Evasion attacks against machine learning at test time

Battista Biggio[1], Igino Corona[1], Davide Maiorca[1], Blaine Nelson[2], Nedim Šrndić[3], Pavel Laskov[3], Giorgio Giacinto[1], and Fabio Roli[1]

[1] Dept. of Electrical and Electronic Engineering, University of Cagliari, Piazza d'Armi, 09123 Cagliari, Italy
{battista.biggio, igino.corona, davide.maiorca, giacinto, roli}@diee.unica.it,
WWW home page: http://prag.diee.unica.it/
[2] Institut für Informatik, Universität Potsdam, August-Bebel-Straße 89, 14482 Potsdam, Germany
bnelson@cs.uni-potsdam.de
[3] Wilhelm Schickard Institute for Computer Science, University of Tübingen, Sand 1, 72076 Tübingen, Germany
{nedim.srndic, pavel.laskov}@uni-tuebingen.de

Abstract. In security-sensitive applications, the success of machine learning depends on a thorough vetting of their resistance to adversarial data. In one pertinent, well-motivated attack scenario, an adversary may attempt to evade a deployed system at test time by carefully manipulating attack samples. In this work, we present a simple but effective gradient-based approach that can be exploited to systematically assess the security of several, widely-used classification algorithms against evasion attacks. Following a recently proposed framework for security evaluation, we simulate attack scenarios that exhibit different risk levels for the classifier by increasing the attacker's knowledge of the system and her ability to manipulate attack samples. This gives the classifier designer a better picture of the classifier performance under evasion attacks, and allows him to perform a more informed model selection (or parameter setting). We evaluate our approach on the relevant security task of malware detection in PDF files, and show that such systems can be easily evaded. We also sketch some countermeasures suggested by our analysis.

Keywords: adversarial machine learning, evasion attacks, support vector machines, neural networks

1 Introduction

Machine learning is being increasingly used in security-sensitive applications such as spam filtering, malware detection, and network intrusion detection [3, 5, 9, 11, 14–16, 19, 21]. Due to their intrinsic adversarial nature, these applications differ from the classical machine learning setting in which the underlying data distribution is assumed to be *stationary*. To the contrary, in security-sensitive

applications, samples (and, thus, their distribution) can be actively manipulated by an intelligent, adaptive adversary to confound learning; *e.g.*, to avoid detection, spam emails are often modified by obfuscating common spam words or inserting words associated with legitimate emails [3, 9, 16, 19]. This has led to an arms race between the designers of learning systems and their adversaries, which is evidenced by the increasing complexity of modern attacks and countermeasures. For these reasons, classical performance evaluation techniques are not suitable to reliably assess the security of learning algorithms, *i.e.*, the performance degradation caused by carefully crafted attacks [5].

To better understand the security properties of machine learning systems in adversarial settings, paradigms from security engineering and cryptography have been adapted to the machine learning field [2, 5, 14]. Following common security protocols, the learning system designer should use *proactive* protection mechanisms that anticipate and prevent the adversarial impact. This requires (*i*) finding potential vulnerabilities of learning *before* they are exploited by the adversary; (*ii*) investigating the impact of the corresponding attacks (*i.e.*, evaluating classifier security); and (*iii*) devising appropriate countermeasures if an attack is found to significantly degrade the classifier's performance.

Two approaches have previously addressed security issues in learning. The min-max approach assumes the learner and attacker's loss functions are antagonistic, which yields relatively simple optimization problems [10, 12]. A more general game-theoretic approach applies for non-antagonistic losses; *e.g.*, a spam filter wants to accurately identify legitimate email while a spammer seeks to boost his spam's appeal. Under certain conditions, such problems can be solved using a Nash equilibrium approach [7, 8]. Both approaches provide a *secure* counterpart to their respective learning problems; *i.e.*, an optimal anticipatory classifier.

Realistic constraints, however, are too complex and multi-faceted to be incorporated into existing game-theoretic approaches. Instead, we investigate the vulnerabilities of classification algorithms by deriving *evasion attacks* in which the adversary aims to avoid detection by manipulating malicious test samples.[4] We systematically assess classifier security in attack scenarios that exhibit increasing risk levels, simulated by increasing the attacker's knowledge of the system and her ability to manipulate attack samples. Our analysis allows a classifier designer to understand how the classification performance of each considered model degrades under attack, and thus, to make more informed design choices.

The problem of evasion at test time was addressed in prior work, but limited to linear and convex-inducing classifiers [9, 19, 22]. In contrast, the methods presented in Sections 2 and 3 can generally evade linear or non-linear classifiers using a gradient-descent approach inspired by Golland's discriminative directions technique [13]. Although we focus our analysis on widely-used classifiers such as Support Vector Machines (SVMs) and neural networks, our approach is applicable to any classifier with a differentiable discriminant function.

[4] Note that other kinds of attacks are possible, *e.g.*, if the adversary can manipulate the training data. A comprehensive taxonomy of attacks can be found in [2, 14].

This paper is organized as follows. We present the evasion problem in Section 2 and our gradient-descent approach in Section 3. In Section 4 we first visually demonstrate our attack on the task of handwritten digit recognition, and then show its effectiveness on a realistic application related to the detection of PDF malware. Finally in Section 5, we summarize our contributions, discuss possibilities for improving security, and suggest future extensions of this work.

2 Optimal evasion at test time

We consider a classification algorithm $f : \mathcal{X} \mapsto \mathcal{Y}$ that assigns samples represented in some feature space $\mathbf{x} \in \mathcal{X}$ to a label in the set of predefined classes $y \in \mathcal{Y} = \{-1, +1\}$, where -1 $(+1)$ represents the legitimate (malicious) class. The classifier f is trained on a dataset $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^n$ sampled from an underlying distribution $p(\mathbf{X}, Y)$. The label $y^c = f(\mathbf{x})$ given by a classifier is typically obtained by thresholding a continuous discriminant function $g : \mathcal{X} \mapsto \mathbb{R}$. In the sequel, we use y^c to refer to the label assigned by the classifier as opposed to the true label y. We further assume that $f(\mathbf{x}) = -1$ if $g(\mathbf{x}) < 0$, and $+1$ otherwise.

2.1 Adversary model

To motivate the optimal attack strategy for evasion, it is necessary to disclose one's assumptions of the adversary's knowledge and ability to manipulate the data. To this end, we exploit a general model of the adversary that elucidates specific assumptions about adversary's goal, knowledge of the system, and capability to modify the underlying data distribution. The considered model is part of a more general framework investigated in our recent work [5], which subsumes evasion and other attack scenarios. This model can incorporate application-specific constraints in the definition of the adversary's capability, and can thus be exploited to derive practical guidelines for developing the optimal attack strategy.

Adversary's goal. As suggested by Laskov and Kloft [17], the adversary's goal should be defined in terms of a utility (loss) function that the adversary seeks to maximize (minimize). In the evasion setting, the attacker's goal is to manipulate a single (without loss of generality, positive) sample that should be misclassified. Strictly speaking, it would suffice to find a sample $\mathbf{x}$ such that $g(\mathbf{x}) < -\epsilon$ for any $\epsilon > 0$; *i.e.*, the attack sample only just crosses the decision boundary.[5] Such attacks, however, are easily thwarted by slightly adjusting the decision threshold. A better strategy for an attacker would thus be to create a sample that is misclassified with high confidence; *i.e.*, a sample minimizing the value of the classifier's discriminant function, $g(\mathbf{x})$, subject to some feasibility constraints.

Adversary's knowledge. The adversary's knowledge about her targeted learning system may vary significantly. Such knowledge may include:

- the training set or part of it;

[5] This is also the setting adopted in previous work [9, 19, 22].

– the feature representation of each sample; *i.e.*, how *real* objects such as emails, network packets are mapped into the classifier's feature space;
– the type of a learning algorithm and the form of its decision function;
– the (trained) classifier model; *e.g.*, weights of a linear classifier;
– or feedback from the classifier; *e.g.*, classifier labels for samples chosen by the adversary.

Adversary's capability. In the evasion scenario, the adversary's capability is limited to modifications of test data; *i.e.*altering the training data is not allowed. However, under this restriction, variations in attacker's power may include:

– modifications to the input data (limited or unlimited);
– modifications to the feature vectors (limited or unlimited);
– or independent modifications to specific features (the semantics of the input data may dictate that certain features are interdependent).

Most of the previous work on evasion attacks assumes that the attacker can arbitrarily change every feature [8, 10, 12], but they constrain the degree of manipulation, *e.g.*, limiting the number of modifications, or their total cost. However, many real domains impose stricter restrictions. For example, in the task of PDF malware detection [20, 24, 25], removal of content is not feasible, and content addition may cause correlated changes in the feature vectors.

2.2 Attack scenarios

In the sequel, we consider two attack scenarios characterized by different levels of adversary's knowledge of the attacked system discussed below.

Perfect knowledge (PK). In this setting, we assume that the adversary's goal is to minimize $g(\mathbf{x})$, and that she has perfect knowledge of the targeted classifier; *i.e.*, the adversary knows the feature space, the type of the classifier, and the trained model. The adversary can transform attack points in the test data but must remain within a maximum distance of $d_{\max}$ from the original attack sample. We use $d_{\max}$ as parameter in our evaluation to simulate increasingly pessimistic attack scenarios by giving the adversary greater freedom to alter the data.

The choice of a suitable distance measure $d : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}^+$ is application specific [9, 19, 22]. Such a distance measure should reflect the adversary's effort required to manipulate samples or the cost of these manipulations. For example, in spam filtering, the attacker may be bounded by a certain number of words she can manipulate, so as not to lose the semantics of the spam message.

Limited knowledge (LK). Here, we again assume that the adversary aims to minimize the discriminant function $g(\mathbf{x})$ under the same constraint that each transformed attack point must remain within a maximum distance of $d_{\max}$ from the corresponding original attack sample. We further assume that the attacker knows the feature representation and the type of the classifier, but does not know either the learned classifier f or its training data $\mathcal{D}$, and hence can not directly

compute $g(\mathbf{x})$. However, we assume that she can collect a surrogate dataset $\mathcal{D}' = \{\hat{\mathbf{x}}_i, \hat{y}_i\}_{i=1}^{n_q}$ of n_q samples drawn from the same underlying distribution $p(\mathbf{X}, Y)$ from which $\mathcal{D}$ was drawn. This data may be collected by an adversary in several ways; *e.g.*, by sniffing some network traffic during the classifier operation, or by collecting legitimate and spam emails from an alternate source.

Under this scenario, the adversary proceeds by approximating the discriminant function $g(\mathbf{x})$ as $\hat{g}(\mathbf{x})$, where $\hat{g}(\mathbf{x})$ is the discriminant function of a surrogate classifier $\hat{f}$ learnt on $\mathcal{D}'$. The amount of the surrogate data, n_q, is an attack parameter in our experiments. Since the adversary wants her surrogate $\hat{f}$ to closely approximate the targeted classifier f, it stands to reason that she should learn $\hat{f}$ using the labels assigned by the targeted classifier f, when such feedback is available. In this case, instead of using the true class labels $\hat{y}_i$ to train $\hat{f}$, the adversary can query f with the samples of $\mathcal{D}'$ and subsequently learn using the labels $\hat{y}_i^c = f(\hat{\mathbf{x}}_i)$ for each $\mathbf{x}_i$.

2.3 Attack strategy

Under the above assumptions, for any target malicious sample $\mathbf{x}^0$ (the adversary's desired instance), an optimal attack strategy finds a sample $\mathbf{x}^*$ to minimize $g(\cdot)$ or its estimate $\hat{g}(\cdot)$, subject to a bound on its distance[6] from $\mathbf{x}^0$:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} \quad \hat{g}(\mathbf{x}) \qquad (1)$$
$$\text{s.t.} \quad d(\mathbf{x}, \mathbf{x}^0) \leq d_{\max}.$$

Generally, this is a non-linear optimization problem. One may approach it with many well-known techniques, like gradient descent, or quadratic techniques such as Newton's method, BFGS, or L-BFGS. We choose a gradient-descent procedure. However, $\hat{g}(\mathbf{x})$ may be non-convex and descent approaches may not achieve a global optima. Instead, the descent path may lead to a flat region (local minimum) outside of the samples' support (*i.e.*, where $p(\mathbf{x}) \approx 0$) where the attack sample may or may not evade depending on the behavior of g in this unsupported region (see left and middle plots in Figure 1).

Locally optimizing $\hat{g}(\mathbf{x})$ with gradient descent is particularly susceptible to failure due to the nature of a discriminant function. Besides its shape, for many classifiers, $g(\mathbf{x})$ is equivalent to a posterior estimate $p(y^c = -1|\mathbf{x})$; *e.g.*, for neural networks, and SVMs [23]. The discriminant function does not incorporate the evidence we have about the data distribution, $p(\mathbf{x})$, and thus, using gradient descent to optimize Eq. 1 may lead into unsupported regions ($p(\mathbf{x}) \approx 0$). Because of the insignificance of these regions, the value of g is relatively unconstrained by criteria such as risk minimization. This problem is compounded by our finite (and possibly small) training set, since it provides little evidence in these regions

[6] One can also incorporate additional application-specific constraints on the attack samples. For instance, the box constraint $0 \leq x_f \leq 1$ can be imposed if the f^{th} feature is normalized in $[0, 1]$, or $x_f^0 \leq x_f$ can be used if the f^{th} feature of the target $\mathbf{x}^0$ can be only incremented.

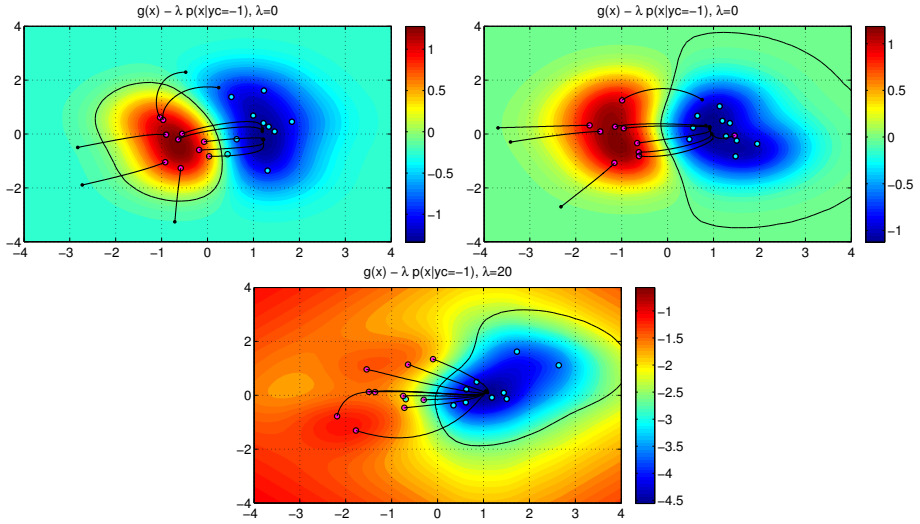

Fig. 1. Different scenarios for gradient-descent-based evasion procedures. In each, the function $g(\mathbf{x})$ of the learned classifier is plotted with a color map with high values (red-orange-yellow) for the malicious class, and low values (green-cyan-blue) for the legitimate class. The decision boundary is shown in black. For every malicious sample, we plot the gradient descent path against a classifier with a closed boundary around the malicious class (**top-left**) and against a classifier with a closed boundary around the benign class (**top-right**). Finally, we plot the modified objective function of Eq. (2) and the resulting descent paths against a classifier with a closed boundary around the benign class (**bottom**).

to constrain the shape of g. Thus, when our gradient descent procedure produces an evasion example in these regions, the attacker cannot be confident that this sample will actually evade the corresponding classifier. Therefore, to increase the probability of successful evasion, the attacker should favor attack points from densely populated regions of legitimate points, where the estimate $\hat{g}(\mathbf{x})$ is more reliable (closer to the real $g(\mathbf{x})$), and tends to become negative in value.

To overcome this shortcoming, we introduce an additional component into our attack objective, which estimates $p(\mathbf{x}|y^c = -1)$ using a density estimator. This term acts as a penalizer for $\mathbf{x}$ in low density regions and is weighted by a parameter $\lambda \geq 0$ yielding the following modified optimization problem:

$$\arg\min_{x} \; F(\mathbf{x}) = \hat{g}(\mathbf{x}) - \frac{\lambda}{n} \sum_{i|y_i^c=-1} k\left(\tfrac{\mathbf{x}-\mathbf{x}_i}{h}\right) \tag{2}$$

$$\text{s.t.} \;\; d(\mathbf{x}, \mathbf{x}^0) \leq d_{\max} \;\; , \tag{3}$$

where h is a bandwidth parameter for a kernel density estimator (KDE), and n is the number of benign samples ($y^c = -1$) available to the adversary. This

Algorithm 1 Gradient-descent evasion attack

Input: $\mathbf{x}^0$, the initial attack point; t, the step size; λ, the trade-off parameter; $\epsilon > 0$ a small constant.
Output: $\mathbf{x}^*$, the final attack point.

$m \leftarrow 0$.
repeat
  $m \leftarrow m + 1$
  Set $\nabla F(\mathbf{x}^{m-1})$ to a unit vector aligned with $\nabla g(\mathbf{x}^{m-1}) - \lambda \nabla p(\mathbf{x}^{m-1} | y^c = -1)$.
  $\mathbf{x}^m \leftarrow \mathbf{x}^{m-1} - t \nabla F(\mathbf{x}^{m-1})$
  if $d(\mathbf{x}^m, \mathbf{x}^0) > d_{\max}$ **then**
    Project $\mathbf{x}^m$ onto the boundary of the feasible region.
  end if
until $F(\mathbf{x}^m) - F\left(\mathbf{x}^{m-1}\right) < \epsilon$
return: $\mathbf{x}^* = \mathbf{x}^m$

alternate objective trades off between minimizing $\hat{g}(\mathbf{x})$ (or $p(y^c = -1|\mathbf{x})$) and maximizing the estimated density $p(\mathbf{x}|y^c = -1)$. The extra component favors attack points that imitate features of known legitimate samples. In doing so, it reshapes the objective function and thereby biases the resulting gradient descent towards regions where the negative class is concentrated (see the bottom plot in Fig. 1). This produces a similar effect to that shown by *mimicry* attacks in network intrusion detection [11].[7] For this reason, although our setting is rather different, in the sequel we refer to this extra term as the *mimicry* component.

Finally, we point out that, when mimicry is used ($\lambda > 0$), our gradient descent clearly follows a suboptimal path compared to the case when only $g(\mathbf{x})$ is minimized ($\lambda = 0$). Therefore, more modifications may be required to reach the same value of $g(\mathbf{x})$ attained when $\lambda = 0$. However, as previously discussed, when $\lambda = 0$, our descent approach may terminate at a local minimum where $g(\mathbf{x}) > 0$, without successfully evading detection. This behavior can thus be qualitatively regarded as a trade-off between the probability of evading the targeted classifier and the number of times that the adversary must modify her samples.

3 Gradient descent attacks

Algorithm 1 solves the optimization problem in Eq. 2 via gradient descent. We assume $g(\mathbf{x})$ to be differentiable almost everywhere (subgradients may be used at discontinuities). However, note that if g is non-differentiable or insufficiently smooth, one may still use the mimicry / KDE term of Eq. (2) as a search heuristic. This investigation is left to future work.

[7] Mimicry attacks [11] consist of camouflaging malicious network packets to evade anomaly-based intrusion detection systems by mimicking the characteristics of the legitimate traffic distribution.

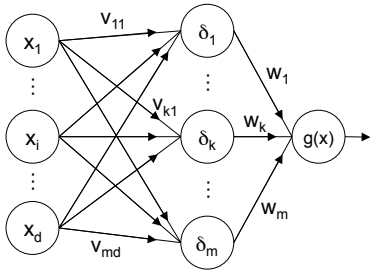

Fig. 2. The architecture of a multi-layer perceptron with a single hidden layer.

3.1 Gradients of discriminant functions

Linear classifiers. Linear discriminant functions are $g(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b$ where $\mathbf{w} \in \mathbb{R}^d$ is the feature weights and $b \in \mathbb{R}$ is the bias. Its gradient is $\nabla g(\mathbf{x}) = \mathbf{w}$.

Support vector machines. For SVMs, $g(\mathbf{x}) = \sum_i \alpha_i y_i k(\mathbf{x}, \mathbf{x}_i) + b$. The gradient is thus $\nabla g(\mathbf{x}) = \sum_i \alpha_i y_i \nabla k(\mathbf{x}, \mathbf{x}_i)$. In this case, the feasibility of our approach depends on whether the kernel gradient $\nabla k(\mathbf{x}, \mathbf{x}_i)$ is computable as it is for many numeric kernels. For instance, the gradient of the RBF kernel, $k(\mathbf{x}, \mathbf{x}_i) = \exp\{-\gamma \|\mathbf{x} - \mathbf{x}_i\|^2\}$, is $\nabla k(\mathbf{x}, \mathbf{x}_i) = -2\gamma \exp\{-\gamma \|\mathbf{x} - \mathbf{x}_i\|^2\}(\mathbf{x} - \mathbf{x}_i)$, and for the polynomial kernel, $k(\mathbf{x}, \mathbf{x}_i) = (\langle \mathbf{x}, \mathbf{x}_i \rangle + c)^p$, it is $\nabla k(\mathbf{x}, \mathbf{x}_i) = p(\langle \mathbf{x}, \mathbf{x}_i \rangle + c)^{p-1} \mathbf{x}_i$.

Neural networks. For a multi-layer perceptron with a single hidden layer of m neurons and a sigmoidal activation function, we decompose its discriminant function g as follows (see Fig. 2): $g(\mathbf{x}) = (1+e^{-h(\mathbf{x})})^{-1}$, $h(\mathbf{x}) = \sum_{k=1}^m w_k \delta_k(\mathbf{x}) + b$, $\delta_k(\mathbf{x}) = (1 + e^{-h_k(\mathbf{x})})^{-1}$, $h_k(\mathbf{x}) = \sum_{j=1}^d v_{kj} x_j + b_k$. From the chain rule, the i^{th} component of $\nabla g(\mathbf{x})$ is thus given by:

$$\frac{\partial g}{\partial x_i} = \frac{\partial g}{\partial h} \sum_{k=1}^m \frac{\partial h}{\partial \delta_k} \frac{\partial \delta_k}{\partial h_k} \frac{\partial h_k}{\partial x_i} = g(\mathbf{x})(1 - g(\mathbf{x})) \sum_{k=1}^m w_k \delta_k(\mathbf{x})(1 - \delta_k(\mathbf{x})) v_{ki} \ .$$

3.2 Gradients of kernel density estimators

Similarly to SVMs, the gradient of kernel density estimators depends on the kernel gradient. We consider generalized RBF kernels of the form $k\left(\frac{\mathbf{x}-\mathbf{x}_i}{h}\right) = \exp\left(-\frac{d(\mathbf{x},\mathbf{x}_i)}{h}\right)$, where $d(\cdot,\cdot)$ is any suitable distance function. Here we use the same distance $d(\cdot,\cdot)$ defined in Eq. (3), but, in general, they can be different. For ℓ_2- and ℓ_1-norms (*i.e.*, RBF and Laplacian kernels), the KDE (sub)gradients are respectively given by:

$$-\frac{2}{nh} \sum_{i|y_i^c=-1} \exp\left(-\frac{\|\mathbf{x}-\mathbf{x}_i\|_2^2}{h}\right)(\mathbf{x} - \mathbf{x}_i) \ ,$$
$$-\frac{1}{nh} \sum_{i|y_i^c=-1} \exp\left(-\frac{\|\mathbf{x}-\mathbf{x}_i\|_1}{h}\right)(\mathbf{x} - \mathbf{x}_i) \ .$$

Note that the scaling factor here is proportional to $O(\frac{1}{nh})$. Therefore, to influence gradient descent with a significant mimicking effect, the value of λ in the objective function should be chosen such that the value of $\frac{\lambda}{nh}$ is comparable with (or higher than) the range of values of the discriminant function $\hat{g}(\mathbf{x})$.

3.3 Descent in discrete spaces

In discrete spaces, gradient approaches travel through infeasible portions of the feature space. In such cases, we need to find a feasible neighbor $\mathbf{x}$ that maximally decrease $F(\mathbf{x})$. A simple approach to this problem is to probe F at every point in a small neighborhood of $\mathbf{x}$, which would however require a large number of queries. For classifiers with a differentiable decision function, we can instead select the neighbor whose change best aligns with $\nabla F(\mathbf{x})$ and decreases the objective function; *i.e.*, to prevent overshooting a minimum.

4 Experiments

In this section, we first report a toy example from the MNIST handwritten digit classification task [18] to visually demonstrate how the proposed algorithm modifies digits to mislead classification. We then show the effectiveness of the proposed attack on a more realistic and practical scenario: the detection of malware in PDF files.

4.1 A toy example on handwritten digits

Similar to Globerson and Roweis [12], we consider discriminating between two distinct digits from the MNIST dataset [18]. Each digit example is represented as a gray-scale image of 28×28 pixels arranged in raster-scan-order to give feature vectors of $d = 28 \times 28 = 784$ values. We normalized each feature (pixel) $\mathbf{x} \in [0, 1]^d$ by dividing its value by 255, and we constrained the attack samples to this range. Accordingly, we optimized Eq. (2) subject to $0 \leq x_f \leq 1$ for all f.

We only consider the perfect knowledge (PK) attack scenario. We used the *Manhattan* distance (ℓ_1-norm), d, both for the kernel density estimator (*i.e.*, a Laplacian kernel) and for the constraint $d(\mathbf{x}, \mathbf{x}^0) \leq d_{\max}$ in Eq. (3), which bounds the total difference between the gray level values of the original image $\mathbf{x}^0$ and the attack image $\mathbf{x}$. We used $d_{\max} = \frac{5000}{255}$ to limit the total gray-level change to 5000. At each iteration, we increased the ℓ_1-norm value of $\mathbf{x} - \mathbf{x}^0$ by $\frac{10}{255}$, or equivalently, we changed the total gray level by 10. This is effectively the gradient step size. The targeted classifier was an SVM with the linear kernel and $C = 1$. We randomly chose 100 training samples and applied the attacks to a correctly-classified positive sample.

In Fig. 3 we illustrate gradient attacks in which a "3" is to be misclassified as a "7". The left image shows the initial attack point, the middle image shows the first attack image misclassified as legitimate, and the right image shows the attack point after 500 iterations. When $\lambda = 0$, the attack images exhibit only

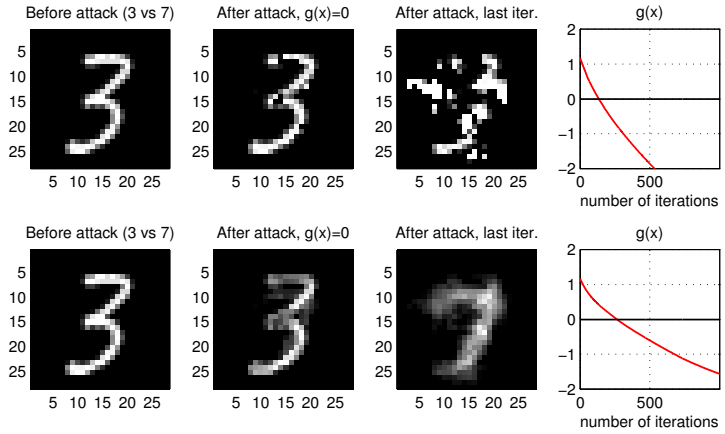

Fig. 3. Illustration of the gradient attack on the digit data, for $\lambda = 0$ (**top row**) and $\lambda = 10$ (**bottom row**). Without a mimicry component ($\lambda = 0$) gradient descent quickly decreases g but the resulting attack image does not resemble a "7". In contrast, the attack minimizes g slower when mimicry is applied ($\lambda = 0$) but the final attack image closely resembles a mixture between "3" and "7", as the term "mimicry" suggests.

a weak resemblance to the target class "7" but are, nevertheless, reliably misclassified. This is the same effect demonstrated in the top-left plot of Fig. 1: the classifier is evaded by making the attack sample sufficiently dissimilar from the malicious class. Conversely, when $\lambda = 10$, the attack images strongly resemble the target class because the mimicry term favors samples that are more similar to the target class. This is the same effect seen in the bottom plot of Fig. 1.

Finally note that, as expected, $g(\mathbf{x})$ tends to decrease more gracefully when mimicry is used, as we follow a suboptimal descent path. Since the targeted classifier can be easily evaded when $\lambda = 0$, exploiting the mimicry component would not be the optimal choice in this case. However, in the case of limited knowledge, as discussed at the end of Section 2.3, mimicry may allow us to trade for a higher probability of evading the targeted classifier, at the expense of a higher number of modifications.

4.2 Malware detection in PDF files

We now focus on the task of discriminating between legitimate and malicious PDF files, a popular medium for disseminating malware [26]. PDF files are excellent vectors for malicious-code, due to their flexible *logical structure*, which can described by a hierarchy of interconnected objects. As a result, an attack can be easily hidden in a PDF to circumvent file-type filtering. The PDF format further allows a wide variety of resources to be embedded in the document including `JavaScript`, `Flash`, and even binary programs. The type of the embedded object is specified by *keywords*, and its content is in a *data stream*. Several recent

works proposed machine-learning techniques for detecting malicious PDFs using the file's logical structure to accurately identify the malware [20, 24, 25]. In this case study, we use the feature representation of Maiorca *et al.* [20] in which each feature corresponds to the tally of occurrences of a given keyword.

The PDF structure imposes natural constraints on attacks. Although it is difficult to *remove* an embedded object (and its keywords) from a PDF without corrupting the PDF's file structure, it is rather easy to *insert* new objects (and, thus, keywords) through the addition of a new *version* to the PDF file [1]. In our feature representation, this is equivalent to allowing only feature increments, *i.e.*, requiring $\mathbf{x}^0 \leq \mathbf{x}$ as an additional constraint in the optimization problem given by Eq. (2). Further, the total difference in keyword counts between two samples is their *Manhattan* distance, which we again use for the kernel density estimator and the constraint in Eq. (3). Accordingly, $d_{\max}$ is the maximum number of additional keywords that an attacker can add to the original $\mathbf{x}^0$.

Experimental setup. For experiments, we used a PDF corpus with 500 malicious samples from the *Contagio* dataset[8] and 500 benign samples collected from the web. We randomly split the data into five pairs of training and testing sets with 500 samples each to average the final results. The features (keywords) were extracted from each training set as described in [20]. On average, 100 keywords were found in each run. Further, we also bounded the maximum value of each feature to 100, as this value was found to be close to the 95^{th} percentile for each feature. This limited the influence of outlying samples.

We simulated the *perfect* knowledge (PK) and the *limited* knowledge (LK) scenarios described in Section 2.1. In the LK case, we set the number of samples used to learn the surrogate classifier to $n_g = 100$. The reason is to demonstrate that even with a dataset as small as the 20% of the original training set size, the adversary may be able to evade the targeted classifier with high reliability. Further, we assumed that the adversary uses feedback from the *targeted* classifier f; *i.e.*, the labels $\hat{y}_i^c = f(\hat{\mathbf{x}}_i)$ for each surrogate sample $\hat{\mathbf{x}}_i \in \mathcal{D}'$.[9]

As discussed in Section 3.2, the value of λ is chosen according to the scale of the discriminant function $g(\mathbf{x})$, the bandwidth parameter h of the kernel density estimator, and the number of legitimate samples n in the surrogate training set. For computational reasons, to estimate the value of the KDE at $\mathbf{x}$, we only consider the 50 nearest (legitimate) training samples to $\mathbf{x}$; therefore, $n \leq 50$ in our case. The bandwidth parameter was set to $h = 10$, as this value provided a proper rescaling of the Manhattan distances observed in our dataset for the KDE. We thus set $\lambda = 500$ to be comparable with $O(nh)$.

For each targeted classifier and training/testing pair, we learned five surrogate classifiers by randomly selecting n_g samples from the test set, and we averaged their results. For SVMs, we sought a surrogate classifier that would correctly match the labels from the targeted classifier; thus, we used parameters $C = 100$, and $\gamma = 0.1$ (for the RBF kernel) to heavily penalize training errors.

[8] http://contagiodump.blogspot.it

[9] Similar results were also obtained using the true labels (without relabeling), since the targeted classifiers correctly classified almost all samples in the test set.

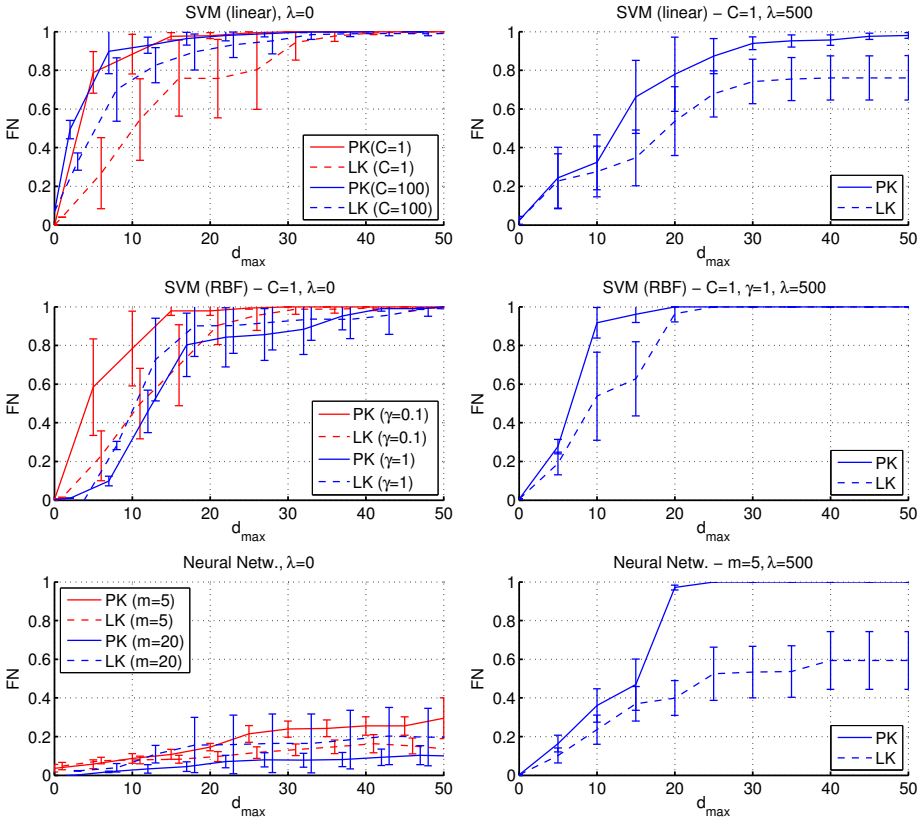

Fig. 4. Experimental results for SVMs with linear and RBF kernel (first and second row), and for neural networks (third row). We report the FN values (attained at FP=0.5%) for increasing $d_{\max}$. For the sake of readability, we report the average FN value $\pm$ half standard deviation (shown with error bars). Results for perfect (PK) and limited (LK) knowledge attacks with $\lambda = 0$ (without mimicry) are shown in the first column, while results with $\lambda = 500$ (with mimicry) are shown in the second column. In each plot we considered different values of the classifier parameters, *i.e.*, the regularization parameter C for the linear SVM, the kernel parameter γ for the SVM with RBF kernel, and the number of neurons m in the hidden layer for the neural network, as reported in the plot title and legend.

Experimental results. We report our results in Figure 4, in terms of the false negative (FN) rate attained by the targeted classifiers as a function of the maximum allowable number of modifications, $d_{\max} \in [0, 50]$. We compute the FN rate corresponding to a fixed false positive (FP) rate of FP= 0.5%. For $d_{\max} = 0$, the FN rate corresponds to a standard performance evaluation using unmodified PDFs. As expected, the FN rate increases with $d_{\max}$ as the PDF is increasingly modified. Accordingly, a more secure classifier will exhibit a more graceful increase of the FN rate.

Results for $\lambda = 0$. We first investigate the effect of the proposed attack in the PK case, without considering the mimicry component (Figure 4, first column), for varying parameters of the considered classifiers. The linear SVM (Figure 4, top-left plot) is almost always evaded with as few as 5 to 10 modifications, independent of the regularization parameter C. It is worth noting that attacking a linear classifier amounts to always incrementing the value of the same highest-weighted feature (corresponding to the `/Linearized` keyword in the majority of the cases) until it reaches its upper bound. This continues with the next highest weighted non-bounded feature until termination. This occurs simply because the gradient of $g(\mathbf{x})$ does not depend on $\mathbf{x}$ for a linear classifier (see Section 3.1). With the RBF kernel (Figure 4, middle-left plot), SVMs exhibit a similar behavior with $C = 1$ and various values of its γ parameter,[10] and the RBF SVM provides a higher degree of security compared to linear SVMs (*cf.* top-left plot and middle-left plot in Figure 4). Interestingly, compared to SVMs, neural networks (Figure 4, bottom-left plot) seem to be much more robust against the proposed evasion attack. This behavior can be explained by observing that the decision function of neural networks may be characterized by flat regions (*i.e.*, regions where the gradient of $g(\mathbf{x})$ is close to zero). Hence, the gradient descent algorithm based solely on $g(\mathbf{x})$ essentially stops after few attack iterations for most of the malicious samples, without being able to find a suitable attack.

In the LK case, without mimicry, classifiers are evaded with a probability only *slightly* lower than that found in the PK case, even when only $n_g = 100$ surrogate samples are used to learn the surrogate classifier. This aspect highlights the threat posed by a skilled adversary with incomplete knowledge: only a small set of samples may be required to successfully attack the target classifier using the proposed algorithm.

Results for $\lambda = 500$. When mimicry is used (Figure 4, second column), the success of the evasion of linear SVMs (with $C = 1$) decreases both in the PK (*e.g.*, compare the blue curve in the top-left plot with the solid blue curve in the top-right plot) and LK case (*e.g.*, compare the dashed red curve in the top-left plot with the dashed blue curve in the top-right plot). The reason is that the computed direction tends to lead to a slower descent; *i.e.*, a less direct path that often requires more modifications to evade the classifier. In the non-linear case (Figure 4, middle-right and bottom-right plot), instead, mimicking exhibits some beneficial aspects for the attacker, although the constraint on feature addition

[10] We also conducted experiments using $C = 0.1$ and $C = 100$, but did not find significant differences compared to the presented results using $C = 1$.

may make it difficult to properly mimic legitimate samples. In particular, note how the targeted SVMs with RBF kernel (with $C = 1$ and $\gamma = 1$) in the PK case (*e.g.*, compare the solid blue curve in the middle-left plot with the solid blue curve in the middle-right plot) is evaded with a significantly higher probability than in the case of $\lambda = 0$. The reason is that, as explained at the end of Section 2.3, a pure descent strategy on $g(\mathbf{x})$ may find local minima (*i.e.*, attack samples) that do not evade detection, while the mimicry component biases the descent towards regions of the feature space more densely populated by legitimate samples, where $g(\mathbf{x})$ eventually attains lower values. For neural networks, this aspect is even more evident, in both the PK and LK settings (compare the dashed/solid curves in the bottom-left plot with those in the bottom-right plot), since $g(\mathbf{x})$ is essentially flat far from the decision boundary, and thus pure gradient descent on g can not even commence for many malicious samples, as previously mentioned. In this case, the mimicry term is thus critical for finding a reasonable descent path to evasion.

Discussion. Our attacks raise questions about the feasibility of detecting malicious PDFs solely based on logical structure. We found that `/Linearized`, `/OpenAction`, `/Comment`, `/Root` and `/PageLayout` were among the most commonly manipulated keywords. They indeed are found mainly in legitimate PDFs, but can be easily added to malicious PDFs by the versioning mechanism. The attacker can simply insert comments inside the malicious PDF file to augment its `/Comment` count. Similarly, she can embed *legitimate* OpenAction code to add `/OpenAction` keywords or add new pages to insert `/PageLayout` keywords.

5 Conclusions, limitations and future work

In this work we proposed a simple algorithm for evasion of classifiers with differentiable discriminant functions. We investigated the attack effectiveness in the case of perfect and limited knowledge of the attacked system, and empirically showed that very popular classification algorithms (in particular, SVMs and neural networks) can still be evaded with high probability even if the adversary can only learn a copy of the classifier from a small surrogate dataset. Thus, our investigation raises important questions on whether such algorithms can be reliably employed in security-sensitive applications.

We believe that the proposed attack formulation can be extended to classifiers with non-differentiable discriminant functions as well, such as decision trees and k-nearest neighbors; *e.g.*, by defining suitable search heuristics similar to our mimicry term to minimize $g(\mathbf{x})$.

Interestingly our analysis also suggests improvements for classifier security. From Fig. 1, it is clear that a tighter *enclosure* of the legitimate samples increasingly forces the adversary to mimic the legitimate class, which may not always be possible; *e.g.*, malicious network packets or PDF files must contain a valid exploit for the attack to be successful. Accordingly, more secure classifiers can be designed by employing regularization terms that promote enclosure of the legitimate class; *e.g.*, by penalizing "blind spots" - regions with low $p(\mathbf{x})$ - classified

as legitimate. Alternatively, one may explicitly model the attack distribution, as in [4]; or add the generated attack samples to the training set. Nevertheless, improving security probably must be balanced with a higher FP rate.

In our example applications, the feature representations could be *inverted* to obtain a corresponding real-world objects (*e.g.*, spam emails, or PDF files); *i.e.*, it is straightforward to manipulate the given real-world object to obtain the desired feature vector $\mathbf{x}^*$ of the *optimal* attack. However, in practice some complex feature mappings can not be easily inverted; *e.g.*, n-gram features [11]. Another idea would be to modify the real-world object at each step of the gradient descent to obtain a sample in the feature space which is as close as possible to the sample that would be obtained at the next attack iteration. A similar technique has been already exploited by [6] to overcome the pre-image problem.

Other interesting extensions of our work may be to (*i*) consider more effective strategies such as those proposed by [19, 22] to build a small but representative set of surrogate data; and (*ii*) improve the classifier estimate $\hat{g}(\mathbf{x})$. To this end, one may exploit ensemble techniques such as bagging or the random subspace method to train several classifiers and then average their output.

Acknowledgments. This work has been partly supported by the project CRP-18293, L.R. 7/2007, Bando 2009, and by the project "Advanced and secure sharing of multimedia data over social networks in the future Internet" (CUP F71J11000690002), both funded by Regione Autonoma della Sardegna. Davide Maiorca gratefully acknowledges Regione Autonoma della Sardegna for the financial support of his PhD scholarship. Blaine Nelson thanks the Alexander von Humboldt Foundation for providing additional financial support. The opinions expressed in this paper are solely those of the authors and do not necessarily reflect the opinions of any sponsor.

References

1. Adobe: PDF Reference, sixth edition, version 1.7
2. Barreno, M., Nelson, B., Sears, R., Joseph, A.D., Tygar, J.D.: Can machine learning be secure? In: ASIACCS '06: Proc. of the 2006 ACM Symp. on Information, computer and comm. security. pp. 16–25. ACM, New York, NY, USA (2006)
3. Biggio, B., Fumera, G., Roli, F.: Multiple classifier systems for robust classifier design in adversarial environments. Int'l J. of Machine Learning and Cybernetics 1(1), 27–41 (2010)
4. Biggio, B., Fumera, G., Roli, F.: Design of robust classifiers for adversarial environments. In: IEEE Int'l Conf. on Systems, Man, and Cybernetics (SMC). pp. 977–982 (2011)
5. Biggio, B., Fumera, G., Roli, F.: Security evaluation of pattern classifiers under attack. IEEE Trans. on Knowl. and Data Eng. 99(PrePrints), 1 (2013)
6. Biggio, B., Nelson, B., Laskov, P.: Poisoning attacks against support vector machines. In: Langford, J., Pineau, J. (eds.) 29th Int'l Conf. on Mach. Learn. (2012)
7. Brückner, M., Scheffer, T.: Stackelberg games for adversarial prediction problems. In: Knowl. Disc. and D. Mining (KDD). pp. 547–555 (2011)

8. Brückner, M., Kanzow, C., Scheffer, T.: Static prediction games for adversarial learning problems. J. Mach. Learn. Res. 13, 2617–2654 (2012)
9. Dalvi, N., Domingos, P., Mausam, Sanghai, S., Verma, D.: Adversarial classification. In: 10th ACM SIGKDD Int'l Conf. on Knowl. Discovery and Data Mining (KDD). pp. 99–108. (2004)
10. Dekel, O., Shamir, O., Xiao, L.: Learning to classify with missing and corrupted features. Mach. Learn. 81, 149–178 (2010)
11. Fogla, P., Sharif, M., Perdisci, R., Kolesnikov, O., Lee, W.: Polymorphic blending attacks. In: Proc. 15th Conf. on USENIX Sec. Symp. USENIX Association, CA, USA (2006)
12. Globerson, A., Roweis, S.T.: Nightmare at test time: robust learning by feature deletion. In: Cohen, W.W., Moore, A. (eds.) Proc. of the 23rd Int'l Conf. on Mach. Learn. vol. 148, pp. 353–360. ACM (2006)
13. Golland, P.: Discriminative direction for kernel classifiers. In: Neu. Inf. Proc. Syst. (NIPS). pp. 745–752 (2002)
14. Huang, L., Joseph, A.D., Nelson, B., Rubinstein, B., Tygar, J.D.: Adversarial machine learning. In: 4th ACM Workshop on Art. Int. and Sec. (AISec 2011). pp. 43–57. Chicago, IL, USA (2011)
15. Kloft, M., Laskov, P.: Online anomaly detection under adversarial impact. In: Proc. of the 13th Int'l Conf. on Art. Int. and Stats. (AISTATS). pp. 405–412 (2010)
16. Kolcz, A., Teo, C.H.: Feature weighting for improved classifier robustness. In: Sixth Conf. on Email and Anti-Spam (CEAS). Mountain View, CA, USA (2009)
17. Laskov, P., Kloft, M.: A framework for quantitative security analysis of machine learning. In: AISec '09: Proc. of the 2nd ACM works. on Sec. and art. int. pp. 1–4. ACM, New York, NY, USA (2009)
18. LeCun, Y., Jackel, L., Bottou, L., Brunot, A., Cortes, C., Denker, J., Drucker, H., Guyon, I., Müller, U., Säckinger, E., Simard, P., Vapnik, V.: Comparison of learning algorithms for handwritten digit recognition. In: Int'l Conf. on Art. Neu. Net. pp. 53–60 (1995)
19. Lowd, D., Meek, C.: Adversarial learning. In: Press, A. (ed.) Proc. of the Eleventh ACM SIGKDD Int'l Conf. on Knowl. Disc. and D. Mining (KDD). pp. 641–647. Chicago, IL. (2005)
20. Maiorca, D., Giacinto, G., Corona, I.: A pattern recognition system for malicious pdf files detection. In: MLDM. pp. 510–524 (2012)
21. Nelson, B., Barreno, M., Chi, F.J., Joseph, A.D., Rubinstein, B.I.P., Saini, U., Sutton, C., Tygar, J.D., Xia, K.: Exploiting machine learning to subvert your spam filter. In: LEET'08: Proc. of the 1st Usenix Work. on L.-S. Exp. and Emerg. Threats. pp. 1–9. USENIX Association, Berkeley, CA, USA (2008)
22. Nelson, B., Rubinstein, B.I., Huang, L., Joseph, A.D., Lee, S.J., Rao, S., Tygar, J.D.: Query strategies for evading convex-inducing classifiers. J. Mach. Learn. Res. 13, 1293–1332 (2012)
23. Platt, J.: Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. In: Smola, A., Bartlett, P., Schölkopf, B., Schuurmans, D. (eds.) Adv. in L. M. Class. pp. 61–74 (2000)
24. Smutz, C., Stavrou, A.: Malicious pdf detection using metadata and structural features. In: Proc. of the 28th Annual Comp. Sec. App. Conf.. pp. 239–248 (2012)
25. Šrndić, N., Laskov, P.: Detection of malicious pdf files based on hierarchical document structure. In: Proc. 20th Annual Net. & Dist. Sys. Sec. Symp. (2013)
26. Young, R.: 2010 IBM X-force mid-year trend & risk report. Tech. rep., IBM (2010)