Is Feature Selection Secure against Training Data Poisoning?

Is Feature Selection Secure against Training Data Poisoning?

Huang Xiao XIAOHU@IN.TUM.DE
Department of Computer Science, Technische Universität München, Boltzmannstr.3, 85748 Garching, Germany

Battista Biggio BATTISTA.BIGGIO@DIEE.UNICA.IT
Department of Electrical and Electronic Engineering, University of Cagliari, Piazza d'Armi, 09123 Cagliari, Italy

Gavin Brown GAVIN.BROWN@MANCHESTER.AC.UK
School of Computer Science, University of Manchester, Oxford Road, M13 9PL, UK

Giorgio Fumera FUMERA@DIEE.UNICA.IT
Department of Electrical and Electronic Engineering, University of Cagliari, Piazza d'Armi, 09123 Cagliari, Italy

Claudia Eckert CLAUDIA.ECKERT@IN.TUM.DE
Department of Computer Science, Technische Universität München, Boltzmannstr.3, 85748 Garching, Germany

Fabio Roli ROLI@DIEE.UNICA.IT
Department of Electrical and Electronic Engineering, University of Cagliari, Piazza d'Armi, 09123 Cagliari, Italy

Abstract

Learning in adversarial settings is becoming an important task for application domains where attackers may inject malicious data into the training set to subvert normal operation of data-driven technologies. Feature selection has been widely used in machine learning for security applications to improve generalization and computational efficiency, although it is not clear whether its use may be beneficial or even counterproductive when training data are poisoned by intelligent attackers. In this work, we shed light on this issue by providing a framework to investigate the robustness of popular feature selection methods, including LASSO, ridge regression and the elastic net. Our results on malware detection show that feature selection methods can be significantly compromised under attack (we can reduce LASSO to almost random choices of feature sets by careful insertion of less than 5% poisoned training samples), highlighting the need for specific countermeasures.

Proceedings of the 32^{nd} International Conference on Machine Learning, Lille, France, 2015. JMLR: W&CP volume 37.

1. Introduction

With the advent of the modern Internet, the number of interconnected users and devices, along with the available number of services, has tremendously increased. This has not only simplified our lives, through accessibility and ease-of-use of novel services (*e.g.*, think to the use of maps and geolocation on smartphones), but it has also provided great opportunities for attackers to perform novel and profitable malicious activities. To cope with this phenomenon, machine learning has been adopted in security-sensitive settings like spam and malware detection, web-page ranking and network protocol verification (Sahami et al., 1998; McCallum & Nigam, 1998; Rubinstein et al., 2009; Barreno et al., 2010; Smutz & Stavrou, 2012; Brückner et al., 2012; Biggio et al., 2012; 2013b; 2014). In these applications, the challenge is that of inferring actionable knowledge from a large, usually high-dimensional data collection, to correctly prevent malware (*i.e.*, malicious software) infections or other threats. For instance, detection of malware in PDF files relies on the analysis of the PDF logical structure, which consists of a large set of different kinds of objects and metadata, yielding a high-dimensional data representation (Smutz & Stavrou, 2012; Maiorca et al., 2012; 2013; Šrndić & Laskov, 2013). Similarly, text classifiers for spam filtering rely on the construction of a large dictionary to identify words that are mostly discriminant of spam and legitimate emails (Sahami et al., 1998; McCallum & Nigam, 1998; Graham, 2002; Robinson, 2003).

Due to the large number of available features, learning in these tasks is particularly challenging. Feature selection is thus a crucial step for reducing the impact of the curse of dimensionality on classifier's generalization, and for learning efficient models providing easier-to-interpret decisions.

For the same reasons behind the growing sophistication and variability of modern attacks, it is reasonable to expect that, being increasingly adopted in these tasks, machine learning techniques will be soon targeted by specific attacks, crafted by skilled attackers. In the last years, relevant work in the area of adversarial machine learning has addressed this issue, and proposed some pioneering methods for secure learning against particular kinds of attacks (Barreno et al., 2006; Huang et al., 2011; Biggio et al., 2014; 2012; 2013a; Brückner et al., 2012; Globerson & Roweis, 2006).

While the majority of work has focused on analyzing vulnerabilities of classification and clustering algorithms, only recent work has considered intrinsic vulnerabilities introduced by the use of feature selection methods. In particular, it has been shown that classifier evasion can be facilitated if features are not selected according to an adversary-aware procedure that explicitly accounts for adversarial data manipulation at test time (Li & Vorobeychik, 2014; Wang et al., 2014; Zhang et al., 2015). Although these attacks do not directly target feature selection, but rather the resulting classification system, they highlight the need for adversarial feature selection procedures. Attacks that more explicitly target feature selection fall into the category of *poisoning attacks*. Under this setting, the attacker has access to the training data, and contaminates it to subvert or control the selection of the reduced feature set.

As advocated in a recent workshop (Joseph et al., 2013), poisoning attacks are an emerging security threat for data-driven technologies, and could become the most relevant one in the coming years, especially in the so-called big-data scenario dominated by data-driven technologies. From a practical perspective, poisoning attacks are already a pertinent scenario in several applications. For instance, in collaborative spam filtering, classifiers are retrained on emails labeled by end users. Attackers owning an authorized email account protected by the same anti-spam filter may thus arbitrarily manipulate emails in their inbox, *i.e.*, part of the training data used to update the classifier. Some systems may even ask directly to users to validate their decisions on some submitted samples, and use their feedback to update the classifier (see, *e.g.*, PDFRate, an online tool for detecting PDF malware designed by Smutz & Stavrou, 2012). Furthermore, in several cases obtaining accurate labels, or validating the available ground truth may be expensive and time-consuming; *e.g.*, if malware samples are collected from the Internet, by means of honeypots, *i.e.*, machines that purposely expose known vulnerabilities to be infected by malware (Spitzner, 2002), or other online services, like VirusTotal,[1] labeling errors are possible.

Work by Rubinstein et al. (2009) and Nelson et al. (2008) has shown the potential of poisoning attacks against PCA-based malicious traffic detectors and spam filters, and proposed robust techniques to mitigate their impact. More recently, Mei & Zhu (2015) have demonstrated how to poison latent Dirichlet allocation to drive its selection of relevant topics towards the attacker's choice.

In this work, we propose a framework to categorize and provide a better understanding of the different attacks that may target feature selection algorithms, building on previously-proposed attack models for the security evaluation of supervised and unsupervised learning algorithms (Biggio et al., 2014; 2013b; 2012; Huang et al., 2011; Barreno et al., 2006) (Sect. 2). We then exploit this framework to formalize poisoning attack strategies against popular embedded feature selection methods, including the so-called *least absolute shrinkage and selection operator* (LASSO) (Tibshirani, 1996), *ridge* regression (Hoerl & Kennard, 1970), and the *elastic net* (Zou & Hastie, 2005) (Sect. 3). We report experiments on PDF malware detection, assessing how poisoning affects both feature selection and classification error (Sect. 4). We conclude the paper by discussing our findings and contributions (Sect. 5), and sketching promising future research directions (Sect. 6).

2. Feature Selection Under Attack

In this section, we present our framework for the security evaluation of feature selection algorithms. It builds on the framework proposed by Biggio et al. (2014; 2013b) to assess the security of classification and clustering algorithms, which in turn relies on a taxonomy of attacks against learning algorithms originally proposed by Huang et al. (2011); Barreno et al. (2006). Following the framework of Biggio et al. (2014; 2013b), we define ours in terms of assumptions on the attacker's goal, knowledge of the system, and capability of manipulating the input data.

Notation. In the following, we assume data is generated according to an underlying i.i.d. process $p : \mathcal{X} \mapsto \mathcal{Y}$, for which we are only given a set $\mathcal{D} = \{\boldsymbol{x}_i, y_i\}_{i=1}^n$ of n samples, each consisting of a d-dimensional feature vector $\boldsymbol{x}_i = [x_i^1, \ldots, x_i^d]^\top \in \mathcal{X}$, and a target variable $y_i \in \mathcal{Y}$. Learning amounts to inferring the underlying process p from $\mathcal{D}$. Feature selection can be exploited to facilitate this task by selecting a suitable, relevant feature subset from $\mathcal{D}$, according to a given criterion. For instance, although in different forms, wrapper and embedded methods aim to minimize classification error, while information theoretic filters optimize different estimates of the information

[1] http://virustotal.com

gain (Brown et al., 2012). To denote a given feature subset, we introduce a vector $\boldsymbol{\pi} \in \{0,1\}^d$, where each element denotes whether the corresponding feature has been selected (1) or not (0). Then, a feature selection algorithm can be represented in terms of a function $h(\mathcal{D})$ that selects a feature subset $\boldsymbol{\pi}$ by minimizing a given selection criterion $\mathcal{L}(\mathcal{D}, \boldsymbol{\pi})$ (*e.g.*, the classification error).

2.1. Attacker's Goal

The attacker's goal is defined in terms of the desired *security violation*, which can be an **integrity**, **availability**, or **privacy** violation, and of the *attack specificity*, which can be **targeted** or **indiscriminate** (Barreno et al., 2006; Huang et al., 2011; Biggio et al., 2014; 2013b).

Integrity is violated if malicious activities are performed without compromising normal system operation, *e.g.*, attacks that evade classifier detection without affecting the classification of legitimate samples. In the case of feature selection, we thus regard integrity violations as attacks that only slightly modify the selected feature subset, aiming to facilitate subsequent evasion; *e.g.*, an attacker may aim to avoid the selection of some specific words by an anti-spam filter, as they are frequently used in her spam emails.

Availability is violated if the functionality of the system is compromised, causing a denial of service. For classification and clustering, this respectively amounts to causing the largest possible classification error and to maximally altering the clustering process on the input data (Huang et al., 2011; Biggio et al., 2014; 2013b). Following the same rationale, the availability of a feature selection algorithm is compromised if the attack enforces selection of a feature subset which yields the largest generalization error.

Privacy is violated if the attacker is able to obtain information about the system's users by reverse-engineering the attacked system. In our case, this would require the attacker to reverse-engineer the feature selection process, and, getting to know the selected features, infer information about the training data and the system users.[2]

Finally, the attack specificity is **targeted**, if the attack affects the selection of a *specific* feature subset, and **indiscriminate**, if the attack affects the selection of *any* feature.

2.2. Attacker's Knowledge

The attacker can have different levels of knowledge of the system, according to specific assumptions on the points $(k.i)$-$(k.iii)$ described in the following.

$(k.i)$ **Knowledge of the training data** $\mathcal{D}$: The attacker may have partial or full access to the training data $\mathcal{D}$. If no access is possible, she may collect a *surrogate* dataset $\hat{\mathcal{D}} = \{\hat{\boldsymbol{x}}_i, \hat{y}_i\}_{i=1}^m$, ideally drawn from the same underlying process p from which $\mathcal{D}$ was drawn, *i.e.*, from the same source from which samples in $\mathcal{D}$ were collected; *e.g.*, honeypots for malware samples (Spitzner, 2002).

$(k.ii)$ **Knowledge of the feature representation** $\mathcal{X}$: The attacker may partially or fully know how features are computed for each sample, before performing feature selection.

$(k.iii)$ **Knowledge of the feature selection algorithm** $h(\mathcal{D})$: The attacker may know that a specific feature selection algorithm is used, along with a specific selection criterion $\mathcal{L}(\mathcal{D})$; *e.g.*, the accuracy of a given classifier, if a wrapper method is used. In a very pessimistic setting, the attacker may even discover the selected feature subset.

Perfect Knowledge (PK). The worst-case scenario in which the attacker has full knowledge of the attacked system, is usually referred to as *perfect knowledge* case (Biggio et al., 2014; 2013b; 2012; Kloft & Laskov, 2010; Brückner et al., 2012; Huang et al., 2011; Barreno et al., 2006), and it allows one to empirically evaluate an upper bound on the performance degradation that can be incurred by the system under attack. In our case, it amounts to completely knowing: $(k.i)$ the data, $(k.ii)$ the feature set, and $(k.iii)$ the feature selection algorithm.

Limited Knowledge (LK). Attacks with *limited knowledge* have also been often considered, to simulate more realistic settings (Biggio et al., 2014; 2013b;a). In this case, the attacker is assumed to have only partial knowledge of $(k.i)$ the data, *i.e.*, she can collect a surrogate dataset $\hat{\mathcal{D}}$, but knows $(k.ii)$ the feature set, and $(k.iii)$ the feature selection algorithm. She can thus replicate the behavior of the attacked system using the surrogate data $\hat{\mathcal{D}}$, to construct a set of attack samples. The effectiveness of these attacks is then assessed against the targeted system (trained on $\mathcal{D}$).

2.3. Attacker's Capability

The attacker's capability defines how and to what extent the attacker can control the feature selection process. As for supervised learning, the attacker can influence both training and test data, or only test data, respectively exercising a **causative** or **exploratory** influence (more commonly referred to as poisoning and evasion attacks) (Barreno et al., 2006; Huang et al., 2011; Biggio et al., 2014). Although modifying data at test time does not affect the feature selection process directly, it may nevertheless influence the security of the corresponding classifier against evasion attacks at test time, as also highlighted in recent work (Li & Vorobeychik, 2014; Wang et al., 2014). Therefore, evasion

[2] Note that privacy attacks against machine learning are very speculative, and only considered here for completeness. Further, feature selection algorithms are typically unstable, making them very difficult to reverse-engineer. It would be thus of interest to understand whether feature selection exhibits some privacy guarantees, *e.g.*, if it can be intrinsically differentially private.

should be considered as a plausible attack scenario even against feature selection algorithms.

In poisoning attacks, the attacker is often assumed to control a small percentage of the training data $\mathcal{D}$ by injecting a fraction of well-crafted attack samples. The ability to manipulate their feature values and labels depends on how labels are assigned to the training data; *e.g.*, if malware is collected via *honeypots* (Spitzner, 2002), and labeled with some anti-virus software, the attacker has to construct poisoning samples under the constraint that they will be labeled as expected by the given anti-virus.

In evasion attacks, the attacker manipulates malicious data at test time to evade detection. Clearly, malicious samples have to be manipulated without affecting their malicious functionality, *e.g.*, malware code has to be obfuscated without compromising the exploitation routine. In several cases, these constraints can be encoded in terms of distance measures between the original, non-manipulated attack samples and the manipulated ones (Dalvi et al., 2004; Lowd & Meek, 2005; Globerson & Roweis, 2006; Teo et al., 2008; Brückner et al., 2012; Biggio et al., 2013a).

2.4. Attack Strategy

Following the approach in Biggio et al. (2013b), we define an *optimal* attack strategy to reach the attacker's goal, under the constraints imposed by her knowledge of the system and capabilities of manipulating the input data. To this end, we characterize the attacker's knowledge in terms of a space Θ that encodes assumptions $(k.i)$-$(k.iii)$ on the knowledge of the data $\mathcal{D}$, the feature space $\mathcal{X}$, the feature selection algorithm h, and the selection criterion $\mathcal{L}$. Accordingly, for PK and LK attacks, the attacker's knowledge can be respectively represented as $\boldsymbol{\theta}_{\rm PK} = (\mathcal{D}, \mathcal{X}, h, \mathcal{L})$ and $\boldsymbol{\theta}_{\rm LK} = (\hat{\mathcal{D}}, \mathcal{X}, h, \mathcal{L})$. We characterize the attacker's capability by assuming that an initial set of samples $\mathcal{A}$ is given, and that it is modified according to a space of possible modifications $\Phi(\mathcal{A})$. Given the attacker's knowledge $\boldsymbol{\theta} \in \Theta$ and a set of manipulated attacks $\mathcal{A}' \in \Phi(\mathcal{A})$, the attacker's goal can be characterized in terms of an objective function $\mathcal{W}(\mathcal{A}', \boldsymbol{\theta}) \in \mathbb{R}$ which evaluates how effective the attacks $\mathcal{A}'$ are. The optimal attack strategy can be thus given as:

$$\begin{aligned} \max_{\mathcal{A}'} \quad & \mathcal{W}(\mathcal{A}'; \boldsymbol{\theta}) \\ \text{s.t.} \quad & \mathcal{A}' \in \Phi(\mathcal{A})\,. \end{aligned} \tag{1}$$

2.5. Attack Scenarios

Some relevant attack scenarios that can be formalized according to our framework are briefly sketched here, also mentioning related work. For the sake of space, we do not provide a thorough formulation of all these attacks. However, this can be obtained similarly to the formulation of poisoning attacks given in the next section.

Evasion attacks. Under this setting, the attacker's goal is to manipulate malicious data at test time to evade detection, as in the recent attacks envisioned by Li & Vorobeychik (2014); Wang et al. (2014) (*i.e.*, an integrity, indiscriminate violation with exploratory influence). Although evasion does not affect feature selection directly, the aforementioned works have shown that selecting features without taking into account the adversarial presence may lead one to design much more vulnerable systems. Thus, evasion attacks should be considered as a potential scenario to explore vulnerabilities of classifiers learnt on reduced feature sets, and to properly design more secure, adversary-aware feature selection algorithms.

Poisoning (integrity) attacks. Here, we envisage another scenario in which the attacker tampers with the training data to enforce selection of a feature subset that will facilitate classifier evasion at test time (*i.e.*, an integrity, targeted violation with causative influence). For instance, an attacker may craft poisoning samples to enforce selection of a given subset of features, whose values are easily changed with trivial manipulations to the malicious data at test time.

Poisoning (availability) attacks. Here the attacker aims to inject well-crafted poisoning points into the training data to maximally compromise the output of the feature selection algorithm, or directly of the learning algorithm (Mei & Zhu, 2015; Biggio et al., 2012; Rubinstein et al., 2009; Nelson et al., 2008) (*i.e.*, an availability, indiscriminate violation with causative influence). This attack scenario is formalized in detail according to our framework in the next section, to target embedded feature selection algorithms.

3. Poisoning Embedded Feature Selection

We report now a detailed case study on poisoning (availability) attacks against embedded feature selection algorithms, including LASSO, ridge regression, and the elastic net. These algorithms perform feature selection by learning a linear function $f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} + b$ that minimizes the trade-off between a loss function $\ell\left(y, f(\boldsymbol{x})\right)$ computed on the training data $\mathcal{D}$ and a regularization term $\Omega(\boldsymbol{w})$. The selection criterion $\mathcal{L}$ can be thus generally expressed as:

$$\min_{\boldsymbol{w}, b} \; \mathcal{L} = \frac{1}{n} \sum_{i=1}^{n} \ell\left(y_i, f(\boldsymbol{x}_i)\right) + \lambda \Omega(\boldsymbol{w})\,, \tag{2}$$

where λ is the trade-off parameter.[3] The quadratic loss $\ell(y, f(\boldsymbol{x})) = \frac{1}{2}\left(f(\boldsymbol{x}) - y\right)^2$ is used by all the three considered algorithms. As for the regularization term $\Omega(\boldsymbol{w})$, ideally, one would like to consider the ℓ_0-norm of $\boldsymbol{w}$ to exactly select a given number of features, which however makes the problem NP-hard (Natarajan, 1995). LASSO

[3] Note that this is equivalent to minimizing the loss subject to $\Omega(\boldsymbol{w}) \leq t$, for proper choices of λ and t (Tibshirani, 1996).

uses ℓ_1 regularization, *i.e.*, $\Omega(\boldsymbol{w}) = \|\boldsymbol{w}\|_1$, yielding the tighter convex relaxation to the ideal problem formulation. Ridge regression uses ℓ_2 regularization, $\Omega(\boldsymbol{w}) = \frac{1}{2}\|\boldsymbol{w}\|_2^2$. The elastic net is a hybrid approach between the aforementioned ones, as it exploits a convex combination of ℓ_1 and ℓ_2 regularization, *i.e.*, $\Omega(\boldsymbol{w}) = \rho\|\boldsymbol{w}\|_1 + (1-\rho)\frac{1}{2}\|\boldsymbol{w}\|_2^2$, where $\rho \in (0,1)$. Eventually, if one is given a maximum number of features $k < d$ to be selected, the ones corresponding to the first k feature weights sorted in descending order of their absolute values can be thus retained.

In the considered setting, the **attacker's goal** is to maximally increase the classification error of these algorithms, by enforcing the selection of a wrong subset of features. As for the **attacker's knowledge**, we consider both PK and LK as discussed in Sect. 2.2. In the sequel, we consider LK attacks on the surrogate data $\hat{\mathcal{D}}$, since for PK attacks we can simply set $\hat{\mathcal{D}} = \mathcal{D}$. The **attacker's capability** amounts to injecting a maximum number of poisoning points into the training set. To estimate the classification error, the attacker can evaluate the same criterion $\mathcal{L}$ used by the embedded feature selection algorithm, on her available training set $\hat{\mathcal{D}}$, excluding the attack points (as they will not be part of the test data). The attack samples are thus kept outside from the empirical loss computation of the attacker, while they are clearly taken into account by the learning algorithm. Assuming that a single attack point $\boldsymbol{x}_c$ is added by the attacker, the **attack strategy** can be thus formulated as:

$$\max_{\boldsymbol{x}_c} \mathcal{W} = \frac{1}{m}\sum_{j=1}^{m} \ell\left(\hat{y}_j, f(\hat{\boldsymbol{x}}_j)\right) + \lambda\Omega(\boldsymbol{w}) \qquad (3)$$

where it is worth remarking that f is learnt by minimizing $\mathcal{L}(\hat{\mathcal{D}} \cup \{\boldsymbol{x}_c\})$ (Eq. 2), and thus depends on the attack point $\boldsymbol{x}_c$, as well as the corresponding $\boldsymbol{w}$ and b. The attacker's objective $\mathcal{W}$ (Eq. 3) can be thus optimized by iteratively modifying $\boldsymbol{x}_c$ with a (sub)gradient-ascent algorithm, in which, at each step, the solution $\boldsymbol{w}, b$ is updated by minimizing $\mathcal{L}(\hat{\mathcal{D}} \cup \{\boldsymbol{x}_c\})$, *i.e.*, simulating the behavior of the feature selection algorithm on the poisoned data. Note that the parameters $\boldsymbol{w}, b$ estimated by the attacker are not generally the same ones estimated by the targeted algorithm. The latter will be indeed estimated by minimizing $\mathcal{L}(\mathcal{D} \cup \{\boldsymbol{x}_c\})$.

Gradient Computation. By calculating the partial derivative of Eq. (3) with respect to $\boldsymbol{x}_c$, and substituting $\ell(y, f(\boldsymbol{x}))$ and f with their expressions, one yields:

$$\frac{\partial \mathcal{W}}{\partial \boldsymbol{x}_c} = \frac{1}{m}\sum_{j=1}^{m}\left(f(\hat{\boldsymbol{x}}_j) - \hat{y}_j\right)\left(\hat{\boldsymbol{x}}_j^\top \frac{\partial \boldsymbol{w}}{\partial \boldsymbol{x}_c} + \frac{\partial b}{\partial \boldsymbol{x}_c}\right) + \lambda \boldsymbol{r}\frac{\partial \boldsymbol{w}}{\partial \boldsymbol{x}_c}\,, \qquad (4)$$

where, for notational convenience, we set $\boldsymbol{r} = \frac{\partial \Omega}{\partial \boldsymbol{w}}$. Note that $\boldsymbol{r} = \mathrm{sub}(\boldsymbol{w})$ for LASSO, $\boldsymbol{r} = \boldsymbol{w}$ for ridge regression, and $\boldsymbol{r} = \rho\,\mathrm{sub}(\boldsymbol{w}) + (1-\rho)\boldsymbol{w}$ for the elastic net, being $\mathrm{sub}(\boldsymbol{w})$ the subgradient of the ℓ_1-norm, *i.e.*, a vector whose k^{th} component equals $+1$ (-1) if $w^k > 0$ ($w^k < 0$), and any value in $[-1, +1]$ if $w^k = 0$. As the subgradient is not uniquely determined, a large set of possible ascent directions should be explored, dramatically increasing the computational complexity of the attack algorithm. Further, computing $\frac{\partial \boldsymbol{w}}{\partial \boldsymbol{x}_c}$ and $\frac{\partial b}{\partial \boldsymbol{x}_c}$ requires us to predict how the solution $\boldsymbol{w}, b$ changes while the attack point $\boldsymbol{x}_c$ is modified.

To overcome these issues, as in Cauwenberghs & Poggio (2000); Biggio et al. (2012), we assume that the Karush-Kuhn-Tucker (KKT) conditions under perturbation of the attack point $\boldsymbol{x}_c$ remain satisfied, *i.e.*, we adjust the solution to remain at the optimum. At optimality, the KKT conditions for Problem (2) with quadratic loss and linear f, are:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}}^\top = \frac{1}{n}\sum_{i=1}^{n}\left(f(\hat{\boldsymbol{x}}_i) - \hat{y}_i\right)\hat{\boldsymbol{x}}_i + \lambda \boldsymbol{r}^\top = \boldsymbol{0}\,, \qquad (5)$$

$$\frac{\partial \mathcal{L}}{\partial b} = \frac{1}{n}\sum_{i=1}^{n}\left(f(\hat{\boldsymbol{x}}_i) - \hat{y}_i\right) = 0\,, \qquad (6)$$

where we transposed the first equation to have a column vector, and keep the following derivation consistent. If $\mathcal{L}$ is convex but *not differentiable* (*e.g.*, when using ℓ_1 regularization), one may express these conditions using subgradients. In this case, at optimality a necessary and sufficient condition is that at least one of the subgradients of the objective is null (Boyd & Vandenberghe, 2004). In our case, at optimality, the subgradient is uniquely determined from Eq. (5) as $\boldsymbol{r} = -\frac{1}{\lambda}\frac{1}{n}\sum_{i=1}^{n}\left(f(\hat{\boldsymbol{x}}_i) - \hat{y}_i\right)\hat{\boldsymbol{x}}_i^\top$. This allows us to drastically reduce the complexity of the attack algorithm, as we are not required to explore all possible subgradient ascent paths for Eq. (4), but just the one corresponding to the optimal solution.

Let us assume that the optimality conditions given by Eqs. (5)-(6) remain valid under the perturbation of $\boldsymbol{x}_c$. We can thus set their derivatives with respect to $\boldsymbol{x}_c$ to zero. After deriving and re-arranging in matrix form, one obtains:

$$\begin{bmatrix} \boldsymbol{\Sigma} + \lambda \boldsymbol{v} & \boldsymbol{\mu} \\ \boldsymbol{\mu}^\top & 1 \end{bmatrix} \begin{bmatrix} \frac{\partial \boldsymbol{w}}{\partial \boldsymbol{x}_c} \\ \frac{\partial b}{\partial \boldsymbol{x}_c} \end{bmatrix} = -\frac{1}{n}\begin{bmatrix} \mathbf{M} \\ \boldsymbol{w}^\top \end{bmatrix}\,, \qquad (7)$$

where $\boldsymbol{\Sigma} = \frac{1}{n}\sum_i \hat{\boldsymbol{x}}_i\hat{\boldsymbol{x}}_i^\top$, $\boldsymbol{\mu} = \frac{1}{n}\sum_i \hat{\boldsymbol{x}}_i$, and $\mathbf{M} = \boldsymbol{x}_c\boldsymbol{w}^\top + \left(f(\boldsymbol{x}_c) - y_c\right)\mathbb{I}$. The term $\boldsymbol{v}$ yields zero for LASSO, the identity matrix $\mathbb{I}$ for ridge, and $(1-\rho)\mathbb{I}$ for the elastic net.

The derivatives $\frac{\partial \boldsymbol{w}}{\partial \boldsymbol{x}_c}$ and $\frac{\partial b}{\partial \boldsymbol{x}_c}$ can be finally obtained by solving the linear system given by Eq. (7), and then substituted into Eq. (4) to compute the final gradient.

Poisoning Feature Selection Algorithm. The complete poisoning attack algorithm is given as Algorithm 1, and an exemplary run on a bi-dimensional dataset is reported in Fig. 1. To optimize the attack with respect to multiple attack points, we choose to iteratively adjust one attack point

Algorithm 1 Poisoning Embedded Feature Selection

Input: $\hat{\mathcal{D}}$, the (surrogate) training data; $\{\boldsymbol{x}_c^{(0)}, y_c\}_{c=1}^q$, the q initial attack points with (given) labels; $\beta \in (0,1)$; and σ, ε, two small positive constants.
Output: $\{\boldsymbol{x}_c\}_{c=1}^q$, the final attack points.

$p \leftarrow 0$
repeat
  for $c = 1, \ldots, q$ **do**
    $\{\boldsymbol{w}, b\} \leftarrow$ learn the classifier on $\hat{\mathcal{D}} \cup \{\boldsymbol{x}_c^{(p)}\}_{c=1}^q$.
    Compute $\nabla\mathcal{W} = \frac{\partial \mathcal{W}(\boldsymbol{x}_c^{(p)})}{\partial \boldsymbol{x}_c}$ according to Eq. (4).
    Set $\boldsymbol{d} = \Pi_{\mathcal{B}}\left(x_c^{(p)} + \nabla\mathcal{W}\right) - x_c^{(p)}$ and $k \leftarrow 0$.
    repeat {line search to set the gradient step η}
      Set $\eta \leftarrow \beta^k$ and $k \leftarrow k + 1$
      $\boldsymbol{x}_c^{(p+1)} \leftarrow \boldsymbol{x}_c^{(p)} + \eta \boldsymbol{d}$
    until $\mathcal{W}(\boldsymbol{x}_c^{(p+1)}) \leq \mathcal{W}(\boldsymbol{x}_c^{(p)}) - \sigma\eta\|\boldsymbol{d}\|^2$
  end for
  $p \leftarrow p + 1$
until $|\mathcal{W}(\{\boldsymbol{x}_c^{(p)}\}_{c=1}^q) - \mathcal{W}(\{\boldsymbol{x}_c^{(p-1)}\}_{c=1}^q)| < \varepsilon$
return: $\{\boldsymbol{x}_c\}_{c=1}^q = \{\boldsymbol{x}_c^{(p)}\}_{c=1}^q$

at a time, while updating the current solution $\boldsymbol{w}, b$ at each step (this can be efficiently done using the previous solution $\boldsymbol{w}, b$ as a warm start). This gives the attack much more flexibility than a greedy strategy where points are added one at a time, and never modified after insertion. We also introduce a projection operator $\Pi_{\mathcal{B}}(\boldsymbol{x})$ to project $\boldsymbol{x}$ onto the feasible domain $\mathcal{B}$; *e.g.*, if features are normalized in $[0,1]$, one may consider $\mathcal{B}$ as the corresponding box-constrained domain. This enables us to define a feasible descent direction $\boldsymbol{d}$ within the given domain $\mathcal{B}$, and perform a simple line search to set the gradient step size η.

Descent in Discrete Spaces. If feature values are discrete, it is not possible to follow the gradient-descent direction exactly, as it may map the given sample to a set of non-admissible feature values. It can be however exploited as a search heuristic. Starting from the current sample, one may generate a set of candidate neighbors by perturbing only those features of the current sample which correspond to the maximum absolute values of the gradient, one at a time, in the correct direction. Eventually, one should update the current sample to the neighbor that attained the maximum value of $\mathcal{W}$, and iterate until convergence.

4. Experiments

In this section, we consider an application example involving the detection of malware in PDF files, *i.e.*, one among the most recent and relevant threats in computer security (IBM). The underlying reason is that PDF files are excellent carriers for malicious code, due to the flexibility

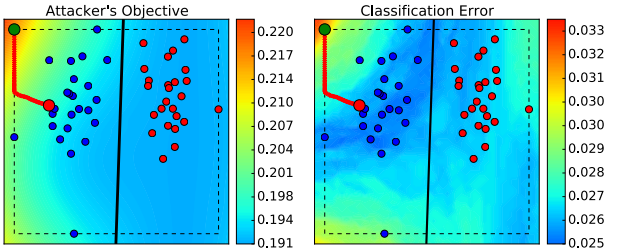

Figure 1. Poisoning LASSO. Red and blue points are the positive ($y = +1$) and negative ($y = -1$) training data $\mathcal{D}$. The decision boundary at $f(\boldsymbol{x}) = 0$ (for $\lambda = 0.01$, in the absence of attack) is shown as a solid black line. The solid red line highlights the path followed by the attack point $\boldsymbol{x}_c$ (*i.e.*, the magnified red point) towards a maximum of $\mathcal{W}(\boldsymbol{x}_c)$ (shown in colors in the *left* plot), which also corresponds to a maximum of the classification error (*right* plot). A box constraint is also considered (dashed black square) to bound the feasible domain (*i.e.*, the attack space).

of their logical structure, which allows embedding of several kinds of resources, including `Flash`, `JavaScript` and even executable code. Resources are simply embedded by specifying their type with *keywords*, and inserting the corresponding content in *data streams*. For instance, an embedded resource in a PDF may look like:

```
13 0 obj << /Kids [ 1 0 R 11 0 R ]
/Type /Page ... >> end obj
```

where keywords are highlighted in bold face. Recent work has promoted the use of machine learning to detect malicious PDF files (apart from legitimate PDFs), based on the analysis of their logical structure and, in particular, of the present keywords rather than the content of data streams (Smutz & Stavrou, 2012; Maiorca et al., 2012; 2013; Šrndić & Laskov, 2013).

Experimental setup. In our experiments, we exploit the feature representation proposed by Maiorca et al. (2012), where each feature simply denotes the number of occurrences of a given keyword in the PDF file. We collected 5993 recent malware samples from the *Contagio* dataset,[4] and 5951 benign samples from the web. As a preliminary step, following the procedure described by Maiorca et al. (2012), we extracted keywords from the first 1,000 samples in chronological order. The resulting 114 keywords were used as our initial feature set $\mathcal{X}$. We then randomly sampled five pairs of training and test sets from the remaining data, respectively consisting of 300 and 5,000 samples, to average the final results. To simulate LK attacks (Sect. 2.2), we also sampled an additional set of five training sets (to serve as $\hat{\mathcal{D}}$) consisting of 300 samples each. We normalized each feature between 0 and 1 by bounding the maximum keyword count to 20, and dividing each feature value by the same value. This value was selected

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

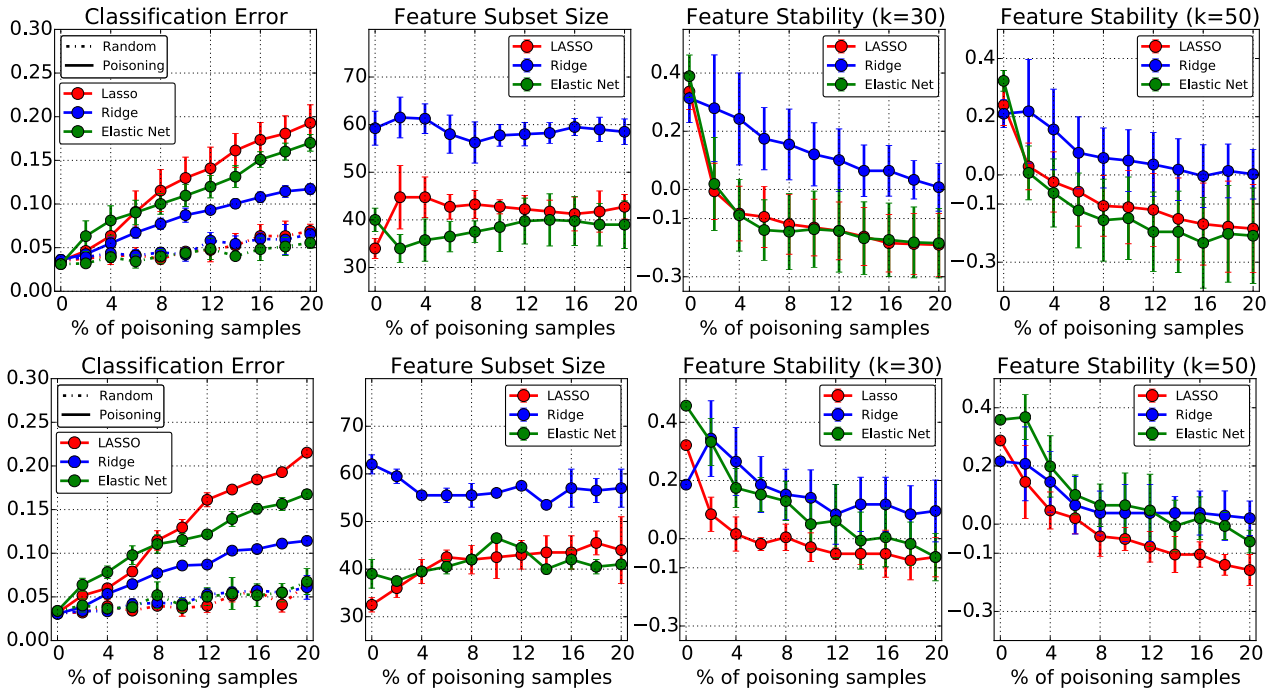

Figure 2. Results on PDF malware detection, for PK (*top row*) and LK (*bottom row*) poisoning attacks against LASSO, ridge, and elastic net, in terms of classification error (*first column*), number of automatically selected features (*second column*), and stability of the top $k = 30$ (*third column*) and $k = 50$ (*fourth column*) selected features, against an increasing percentage of injected poisoning samples. For comparison, we also report the classification error attained by all methods against random label-flip attacks (*first column*). All the reported values are averaged over five independent runs, and error bars correspond to their standard deviation.

to restrict the attacker's capability of manipulating data to a large extent, without affecting generalization accuracy in the absence of attack.[5] We evaluate the impact of poisoning against LASSO, ridge and elastic net. We first set $\rho = 0.5$ for the elastic net, and then optimized the regularization parameter λ for all methods by retaining the best value over the entire regularization path (Friedman et al., 2010; Pedregosa et al., 2011). We evaluate our results by reporting the classification error as a function of the percentage of injected poisoning samples, which was increased from 0% to 20% (where 20% corresponds to adding 75 poisoning samples to the initial data). Furthermore, to understand how feature selection and ranking are affected by the attack, we also evaluate the consistency index originally defined by Kuncheva (2007) to evaluate the stability of feature selection under random perturbations of the training data.

Kuncheva's Stability Index. Given two feature subsets $A, B \subseteq \mathcal{X}$, with $\|A\| = \|B\| = k$, $r = \|A \cap B\|$, and $0 < k < \|\mathcal{X}\| = d$, it is defined as:

$$I_C(A, B) = \frac{rd - k^2}{k(d - k)} \in [-1, +1]\,, \tag{8}$$

where positive values indicate similar sets, zero is equivalent to random selections, and negative values indicate strong anti-correlation between the feature subsets. The underlying idea of this stability index is to normalize the number of common features in the two sets (*i.e.*, the cardinality of their intersection) using a correction for chance that accounts for the average number of common features randomly selected out of k trials.

To evaluate how poisoning affects the feature selection process, we compute this index using for A a feature set selected in the absence of poisoning, and comparing it against a set B selected under attack, at different percentages of poisoning. To compare subsets of equal size k, for each method, we considered the first k features exhibiting the highest absolute weight values. As suggested by Kuncheva (2007), all the corresponding pairwise combinations of such sets were averaged, to compute the expected index value along with its standard deviation.

Experimental results. Results are reported in Fig. 2, for both the PK and LK settings. No substantial differences between these settings are highlighted in our results, meaning that the attacker can reliably construct her poisoning attacks even without having access to the training data $\mathcal{D}$, but only using surrogate data $\hat{\mathcal{D}}$. In the absence of attack (*i.e.*, at 0% poisoning), all methods exhibit reliable perfor-

[5] If no bound on the keyword count is set, an attacker may add an unconstrained number of keywords and arbitrarily influence the training process.

mances and a very small classification error. Poisoning up to 20% of the training data causes the classification error to increase of approximately 10 times, from 2% to 20% for LASSO, and slightly less for elastic net and ridge, which therefore exhibit slightly improved robustness properties against this threat. For comparison, we also considered a *random* attack that generates each attack point by randomly cloning a point in the training data and flipping its label. As one may note from plots in the first column of Fig. 2, our poisoning strategy is clearly much more effective.

Besides the impact of poisoning on the classification error, the most significant result of our analysis is related to the impact of poisoning on feature selection. In particular, from Fig. 2 (third and fourth column), one can immediately note that the (averaged) stability index (Eq. 8) quickly decreases to zero (especially for LASSO and the elastic net) even under a very small fraction of poisoning samples. This means that, in the presence of few poisoning samples, the feature selection algorithm performs as a *random* feature selector in the absence of attack. In other words, the attacker can almost arbitrarily control feature selection. Finally, it is worth remarking that, among the considered methods, ridge exhibited higher robustness under attack. We argue that a possible reason besides selecting larger feature subsets (see plots in the second column of Fig. 2) is that feature weights are more evenly spread among the features, reducing the impact of each training point on the embedded feature selection process. We discuss in detail the importance of selecting larger feature subsets against poisoning attacks in the next section.

5. Discussion

We think that our work gives a two-fold contribution to the state of the art. The first contribution is to the field of adversarial machine learning. We are the first to propose a framework to evaluate the vulnerability of feature selection algorithms and to use it to analyze poisoning attacks against popular embedded feature selection methods, including LASSO, ridge regression, and the elastic net. The second contribution concerns the robustness properties of ℓ_1 regularization. Despite our results are seemingly in contrast with the claimed robustness of ℓ_1 regularization, it is worth remarking that ℓ_1 regularization is robust against *non-adversarial* data perturbations; in particular, it aims to reduce the variance component of the error by selecting smaller feature subsets, at the expense of a higher bias. Conversely, poisoning attacks induce a systematic *bias* into the training set. This means that an attacker may more easily compromise feature selection algorithms that promote *sparsity* by increasing the bias component of the error. The fact that ℓ_1 regularization may worsen performance under attack is also confirmed by Li & Vorobeychik (2014), although in the context of evasion attacks. Even if the underlying attack scenario is different, also evasion attacks induce a specific bias in the manipulation of data, and are thus more effective against *sparse* algorithms that exploit smaller feature sets to make decisions.

6. Conclusions and Future Work

Due to the massive use of data-driven technologies, the variability and sophistication of cyberthreats and attacks have tremendously increased. In response to this phenomenon, machine learning has been widely applied in security settings. However, these techniques have not been originally designed to cope with intelligent attackers, and are thus vulnerable to well-crafted attacks. While attacks against learning and clustering algorithms have been widely analyzed in previous work (Barreno et al., 2006; Huang et al., 2011; Brückner et al., 2012; Biggio et al., 2012; 2013b; 2014), only few attacks targeting feature selection algorithms have been recently considered (Li & Vorobeychik, 2014; Mei & Zhu, 2015; Zhang et al., 2015).

In this work, we have provided a framework that allows one to model potential attack scenarios against feature selection algorithms in a consistent way, making clear assumptions on the attacker's goal, knowledge and capabilities. We have exploited this framework to characterize the relevant threat of poisoning attacks against feature selection algorithms, and reported a detailed case study on the vulnerability of popular embedded methods (LASSO, ridge, and elastic net) against these attack. Our security analysis on a real-world security application involving PDF malware detection has shown that attackers can completely control the selection of reduced feature subsets even by only injecting a small fraction of poisoning training points, especially if *sparsity* is enforced by the feature selection algorithm.

This demands for the engineering of *secure* feature selection algorithms against poisoning attacks. To this end, one may follow the intuition behind the recently-proposed feature selection algorithms to contrast evasion attacks, *i.e.*, to model interactions between the attacker and the feature selection algorithm (Li & Vorobeychik, 2014; Wang et al., 2014; Zhang et al., 2015). Recent work on robust LASSO and robust regression may be another interesting future direction to implement secure feature selection against poisoning (Nasrabadi et al., 2011; Nguyen & Tran, 2013). From a more theoretical perspective, it may be of interest to analyze: (i) the impact of poisoning attacks on feature selection in relation to the ratio between the training set size and the dimensionality of the feature set; and (ii) the impact of poisoning and evasion on the bias-variance decomposition of the mean squared error. These aspects may reveal additional interesting insights also for designing secure feature selection procedures.

Acknowledgments

This work has been partly supported by the project CRP-59872 funded by Regione Autonoma della Sardegna, L.R. 7/2007, Bando 2012.

References

Barreno, Marco, Nelson, Blaine, Sears, Russell, Joseph, Anthony D., and Tygar, J. D. Can machine learning be secure? In *ASIACCS*, pp. 16–25, NY, USA, 2006. ACM.

Barreno, Marco, Nelson, Blaine, Joseph, Anthony, and Tygar, J. The security of machine learning. *Machine Learning*, 81:121–148, 2010.

Biggio, B., Corona, I., Maiorca, D., Nelson, B., Šrndić, N., Laskov, P., Giacinto, G., and Roli, F. Evasion attacks against machine learning at test time. In Blockeel, H. et al. (eds.), *ECML PKDD, Part III*, vol. 8190 of *LNCS*, pp. 387–402. Springer Berlin Heidelberg, 2013a.

Biggio, Battista, Nelson, Blaine, and Laskov, Pavel. Poisoning attacks against support vector machines. In Langford, J. and Pineau, J. (eds.), *29th Int'l Conf. on Machine Learning*, pp. 1807–1814. Omnipress, 2012.

Biggio, Battista, Pillai, Ignazio, Bulò, Samuel Rota, Ariu, Davide, Pelillo, Marcello, and Roli, Fabio. Is data clustering in adversarial settings secure? In *ACM Workshop on Artificial Intell. and Sec.*, pp. 87–98, 2013b. ACM.

Biggio, Battista, Fumera, Giorgio, and Roli, Fabio. Security evaluation of pattern classifiers under attack. *IEEE Trans. Knowl. and Data Eng.*, 26(4):984–996, 2014.

Boyd, Stephen and Vandenberghe, Lieven. *Convex Optimization*. Cambridge University Press, 2004.

Brown, Gavin, Pocock, Adam, Zhao, Ming-Jie, and Luján, Mikel. Conditional likelihood maximisation: A unifying framework for information theoretic feature selection. *J. Mach. Learn. Res.*, 13:27–66, 2012.

Brückner, Michael, Kanzow, Christian, and Scheffer, Tobias. Static prediction games for adversarial learning problems. *J. Mach. Learn. Res.*, 13:2617–2654, 2012.

Cauwenberghs, Gert and Poggio, Tomaso. Incremental and decremental support vector machine learning. In Leen, T. K. et al. (eds.), *NIPS*, pp. 409–415. MIT Press, 2000.

Dalvi, Nilesh, Domingos, Pedro, Mausam, Sanghai, Sumit, and Verma, Deepak. Adversarial classification. In *Knowl. Disc. and Data Mining*, pp. 99–108, 2004.

Friedman, Jerome H., Hastie, Trevor, and Tibshirani, Rob. Regularization paths for generalized linear models via coordinate descent. *J. Stat. Softw.*, 33(1):1–22, 2 2010.

Globerson, Amir and Roweis, Sam T. Nightmare at test time: robust learning by feature deletion. In Cohen, W. and Moore, A. (eds.), *23rd Int'l Conf. on Machine Learning*, volume 148, pp. 353–360. ACM, 2006.

Graham, Paul. A plan for spam, 2002. URL http://paulgraham.com/spam.html.

Hoerl, A. E. and Kennard, R. W. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, Feb. 1970.

Huang, L., Joseph, A. D., Nelson, B., Rubinstein, B., and Tygar, J. D. Adversarial machine learning. In *ACM Workshop on Artificial Intell. and Sec.*, pp. 43–57, Chicago, IL, USA, 2011.

IBM. The X-Force threat and risk report. URL http://www-03.ibm.com/security/xforce/.

Joseph, Anthony D., Laskov, Pavel, Roli, Fabio, Tygar, J. Doug, and Nelson, Blaine. Machine Learning Methods for Computer Security (Dagstuhl Perspectives Workshop 12371). *Dagstuhl Manifestos*, 3(1):1–30, 2013.

Kloft, M. and Laskov, P. Online anomaly detection under adversarial impact. In *13th Int'l Conf. on Artificial Intell. and Statistics*, pp. 405–412, 2010.

Kuncheva, Ludmila I. A stability index for feature selection. In *Proc. 25th IASTED Int'l Multi-Conf.: Artificial Intell. and Applications*, pp. 390–395, 2007. ACTA Press.

Li, Bo and Vorobeychik, Yevgeniy. Feature cross-substitution in adversarial classification. In Ghahramani, Z. et al. (eds.), *NIPS 27*, pp. 2087–2095. Curran Associates, Inc., 2014.

Lowd, Daniel and Meek, Christopher. Adversarial learning. In *Proc. 11th ACM SIGKDD Int'l Conf. on Knowl. Disc. and Data Mining*, pp. 641–647, 2005. ACM Press.

Maiorca, Davide, Giacinto, Giorgio, and Corona, Igino. A pattern recognition system for malicious PDF files detection. In Perner, P. (ed.), *MLDM*, vol. 7376 of *LNCS*, pp. 510–524. Springer Berlin Heidelberg, 2012.

Maiorca, Davide, Corona, Igino, and Giacinto, Giorgio. Looking at the bag is not enough to find the bomb: an evasion of structural methods for malicious pdf files detection. In *ASIACCS*, pp. 119–130, 2013. ACM.

McCallum, A. and Nigam, K. A comparison of event models for naive bayes text classification. In *Proc. AAAI Workshop Learn. for Text Categoriz.*, pp. 41–48, 1998.

Mei, Shike and Zhu, Xiaojin. The security of latent dirichlet allocation. In *18th Int'l Conf. on Artificial Intell. and Statistics*, 2015.

Nasrabadi, Nasser M., Tran, Trac D., and Nguyen, Nam. Robust lasso with missing and grossly corrupted observations. In Shawe-Taylor, J. et al. (eds.), *NIPS 24*, pp. 1881–1889. Curran Associates, Inc., 2011.

Natarajan, B. K. Sparse approximate solutions to linear systems. *SIAM J. Comput.*, 24(2):227–234, April 1995.

Nelson, Blaine, Barreno, Marco, Chi, Fuching Jack, Joseph, Anthony D., Rubinstein, Benjamin I. P., Saini, Udam, Sutton, Charles, Tygar, J. D., and Xia, Kai. Exploiting machine learning to subvert your spam filter. In *1st Workshop on Large-Scale Exploits and Emergent Threats*, pp. 1–9, 2008. USENIX Association.

Nguyen, N.H. and Tran, T.D. Robust lasso with missing and grossly corrupted observations. *IEEE Trans. Inf. Theor.*, 59(4):2036–2058, 2013.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.*, 12:2825–2830, 2011.

Robinson, Gary. A statistical approach to the spam problem. *Linux J.*, 2003(107):3, 2003.

Rubinstein, Benjamin I.P., Nelson, Blaine, Huang, Ling, Joseph, Anthony D., Lau, Shing-hon, Rao, Satish, Taft, Nina, and Tygar, J. D. Antidote: understanding and defending against poisoning of anomaly detectors. In *9th Internet Meas. Conf.*, pp. 1–14, 2009. ACM.

Sahami, M., Dumais, S., Heckerman, D., and Horvitz, E. A bayesian approach to filtering junk e-mail. *AAAI Technical Report WS-98-05, Madison, Wisconsin*, 1998.

Smutz, Charles and Stavrou, Angelos. Malicious PDF detection using metadata and structural features. In *28th Annual Computer Security Applications Conf.*, pp. 239–248, 2012. ACM.

Spitzner, Lance. *Honeypots: Tracking Hackers*. Addison-Wesley Professional, 2002.

Teo, Choon Hui, Globerson, Amir, Roweis, Sam, and Smola, Alex. Convex learning with invariances. In *NIPS 20*, pp. 1489–1496. MIT Press, 2008.

Tibshirani, R. Regression shrinkage and selection via the lasso. *J. Royal Stat. Soc. (Ser. B)*, 58:267–288, 1996.

Šrndić, Nedim and Laskov, Pavel. Detection of malicious pdf files based on hierarchical document structure. In *NDSS*. The Internet Society, 2013.

Wang, Fei, Liu, Wei, and Chawla, Sanjay. On sparse feature attacks in adversarial learning. In *IEEE Int'l Conf. on Data Mining (ICDM)*, pp. 1013–1018. IEEE, 2014.

Zhang, F., Chan, P.P.K., Biggio, B., Yeung, D.S., and Roli, F. Adversarial feature selection against evasion attacks. *IEEE Trans. on Cybernetics*, PP(99):1–1, 2015.

Zou, Hui and Hastie, Trevor. Regularization and variable selection via the elastic net. *J. Royal Stat. Soc. (Ser. B)*, 67(2):301–320, 2005.