# Secure Kernel Machines against Evasion Attacks

Paolo Russu
Università di Cagliari
Piazza d'Armi
09123, Cagliari, Italy
paolo.russu@diee.unica.it

Ambra Demontis
Università di Cagliari
Piazza d'Armi
09123, Cagliari, Italy
ambra.demontis@diee.unica.it

Battista Biggio
Università di Cagliari
Piazza d'Armi
09123, Cagliari, Italy
battista.biggio@diee.unica.it

Giorgio Fumera
Università di Cagliari
Piazza d'Armi
09123, Cagliari, Italy
fumera@diee.unica.it

Fabio Roli
Università di Cagliari
Piazza d'Armi
09123, Cagliari, Italy
roli@diee.unica.it

## ABSTRACT

Machine learning is widely used in security-sensitive settings like spam and malware detection, although it has been shown that malicious data can be carefully modified at test time to evade detection. To overcome this limitation, adversary-aware learning algorithms have been developed, exploiting robust optimization and game-theoretical models to incorporate knowledge of potential adversarial data manipulations into the learning algorithm. Despite these techniques have been shown to be effective in some adversarial learning tasks, their adoption in practice is hindered by different factors, including the difficulty of meeting specific theoretical requirements, the complexity of implementation, and scalability issues, in terms of computational time and space required during training. In this work, we aim to develop secure kernel machines against evasion attacks that are not computationally more demanding than their non-secure counterparts. In particular, leveraging recent work on robustness and regularization, we show that the security of a linear classifier can be drastically improved by selecting a proper regularizer, depending on the kind of evasion attack, as well as unbalancing the cost of classification errors. We then discuss the security of nonlinear kernel machines, and show that a proper choice of the kernel function is crucial. We also show that unbalancing the cost of classification errors and varying some kernel parameters can further improve classifier security, yielding decision functions that better enclose the legitimate data. Our results on spam and PDF malware detection corroborate our analysis.

## Keywords

adversarial machine learning; evasion attacks; secure learning; kernel methods



*AISec'16, October 28 2016, Vienna, Austria*



DOI: http://dx.doi.org/10.1145/2996758.2996771

## 1. INTRODUCTION

In recent years, machine learning has been increasingly used in security-related applications, including spam, malware and network intrusion detection [1, 4, 6–8, 15, 33]. One of the main reasons is that, thanks to its generalization capability, machine learning has the potential to detect never-before-seen attacks and threats. However, it has been shown that specific vulnerabilities of learning algorithms can be exploited by skilled attackers to mislead learning, *i.e.*, the learning algorithm can be itself the weakest link in the security chain. In one relevant scenario, usually referred to as *evasion*, malicious samples are carefully modified at test time to evade detection. Another pertinent setting is related to the so-called *poisoning* attacks, where the attacker can inject poisoning samples into the training data used to learn the classifier, in order to compromise the training phase [8, 33]. In this paper we restrict our focus on evasion attacks, and on how to learn more secure classifiers against them, without increasing training complexity.

The intrinsic vulnerabilities of learning algorithms to well-crafted attacks are essentially rooted on their underlying *stationarity* assumption. In particular, learning algorithms have been originally designed by assuming that training and testing samples are drawn from the same distribution. However, such an assumption is clearly violated when attackers manipulate the input data either at training or testing time. From a very general theoretical viewpoint, this means that the class-conditional distribution of malicious samples observed at test time is different from that observed at training time. In the case of evasion attacks, only malicious data at test time is affected. Thus, by denoting the input samples with $\boldsymbol{x} \in \mathcal{X}$ (in a continuous feature space), and their class labels with $y \in \{-1, +1\}$ (respectively, for legitimate and malicious samples), this can be formalized as:

$$p_{\rm ts}(\boldsymbol{x}'|y=+1) = \int_{\boldsymbol{x}\in\mathcal{X}} p(\boldsymbol{x}'|\boldsymbol{x}, y=+1) p_{\rm tr}(\boldsymbol{x}|y=+1) d\boldsymbol{x}\,,$$

where $\boldsymbol{x}' = a(\boldsymbol{x})$, being $a : \mathcal{X} \mapsto \mathcal{X}$ a manipulation function representing the attack strategy, *i.e.*, defining how the attacker manipulates the initial malicious data $\boldsymbol{x}$ as $\boldsymbol{x}'$ to evade detection at test time. The term $p(\boldsymbol{x}'|\boldsymbol{x}, y=+1)$ characterizes the probability of having the initial malicious sample $\boldsymbol{x}$ modified as $\boldsymbol{x}'$, and $p_{\rm ts}$ and $p_{\rm tr}$ respectively denote

the testing and training class-conditional distributions of the malicious samples. This straightforward model, albeit being not directly useful to design secure learning algorithms (see, *e.g.*, [2]), clarifies the connection between the manipulation function $a$ and the *adversarial drift* that it induces in the probability distribution of malicious samples.

To account for this potential, adversarial drift between training and testing distributions, adversary-aware learning algorithms have been developed, based on robust optimization, and probabilistic and game-theoretical models (see, *e.g.*, [9, 14, 31]). The underlying idea of these algorithms is that of incorporating knowledge of the potential adversarial data manipulations into the learning phase, either by simulating such manipulations at training time, through the definition of suitable manipulation functions $a(\boldsymbol{x})$, or by modeling the distribution drift directly (in generative models). In practice, both options reflect a similar effect, *i.e.*, an adversarial shift of the malicious distribution, as witnessed by the aforementioned probability model. The only difference is the level at which assumptions on the attacker model are made, *i.e.*, either at the level of each malicious sample, or at the higher level of their global probability distribution. Clearly, making assumptions at the sample level allows one to more finely define the potential adversarial data manipulations, which can be advantageous when application-specific constraints on data manipulation can be accounted for. On the other hand, secure learning techniques based on this approach tends to exhibit a much higher training complexity, especially in terms of computational time and space. This is one of the main factors that hinders the adoption of these algorithms in practice, along with the difficulty of meeting some theoretical requirements, and, in some cases, the complexity of their implementation.

In this work, we aim to overcome these limitations by developing secure kernel machines against evasion attacks that are not computationally more demanding than their non-secure counterparts. To this end, we first clarify differences between sparse and dense evasion attacks, depending on whether it is convenient for the attacker to significantly modify few features, or slightly modify many of them. We also intuitively explain the source of classifier vulnerability to evasion attacks, depending on the shape of the decision function. As an aside contribution of this work, we also discuss briefly how to evade non-differentiable classification functions, like those exhibited by decision trees and random forests, through the use of a surrogate classifier (Sect. 2). In Sect. 3, we summarize recent findings on regularization and robustness properties of learning algorithms [17, 34], which will subsequently help us to shed light on the role of regularization and cost-sensitive learning to design more secure *linear* classifiers, depending on the kind of evasion attack. In Sect. 4, we devise some upper bounds on the worst-case impact of sparse and dense evasion attacks on linear and nonlinear classification functions. Besides confirming the findings in [17, 34], our analysis also allows us to investigate robustness properties of nonlinear kernel machines. We indeed show that a proper choice of the kernel function is crucial to improve security depending on the hypothesized kind of evasion attack, similarly to the role of regularization in linear classifiers. Our analysis also suggests that unbalancing the cost of classification errors and varying some kernel parameters can further improve classifier security. In Sects. 5-6, we describe some secure (linear and nonlinear) classifiers developed according to our findings. In Sect. 7, we conduct an experimental analysis on spam and PDF malware detection to corroborate our analysis. We conclude this paper by discussing related work (Sect. 8) and future research directions (Sect. 9).

## 2. EVASION ATTACKS

To analyze possible attacks against machine learning and devise principled countermeasures in a more systematic manner, a formal model of the attacker has been proposed in [1, 4, 6, 7, 15]. The model relies upon the definition of the attacker's goal, knowledge of the classifier, and capability of manipulating the input data. Here we formalize evasion attacks in terms of this model, as done in [4, 7].

**Attacker's goal.** In evasion attacks, the goal is to modify a single malicious sample (*e.g.*, a spam email) to have it misclassified as legitimate (with the largest confidence) by the classifier.

**Attacker's knowledge.** The attacker can have different levels of knowledge of the targeted classifier; she may have *limited* or *perfect* knowledge about the training data, the feature set, and the classification algorithm [4, 7]. In this work, we focus on *perfect-knowledge* (worst-case) attacks. Although it may be overly pessimistic to assume that the attacker knows everything about the targeted system, this often reveals interesting properties of learning algorithms, as it highlights the worst possible performance degradation that may be incurred under attack.

**Attacker's capability.** In evasion attacks, the attacker can only modify malicious samples, and the amount of feasible manipulations is often bounded, as the malicious data has to preserve its intrusive functionality. For example, malware has to embed a valid exploitation code for the attack to be effective, and spam emails have to remain readable by humans. It is thus clear that excessive obfuscation or manipulation of such samples is not possible without compromising the functionality of the attack. This behavior has been formalized in terms of application-dependent constraints in previous work [4, 32]; in particular, in terms of bounds on the distance between the initial malicious sample $\boldsymbol{x}$ and the manipulated one $\boldsymbol{x}'$ in feature space. As discussed in [32], two kinds of constraints have been mostly used when modeling real-world adversarial settings, leading one to define *sparse* ($\ell_1$) and *dense* ($\ell_2$) attacks. Bounding the $\ell_1$ distance between $\boldsymbol{x}$ and $\boldsymbol{x}'$ yields a sparse attack, as it represents the case when the cost depends on the number of modified features. For instance, when instances correspond to text (*e.g.*, the email's body) and each feature represents the occurrences of a given term in the text, the attacker usually aims to change as few words as possible. Instead, the $\ell_2$ distance yields a dense attack, as it represents the case when the cost of modifying features is proportional to the distance between the original and modified sample in Euclidean space. For example, when considering images as the input data, usually the attacker prefers making small changes to many or even all pixels, rather than significantly modifying only few of them. This amounts to only slightly blurring the image, instead of obtaining a salt-and-pepper noise effect (as produced by sparse attacks), and the final effect is less visible to the human eye (as we will show in the examples of manipulated handwritten digits in Sect. 7).

**Attack strategy.** Having defined the attacker's goal, knowl-

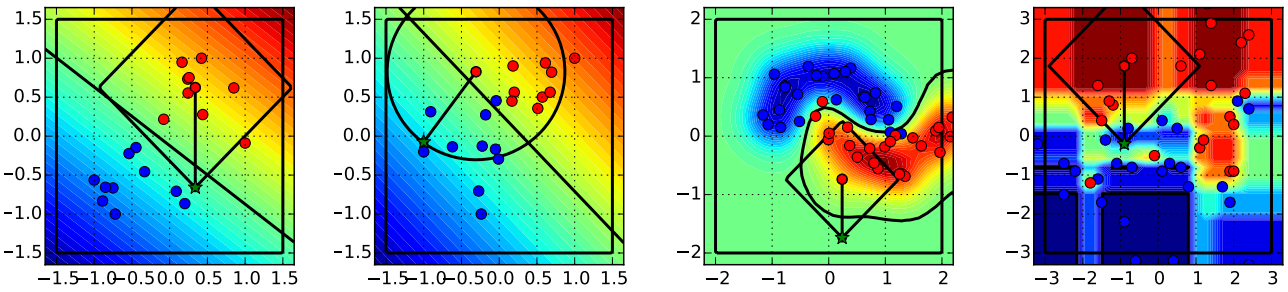

**Figure 1: Evasion attacks against different classifiers, trained on blue (legitimate) and red (malicious) samples. A linear SVM classifier against sparse (first plot) and dense (second plot) evasion attacks, an SVM with the RBF kernel (third plot) and a random forest (fourth plot) against sparse evasion attacks. The initial malicious point $\boldsymbol{x}$ is found at the center of the distance constraint, while the evasion sample $\boldsymbol{x}^\star$ is denoted with a green star. For each classifier, $g(\boldsymbol{x})$ values are shown in colors, and the black line denotes the decision boundary.**

edge and capability, one can finally formalize the attack strategy, *i.e.*, the procedure for obfuscating malicious data to evade detection, in terms of an optimization problem. Let us denote the legitimate and malicious class labels respectively with $-1$ and $+1$, and assume that the classifier's decision function is $f(\boldsymbol{x}) = \text{sign}\,(g(\boldsymbol{x}))$, where $g(\boldsymbol{x}) \in \mathbb{R}$ is the classifiers' linear discriminant function, and $\boldsymbol{x}$ is the representation of a sample in a $\mathsf{d}$-dimensional feature space. For example, for linear classifier, $g(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} + b \in \mathbb{R}$, where $\boldsymbol{w} \in \mathbb{R}^{\mathsf{d}}$ are the feature weights, and $b \in \mathbb{R}$ is the bias. Given a malicious sample $\boldsymbol{x}$, the goal is to find the sample $\boldsymbol{x}^*$ that minimizes the classifier's discriminant function $g(\cdot)$ (*i.e.*, that is classified as legitimate with the highest possible confidence) subject to the constraint that $\boldsymbol{x}^*$ lies within a distance $d_{\max}$ from $\boldsymbol{x}$:

$$\boldsymbol{x}^* = \quad \arg\min_{\boldsymbol{x}'} g(\boldsymbol{x}') \tag{1}$$

$$\text{s.t.} \quad d(\boldsymbol{x}', \boldsymbol{x}) \leq d_{\max}\ ,\ \boldsymbol{x}_{\text{lb}} \preceq \boldsymbol{x}' \preceq \boldsymbol{x}_{\text{ub}}\ , \tag{2}$$

where $\boldsymbol{x}_{\text{lb}} \preceq \boldsymbol{x}' \preceq \boldsymbol{x}_{\text{ub}}$ represent a box constraint (as features are often normalized onto a compact domain), and the distance measure $d(\cdot, \cdot)$ is defined in terms of the cost of data manipulation (*e.g.*, the number of modified words in each spam) [4, 7, 13, 21, 36]. Sparse and dense evasion attacks are simply defined based on whether $d(\cdot, \cdot)$ corresponds to the $\ell_1$ or to the $\ell_2$ distance, respectively.

**Solving the evasion problem.** Depending on the kind of decision function and distance metric, Problem (1)-(2) can be casted in terms of a linear or a nonlinear programming problem.[1] For linear classifiers, the global minimum can be found, either in the case of $\ell_1$ or $\ell_2$ constraints. For nonlinear $g(\boldsymbol{x})$, the solution is typically found at a local minimum of the objective function. Problem (1)-(2) can be solved with standard solvers in both cases, although they may not be very efficient, as they do not exploit specific knowledge about the evasion problem (*e.g.*, sparsity of the solution, compact domain, *etc.*). We thus devise an ad-hoc solver based on exploring a descent direction aligned with the gradient of $g(\boldsymbol{x})$ by means of a bisect line search. Its basic structure is given as Algorithm 1. To reduce the number of iterations, and ensure quick convergence, we explore one feature at a time in the case of $\ell_1$ attacks (starting from the more promising feature, *i.e.*, the one exhibiting the highest gradient variation), as the solution will be sparse. Conversely, we simultaneously explore all the features in the case of $\ell_2$ attacks, as the solution will be likely to modify all feature values. We also minimize the number of gradient and function evaluations to further speed up our evasion algorithm; *e.g.*, we only re-compute the gradient of $g(\boldsymbol{x})$ when no better point is found on the descent direction under exploration. Finally, in the case of nonlinear $g(\boldsymbol{x})$, we exploit multiple initializations for $\boldsymbol{x}^{(0)}$ to mitigate issues related to the presence of multiple local minima.

[1]Note that Problem (1)-(2) becomes linear only for linear classifiers and sparse evasion attacks (using the $\ell_1$ distance).

**Algorithm 1** Evasion Attack

**Input:** $\boldsymbol{x}$: the malicious sample; $\boldsymbol{x}^{(0)}$: the initial location of the attack sample; $d_{\max}$: the maximum distance between $\boldsymbol{x}$ and $\boldsymbol{x}'$ (Eq. 2); $\boldsymbol{x}_{\text{lb}}$, $\boldsymbol{x}_{\text{ub}}$: the box constraint bounds (Eq. 2); $\epsilon$: a small positive constant.
**Output:** $\boldsymbol{x}'$: the evasion attack sample.

$i \leftarrow 0$
**repeat**
  $i \leftarrow i + 1$
  $t' = \arg\min_t g(\boldsymbol{x}^{(i-1)} - t\nabla g(\boldsymbol{x}^{(i-1)}))$ (line search)
  $\boldsymbol{x}^{(i)} \leftarrow \boldsymbol{x}^{(i-1)} - t'\nabla g(\boldsymbol{x}^{(i-1)})$
  **if** constraints in Eq. (2) are violated **then**
    Project $\boldsymbol{x}^{(i)}$ onto the feasible domain
  **end if**
**until** $g(\boldsymbol{x}^{(i)}) - g(\boldsymbol{x}^{(i-1)}) > \epsilon$
**return** $\boldsymbol{x}^{(i)}$

Examples of (sparse and dense) evasion attacks against Support Vector Machines (SVMs) and random forests are shown in Fig. 1. As random forests have a non-differentiable discriminant function $g(\boldsymbol{x})$, we construct a differentiable approximation $\hat{g}(\boldsymbol{x})$ by learning a surrogate SVM on the same training data used to learn the random forest, but replacing the (true) training labels with the classification labels assigned by the random forest to such data. Then, the surrogate SVM can be used to find a suitable descent direction, and run the evasion attack against the random forest. To our knowledge, only preliminary work has attempted the evasion of non-differentiable classifiers like decision trees, using black-box optimization strategies like genetic algorithms

and greedy descent techniques [16, 35]. Learning a surrogate (differentiable) model to solve the evasion problem should be more computationally efficient, at least in principle.

**Understanding classifier security.** Observing the shape of the nonlinear decision functions in Fig. 1 (third and fourth plot), and the corresponding evasion samples, one may interestingly note that, in some cases (see, *e.g.*, the evasion sample in the third plot), to evade detection, it suffices to create a sample that is *far enough* from the known malicious samples (learned by the classifier during training), without even mimicking any legitimate sample. In some other cases (see, *e.g.*, the evasion sample in the fourth plot), the attacker is instead required to *mimic* the characteristics exhibited by the *legitimate samples*, which can be a much harder task in real-world applications. This reveals an interesting insight on the security of nonlinear classifiers, as also already pointed out in recent work [3, 4], *i.e.*, that decision functions that better enclose the legitimate data tend to be more secure against evasion attacks. In practice, the main vulnerability of learning algorithms relies upon the fact that, sometimes, it is possible to evade detection by creating samples which are far enough from the rest of the training data.[2] In previous work [3], this vulnerability has been referred to as a vulnerability of the classification algorithm. Conversely, if the classifier only allows evasion if the attack sample is close enough to the legitimate data, and the attacker can nevertheless construct evasion samples successfully, then the vulnerability is related to the feature representation. In fact, if a legitimate and a malicious sample become indistinguishable from each other in terms of their feature values, then the vulnerability is clearly related to the choice of the feature representation.

**Constructing real-world attack samples.** The attack strategy discussed in this section allows one to find an evasion sample in terms of a set of desired feature values. Clearly, feature mappings in real-world security-related tasks can be very difficult to reverse-engineer and, accordingly, constructing the corresponding real-world attack samples (*e.g.*, malware samples) may not be trivial. As widely discussed in previous work [4, 6, 7, 15], it is clear that this problem demands for application-specific solutions, and it is thus out of the scope of this work. On the other hand, in some cases, feature mappings can be easily inverted and the corresponding samples easily constructed.

# 3. ROBUSTNESS AND REGULARIZATION

We clarify here the connection between regularization and input data uncertainty highlighted by the recent findings in [17, 20, 23, 30, 34]. In particular, Xu *et al.* [34] have considered the following *robust* optimization problem:

$$\min_{\boldsymbol{w},b} \max_{\boldsymbol{u}_1,..,\boldsymbol{u}_\mathsf{n}\in\mathcal{U}} \sum_{i=1}^{\mathsf{n}} \left(1 - y_i(\boldsymbol{w}^\top(\boldsymbol{x}_i - \boldsymbol{u}_i) + b)\right)_+ , \quad (3)$$

where $(z)_+$ is equal to $z \in \mathbb{R}$ if $z > 0$ and 0 otherwise, $\boldsymbol{u}_1, ..., \boldsymbol{u}_\mathsf{n} \in \mathcal{U}$ define a set of bounded perturbations of the training data $\{\boldsymbol{x}_i, y_i\}_{i=1}^{\mathsf{n}} \in \mathbb{R}^\mathsf{n} \times \{-1, +1\}^\mathsf{n}$, and the so-called *uncertainty set* $\mathcal{U}$ is defined as

$$\mathcal{U} \triangleq \{(\boldsymbol{u}_1, \ldots, \boldsymbol{u}_\mathsf{n}) | \sum_{i=1}^{\mathsf{n}} \|\boldsymbol{u}_i\|^* \leq c\}, \quad (4)$$

being $\|\cdot\|^*$ the dual norm of $\|\cdot\|$. Typical examples of uncertainty sets according to the above definition include $\ell_1$ and $\ell_2$ balls [30, 34].

Problem (3) amounts to minimizing the hinge loss for a two-class classification problem under worst-case, bounded perturbations of the training samples $\boldsymbol{x}_i$, *i.e.*, a typical setting in robust optimization [17, 20, 23, 30, 34]. Under some mild assumptions easily verified in practice (including non-separability of the training data), the authors have shown that the above problem is equivalent to the following non-robust, regularized optimization problem (*cf.* Th. 3 in [34]):

$$\min_{\boldsymbol{w},b} \; c\|\boldsymbol{w}\| + \sum_{i=1}^{\mathsf{n}} \left(1 - y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b)\right)_+ . \quad (5)$$

This means that, if the $\ell_2$ norm is chosen as the dual norm characterizing the uncertainty set $\mathcal{U}$, then $\boldsymbol{w}$ is regularized with the $\ell_2$ norm, and the above problem is equivalent to a standard SVM. If input data uncertainty is modeled with the $\ell_1$ norm, instead, the optimal regularizer would be the $\ell_\infty$ regularizer, and vice versa.[3]

**Uncertainty sets and cost-sensitive learning.** The work by Xu *et al.* [34] only considers uncertainty sets of the same size, *i.e.*, the same perturbation is applied on both the legitimate and the malicious class. However, it is clear that, under evasion, the malicious samples are potentially affected by a stronger worst-case perturbation than legitimate data. Interestingly, in their recent work, Katsumata and Takeda [17] have shown that different uncertainty sets can be accounted for on each sample (and thus, on each class too), by simply modifying the cost of each classification error. This means that it suffices to penalize differently errors in different classes to consider uncertainty sets of different sizes. In the SVM learning algorithm, this can be simply accounted for by setting a different $C$ value for legitimate and malicious samples. This in turn suggests that classifier security can be improved by unbalancing the costs of classification errors in different classes.

**Kernelization.** To conclude, note that these findings mostly hold for linear classifiers. In the case of nonlinear (kernelized) classifiers, the authors have essentially repeated their analysis but considering perturbations directly in the feature space induced by the kernel function, instead of retaining them in the input space. Although this may be useful to understand how to regularize nonlinear functions, the resulting perturbation in the feature/kernel space depends on the kernel mapping itself, and it is not trivial to understand how it could be modified by the kernel function. For instance, if one considers an $\ell_1$ perturbation in input space, and an RBF kernel $k(\boldsymbol{x}, \boldsymbol{z}) = \exp(-\gamma\|\boldsymbol{x} - \boldsymbol{z}\|_2^2)$, the corresponding perturbation in the feature/kernel space is likely to become dense for sufficiently small $\gamma$ values. Intuitively, this can be explained by the fact that a sparse modification on an input sample $\boldsymbol{x}$ tends to affect almost all kernel values computed between $\boldsymbol{x}$ and the rest of the training data. This analysis is thus not very helpful in the case of evasion attacks, as the corresponding perturbations are clearly applied in the input space. As we will see in the next section, in fact, for nonlinear classifiers it is not the choice of the regularization term that plays a crucial role for improving security, but rather the selection of a proper *kernel function*.

[2] They have been also referred to as *blind spots* in [24], and as *adversarial examples* in recent work related to the evasion of deep learning algorithms [25, 26].

[3] Note that the $\ell_1$ norm is the dual norm of the $\ell_\infty$ norm, and vice versa, while the $\ell_2$ norm is the dual norm of itself.

# 4. CLASSIFIER SECURITY

We discuss here different strategies that can be exploited to improve security of linear and nonlinear classifiers, respectively. Our rationale is to show that the maximum variation of a classifier's discriminant function under an evasion attack can be bounded, highlighting the factors that may harm classifier security, and discussing how to limit their impact. This will give us a set of guidelines to help designing more secure learning algorithms against evasion. It is also worth remarking that, very interestingly, some of the results arising from our analysis corroborate the findings discussed in the previous section for linear classifiers.

## 4.1 Linear Classifiers

We start by analyzing the worst-case variation of the discriminant function of a linear classifier under evasion. The discriminant function of a linear classifier is simply given as $g(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} + b$. Assuming that $\boldsymbol{x}$ is an initial malicious sample, and $\boldsymbol{x}'$ the corresponding manipulated evasion sample, one yields:

$$\Delta g = g(\boldsymbol{x}) - g(\boldsymbol{x}') = \boldsymbol{w}^\top (\boldsymbol{x} - \boldsymbol{x}') \,. \tag{6}$$

Note that, from the attacker's perspective, this variation has to be maximized to increase chances of successfully evade the targeted classifier, as the attacker's strategy in Problem (1)-(2) aims to minimize $g(\boldsymbol{x}')$.

**Sparse Attacks.** Under sparse evasion attacks, it is not difficult to see that $\Delta g$ (from Eq. 6) is upper bounded by the following quantity:

$$\Delta g \leq \|\boldsymbol{w}\|_\infty \times \|\boldsymbol{x} - \boldsymbol{x}'\|_1 \,, \tag{7}$$

where we remind the reader that $\|\boldsymbol{w}\|_\infty = \max_{j=1,\ldots,\mathsf{d}} |w_j|$. In fact, for sparse attacks, the solution $\boldsymbol{x}'$ is found by modifying the features that have been assigned the highest absolute weight values (see, *e.g.*, Fig. 1, first plot). In the worst case, the maximum $\Delta g$ is attained by modifying the most relevant feature of a quantity equal to $\|\boldsymbol{x} - \boldsymbol{x}'\|_1$.

**Dense Attacks.** Under dense evasion attacks, instead, the worst-case increase of $\Delta g$ corresponds to a linear shift of $\boldsymbol{x}$ towards the decision boundary (along the opposite direction to the hyperplane normal $\boldsymbol{w}$), *i.e.*, $\boldsymbol{x}' = \boldsymbol{x} - \|\boldsymbol{x} - \boldsymbol{x}'\|_2 \frac{\boldsymbol{w}}{\|\boldsymbol{w}\|_2}$ (see Fig. 1, second plot), which implies that:

$$\Delta g \leq \frac{\boldsymbol{w}^\top \boldsymbol{w}}{\|\boldsymbol{w}\|_2} \times \|\boldsymbol{x} - \boldsymbol{x}'\|_2 = \|\boldsymbol{w}\|_2 \times \|\boldsymbol{x} - \boldsymbol{x}'\|_2 \,. \tag{8}$$

The analysis of the worst-case $\Delta g$ values for linear classifiers highlights two interesting facts. The former is that the feature values should be bounded, to bound the maximum variation of the relevant features. This is normally not a problem, if feature normalization is used, as normalization techniques often map the input samples onto a compact domain. The latter fact is that, under sparse attacks, one should bound the infinity-norm of $\boldsymbol{w}$, while under dense attacks, it is better to penalize its $\ell_2$ norm. This means that it is better to use $\ell_\infty$ and $\ell_2$ regularization respectively against sparse and dense evasion attacks. This novel result in the context of adversarial learning also confirms the findings by Xu *et al.* [34] related to the relationship between robustness and regularization of learning algorithms.

## 4.2 Nonlinear Kernel Machines

Let us now analyze how to bound the maximum variation of $\Delta g$ for decision functions of the form:

$$g(\boldsymbol{x}) = \textstyle\sum_{i=1}^{\mathsf{n}} \alpha_i k(\boldsymbol{x}, \boldsymbol{x}_i) + b \,, \tag{9}$$

where $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ is the *kernel* function, and $\boldsymbol{x}_i$'s are the training samples. For example, for SVMs, the $\alpha_i$'s are not null only for the support vectors, and positive (respectively, negative) for malicious (legitimate) samples. The value of $\Delta g$ in these cases is simply given as:

$$\Delta g = \sum_{i=1}^{\mathsf{n}} \alpha_i \left( k(\boldsymbol{x}, \boldsymbol{x}_i) - k(\boldsymbol{x}', \boldsymbol{x}_i) \right) \,. \tag{10}$$

As we aim to obtain decision functions that can potentially enclose the legitimate data (as discussed in Sect. 2), we focus here on kernels with an exponential form, including the RBF and the Laplacian kernel. They are respectively given by $k(\boldsymbol{x}, \boldsymbol{x}') = \exp(-\gamma \|\boldsymbol{x} - \boldsymbol{x}'\|_p^p)$, with $p = 1, 2$. The reason is that such kernels yield decision functions whose values tend to decrease while getting farther from the training data, thus yielding enclosing decision functions around one of the two classes.[4] For these kernels, it is not difficult to see that:

$$k(\boldsymbol{x}', \boldsymbol{x}_i) = e^{-\gamma \|\boldsymbol{x}' - \boldsymbol{x}_i\|_p^p} \geq e^{-\gamma \|\boldsymbol{x}' - \boldsymbol{x}\|_p^p} e^{-\gamma \|\boldsymbol{x} - \boldsymbol{x}_i\|_p^p} \,, \tag{11}$$

where we use the triangle inequality $\|(\boldsymbol{x}' - \boldsymbol{x}) + (\boldsymbol{x} - \boldsymbol{x}_i)\|_p \leq \|\boldsymbol{x}' - \boldsymbol{x}\|_p + \|\boldsymbol{x} - \boldsymbol{x}_i\|_p$ to upper bound the $\ell_p$ norm (valid for $p = 1, 2$). Substituting the above lower bound for $k(\boldsymbol{x}', \boldsymbol{x}_i)$ into Eq. (10), one yields:

$$\Delta g \leq \sum_{i=1}^{\mathsf{n}} \alpha_i k(\boldsymbol{x}, \boldsymbol{x}_i) \left( 1 - e^{-\gamma \|\boldsymbol{x}' - \boldsymbol{x}\|_p^p} \right) \,. \tag{12}$$

This upper bound reveals some interesting properties about the security of nonlinear kernels. First, it is clear that, if $\boldsymbol{x}' = \boldsymbol{x}$, $\Delta g = 0$. Instead, if $\|\boldsymbol{x}' - \boldsymbol{x}\|_p^p \to \infty$, $\Delta g = \sum_i \alpha_i k(\boldsymbol{x}, \boldsymbol{x}_i)$ and, thus, $g(\boldsymbol{x}') = b$. This means that, if $b \geq 0$ and $\boldsymbol{x}'$ is far enough from the training data, the decision function encloses the legitimate class, and $\boldsymbol{x}'$ is classified as malicious (and vice versa for $b < 0$), confirming the class-enclosing property of such kernels.

**Kernel selection.** The upper bound in Eq. (12) depends on $\|\boldsymbol{x}' - \boldsymbol{x}\|_p^p$. Since sparse and dense evasion attacks tend to minimize the $\ell_1$ and the $\ell_2$ distance between the same points, it should be clear that $p$ should be chosen accordingly. Namely, if the evasion attack is sparse, then one should select the Laplacian kernel; otherwise, in case of dense attacks, the RBF kernel should be preferred. The reason is that such choices will minimize the value of $\|\boldsymbol{x}' - \boldsymbol{x}\|_p^p$, *i.e.*, they will enable one to map the evasion samples in a region of the kernel space which is closer to the non-manipulated malicious samples (thus yielding a lower variation of $g$, and requiring more modifications to evade detection). This is an important observation, and it has a similar effect to the choice of the regularization term for linear classifiers; in fact, if one knows whether a sparse or a dense attack is deemed more likely, then a better regularizer (for

[4] This has also been discussed in [28], exploiting a probabilistic model defined for open-set recognition, where the goal is to find enclosed decision functions around known training classes, to be able to detect novel classes at test time.

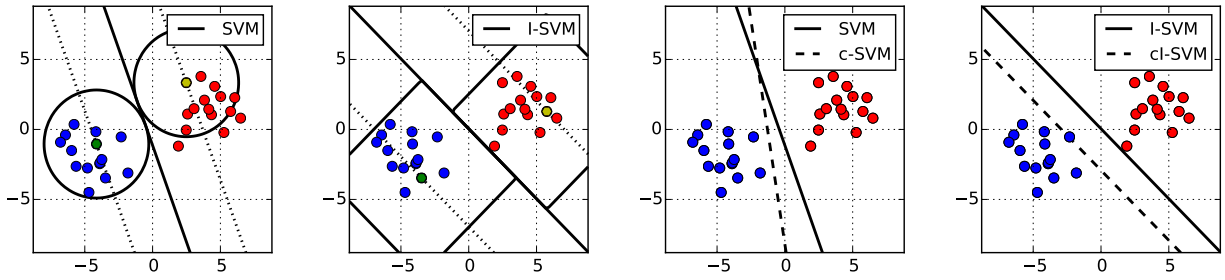


**Figure 2: Decision boundaries for SVM (first plot), I-SVM (second plot), and their cost-sensitive versions, C-SVM (third plot) and cI-SVM (fourth plot). In the first and the second plot, we also report $\ell_2$ and $\ell_1$ balls over the margin support vectors, to visually clarify why the orientation of the decision hyperplane changes.**

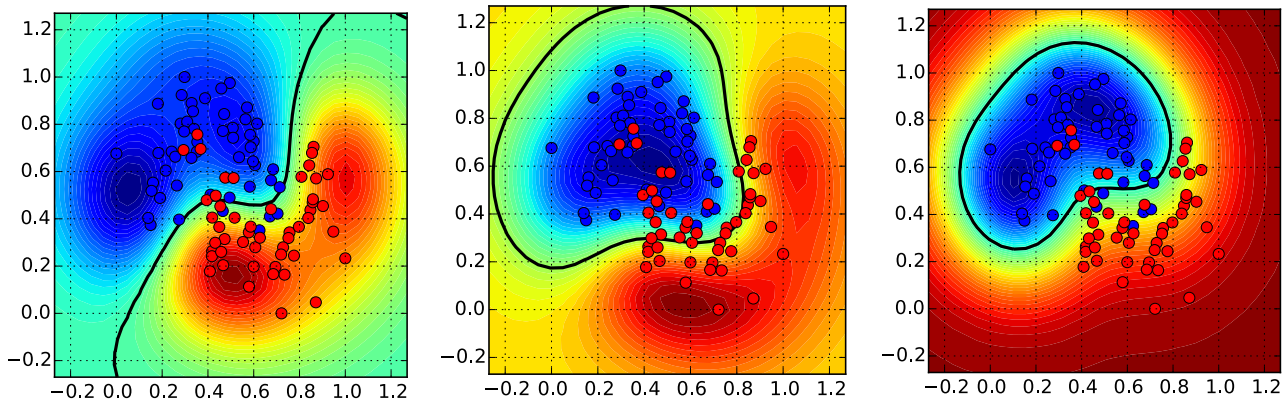

**Figure 3: Decision boundaries, and $g(\boldsymbol{x})$ values (in colors), for RBF-SVM (first plot), cRBF-SVM (second plot), and $\gamma$RBF-SVM (third plot). Note how the classifiers in the second and third plot provide a better enclosing of the legitimate data.**

linear classifiers) or kernel function (for nonlinear classifiers) can be selected.

**Cost-sensitive Learning.** Another non-trivial suggestion coming from Eq. (12) is to set a lower value of the cost of classification for malicious samples. The reason is that the (absolute) $\alpha_i$ values obtained from SVM learning are bounded by the corresponding value of the SVM parameter $C$. Thus, if we set a lower $C$ value for the malicious samples, their $\alpha_i$ values will decrease. This will in turn decrease the value of $\Delta g$, and thus, the impact of evasion attacks. Similarly, one may think of increasing $C$ for legitimate data, to further decrease $\Delta g$. Recall however that the balance condition in SVM learning requires $\sum_{i=1}^{n} \alpha_i = 0$ and, thus, the final $\alpha_i$ values will clearly depend on the data at hand (it is not generally the case that they will be equal to the corresponding $C$ value). Moreover, if one subsequently adjusts the value of $b$ on a validation set to meet some specific requirements (*e.g.*, a desired false positive rate), then it may be even convenient to unbalance costs in a very different way. It is thus difficult to draw general conclusions from Eq. (12), despite the fact that cost-sensitive learning may be useful to shape the decision boundary in a different way, potentially improving security.

**Kernel correction.** Our analysis also suggests that reducing the value of $\gamma$ should be beneficial, as it yields smoother functions. Furthermore, one may also think of assigning a different $\gamma$ to each class, and reduce only that assigned to the malicious training samples (as they are in turn assigned positive $\alpha_i$ values). Despite this breaks the positive-semidefiniteness and symmetry of the kernel function, it could improve classifier security by reducing the maximum value of $\Delta g$. Although standard SVM learning algorithms may not converge if the kernel function is not symmetric, we can still learn a linear SVM in the similarity space induced by our kernel, by essentially using the kernel matrix as the set of input features. This is a well-known technique in similarity-based classification, which amounts to learning the SVM on the squared kernel [10, 27].

# 5. SECURE LINEAR CLASSIFIERS

Here we discuss how to improve security of linear classifiers against dense and sparse attacks, respectively.

## 5.1 Countering Dense Attacks

Based on our discussion in Sect. 4.1, and on the findings in [34], one should consider $\ell_2$ regularization to counter $\ell_2$ attacks. Furthermore, as suggested in [17], also using unbalanced classification costs may improve classifier security.

**SVM.** Accordingly, the SVM learning algorithm, with different values of $C$ for each class, should guarantee a higher level of security against dense evasion attacks. It finds $\boldsymbol{w}$ and

$b$ by solving the following quadratic programming problem:

$$\min_{\boldsymbol{w},b} \quad \tfrac{1}{2}\|\boldsymbol{w}\|_2^2 + \sum_{i=1}^{n} c_i \left(1 - y_i g(\boldsymbol{x}_i)\right)_+ , \quad (13)$$

where $c_i = C_+$ ($c_i = C_-$) for malicious (legitimate) data.

## 5.2 Countering Sparse Attacks

For sparse attacks, our analysis, as that in [17, 34], suggests the use of $\ell_\infty$ regularization, potentially with unbalanced costs.

**Infinity-norm SVM (I-SVM).** We thus consider the SVM formulation, but changing the regularization term:

$$\min_{\boldsymbol{w},b} \quad \|\boldsymbol{w}\|_\infty + \sum_{i=1}^{n} c_i \left(1 - y_i g(\boldsymbol{x}_i)\right)_+ , \quad (14)$$

where the $c_i$'s are set as in the SVM case. Differently from the SVM learning problem, this one can be solved using a simpler linear programming approach.

**Remark I.** Examples of decision boundaries for the considered classifiers are shown in Fig. 2. Note that the effect of unbalanced classification costs tends to shift the decision boundary farther from the malicious class. Despite this may yield higher security, it may increase the fraction of misclassified legitimate samples (*i.e.*, the false positive rate). Therefore, its effectiveness as a valid defense strategy needs to be empirically assessed in detail, especially in those applications where keeping the false positive rate low is crucial.

**Remark II.** Another interesting observation is that the decision hyperplane of the I-SVM tends to yield more evenly-distributed weight values, *i.e.*, weights that are all equal in absolute value. This is clear from Fig. 2, as the hyperplane normal tends to align with a bisect line, *i.e.*, $\boldsymbol{w} \approxeq (1, 1)$. This is an important property for the security of linear classifiers, empirically validated in previous work [5, 18]. However, based on the interpretation of robustness and regularization in [34], and on our analysis, we have provided more theoretically-sound explanations behind the meaning of "evenly-distributed weights," and on how to enforce this behavior with a proper regularizer (instead of exploiting heuristic techniques).

# 6. SECURE KERNEL MACHINES

Similarly to the previous section, we consider here secure kernel machines against dense and sparse attacks.

## 6.1 Countering Dense Attacks

Based on the discussion in Sect. 4.2, to counter $\ell_2$ attacks, one may train a standard SVM with the RBF kernel (RBF-SVM), potentially using unbalanced classification costs (cRBF-SVM). A further option could be to assign distinct values of $\gamma$ for the malicious and legitimate training samples ($\gamma$RBF-SVM). Examples of decision boundaries for these classifiers are shown in Fig. 3.

## 6.2 Countering Sparse Attacks

In the case of sparse attacks, similar considerations can be made, despite the fact that one should use a Laplacian kernel. We thus consider the following classifiers: SVM with the Laplacian kernel (Lap-SVM), Lap-SVM with unbalanced costs (cLap-SVM), and Lap-SVM with different $\gamma$ values for each class ($\gamma$Lap-SVM).

**Secure LLR with unbalanced $\gamma$ ($\gamma$Sec-LLR).** To provide an example of a secure generative classifier, we consider the well-known log-likelihood ratio rule, for which the discriminant function is computed as:

$$g(\boldsymbol{x}) = \log p(\boldsymbol{x}|y = +1) - \log p(\boldsymbol{x}|y = -1) , \quad (15)$$

where the class-conditional probabilities are estimated through kernel density estimation, *i.e.*, $p(\boldsymbol{x}|y) = \frac{1}{n_y} \sum_{i|y_i=y} k(\frac{\boldsymbol{x}-\boldsymbol{x}_i}{h})$. Clearly, to counter sparse attacks, we select again the Laplacian kernel, and set a higher kernel variance through the parameter $h$ (*i.e.*, lower $\gamma$) for the malicious (positive) class.

# 7. EXPERIMENTAL ANALYSIS

We report here our experiments on linear and nonlinear secure learning algorithms.

## 7.1 Experiments on Linear Classifiers

For linear classifiers, we consider dense and sparse evasion attacks on handwritten digit data, to visually demonstrate their blurring and salt-and-pepper effect on images. We then consider an adversarial application example on spam filtering, against sparse evasion attacks. We first discuss the datasets used in this set of experiments.

**Handwritten Digit Classification.** For this task, we use the MNIST digit data [19], where each image is represented by a vector of 784 features, corresponding to its gray-level pixel values. As in [4], we simulate an adversarial classification problem where the digits 8 and 9 correspond to the legitimate and malicious class, respectively.

**Spam Filtering.** We consider this task, as spam filtering is a well-known application subject to adversarial attacks. Most spam filters include an automatic text classifier that analyzes the email's body text. In the simplest case Boolean features are used, each representing the presence or absence of a given term. For our experiments we use the TREC 2007 spam track data, consisting of about 25000 legitimate and 50000 spam emails [11]. We extract a dictionary of terms (features) from the first 5000 emails (in chronological order) using the same parsing mechanism of SpamAssassin, and then select the 200 most discriminant features according to the information gain criterion [29]. We simulate a well-known (*sparse*) evasion attack in which the attacker aims to modify only few terms. Adding or removing a term amounts to switching the value of the corresponding Boolean feature [4, 7, 18, 22, 36].

**Experimental Setup.** In these experiments, we consider the classifiers described in Sect. 5, namely, SVM, cSVM, I-SVM, cI-SVM. We randomly select 500 legitimate and 500 malicious samples from MNIST dataset, 2500 legitimate and 2500 malicious samples from the Spam dataset, and equally subdivide them to create a training and a testing set. We optimize the regularization parameter $C$ (or the cost-sensitive parameters $C_+$, $C_-$) of each SVM through 3-fold cross-validation, maximizing a trade-off between the detection rate (*i.e.*, the fraction of correctly-classified malicious samples, also referred to as true positive rate, TP) at 1% false positive rate (FP) in the absence of attack, and under attack (estimated by simulating the attacks on the validation set, for different $d_{\max}$ values).

After classifier training, we perform sparse and dense evasion attacks on all malicious digit testing samples, and sparse evasion attacks on all malicious spam testing samples, for increasing values of $d_{\max}$. For the digit data, $d_{\max}$ represents either the $\ell_2$ or $\ell_1$ distance between the non-manipulated

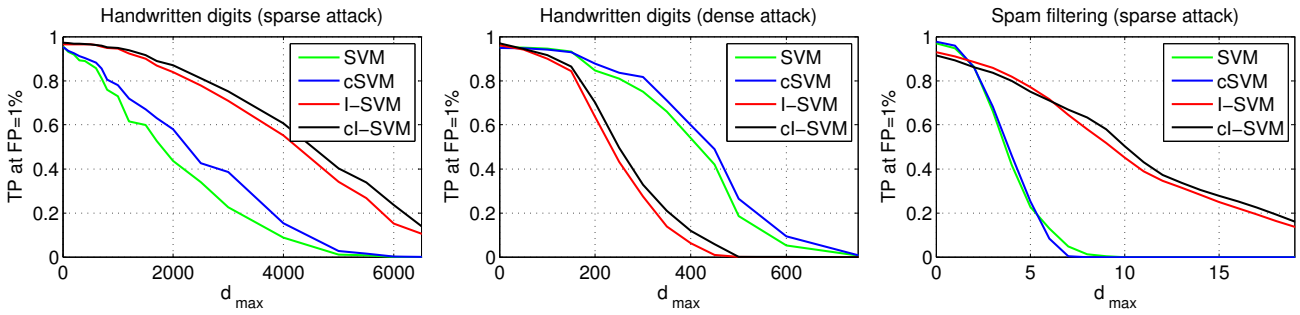


**Figure 4: Security evaluation curves (TP at FP=1% vs $d_{\max}$) for the 9-vs-8 digit classification task against dense (first plot) and sparse (second plot) evasion attacks, and for the spam filtering data against sparse evasion attacks (third plot).**

and the manipulated image, respectively, for dense and sparse attacks. In the case of sparse attacks, this corresponds to the number of gray-level pixel values modified by the attack. For spam filtering, it instead represents the number of modified words in each spam. We evaluate the corresponding performance in terms of TP at FP=1%, against an increasing value of $d_{\max}$ (recall that the performance in the absence of attack corresponds to $d_{\max} = 0$). We repeat this procedure five times, and report the average results on the original and modified testing data. The corresponding (averaged) curves are called *security evaluation curves* [4, 7].

**Results.** Results are reported in Fig. 4. The reported security evaluation curves show that the main improvement in classifier security is due to the choice of a proper regularizer. In fact, I-SVM (*i.e.*, infinity-norm regularization) is much more secure than SVM (*i.e.*, $\ell_2$ regularization) against sparse attacks, and vice versa for dense attacks, confirming the discussion in Sect. 5. The use of different classification costs only introduces a slight improvement in terms of security, as it is difficult to achieve an improved level of security while retaining FP=1%. Images of manipulated digits under dense and sparse evasion attacks are reported in Fig. 5.

## 7.2 Experiments on Nonlinear Classifiers

For nonlinear classifiers, we consider another adversarial application example involving the detection of malware in PDF files, following the detection approach proposed in [12].

**PDF Malware Detection.** This is another application that is often targeted by attackers. A PDF file can host different kinds of contents, like Flash and JavaScript, making it an appealing vector to convey malware. In fact, such third-party applications can be exploited by attacker to execute arbitrary operations. We use here the Lux0r dataset [12], which consists of 17,782 unique PDF documents embedding JavaScript code: 12,548 malicious samples and 5,234 benign samples. The whole dataset is the result of an extensive collection of PDF files from security blogs such as "Contagio", and "Malware don't need Coffee", analysis engines such as VirusTotal, and search engines such as Google and Yahoo. Every file is represented using 736 features that correspond to the number of occurrences of a predefined set of Javascript function calls (API), where every API represents an action performed by one of the objects that are contained into the PDF file (*e.g.*, opening another document that is stored inside the file). An attacker cannot trivially remove an API from a PDF file without corrupting its functionality. Conversely, she can easily add new APIs by inserting new function calls. For this reason, we simulate this attack by only considering feature increments (*i.e.*, decrementing a feature value is not allowed). Accordingly, the most convenient strategy to mislead a malware detector (classifier) is to insert as many occurrences of a *given* API as possible, which is a sparse attack.[5]

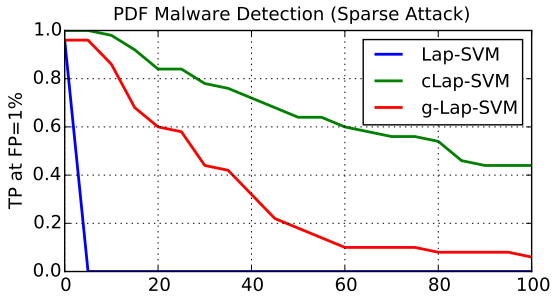


**Figure 6: Security evaluation curves (TP at FP=1% vs $d_{\max}$) for PDF malware detection against sparse evasion attacks.**

**Experimental Setup.** We consider the classifiers secure to sparse evasion attacks described in Sect. 6, namely, Lap-SVM, cLap-SVM, and $\gamma$Lap-SVM. We randomly split the dataset into training and testing sets of 5,000 samples each, and optimize the classifiers' parameters as done in the previous experimental setup. As in [12], we select a subset of 100 features from the training data, by retaining those exhibiting higher values in malicious data, such that mimicking legitimate samples becomes more difficult.

**Results.** The results averaged over five repetitions are reported in Fig. 6. It is easy to appreciate how the secure variants of the Lap-SVM algorithms outperform the baseline algorithm.

# 8. RELATED WORK

As mentioned in Sect. 1, several adversary-aware learning algorithms have been proposed to date, each relying on a different model of the attacker. As for linear classifiers,

[5]Despite no upper bound on the number of injected APIs may be set, we set the maximum value for each API to the corresponding one observed during training.

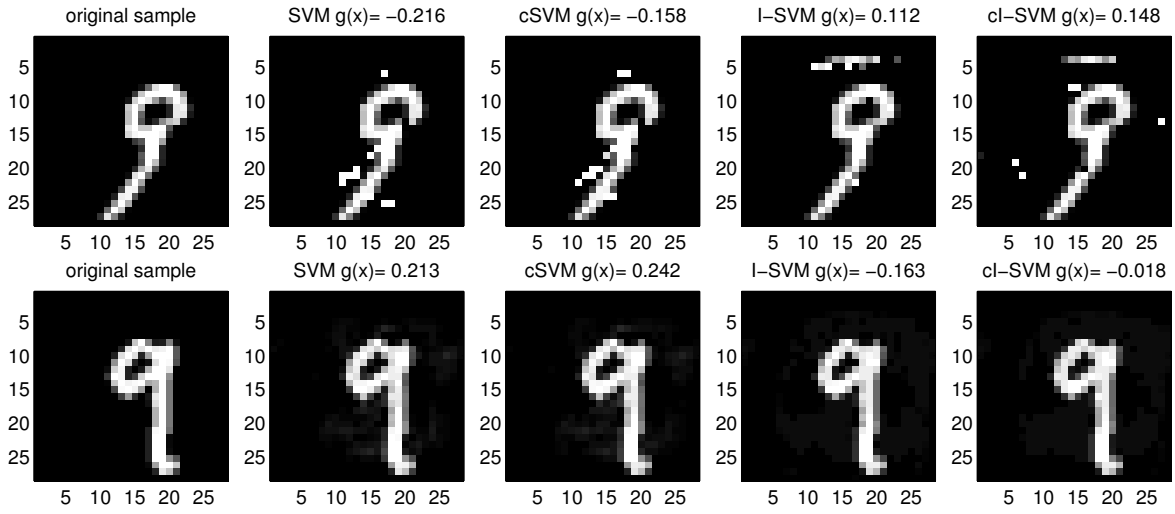


**Figure 5: Original and manipulated handwritten digits at $d_{\max} = 3000$ by sparse attacks (top row), and at $d_{\max} = 250$ by dense attacks (bottom row), against SVM (second column), c-SVM (third column), I-SVM (fourth column), and cI-SVM (fifth column). Values of $g(\boldsymbol{x})$ are also reported for each digit and classifier, confirming that sparse attacks are less effective against I-SVM and cI-SVM, and that dense attacks are less effective against SVM and cSVM. Note also how the blurring effect induced by dense attacks is more difficult to spot for humans than the salt-and-pepper noise induced by sparse attacks.**

the underlying rationale has been that of devising classifiers with more evenly-distributed feature weights, to intuitively enforce the attacker to manipulate more feature values to evade detection [5, 18]. Based on this intuition, different heuristic techniques have been proposed. In this work, highlighting connections between robustness and regularization, and thanks to our analysis in Sect. 4, we have provided a clearer explanation and a theoretically-sound approach to devise secure linear classifiers. First, we have clarified the notion of sparse and dense attacks, as done in [32]. Then, we have shown that infinity-norm regularization is the optimal solution against worst-case sparse attacks (as the ones only qualitatively envisioned in [5, 18]).

More complicated approaches have been exploited in [9, 14, 31], relying on game theory and robust optimization to model interactions between the classifier and the attacker. The main drawback of these approaches is their increased training complexity. For example, game-theoretical approaches (as that advocated in [9]) require simulating the attacks during training, and iteratively adjust the classification function. Another issue is that such approaches require specific assumptions to be met, to guarantee existence of a unique equilibrium in the game. For instance, in [9] the objective function of the attacker is required to be twice differentiable, and this is clearly not the case for sparse attacks (since the $\ell_1$ distance representing the attacker's cost to modify data is not differentiable). This means in turn that this approach can not deal with sparse attacks, at least in principle. The underling idea of the work in [14, 31] is instead to consider a worst-case loss suited to sparse attacks in which features can be set to their maximum/minimum values. In this case too, training complexity becomes higher with respect to the non-robust versions of the same algorithms.

## 9. CONCLUSIONS AND FUTURE WORK

In this work, we have provided several insights on how to enforce security of linear and nonlinear classifiers, from the choice of the regularization term and the kernel function, to the selection of different classification costs and kernel parameters. We have developed secure kernel machines that are not computationally more demanding than their non-secure counterparts, to reduce the risks associated to evasion attacks at test time. We believe that this would help overcoming the intrinsic limitations of current secure learning algorithms, namely, their strong theoretical requirements, complexity of implementation, and scalability issues due to their training complexity, to finally favor the wide adoption of more secure learning algorithms in practical settings.

As for future work, we aim to better systematize the state of the art in secure learning, and to extend our experimental analysis also to current secure-learning approaches. It may be worth considering also application settings in which the attack can be a combination of sparse and dense attacks, and try to mitigate their impact by exploiting a convex combination of $\ell_\infty$ and $\ell_2$ regularization. As another extension of this work, we plan to investigate, in a similar manner, the security properties of learning algorithms against poisoning attacks. Given that in a poisoning scenario the attacker can only inject few samples into the training set to mislead learning, poisoning can be considered a sparse attack (in terms of the number of samples). Thus, an interesting idea may be to consider an infinity-norm regularization on the $\alpha$ values of nonlinear kernel machines, such that more evenly-distributed weights are assigned to the training samples. This should indeed reduce the impact of each poisoning (outlying) sample in the training set, making learning more robust even in the presence of poisoning.

## References

[1] M. Barreno, B. Nelson, R. Sears, A. D. Joseph, and J. D. Tygar. Can machine learning be secure? In *Proc.*

*ACM Symp. Information, Computer and Comm. Sec.*, ASIACCS '06, pages 16–25, New York, NY, USA, 2006. ACM.

[2] J. Bi and T. Zhang. Support vector classification with input data uncertainty. In *Advances in Neural Information Processing Systems 17*, 2004.

[3] B. Biggio, I. Corona, Z.-M. He, P. P. K. Chan, G. Giacinto, D. S. Yeung, and F. Roli. One-and-a-half-class multiple classifier systems for secure learning against evasion attacks at test time. In F. Schwenker, F. Roli, and J. Kittler, editors, *Multiple Classifier Systems*, volume 9132 of *Lecture Notes in Computer Science*, pages 168–180. Springer International Publishing, 2015.

[4] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, and F. Roli. Evasion attacks against machine learning at test time. In H. Blockeel, K. Kersting, S. Nijssen, and F. Železný, editors, *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD), Part III*, volume 8190 of *Lecture Notes in Computer Science*, pages 387–402. Springer Berlin Heidelberg, 2013.

[5] B. Biggio, G. Fumera, and F. Roli. Multiple classifier systems for robust classifier design in adversarial environments. *Int'l J. Mach. Learn. and Cybernetics*, 1(1):27–41, 2010.

[6] B. Biggio, G. Fumera, and F. Roli. Pattern recognition systems under attack: Design issues and research challenges. *Int'l J. Patt. Recogn. Artif. Intell.*, 28(7):1460002, 2014.

[7] B. Biggio, G. Fumera, and F. Roli. Security evaluation of pattern classifiers under attack. *IEEE Transactions on Knowledge and Data Engineering*, 26(4):984–996, April 2014.

[8] B. Biggio, B. Nelson, and P. Laskov. Poisoning attacks against support vector machines. In J. Langford and J. Pineau, editors, *29th Int'l Conf. on Machine Learning*, pages 1807–1814. Omnipress, 2012.

[9] M. Brückner, C. Kanzow, and T. Scheffer. Static prediction games for adversarial learning problems. *J. Mach. Learn. Res.*, 13:2617–2654, September 2012.

[10] Y. Chen, E. K. Garcia, M. R. Gupta, A. Rahimi, and L. Cazzanti. Similarity-based classification: Concepts and algorithms. *J. Mach. Learn. Res.*, 10:747–776, June 2009.

[11] G. V. Cormack. Trec 2007 spam track overview. In E. M. Voorhees and L. P. Buckland, editors, *TREC*, volume Special Publication 500-274. National Institute of Standards and Technology (NIST), 2007.

[12] I. Corona, D. Maiorca, D. Ariu, and G. Giacinto. Lux0r: Detection of malicious pdf-embedded javascript code through discriminant analysis of API references. In *Proc. 2014 Workshop on Artificial Intelligent and Security Workshop*, AISec '14, pages 47–57, New York, NY, USA, 2014. ACM.

[13] N. Dalvi, P. Domingos, Mausam, S. Sanghai, and D. Verma. Adversarial classification. In *Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 99–108, Seattle, 2004.

[14] A. Globerson and S. T. Roweis. Nightmare at test time: robust learning by feature deletion. In W. W. Cohen and A. Moore, editors, *Proceedings of the 23rd International Conference on Machine Learning*, volume 148, pages 353–360. ACM, 2006.

[15] L. Huang, A. D. Joseph, B. Nelson, B. Rubinstein, and J. D. Tygar. Adversarial machine learning. In *4th ACM Workshop on Artificial Intelligence and Security (AISec 2011)*, pages 43–57, Chicago, IL, USA, 2011.

[16] A. Kantchelian, J. D. Tygar, and A. D. Joseph. Evasion and hardening of tree ensemble classifiers. In *33rd ICML*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 2387–2396. JMLR.org, 2016.

[17] S. Katsumata and A. Takeda. Robust cost sensitive support vector machine. In G. Lebanon and S. Vishwanathan, editors, *18th Int'l Conf. on Artificial Intelligence and Statistics (AISTATS)*, volume 38 of *JMLR Workshop and Conference Proceedings*, pages 434–443. JMLR.org, 2015.

[18] A. Kolcz and C. H. Teo. Feature weighting for improved classifier robustness. In *Sixth Conference on Email and Anti-Spam (CEAS)*, Mountain View, CA, USA, 2009.

[19] Y. LeCun, L. Jackel, L. Bottou, A. Brunot, C. Cortes, J. Denker, H. Drucker, I. Guyon, U. Müller, E. Säckinger, P. Simard, and V. Vapnik. Comparison of learning algorithms for handwritten digit recognition. In *Int'l Conf. on Artificial Neural Networks*, pages 53–60, 1995.

[20] R. Livni, K. Crammer, A. Globerson, E.-i. Edmond, and L. Safra. A simple geometric interpretation of SVM using stochastic adversaries. In *JMLR W&CP - Proc.*, volume 22 of *AISTATS '12*, pages 722–730, 2012.

[21] D. Lowd and C. Meek. Adversarial learning. In *Proc. 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 641–647, Chicago, IL, USA, 2005. ACM Press.

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

[23] T. Miyato, S.-i. Maeda, M. Koyama, K. Nakae, and S. Ishii. Distributional smoothing with virtual adversarial training. In *International Conference on Learning Representation*, 2016.

[24] B. Nelson, B. I. Rubinstein, L. Huang, A. D. Joseph, S. J. Lee, S. Rao, and J. D. Tygar. Query strategies for evading convex-inducing classifiers. *J. Mach. Learn. Res.*, 13:1293–1332, May 2012.

[25] A. M. Nguyen, J. Yosinski, and J. Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 427–436. IEEE, 2015.

[26] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami. The limitations of deep learning in adversarial settings. In *Proc. 1st IEEE European Symposium on Security and Privacy*, pages 372–387. IEEE, 2016.

[27] E. Pekalska, P. Paclik, and R. P. Duin. A generalized kernel approach to dissimilarity based classification. *Journal of Machine Learning Research*, 2:175–211, 2001.

[28] W. Scheirer, L. Jain, and T. Boult. Probability models for open set recognition. *IEEE Trans. Patt. An. Mach. Intell.*, 36(11):2317–2324, 2014.

[29] F. Sebastiani. Machine learning in automated text categorization. *ACM Comput. Surv.*, 34:1–47, March 2002.

[30] S. Sra, S. Nowozin, and S. J. Wright. *Optimization for Machine Learning*. The MIT Press, 2011.

[31] C. H. Teo, A. Globerson, S. Roweis, and A. Smola. Convex learning with invariances. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 1489–1496. MIT Press, Cambridge, MA, 2008.

[32] F. Wang, W. Liu, and S. Chawla. On sparse feature attacks in adversarial learning. In *IEEE Int'l Conf. on Data Mining (ICDM)*, pages 1013–1018. IEEE, 2014.

[33] H. Xiao, B. Biggio, G. Brown, G. Fumera, C. Eckert, and F. Roli. Is feature selection secure against training data poisoning? In F. Bach and D. Blei, editors, *JMLR W&CP - Proc. 32nd Int'l Conf. Mach. Learning (ICML)*, volume 37, pages 1689–1698, 2015.

[34] H. Xu, C. Caramanis, and S. Mannor. Robustness and regularization of support vector machines. *Journal of Machine Learning Research*, 10:1485–1510, July 2009.

[35] W. Xu, Y. Qi, and D. Evans. Automatically evading classifiers. In *Proc. 23rd Annual Network & Distributed System Security Symposium (NDSS)*. The Internet Society, 2016.

[36] F. Zhang, P. Chan, B. Biggio, D. Yeung, and F. Roli. Adversarial feature selection against evasion attacks. *IEEE Transactions on Cybernetics*, 46(3):766–777, 2016.