source projects other than Firefox.

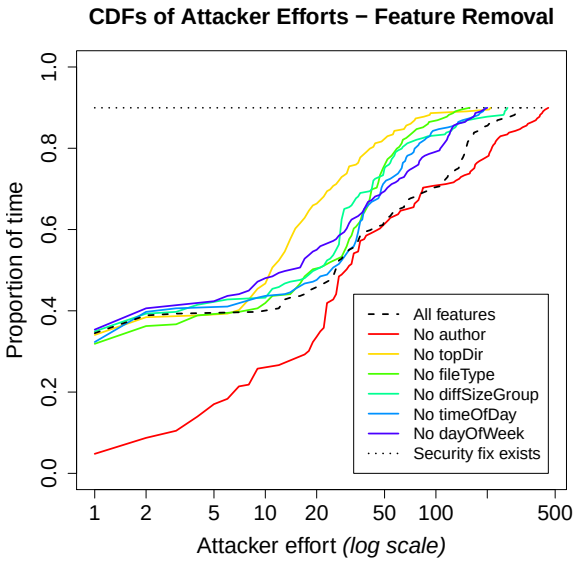


Figure 6.25: The effect of removing individual features on the SVM-assisted attacker effort CDFs, from 11/13/2008 onwards.

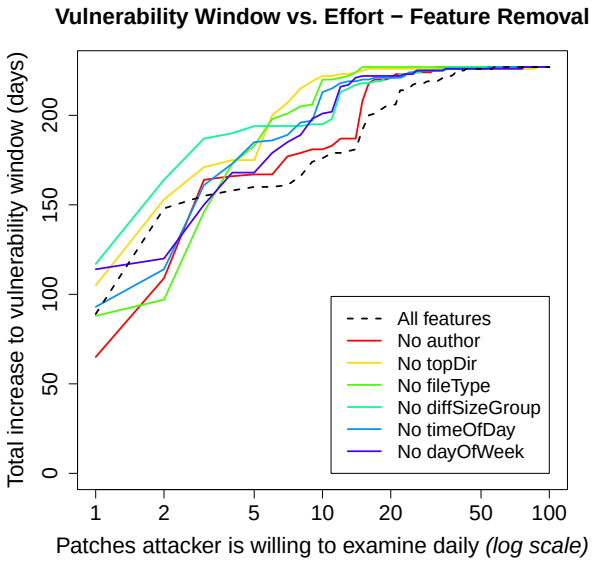


Figure 6.26: The effect of removing features on the SVM-assisted increase to the vulnerability window, from 11/13/2008 onwards.

6.5.5 Feature Analysis Redux: the Effect of Obfuscation

In Section 6.5.1 we perform a filter-based feature analysis for discriminating between security patches and non-security patches. In this section we ask: what is the effect of obfuscating individual features? We answer this question through a wrapper-based feature analysis in which we perform the same simulation of an SVM-assisted ranker as above, but now with one feature removed.

Figure 6.25 depicts the attacker effort CDFs for the SVM-assisted ranker when trained with all features, and trained with either the author, top directory, file type, time of day, day of week, or the set of diff size features removed. We remove the number of characters in the diff, number of lines in the diff, number of files in the diff, and file size simultaneously, since we observed no difference when only one of these features was removed. A plausible explanation for this invariance would be high correlation among these features. Removing the author feature has the most negative impact on the attacker effort CDF, reducing the proportion by 0.048 on average over attacker efforts in $[1, 315]$. That is, on average over attacker efforts for 5% of the 8 month period a security patch is found by the SVM-assisted ranker trained with the author feature while the attacker without access to patch author information find no security patch; and while the effect is most significant over attacker efforts in $[1, 10]$, the performance of the SVM-assisted ranker without the author feature is still strong in this range. Removing the file type, time of day, diff size, day of week, or top directory have increasingly positive impacts on the overall attacker effort CDF. Despite `libsvm`'s use of cross-validation for tuning the SVM's parameters, the positive improvements

point to overfitting which could be a product of the high dimensionality of the learning problem together with a very small sample of security patches: as noted above, our goal is merely to lower bound the performance of an attacker assisted by machine learning.

The increase to the window of vulnerability achieved by an SVM-assisted attacker without access to individual features (or the group of diff size features) is shown in Figure 6.26. For some attacker efforts the increase is less without certain features, but overall we see a more positive effect. The least positive effect is observed when removing the author feature: the increase in window size is only 5 days more on average over attacker efforts in $[1, 57]$ than when the author feature is included.

We thus draw the following conclusion, which agrees with the filter-based feature analysis presented in Section 6.5.1.

Remark 66. *Obfuscating the patch author has the greatest negative impact on the SVM-assisted ranker's performance, relative to obfuscating other features individually. However the magnitude of impact is negligible.*

As noted above, even if the impact of obfuscating patch authors were greater, doing so would violate the Mozilla Committer's Agreement.

6.6 Improving the Security Life-Cycle

In this section, we explore ways in which open-source projects can avoid information leaks in their security life-cycle. Instead of attempting to obfuscate the features an attacker could use to find security patches, we recommend that the developers land vulnerability fixes in a "private" repository and use a set of trusted testers to ensure the quality of releases.

6.6.1 Workflow

A natural reaction to our experiment is to attempt to plug the information leaks by obfuscating patches. However, we argue that this approach does not scale well enough to prevent a sophisticated attacker from detecting security patches before announcement because an attacker can use standard machine learning techniques to aggregate information from a number of weak indicators. In general, it is difficult to predict how such a "cat-and-mouse" game would play out, but, in this case, the attacker appears to have significant advantage over the defender.

Instead of trying to plug each information leak individually, we recommend re-organizing the vulnerability life-cycle to prevent information about vulnerabilities from flowing to the public (regardless of how well the information is obfuscated). Instead of landing security patches in the public `mozilla-central` repository first, we propose landing them in a private release branch. This release branch can then be made public (and the security patches merged into the public repository) on the day the patch is deployed to users. This workflow reverses the usual integration path by merging security fixes from the release branch to `mozilla-central` instead of from `mozilla-central` to the release branch.

6.6.2 Quality Assurance

The main cost of landing security patches later is that the patches receive less testing before release. When the Firefox developers land security patches in `mozilla-central`, those patches are tested by a large number of users who run nightly builds of Firefox. If a security patch causes a regression (for example, a crash), these users can report the issue to the Firefox developers before the patch is deployed to all users. The Firefox developers can then iterate on the patch and improve the quality of security updates (thereby making it less costly for users to apply security updates as soon as they are available).

Instead of having the public at large test security updates prior to release, we recommend that testing be limited to a set of trusted testers. Ideally, this set of trusted testers would be vetted by members of the security team and potentially sign a non-disclosure agreement regarding the contents of security updates. The size of the trusted tester pool is a trade-off between test coverage and the ease with which an attacker can infiltrate the program, which is a risk management decision.

6.6.3 Residual Risks

There are two residual risks with this approach. First, the bug report itself still leaks some amount of information because the bug is assigned a sequential bug number that the attacker can probe to determine when a security bug was filed. This information leak seems fairly innocuous. Second, the process leaks information about security fixes on the day the patch becomes available. This leak is problematic because not all users are updated instantaneously (Duebendorfer and Frei, 2009). However, disclosing the source code contained in each release is required by many open-source licenses. As a practical matter, source patches are easier to analyze than binary-only patches, but attackers can reverse engineer vulnerabilities from binaries alone (Brumley et al., 2008). One way to mitigate this risk is to update all users as quickly as possible (Duebendorfer and Frei, 2009).

6.7 Summary

Landing security patches in public source code repositories significantly increases the window of vulnerability of open source projects. Even though security patches are landed amid a cacophony of non-security patches, we show that an attacker can use off-the-shelf machine learning techniques to rank patches based on intrinsic metadata. Our results show that a handful of features are sufficient (in aggregate) to reduce the number of non-security patches an attacker need examine before encountering a security patch. For 22% of the period we study, the highest ranked patch actually fixes a security vulnerability. Because our algorithm establishes only a lower bound on attacker efficacy, it is likely that real attackers will be able to perform even better by considering more features or using more sophisticated detection algorithms.

A natural reaction to these findings is to obfuscate more features in an attempt to make the security patches harder to detect. However, our analysis shows that no single

feature contains much information about whether a patch fixes a vulnerability. Instead, the detection algorithm aggregates information from a number of weak signals to rank patches, suggesting that obfuscation is a losing battle. Instead of obfuscating patch metadata, we recommend changing the security life-cycle of open-source projects to avoid landing security fixes in public repositories. We suggest landing these fixes in private repositories and having a pool of trusted testers test security updates (rather than the public at large).

Our recommendations reduce the openness of open-source projects by withholding some patches from the community until the project is ready to release those patches to end users. However, open-source projects already recognize the need to withhold some security-sensitive information from the community (as evidenced by these projects limiting access to security bugs to a vetted security group). In a broad view, limiting access to the security patches themselves prior to release is a small price to pay to significantly reduce the window of vulnerability.

Chapter 7

Conclusions and Open Problems

As for the future, your task is not to foresee it, but to enable it.
– Antoine de Saint-Exupéry

Machine Learning, Statistics and Security stand to gain much from their cross-disciplinary research. On the one hand, many real-world security-sensitive systems are now using Machine Learning, opening up the possibility of new vulnerabilities due to *attacks on Machine Learning algorithms themselves*. Understanding the effects of various kinds of attacks on learners and designing algorithms for learning in adversarial environments are important first steps before users will trust 'black-box' adaptive systems. On the other hand, many defenses and even attacks on non-adaptive systems, can greatly benefit by leveraging the ability of Statistical Machine Learning based approaches to model both malicious and benign patterns in data.

This dissertation's main contributions lie in this intersection of Machine Learning, Statistics and Security. Viewing Machine Learning under a lens of Security and Privacy, Part I explores three kinds of attacks on adaptive systems in which an adversary can manipulate a learner by poisoning its training data, submit queries to a previously-trained learner in order to evade detection, or try to infer information about a learner's privacy-sensitive training data by observing models trained on that data. In the first and last cases, defenses are proposed that are either evaluated experimentally or analyzed theoretically to provide strong guarantees. In Part II Machine Learning is leveraged for building general defenses, and for constructing a specific attack on a non-adaptive software system. Again, either strong theoretical guarantees or extensive experimental evaluation demonstrate the significant gains made by using learning over non-adaptive approaches.

Chapter Organization. The remainder of this chapter summarizes the main contributions of this dissertation in greater detail in Section 7.1, and lists several open problems in the intersection of Machine Learning and Security in Section 7.2.

7.1 Summary of Contributions

The contributions of this dissertation span the spectrum of practical attacks on real systems, theoretical bounds, new algorithms, and thorough experimental evaluation. We detail these contributions to the state-of-the-art in Machine Learning and Computer Security research below.

7.1.1 Attacks on Learners

The taxonomy of attacks on Machine Learning systems of Barreno et al. (2006), amended with the attacker goal of breaching training data privacy in Section 1.2.2, considers adversaries aiming to affect one of three security violations: *Integrity* (False Negative events), *Availability* (False Positive events), and *Confidentiality* (unauthorized access to information), by either manipulating the training data (a *Causative* attack) or the learner's test data (an *Exploratory* attack). The three chapters of Part I consider Causative attacks, Exploratory attacks, and Confidentiality attacks respectively. In the two former cases, both Integrity and Availability goals are considered. And in the latter case, Confidentiality attacks may result from either Causative or Exploratory access to a learner.

Attacks that Poison the Training Data. Chapter 2 reports on two large experimental case-studies on Causative attacks, where the adversary manipulates the learner by poisoning its training data. In the first case-study on SpamBayes (Meyer and Whateley, 2004; Robinson, 2003), we construct poisoning attacks with the goal of increasing the False Positive rate of the open-source email spam filter, as a Denial of Service (DoS) attack on the learner itself. By contrast, previous work on attacking statistical spam filters has focussed on Exploratory attacks in which good words are inserted into spam messages (Lowd and Meek, 2005a; Wittel and Wu, 2004), or spammy words are replaced with synonyms (Karlberger et al., 2007). Our general approach is to send messages containing representative non-spam words to the victim. By flagging such messages as spam, the victim unwittingly trains SpamBayes to block legitimate mail. We study the effect of *adversarial information* and *control* on the effectiveness of our attacks. We experiment with attacks using knowledge of the victim's native language (English) by including an entire dictionary in the message, knowledge of the user's colloquialism's modeled by incorporating words from a Usenet newsgroup, or intimate knowledge of the tokens used to train the filter. Depending on the adversary's knowledge, the attack's spams can be better tailored to the filter or can be shorter in size. We also experiment with varying amounts of adversarial control over the training corpus through tuning the proportion of training messages that are attack spams. We determine, for example, that with only 1% control over the training corpus and intermediate knowledge of the non-attack training corpus, the filter's FPR can be increased to 40%. At this point most victim's would shut-off their filter, resulting in all subsequent spam messages being sent straight to the user's inbox. We also experiment with *Targeted* attacks in which the adversary's goal is to block specific legitimate messages. Here again we experiment with adversarial control and information through the level of knowledge the adversary possess

about the specific message's tokens. With knowledge of only 30% of the target message, the attack results in 60% of the target messages being incorrectly filtered. Finally we consider two defenses based on measuring the impact of new messages on the classifier's predictions before inclusion in training, and dynamic thresholds on the spam scores. Experimental results show these to be effective counter-measures against our Indiscriminate dictionary attacks.

In the second case-study of Chapter 2 we consider Integrity attacks that poison the training data of Principal Component Analysis (PCA) based network-wide volume anomaly detection, with the aim of increasing the False Negative Rate for evasion during test time. PCA became a popular tool in the Systems Measurement community for detecting DoS flows in networks based on (relatively cheap to monitor) link traffic volume measurements (Guavus, 2010; Lakhina et al., 2004a,b, 2005a,b; Narus, 2010). While it has recently been observed that PCA can in certain situations be sensitive to benign network faults (Ringberg et al., 2007), our study is the first to *quantify malicious tampering* of PCA. To combat PCA, which models the principal components in link space that capture the maximum variance in the training set, our *variance injection attacks* inject chaff into the network in such a way as to increase variance in a desired direction while minimally impacting traffic volume; the goal being to enable future evasion of the detector by manipulating its model of normal traffic patterns, while *poisoning covertly so that the manipulation itself is not caught.* We design attacks that exploit increasing adversarial information about the underlying network traffic: uninformed poisoning in which the attacker cannot monitor traffic and must add Bernoulli noise to the network, locally-informed poisoning in which the attacker can monitor the traffic passing along a single ingress link, and globally-informed poisoning where a worst-case attacker can monitor all links of the network. In each poisoning scheme, we include a parameter for tuning the adversary's control over the data in the form of the amount of traffic injected into the network. Experiments on a single week of training and a single week of test data show that, for example, the locally-informed scheme can increase the FNR seven-fold while increasing the mean traffic volume on the links of the target flow by only 10%. As in the case of our SpamBayes attacks, increased information or increased control hands the adversary a distinct advantage. We explore covert attacks on PCA in which the attacker slowly increases his amount of poison chaff over the course of several weeks. Even when allowing PCA to reject data from its training set, a 5% compound growth rate of traffic volume per week resulted in a 13-fold increase of the FNR to 50% over just a 3 week period. To counter our variance-injection attacks, we propose a detector based on Robust Statistics. ANTIDOTE selects its subspace using the PCA-GRID algorithm that maximizes the robust MAD estimator of scale instead of variance, and a new Laplace threshold. Experiments show that ANTIDOTE can halve the FNR due to the most realistic locally-informed poisoning scheme which maintaining the performance of PCA on un-poisoned data.

Attacks that Query a Classifier to Evade Detection. In this dissertation's second set of contributions on attacks on learners, Chapter 3 considers algorithms for querying previously-trained classifiers in order to find minimal-cost instances labeled negative by the classifier. In this theoretical study, we build on the abstract model of the *evasion*

problem due to Lowd and Meek (2005b): given query access to a classifier, a target positive instance $\mathbf{x}^{\mathrm{A}}$ and a cost function A measuring distance from $\mathbf{x}^{\mathrm{A}}$, our goal is to find a negative instance that almost-minimizes A while submitting only a small number of queries to the classifier. Lowd and Meek (2005b) showed that by querying to learn the decision boundary—what we call *reverse engineering*—an attacker can evade linear classifiers with near-minimal L_1 cost with query complexity $\mathcal{O}\left(D \log \frac{1}{\epsilon}\right)$ for feature space dimension D and a cost of factor $1 + \epsilon$ from optimal. Our work extends this result in several ways. We consider the much larger class of classifiers which partition feature space into two classes, one of which is convex; and we consider more general L_p cost functions. For $p \leq 1$ our multiline search algorithm can evade detection by classifiers with convex *positive* class using very low query complexity $\mathcal{O}\left(\log \frac{1}{\epsilon} + D\sqrt{\log \frac{1}{\epsilon}}\right)$ which improves on the result of Lowd and Meek (2005b) without reverse engineering the decision boundary. Moreover a new lower bound of $\mathcal{O}\left(\log \frac{1}{\epsilon} + D\right)$ shows that our query complexity is close to optimal. For the case of $p > 1$, evasion for convex positive classes takes exponential query complexity to achieve good approximations. This result is a threshold-type phenomenon where the p threshold for query complexity is at 1. For approximations that worsen with D our multiline search yields polynomial complexity solutions. For the case of $p \geq 1$ and classifiers with convex *negative* classes, we apply the geometric random walk-based method of Bertsimas and Vempala (2004) that tests for intersections between convex sets using a query oracle. In this case our randomized method has polynomial query complexity $\mathcal{O}^*\left(D^5 \log \frac{1}{\epsilon}\right)$. An important corollary of polynomial complexity for evading convex-inducing classifiers is that reverse engineering is sufficient but not necessary for evasion, and can be much harder (*i.e.*, reverse engineering convex-inducing classifiers requires exponential complexity for some classifiers).

Attacks that Violate Training Data Privacy. As this dissertation's third and final study on attacks on learners, Chapter 4 explores privacy-preserving learning in the setting where a statistician wishes to release a Support Vector Machine (SVM) classifier trained on a database of examples, without disclosing significant information about any individual example. We adopt the strong definition of β-differential privacy (Dwork, 2006) which allows an attacker knowledge and/or control over $n - 1$ of the n rows in the database, knowledge of the release mechanism's mapping, and access to the released classifier. Even in the presence of such an (arguably unrealistically) powerful adversary differential privacy guarantees the privacy of a single hidden training example, where lower β guarantees more privacy. The chapter includes two positive results in the form of mechanisms for SVM with finite feature spaces (*e.g.*, linear SVM and some nonlinear SVMs including the polynomial kernel with any degree) and nonlinear SVM with translation-invariant kernels inducing *infinite dimensional feature spaces* (*e.g.*, the RBF or Gaussian kernel). For both mechanisms we guarantee differential privacy given sufficient noise is added to the SVM's weight vector by the mechanism. For a new notion of utility, which states that a privacy-preserving mechanism's classifier makes predictions that closely match those of the original SVM (a property strictly stronger than good accuracy), we prove that both mechanisms are useful provided that not *too much* noise is added. Thus we quantify what is an intuitive trade-off between privacy and utility for the case of SVM learning. Finally we provide negative

results in the form of lower bounds that state that no mechanism that approximates the SVM well (*i.e.*, has very good utility) can have β-differential privacy for low values of β: *i.e.*, it is not possible to have very high levels of utility and privacy simultaneously. This work makes several contributions within the area of privacy-preserving learning and statistical databases. First our mechanisms respond with parametrizations of *functions* that can belong to classes of infinite VC-dimension; by contrast existing mechanisms parametrize scalars or vectors, or in a few cases relatively simple functions. Second the SVM does not reduce to simple subset-sum computations, and so does not fit within the most common technique for calculating sensitivity and proving differential privacy. Third the SVM is a particularly practical learning method, while in many previous studies the emphasis was on deep analysis of more simplistic learning methods. Finally our proofs draw new connections between privacy, algorithmic stability and large-scale learning.

7.1.2 Learning for Attack and Defense

Just as Security and Privacy offers potential benefits for improving applicability and understanding of Machine Learning methods in practice, Machine Learning can be leveraged to build effective defenses or to construct attacks on large software systems. Part II of this dissertation explores two case-studies in which Machine Learning can be used to build powerful defenses and attacks, greatly improving on the performance of related non-adaptive approaches.

Learning-Based Risk Management. Chapter 5 serves as a case-study in applying Machine Learning as a defense, where known guarantees from the learning literature have import consequences in Security. In the *CISO problem*, a Chief Information Security Officer (CISO) must allocate her security budget over her organization with the goal of minimizing the attacker's additive profit or multiplicative return on attack (ROA). We model this incentives-based risk management problem as a repeated game on a graph where nodes represent states that can be reached by the attacker (*e.g.*, root access to a server), and edges represent actions the attacker can take to reach new nodes (*e.g.*, exploit a buffer overflow). Upon each attack, the adversary chooses a subgraph to attack. From reaching certain nodes in the graph, the attacker can receive a payoff. On the other hand budget allocated by the CISO to edges result in a cost incurred by the attacker. Moreover some edges are more difficult to defend than others, in the sense that more budget must be allocated by the CISO in order to force the same cost on the adversary. By viewing the risk management problem as a repeated game, we model high-level organizations, systems, or country-level cyber-warfare battlefields in which attackers repeatedly attack the system causing damage to the organization but not critical organization-ending damage. We construct a *reactive risk manager* using the exponential weights algorithm from Online Learning Theory which learns to allocate budget according to past attacks. Using a reduction to known regret bounds from learning theory, we show that the average profit or ROA enjoyed by an adversary attacking the reactive defender approaches that of an adversary attacking *any* fixed strategy. In particular this includes the rational *proactive risk manager* that is aware of the vulnerabilities

(edges) in her organization, the valuation of the attacker, and uses this information to play a minimax strategy over the course of the game. By contrast, through a simple modification to the exponential weights algorithm, the reactive defender need not know any vulnerabilities (edges) before-hand. This result is at odds with the conventional security wisdom that reactive defense is akin to myopic bug chasing and is almost always inferior to proactive approaches. Moreover we show that in several *realistic* situations, the reactive defender vastly outperforms the fixed proactive defender. Example situations include when the proactive defender minimizes attacker profit instead of ROA (or vice versa), when the attacker is not rational, and when the proactive defender makes incorrect assumptions about the attacker's valuations.

Learning-Based Attacks on Open-Source Software. As detailed in Chapter 6, in order to effectively attack open-source software systems, we can use Machine Learning to exploit features of the source code that lands in public repositories long before users' systems are patched for vulnerabilities. We consider Firefox 3 as a representative open-source system, where security patches and non-security patches land in the project's public trunk in between (roughly) monthly releases of the project. Upon each minor release of Firefox, Mozilla retroactively discloses discovered vulnerabilities in the previous version of the software which are patched in the latest release. We use these disclosures to *label* the source code available in the repository up to the latest release. Using features of the source code change-sets such as author, time the patch landed in the repository, top-level directory of the project and file-types most effected by the diff, and diff size, we can train a discriminative model to differentiate between non-security and security patches. As new patches land in the repository we use the model to rank them according to the likelihood of fixing a vulnerability. An attacker using this ranking would then examine the patches by-hand (or perhaps with expensive program analysis) until a security patch is found. The benefit of this attack is that the window of vulnerability is extended *back in time* to the point at which the first security patch is found within an inter-release period. The attack's cost is simply the number of patches that must be examined to find a security patch. We use off-the-shelf software for Support Vector Machine (SVM) learning which requires no knowledge of the SVM, to learn to rank patches online throughout the year of active Firefox 3 development. We show that after a warm-up period, for 39% of the days within a span of 8 months the SVM-assisted attacker need only examine one or two patches to find a security patch. Moreover the same attacker, by examining the top two ranked patches every day during the same 8 month period extends her aggregate window of vulnerability (over all the inter-release periods during the 8 months) by 5 months. We compare these results to an unassisted attacker, who selects patches to examine uniformly at random. Finally we propose that Mozilla alter their vulnerability life-cycle by landing security patches in a private release branch instead of the public trunk. The cost of such a counter-measure (which can be mitigated by opening the private branch prior to the next release) is to quality assurance, since security patches would not be tested as thoroughly and bugs could be introduced by merging branches further into the process.

7.2 Open Problems

While the research presented here makes several contributions to Machine Learning and Security as described above, a number of new problems remain open.

7.2.1 Adversarial Information and Control

Section 1.3 defined the adversarial capabilities of information and control as the information available to the adversary regarding the learning map, learner state, feature space and benign data generation process, and the amount of control the adversary can exert over the learner's training or test data.

A common theme throughout this dissertation is that the level of adversarial information and control governing an adversary greatly affects the performance of that adversary's attacks on a learning system.

Similar conclusions can be made based on previous studies. For example, Kearns and Li (1993) characterized the effects of a β proportion of malicious noise on learning under the PAC model (Valiant, 1984). They also bound the maximum level $\beta^\star$ of malicious noise under which learnability is still possible. In Robust Statistics, the notion of break down point is similar, characterizing the proportion of a sample that can be arbitrarily corrupted without an estimator being forced to diverge to ∞. However, in these cases and many others including standard Online Learning Theory results, worst-case assumptions are made on the adversary. In Online Learning the adversary has complete information and control; and for breakdown points and in the work of Kearns and Li (1993), the data that is corrupted gets corrupted arbitrarily.

In reality, as is highlighted by the two case-studies of Chapter 2, useful threat models tend to limit the adversary's capabilities. As described in the examples of Section 1.3, the form of information or control possessed by the adversary can vary greatly from one adversarial domain to another. For example, in the email spam domain control corresponds to inserting messages of label spam into the training corpus without altering existing training messages. In the network intrusion domain, by contrast, control is most naturally viewed as adding volume to a small number of columns of the link traffic matrix (corresponding to injecting chaff into a single flow). Thus we may model these domains, and perhaps many others, by a benign data generation process whose output is corrupted according to some transformation $T \in \mathcal{T}_\theta$ before being revealed to the learner. In the above examples, $\mathcal{T}_\theta$ corresponds to the set of possible transformations the adversary could apply, having level of control θ. For example this may be injections of a proportion θ of spam messages, or addition of chaff of average volume θ to a flow matrix. The adversary (or perhaps an adversarial 'nature') selects which element of $\mathcal{T}$ is used. We might assume that the learner is aware of the family of transformations $\mathcal{T}_\theta$ but not the particular T.

One open question is to represent typical forms of adversarial control found in real-world applications of Machine Learning. Given such example $\mathcal{T}$'s we may ask how much control is too much.

Open Problem 67. *For a given transformation class $\mathcal{T}_\theta$, under what levels of control θ is*

learning possible?

We may further want to characterize which forms of control are tolerable and which are not.

Open Problem 68. *What combinatorial properties of transformation classes $\mathcal{T}$ characterize learnability under transformations $T \in \mathcal{T}$?*

In addition to control, we may wish to understand the fundamental benefits to the attacker of adversarial information. In this case one could form a repeated game as is typical in Online Learning. However again we could ask that benign data come from some data generation process. This time the adversary, prior to transforming the data, is given access to some limited view of the data $V \in \mathcal{V}$. For example in the email spam domain our attackers had access to a sample (of a Usenet newsgroup) drawn from a distribution like that which generated the training corpus. The level of information ϕ corresponded to the similarity between the adversary's and victim's generating distributions. In the network anomaly detection problem, the adversary may be able to view some subset of the link traffic matrix columns. The adversary may not know which view $V \in \mathcal{V}_\phi$ will be in effect, but she may have knowledge of $\mathcal{V}_\phi$ including an estimate of ϕ. Thus the adversary is given access to the image of the data under some $V \in \mathcal{V}_\phi$ after which she applies the transformation $T \in \mathcal{T}_\theta$ to the data. Of course this transformation (or the entire family) may depend on the information the attacker received. The learner is then revealed the corrupted data as before. We may ask similar questions to above.

Open Problem 69. *Under what levels of information ϕ is learning possible, for given information class $\mathcal{V}_\phi$?*

Open Problem 70. *What combinatorial properties of information class $\mathcal{V}_\phi$ characterize learnability? What about when the data revealed to the learner is also transformed by some $T \in \mathcal{T}_\theta$?*

Finally, it would be interesting to understand the relationship between information, control and learnability.

Open Problem 71. *What are the trade-offs between learnability, adversarial information, and adversarial control? Both for varying levels ϕ and θ, but also for different classes of information and control.*

These questions would make interesting extensions to the basic Online Learning Theory model, for example. Similar modifications to ideas of influence and breakdown points could also be conceivably made in Robust Statistics.

7.2.2 Covert Attacks

Several attacks on Machine Learning have been constructed in this dissertation: the Causative Availability attacks on the SpamBayes email spam filter, the Causative Integrity

attacks on the PCA-based network-wide anomaly detector, and the Exploratory attacks on convex-inducing classifiers. In each case, our motivation for limiting the attack intensity was implicitly related to desiring a covert attack. Stealthiness was explicitly measured in the case-study on poisoning PCA, in which during the *Boiling Frog* poisoning attacks, we allowed the model (PCA and ANTIDOTE) to reject traffic from being included in the training set. The lower the level of rejection, the more stealthy the level of poisoning. In the evasion problem, we motivated the secondary goal of low query complexity by the desire to be covert. Intuitively, given too many queries, the classifier may become suspicious that an attack is underway. An interesting direction for future research is attacks that are covert by design.

Open Problem 72. *What are good general strategies for designing attacks on learners that are covert?*

One way to view the stealthiness question is as a kind of complement of designing learners that have good worst-case performance. For example the reactive risk management strategy of Chapter 5 came with guarantees about its performance for *any* sequence of attacks. In this way we could be confident that it would perform well as a defense.

Open Problem 73. *Consider a learning-based defender that is periodically re-trained; during training, the previously learned model is used to reject training data that appears to be malicious. How can data poisoning attacks be designed with guarantees on stealthiness i.e., guarantees on a minimal level of poisoned data being included in the training set despite the learner's best efforts?*

We have used stealthiness to justify limited adversarial control in the learner's threat model. Conversely it is clear that some forms of adversarial control will be stealthier than others. On the other hand, those forms of control that are stealthy would likely have less of a manipulative effect on the learner.

Open Problem 74. *What trade-offs between stealthiness and learner manipulation are possible?*

7.2.3 Privacy-Preserving Learning

Chapter 4 drew several new connections between differential privacy and learning theory. To prove that our mechanisms preserve differential privacy we computed the global sensitivity of the SVM's primal solution using results from algorithmic stability.

Open Problem 75. *What other stable learning algorithms can be made to preserve differential privacy using techniques similar to our own?*

Various notions of stability have been shown to be necessary and/or sufficient for learnability (Bousquet and Elisseeff, 2002; Mukherjee et al., 2006). How could such results be connected to differential privacy?

Open Problem 76. *What is the relationship between algorithms that learn (e.g., are consistent) and those that can be made differentially private with minimal perturbations? In particular, can necessary notions of stability for learnability be used to achieve differential privacy?*

If the answer is positive, then learnability implies differential privacy (with minimal changes to the algorithm). Many similar questions to these exist, such as what are the fundamental trade-offs between privacy and utility? And how can Robust Statistics be related to differential privacy in general, extending the results of Dwork and Lei (2009)?

Bibliography

Nabil R. Adam and John C. Worthmann. Security-control methods for statistical databases: a comparative study. *ACM Computing Surveys*, 21(4):515–556, 1989.

Lucas Adamski. Security severity ratings, 2009. `https://wiki.mozilla.org/Security_Severity_Ratings` [Online; accessed 6-May-2010].

Alekh Agarwal, Peter Bartlett, and Max Dama. Optimal allocation strategies for the dark pool problem. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS'2010)*, volume 9 of *Journal of Machine Learning Research*, pages 9–16, 2010.

Eugene Agichtein, Eric Brill, and Susan Dumais. Improving web search ranking by incorporating user behavior information. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '06)*, pages 19–26, 2006.

Ross Anderson. Why information security is hard—An economic perspective. *Proceedings of the 17th Annual Computer Security Applications Conference (ACSAC '01)*, pages 358–365, 2001.

Jaime Arguello, Fernando Diaz, Jamie Callan, and Jean-Francois Crespo. Sources of evidence for vertical selection. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '09)*, pages 315–322, 2009.

Terrence August and Tunay I. Tunca. Network software security and user incentives. *Management Science*, 52(11):1703–1720, 2006.

Paramvir Bahl, Ranveer Chandra, Albert Greenberg, Srikanth Kandula, David A. Maltz, and Ming Zhang. Towards highly reliable enterprise network services via inference of multi-level dependencies. In *Proceedings of the ACM SIGCOMM 2007 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pages 13–24, 2007.

Pierre Baldi and Søren Brunak. *Bioinformatics: the machine learning approach*. MIT Press, Cambridge, MA, USA, 2001.

Boaz Barak, Kamalika Chaudhuri, Cynthia Dwork, Satyen Kale, Frank McSherry, and Kunal Talwar. Privacy, accuracy, and consistency too: a holistic solution to contingency table release. In *Proceedings of the Twenty-Sixth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS '07)*, pages 273–282, 2007.

Michael Barbaro and Tom Zeller Jr. A face is exposed for aol searcher no. 4417749. New York Times. Aug 9, 2006.

Marco Barreno, Blaine Nelson, Russel Sears, Anthony D. Joseph, and J. D. Tygar. Can machine learning be secure? In *Proceedings of the ACM Symposium on InformAtion, Computer and Communications Security (ASIACCS'06)*, pages 16–25, 2006.

Marco Barreno, Blaine Nelson, Anthony D. Joseph, and J. D. Tygar. The security of machine learning. *Machine Learning*, 2010. to appear.

Marco Antonio Barreno. Evaluating the security of machine learning algorithms. Dissertation UCB/EECS-2008-63, Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, 2008.

Chris Beard. Introducing Test Pilot, March 2008. `http://labs.mozilla.com/2008/03/introducing-test-pilot/` [Online; accessed 6-May-2010].

Dimitris Bertsimas and Santosh Vempala. Solving convex programs by random walks. *Journal of the ACM*, 51(4):540–556, 2004.

Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag, 2006.

Avrim Blum, Cynthia Dwork, Frank McSherry, and Kobbi Nissim. Practical privacy: the SuLQ framework. In *Proceedings of the Twenty-Fourth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS '05)*, pages 128–138, 2005.

Avrim Blum, Katrina Ligett, and Aaron Roth. A learning theory approach to non-interactive database privacy. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC '08)*, pages 609–618, 2008.

Peter Bodik, Rean Griffith, Charles Sutton, Armando Fox, Michael I. Jordan, and David A. Patterson. Statistical machine learning makes automatic control practical for internet datacenters. In *Proceedings of the Workshop on Hot Topics in Cloud Computing (HotCloud '09)*, 2009.

Peter Bodik, Moises Goldszmidt, Armando Fox, Dawn B. Woodard, and Hans Andersen. Fingerprinting the datacenter: Automated classification of performance crises. In *Proceedings of EuroSys 2010*, 2010. To appear.

Olivier Bousquet and André Elisseeff. Stability and generalization. *Journal of Machine Learning Research*, 2(Mar):499–526, 2002.

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

Daniela Brauckhoff, Kave Salamatian, and Martin May. Applying PCA for traffic anomaly detection: Problems and solutions. In *Proceedings of the 28th IEEE International Conference on Computer Communications (INFOCOM 2009)*, pages 2866–2870, 2009.

Michael P. S. Brown, William Noble Grundy, David Lin, Nello Cristianini, Charles Walsh Sugnet, Terrence S. Furey, Manuel Ares, Jr., and David Haussler. Knowledge-based analysis of microarray gene expression data by using support vector machines. *Proceedings of the National Academy of Sciences*, 97(1):262–267, 2000.

David Brumley, Pongsin Poosankam, Dawn Song, and Jiang Zheng. Automatic patch-based exploit generation is possible: Techniques and implications. In *Proceedings of the 2008 IEEE Symposium on Security and Privacy (SP '08)*, pages 143–157, 2008.

Christopher J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

Huseyin Cavusoglu, Srinivasan Raghunathan, and Wei Yue. Decision-theoretic and game-theoretic approaches to IT security investment. *Journal of Management Information Systems*, 25(2):281–304, 2008.

Nicolò Cesa-Bianchi and Gábor Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, 2006.

Nicolò Cesa-Bianchi, Yoav Freund, David P. Helmbold, David Haussler, Robert E. Schapire, and Manfred K. Warmuth. How to use expert advice. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing*, pages 382–391, 1993.

Nicolò Cesa-Bianchi, Yoav Freund, David Haussler, David P. Helmbold, Robert E. Schapire, and Manfred K. Warmuth. How to use expert advice. *Journal of the Association for Computing Machinery*, 44(3):427–485, May 1997.

Deeparnab Chakrabarty, Aranyak Mehta, and Vijay V. Vazirani. Design is as easy as optimization. In *Proceedings of the 33rd International Colloquium on Automata, Languages and Programming (ICALP)*, volume Part I of *LNCS 4051*, pages 477–488, 2006.

Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm` [Online; accessed 5-May-2010].

Kamalika Chaudhuri and Claire Monteleoni. Privacy-preserving logistic regression. In *Advances in Neural Information Processing Systems 21*, pages 289–296, 2009.

Yu-Chung Cheng, Mikhail Afanasyev, Patrick Verkaik, Péter Benkö, Jennifer Chiang, Alex C. Snoeren, Stefan Savage, and Geoffrey M. Voelker. Automating cross-layer diagnosis of enterprise wireless networks. In *Proceedings of the ACM SIGCOMM 2007 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pages 25–36, 2007.

Simon P. Chung and Aloysius K. Mok. Allergy attack against automatic signature generation. In *Proceedings of the International Symposium on Recent Advances in Intrusion Detection (RAID)*, pages 61–80, September 2006.

Simon P. Chung and Aloysius K. Mok. Advanced allergy attacks: Does a corpus really help? In *Proceedings of the International Symposium on Recent Advances in Intrusion Detection (RAID)*, pages 236–255, September 2007.

Massimiliano Ciaramita, Vanessa Murdock, and Vassilis Plachouras. Online learning from click data for sponsored search. In *Proceeding of the 17th International Conference on World Wide Web (WWW '08)*, pages 227–236, 2008.

Gordon Cormack and Thomas Lynam. Spam corpus creation for TREC. In *Proceedings of the Conference on Email and Anti-Spam (CEAS)*, July 2005.

Marco Cremonini. Evaluating information security investments from attackers perspective: the return-on-attack (ROA). In *Fourth Workshop on the Economics of Information Security*, 2005.

Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, 2000.

Christophe Croux and Anne Ruiz-Gazen. A fast algorithm for robust principal components based on projection pursuit. In *Proccedings in Computational Statistics (Compstat'96)*, pages 211–216, 1996.

Christophe Croux and Anne Ruiz-Gazen. High breakdown estimators for principal components: the projection-pursuit approach revisited. *Journal of Multivariate Analysis*, 95(1), 2005.

Christophe Croux, Peter Filzmoser, and M. Rosario Oliveira. Algorithms for projection-pursuit robust principal component analysis. *Chemometrics and Intelligent Laboratory Systems*, 87(2), 2007.

Hengjian Cui, Xuming He, and Kai W. Ng. Asymptotic distributions of principal components based on robust dispersions. *Biometrika*, 90(4):953–966, 2003.

Nilesh Dalvi, Pedro Domingos, Mausam, Sumit Sanghai, and Deepak Verma. Adversarial classification. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'04)*, pages 99–108, 2004.

Susan J. Devlin, Ramanathan Gnanadesikan, and Jon R. Kettenring. Robust estimation of dispersion matrices and principal components. *Journal of the American Statistical Association*, 76(374):354–362, 1981.

Luc P. Devroye and T. J. Wagner. Distribution-free performance bounds for potential function rules. *IEEE Transactions on Information Theory*, 25(5):601–604, 1979.

Irit Dinur and Kobbi Nissim. Revealing information while preserving privacy. In *Proceedings of the Twenty-Second ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS '03)*, pages 202–210, 2003.

Pat Doyle, Julia I. Lane, Jules J. M. Theeuwes, and Laura V. Zayatz, editors. *Confidentiality, Disclosure and Data Access: Theory and Practical Application for Statistical Agencies*. Elsevier, 2001.

Isabel Drost and Tobias Scheffer. Thwarting the nigritude ultramarine: Learning to identify link spam. In *Proceedings of the European Conference on Machine Learning (ECML '05)*, pages 96–107, 2005.

Thomas Duebendorfer and Stefan Frei. Why silent updates boost security. Tech Report TIK 302, ETH, 2009.

Cynthia Dwork. Differential privacy. In *Proceedings of the 33rd International Colloquium on Automata, Languages and Programming (ICALP)*, pages 1–12, 2006.

Cynthia Dwork. Differential privacy: A survey of results. In *Proceedings of the 5th International Conference on Theory and Applications of Models of Computation (TAMC)*, volume 4978 of *Lecture Notes in Computer Science*, pages 1–19, 2008.

Cynthia Dwork. A firm foundation for private data analysis. *Communications of the ACM*, 2010. to appear.

Cynthia Dwork and Jing Lei. Differential privacy and robust statistics. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing (STOC '09)*, pages 371–380, 2009.

Cynthia Dwork and Sergey Yekhanin. New efficient attacks on statistical disclosure control mechanisms. In *Proceedings of the 28th Annual Conference on Cryptology (CRYPTO 2008)*, pages 469–480, 2008.

Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Proceedings of the 3rd Theory of Cryptography Conference (TCC 2006)*, pages 265–284, 2006.

Cynthia Dwork, Frank McSherry, and Kunal Talwar. The price of privacy and the limits of LP decoding. In *Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing (STOC '07)*, pages 85–94, 2007.

Cynthia Dwork, Moni Naor, Omer Reingold, Guy N. Rothblum, and Salil Vadhan. On the complexity of differentially private data release: efficient algorithms and hardness results. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing (STOC '09)*, pages 381–390, 2009.

Martin Dyer and Alan Frieze. Computing the volume of convex bodies: A case where randomness provably helps. In *Proceedings of the AMS Symposium on Probabilistic Combinatorics and Its Applications*, pages 123–170, 1992.

Usama Mohammad Fayyad. *On the induction of decision trees for multiple concept learning*. PhD thesis, University of Michigan, Ann Arbor, MI, USA, 1992.

Darin Fisher. Multi-process architecture, July 2008. `http://dev.chromium.org/developers/design-documents/multi-process-architecture` [Online; accessed 6-May-2010].

Ronald A. Fisher. Question 14: Combining independent tests of significance. *American Statistician*, 2(5):30–30J, 1948.

Prahlad Fogla and Wenke Lee. Evading network anomaly detection systems: Formal reasoning and practical techniques. In *Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS'06)*, pages 59–68, 2006.

Jason Franklin, Vern Paxson, Adrian Perrig, and Stefan Savage. An inquiry into the nature and causes of the wealth of internet miscreants. In *Proceedings of the 2007 ACM Conference on Computer and Communications Security*, pages 375–388, 2007.

Stefan Frei, Thomas Duebendorfer, and Bernhard Plattner. Firefox (in) security update dynamics exposed. *SIGCOMM Computer Communication Review*, 39(1):16–22, 2009.

Yoav Freund and Robert Schapire. A short introduction to boosting. *Journal of the Japanese Society for Artificial Intelligence*, 14(5):771–780, 1999a.

Yoav Freund and Robert E. Schapire. Adaptive game playing using multiplicative weights. *Games and Economic Behavior*, 29:79–103, 1999b.

Jeffrey Friedberg. Internet fraud battlefield, April 2007. `http://www.ftc.gov/bcp/workshops/proofpositive/Battlefield_Overview.pdf` [Online; accessed 6-May-2010].

Neal Fultz and Jens Grossklags. Blue versus red: Towards a model of distributed security attacks. In *Proceedings of the Thirteenth International Conference Financial Cryptography and Data Security*, pages 167–183, 2009.

Arpita Ghosh, Benjamin I. P. Rubinstein, Sergei Vassilvitskii, and Martin Zinkevich. Adaptive bidding for display advertising. In *Proceedings of the 18th International World Wide Web Conference (WWW 2009)*, pages 251–260, 2009.

Lawrence A. Gordon and Martin P. Loeb. The economics of information security investment. *ACM Transactions on Information and System Security*, 5(4):438–457, 2002.

Paul Graham. A plan for spam. `http://www.paulgraham.com/spam.html`, August 2002.

Jens Grossklags, Nicolas Christin, and John Chuang. Secure or insure?: A game-theoretic analysis of information security games. In *Proceeding of the 17th International Conference on World Wide Web*, pages 209–218, 2008.

Guavus, 2010. `http://www.guavus.com` [Online; accessed 22-April-2010].

Zoltán Gyöngyi and Hector Garcia-Molina. Link spam alliances. In *Proceedings of the 31st International Conference on Very Large Data Bases (VLDB '05)*, pages 517–528, 2005.

Frank R Hampel, Elvezio M Ronchetti, Peter J Rousseeuw, and Werner A Stahel. *Robust Statistics: The Approach Based on Influence Functions*. Wiley Series in Probability and Mathematical Statistics. Wiley, 1980.

Kjell Hausken. Returns to information security investment: The effect of alternative information security breach functions on optimal investment and sensitivity to vulnerability. *Information Systems Frontiers*, 8(5):338–349, 2006.

Michael Hay, Chao Li, Gerome Miklau, and David Jensen. Accurate estimation of the degree distribution of private networks. In *Proceedings of the 2009 Ninth IEEE International Conference on Data Mining (ICDM'09)*, pages 169–178, 2009.

Elad Hazan and Satyen Kale. On stochastic and worst-case models for investing. In *Advances in Neural Information Processing Systems (NIPS) 22*, pages 709–717, 2010.

Mark Herbster and Manfred K. Warmuth. Tracking the best expert. *Machine Learning*, 32 (2):151–178, 1998.

Ola Hössjer and Christophe Croux. Generalizing univariate signed rank statistics for testing and estimating a multivariate location parameter. *Journal of Nonparametric Statistics*, 4 (3):293–308, 1995.

Michael Howard. Attack surface: Mitigate security risks by minimizing the code you expose to untrusted users. *MSDN Magazine*, November 2004.

Ling Huang, Michael I. Jordan, Anthony Joseph, Minos Garofalakis, and Nina Taft. In-network PCA and anomaly detection. In *Advances in Neural Information Processing Systems 19 (NIPS 2006)*, pages 617–624, 2007.

Peter. J. Huber. *Robust Statistics*. Wiley Series in Probability and Mathematical Statistics. Wiley, 1981.

Nicole Immorlica, Kamal Jain, Mohammad Mahdian, and Kunal Talwar. Click fraud resistant methods for learning click-through rates. In *Proceedings of the First International Workshop on Internet and Network Economics (WINE 2005)*, volume 3828 of *Lecture Notes in Computer Science*, pages 34–45, 2005.

International Organization for Standardization. Information technology – security techniques – code of practice for information security management. ISO/IEC 17799:2005, ISO, 2005.

J. Edward Jackson and Govind S. Mudholkar. Control procedures for residuals associated with principal component analysis. *Technometrics*, 21(3):341–349, 1979.

Thorsten Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '02)*, pages 133–142, 2002.

Srikanth Kandula, Ranveer Chandra, and Dina Katabi. What's going on? Learning communication rules in edge networks. In *Proceedings of the ACM SIGCOMM 2008 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pages 87–98, 2008.

Chris Kanich, Christian Kreibich, Kirill Levchenko, Brandon Enright, Geoffrey M. Voelker, Vern Paxson, and Stefan Savage. Spamalytics: An empirical analysis of spam marketing conversion. In *Proceedings of the 2008 ACM Conference on Computer and Communications Security*, pages 3–14, 2008.

Khalid Kark, Jonathan Penn, and Alissa Dill. 2008 CISO priorities: The right objectives but the wrong focus. *Le Magazine de la Sécurité Informatique*, April 2009.

Christoph Karlberger, Günther Bayler, Christopher Kruegel, and Engin Kirda. Exploiting redundancy in natural language to penetrate Bayesian spam filters. In *Proceedings of the USENIX Workshop on Offensive Technologies (WOOT)*, pages 1–7, August 2007.

Shiva Prasad Kasiviswanathan, Homin K. Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. What can we learn privately? In *Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS '08)*, pages 531–540, 2008.

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

Michael Kearns and Dana Ron. Algorithmic stability and sanity-check bounds for leave-one-out cross-validation. *Neural Computation*, 11:1427–1453, 1999.

Hyang-Ah Kim and Brad Karp. Autograph: Toward automated, distributed worm signature detection. In *Proceedings of the USENIX Security Symposium*, pages 271–286, 2004.

George Kimeldorf and Grace Wahba. Some results on Tchebycheffian spline functions. *Journal of Mathematical Analysis and Applications*, 33(1):82–95, 1971.

Bryan Klimt and Yiming Yang. Introducing the Enron corpus. In *Proceedings of the Conference on Email and Anti-Spam (CEAS)*, July 2004.

Aleksandra Korolova, Krishnaram Kenthapadi, Nina Mishra, and Alex Ntoulas. Releasing search queries and clicks privately. In *Proceedings of 18th International World Wide Web Conference (WWW'09)*, pages 171–180, 2009.

Vineet Kumar, Rahul Telang, and Tridas Mukhopadhyay. Optimal information security architecture for the enterprise, 2008. `http://ssrn.com/abstract=1086690` [Online; accessed 6-May-2010].

Samuel Kutin and Partha Niyogi. Almost-everywhere algorithmic stability and generalization error. Technical report TR-2002-03, Computer Science Department, University of Chicago, 2002.

Anukool Lakhina, Mark Crovella, and Christophe Diot. Diagnosing network-wide traffic anomalies. In *Proceedings of the ACM SIGCOMM 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pages 219–230, 2004a.

Anukool Lakhina, Mark Crovella, and Christophe Diot. Characterization of network-wide anomalies in traffic flows. In *Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement (IMC '04)*, pages 201–206, 2004b.

Anukool Lakhina, Konstantina Papagiannaki, Mark Crovella, Christophe Diot, Eric D. Kolaczyk, and Nina Taft. Structural analysis of network traffic flows. *SIGMETRICS Performance Evaluation Review*, 32(1):61–72, 2004c.

Anukool Lakhina, Mark Crovella, and Christophe Diot. Mining anomalies using traffic feature distributions. In *Proceedings of the 2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM '05)*, pages 217–228, 2005a.

Anukool Lakhina, Mark Crovella, and Christophe Diot. Detecting distributed attacks using network-wide flow traffic. In *Proceedings of the FloCon 2005 Analysis Workshop*, 2005b.

Aleksandar Lazarevic, Levent Ertöz, Vipin Kumar, Aysel Ozgur, and Jaideep Srivastava. A comparative study of anomaly detection schemes in network intrusion detection. In *Proceedings of the SIAM International Conference on Data Mining*, pages 25–36, 2003.

Guoying Li and Zhonglian Chen. Projection-pursuit approach to robust dispersion matrices and principal components: Primary theory and Monte Carlo. *Journal of the American Statistical Association*, 80(391):759–766, 1985.

Xin Li, Fang Bian, Mark Crovella, Christophe Diot, Ramesh Govindan, Gianluca Iannaccone, and Anukool Lakhina. Detection and identification of network anomalies using sketch subspaces. In *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement (IMC '06)*, pages 147–152, 2006a.

Xin Li, Fang Bian, Hui Zhang, Christophe Diot, Ramesh Govindan, Wei Hong, , and Gianluca Iannaccone. MIND: A distributed multidimensional indexing for network diagnosis. In *Proceedings of the 25th IEEE International Conference on Computer Communications (INFOCOM 2006)*, pages 1422–1433, 2006b.

Yihua Liao and V. Rao Vemuri. Using text categorization techniques for intrusion detection. In *Proceedings of the USENIX Security Symposium*, pages 51–59, 2002.

Hsuan-Tien Lin, Chih-Jen Lin, and Ruby C. Weng. A note on Platt's probabilistic outputs for support vector machines. *Machine Learning*, 68:267–276, 2007.

László Lovász and Santosh Vempala. Simulated annealing in convex bodies and an $O^*(n^4)$ volume algorithm. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science (FOCS '03)*, pages 650–659, 2003.

László Lovász and Santosh Vempala. Hit-and-run from a corner. In *Proceedings of the Thirty-Sixth Annual ACM Symposium on Theory of Computing (STOC '04)*, pages 310–314, 2004.

Daniel Lowd and Christopher Meek. Good word attacks on statistical spam filters. In *Proceedings of the Conference on Email and Anti-Spam (CEAS)*, July 2005a.

Daniel Lowd and Christopher Meek. Adversarial learning. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining (KDD '05)*, pages 641–647, 2005b.

Kong-wei Lye and Jeannette M. Wing. Game strategies in network security. In *Proceedings of the Foundations of Computer Security Workshop*, pages 13–22, 2002.

Ricardo Maronna. Principal components and orthogonal regression based on robust scales. *Technometrics*, 47(3):264–273, 2005.

Frank McSherry and Ilya Mironov. Differentially private recommender systems: building privacy into the net. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '09)*, pages 627–636, 2009.

Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS '07)*, pages 94–103, 2007.

Tony Meyer and Brendon Whateley. SpamBayes: Effective open-source, Bayesian based, email classification system. In *Proceedings of the Conference on Email and Anti-Spam (CEAS)*, July 2004.

Tom M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

R. Ann Miura-Ko and Nicholas Bambos. SecureRank: A risk-based vulnerability management scheme for computing infrastructures. In *Proceedings of IEEE International Conference on Communications*, pages 1455–1460, 2007.

R. Ann Miura-Ko, Benjamin Yolken, John Mitchell, and Nicholas Bambos. Security decision-making among interdependent organizations. In *Proceedings of the 21st IEEE Computer Security Foundations Symposium*, pages 66–80, 2008.

Mozilla Foundation. Known vulnerabilities in Mozilla products, 2010. `http://www.mozilla.org/security/known-vulnerabilities/` [Online; accessed 14-January-2010].

Sayan Mukherjee, Partha Niyogi, Tomaso Poggio, and Ryan Rifkin. Learning theory: Stability is sufficient for generalization and necessary and sufficient for consistency of empirical risk minimization. *Advances in Computational Mathematics*, 25:161–193, 2006.

Srinivas Mukkamala, Guadalupe Janoski, and Andrew Sung. Intrusion detection using neural networks and support vector machines. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, pages 1702–1707, 2002.

Arvind Narayanan and Vitaly Shmatikov. Robust de-anonymization of large sparse datasets. In *Proceedings of the 2008 IEEE Symposium on Security and Privacy*, pages 111–125, 2008.

Narus, 2010. `http://www.narus.com` [Online; accessed 22-April-2010].

Nature. Security ethics. *Nature*, 463(7278):136, 14 Jan 2010. Editorial.

James Newsome, Brad Karp, and Dawn Song. Polygraph: Automatically generating signatures for polymorphic worms. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 226–241, 2005.

James Newsome, Brad Karp, and Dawn Song. Paragraph: Thwarting signature learning by training maliciously. In *Proceedings of the 9th International Symposium on Recent Advances in Intrusion Detection (RAID 2006)*, pages 81–105, 2006.

Erik Ordentlich and Thomas M. Cover. The cost of achieving the best portfolio in hindsight. *Mathematics of Operations Research*, 23(4):960–982, 1998.

Xinming Ou, Wayne F. Boyer, and Miles A. McQueen. A scalable approach to attack graph generation. In *Proceedings of the 13th ACM Conference on Computer and Communications Security*, pages 336–345, 2006.

PhishTank. `http://www.phishtank.com`, 2010. [Online; accessed 13-April-2010].

John P. Pironti. Key elements of an information security program. *Information Systems Control Journal*, 1, 2005.

John Ross Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

Luis Rademacher and Navin Goyal. Learning convex bodies is hard. In *Proceedings of the 22nd Annual Conference on Learning Theory (COLT 2009)*, pages 303–308, 2009.

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems 20*, pages 1177–1184, 2008.

Anirudh Ramachandran, Nick Feamster, and Santosh Vempala. Filtering spam with behavioral blacklisting. In *Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS '07)*, pages 342–351, 2007.

Eric Rescorla. Is finding security holes a good idea? *IEEE Security and Privacy*, 3(1): 14–19, 2005.

Thomas C. Rindfleisch. Privacy, information technology, and health care. *Communications of the ACM*, 40(8):92–100, 1997.

Haakon Ringberg, Augustin Soule, Jennifer Rexford, and Christophe Diot. Sensitivity of PCA for traffic anomaly detection. In *Proceedings of the 2007 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS'07)*, pages 109–120, 2007.

Gary Robinson. A statistical approach to the spam problem. *Linux Journal*, March 2003.

Benjamin I. P. Rubinstein, Peter L. Bartlett, Ling Huang, and Nina Taft. Learning in a large function space: Privacy-preserving mechanisms for SVM learning. *CoRR*, abs/0911.5708, 2009. Submitted 30 Nov 2009.

Walter Rudin. *Fourier Analysis on Groups*. Wiley Classics Library. Wiley-Interscience, reprint edition, 1994.

Udam Saini. Machine learning in the presence of an adversary: Attacking and defending the spambayes spam filter. Dissertation UCB/EECS-2008-62, Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, 2008.

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

Anand D. Sarwate, Kamalika Chaudhuri, and Claire Monteleoni. Differentially private support vector machines. *CoRR*, abs/0912.0071, 2009. Submitted 1 Dec 2009.

Greg Schohn and David Cohn. Less is more: Active learning with support vector machines. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000)*, pages 839–846, 2000.

Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Adaptive Computation and Machine Learning. MIT Press, 2001.

Cyrus Shaoul and Chris Westbury. A USENET corpus (2005-2007), October 2007.

Robert L. Smith. The hit-and-run sampler: A globally reaching Markov chain sampler for generating arbitrary multivariate distributions. In *Proceedings of the 28th Conference on Winter Simulation (WSC '96)*, pages 260–264, 1996.

Augustin Soule, Kavé Salamatian, and Nina Taft. Combining filtering and statistical methods for anomaly detection. In *Proceedings of the 5th ACM SIGCOMM Conference on Internet Measurement (IMC '05)*, pages 31–31, 2005.

Salvatore J. Stolfo, Shlomo Hershkop, Chia-Wei Hu, Wei-Jen Li, Olivier Nimeskern, and Ke Wang. Behavior-based modeling and its application to Email analysis. *ACM Transactions on Internet Technology*, pages 187–221, 2006.

Gilles Stoltz and Gábor Lugosi. Internal regret in on-line portfolio selection. *Machine Learning*, 59(1-2):125–159, 2005.

Latanya Sweeney. k-anonymity: a model for protecting privacy. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 10(5):557–570, 2002.

Kymie M. C. Tan, Kevin S. Killourhy, and Roy A. Maxion. Undermining an anomaly-based intrusion detection system using common exploits. In *Proceedings of the 5th International Conference on Recent Advances in Intrusion Detection (RAID'02)*, pages 54–73, 2002.

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

Hal R. Varian. Managing online security risks. New York Times. Jun 1, 2000.

Hal R. Varian. System reliability and free riding, 2001. Note available at `http://www.sims.berkeley.edu/~hal/Papers/2004/reliability`.

Daniel Veditz. Personal communication, 2009.

Daniel Veditz. Mozilla security group, 2010. `http://www.mozilla.org/projects/security/secgrouplist.html`.

Shobha Venkataraman, Avrim Blum, and Dawn Song. Limits of learning-based signature generation with adversaries. In *Proceedings of the Network and Distributed System Security Symposium (NDSS'2008)*, 2008.

David Wagner and Paolo Soto. Mimicry attacks on host-based intrusion detection systems. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pages 255–264, 2002.

Bernhard Warner. Home PCs rented out in sabotage-for-hire racket. *Reuters*, July 2004.

Wikipedia. Information security — Wikipedia, The Free Encyclopedia, 2010. URL http://en.wikipedia.org/w/index.php?title=Information_security&oldid=355701059. [Online; accessed 13-April-2010].

Leon Willenborg and Ton de Waal. *Elements of Statistical Disclosure Control*. Springer-Verlag, 2001.

Gregory L. Wittel and S. Felix Wu. On attacking statistical spam filters. In *Proceedings of the Conference on Email and Anti-Spam (CEAS'04)*, 2004.

Aaron D. Wyner. Capabilities of bounded discrepancy decoding. *The Bell System Technical Journal*, 44:1061–1122, Jul/Aug 1965.

Yin Zhang, Zihui Ge, Albert Greenberg, and Matthew Roughan. Network anomography. In *Proceedings of the 5th ACM SIGCOMM Conference on Internet Measurement (IMC '05)*, pages 30–30, 2005.