Invited Talk

Can Machine Learning Be Secure?

Marco Barreno Blaine Nelson Russell Sears Anthony D. Joseph J. D. Tygar

Computer Science Division
University of California, Berkeley

{barreno,nelsonb,sears,adj,tygar}@cs.berkeley.edu

ABSTRACT

Machine learning systems offer unparalled flexibility in dealing with evolving input in a variety of applications, such as intrusion detection systems and spam e-mail filtering. However, machine learning algorithms themselves can be a target of attack by a malicious adversary. This paper provides a framework for answering the question, "Can machine learning be secure?" Novel contributions of this paper include a taxonomy of different types of attacks on machine learning techniques and systems, a variety of defenses against those attacks, a discussion of ideas that are important to security for machine learning, an analytical model giving a lower bound on attacker's work function, and a list of open problems.

Categories and Subject Descriptors

D.4.6 [**Security and Protection**]: Invasive software (e.g., viruses, worms, Trojan horses); I.5.1 [**Models**]: Statistical; I.5.2 [**Design Methodology**]

General Terms

Security

Keywords

Adversarial Learning, Computer Networks, Computer Security, Game Theory, Intrusion Detection, Machine Learning, Security Metrics, Spam Filters, Statistical Learning

1. INTRODUCTION

Machine learning techniques are being applied to a growing number of systems and networking problems, particularly those problems where the intention is to detect anomalous system behavior. For instance, network Intrusion Detection Systems (IDS) monitor network traffic to detect abnormal activities, such as attacks against hosts or servers. Machine learning techniques offer the benefit that they can detect novel differences in traffic (which presumably represent attack traffic) by being trained on normal (known good) and attack (known bad) traffic. The traditional approach to designing an IDS relied on an expert codifying rules defining normal behavior and intrusions [26]. Because this approach often fails to detect novel intrusions, a variety of researchers have proposed incorporating machine learning techniques into intrusion detection systems [1, 16, 18, 24, 38, 41]. On the other hand, use of machine learning opens the possibility of an adversary who maliciously "mis-trains" a learning system in an IDS. A natural question arises: what techniques (in their attacks) can an adversary use to confuse a learning system?

This paper explores the larger question, as posed in the title of this paper, can machine learning be secure? Specific questions that we examine include:

- Can the adversary manipulate a learning system to permit a specific attack? For example, can an attacker leverage knowledge about the machine learning system used by a spam e-mail filtering system to bypass the filtering?
- Can an adversary degrade the performance of a learning system to the extent that system administrators are forced to disable the IDS? For example, could the attacker confuse the system and cause valid e-mail to be rejected?
- What defenses exist against adversaries manipulating (attacking) learning systems?
- More generally, what is the potential impact from a security standpoint of using machine learning on a system? Can an attacker exploit properties of the machine learning technique to disrupt the system?

The issue of machine learning security goes beyond intrusion detection systems and spam e-mail filters. Machine learning is a powerful technique and has been used in a variety of applications, including web services, online agent systems, virus detection, cluster monitoring, and a variety of applications that must deal with dynamically changing data patterns.

Novel contributions of this paper include a taxonomy of different types of attacks on machine learning techniques and systems, a variety of defenses against those attacks, a discussion of ideas that are important to security for machine learning, an analytical model giving a lower bound on attacker's work function, and a list of open problems.

The rest of this paper is organized as follows: Section 2 discusses machine learning and how it is typically used in a system, Section 3 develops a taxonomy of attacks, Section 4 introduces potential defenses against attacks and explores their potential costs, Section 5 identifies several of the ideas that are important to security for machine learning, Section 6 presents an analytical model that examines an attack to manipulate a naive learning algorithm, Section 7 discusses related work, potential research directions, and our conclusions.

2. REVIEW

2.1 The Learning Problem

A machine learning system attempts to find a hypothesis function f that maps events (which we call points below) into different classes. For example, an intrusion detection system would find a hypothesis function f that maps an event point (an instance of network behavior) into one of two results: *normal* or *intrusion*.

One kind of learning system called *supervised learning* works by taking a *training data set* together with labels identifying the class for every point in the training data set.

For example, a supervised learning algorithm for an IDS would have a training set consisting of points corresponding to normal behavior and points corresponding to intrusion behavior. The learning algorithm selects the hypothesis function f that best predicts the classification of a point. More complicated learning algorithms can deal with event points that are both labeled and unlabeled and furthermore can deal with continuous streams of unlabeled points so that training is an ongoing process. In this paper, we call these algorithms *online learning systems.*

This remainder of this subsection presents a concise overview of concepts in statistical learning theory. The presentation below is formal and can be skipped on a first reading. For a fuller discussion with motivation, refer to [11, 31].

A *predictive learning problem* is defined over an input space $\mathcal{X}$, an output space $\mathcal{Y}$, and a loss function $\ell : Y \times Y \to \mathbb{R}$. The input to the problem is a training set $\mathcal{S}$, specified as $\{(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}\}$, and the output is a hypothesis function $f : \mathcal{X} \to \mathcal{Y}$. We choose f from a hypothesis space (or function class) $\mathcal{F}$ to minimize the prediction error given by the loss function. In many cases, researchers assume *stationarity*, that the distribution of data points encountered in the future will be the same as the distribution of the training set. Stationarity allows us to reduce the predictive learning problem to a minimization of the sum of the loss over the training set:

$$f^* = \operatorname*{argmin}_{f \in \mathcal{F}} \sum_{(x_i, y_i) \in \mathcal{S}} \ell(f(x_i), y_i) \qquad (1)$$

Loss functions are typically defined to be non-negative over all inputs and zero when $f(x_i) = y_i$. A commonly used loss function is the squared-error loss $\ell_{sq}(f(x_i), y) = (f(x_i) - y)^2$. The hypothesis space (or function class) $\mathcal{F}$ can be any representation of functions from $\mathcal{X}$ to $\mathcal{Y}$, such as linear functions, polynomials, boolean functions, or neural networks. The choice of $\mathcal{F}$ involves a tradeoff between expressiveness and ability to generalize. If $\mathcal{F}$ is too expressive, it can *overfit* the training data. The extreme case is a lookup table that maps x_i to y_i for each instance of the training set but will not generalize to new data. A linear function, on the other hand, will generalize what it learns on the training set to new points, though it may not be sufficiently expressive to describe intricate data sets. We typically use simple function classes, such as linear functions, to avoid overfitting.

We can describe a more general learning problem by dropping the requirement that the training examples include all the labels y_i. The case where all labels are present is referred to as *supervised* learning, when no labels are present the problem is *unsupervised*, and when some labels are present the problem is *semi-supervised*. In all these cases we can pose the learning problem as the minimization of some measure over the training set:

$$f^* = \operatorname*{argmin}_{f \in \mathcal{F}} \sum_{(x_i) \in \mathcal{S}} L(x_i, f) \qquad (2)$$

2.2 Terminology and Running Example

To illustrate some of our contributions, we use a running example throughout this paper: a network Intrusion Detection System (IDS). This IDS receives network events $x \in \mathcal{X}$ and classifies each event x as either $f(x) =$ *normal* or $f(x) =$ *intrusion*. The literature describes a number of algorithms for learning f over time, but we wish to consider the impact of malicious input on the learning algorithm. This paper poses the question: can a malicious party send events to the IDS that will cause it to malfunction? Possible types of attacks on the IDS include attacks on the learning algorithm, causing the IDS to create an f that misclassifies events. As we discuss in the next section, this is only one of a number of types of attacks that an adversary can make on an IDS.

It is important to be careful about notation here. When we speak of attacks, we mean an attack on the learning system (e.g., the learner in an IDS). Attacks may try to make the learner mis-learn, fail because of denial of service, report information about its internal state, etc. "Attack" should be distinguished from "intrusion." An attack targets a learning system; an intrusion targets a computer system (such as a system protected by an IDS). While many researchers use the word "attack" to include intrusions, in this paper we are careful to use the word "attack" only to mean an attack on a learner.

We do not want to restrict ourselves to particular learning algorithms used by intrusion detection systems to choose hypotheses. However, we allow adversaries that have deep understanding of the learning algorithms.

Similarly, we do not discuss mechanisms for translating network level events into a form relevant to the learner. We call each unit of data seen by the learner a *data point*, or simply a *point*. In the context of the IDS, our discussion encompasses continuous, discrete, or mixed data. We assume that $\mathcal{X}$ is a metric space, allowing us to freely discuss distances

		Integrity	*Availability*
Causative:	*Targeted*	Permit a specific intrusion	Create sufficient errors to make system unusable for one person or service
	Indiscriminate	Permit at least one intrusion	Create sufficient errors to make learner unusable
Exploratory:	*Targeted*	Find a permitted intrusion from a small set of possibilities	Find a set of points misclassified by the learner
	Indiscriminate	Find a permitted intrusion	

Table 1: The attack model.

between points. Furthermore, we assume the set of points classified as normal by the IDS forms multiple contiguous subsets in $\mathcal{X}$. The border of this set is called the *decision boundary.*

Below, we consider a variety of scenarios and assumptions.

3. ATTACKS

3.1 Attack Model

We give relevant properties for analyzing attacks on machine learning systems.

Influence

- **Causative -** *Causative* attacks alter the training process through influence over the training data.
- **Exploratory -** *Exploratory* attacks do not alter the training process but use other techniques, such as probing the learner or offline analysis, to discover information.

Specificity

- **Targeted -** The specificity of an attack is a continuous spectrum. At the *targeted* end, the focus of the attack is on a particular point or a small set of points.
- **Indiscriminate -** At the *indiscriminate* end, the adversary has a more flexible goal that involves a very general class of points, such as "any false negative."

Security violation

- **Integrity -** An *integrity* attack results in intrusion points being classified as normal (false negatives).
- **Availability -** An *availability* attack is a broader class of attack than an integrity attack. An *availability* attack results in so many classification errors, both false negatives and false positives, that the system becomes effectively unusable.

These three axes define a space of attacks; Table 1 provides a concise summary.

In *causative* attacks, the adversary has some measure of control over the training of the learner. An attack that causes the learner to misclassify intrusion points, for example an attack that fools an IDS into not flagging a known exploit as an intrusion, is a *causative integrity* attack. The distinction between *targeted* and *indiscriminate causative integrity* attacks is the difference between choosing one particular exploit or just finding any exploit. A *causative availability* attack causes the learner's performance to degrade. For example, an adversary might cause an IDS to reject many legitimate HTTP connections. A *causative availability attack* may be used to force the system administrator to disable the IDS. A *targeted* attack focuses on a particular service, while an *indiscriminate* attack has a wider scope.

Exploratory attacks do not attempt to influence learning; they instead attempt to discover information about the state of the learner. *Exploratory integrity* attacks seek to find intrusions that are not recognized by the learner.

3.2 Online Learning

A learner can have an explicit training phase or can be continuously trained (online learner). Online learning allows the learner to adapt to changing conditions; the assumption of stationarity is weakened to accommodate long-term changes in the distribution of data seen by the learner. Online learning is more flexible, but potentially simplifies *causative* attacks. By definition, an online learner changes its prediction function over time, so an adversary has the opportunity to shape this change. Gradual *causative* attacks may be difficult to detect.

4. DEFENSES

In this section we discuss potential defenses against attacks. This section describes speculative work, and the efficacy of these techniques in practice is a topic for future research.

4.1 Robustness

To increase robustness against *causative* attacks we constrain the class of functions (hypotheses) that the learner considers. The constraint we consider is the statistical technique of *regularization.* Regularization extends the basic learning optimization in Equation (1) by adding a term $J(f)$ that penalizes complex hypotheses:

$$f^* = \operatorname*{argmin}_{f \in \mathcal{F}} \left\{ \sum_{(x_i, y_i) \in \mathcal{S}} \ell(f(x_i), y_i) + \lambda J(f) \right\} \tag{3}$$

Here λ adjusts the trade-off. The penalty term $J(f)$ can be as simple as the sum of squares of the parameters of f. Regularization is used in statistics to restrict or bias the choice of hypothesis when the problem suffers from lack of

		Integrity	*Availability*
Causative:	*Targeted*	• Regularization • Randomization	• Regularization • Randomization
	Indiscriminate	• Regularization	• Regularization
Exploratory:	*Targeted*	• Information hiding • Randomization	• Information hiding
	Indiscriminate	• Information hiding	

Table 2: Defenses against the attacks in Table 1.

data or noisy data. It can also be interpreted as encoding a prior distribution on the parameters, penalizing parameter choices that are less likely *a priori*. Regularization and prior distributions can both be viewed as penalty functions in Equation (3) [42].

The constraint added to the learning problem by the penalty term may help our defenses in two ways. First, it has the effect of smoothing the solution, removing complexity that an adversary might exploit in attacks. Second, prior distributions can be a useful way to encode expert knowledge about a domain or use domain structure learned from a preprocessing step. In the simplest case, we might have a reasonable guess for the parameters (such as the mean) that we wish to refine; in a more complex situation, we could perform an analysis of a related dataset giving correlation information which informs a multivariate Gaussian prior on the parameters [28]. When the learner has more prior information (or constraints) on which to base the learning, there is less dependence on exact data fitting, so there is less opportunity for the adversary to exert influence over the learning process.

4.2 Detecting Attacks

The learner can benefit from the ability to detect attacks even if they are not prevented. Detecting attacks can be difficult even when the adversary is not attempting to conceal them. However, we may be able to detect *causative* attacks by using a special test set. This test set could include several known intrusions and intrusion variants, as well as some random points that are similar to the intrusions. After the learner has been trained, misclassifying a disproportionately high number of intrusions could indicate compromises.

To detect naive *exploratory* attacks, a separate clustering algorithm could be run against data classified by the learner. The sudden appearance of a large cluster near the decision boundary could indicate systematic probing. This type of defense is akin to port scan detection, which has become an arms race between port scanners and IDS [26].

Detecting an attack gives the learner information about the adversary's capabilities. This information may be used to reformulate defense strategies.

As the adversary's control over the data increases, the best strategy for the learner is to ignore potentially tainted data. Otherwise, the adversary can exploit misplaced trust. These ideas have been formalized within the context of deception games [14, 32], which typically assume all players know the extent to which other players may manipulate data. However, if the parties *estimate* each other's abilities, more sophisticated strategies emerge.

4.3 Disinformation

In some circumstances, the learner may be able to alter the data seen by the adversary. This strategy of *disinformation* has the goal of confusing the adversary's estimate of the learner's state. In the simplest case, the adversary would then be faced with a situation not unlike a learner under an *indiscriminate causative availability* attack. The goal of the learner is to prevent the adversary from learning the decision boundary. Please note how the roles of adversary and learner have been reversed.

A more sophisticated learner could trick the adversary into believing that a particular intrusion was not included in the training set. This apparently permitted "intrusion" would act as a honeypot [27], causing the adversary to reveal itself. An increase in the incidence of that particular attack would be detected, revealing the existence of an adversary. In this case again, roles would reverse, and the adversary would face a situation analogous to a learner subjected to a *targeted causative integrity* attack.

4.4 Randomization for Targeted Attacks

Targeted attacks hinge on the classification of one point or a small set of points. They are more sensitive to variations in the decision boundary than *indiscriminate* attacks because boundary movement is more likely to change the classification of the relevant points.

This suggests randomization as a potential tool against *targeted causative* attacks. In such an attack, the adversary has to do a particular amount of work to move the decision boundary past the targeted point. If there is some randomization in the placement of the boundary and the adversary has imperfect feedback from the learner, more work is required.

4.5 Cost of Countermeasures

The more we know about the distribution of training data, the less room there is for an adversary to manipulate the learner. The disadvantage, however, is that the legitimate data has less influence in the learning process. A tension exists between expressivity and constraint: as the learner includes more prior information, it loses flexibility to adapt to the data, but as it incorporates more information from the data, it becomes more vulnerable to attack.

Equation (3) makes this tradeoff explicit with λ. In the adversarial scenario, this tradeoff becomes more relevant because the adversary may have influence over the data.

Randomization increases the adversary's work, but it also will increase the learner's base error rate. Determining the right amount of randomization is an open problem.

4.6 Summary of Defenses

Table 2 shows how our defenses discussed here relate to attack classes presented in Table 1. (*Information hiding* is an additional technique discussed in Section 5 below.)

5. DISCUSSION

5.1 Secrets and Computational Complexity

A number of defenses and attacks upon machine learning algorithms hinge upon the types of information available to the adversary. Some of these involve information about the decision boundary. Below we consider factors that influence the security and secrecy of the decision boundary.

5.2 Scale of Training

Some machine learning systems are trained by the end user, while others are trained using data from many users or organizations. The choice between these two models is sometimes cast as a tradeoff between the amount of training data and the secrecy of the resulting classifier [3]. This issue also applies to an IDS; if an IDS is trained each time it is deployed then it will have comparatively little data regarding normal network traffic. It will also have no chance to learn about novel intrusions before seeing them in the wild.

Conversely, an IDS that uses a global set of rules would be able to adapt to novel intrusion attempts more quickly. Unfortunately, any adversary with access to a public IDS classification function can test to ensure that its intrusion points will be accepted by deployments of the same classification function.

These issues are instances of a more general problem. In some cases, it seems reasonable to assume the adversary has little access to information available to the learner. However, unless the adversary has no prior knowledge about the learning problem at hand, we cannot assume all of the information provided in the training set is secret. Therefore, it is unclear how much is gained by attempting to keep the training set, and therefore the state of the classifier, secret.

Many systems already attempt to achieve a balance between global and local retraining [3]. Systems that take this approach have the potential to outperform systems that perform training at a single level. However, the relationships between multilevel training, the adversary's domain knowledge, and secrecy are not yet well understood.

5.2.1 *Adversary Observations*

Even without prior knowledge regarding a particular system, an adversary still may deduce the state of the learning algorithm. For example, if the learning system provides feedback to the adversary (e.g., "Request denied"), then a probing attack could be used to map the space of acceptable inputs.

If the adversary has no information regarding the type of decision boundary used by the learner, this process could require a number of probes proportional to the size of the space. On the other hand, if the adversary knows which learning algorithm is being used, a few well-chosen probes could give the adversary sufficient knowledge of the learner's state. As a standard security practice, we assume the learning algorithm itself to be common knowledge.

Instead of expecting the learning algorithm to be a secret, some systems attempt to prevent the adversary from discovering the set of features the learning algorithm uses. This may be realistic in systems with a small number of deployments.

Ideally, we could produce an information theoretic bound on the amount of information an adversary could gain by observing the behavior of a particular algorithm on a particular point. Using these bounds, we could reason about the algorithm's robustness against probing attacks. In this setting, it may also be interesting to distinguish between information gained from normal points drawn from the data's underlying distribution, intrusion points from a third party, and (normal or intrusion) attack points of the adversary's choosing.

An adversary with sufficient information regarding training data, classifications of data points, or the internal state of a learner would be able to deduce the learner's decision boundary. This knowledge could simplify other types of attacks.

For instance, the adversary could avoid detection by choosing intrusion points that will be misclassified by the learner, or launch an *availability* attack by manipulating normal points in a way that leads to misclassification. In either case, by increasing the number of points that are in the region that the defender incorrectly classifies, the adversary could increase the error rate.

Some algorithms classify points by translating them into an abstract space and performing the actual classification in that space. The mapping between raw data and the abstract space is often difficult to reason about. Therefore, it may be computationally difficult for an adversary to use knowledge of a classifier's decision boundary to generate "interesting" attack points that will be misclassified.

One can imagine classes of decision boundaries that are meaningful, yet provably provide an adversary with no information regarding unclassified points. Even with complete knowledge of the state of a learner that uses such a decision boundary, it would be computationally intractable to find one of a few "interesting" points in a sufficiently large search space.

In some cases, the decision boundary itself may contain sensitive information. For example, knowledge of the boundary may allow an adversary to infer confidential information about the training set. Alternatively, the way the decision boundary was constructed might be a secret.

5.2.2 *Security Properties*

The performance of different algorithms will likely degrade differently as the adversary controls larger fractions of the training set. A measurement of an algorithm's ability to deal with malicious training errors could help system designers reason about and decide between different learners. A simple approach would be to characterize an algorithm's performance when subjected to a particular type of attack, but this would lead to an arms race as adversaries devise classes of attacks not well represented during the evaluation of the algorithm.

Depending on the exact nature of the classification problem, it may be possible to make statements regarding the strength of predictions. For example, after making a classification a learning algorithm could examine the training set for that classification. It could measure the effect of small changes to that training set; if small changes generate large effects, the training set is more vulnerable to manipulation.

6. THEORETICAL RESULTS

In this section we present an analytic model that examines a *causative* attack to manipulate a naive learning algorithm. The model's simplicity yields an optimal policy for the adversary and a bound on the effort required to achieve the adversary's objective. We interpret the resulting bound and discuss possible extensions to this model to capture more realistic settings.

We discuss an *outlier detection* technique. Outlier detection is the task of identifying anomalous data and is a widely used paradigm in fault detection [40], intrusion detection [23], and virus detection [33, 34]. We find the smallest region that contains some fixed percentage of the observed data, which is called the *support* of the data's distribution. The outlier detector classifies points inside the support as normal and those outside as anomalous. Outlier detection is often used in scenarios where anomalous data is scarce or novel anomalies could arise.

6.1 A Simple Model

One simple approach to outlier detection is to estimate the support of the normal data by a multi-dimensional *hypersphere*. As depicted in Figure 1(a) every point in the hypersphere is classified as normal and those outside the hypersphere are classified as outliers. The training algorithm fixes the radius of the hypersphere and centers it at the mean of the training data. The hypersphere can be fit into the learning framework presented above by a squared loss function, $\ell_{sphere}(\bar{X}, x_i) = \left(x_i - \bar{X}\right)^2$, where $\bar{X}$ is the centroid of the data $\{x_i\}$. It is easy to show that the parameter that minimizes Equation (1) is the mean of the training data.

To make the hypersphere adaptive, the hypersphere is retrained on new data allowing for a repeated attack. To prevent arbitrary data from being introduced, we employ a conservative retraining strategy that only admits new points to the training set if they are classified as normal; we say the classifier *bootstraps* itself. This learning framework is not meant to represent the state of the art in learning techniques; instead, it is a illustrative technique that allows for an exact analysis.

6.2 Attack Strategy

The attack we analyze involves an adversary determined to alter our detector to include a specific point G by constructing data to shift the hypersphere toward the target as the hypersphere is retrained. We assume the goal G is initially correctly classified as an anomaly by our algorithm. For instance, in the IDS domain, the adversary has an intrusion packet that our detector currently classifies as anomalous. The adversary wants to change the state of our detector to misclassify the packet as normal. This scenario is a *causative targeted integrity* attack. Before the attack, the hypersphere is centered at $\bar{X}_0$ and it has a fixed radius R. The attack is iterated over the course of $T > 1$ training iterations. At the i-th iteration the mean of the hypersphere is denoted by $\bar{X}_i$.

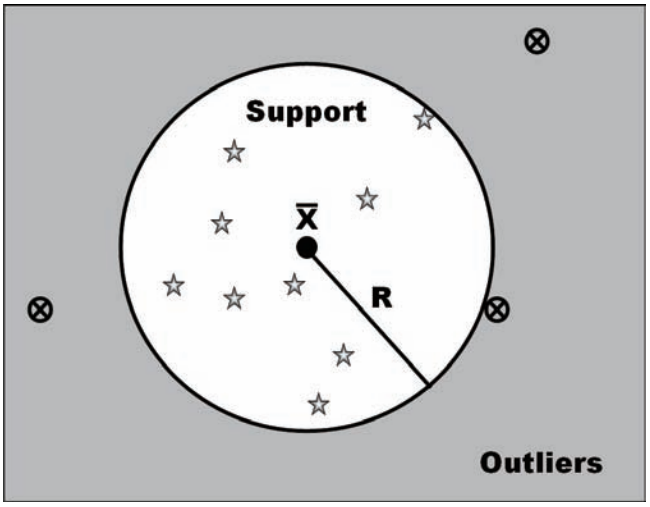
We give the adversary complete control: the adversary knows the algorithm, its feature set, and its current state, and all points are attack points. At each iteration, the bootstrapping policy retrains on all points that were classified as normal in a previous iteration. Under this policy, the adversary's optimal strategy is straightforward — as depicted in Figure 1(b) the adversary places points at the location where the line between the mean and G intersects with the boundary. This reduces the attack to a single dimension along this line. Suppose that in the i-th iteration, the adversary strategically places α_i points at the i-th optimal location achieving optimal displacement of the mean toward the adversary's goal, G. The effort of the adversary is measured by M defined as $\sum_{i=1}^{T} \alpha_i$.

Placing all attack points in the first iteration is not optimal. It achieves a finite shift while optimal strategies achieve unbounded gains. As we discuss below, the attack strategy must be balanced. The more points placed during an iteration, the further the hypersphere is displaced on that iteration. However, the points placed early in the attack effectively weigh down the hypersphere making it more difficult to move. The adversary must balance current gain against future gain. Another tradeoff is the number of rounds of iteration versus the total effort.

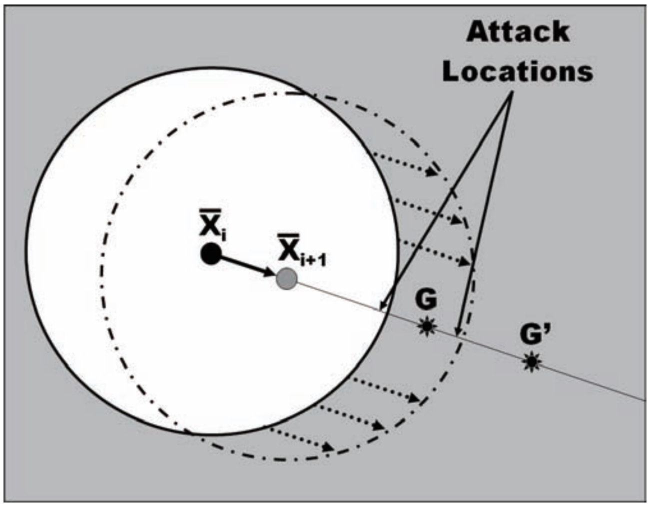
6.3 Optimal Attack Displacement

We calculate the displacement caused by a sequence $\{\alpha_i\}$ of attack points. For T iterations and M total attack points, the function $D_{R,T}(\{\alpha_i\})$ denotes the *relative displacement* caused by the attack sequence. The relative displacement is the total displacement over the radius of the hypersphere, $\frac{\bar{X}_T - \bar{X}_0}{R}$. Let M_i be defined as $\sum_{j=1}^{i} \alpha_j$, the cumulative mass. Using these terms, the relative distance is

$$D_{R,T}(\{M_i\}) = T - \sum_{i=2}^{T} \frac{M_{i-1}}{M_i} \qquad (4)$$



(a) Hypersphere Outlier Detection



(b) Attack on a Hypersphere Outlier Detector

Figure 1: Depictions of the concept of hypersphere outlier detection and the vulnerability of naive approaches. In Figure 1(a) a bounding hypersphere centered at $\bar{X}$ of fixed radius R is used to estimate the empirical *support* of a distribution excluding outliers. Samples from the "normal" distribution being modeled are indicated by $\star$ with three outliers indicated by $\otimes$. Meanwhile, Figure 1(b) depicts how an adversary with knowledge of the state of the outlier detector could shift the outlier detector toward a first goal G. It could take several iterations of attacks to shift the hypersphere further to include the second goal G'.

where we constrain $M_1 = 1$ and $M_T = M$ [25].

By finding an upper bound to Equation (4), we can bound the minimal effort M^* of the adversary. For a particular M, we desire an optimal sequence $\{M_i^*\}$ that achieves the maximum relative displacement, $D_{R,T}(M)$. If the adversary has no time constraint, the solution is $M_i^* = i$, which corresponds to placing a single point at each iteration. However, if the adversary expedites the attack to $T < M$ iterations, the optimal strategy is given by $M_i^* = M^{\frac{i-1}{T-1}}$. This value is not always an integer, so we have:

$$D_{R,T}(M) \leq T - (T-1) \cdot M^{\frac{-1}{T-1}} \leq T \tag{5}$$

6.4 Bounding the Adversary's Effort

From these results we find a bound on the adversary's effort M. Since $M \geq 1$ and $T > 1$, Equation (5) is monotonically increasing in M. If the desired relative displacement to the goal is D_R, the bound in Equation (5) can be inverted to bound the minimal effort M^* required to achieve the goal. Since $D_R < T$, this bound is given by:

$$M^* \geq \left(\frac{T-1}{T-D_R}\right)^{T-1} \tag{6}$$

The bound in Equation (6) gives us a worst-case bound on the adversary's capability when the adversary has complete control of the learner's training. For large relative displacements $D_R > 1$, the bound decreases exponentially as the number of iterations is increased. The bound has a limiting value of $M^* \geq e^{D_R - 1}$. The adversary must tradeoff between using a large number of attack points or extending the attack over many iterations. A tightly-fit hypersphere with small radius will be more robust since our displacement is relative to its radius.

An apparent deficiency of this analysis is the weak bound of $M^* \geq \epsilon$ where $0 < \epsilon \leq 1$ that occurs when $D_R \leq 1$. This an important range since the adversary's goal may be near the boundary. The deficiency comes directly from our assumption of complete adversarial control. The lack of initial non-adversarial data allows our adversary to ensure a first step of one radius regardless of M. Therefore, the adversary can reach the objective of $D_R \leq 1$ with any $M \geq 1$ in a single iteration.

A more complex model could allow for initial data. By considering an initial N training points that support the hypersphere before the attack, we can obtain a stronger bound:

$$M^* \geq N\left[e^{D_R} - 1\right] \tag{7}$$

This stronger bound ensures that even for small D_R, the adversary's effort is a multiple of N that increases exponentially in the desired displacement [25].

We could extend the model by adding non-adversarial data at every training iteration, for this corresponds to scenarios where the adversary only controls part of the data.

7. CONCLUSIONS

7.1 Related Work

The earliest theoretical work we know of that approaches learning in the presence of an adversary was done by Kearns and Li [15]. They worked in the context of Valiant's Probably Approximately Correct (PAC) learning framework [35, 36], extending it to prove bounds for maliciously chosen errors in the training data. Specifically, they proved that if the learner is to perform correctly, in general the fraction of training points controlled by the adversary must be less than $\epsilon/(1+\epsilon)$, where ϵ is the desired bound on classification errors by the learner [4, 6, 30].

Results from game theory may be relevant to adversarial learning systems. In particular, *deception games* involve players that have partial information and influence the information seen by other players. Some of these games involve continuous variables generated by various probability distributions [5, 9, 17, 29, 32], while others apply to scenarios with discrete states [14]. This work and adversarial learning both ask many of the same questions, and they both address the same underlying issues. Integration of game theoretic concepts is a promising direction for work in this area.

Dalvi et al. examine the learn-adapt-relearn cycle from a game-theoretic point of view [8]. In their model, the learner has a cost for measuring each feature of the data and the adversary has a cost for changing each feature in attack points. If the adversary and learner have complete information about each other and we accept some other assumptions, they find an optimal strategy for the learner to defend against the adversary's adaptations.

Research has also begun to examine the vulnerability of learners to reverse engineering. Lowd and Meek introduce a novel learning problem for adversarial classifier reverse engineering in which an adversary conducts an attack that minimizes a cost function [21]. Under their framework, Lowd and Meek construct algorithms for reverse engineering linear classifiers. Moreover, they build an attack to reverse engineer spam filters [22].

Although they are not machine learning systems, publicly verifiable digital watermarks also must deal with sensitivity (probing) attacks. An information theoretic analysis of the sensitivity attack quantifies the amount of information revealed per probe. Randomization of thresholds within the watermark verification algorithm increase the number of probes necessary to remove a digital watermark [19].

An interesting junction of learning and game theory has dealt with combining advice from a set of experts to predict a sequence with the goal of doing at least as well as the best expert in all possible sequences [7, 13, 37]. In this domain, adaptive weighting schemes are used to combine the experts, each accessed by how well it performs compared to the best expert for an adversarially chosen sequence. Amongst these schemes are the Aggregating Algorithm [37] and the Weighted Majority Algorithm [20].

There has also been work on attacking statistical spam filters. Wittel and Wu [39] discuss the possibility of crafting attacks designed to take advantage of the statistical nature of such spam filters, and they implement a simple attack. John Graham-Cumming describes implementing an attack he calls "Bayes vs. Bayes," in which the adversary trains a second statistical spam filter based on feedback from the filter under attack and then uses the second filter to find words that make spam messages undetectable by the original filter [10].

Methods exist to perform exact learning of a concept using answers to a series of queries. These queries return a counterexample when a "no" response is generated. In many scenarios, it has been shown that learning is possible even in the worst case [2].

Control theory has been proposed as an alternative to game theory and search oriented expert-systems for military command and control systems [12]. The motivation behind this proposal is the difficulty associated with modeling (or even predicting) the goals of a military adversary.

7.2 Research Directions

Can machine learning be secure? Does adding machine learning to a system introduce vulnerability? This paper proposes a framework for understanding these questions. We present a model for describing attacks against learning algorithms, and we analyze a simple attack in detail. We discuss potential defenses against attacks and speculate about their effectiveness.

Here we lay out the directions for research that we see as most promising. To evaluate and ensure the security of machine learning, these are among the most important areas that must be addressed:

Information

How crucial is it to keep information secret from an adversary? If an adversary has full knowledge of the system, are all the *exploratory* attacks trivial? If the adversary has no knowledge about the system, which attacks are still possible?

Arms race

Can we avoid arms races in online learning systems? Arms races have occurred in spam filters. Can game theory suggest a strategy for secure re-training?

Quantitative measurement

Can we measure the effects of attacks? Such information would allow comparison of the security performance of learning algorithms. We could calculate risk based on probability and damage assessments of attacks.

Security proofs

Can we bound the amount of information leaked by the learner? If so, we can bound the accuracy of the adversary's approximation of the learner's current state.

Detecting adversaries

Attacks introduce potentially detectable side effects such as drift, unusual patterns in the data observed by the learner, etc. These attacks are more pronounced in online learning. When do these side effects reveal the adversary's attack?

8. ACKNOWLEDGMENTS

Thanks to Michael I. Jordan, Peter Bartlett and David Molnar for their insightful discussions and comments regarding this work. We gratefully acknowledge support from the National Science Foundation and the Homeland Security Advanced Research Projects Agency. The views expressed here are solely those of the authors and do not necessarily reflect the views of the funding agencies or any agency of the U.S. government.

9. REFERENCES

[1] I. Androutsopoulos, J. Koutsias, K. V. Chandrinos, G. Paliouras, and C. D. Spyropolous. An evaluation of naive Bayesian anti-spam filtering. *Proceedings of the Workshop on Machine Learning in the New Information Age*, pages 9–17, 2000.

[2] D. Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, Apr. 1988.

[3] Apache, http://spamassassin.apache.org/. *SpamAssassin*.

[4] P. Auer. Learning nested differences in the presence of malicious noise. *Theoretical Computer Science*, 185(1):159–175, 1997.

[5] V. J. Baston and F. Bostock. Deception games. *International Journal of Game Theory*, 17(2):129–134, 1988.

[6] N. H. Bshouty, N. Eiron, and E. Kushilevitz. PAC learning with nasty noise. *Theoretical Computer Science*, 288(2):255–275, 2002.

[7] N. Cesa-Bianchi, Y. Freund, D. P. Helmbold, D. Haussler, R. E. Schapire, and M. K. Warmuth. How to use expert advice. *Journal of the ACM*, 44(3):427–485, May 1997.

[8] N. Dalvi, P. Domingos, Mausam, S. Sanghai, and D. Verma. Adversarial classification. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 99–108, Seattle, WA, 2004. ACM Press.

[9] B. Fristedt. The deceptive number changing game in the absence of symmetry. *International Journal of Game Theory*, 26:183–191, 1997.

[10] J. Graham-Cumming. How to beat an adaptive spam filter. Presentation at the MIT Spam Conference, Jan. 2004.

[11] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference and Prediction*. Springer, 2003.

[12] S. A. Heise and H. S. Morse. The DARPA JFACC program: Modeling and control of military operations. In *Proceedings of the 39th IEEE Conference on Decision and Control*, pages 2551–2555. IEEE, 2000.

[13] M. Herbster and M. K. Warmuth. Tracking the best expert. *Machine Learning*, 32(2):151–178, Aug. 1998.

[14] J. P. Hespanha, Y. S. Ateşkan, and H. H. Kizilocak. Deception in non-cooperative games with partial information. In *Proceedings of the 2nd DARPA-JFACC Symposium on Advances in Enterprise Control*, 2000.

[15] M. Kearns and M. Li. Learning in the presence of malicious errors. *SIAM Journal on Computing*, 22:807–837, 1993.

[16] A. Lazarevic, L. Ertöz, V. Kumar, A. Ozgur, and J. Srivastava. A comparative study of anomaly detection schemes in network intrusion detection. In D. Barbará and C. Kamath, editors, *Proceedings of the Third SIAM International Conference on Data Mining*, May 2003.

[17] K.-T. Lee. On a deception game with three boxes. *International Journal of Game Theory*, 22:89–95, 1993.

[18] Y. Liao and V. R. Vemuri. Using text categorization techniques for intrusion detection. In *Proceedings of the 11th USENIX Security Symposium*, pages 51–59, Aug. 2002.

[19] J.-P. M. Linnartz and M. van Dijk. Analysis of the sensitivity attack against electronic watermarks in images. In D. Aucsmith, editor, *Information Hiding '98*, pages 258–272. Springer-Verlag, 1998.

[20] N. Littlestone and M. K. Warmuth. The weighted majority algorithm. *Information and Computation*, 108(2):212–261, 1994.

[21] D. Lowd and C. Meek. Adversarial learning. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 641–647, 2005.

[22] D. Lowd and C. Meek. Good word attacks on statistical spam filters. In *Proceedings of the Second Conference on Email and Anti-Spam (CEAS)*, 2005.

[23] M. V. Mahoney and P. K. Chan. Learning nonstationary models of normal network traffic for detecting novel attacks. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 376–385, 2002.

[24] S. Mukkamala, G. Janoski, and A. Sung. Intrusion detection using neural networks and support vector machines. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN'02)*, pages 1702–1707, 2002.

[25] B. Nelson. Designing, Implementing, and Analyzing a System for Virus Detection. Master's thesis, University of California at Berkeley, Dec. 2005.

[26] V. Paxson. Bro: A system for detecting network intruders in real-time. *Computer Networks*, 31(23):2435–2463, Dec. 1999.

[27] N. Provos. A virtual honeypot framework. In *Proceedings of the 13th USENIX Security Symposium*, 2004.

[28] R. Raina, A. Y. Ng, and D. Koller. Transfer learning by constructing informative priors. In *Neural Information Processing Systems Workshop on Inductive Transfer: 10 Years Later*, 2005.

[29] M. Sakaguchi. Effect of correlation in a simple deception game. *Mathematica Japonica*, 35(3):527–536, 1990.

[30] R. A. Servedio. Smooth boosting and learning with malicious noise. *Journal of Machine Learning Research (JMLR)*, 4:633–648, Sept. 2003.

[31] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[32] J. Spencer. A deception game. *American Math Monthly*, 80:416–417, 1973.

[33] S. J. Stolfo, S. Hershkop, K. Wang, O. Nimeskern, and C. W. Hu. A behavior-based approach to secure email systems. In *Mathematical Methods, Models and Architectures for Computer Networks Security*, 2003.

[34] S. J. Stolfo, W. J. Li, S. Hershkop, K. Wang, C. W. Hu, and O. Nimeskern. Detecting viral propagations using email behavior profiles. In *ACM Transactions on Internet Technology*, 2004.

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

[36] L. G. Valiant. Learning disjunctions of conjunctions. In *Proceedings of the 9th International Joint Conference on Artificial Intelligence*, pages 560–566, 1985.

[37] V. Vovk. Aggregating strategies. In M. Fulk and J. Case, editors, *Proceedings of the 7th Annual Workshop on Computational Learning Theory*, pages 371–383, San Mateo, CA, 1990. Morgan-Kaufmann.

[38] L. Wehenkel. Machine learning approaches to power system security assessment. *IEEE Intelligent Systems and Their Applications*, 12(5):60–72, Sept.–Oct. 1997.

[39] G. L. Wittel and S. F. Wu. On attacking statistical spam filters. In *Proceedings of the First Conference on Email and Anti-Spam (CEAS)*, 2004.

[40] W. Xu, P. Bodik, and D. Patterson. A flexible architecture for statistical learning and data mining from system log streams. In *Temporal Data Mining: Algorithms, Theory and Applications*, Brighton, UK, Nov. 2004. The Fourth IEEE International Conference on Data Mining.

[41] D.-Y. Yeung and C. Chow. Parzen-window network intrusion detectors. In *Proceedings of the Sixteenth International Conference on Pattern Recognition*, pages 385–388, Aug. 2002.

[42] K. Yu and V. Tresp. Learning to learn and collaborative filtering. In *Neural Information Processing Systems Workshop on Inductive Transfer: 10 Years Later*, 2005.