Using Machine Learning for Operational Decisions in Adversarial Environments

Yevgeniy Vorobeychik[1] and John Ross Wallrabenstein[2]

[1] Vanderbilt University
Electrical Engineering and Computer Science
Nashville, TN
yevgeniy.vorobeychik@vanderbilt.edu
[2] Purdue University
Computer Science
West Lafayette, IN
jwallrab@cs.purdue.edu

Abstract. Classical supervised learning assumes that training data is representative of the data expected to be observed in the future. This assumption is clearly violated when an intelligent adversary actively tries to deceive the learner by generating instances very different from those previously seen. The literature on adversarial machine learning aims to address this problem, but often assumes constraints that sophisticated and determined adversaries need not abide by. We model the adversarial machine learning problem by considering an unconstrained, but utility-maximizing, adversary. In addition, rather than modifying the learning algorithm to increase its robustness to adversarial manipulation, we use an output of an arbitrary probabilistic classifier (such as Naïve Bayes) in a linear optimization program that computes optimal randomized operational decisions based on machine learning predictions, operational constraints, and our adversarial model. Our approach is simpler than its predecessors, highly scalable, and we experimentally demonstrate that it outperforms the state of the art on several metrics.

1 Introduction

In a classical supervised learning setting one starts with a data set of instances generated according to some fixed distribution, and learns a function which (one hopes) effectively evaluates new instances generated from the same distribution. While this assumption is often reasonable, it is clearly violated in adversarial settings. For example, if machine learning is used for network intrusion detection, an intelligent adversary will try to avoid detection by deliberately changing behavior to appear benign.

We study the problem of *adversarial machine learning*, which we view as a game between a *defender* (*learner*), who uses past data to predict and respond to potential threats, and a collection of *attackers* who aim to bypass defensive response to their activities while achieving some malicious end. The issue of learning in adversarial environments has been addressed from a variety of angles, such as robustness to data corruption [1], analysis of the problem of manipulating a learning algorithm [2, 3], and design of learning algorithms in adversarial settings [4–7]. The approaches aspiring to adjust learning algorithms to cope with adversarial response suffer from the following five problems: 1) the proposals require significant restrictions on the nature of learning algorithms (e.g., logistic or linear loss functions); 2) it is typically assumed that the attacker is concerned with the total loss (error), whereas it is usually more reasonable to assume that the attackers are interested in false negatives (i.e., not being detected); 3) methods to date fail to account for operational constraints, making them largely impractical [8]; 4) current methods make strong restrictions on transformation of test data by attackers (e.g., a common assumption is a linear transformation), and 5), current methods are almost universally restricted to deterministic decisions for the learner, and cannot take advantage of the power of randomization in adversarial settings [9]. Most of these limitations are due to the fact that past approaches attempt to modify a learning algorithm to account for adversarial behavior. In contrast, the approach we take separates the problem of prediction, for which machine learning is used, from the problem of operational response that uses machine learning to quantify uncertainty for the associated optimization problem. This approach enables us to use an arbitrary machine learning algorithm *without modification* (as a black box), as we embed adversarial reasoning into the optimization problem which determines operational decisions, rather than into the learning algorithm. Our key insight is to interpret the predictions obtained from past data (using learning) as *revealed preferences of the attackers* regarding "ideal" malicious content. Our approach is to use the resulting quantification of uncertainty about attacker preferences in a Bayesian Stackelberg game model to optimally compute randomized operational decisions under budget constraints. We develop a linear programming approach for computing optimal commitment in this game, where the attackers are modeled as utility maximizers.

In all, we make the following contributions:

1. A general framework for optimizing operational decisions based on machine learning,
2. a model of attacker evasion of a randomized classification scheme
3. a linear programming formulation to compute optimal oeprational decisions under budget constraints, and
4. an extensive evaluation of our approach, which we show to significantly outperform the state of the art.

2 Model

We consider the problem of adversarial binary classification over a space $\mathcal{X}$ of inputs, where each input feature vector $\boldsymbol{x} \in \mathcal{X}$ can be categorized as benign or malicious. The defender, $\mathcal{D}$, starts with a data set of labeled instances,

$\mathcal{I} = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m)\}$, which we assume to accurately represent the current distribution of input instances and corresponding categories.[3] $\mathcal{D}$ then uses an algorithm of choice, such as Naive Bayes, to obtain a probabilistic classifier $p(\boldsymbol{x})$ which assigns to an arbitrary input vector a probability that it (or, rather, a producer of it) is malicious. In traditional application of machine learning, adversarial or not, one would then use a threshold, θ, and classify an instance $\boldsymbol{x}$ as malicious if $p(\boldsymbol{x}) \geq \theta$, and benign otherwise, with adversarial aspects of the problem folded into the algorithm that derives the function $p(\cdot)$. It is on this point that our approach diverges from current art. Specifically, we introduce a function $q(\boldsymbol{x}, p(\cdot)) \in [0, 1]$ which prescribes a possibly randomized operational decision (e.g., the probability of filtering an email or manually investigating an observed network access pattern) for an instance $\boldsymbol{x}$ given a prediction $p(\boldsymbol{x})$. Clearly, the threshold function typically used is a special case, but we will productively consider alternative possibilities. To simplify notation, where $p(\cdot)$ is clear from context, we use instead $q(\boldsymbol{x})$, keeping in mind its implicit dependence on the prediction made by the learning algorithm.

We model the adversarial machine learning setting as a Stackelberg game between a defender and a population of attackers. In this game, the defender moves first, choosing $q(\cdot)$. Next, the attackers learn $q(\cdot)$ (for example, through extensive probing), and each attacker subsequently chooses an input vector $\boldsymbol{x}$ (e.g., a phishing email) so as to maximize their expected return (a combination of bypassing defensive countermeasures and achieving a desired outcome upon successfully penetrating the defense, such as a high response rate to a phishing attack). Our assumption that the operational policy $q(\cdot)$ is known to attackers reflects threats that have significant time and/or resources to probe and respond to defensive measures, a feature characteristic of advanced cyber criminals [10].

We view the data set $\mathcal{I}$ of labeled malware instances as representing *revealed preferences* of a sample of attackers, that is, their preference for input vectors $\boldsymbol{x}$ (if an attacker preferred another input $\boldsymbol{x}'$, we assume that this attacker would have chosen $\boldsymbol{x}'$ instead of $\boldsymbol{x}$). To appreciate this modeling choice, it is worth noting that much variation in malware is due either to differences in perpetrators themselves, or differences in their goals (even for the same attackers), and labeled data provides information, albeit indirectly, about these differences. Therefore, in our framework $p(\boldsymbol{x})$ takes on a dual-meaning: first, it is the probability that $\boldsymbol{x}$ reflects a malicious action, and second, if malicious, $\boldsymbol{x}$ represents an attacker's "type", or ideal method of attack. Insofar as we view an attack $\boldsymbol{x}$ as ideal for an attacker, it is just as natural to posit that an attacker would prefer attack patterns that are close to $\boldsymbol{x}$ in feature space to those distant from it. For example, a model in which an attacker would minimize the number of feature values to alter in order to bypass defensive activities has this characteristic, as do models which use a regularization term to reduce the scale of attack manipulation of data [11, 2, 4, 6, 7].

[3] The problem of adversarial tampering of such *training* data is outside the scope of our work, and can be viewed as an extension of our setup.

Suppose that if an attack $\boldsymbol{x}$, succeeds, the attacker gains $V(\boldsymbol{x})$, which is also the value lost to the defender. On the other hand, if an attack is filtered or caught by the defender, both receive 0. Finally, if the attacker with a preference for $\boldsymbol{x}$ chooses an alternative attack vector $\boldsymbol{x}'$, his utility from successfully bypassing defenses becomes $V(\boldsymbol{x})Q(\boldsymbol{x}, \boldsymbol{x}')$, where

$$Q(\boldsymbol{x}, \boldsymbol{x}') = e^{-\delta||\boldsymbol{x}-\boldsymbol{x}'||}, \tag{1}$$

with $||\cdot||$ a norm (we use Hamming distance), and δ corresponding to importance of being close to the preferred $\boldsymbol{x}$. Observe that when $\delta = 0$, the attacker is indifferent among attack vectors, and all that matters is success at bypassing defensive action, while $\delta \to \infty$ results in an attacker who does not react to defensive action at all, either because it is too costly to change, or because this attacker simply does not have the capability of doing so (e.g., someone who merely reuses attack templates previously developed by others). The full utility function of an attacker with type $\boldsymbol{x}$ for choosing another input $\boldsymbol{x'}$ when the defense strategy is $q(\cdot)$ is then

$$\mu(\boldsymbol{x}, \boldsymbol{x'}; q) = V(\boldsymbol{x})Q(\boldsymbol{x}, \boldsymbol{x'})(1 - q(\boldsymbol{x'})), \tag{2}$$

since $1 - q(\cdot)$ is the probability that the attacker successfully bypasses the defensive action.

While the above attacker model admits considerable generality, we assume that attackers fall into two classes: adaptive, as described above, and static, corresponding to the limiting case of $\delta \to \infty$. Let $v_t(\boldsymbol{x}; q)$ be the value function of an attacker with class (type) t and preference for $\boldsymbol{x}$, when the defender chooses a policy q. $v_t(\boldsymbol{x}; q)$ represents the maximum utility that the attacker with type t can achieve given q. For a static attacker, the value function is

$$v_S(\boldsymbol{x}; q) = V(\boldsymbol{x})(1 - q(\boldsymbol{x})),$$

that is, a static attacker always uses his preferred input $\boldsymbol{x}$, and receives his corresponding value for it whenever the defender (operator) does not take action upon observing $\boldsymbol{x}$. For an adaptive attacker, the value function is

$$v_A(\boldsymbol{x}; q) = \max_{\boldsymbol{x'} \in \mathcal{X}} \mu(\boldsymbol{x}, \boldsymbol{x'}; q),$$

that is, the maximum utility that the attacker obtains from using an *arbitrary* input $\boldsymbol{x'}$ (that is, we assume that the adaptive attacker is unconstrained). Finally, let P_A be the probability that an arbitrary malicious input was generated by an adaptive adversary; the probability that the adversary was static is then $P_S = 1 - P_A$.

Having described in some detail our model of the adversarial response to defensive choice of $q(\cdot)$, we now turn to the objective of the defender. At the high level, a natural goal for the defender is to maximize expected value of benign traffic that is classified as benign, less the expected losses due to attacks that successfully bypass the operator (i.e., incorrectly classified as benign). Presently, we show that a special case of this is equivalent to maximizing accuracy or minimizing loss. To formalize, we make two assumptions. First, we assume that the set of all possible instances $\mathcal{X}$ is finite, and use $\boldsymbol{q}$ and $\boldsymbol{p}$ as vectors corresponding to $q(\boldsymbol{x})$ and $p(\boldsymbol{x})$ respectively, using some fixed arbitrary ordering over $\mathcal{X}$. This assumption is clearly unrealistic (even if $\mathcal{X}$ is technically finite, it will typically be intractably large), but will help with exposition below. We subsequently (in Section 3) describe how to apply our approach in practice, when this assumption will not hold. Second, we assume that the defender gains a positive value $G(\boldsymbol{x})$ from a benign input $\boldsymbol{x}$ only if it is not inspected. In the case of email traffic, this is certainly sensible if our action is to filter a suspected email. More generally, inspection can be a lengthy process, in which case we can interpret $G(\boldsymbol{x})$ as the value of time lost if $\boldsymbol{x}$ is, in fact, benign, but is carefully screened before it can have its beneficial impact. Formally, we suppose that the defender maximizes $U_{\mathcal{D}}(\boldsymbol{q}, \boldsymbol{p}, \mathcal{X})$, defined as

$$U_{\mathcal{D}}(\boldsymbol{q}, \boldsymbol{p}, \mathcal{X}) = \sum_{\boldsymbol{x} \in \mathcal{X}} [(1 - q(\boldsymbol{x}))G(\boldsymbol{x})(1 - p(\boldsymbol{x})) - p(\boldsymbol{x})(P_S v_S(\boldsymbol{x}; q) + P_A v_A(\boldsymbol{x}; q))] . \qquad (3)$$

To appreciate that this formal definition of the defender's objective is sensible, let us first rewrite it for a special case when $V(\boldsymbol{x}) = G(\boldsymbol{x}) = 1$ and $P_S = 1$, reducing the utility function to $\sum_{\boldsymbol{x} \in \mathcal{X}} (1 - q(\boldsymbol{x}))(1 - p(\boldsymbol{x})) - p(\boldsymbol{x})(1 - q(\boldsymbol{x}))$. Since $p(x)$ is constant, this is equivalent to minimizing

$$\sum_{\boldsymbol{x} \in \mathcal{X}} q(\boldsymbol{x})(1 - p(\boldsymbol{x})) + p(\boldsymbol{x})(1 - q(\boldsymbol{x})),$$

or, for each $\boldsymbol{x}$, the sum of probability that it is benign and misclassified as malicious, and probability that it is malicious but misclassified as benign; i.e., *expected loss.*

The final aspect of our model is a resource constraint on the defender. Sommer and Paxson [8] identify the cost of false positives and the gap between the output of machine learning algorithms and its use in operational decisions as two of the crucial gaps that prevent widespread use of machine learning in network intrusion detection. Our framework directly addresses the latter point, and we now turn focus to the former. False positives are quite costly because following up on an alert is a very expensive proposition, involving the use of a scarce resource, a security expert's time understanding the nature of the alert. In practice, it is simply not feasible to follow up on every alert, and there is a need for a principled approach that accounts for such budget constraints. An additional cost of false positives comes from the fact that, depending on the nature of operational decision, it results in some loss of value, either because a valuable email gets filtered, or because important communication is delayed due to deeper inspection it needs to undergo. In fact, $G(\boldsymbol{x})$ in our model already serves to quantify this loss of value. We handle the typically harder constraint on defensive resources by introducing a budget constraint, where we ensure that our solution inspects at most a fraction c of events, on average.

3 Computing Optimal Operational Decisions

Now that we have described our model of adversarial machine learning, the natural next question is: how do we solve it? Since our objective and constraints are linear (using the assumption that the attacker's gains translate directly into defender's losses), we can formulate our optimization problem as the following linear program (LP):

$$\begin{aligned}
\max_{\boldsymbol{q}} \quad & U_{\mathcal{D}}(\boldsymbol{q}, \boldsymbol{p}, \mathcal{X}) && && \text{(4a)}\\
\text{s.t.}: \quad & 0 \leq q(\boldsymbol{x}) \leq 1 && \forall\, \boldsymbol{x} \in \mathcal{X} && \text{(4b)}\\
& v_A(\boldsymbol{x}; q) \geq \mu(\boldsymbol{x}, \boldsymbol{x'}; q) && \forall\, \boldsymbol{x}, \boldsymbol{x'} \in \mathcal{X} && \text{(4c)}\\
& v_S(\boldsymbol{x}; q) = V(\boldsymbol{x})(1 - q(\boldsymbol{x})) && \forall\, \boldsymbol{x} \in \mathcal{X} && \text{(4d)}\\
& \sum_{\boldsymbol{x}} q(\boldsymbol{x}) \leq c|\mathcal{X}|. && && \text{(4e)}
\end{aligned}$$

Since the number of variables in this LP is linear in $|\mathcal{X}|$, while the number of constraints is quadratic in this quantity, clearly we cannot hope to use this when the space of all possible inputs is large (let alone infinite). Note, however, that we only need to compute the decisions $q(\boldsymbol{x})$ for inputs $\boldsymbol{x}$ we actually see in reality. Therefore, in practice we batch observations into small sets $\bar{\mathcal{X}} \subset \mathcal{X}$, and solve this optimization program using inputs restricted to $\bar{\mathcal{X}}$. In this setup, we assume that the attacker spends significant offline effort probing the classification scheme to learn the probabilities $q(\boldsymbol{x})$, and deploys an optimal attack once these are satisfactorily learned. Consequently, the game is *effectively* one-shot.

A natural sanity check that our formulation is reasonable is that the solution is particularly intuitive when there is no budget constraint or adaptive adversary. We now show that in this case, the policy $q(\boldsymbol{x})$ which uses a simple threshold on $p(\boldsymbol{x})$ (as commonly done) is, in fact optimal.

Proposition 1. *Suppose that $P_A = 0$ and $c = 1$ (i.e., no budget constraint). Then the optimal policy is*

$$q(\boldsymbol{x}) = \begin{cases} 1 & \text{if } p(\boldsymbol{x}) \geq \frac{G(\boldsymbol{x})}{G(\boldsymbol{x})+V(\boldsymbol{x})} \\ 0 & o.w. \end{cases}$$

Proof. Since we consider only static adversaries and there is no budget constraint, the objective becomes

$$\max_{\boldsymbol{q}} \sum_{\boldsymbol{x} \in \mathcal{X}} \left[(1 - q(\boldsymbol{x}))G(\boldsymbol{x})(1 - p(\boldsymbol{x})) - p(\boldsymbol{x})v_S(\boldsymbol{x})\right],$$

and the only remaining constraint is that $q(\boldsymbol{x}) \in [0, 1]$ for all $\boldsymbol{x}$. Since now the objective function is entirely decoupled for each $\boldsymbol{x}$, we can optimize each $q(\boldsymbol{x})$ in isolation for each $\boldsymbol{x} \in \mathcal{X}$. Rewriting, maximizing the objective for a given $\boldsymbol{x}$ is equivalent to minimizing $q(\boldsymbol{x})[G(\boldsymbol{x}) - p(\boldsymbol{x})(G(\boldsymbol{x}) + V(\boldsymbol{x}))]$. Whenever the right multiplicand is negative, the quantity is minimized when $q(\boldsymbol{x}) = 1$, and when it is positive, the quantity is minimized when $q(\boldsymbol{x}) = 0$. Since $p(\boldsymbol{x}) \geq \frac{G(\boldsymbol{x})}{G(\boldsymbol{x})+V(\boldsymbol{x})}$ implies that the right multiplicand is negative (more accurately, non-positive), the result follows.

While traditional approaches threshold an odds ratio (or log-odds) rather than the probability $p(\boldsymbol{x})$, the two are, in fact equivalent. To see this, let us consider the generalized (cost-sensitive) threshold on odds ratio used by the Dalvi et al. [11] model. In their notation, $U_{\mathcal{C}}(+,+)$, $U_{\mathcal{C}}(+,-)$, $U_{\mathcal{C}}(-,+)$, and $U_{\mathcal{C}}(-,-)$ denote the utility of the defender (classifier) when he correctly identifies a malicious input, incorrectly identifies a benign input, incorrectly identifies a malicious input, and correctly identifies a benign input, respectively. In our setting, we have $U_{\mathcal{C}}(+,+) = 0$ (i.e., no loss), $U_{\mathcal{C}}(+,-) = 0$ (and capture the costs of false positives as operational constraints instead), $U_{\mathcal{C}}(-,+) = -V(\boldsymbol{x})$, and $U_{\mathcal{C}}(-,-) = G(\boldsymbol{x})$ (note that we augment the utility functions to depend on input vector $\boldsymbol{x}$). The odds-ratio test used by Dalvi et al. therefore checks

$$\frac{p(\boldsymbol{x})}{1-p(\boldsymbol{x})} \geq \frac{U_{\mathcal{C}}(-,-) - U_{\mathcal{C}}(+,-)}{U_{\mathcal{C}}(+,+) - U_{\mathcal{C}}(-,+)} = \frac{G(x)}{V(x)}. \tag{5}$$

and it is easy to verify that inequality 5 is equivalent to the threshold test in Proposition 1.

Consider now a more general setting where $P_A = 0$, but now with a budget constraint. In this context, we now show that the optimal policy is to first set $q(\boldsymbol{x}) = 0$ for all $\boldsymbol{x}$ with $p(\boldsymbol{x})$ below the threshold described in Proposition 1, then rank the remainder in descending order of $p(\boldsymbol{x})$, and assign $q(\boldsymbol{x}) = 1$ in this order until the budget is exhausted.

Proposition 2. *Suppose that $P_A = 0$ and $c|\mathcal{X}|$ is an integer. Then the optimal policy is to let $q(\boldsymbol{x}) = 0$ for all $\boldsymbol{x}$ with*

$$p(\boldsymbol{x}) < \frac{G(\boldsymbol{x})}{G(\boldsymbol{x}) + V(\boldsymbol{x})}.$$

Rank the remaining $\boldsymbol{x}$ in descending order of $p(\boldsymbol{x})$ and set $q(\boldsymbol{x}) = 1$ for the top $c|\mathcal{X}|$ inputs, with $q(\boldsymbol{x}) = 0$ for the rest.

Proof. The LP can be rewritten so as to minimize

$$\sum_{\boldsymbol{x}} q(\boldsymbol{x})[G(\boldsymbol{x}) - p(\boldsymbol{x})(G(\boldsymbol{x}) + V(\boldsymbol{x}))]$$

subject to the budget constraint. By the same argument as above, whenever $p(\boldsymbol{x})$ is below the threshold, the optimal $q(\boldsymbol{x}) = 0$. Removing the corresponding $\boldsymbol{x}$ from the objective, we obtain a special knapsack problem in which the above greedy solution is optimal, since the coefficient on the budget constraint is 1.

In a nutshell, Proposition 2 suggests an intuitive policy that whenever the budget constraint binds, we should simply inspect the highest priority items. Therefore, randomization becomes important only when there is an adversary actively responding to our inspection efforts.

4 Experiments

Experimentally validating a scheme for adversarial machine learning is inherently difficult using publicly available data, such as spam. The reason is that insofar as this data captures evolution of spam, it is in response to the ecology of spam filters, and, in addition, the precise nature of the actually deployed filters is not a part of such public databases. In addition, spam is in itself a rather benign attack, as compared to, say, a spear phish aimed at stealing intellectual property. The latter is clearly much more targeted, much more costly to the organizations, and involves far more sophisticated and adaptive adversaries. All of the previous attempts to address machine learning in adversarial settings struggled with this problem, and evaluation is typically either (a) nevertheless involving public spam data [4–7], or (b) generating synthetic data according to their model of the adversary [11, 7]. We do *both*: evaluate our approach on *actual public spam data*, and using synthetically generated attacks. There is a clear limitation of using one's own model for validation: it naturally favors the proposed approach if the model is assumed to be an accurate description of attacker's behavior. We address this limitation by evaluating the robustness of our approach to errors in the adversarial model it uses (see online appendix, `http://appendices.webs.com/amlrobust.pdf`).

4.1 Setup

In all our experiments we use the TREC spam corpora from $2005 - 2008$. First, we use this data as is to compare the performance of our approach in a spam filtering task, compared to state-of-the-art alternatives. Subequently, we use this data only for training, and simulate adversarial behavior according to our model (as done, for example, by Dalvi et al. [11]). Throughout, we performed 10-fold cross-validation and analyzed the results using the approach outlined by Demšar [12]. We compare our approach against using a classifier it is based upon (i.e., $p(\boldsymbol{x})$) directly using pairs of the form $\{\mathcal{C}, E[OPT(\mathcal{C})]\}$, where $\mathcal{C}$ is the classifier providing $p(\boldsymbol{x})$ for our model, and $E[OPT(\mathcal{C})]$ denotes our approach using $\mathcal{C}$. We use Friedman's test to compute the p-values, using $N = 4$ data sets and $k = 2$ classifiers, as we are only concerned with the performance of our approach with respect to the corresponding classifier. We use the post-hoc Bonferroni test, which does not alter α as $\alpha/(k-1) = \alpha$ when $k = 2$, as in all of our comparisons. As detailed by Salzberg [13], the feature criteria were chosen to optimize the performance of Naïve Bayes on the TREC 2005 spam corpus. Feature vectors were generated from the raw emails, and the same criteria were used for each corpus. None of the algorithms have been optimized or tuned on *future* data. Below we train on a fold of the TREC 2005 data, evaluate the performance over the test fold for the TREC 2005 corpus, and test over the entire set of future corpora.

Our approach uses predictions $p(\boldsymbol{x})$ obtained using three existing classifiers: Naïve Bayes (our non-adversarial baseline), and the adversarial classifiers developed by Bruckner and Scheffer [6] and Dalvi [11], which are state-of-the-art alternatives.[4] We denote the expected utility of our approach as $E[OPT(\cdot)]$, where the argument is an existing classifier that provides $p(\boldsymbol{x})$. We solve the LP (Equations 4a-4e) using CPLEX version 12.2.

Our optimization approach explicitly bounds the number of instances that can be inspected. We consider two principled ways of imposing the same restriction on existing classifiers:

1. Let $\bar{\mathcal{X}}$ be all $\boldsymbol{x}$ with $p(\boldsymbol{x})$ above a threshold from Proposition 1. Then set $q(\boldsymbol{x}) = 1$ if $|\bar{\mathcal{X}}| \leq c|\mathcal{X}|$, while $q(\boldsymbol{x}) = c$ otherwise. This policy is optimal when there are only stationary attackers and $p(\boldsymbol{x}) \in \{0, 1\}$. We use this as the default.
2. Rank the instances in descending order of $p(\boldsymbol{x})$, and set $q(\boldsymbol{x}) = 1$ for the first $c|\mathcal{X}|$ of these (as long as $p(\boldsymbol{x})$ exceeds the threshold from Proposition 1). This policy is optimal when there are only stationary attackers, as we showed in Proposition 2. We call this "Naïve Ranking".

We used the **ifile** tool by Rennie [14] to select tokens for the feature vectors. Many of the desirable tokens for the TREC 2005 corpus are specific to the company where the emails were collected. Since our experiments evaluate performance on future TREC data which includes emails collected from different sources, we selected a subset of tokens that are environment invariant.

We compare the algorithms below using an empirical utility function, which we normalize to facilitate comparison across different cost settings (this utility is a generalization of accuracy that accounts for costs $V(\boldsymbol{x})$ and $G(\boldsymbol{x})$, which are fixed to V and G respectively):

$$\tilde{U}_{\mathcal{D}} = 1 - \frac{w|\mathcal{X}^+| + |\mathcal{X}^-|}{w|\mathcal{X}_{TN}| + |\mathcal{X}_{TP}|}, \tag{6}$$

where $|\mathcal{X}_{TN}|$ is the number of true negatives, $|\mathcal{X}_{TP}|$ the number of true positives, $|\mathcal{X}^-| = \sum_{\boldsymbol{x}} y(\boldsymbol{x})(1 - q(\boldsymbol{x}))$ the expected number of false negatives, $|\mathcal{X}^+| = \sum_{\boldsymbol{x}}(1 - y(\boldsymbol{x}))q(\boldsymbol{x})$ the expected number of false positives, and $w = \frac{G}{V}$ (note that when $w = 1$, this measure becomes exactly the total expected accuracy achieved by $q(\boldsymbol{x})$).

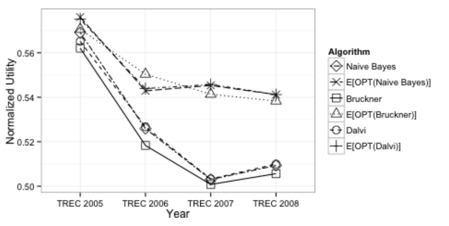
4.2 Performance on Public Spam Data

Our first set of Experiments is a direct comparison of the performance of our model as compared to state-of-the-art alternatives described above evaluated on public spam data. In this experiment, we use TREC 2005 data to train the classifiers, compute the optimal $q(\boldsymbol{x})$ for our approach while using the other alternatives as prescribed, and evaluate (by computing the expected normalized utility shown in Equation 6) on TREC data for years 2005-2008. As in all past evaluations of adversarial machine learning algorithms ([4–7]) we do not retrain the classifiers, since our intent is not merely to demonstrate value on spam data,

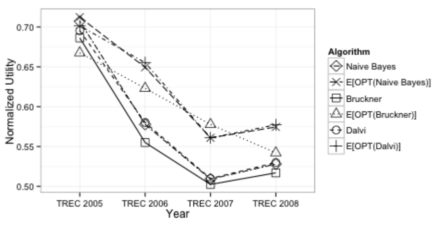
[4] We use the variant of Bruckner and Scheffer's classifier with the linear loss function.

but to anticipate far more actively adversarial environments in which attackers adapt to defense decisions quickly, and the defender wishes to have success in *anticipating adversarial response*.[5]

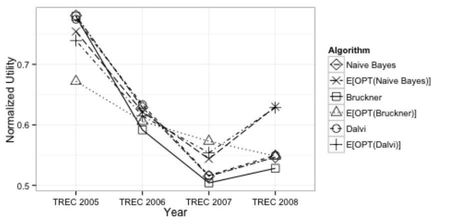
Our first set of results, shown in Figure 1, compares our optimization-based approach to alternatives when $V(\boldsymbol{x}) = G(\boldsymbol{x}) = 1$ for all $\boldsymbol{x}$ and $P_A = 0.5$ (this choice was made somewhat arbitrarily and not optimized to data), under a variety of budget constraints. Since our optimization can take as input an arbitrary $p(\boldsymbol{x})$, we compare the results of using the alternative machine learning approaches as input. From considering the four plots in Figure 1, each corresponding to a different budget constraint, it is apparent that the relative advantage of our approach (using any of the alternative $p(\boldsymbol{x})$ in the optimization problem) is pronounced (exhibiting 10-20% improvement over baseline) when the budget is relatively tight. Additionally, as we would intuitively expect, our approach performs better than alternatives as we move further into the future (giving the spammers more time to react to countermeasures from 2005). With a sufficiently generous budget constraint, it is also interesting to observe the tradeoff one would expect: the accuracy of our approaches is inferior to alternatives



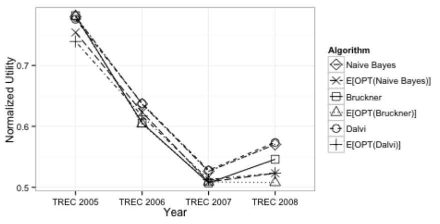
(a)



(b)



(c)



(d)

Fig. 1. Comparison of algorithms on TREC data, trained on year 2005, and tested on years 2005-2008. Our approach is labeled as $E[OPT(\cdot)]$, where the parameter is the classifier that serves as our $p(\boldsymbol{x})$. We use the following parameters: $\delta = 1$, $V(x) = G(x) = 1, P_A = 0.5$. (a) $c = 0.1$; (b) $c = 0.3$; (c) $c = 0.5$; (d) $c = 0.9$.

[5] In a separate set of experiments which we omit due to space constraints, we verified that even after retraining the classifiers each year, our approach typically outperforms the alternatives.

on training data, but the decisions are more robust to adversarial manipulation embedded in future data.

In Figure 2, we consider a higher cost of malware relative to benign instances, fixing $G(x) = 1$ and considering $V(x) = 2$ and 10. Perhaps the most surprising finding in these plots is that here Naïve Bayes outperforms Dalvi et al. and Bruckner and Scheffer in several instances, even though these are specifically tailored to adversarial situations. Our approaches, however, perform consistently better than the alternatives.

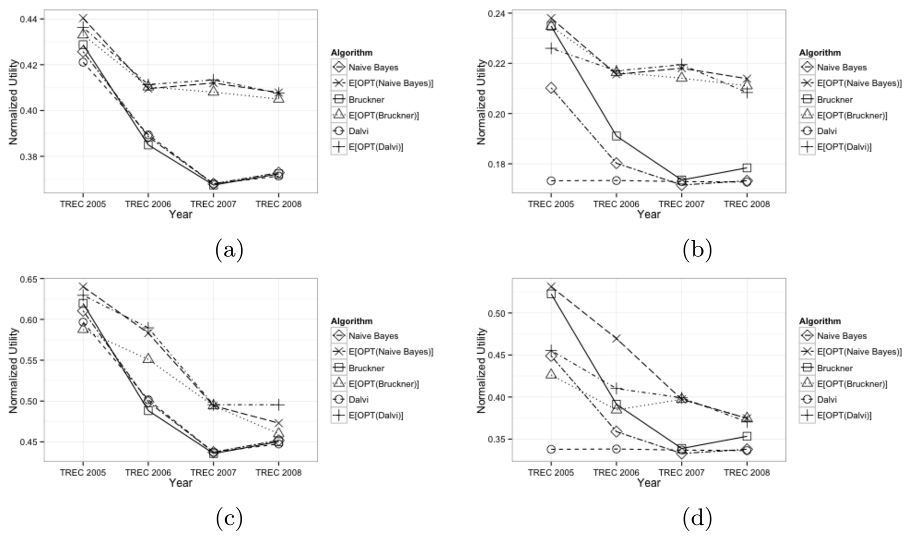


Fig. 2. Comparison of algorithms on TREC data, trained on year 2005, and tested on years 2005-2008. Our approach is labeled as $E[OPT(\cdot)]$, where the parameter is the classifier that serves as our $p(\boldsymbol{x})$. We fix $\delta = 1, G(x) = 1, P_A = 0.5$, and vary $V(x)$ and c. (a) $V(x) = 2$, $c = 0.1$; (b) $V(x) = 10$, $c = 0.1$; (c) $V(x) = 2$, $c = 0.3$; (d) $V(x) = 10$, $c = 0.3$.

We performed a statistical comparison between our approach and a corresponding classifier $p(\boldsymbol{x})$ on which it is based using Friedman's test with the post-hoc Bonferroni correction [12]. For all classifier pairs of the form $\{\mathcal{C}, E[OPT(\mathcal{C})]\}$ with $\mathcal{C} \in \{\textit{Naïve Bayes}, \textit{Bruckner}, \textit{Dalvi}\}$ and for $c \in \{0.1, 0.3\}$, $V(x) \in \{1, 2, 10\}$, our approach is statistically better than the alternative at the $\alpha = 0.05$ confidence level.

4.3 Performance with an Optimizing Attacker

Evaluating performance on future TREC data as done above is fundamentally limited since this data set represents spam, where adversaries generally do not target a specific classifier or organization but a relatively large population of spam filters. In contrast, our approach is tailored to highly sophisticated and targeted attacks. The problem is that data of this nature is highly sensitive and not publicly available. Indeed, the ideal, infeasible, experiment is to observe adversarial response to our model as well as other alternatives and evaluate the approaches with respect to such adversarial response. As the next best alternative which has become relatively standard [11, 7], we complement the spam evaluation with an alternative set of experiments aimed at modeling highly adaptive adversaries who maximize their expected utility in response to operational decisions $q(\boldsymbol{x})$. Specifically, we assume that a machine learning algorithm provides an accurate assessment of current or near-term threats, $p(\boldsymbol{x})$, and that all of the attackers are adaptive (i.e., that $P_A = 1$). Moreover, we assume that the learner/defender has correct knowledge of these parameters, as well as the parameter of the adaptive attacker's objective function, δ (we relax this assumption in the online appendix). Finally, we let $V(x) = G(x) = 1$ for all $\boldsymbol{x}$. For each year Y in the TREC data set (e.g., $Y = 2005$), we perform 10-fold cross-validation. However, rather than computing the utility directly using the test fold, we compute the expected utility, assuming the adaptive attacker per our model above. Equivalently, we can think of this as the following exercise: for each $\boldsymbol{x}$ in the test fold, we assign it a benign label with probability $1-p(\boldsymbol{x})$, assign a malicious label with probability $p(\boldsymbol{x})P_S$, and with probability $p(\boldsymbol{x})P_A$ generate a new malicious input $\boldsymbol{x}'$ that maximizes the attacker's expected utility given $q(\boldsymbol{x})$ computed by our algorithm.

In the first set of experiments, we choose $p(\boldsymbol{x})$ as generated by each alternative learning model that we consider (i.e., Naïve Bayes, Dalvi et al., and Bruckner and Scheffer). Figure 3 shows the results comparing the direct use of the three classifiers, and as a part of our optimization program, when $B = c|\mathcal{X}|$ with $c = 0.1$ and $c = 0.3$. This figure exhibits several findings. First, all three alternatives, including the two state-of-the-art approaches to adversarial classification, are exploitable by a sophisticated adversary. By comparison, all three of our optimization-based counterparts are more robust and beat their respective classifiers in paired comparisons. Second, the classifier of Dalvi et al. is in all cases far more robust to adversarial manipulation than the one derived from Bruckner and Scheffer. Finally, we did not display the results of using Naïve Ranking here, as it performs far worse; clearly, randomization is crucial when facing a sophisticated adversary.

In another set of experiments, we use Naïve Bayes as $p(\boldsymbol{x})$, and evaluate the quality of Dalvi et al., Bruckner and Scheffer, and our optimized approach (still using a synthetic attacker). Figure 4 shows the results. As in the previous set of experiments, our model outperforms all of the alternatives. Surprisingly, however, Dalvi et al. and Bruckner and Scheffer do not much improve upon the baseline Naïve Bayes in this setting, and in some cases are even slightly worse.

In our final set of experiments in this section, we consider the impact of varying $V(x)$. The results are shown in Figure 5. Again, our model consistently outperforms alternatives in paired comparisons, at times by a considerable margin (up to 50% improvement in some cases). In all experiments in this section, we

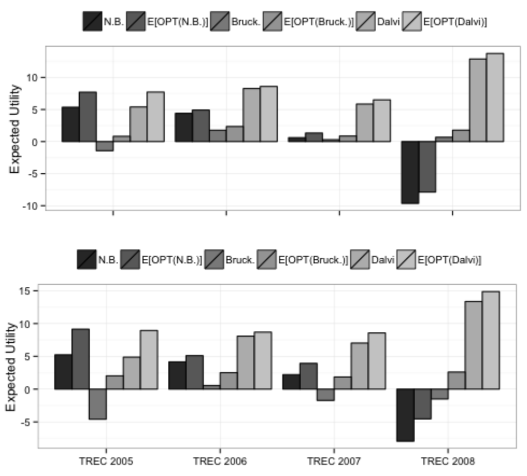


Fig. 3. The expected utilities, assuming $P_A = 1$ and that our attacker model is correct; top: $c = 0.1$; bottom: $c = 0.3$.

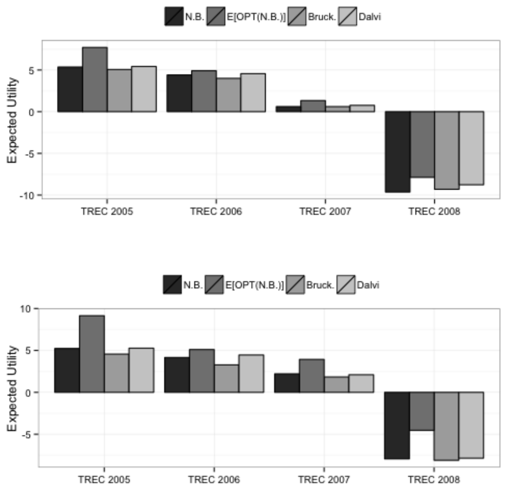


Fig. 4. The expected utilities, assuming $P_A = 1$ and that our attacker model is correct, where $p(x)$ is provided by Naïve Bayes; top: $c = 0.1$; bottom: $c = 0.3$.

verified that our approach is statistically better than alternatives at the $\alpha = 0.05$ confidence level.

A clear limitation of our evaluation above is that the comparison which simulates attacker behavior according to our modeling assumptions unduly favors our approach. In fact, we ran extensive experiments relaxing this assumption,

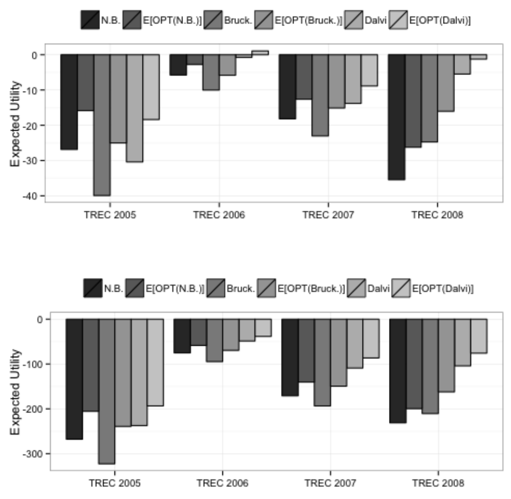


Fig. 5. The expected utilities, assuming $P_A = 1$ and that our attacker model is correct. Top: $V(x) = 2$; bottom: $V(x) = 10$. $c = 0.3$.

all of which show that our approach is quite robust to errors in the model specification. For lack of space, we deferred this discussion to the online appendix (`http://appendices.webs.com/amlrobust.pdf`).

5 Conclusion

We have presented a general approach for finding the optimal inspection policy against both static and adaptive adversaries. We showed that in the special case when an adversary is static and with no operational budget constraints, our model is equivalent to traditional likelihood ratio approaches (equivalently, using a threshold on the probability of malware/spam). Our experiments demonstrated that our model consistently outperforms both a baseline, non-adversarial machine learning approach, as well as several state-of-the-art adversarial classification alternatives. Overall, our approach demonstrates a clear advantage when inspection is costly, events have weighted importance, and when there are sophisticated, adaptive attackers. From a practical perspective, our approach is very simple, highly scalable (it involves solving a linear program), and can use an arbitrary classifier as input (indeed, a better classifier would improve the performance of our optimization method). Our model is, of course, a severe simplification of reality, and in future work one could consider attackers that strategically manipulate training data, and/or multi-stage games in which defender and attackers move in sequence. Despite the apparent simplicity of our model, however, we demonstrate that it outperforms alternatives on *actual* data and, thus, is a good starting point for future, more complex, modeling advances, which would need to demonstrate sufficient added value to compensate for additional complexity.

References

1. Globerson, A., Roweis, S.: Nightmare at test time: robust learning by feature deletion. In: ICML '06: Proceedings of the 23rd international conference on Machine learning, New York, NY, USA, ACM Press (2006) 353–360
2. Lowd, D., Meek, C.: Adversarial learning. In: ACM SIGKDD International Conference on Knowledge Discovery in Data Mining. (2005) 641–647
3. Huang, L., Joseph, A.D., Nelson, B., Rubinstein, B.I., Tygar, J.D.: Adversarial machine learning. In: Proceedings of the 4th ACM workshop on Security and artificial intelligence. AISec '11, New York, NY, USA, ACM (2011) 43–58
4. Brückner, M., Scheffer, T.: Nash equilibria of static prediction games. In: Advances in Neural Information Processing Systems. (2009) 171–179
5. Liu, W., Chawla, S.: Mining adversarial patterns via regularized loss minimization. Machine Learning **81** (2010) 69–83
6. Brückner, M., Scheffer, T.: Stackelberg games for adversarial prediction problems. In: Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining. KDD '11, New York, NY, USA, ACM (2011) 547–555
7. Colbaugh, R., Glass, K.: Predictive defense against evolving adversaries. In: IEEE International Conference on Intelligence and Security Informatics. (2012) 18–23
8. Sommer, R., Paxson, V.: Outside the closed world: On using machine learning for network intrusion detection. In: IEEE Symposium on Security and Privacy. (2010) 305–316
9. Paruchuri, P., Pearce, J., Marecki, J., Tambe, M.: Playing games for security: An efficient exact algorithm for solving Bayesian Stackelberg games. In: Seventh International Conference on Autonomous Agents and Multiagent Systems. (2008) 895–902
10. Filshtinskiy, S.: Cybercrime, cyberweapons, cyber wars: Is there too much of it in the air? Communications of the ACM **56**(6) (2013) 28–30
11. Dalvi, N., Domingos, P., Mausam, Sanghai, S., Verma, D.: Adversarial classification. In: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining. KDD '04, New York, NY, USA, ACM (2004) 99–108
12. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. **7** (December 2006) 1–30
13. Salzberg, S.L.: On comparing classifiers: Pitfalls to avoid and a recommended approach. Data Min. Knowl. Discov. **1**(3) (January 1997) 317–328
14. Rennie, J.D.M.: ifile: An application of machine learning to e-mail filtering. In: Proc. KDD Workshop on Text Mining. (2000)