Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 8 September 2020 doi:10.20944/preprints202009.0184.v1

Article

Perspectives on Adversarial Classification

David Rios Insua [1], Roi Naveiro [2] and Victor Gallego [3]

[1] School of Management, University of Shanghai for Science and Technology and ICMAT-CSIC; david.rios@icmat.es
[2] ICMAT-CSIC; roi.naveiro@icmat.es
[3] ICMAT-CSIC; victor.gallego@icmat.es
* Correspondence: david.rios@icmat.es

Version September 6, 2020 submitted to Mathematics

Abstract: Adversarial Classification (AC) is a major subfield within the increasingly important domain of adversarial machine learning (AML). Most approaches to AC so far have followed a classical game-theoretic framework. This requires unrealistic common knowledge conditions untenable in the security settings typical of the AML realm. After reviewing such approaches, we present alternative perspectives on AC based on Adversarial Risk Analysis.

Keywords: Classification; Adversarial Machine Learning; Security; Robustness; Adversarial Risk Analysis

1. Introduction

Statistical classification is a major research area in security and cybersecurity. Applications abound, including fraud detection [Bolton and Hand 2002]; phishing detection [El Aassal *et al.* 2020] or terrorism [Simanjuntak *et al.* 2010]. Moreover, an increasing number of processes are being automated through classification algorithms, being essential that these are robust to trust key operations based on their output. State-of-the-art classifiers perform extraordinarily well on standard data, but they have been shown to be vulnerable to adversarial examples, that is, data instances specifically targeted at fooling their underlying algorithms. Comiter [2019] provides an excellent introduction from a policy perspective, pointing out the potentially enormous security impacts that such attacks may have over systems for filter content, predictive policing or autonomous driving, to name but a few.

Most research in classification has focused on obtaining more accurate algorithms, largely ignoring the eventual presence of adversaries who actively manipulate data to fool the classifier in pursue of a benefit. Consider the example of spam detection: as classification algorithms are incorporated to such task, spammers learn how to evade them. Thus, rather than sending their spam messages in standard language, they slightly modify spam words (frequent in spam messages but not so much in legitimate ones), misspelling or changing them with synonyms; or they add good words (frequent in legitimate emails but not in spam ones) to fool the detection system.

Consequently, classification algorithms in critical AI-based systems must be robust against adversarial data manipulations. To this end, they have to take into account possible modifications of input data due to adversaries. The subfield of statistical classification that seeks for algorithms with robust behavior against adversarial perturbations is known as adversarial classification (AC) and was pioneered by Dalvi *et al.* [2004]. Stemming from their work, the prevailing paradigm used to model the confrontation between classification systems and adversaries has been game theory, see recent reviews by Biggio and Roli [2018] and Zhou *et al.* [2018]. This entails well-known common knowledge hypothesis [Antos and Pfeffer 2010] according to which agents share information about their utilities and probabilities. From a fundamental point of view, this is clearly not sustainable in application areas such as security or cibersecurity, as participants try to hide and conceal information.

© 2020 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 8 September 2020 doi:10.20944/preprints202009.0184.v1

After reviewing key developments in game-theoretic approaches to AC in Sections 2 and 3, in Sections 4 and 5 we cover novel techniques based on Adversarial Risk Analysis (ARA) [Rios Insua *et al.* 2009], which do not assume standard common knowledge hypothesis. In thi, we unify, expand and improve upon earlier work in Naveiro *et al.* [2019] and Gallego *et al.* [2020]. Our focus will be on *binary classification* problems in face only of *exploratory attacks*, defined to have influence over operational data but not over training ones. In addition, we restrict our attention to attacks not affecting innocent instances, denominated *integrity-violation attacks*, the usual context in most security scenarios. Moreover, attacks will be assumed to be *deterministic*, in that we can predict for sure the results of their application over a given instance. Huang *et al.* [2011] and Barreno *et al.* [2006] provide taxonomies of attacks against classifiers. We first consider, Section 4, approaches in which learning about the adversary is performed in the operational phase, studying how to robustify generative and discriminative classifiers against attacks. In certain applications, these could be very demanding from a computational perspective. For those cases, we present, Section 5, an approach in which adversarial aspects are incorporated in the training phase to robustify the classifier.

2. Attacking classification algorithms

2.1. Binary classification algorithms

In binary classification settings, an agent that we call classifier (C, she), may receive two types of objects denoted as malicious ($y = y_1$) or innocent ($y = y_2$). Objects have features $x \in \mathbb{R}^d$ whose distribution informs about their type y. Classification approaches can be typically broken down into two separate stages [Bishop 2006]. The first one is a *training* stage to learn the distribution $p_C(y|x)$, modelling the classifier's beliefs about the instance type y given its features x. Frequently, a distinction is introduced between *generative* and *discriminative* models. In the first case, models $p_C(x|y)$ and $p_C(y)$ are learnt from training data and, based on them, $p_C(y|x)$ is deduced through Bayes formula. Typical examples include Naive Bayes [Rish *et al.* 2001] and (conditional) variational autoencoders [Kingma *et al.* 2014]. In discriminative cases, $p_C(y|x)$ is directly learnt from data. An important class of methods uses for this a parameterized function $f_\beta : \mathbb{R}^d \to \mathbb{R}^2$ so that the prediction is given through $p_C(y|x, \beta) = \text{softmax}(f_\beta(x))[y]$. When $f_\beta(x) = \beta' x$, we recover the logistic regression model [McCullagh and Nelder 1989]; if we take f_β as a sequence of linear transformations alternating certain non-linear activation functions, we obtain a feed-forward neural network [Bishop 2006]. Inference depends then on the underlying methodology adopted.

- In classical approaches, training data $\mathcal{D}$ is typically used to construct a maximum likelihood estimate $\hat{\beta}$, and $p_C(y|\hat{\beta}, x)$ is employed to classify. Parametric differentiable models are amenable to training with *stochastic gradient descent* (SGD) using a minibatch of samples at each iteration. This facilitates, e.g., training deep neural networks with large amounts of high-dimensional data as with images or text data [Goodfellow *et al.* 2016].
- In Bayesian approaches, a prior $p_C(\beta)$ is used to compute the posterior $p_C(\beta|\mathcal{D})$ and the predictive distribution

$$p_C(y|x, \mathcal{D}) = \int p_C(y|\beta, x) p_C(\beta|\mathcal{D}) d\beta \tag{1}$$

  is used to classify. In complex environments, given current technology, we are sometimes only able to approximate the posterior mode $\hat{\beta}$ and use $p_C(y|\hat{\beta}, x)$.

In any case, and whatever the inference approach we adopt, we shall use the notation $p_C(y|x)$.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 8 September 2020 doi:10.20944/preprints202009.0184.v1

The second stage is *operational*. The agent makes class assignment decisions based on $p_C(y|x)$. This may be formulated through the influence diagram (ID) [Shachter 1986] in Figure 1[1]. The classifier

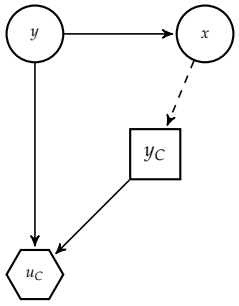

Figure 1. Classification as an Influence Diagram

guess y_c for an observed x provides her a utility $u_C(y_c, y_i)$ when the actual type is y_i. She aims at maximizing expected utility through

$$\arg\max_{y_c} \sum_{i=1}^{2} u_C(y_c, y_i) p_C(y_i|x). \tag{2}$$

An important class of utility functions is the 0-1 utility: $u_C(y_c, y_i) = \mathbb{I}(y_c = y_i)$, where $\mathbb{I}$ is the indicator function. This leads to deciding based on maximizing the predictive probability of correct classification

$$\arg\max_{y_c} \sum_{i=1}^{2} \mathbb{I}(y_c = y_i) p_C(y_i|x) = \arg\max_{y_c} p_C(y_c|x). \tag{3}$$

Note that there are classification techniques, such as those based on SVMs, that rather than breaking classification in training and operational stages, directly learn a function that maps features x into labels y. If we were to apply the subsequent methodology to this type of classifiers, we would need to produce estimates of $p_C(y|x)$ using their outputs. This can be done using calibration techniques such as Platt [1999] scaling.

2.2. Attacks to binary classification algorithms

Consider now another agent called adversary (A, he). The attacker aims at fooling C and make her err in classifying objects to attain some benefit. A applies an attack a to the features x leading to $x' = a(x)$, which is the actual observation received by C. For notational convenience, we shall sometimes write $x = a^{-1}(x')$. Upon observing x', C needs to determine the object type. The originating instance x is not observed by C. As we illustrate next, an adversary unaware classifier may incur in gross mistakes if she classifies based on features x', instead of the original ones.

Attacks to spam detection systems Consider attacks to standard classifiers used in spam detection. Experiments are carried out with the UCI Spam Data Set [Hopkins *et al.* 1999]. This set contains data about 4601 emails, out of which 39.4% are spam. For classification purposes, we represent each email through 54 binary variables indicating the presence (1) or absence (0) of 54 designated words in a dictionary. The adversary is allowed to either insert good words (frequent in legitimate emails) or remove bad ones (frequent in junk messages) from spam emails trying to make the defender misclassify them as legitimate. In our example, the adversary is allowed to modify at

[1] Square nodes describe decisions; circle nodes, uncertainties; hexagonal nodes refer to the associated utilities. Arcs pointing to decision nodes are dashed and represent information available when the corresponding decisions are made. Arcs pointing to chance and value nodes suggest conditional dependence.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 8 September 2020 doi:10.20944/preprints202009.0184.v1

most two words in a message. These were chosen using the procedure described in Gallego *et al.* [2020].

Table 1 presents the accuracy of four standard classifiers (*Naive Bayes*, *logistic regression*, *neural net* and *random forest*) based on a $0-1$ utility function, against tainted and untainted data. The neural network model is a two layer one; the logistic regression is applied with L1 regularization[2]. Means and standard deviations of accuracies are estimated via repeated hold-out validation over ten repetitions [Kim 2009].

Classifier	Acc. Unt.	Acc. Taint.
Naive Bayes	0.891 ± 0.003	0.774 ± 0.026
Logistic Reg.	0.928 ± 0.004	0.681 ± 0.009
Neural Net	0.905 ± 0.003	0.764 ± 0.007
Random Forest	0.946 ± 0.002	0.663 ± 0.006

Table 1. Accuracy comparison (with precision) of four classifiers on clean (untainted) and attacked (tainted) data.

Observe the important loss in accuracy of the four classifiers, showcasing a major degradation in performance of adversary unaware classifiers when facing attacks. △

3. Adversarial classification: game-theoretic approaches

As exemplified, an adversary unaware classifier may be fooled into issuing wrong classifications leading to severe performance deterioration. Strategies to mitigate this problem are thus needed. These may be based on building models of the attacks likely to be undertaken by the adversaries and enhancing classification algorithms to be robust against such attacks.

For this, the ID describing the classification problem (Figure 1) is augmented to incorporate adversarial decisions, leading to a bi-agent influence diagram (BAID) [Banks *et al.* 2015], Figure 2[3]. We only describe the elements that are new. First, the adversary decision is represented through node a (the chosen attack). The impact of the data transformation over x implemented by A is described through node x', the data actually observed by the classifier; the corresponding node is deterministic (double circle) as we assume deterministic attacks. Finally, the utility of A is represented with node u_A, with form $u_A(y_c, y)$, when C says y_c and the actual label is y. We assume that attack implementation has negligible costs. As before, C aims at maximizing her expected utility; A also aims at maximizing his expected utility by trying to confuse the classifier (and, consequently reducing her expected utility).

3.1. Adversarial classification. The model of Dalvi et al. [2004]

Dalvi *et al.* [2004] provided a pioneering approach to enhance classification algorithms when an adversary is present, calling it adversarial classification (AC). Because of its importance, we review it in some detail, using our notation.

The authors view AC as a game between a classifier C and an adversary A. The classifier aims at finding an optimal classification strategy against A's optimal attacking strategy. However, computing Nash equilibria [Menache and Ozdaglar 2011] in such general games becomes overly complex. Therefore, they propose the following simplified forward myopic version.

1. C first assumes that data is untainted and computes her optimal classifier through (2). They focus on a utility sensitive Naive Bayes algorithm [Elkan 2001].

---

[2] This is equivalent to performing maximum a posteriori estimation in a logistic regression model with a Laplace prior [Park and Casella 2008].

[3] Grey nodes refer to elements solely affecting A's decision; white nodes to issues solely pertaining to C's decision; striped nodes affect both agents' decisions

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 8 September 2020 doi:10.20944/preprints202009.0184.v1

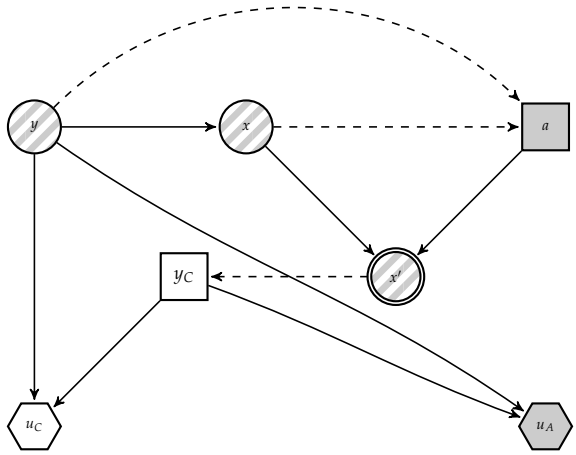

Figure 2. Adversarial classification as a Bi-agent Influence Diagram

2. Then, the authors, assuming that A has complete information about the classifier's elements (a common knowledge assumption) and that she is not aware of his presence, compute A's optimal attack. To that end, they propose solving the integer programming problem

$$\begin{aligned} &\min \left\{ \sum_{X_i \in \mathcal{X}_C} \sum_{x'_i \in \mathcal{X}_i} \mathcal{C}(x_i, x'_i) \delta_{i,x'_i} \right\} \qquad \text{s.t.} \\ &\sum_{X_i \in \mathcal{X}_C} \sum_{x'_i \in \mathcal{X}_i} \left(\log \frac{p_C(x_i|y_1)}{p_C(x_i|y_2)} - \log \frac{p_C(x'_i|y_1)}{p_C(x'_i|y_2)} \right) \delta_{i,x'_i} \geq gap(x). \end{aligned} \tag{4}$$

To wit, let $\mathcal{X}_C$ be the set of features the classifier uses for making her decision and X_i, the i-th feature, with original value $x_i \in \mathcal{X}_i$, assumed to be discrete. δ_{i,x'_i} is a binary variable adopting value 1 when feature X_i is changed from x_i to x'_i, being $\mathcal{C}(x_i, x'_i)$ the cost of such change; and 0, otherwise. Thus, the objective function in (4), expresses that the adversary tries to change features minimizing his total cost. In turn, the constraint ensures that feature modification will induce a change in the classification decision. It is easy to see that the classifier will declare an instance as malicious if

$$gap(x) = \log \frac{p_C(x|y_1)}{p_C(x|y_2)} - \log \frac{u_C(y_2, y_2) - u_C(y_1, y_2)}{u_C(y_1, y_1) - u_C(y_2, y_1)} > 0.$$

The adversary is thus interested in modifying the features of malicious instances from x to x', such that they are classified as legitimate, that is $gap(x') < 0$. A feature modification induces a change on the log odds, from $\log \frac{p_C(x|y_1)}{p_C(x|y_2)}$ to $\log \frac{p_C(x'|y_1)}{p_C(x'|y_2)}$. It is straightforward to see that, in order for the new instance to have $gap(x') < 0$, old minus new odds must surpass $gap(x)$, as reflected by the constraint in (4).

3. Subsequently, the classifier, assuming that A implements the previous attack (again a common knowledge assumption) and that the training data is untainted, deploys her optimal classifier against it: she chooses y_c maximizing $\sum_{i=1}^{2} u_C(y_c, y_i) p_C(y_i|x')$, her posterior expected utility given that she observes the possibly modified instance x'. This is equivalent to optimizing

$$u_C(y_c, y_1) p_C(x'|y_1) p_C(y_1) + u_C(y_c, y_2) p_C(x'|y_2) p_C(y_2). \tag{5}$$

Estimating $p(y_1)$ (and $p(y_2)$) is simple, based on training data, clean by assumption. In addition, as the authors assume that legitimate instances are not modified, $p_C(x'|y_2) = p_C(x|y_2)$ which can be estimated as well from training data. To estimate $p_C(x'|y_1)$, the authors appeal yet again

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 8 September 2020 doi:10.20944/preprints202009.0184.v1

to a common knowledge assumption: if the adversary receives instance x, then the classifier, who knows all its involved elements, can solve problem (4) and compute $x' = a(x)$. Thus

$$p_C(x'|y_1) = \sum_{x \in \mathcal{X}'} p_C(x|y_1) p_C(x'|x, y_1)$$

where $\mathcal{X}'$ is the set of possible instances leading to the observed one and $p_C(x'|x, y_1) = 1$ if $a(x) = x'$ and 0 otherwise.

The procedure could continue for more stages. However, Dalvi *et al.* [2004] consider sufficient to use these three.

As presented (and the authors actually stress in their paper) very strong common knowledge assumptions are made: all parameters of both players are known to each other. Although standard in game theory, such assumption is unrealistic in the security scenarios typical of AC.

3.2. Other AC game-theoretic developments

In spite of this, stemming from Dalvi *et al.* [2004], AC has been predated by game-theoretic approaches, reviewed in Biggio *et al.* [2014] or Li and Vorobeychik [2014]. Subsequent attempts have focused on analyzing attacks over classification algorithms and assessing their robustness against such attacks, under various assumptions about the adversary. Regarding these, attacks have been classified as *white box*, when the adversary knows every aspect of the defender's system including data, algorithms and the entire feature space; *black box*, that assume limited capabilities for the adversary, e.g. he is able to send membership queries to the classification system as in Lowd and Meek [2005]; and *gray box*, which are in between the previous ones, e.g. as in Zhou *et al.* [2012] where the adversary, who has no knowledge about the data and the algorithm used, seeks to push his malicious instances into innocuous ones, thus assuming that he is able to estimate such instances and has knowledge about the feature space.

Of special importance in the AC field, mainly within the deep learning community, are the so called *adversarial examples* [Goodfellow *et al.* 2014] which may be formulated in game-theoretic terms as optimal attacks to a deployed classifier, requiring, in principle, precise knowledge about the model used by the classifier. To create such examples, A finds the best attack which leads to perturbed data instances obtained from solving problem $\min_{\|\delta\| \leq \epsilon} \widehat{c}_A(h_\theta(a(x)), y)$, with $a(x) = x + \delta$, a suitable perturbation of the original data instance x; $h_\theta(x)$, the output of a predictive model with parameters θ; and $\widehat{c}_A(h_\theta(x), y)$ the adversary's cost when instance x of class y is classified as of being of class $h_\theta(x)$. This cost is usually taken to be $-\widehat{c}_D(h_\theta(x), y)$, where c_D is the defender's cost. The Fast Gradient Signed Method (FGSM) and related attacks in the literature [Vorobeichyk and Kantarcioglu 2019] assume that the attacker has precise knowledge of the underlying model and parameters of the involved classifier, debatable in most security settings.

A few methods have been proposed to robustify classification algorithms in adversarial environments. Most of them have focused on application-specific domains, as Kołcz and Teo [2009] on spam detection. Vorobeychik and Li [2014] study the impact of randomization schemes over different classifiers against adversarial attacks proposing an optimal randomization scheme as best defense. To date, there is one promising defence technique: *adversarial training* (AT) [Madry *et al.* 2018] trains the defender model using attacked samples, solving the problem

$$\min_\theta \mathbb{E}_{(x,y)\sim\mathcal{D}} \left[\max_{\|\delta_x\| \leq \epsilon} \widehat{c}_D(h_\theta(a(x)), y) \right],$$

thus minimizing the empirical risk of the model under worst case perturbations of the data $\mathcal{D}$. AT can be formulated as a zero-sum game. The inner maximization problem is solved through project gradient descent (PGD) with iterations $x_{t+1} = \Pi_{B(x)}(x_t - \alpha \nabla_x \widehat{c}_A(h_\theta(x_t), y))$, where Π is a projection operator ensuring that the perturbed input falls within an acceptable boundary $B(x)$. After T PGD

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 8 September 2020 doi:10.20944/preprints202009.0184.v1

iterations, set $a(x) = x_T$ and optimize with respect to θ. The authors argue that the PGD attack is the strongest one using only gradient information from the target model. However, there is evidence that it is not sufficient for full defence in neural models [Gowal *et al.* 2018].

Other approaches have focused on improving the game theoretic model in Dalvi *et al.* [2004] but, to our knowledge, none has been able to overcome the unrealistic common knowledge assumptions, as may be seen in recent reviews by Biggio and Roli [2018] and Zhou *et al.* [2018], who point out the importance of this issue. As an example, Kantarcıoğlu *et al.* [2011] use a Stackelberg game in which both players know each other payoff functions. Only Großhans *et al.* [2013] have attempted to relax common knowledge assumptions in adversarial regression settings, reformulating the corresponding problem as a Bayesian game.

4. Adversarial classification: Adversarial risk analysis approaches

Given the above mentioned issue, we provide ARA solutions to the AC problem. We focus first on modelling the adversary's problem in the operation phase. We present the classification problem faced by C as a Bayesian decision analysis problem in Figure 3, derived from Figure 2. In it, A's decision appears as random to the classifier, since she does not know how the adversary will attack the data[4]. An adversary unaware classifier would classify the observed instance x' based on

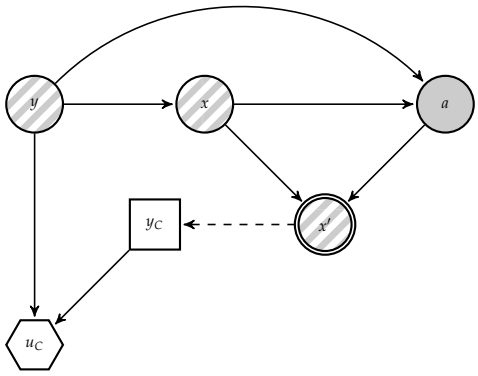

Figure 3. Classifier problem

$$\arg\max_{y_c} \sum_{i=1}^{2} u_C(y_c, y_i) p_C(y_i | X = x'). \tag{6}$$

This leads to performance degradation as reflected in Section 2.2. In contrast, an adversary aware classifier would use

$$\arg\max_{y_c} \sum_{i=1}^{2} u_C(y_c, y_i) p_C(y_i | X' = x'). \tag{7}$$

In the adversary unaware case, $p_C(y_i|X = x')$ is easily estimated using the training data. However, estimating the probabilities $p_C(y_i|X' = x')$ is harder, as it entails modeling how the adversary will modify the original instance x, which is unobserved. Moreover, recall that common knowledge is not available, so we actually lack A's beliefs and preferences. ARA helps us in modeling our uncertainty about them. In doing so, model robustness is typically enhanced. We discuss two strategies depending on whether we use generative or discriminative classifiers as base models.

---

[4] For notational convenience, when necessary we distinguish between random variables and realizations using upper and lower case letters, respectively. In particular, we denote by X the random variable referring to the original instance (before the attack) and X' that referring to the possibly attacked instance. $\hat{z}$ will indicate an estimate of z.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 8 September 2020 doi:10.20944/preprints202009.0184.v1

4.1. The case of generative classifiers

Suppose a generative classifier is required. As training data is clean by assumption, we can estimate $p_C(y)$ (modeling the classifier's beliefs about the class distribution) and $p_C(X = x|y)$ (modeling her beliefs about the feature distribution given the class when A is not present). In addition, assume that when the classifier observes $X' = x'$, she can estimate the set $\mathcal{X}'$ of original instances x potentially leading to the observed x'. When the feature space is endowed with a metric d, an heuristic to approximate $\mathcal{X}'$ would be to consider $\mathcal{X}' = \{x : d(x, x') < \rho\}$ for a certain threshold ρ.

Given the above, when observing x' the classifier should choose the class with maximum posterior expected utility (7). Applying Bayes formula, and ignoring the denominator, irrelevant for optimisation purposes, she must find the class

$$\begin{aligned} y_c^*(x') &= \arg\max_{y_c} \sum_{i=1}^{2} u_C(y_c, y) p_C(y_i) p_C(X' = x'|y_i) \\ &= \arg\max_{y_c} \sum_{i=1}^{2} u_C(y_c, y_i) p_C(y_i) \left[\sum_{x \in \mathcal{X}'} p_C(X' = x'|X = x, y_i) p_C(X = x|y_i) \right]. \end{aligned} \tag{8}$$

In such a way, A's modifications are taken into account through the probabilities $p_C(X' = x'|X = x, y)$. At this point, recall that the focus is restricted to integrity violation attacks. Then, $p_C(X' = x'|X = x, y_2) = \delta(x' - x)$ and problem (8) becomes

$$\begin{aligned} \arg\max_{y_c} \Big[& u_C(y_c, y_1) p_C(y_1) \textstyle\sum_{x \in \mathcal{X}'} p_C(X' = x'|X = x, y_1) p_C(X = x|y_1) \\ & + u_C(y_c, y_2) p_C(y_2) p_C(X = x'|y_2) \Big]. \end{aligned} \tag{9}$$

Note that should we assume common knowledge, we would know A's beliefs and preferences and, therefore, we would be able to solve his problem exactly: when A receives instance x with label y_1, we could compute the transformed instance. In this case, $p_C(X'|X = x, y_1)$ would be 1 just for the x whose transformed instance coincides with that observed by the classifier and 0, otherwise. Inserting this in (9), we would recover Dalvi's formulation (5). However, common knowledge does not hold. Thus, when solving A's problem we have to take into account our uncertainty about his elements and, given that he receives x with label y_1, we will not be certain about the attacked output x'. This will be reflected in our estimate $p_C(x'|x, y_1)$ which will not be 0 or 1 as in Dalvi's approach (stage 3). With this estimate, we would solve problem (9), summing $p_C(x|y_1)$ over all possible originating instances, with each element weighted by $p_C(x'|x, y_1)$.

To estimate these missing elements, we resort to A's problem, assuming that the attacker aims at modifying x to maximize his expected utility by making C classify malicious instances as innocent. The decision problem faced by A is presented in Figure 4, derived from Figure 2. In it, C's decision appears as an uncertainty to A. To solve this problem, we need $p_A(y_c^*(x')|x')$, which models A's beliefs about C's decision when she observes x'. Let us designate by p the probability $p_A(y_c^*(a(x)) = y_1|a(x))$ that A concedes to C saying that the instance is malicious when she observes $x' = a(x)$. Since A will have uncertainty about it, let us model its density using $f_A(p|x' = a(x))$ with expectation $p^A_{x'=a(x)}$. Then, upon observing instance x of class y_1, A would choose the data transformation maximizing his expected utility:

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 8 September 2020 doi:10.20944/preprints202009.0184.v1

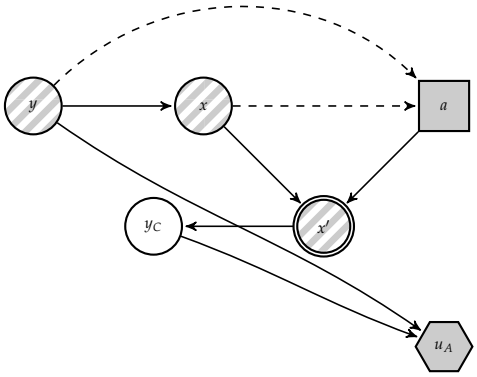

Figure 4. Adversary problem

$$\begin{aligned} x'(x, y_1) &\equiv a^*(x, y_1) = \arg\max_z \int \Big[u_A(y_c^*(z) = y_1, y_1) \cdot p \\ &+ u_A(y_c^*(z) = y_2, y_1) \cdot (1 - p) \Big] f_A(p|z = a(x)) \mathrm{d}p \\ &= \arg\max_z \left[u_A(y_1, y_1) - u_A(y_2, y_1) \right] p^A_{z=a(x)} + u_A(y_2, y_1), \end{aligned} \tag{10}$$

where $u_A(y_i, y_j)$ is the attacker's utility when the defender classifies an instance of class y_j as one of class y_i.

However, the classifier does not know the involved utilities u_A and probabilities $p^A_{z=a(x)}$ from the adversary. Let us model such uncertainty through a random utility function U_A and a random expectation $P^A_{z=a(x)}$. Then, we could solve for the random attack, optimizing the random expected utility

$$X'(x, y_1) \equiv A^*(x, y_1) = \arg\max_z \left(\left[U_A(y_1, y_1) - U_A(y_2, y_1) \right] P^A_{z=a(x)} + U_A(y_2, y_1) \right),$$

and making $p_C(x'|x, y_1) = Pr(X'(x, y_1) = x')$, assuming that the set of attacks is discrete.

Without loss of generality, we associate utility 0 with the worst consequence and 1 with the best one, having the other consequences intermediate utilities [French and Rios Insua 2000]. In the attacker's problem, his best consequence holds when the classifier accepts a malicious instance as benign (he has opportunities to continue with his operations) while the worst consequence appears when the defender stops a instance (he has wasted effort in a lost opportunity). Therefore, we may adopt $U_A(y_1, y_1) \sim \delta_0$ and $U_A(y_2, y_1) \sim \delta_1$. Then, the Attacker's random optimal attack would be

$$X'(x, y_1) \equiv A^*(x, y_1) = \arg\max_z \left[\Big(0 - 1\Big) P^A_{z=a(x)} + 1 \right] = \arg\min_z P^A_{z=a(x)}. \tag{11}$$

Modeling $P^A_{z=a(x)}$ is more delicate. It entails strategic thinking and could lead to a hierarchy of decision making problems, as described in Rios and Rios Insua [2012] in a simpler context. An heuristic to assess it is based on using the probability $r = Pr_C(y_c^*(z) = y_1|z)$ that C assigns to the object received being malicious assuming that she observed z, with some uncertainty around it. Being a probability, r ranges in $[0, 1]$ and we could make $P^A_{z=a(x)} \sim \beta e(\delta_1, \delta_2)$, with mean $\delta_1/(\delta_1 + \delta_2) = r$ and variance $(\delta_1\delta_2)/[(\delta_1 + \delta_2)^2(\delta_1 + \delta_2 + 1)] = var$ as perceived. var has to be tuned depending on the amount of knowledge C has about A. Details on how to estimate r are problem dependent.

In general, to approximate $p_C(x'|x, y_1)$ we use Monte Carlo (MC) simulation drawing K samples $(P_z^{A,k})$, $k = 1, \ldots, K$ from P_z^A, finding $X'_k(x, y_1) = \arg\min_z P_z^{A,k}$ and estimating $p_C(x'|x, y_1)$ using the

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 8 September 2020 doi:10.20944/preprints202009.0184.v1

proportion of times in which the result of the random optimal attack coincides with the instance actually observed by the defender:

$$\widehat{p}_C(x' \mid x, y_1) = \frac{\#\{X'(x, y_1) = x'\}}{K}. \tag{12}$$

It is easy to prove [Rubinstein and Kroese 2016] that (12) converges almost surely to $p_C(x'|x, y_1)$. Once we have such approach, we implement the scheme described through Algorithm 1.

Algorithm 1 General ARA procedure for AC. Generative

Input: Training data $\mathcal{D}$, test instance x'.
Output: A classification decision $y_c^*(x')$.
Training
Train a generative classifier to estimate $p_C(y)$ and $p_C(x|y)$
End Training
Operation
Read x'.
Estimate $p_C(x'|x, y_1)$ for all $x \in \mathcal{X}'$.
Solve

$$\begin{aligned} y_c^*(x') &= \arg\max_{y_C} \Big[u_C(y_C, y_1)\widehat{p}_C(y_1) \sum_{x \in \mathcal{X}'} \widehat{p}_C(x'|x, y_1)\widehat{p}_C(x|y_1) \\ &+ u_C(y_C, y_2)\widehat{p}_C(x'|y_2)\widehat{p}_C(y_2) \Big]. \end{aligned}$$

Output $y_c^*(x')$.
End Operation

4.2. The case of discriminative classifiers

For discriminative classifiers, we cannot use the previous approach as we lack an estimate of $p_C(X = x|y)$. To motivate an alternative analysis, assume the classifier knows the attack that she has suffered, which is invertible in the sense that she may recover the original $x = a^{-1}(x')$. Then, rather than classifying based on (6), as an adversary unaware classifier would do, she should classify based on

$$\arg\max_{y_c} \sum_{i=1}^{2} u_C(y_c, y_i) p_C(y_i|X = a^{-1}(x')).$$

However, the uncertainty about attack a, induces uncertainty about the originating instance x. Suppose we model our uncertainty about the origin x of the attack through a distribution $p_C(X = x|X' = x')$ with support over the set $\mathcal{X}'$ of reasonable originating features x. Then, marginalizing out all possible originating instances, the expected utility that the classifier would get for her classification decision y_c would be

$$\begin{aligned} \psi(y_c) &= \sum_{x \in \mathcal{X}'} \left(\sum_{i=1}^{2} u_C(y_c, y_i) p_C(y_i|X = x) \right) p_C(X = x|X' = x') \\ &= \sum_{i=1}^{2} u_C(y_c, y_i) \left[\sum_{x \in \mathcal{X}'} p_C(y_i|X = x) p_C(X = x|X' = x') \right], \end{aligned} \tag{13}$$

and we would solve for

$$y_c^*(x') = \arg\max_{y_c} \psi(y_c).$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 8 September 2020 doi:10.20944/preprints202009.0184.v1

Typically, expected utilities (13) are approximated by MC using a sample $\{x_n\}_{n=1}^N$ from $p_C(x|x')$. Algorithm 2 summarises a general procedure.

Algorithm 2 General ARA procedure for AC. Discriminative

Input: Monte Carlo size N, training data $\mathcal{D}$, test instance x'.
Output: A classification decision $y_c^*(x')$.
Training
Based on $\mathcal{D}$, train a discriminative classifier to estimate $p_C(y|x)$.
End Training
Operation
Read x'
Estimate $\mathcal{X}'$ and $p_C(x|x'), x \in \mathcal{X}'$
Draw sample $\{x_n\}_{n=1}^N$ from $p_C(x|x')$.
Find $y_c^*(x') = \arg\max_{y_c} \frac{1}{N} \sum_{i=1}^2 \left(u_C(y_c, y_i) \left[\sum_{n=1}^N p_C(y_i|x_n) \right] \right)$
Output $y_c^*(x')$
End Operation

When implementing this approach, we need to be able to estimate $\mathcal{X}'$ and $p_C(x|x')$ or, at least, sample from this distribution. One possibility would be to use a uniform distribution over $\mathcal{X}'$. Alternatively, in the metric case, we could make $p_C(x|x') = \frac{h}{d(x,x')}$, where $\frac{1}{h} = \sum_{x \in \mathcal{X}'} \frac{1}{d(x,x')}$ (ignoring x' as possible origin) assuming that the closer the feature, the more likely it is to be the origin. However, as shown in Ríos Insua *et al.* [2020], better forecasts hold if we explicitly model the attacker behaviour using the information available about him. One possibility is to sample from $p_C(X|X' = x')$ by leveraging approximate Bayesian computation (ABC) techniques, Csilléry *et al.* [2010]. This requires being able to sample from $p_C(X)$ and $p_C(X'|X = x)$, which we address first.

Estimating $p_C(x)$ is done using training data, untainted by assumption. For this, we can employ an implicit generative model, such as a generative adversarial network [Goodfellow *et al.* 2014] or energy-based modeling [Grathwohl *et al.* 2019]. Instead, sampling from $p_C(x'|x)$ entails strategic thinking. Notice first that

$$p_C(x'|x) = \sum_{c=1}^{2} p_C(x'|x, y_c) p_C(y_c|x) = p_C(x'|x, y_1) p_C(y_1|x) + \delta(x' - x) p_C(y_2|x).$$

We easily generate samples from $p_C(y_c|x)$, as we can estimate those probabilities based on the training data as in Section 2.1. Then, we can obtain samples from $p_C(x'|x)$ sampling $y \sim p_C(y|x)$ first; next, if $y = y_2$ return x or, otherwise, sample $x' \sim p_C(x'|x, y_1)$. To sample from $p_C(x'|x, y_1)$, we need to model the problem faced by the attacker when he receives instance x with label y_1.The attacker will maximize his expected utility by transforming instance x to x' given by the solution of (10). Again, associating utility 0 with the attacker's worst consequence and 1 with the best one; and modeling our uncertainty about the attacker's estimates of $p^A_{z=a(x)}$ using random expected probabilities $P^A_{z=a(x)}$, we would look for random optimal attacks $X'(x, y_1)$ as in (11). By construction, if we sample $p^A_{z=a(x)} \sim P^A_{z=a(x)}$ and solve

$$x' = \arg\min_{z \in \mathcal{X}'} p^A_{z=a(x)},$$

x' is distributed according to $p_C(x'|x, y_i)$ which was the last ingredient required.

Once with these two procedures, we could generate samples from $p_C(X|X' = x')$ with ABC techniques. This entails generating $x \sim p_C(X)$, $\tilde{x}' \sim p_C(X'|X = x)$ and accepting x if $\phi(\tilde{x}', x') < \delta$, where ϕ is a distance function defined in the space of features and δ is a tolerance parameter. The x generated in this way is distributed approximately according to the desired $p_C(x|x')$. However, the probability of generating samples for which $\phi(\tilde{x}', x') < \delta$ decreases with the dimension of x'. A

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 8 September 2020 doi:10.20944/preprints202009.0184.v1

common solution replaces x' with $s(x')$, a set of summary statistics that capture the relevant information in x', and the acceptance criterion would be replaced by $\phi(s(\tilde{x}'), s(x')) < \delta$. The choice of summary statistics is problem specific. We sketch the complete ABC sampling procedure in Algorithm 3, to be integrated within Algorithm 2.

Algorithm 3 ABC scheme to sample from $p_C(x|x')$ within Algorithm 2

Input: Observed instance x', data model $p_C(x)$, U_A, P_A^c, family of statistics s.
Output: A sample approximately distributed according to $p_C(x|x')$.
while $\phi(s(x'), s(\tilde{x}')) > \delta$ **do**
  Sample $x \sim p_C(x)$
  Sample $y \sim p_C(y|x)$
  if $y = y_2$ **then**
    $\tilde{x}' = x$
  else
    Sample $p_z^A \sim P_z^A$
    Compute $\tilde{x}' = \arg\min_z p_z^A$
  end if
  Compute $\phi(s(x'), s(\tilde{x}'))$
end while
Output x

5. Scalable adversarial classifiers

The approach presented in Section 4 performs all the relevant inference about the adversary during operations. This could be too expensive computationally, especially in applications that require fast predictions based on large scale deep models as motivated by the following image processing problem.

Attacks to neural-based classifiers Section 3.2 discussed adversarial examples. This kind of attacks may really harm neural network performance, such as those used in image classification tasks [Szegedy *et al.* 2014]. As an example, with a relatively simple deep convolutional neural net (CNN) model [Goodfellow *et al.* 2016] we accurately predict 99% of the handwritten digits in the MNIST data set [Le Cun 1998]. Figure 5 provides ten MNIST original samples (top row) and the corresponding images (bottom row) perturbed through FGSM, which are misclassified. For example, the original 0 (first column) is classified as such; however the perturbed one is not classifed as a 0 (more specifically, as an 8) even if it looks as such to the human eye.

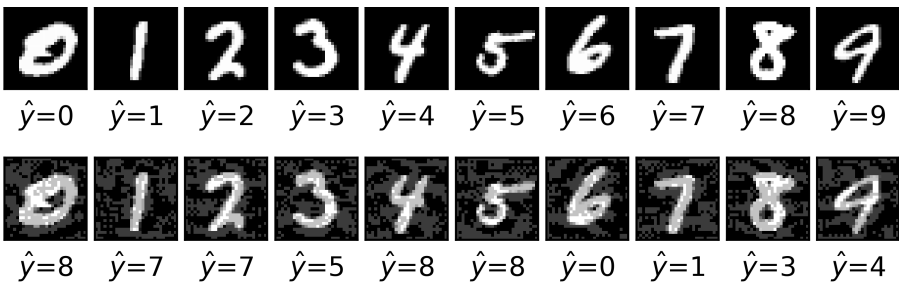

Figure 5. Ten MNIST examples (top) and their perturbations (bottom). Predicted class shown for each example.

Globally, accuracy gets reduced to 62%, showing again a very important performance degradation due to the attack. △

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 8 September 2020 doi:10.20944/preprints202009.0184.v1

The difficulties entailed by the approach in Section 4 stem from three issues:

- Iteration over the set $\mathcal{X}'$ of potentially originating instances. If no assumptions about the attacks are made, this set grows rapidly. For instance, in the spam detection example in Section 2.2, let n be the number of words in the dictionary considered by C to undertake the classification. If we assume that the attacker modifies at most one word, the size of $\mathcal{X}'$ is $\mathcal{O}(n)$; if he modifies at most two words, it is $\mathcal{O}(n^2)$, ..., and if he modifies at most n words, the number of possible adversarial manipulations (and thus the size of $\mathcal{X}'$) is 2^n. Even more extremely, in the image processing example we would have to deal with very high-dimensional data (the MNIST example above consists of 28×28 pixels, each taking 256 possible values depending on the gray level). This deems any enumeration over the set $\mathcal{X}'$ totally unfeasible. In order to tackle this issue, constraints over the attacks could be adopted.
- Sampling from $p_C(x|x')$. In turn, a huge $\mathcal{X}'$ renders this sampling inefficient. As an example, the ABC scheme in Algorithm 3, could converge very slowly, requiring careful tuning of the tolerance rate δ. In addition, such algorithm requires the use of an unconditional generative model $p_C(x)$ to draw realistic samples from the data distribution. This could be computationally demanding, specially when dealing with high-dimensional data.
- Inference about the adversary during operations. As the key adversary modelling steps are taken during operations, the approach could be inefficient in applications that require fast predictions.

5.1. Protecting differentiable classifiers

Alternatively, inference about the adversary could be undertaken during the training phase as now presented. This provides faster predictions during operations and avoids the expensive step of sampling from $p_C(x|x')$. For that, a relatively weak assumption is made: the model can be expressed in a probabilistic form $p_C(y|x, \beta)$, that is differentiable in the β parameters. Mainstream models such as logistic regression or neural networks satisfy such condition. This assumption brings in several benefits. First, we can train the model using *stochastic gradient descent* (SGD, Bottou and Bousquet [2008]) or any of its recent variants, such as Adam [Kingma and Ba 2014] allowing for scaling to both very wide and tall datasets. Then, from SGD we can obtain the posterior distribution of the model by adding a suitable noise term as in SG-MCMC samplers like stochastic gradient Langevin Dynamics (SGLD) [Welling and Teh 2011] or accelerated variants [Ma *et al.* 2015, Gallego and Insua 2018].

We require only sampling attacked instances from $p_C(x'|x)$, an attacker model. Depending on the type of data, this attacker model can come through a discrete optimization problem (as in the attacks of Section 2.2) or a continuous optimization problem (in which we typically resort to gradient information to obtain the most harmful perturbation as in FGSM). These attacks require white-box access to the defender model, which, as mentioned, is usually unrealistic in security. We add realism by incorporating two kinds of uncertainties.

1. Defender uncertainty over the attacker model $p_C(x'|x)$. First, as the attacker modifies data in the operation phase, the defender has access only to training data $\mathcal{D}$; therefore, she will have to simulate attacker's actions using such training set. Next, uncertainty can also come from the adversarial perturbation chosen. If the model is also differentiable wrt the input x (as with continuous data such as images or audio), instead of computing a single, optimal and deterministic perturbation as in AT, we use SGLD to sample adversarial examples from regions of high adversarial loss, adding a noise term to generate uncertainty. Thus, we have iterates of the form

$$x_{t+1} = x_t - \epsilon \operatorname{sign} \nabla_x \log p_C(y|x_t, \beta) + \mathcal{N}(0, 2\epsilon) \tag{14}$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 8 September 2020 doi:10.20944/preprints202009.0184.v1

for $t = 1, \ldots, T$, where ϵ is a step size. We can also consider uncertainty over the hyperparameters ϵ (from a re-scaled Beta distribution, since it is unreasonable to consider too high or too low learning rates) and the number T of iterations (from a Poisson distribution).

2. Attacker uncertainty over the defender model $p_C(y|x, \beta)$. It is reasonable to assume that the specific model architecture and parameters are unknown by the attacker. To reflect his uncertainty, he will instead perform attacks over a Bayesian model $p_C(y|x, \beta)$ reflecting uncertainty over the values of the model parameters β, with continuous support. This can be implemented through scalable Bayesian approaches in deep models: the defended model is trained using SGLD, obtaining posterior samples via the iteration

$$\beta_{t+1} = \beta_t + \eta \nabla_\beta \log p_C(y|x, \beta_t) + \mathcal{N}(0, 2\eta), \tag{15}$$

with η a learning rate, and x sampled either from the set $\mathcal{D}$ (untainted) or using an attacker model as in the previous paragraph (adversarial). We sample maintaining a proportion $1:1$ of clean and attacked data.

The previous approaches incorporate some uncertainty on the attacker's elements to sample from $p_C(x'|x)$. A full ARA sampling from this distribution can be performed as well. Recall that $p_C(x'|x)$ models the classifier's uncertainty about the attack output when A receives instance x, which stems from the lack of knowledge about A's utilities and probabilities. Samples from $p_C(x'|x)$ could thus be obtained as in Section 4.2. This enables explicit modeling of the classifier's uncertainty about A's utilities and probabilities.

Algorithm 4 Large scale ARA-robust training for AC

Input: Defender model $p_C(y|x, \beta)$, attacker model $p_C(x'|x)$.
Output: A set of particles $\{\beta_i\}_{i=1}^K$ approximating the posterior distribution of the defender model learnt using ARA training.
for $t = 1$ to T **do**
Sample $x_1, \ldots, x_K \sim p_C(x'|x)$ with (14) or the approach in Section 4.2 (depending on continuous or discrete data).
$\beta_{i,t+1} = \beta_{i,t} + \eta \nabla_\beta \log p_C(y|x, \beta_t) + \mathcal{N}(0, 2\eta)$ for each i (SGLD)
end for
Output $(\beta_{1,T}, ..., \beta_{K,T}))$

Algorithm 4 describes how to generate samples incorporating both types of uncertainty previously described. On the whole, this algorithm uses the first source to generate perturbations to robustly train the defender's model based on the second source. The outcome is a more robust model than the one we could achieve with just AT protection, since we incorporate some level of adversarial uncertainty. In the end, we collect K posterior samples, $\{\beta_k\}_{k=1}^K$, and compute predictive probabilities for a new sample x via marginalization through $p_C(y|x) = \frac{1}{K} \sum_{k=1}^K p_C(y|x, \beta_k)$, using the previous predictive probability to robustly classify the received sample.

6. Case study

We use the spam classification example from Section 2.2 and its dataset as a benchmark to illustrate the frameworks in Sections 4 and 5. As shown in Section 2.2, simple attacks such as good/bad word insertions are sufficient to critically affect the performance of spam detection algorithms. We first test the performance of the ARA approach to AC in Section 4.2, as a defence mechanism against attacks that modify at most 2 words. For the classifier, we use a 0-1 utility model.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 8 September 2020 doi:10.20944/preprints202009.0184.v1

6.1. ARA defense in spam detection problems

For the first batch of experiments, we use the same classifiers as in Section 2.2. We simulate attacks over the instances in the test set, solving problem (11) for each test spam email, removing the uncertainty that is not present from the adversary's point of view. He would have uncertainty about $p^A_{z=a(x)}$, as this quantity depends on the defender's decision. We test our ARA approach to AC against a worst case attacker who knows the true value of $p_C(y|x)$ and estimates $p^A_{z=a(x)}$ through $\frac{1}{M}\sum_{n=1}^{M} p_C(y_1|x_n)$ for a sample $\{x_n\}_{n=1}^{M}$ from $p^*(x|x')$, a uniform distribution over the set of all instances at distance less than or equal to 2 from the observed x', using $\sum_{i=1}^{54}|x_i - x'_i|$ as distance function. We use $M = 40$.

To model the uncertainty about p^A_z, we use beta distributions centered at the attacker's values of probabilities with variances chosen to guarantee that the distribution is concave in its support: they must be bounded from above by $\min\{[\mu^2(1-\mu)]/(1+\mu), [\mu(1-\mu)^2]/(2-\mu)\}$, were μ is the corresponding mean. We set the variance to be 10% of this upper bound, thus reflecting a moderate lack of knowledge about the attacker's probability judgements.

Our implementations of Algorithms 2 and 3, use as summary statistics s the 11 most relevant features, with relevance based on permutation importance, Altmann *et al.* [2010]; (which assesses how accuracy decreases when a particular feature is not included). We set the ABC tolerance δ to be 1, and distance $\phi(x', x) = \sum_{i=1}^{54}|x_i - x'_i|$. Finally, for each instance in the test set we generate 20 samples from $p_C(x|x')$. Table 2 compares the ARA performance on tainted data with that of the raw classifiers (from Table 1). As can be seen, our approach outperforms all of them, showcasing the benefits of explicitly modeling the attacker's behaviour in adversarial environments, considerably recovering from performance degradation.

Classifier	Acc. Taint.	Acc. ARA Taint.
Naive Bayes	0.774 ± 0.026	0.924 ± 0.004
Logistic Reg.	0.681 ± 0.009	0.917 ± 0.003
Neural Net	0.764 ± 0.007	0.811 ± 0.010
Random Forest	0.663 ± 0.006	0.820 ± 0.005

Table 2. Accuracy comparison (with precision) of four classifiers on attacked data, with and without ARA-enhanced defense.

Interestingly, in the case of the naïve Bayes classifier, our ARA approach outperforms the classifier under raw untainted data (Table 1). This effect has been already observed by Naveiro *et al.* [2019] and Goodfellow *et al.* [2014] for other algorithms and application areas. A possible explanation is that taking into account the presence of an adversary has a regularizing effect, being able to improve the original accuracy of the base algorithm and making it more robust.

6.2. Robustified classifiers

We next evaluate the scalable framework in Section 5 under the two differentiable models among the previous ones: logistic regression and neural network (with two hidden layers). Observe that both models can be trained using stochastic gradient methods plus noise to obtain uncertainty estimates from the posterior as in (15). After, we attack the clean test set using the same procedure as in Section 2.2 and evaluate the performance of the robustification approach. Since we are dealing with discrete attacks, we cannot use the uncertainty over attacks as in (14), and just resort to add the attacker's uncertainty over the defender model via the second uncertainty source described in Section 5.1. To perform classification with this model, we maintain a Bayesian ensemble of 5 different parameter sets obtained after SGLD iteration. Results are in Table 3 which include as entries: Acc. Unt. (accuracy over untainted data); Acc. Tai. (it. over tainted data); Acc. Rob. Taint. (it. over tainted data after our robustification adding uncertainties). Note that the first two columns do not coincide with those

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 8 September 2020 doi:10.20944/preprints202009.0184.v1

in Table 1, as we have changed the optimizers to variants of SGD to be amenable to the robustified procedure in Section 5.1.

Classifier	Acc. Unt.	Acc. Tai.	Acc. Rob. Taint.
Logistic Reg.	0.931 ± 0.007	0.705 ± 0.009	0.946 ± 0.003
Neural Net	0.937 ± 0.005	0.636 ± 0.009	0.960 ± 0.002

Table 3. Accuracy of two classifiers on clean (untainted), and attacked (tainted) data, with and without robustification.

Observe that the robustification process from Section 5 indeed protects differentiable classifiers, achieving even higher accuracies than those attained by the original classifier over clean data, as commented above.

7. Discussion

Adversarial classification aims at enhancing classifiers to achieve robustness in presence of adversarial examples, as usually encountered in many security applications. The pioneering work of Dalvi *et al.* [2004] framed most later approaches to AC within the standard game theoretic paradigm, in spite of the unrealistic common knowledge assumptions required, actually even questioned by the authors. After reviewing them, and analysing critically their assumptions, we have presented two formal probabilistic approaches for AC based on ARA, that avoid strong common knowledge assumptions. They are general in the sense that application-specific assumptions are kept to a minimum. We have presented the framework in two different forms: in Section 4 inference about the adversary is performed in the operational phase, with variants for generative and discriminative classifiers. In Section 5, adversarial aspects are incorporated in the training phase. Depending on the particular application one of the frameworks could be preferred over the other. The first one allows to make real time inference about the adversary, as it explicitly models his decision making process in operation time; its adaptability is better as it does not need to be retrained every time we need to modify the adversary model. However, this comes at a high computational cost. In applications in which there is a computational bottleneck, the second approach may be preferable, with possible changes in the adversary's behaviour incorporated via retraining. This tension between the need to robustify algorithms against attacks (training phase, Section 5) and the fast adaptivity of attackers against defences (operational phase, Section 4) is well exemplified in the phishing detection domain as discussed in El Aassal *et al.* [2020].

Our framework may be extended in several ways. First, we could adapt the proposed approach to situations in which there is repeated play of the AC game, thus introducing the possibility of learning the adversarial utilities and probabilities in a Bayesian way. Learning over opponent types has been explored with success in reinforcement learning scenarios, Gallego *et al.* [2019]. This could be extended to the classification setting. Besides exploratory attacks, attacks over the training data [Biggio *et al.* 2012] may be relevant in certain contexts. Also, the extension to the case of attacks to innocent instances (not just integrity violation ones) seems feasible using the scalable framework. Through this paper, we focused on the binary classification task. An initial extension to multi-class problems is shown in Gallego *et al.* [2020], but further attention to this setting would be helpful.

Additional work should be done on the algorithmic side as well. In our approach we go through a simulation stage to forecast attacks and an optimization stage to determine optimal classification. The whole process might be performed through a single stage, possibly based on augmented probability simulation [Ekin *et al.* 2019].

We have also shown how the robustification procedure from Section 5 can be an efficient way to protect large-scale models, such as those trained using first-order methods. It is well-known that Bayesian marginalization improves the generalization capabilities of flexible models since the ensemble helps in better exploring the posterior parameter space [Wilson and Izmailov 2020]. Our

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 8 September 2020 doi:10.20944/preprints202009.0184.v1

experiments seem to confirm that this behaviour also happens in the domain of adversarial robustness. Thus, bridging the gap between large scale Bayesian methods and Game Theory, as done in the ARA framework, suggests a powerful way to develop principled defences. To this end, strategies to more efficiently explore the highly complex, multi-modal posterior distributions of neural models constitutes another line of further work.

Lastly, several application areas could benefit highly from protecting their underlying ML models. Spam detectors were the running example in this article. Malware and phishing detection are two crucial cybersecurity problems in which the data distribution of computer programs is constantly changing, driven by attacker's interests in evading detectors. Finally, the machine learning algorithms underlying autonomous driving systems need to be robustified from perturbations to their visual processing models and this could be performed through our approaches.

Acknowledgments: This work was partially supported by the NSF under Grant DMS-1638521 to the Statistical and Applied Mathematical Sciences Institute, a BBVA Foundation project and the Trustonomy project, which has received funding from the European Community's Horizon 2020 research and innovation programme under grant agreement No 812003. RN acknowledges support of the Spanish Ministry of Science and Research for his grant FPU15-03636. VG acknowledges support of the Spanish Ministry of Science and Research for his grant FPU16-05034. DRI is grateful to the MTM2017-86875-C3-1-R AEI/ FEDER EU project and the AXA-ICMAT Chair in Adversarial Risk Analysis.

Conflicts of Interest: The authors declare no conflict of interest.

References

Bolton, R.J.; Hand, D.J. Statistical fraud detection: A review. *Statistical science* **2002**, pp. 235–249.

El Aassal, A.; Bakis, S.; Das, A.; Verma, R. An In-depth benchmarking and evaluation of phishing detection research for security needs. *IEEE Access* **2020**, *8*, 22170–22192.

Simanjuntak, D.; Ipung, H.; Lim, C.; Nugroho, A. Classification Techniques Used to Faciliate Cyber Terrorism Investigation. *Second International Conference on Advances in Computing, Control, and Telecommunication Technologies* **2010**, pp. 198–200.

Comiter, M. *Attacking Artificial Intelligence*; Belfer Center Paper, 2019.

Dalvi, N.; Domingos, P.; Mausam.; Sumit, S.; Verma, D. Adversarial classification. Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2004, KDD '04, pp. 99–108.

Biggio, B.; Roli, F. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition* **2018**, *84*, 317 – 331.

Zhou, Y.; Kantarcioglu, M.; Xi, B. A survey of game theoretic approach for adversarial machine learning. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* **2018**, p. e1259.

Antos, D.; Pfeffer, A. Representing Bayesian Games without a Common Prior. In *Proc. AAMAS 2010*; van der Hoek, Kaninka, L.; Sen., Eds.; IFAMAS, 2010.

Rios Insua, D.; Rios, J.; Banks, D. Adversarial risk analysis. *Journal of the American Statistical Association* **2009**, *104*, 841–854.

Naveiro, R.; Redondo, A.; Insua, D.R.; Ruggeri, F. Adversarial classification: An adversarial risk analysis approach. *International Journal Approximate Reasoning* **2019**, p. https://doi.org/10.1016/j.ijar.2019.07.003.

Gallego, V.; Naveiro, R.; Redondo, A.; Insua, D.R.; Ruggeri, F. Protecting Classifiers From Attacks. A Bayesian Approach. *arXiv preprint arXiv:2004.08705* **2020**.

Huang, L.; Joseph, A.D.; Nelson, B.; Rubinstein, B.I.; Tygar, J.D. Adversarial machine learning. Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence, 2011, AISec '11, pp. 43–58.

Barreno, M.; Nelson, B.; Sears, R.; Joseph, A.D.; Tygar, J.D. Can machine learning be secure? Proceedings of the 2006 ACM Symposium on Information, Computer and Communications Security. ACM, 2006, pp. 16–25.

Bishop, C.M. *Pattern recognition and machine learning*; Springer, 2006.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 8 September 2020 doi:10.20944/preprints202009.0184.v1

Rish, I.; others. An empirical study of the naive Bayes classifier. IJCAI 2001 workshop on empirical methods in artificial intelligence, 2001, Vol. 3, pp. 41–46.

Kingma, D.P.; Mohamed, S.; Rezende, D.J.; Welling, M. Semi-supervised learning with deep generative models. Advances in neural information processing systems, 2014, pp. 3581–3589.

McCullagh, P.; Nelder, J. *Generalized Linear Models, Second Edition*; Chapman and Hall/CRC Monographs on Statistics and Applied Probability Series, Chapman & Hall, 1989.

Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; The MIT Press, 2016.

Shachter, R.D. Evaluating Influence Diagrams. *Operations Research* **1986**, *34*, 871–882.

Platt, J. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers* **1999**, *10*, 61–74.

Hopkins, M.; Reeber, E.; Forman, G.; Suermondt, J. Spambase Data Set. https://archive.ics.uci.edu/ml/datasets/Spambase, 1999.

Park, T.; Casella, G. The Bayesian lasso. *Journal of the American Statistical Association* **2008**, *103*, 681–686.

Kim, J.H. Estimating classification error rate: Repeated cross-validation, repeated hold-out and bootstrap. *Computational Statistics and Data Analysis* **2009**, *53*, 3735–3745.

Banks, D.; Rios, J.; Rios Insua, D. *Adversarial Risk Analysis*; Francis Taylor, 2015.

Menache, I.; Ozdaglar, A. *Network Games: Theory, Models, and Dynamics*; Morgan & Claypool, 2011.

Elkan, C. The Foundations of Cost-Sensitive Learning. International Joint Conference on Artificial Intelligence (IJCAI), 2001.

Biggio, B.; Fumera, G.; Roli, F. Security evaluation of pattern classifiers under attack. *IEEE Transactions on Knowledge and Data Engineering* **2014**, *26*, 984–996.

Li, B.; Vorobeychik, Y. Feature cross-substitution in adversarial classification. Advances in Neural Information Processing Systems, 2014, pp. 2087–2095.

Lowd, D.; Meek, C. Adversarial learning. Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining, 2005, KDD '05, pp. 641–647.

Zhou, Y.; Kantarcioglu, M.; Thuraisingham, B.; Xi, B. Adversarial support vector machine learning. Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2012, pp. 1059–1067.

Goodfellow, I.J.; Shlens, J.; Szegedy, C. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572* **2014**.

Vorobeichyk, Y.; Kantarcioglu, M. *Adversarial Machine Learning*; Morgan Clayton, 2019.

Kołcz, A.; Teo, C.H. Feature Weighting for Improved Classifier Robustness. CEAS'09: Sixth Conference on Email and Anti-Spam, 2009.

Vorobeychik, Y.; Li, B. Optimal randomized classification in adversarial settings. Proceedings of the 2014 International Conference on Autonomous Agents and Multi-agent Systems, 2014, AAMAS '14, pp. 485–492.

Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; Vladu, A. Towards Deep Learning Models Resistant to Adversarial Attacks. International Conference on Learning Representations, 2018.

Gowal, S.; Dvijotham, K.; Stanforth, R.; Bunel, R.; Qin, C.; Uesato, J.; Arandjelovic, R.; Mann, T.A.; Kohli, P. On the Effectiveness of Interval Bound Propagation for Training Verifiably Robust Models. *CoRR* **2018**, *abs/1810.12715*, [1810.12715].

Kantarcıoğlu, M.; Xi, B.; Clifton, C. Classifier evaluation and attribute selection against active adversaries. *Data Mining and Knowledge Discovery* **2011**, *22*, 291–335.

Großhans, M.; Sawade, C.; Brückner, M.; Scheffer, T. Bayesian games for adversarial regression problems. International Conference on Machine Learning, 2013, pp. 55–63.

French, S.; Rios Insua, D. *Statistical Decision Theory*; Wiley, 2000.

Rios, J.; Rios Insua, D. Adversarial Risk Analysis for Counterterrorism Modeling. *Risk Analysis* **2012**, *32*, 894–915.

Rubinstein, R.Y.; Kroese, D.P. *Simulation and the Monte Carlo Method*, 3rd ed.; Wiley Publishing, 2016.

Ríos Insua, D.; Banks, D.; Ríos, J.; Gonz´lez-Ortega, J. Adversarial Risk Analysis as a Decomposition Technique for Structured Expert Judgment. *Advances in Expert Judgement for Decision and Risk Analysis* **2020**.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 8 September 2020 doi:10.20944/preprints202009.0184.v1

Csilléry, K.; Blum, M.G.; Gaggiotti, O.E.; François, O. Approximate Bayesian computation (ABC) in practice. *Trends in ecology & evolution* **2010**, *25*, 410–418.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems 27*; Ghahramani, Z.; Welling, M.; Cortes, C.; Lawrence, N.D.; Weinberger, K.Q., Eds.; Curran Associates, Inc., 2014; pp. 2672–2680.

Grathwohl, W.; Wang, K.C.; Jacobsen, J.H.; Duvenaud, D.; Norouzi, M.; Swersky, K. Your classifier is secretly an energy based model and you should treat it like one. International Conference on Learning Representations, 2019.

Szegedy, C.; Inc, G.; Zaremba, W.; Sutskever, I.; Inc, G.; Bruna, J.; Erhan, D.; Inc, G.; Goodfellow, I.; Fergus, R. Intriguing properties of neural networks. In ICLR, 2014.

Le Cun, Y., C.C.B.C. The MNIST Database. *http://yann.lecun.com/exdb/mnist/* **1998**.

Bottou, L.; Bousquet, O. The tradeoffs of large scale learning. Advances in neural information processing systems, 2008, pp. 161–168.

Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* **2014**.

Welling, M.; Teh, Y.W. Bayesian learning via stochastic gradient Langevin dynamics. Proceedings of the 28th international conference on machine learning (ICML-11), 2011, pp. 681–688.

Ma, Y.A.; Chen, T.; Fox, E. A complete recipe for stochastic gradient MCMC. Advances in Neural Information Processing Systems, 2015, pp. 2917–2925.

Gallego, V.; Insua, D.R. Stochastic gradient MCMC with repulsive forces. *NIPS Workshop on Bayesian Deep Learning* **2018**.

Altmann, A.; Toloşi, L.; Sander, O.; Lengauer, T. Permutation importance: a corrected feature importance measure. *Bioinformatics* **2010**, *26*, 1340–1347.

Gallego, V.; Naveiro, R.; Insua, D.R.; Oteiza, D.G.U. Opponent Aware Reinforcement Learning, 2019, [arXiv:cs.LG/1908.08773].

Biggio, B.; Nelson, B.; Laskov, P. Poisoning attacks against support vector machines. *arXiv preprint arXiv:1206.6389* **2012**.

Ekin, T.; Naveiro, R.; Torres-Barrán, A.; Ríos-Insua, D. Augmented probability simulation methods for non-cooperative games. *arXiv preprint arXiv:1910.04574* **2019**.

Wilson, A.G.; Izmailov, P. Bayesian Deep Learning and a Probabilistic Perspective of Generalization. *arXiv preprint arXiv:2002.08791* **2020**.