Adversarial classification: An adversarial risk analysis approach

Roi Naveiro, Alberto Redondo, David Ríos Insua[*],
Institute of Mathematical Sciences (ICMAT-CSIC)
and
Fabrizio Ruggeri[†]
CNR-IMATI

February 22, 2018

Abstract

Classification problems in security settings are usually contemplated as confrontations in which one or more adversaries try to fool a classifier to obtain a benefit. Most approaches to such adversarial classification problems have focused on game theoretical ideas with strong underlying common knowledge assumptions, which are actually not realistic in security domains. We provide an alternative framework to such problem based on adversarial risk analysis, which we illustrate with several examples. Computational and implementation issues are discussed.

Keywords: Adversarial Machine Learning, Classification, Bayesian Methods.

[*]The work of DRI is supported by the Spanish Ministry program MTM2014-56949-C3-1-R and the AXA-ICMAT Chair on Adversarial Risk Analysis. This work has also been partially supported by the Spanish Ministry of Economy through the Severo Ochoa Program for Centers of Excellence in R&D (SEV-2015-0554) and project MTM2015-72907-EXP.

[†]FR acknowledges the contribution of the *Comunidad de Madrid* through its Chair of Excellence programme.

1 Introduction

Classification is one of the most widely used instances of supervised learning, with applications in numerous fields including spam detection, Fan et al. (2016); computer vision, Chen (2015); and genomics, Zhou et al. (2005). In recent years, the field has experienced an enormous growth becoming a major research area in statistics and machine learning, Efron and Hastie (2016). Most efforts in classification have focused on obtaining more accurate algorithms which, however, largely ignore a relevant issue in many applications: the presence of adversaries who actively manipulate the data to fool the classifier so as to attain a benefit. As an example, when a spammer makes the classifier think that a spam is legit, he may profit by selling the information he gets from the victim. In such contexts, as classification algorithms improve, adversaries usually become smarter when making attacks. The presence of adaptive adversaries has been pointed out in areas such as fraud detection, Zeager et al. (2017), and spam detection, Kołcz and Teo (2009).

Dalvi et al. (2004) provided a pioneering approach to enhance classification algorithms when an adversary is present, calling it adversarial classification (AC). They view AC as a game between a classifier C and an adversary A. C aims at finding an optimal classification strategy against A's optimal attacking strategy. Computing Nash equilibria, Ozdaglar and Menache (2011), in this general game becomes overly complex. Therefore, they propose a simplified version in which C first assumes that data is untainted and computes her optimal classifier; then, A deploys his optimal attack against it; subsequently, C implements the optimal classifier against this attack, and so on. As the authors pointed out, a very strong assumption is made: all parameters of both players are known to each other. Although standard in game theory, this common knowledge (CK) assumption is unrealistic in security scenarios.

Stemming from this work, there has been an important literature in AC, reviewed in Biggio et al. (2014) or Li and Vorobeychik (2014). Subsequent approaches have focused on analyzing attacks over classification algorithms and assessing their security against such attacks. To that end, some assumptions about the adversary are made. For instance, Lowd and Meek (2005a) consider that the adversary is able to send membership queries to the classifier, the entire feature space being known to issue optimal attacks. Then, they prove the vulnerability of linear classifiers against adversaries. Similarly, Zhou et al. (2012) consider that the adversary seeks to push his malicious instances into innocuous ones, assuming that the adversary can estimate such instances. 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. 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 CK assumptions. As an example, Kantarcıoğlu et al. (2011) use a Stackelberg game in which both players know each other payoff functions.

In this paper we present a novel framework for AC based on Adversarial Risk Analysis (ARA), Rios Insua et al. (2009). This is an emergent paradigm supporting decision makers (DM) who confront adversaries in problems with random consequences that depend on the actions of all participants. ARA provides one-sided prescriptive support to a DM, maximizing her subjective expected utility, treating the adversaries' decisions as random variables. To forecast them, we model the adversaries' problems; however, our uncertainty about their probabilities and utilities is propagated leading to the corresponding random optimal adversarial decisions which provide the required distributions. ARA operationalizes the

Bayesian approach to games, Kadane and Larkey (1982) or Raiffa (1982), facilitating a procedure to predict adversarial decisions. Compared with standard game theoretic approaches, Hargreaves-Heap and Varoufakis (2004), ARA does not assume the standard CK hypothesis, according to which agents share information about utilities and probabilities. Thus, we propose ACRA, an AC approach based on ARA which stems from the pioneering Dalvi et al. (2004), but avoids CK assumptions.

ACRA is presented in Section 2, followed by a simple numerical example in Section 3 aimed at showcasing concepts. Section 4 presents computational issues illustrated with larger examples in Section 5. We end up with a discussion.

2 Adversarial Classification based on Adversarial Risk Analysis

We consider two agents, a classifier C (she) and an adversary A (he). C may receive two types of objects, denoted as malicious ($+$) or innocent ($-$); the type is designated Y. Objects have features X whose distribution depends on Y. A chooses an attack a which, applied to the features x, leads to $x' = a(x)$, actually observed by C. A general transformation from x to x' will be designated $a_{x \to x'}$. We focus just on exploratory attacks, defined to have no influence over the training data. In addition, as in Dalvi et al. (2004), we study cases in which A does not attack innocent instances ($y = -$), denominated integrity violation attacks. Huang et al. (2011) and Barreno et al. (2006) provide taxonomies of attacks against classifiers. Upon observing x', C needs to determine the object class y. Her guess $y_C = c(x')$ provides her with utility $u_C(y_C, y)$. She aims at maximizing expected utility, French and Rios Insua (2000). A also aims at maximizing his expected utility trying to confuse the classifier. His utility has the form $u_A(y_C, y, a)$, when C says y_C, the actual label is y and the attack is a, which has an implementation cost.

We aim at supporting C in choosing her classifications. We shall need to forecast the attacker's actions. For that we consider A's problem. As we lack CK, we shall model our uncertainty about A's beliefs and preferences and compute A's random optimal attack, obtaining the required forecasting distribution.

2.1 The classifier problem

We present first the problem faced by C as a Bayesian game, Kadane and Larkey (1982) and Raiffa (1982): we formulate a decision problem for C in which A's decision appears as random to the classifier, since she does not know how the adversary will modify data. Section 2.2 provides a procedure to estimate the corresponding probabilities making operational this approach. Suppose we are capable of assessing from the classifier:

1. $p_C(y)$, which describes her beliefs about the class distribution, with $p_C(+) + p_C(-) = 1$, and $p_C(+), p_C(-) \geq 0$.

2. $p_C(x|y)$, modeling her beliefs about the feature distribution given the class, when A is not present. Thus, we need $p_C(x|+)$ and $p_C(x|-)$. Since we focus on exploratory attacks, we can estimate $p_C(x|y)$ and $p_C(y)$ using the training data, which is clean by assumption.

3. $p_C(x'|a, x)$, which models her beliefs about the transformation results. We consider only deterministic transformations. Therefore, $p_C(x'|a, x) = I(x' = a(x))$, where I is the indicator function.

4. $u_C(y_C, y)$, describing C's utility when she classifies as y_C an instance whose actual label is y.

5. $p_C(a|x,y)$, describing C's beliefs about A's action, given x and y.

In addition, we assume that C is able to compute the set $\mathcal{A}(x)$ of possible attacks over instance x. Then, when she observes x', she could compute the set $\mathcal{X}' = \{x : a(x) = x' \text{ for some } a \in \mathcal{A}(x)\}$ of instances potentially leading to x'. She should then aim at choosing the class y_C with maximum posterior expected utility.

In our context, this problem is equivalent to finding

$$\begin{aligned} c(x') &= \arg\max_{y_C} \sum_{y\in\{+,-\}} u_C(y_C,y)p_C(y|x') = \arg\max_{y_C} \sum_{y\in\{+,-\}} u_C(y_C,y)p_C(y)p_C(x'|y) = \\ &= \arg\max_{y_C} \sum_{y\in\{+,-\}} u_C(y_C,y)p_C(y) \sum_{x\in\mathcal{X}'} \sum_{a\in\mathcal{A}(x)} p_C(x',x,a|y). \end{aligned}$$

Observe that the presence of A modifies $p_C(x'|y)$, preventing us from using the training set estimates of these elements. Thus, we need to take into account A's modifications through $p_C(x',x,a|y)$. Expanding the previous expression, we have that

$$\begin{aligned} c(x') &= \arg\max_{y_C} \sum_{y\in\{+,-\}} \left[u_C(y_C,y)p_C(y) \sum_{x\in\mathcal{X}'} \sum_{a\in\mathcal{A}(x)} p_C(x'|x,a,y)p_C(x,a|y) \right] = \\ &= \arg\max_{y_C} \sum_{y\in\{+,-\}} \left[u_C(y_C,y)p_C(y) \sum_{x\in\mathcal{X}'} \sum_{a\in\mathcal{A}(x)} p_C(x'|x,a)p_C(a|x,y)p_C(x|y) \right]. \end{aligned}$$

As we consider only integrity-violation attacks, we have that $p_C(a|x,-) = I(a = id)$, where id stands for the identity attack, leaving x unchanged and I is the indicator function. We finally have that the problem to be solved by C is

$$
\begin{aligned}
c(x') &= \arg\max_{y_C} \Big[u_C(y_C,+) p_C(+) \sum_{x \in \mathcal{X}'} \sum_{a \in \mathcal{A}(x)} I(x' = a(x)) p_C(a|x,+) p_C(x|+) + \\
&+ u_C(y_C,-) p_C(-) \sum_{x \in \mathcal{X}'} \sum_{a \in \mathcal{A}(x)} I(x' = a(x)) I(a = id) p_C(x|-) \Big] = \\
&= \arg\max_{y_C} \Big[u_C(y_C,+) p_C(+) \sum_{x \in \mathcal{X}'} p_C(a_{x \to x'}|x,+) p_C(x|+) + u_C(y_C,-) p_C(x'|-) p_C(-) \Big],
\end{aligned}
\tag{1}
$$

where $p_C(a_{x \to x'}|x,+)$ designates the probability that A will execute an attack that transforms x into x', when he receives $(x, y = +)$.

The ingredients 1-4 required above are standard. However, the fifth one $p_C(a_{x \to x'}|x,y)$ demands strategic thinking by C. To facilitate such forecast and make the approach operational, we next consider A's decision making problem.

2.2 The adversary problem

We assume that A aims at modifying x to maximize his expected utility by making C classify malicious instances as innocent. C's decision appears as random to A. Suppose for a moment that we have available from A:

1. $p_A(x'|a,x)$, describing his beliefs about the transformation results. As with C, $p_A(x'|a,x)$ $= I(x' = a(x))$.

2. $u_A(y_C, y, a)$, which describes the utility that A attains when C says y_C, the actual label is y and the attack is a, thus reflecting attack implementation costs.

3. $p_A(c(x')|x')$, which models A's beliefs about C's decision if she is to observe x'.

Let us designate by $p = p_A(c(a(x)) = +|a(x))$ the probability that A concedes to C saying that the instance is malicious, given that she receives $a(x) = x'$. Since he will have uncertainty about it, we denote its density by $f_A(p|a(x))$ with expectation $p^A_{a(x)}$. Among all attacks, A would then choose the attack maximizing his expected utility

$$a^*(x, y) = \arg\max_a \int \Big[u_A(c(a(x)) = +, y, a) \cdot p + u_A(c(a(x)) = -, y, a) \cdot (1 - p) \Big] f_A(p|a(x)) dp. \quad (2)$$

Since we assume that A does not change the data when $y = -$, we only consider the case $y = +$. Then, A's expected utility when he adopts attack a and the instance is $(x, y = +)$ will be

$$\int \Big[u_A(+, +, a)\ p + u_A(-, +, a)\ (1 - p) \Big] f_A(p|a(x)) dp = \\ = [u_A(+, +, a) - u_A(-, +, a)] p^A_{a(x)} + u_A(-, +, a).$$

However, the classifier does not know the involved elements u_A and $p^A_{a(x)}$ from the adversary. Suppose we may model her uncertainty through a random utility function U_A and a random expectation $P^A_{a(x)}$. Then, we may solve for the random optimal attack, optimizing the random expected utility

$$A^*(x, +) = \arg\max_a \Big([U_A(+, +, a) - U_A(-, +, a)]\ P^A_{a(x)} + U_A(-, +, a) \Big),$$

and C will make $p_C(a_{x \to x'}|x, +) = Pr(A^*(x, +) = a_{x \to x'})$, assuming that the set of attacks is discrete (and similarly in the continuous case).

In general, to approximate $p_C(a_{x \to x'}|x, +)$ we shall use simulation drawing K samples

$\left(U_A^k(y_C,+,a), P_{a(x)}^{A,k}\right)$, $k = 1,\ldots,K$ from the random utilities and probabilities, finding

$$A_k^*(x,+) = \arg\max_a \left(\left[U_A^k(+,+,a) - U_A^k(-,+,a)\right] P_{a(x)}^{Ak} + U_A^k(-,+,a)\right)$$

and making

$$\widehat{p_C}(a_{x\to x'} \,|\, x,+) \approx \frac{\#\{A_k^*(x,+) = a_{x\to x'}\}}{K}.$$

Of the required random elements, it is relatively easy to model the random utility $U_A(y_C,+,a)$ which would typically include two components. The first one refers to A's gain from C's decision. If we adopt the notation $Y_{y_C y}$ to represent the gain when C decides y_C and the true label is y, we may use: $-Y_{++} \sim Ga(\alpha_1,\beta_1)$ with $\alpha_1/\beta_1 = -d$, d being the expected gain for A and variance α_1/β_1^2 as perceived, thus assuming that the utility obtained by A when C classifies a truly malicious instance as malicious is negative; similarly, $Y_{-+} \sim Ga(\alpha_2,\beta_2)$ with $\alpha_2/\beta_2 = e$, the expected gain for A, and variance α_2/β_2^2 as perceived; finally, $Y_{+-} = Y_{--} = \delta_0$, the degenerate distribution at 0, therefore assuming that the adversary does not receive any utility from innocent instances. The second one refers to the random cost B of implementing an attack. Then, the gain of the attacker would be $Y_{y_C y} - B$. Finally, assuming that A is risk prone, French and Rios Insua (2000), the random utility would be (strategically equivalent to) $U_A(y_C,y,a) = \exp(\rho\,(Y_{y_C y} - B))$, with, say, $\rho \sim U[a_1,a_2]$, being $a_1,a_2 > 0$, the random risk proneness coefficient.

On the other hand, modeling $P_{a(x)}^A$, A's (random) expected probability that C declares an instance as malicious when she observes $x' = a(x)$, is more delicate. It entails strategic thinking, as C needs to understand his opponent's beliefs about what classification she will make when she observes the possibly transformed data x'. This could be the beginning of a hierarchy of decision making problems. Rios and Rios Insua (2012) provide a description of the potentially infinite regress in a simpler problem. We illustrate here the initial stage of such hierarchy in our context. First, A does not know the terms in (1). By assuming uncertainty over them through random distributions and utilities $P_C^A(+)$, $P_C^A(-)$, $P_C^A(x|+)$, $P_C^A(x'|-)$, $U_C^A(y_C,+)$, $U_C^A(y_C,-)$, $P_C^A(a_{x\to x'}|x,+)$, he would get the corresponding random optimal decision replacing the incumbent elements in (1) to obtain $P^A_{a(x)}$. However, this requires the assessment of $P_C^A(a_{x\to x'}|x,+)$ (what C believes that A thinks about her beliefs concerning the action he would implement given the observed data) for which there is a strategic component, leading to the next stage in the above mentioned hierarchy. One would typically stop at a level in which no more information is available. At that stage, we could use a non-informative prior over the involved probabilities and utilities.

For the first stage of the hierarchy, a relevant heuristic to assess $P^A_{a(x)}$ would base such assessment on the probability $Pr_C(c(x') = +|x') = r$ that C assigns to the object received being malicious assuming that she observed x', with some uncertainty around it. This could be implemented by making $P^A_{a(x)} \sim \beta e(\delta_1, \delta_2)$,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, which leads to adopting

$$\delta_1 = \left(\frac{1-r}{var} - \frac{1}{r}\right) r^2, \qquad \delta_2 = \delta_1 \left(\frac{1}{r} - 1\right). \tag{3}$$

Specifics would depend on case studies. In general, we could consider all attacks leading to x', differentiating between instances with original label $+$ and those with original label $-$; computing the probabilities of observing the malicious ones and adding them to obtain p_1; performing the same process with innocent ones to obtain p_2; and, finally using $r = p_1/(p_1+p_2)$.

3 A conceptual example in spam detection

We use a spam detection problem to illustrate ACRA. We have data referring to m emails characterized with the *bag-of-words* representation: binary features indicate the presence (1) or not (0) of n relevant words in a dictionary. Additionally, a label indicates whether the message is spam (+) or not (−). An email is, then, assimilated with an n-dimensional vector of 0's or 1's, together with a label. We consider only the so-called good word insertion attacks, adding at most one word (1-GWI), Lowd and Meek (2005b). This entails converting at most one of the 0's of the originally received message into a 1.

Given a message $x = (x_1, x_2, .., x_n)$, with $x_i \in \{0, 1\}$, we designate by $I(x)$ the set of indices such that $x_i = 0$. Then, the set of possible attacks is $\mathcal{A}(x) = \{a_0 = id; a_i, \forall i \in I(x)\}$, where a_i transforms the i-th 0 into a 1. In turn, given a message x' received by C, we designate by $J(x')$ the indices of features with value 1 in x'. If we designate by x'_j a message potentially leading to x', derived by changing the j-th 1 with a 0, the set of possible originating messages would be $\mathcal{X}' = \{x'; x'_j, \forall j \in J(x')\}$.

We also introduce the following notation: if the original label is −, the mail is innocent and the adversary does not change it, thus coinciding with the received one. We denote by $q_0 = p_C(x'|-)p_C(-)$ the probability of this event happening, according to C. If the original label is +, the mail is malicious, and A might change it in an attempt to fool the classifier. According to C, the original message x'_j happens with probability $q_j = p_C(x'_j|+)p_C(+), \forall j \in J(x')$. However, the adversary might decide not to attack even if the email is spam. According to C, this happens with probability $q_{n+1} = p_C(x'|+)p_C(+)$.

3.1 Classifier elements

The elements required to solve the classifier problem include $u_C(y_C, y)$ which is standard and $p_C(y)$ and $p_C(x|y)$, also standard if we just consider, as we do here, exploratory attacks. We could thus use our favorite probabilistic classifier to estimate them. Finally, $p_C(a_{x \to x'}|x, y)$ has a strategic component and we use ARA to approximate it.

3.2 Adversary elements

The adversary's random utilities follow the general arguments in Section 2.2. We also need to assess $P^A_{a(x)}$. We use the heuristic there proposed. If $J(a(x))$ designates the set of indices with feature value 1 in $a(x)$ then

$$r_a = \frac{\sum_{i \in J[a(x)]} q_i + q_{n+1}}{q_0 + \sum_{i \in J[a(x)]} q_i + q_{n+1}} \tag{4}$$

is the probability of C believing that the observation $a(x)$ has label $+$, when she is aware of the presence of A. Then, for such attack a, we could make $\delta_1^a/(\delta_1^a + \delta_2^a) = r_a$ and $(\delta_1^a \delta_2^a)/[(\delta_1^a + \delta_2^a)^2(\delta_1^a + \delta_2^a + 1)] = var$, and solve for δ_1^a and δ_2^a as in (3).

3.3 Implementation

The above ingredients allow us to implement the ACRA simulation-optimization scheme. Given x' (the received message) and var (the adjustable variance), assume that C has a procedure to compute the set $\mathcal{X}'$ of emails that may lead to x' and the q_i probabilities of the corresponding emails. We also need a basic routine to generate from the random utility function (routine 1, Appendix). In addition, for each $x \in \mathcal{X}'$, we need to estimate $p_C(a_{x \to x'}|x, +)$, as in routine 2 (Appendix). We, then, proceed as follows, by first estimating

$p_C(y)$ and $p_C(x|y)$ (preprocessing) and then operating

1. PREPROCESSING

   Train a probabilistic classifier to estimate $p_C(y)$ and $p_C(x|y)$, assuming that the training set has not been tainted.

2. OPERATION

   Read x'.

   ESTIMATE $p_C(a_{x \to x'}|x, +)$.

   Solve

$$c(x') = \arg\max_{y_C} \left[u(y_C, +)\widehat{p}_C(+) \sum_{x \in \mathcal{X}'} \widehat{p}_C(a_{x \to x'}|x, +)\widehat{p}_C(x|+) + u(y_C, -)\widehat{p}_C(x'|-)\widehat{p}_C(-) \right].$$

   Output $c(x')$.

3.4 Example

We illustrate ACRA in a spam filtering problem[1]. We test it against the utility sensitive naive Bayes (NB) classifier, a standard non-adversarial approach in this area, Song et al. (2009). We use the Spambase Data Set from the UCI Machine Learning repository, Lichman (2013). It consists of 4601 emails, out of which 1813 are spam. For each email, the database contains information about 54 relevant words. The *bag-of-words* representation with binary features assimilates each email with a vector of 0's and 1's of dimension 54. The dataset is divided into training and test sets, respectively taking 75% and 25% of the data. The

[1]For the sake of reproducibility, we provide the open source version of the code used for the examples at `https://github.com/roinaveiro/ACRA_spam_experiment.git`. The data is publicly available at `https://archive.ics.uci.edu/ml/datasets/spambase`.

results presented correspond to averages over 100 experiments performed under different training-test splits.

We trained a utility sensitive NB classifier using the training data, unaltered by assumption. To compare ACRA with NB on tampered data, we simulate attacks over the test set, solving the adversary problem (2) for each email in the test set removing the uncertainty not present from the attacker's point of view. The adversary's parameters were fixed at: $-U_A(+,+,a) \sim Ga(\alpha_1, \beta_1)$ with $E[-U_A(+,+,a)] = 5$ and $Var[-U_A(+,+,a)] = 0.01$, entailing $\alpha_1 = 2500$, $\beta_1 = 0.002$; $U_A(-,+,a) \sim Ga(\alpha_2, \beta_2)$, again with $\alpha_2 = 2500$, $\beta_2 = 0.002$; $U_A(-,-,a) = U_A(+,-,a) = \delta_0$. The random cost of implementing a particular attack a was set to $B = d(a) \cdot \alpha$, where $d(a)$ is the number of word changes (0 or 1) of attack a, and $\alpha \sim U[0.4, 0.6]$. The random risk proneness coefficient was set to $\rho \sim U[0.4, 0.6]$.

In order to obtain $P^A_{a(x)}$ for a given attack a, we need to generate from a beta distribution with mean $Pr_C(c(a(x)) = +|a(x))$, requiring its density to be concave in its support. Then, its variance must be bounded from above by $\min\left\{[r^2(1-r)]/(1+r), [r(1-r)^2]/(2-r)\right\}$. We fix the adjustable variance var at $k \in [0,1]$ times the upper bound. Note that the bigger k, the bigger C's uncertainty about A's behavior.

In addition, we fixed $K = 1000$, the Monte Carlo sample size in routine 2. We ran 100 experiments for each $k \in \{0.01, 0.1, 0.2, \cdots, 0.9\}$, with four different utility matrices for the classifier, one of them being the 0/1 utility. For the rest, we chose utility 1 for correctly classified instances and -1 for spam classified as legit. The penalty for classifying non-spam mail as spam was set, respectively, to -2, -5 and -10 in the other three cases. As performance metrics, we used averages of accuracy, utility, false positive (FPR) and false negative rates (FNR) over the experiments. We represent the results of the utility sensitive NB algorithm over the original untampered test set with a dashed blue line, and refer to them as NB-Plain. With red solid and green dotted lines, respectively, we present the results of ACRA and utility sensitive NB on the attacked test, referred to as ACRA and NB-Tainted, respectively. Obviously, NB-Plain and NB-Tainted metrics do not depend on k.

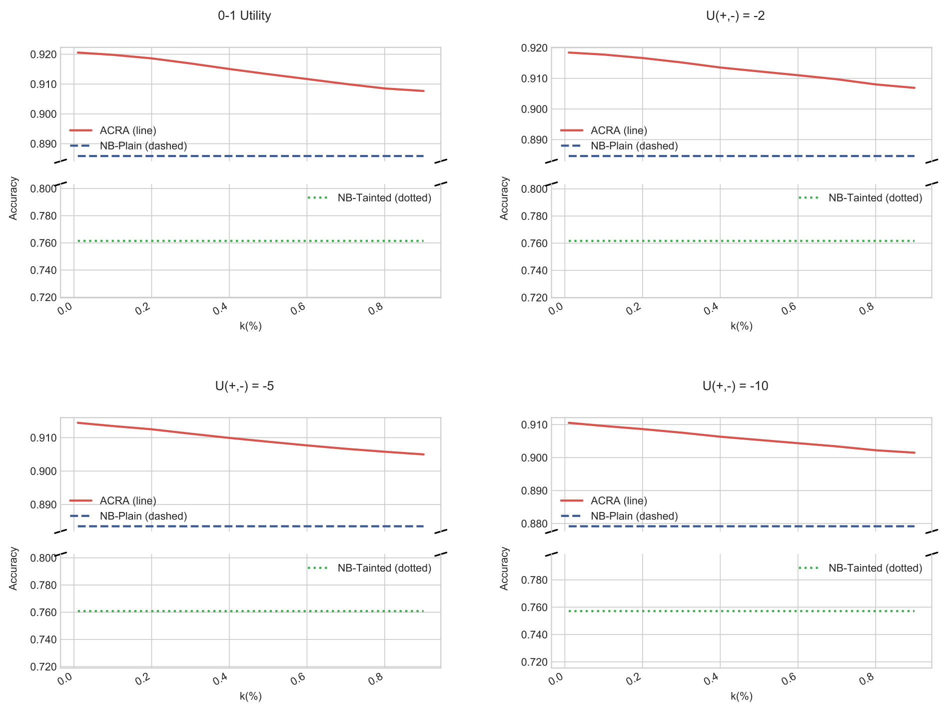


Figure 1: Average accuracy versus k for different utility matrices.

Figures 1 and 2 present average accuracies and utilities for various values of k and the four utility matrices. Observe first that the presence of an adversary significantly degrades NB performance both in accuracy and average utility, as NB-Plain is consistently above NB-Tainted. This one is still correctly classifying the same proportion of non spam as NB-Plain, as such emails have not been attacked. However, NB-Tainted is not able to identify a large proportion of spam emails due to the adversarial attacks. Consequently, as we increase the cost of misclassifying non-spam, reducing the relative importance of misclassifying spam, NB performance degrades. In contrast, ACRA is robust to attacks and identifies most of the spam. Its overall accuracy is above 0.9, thus identifying most non-spam emails. Observe also that ACRA degrades as k grows: the bigger k, the less precise

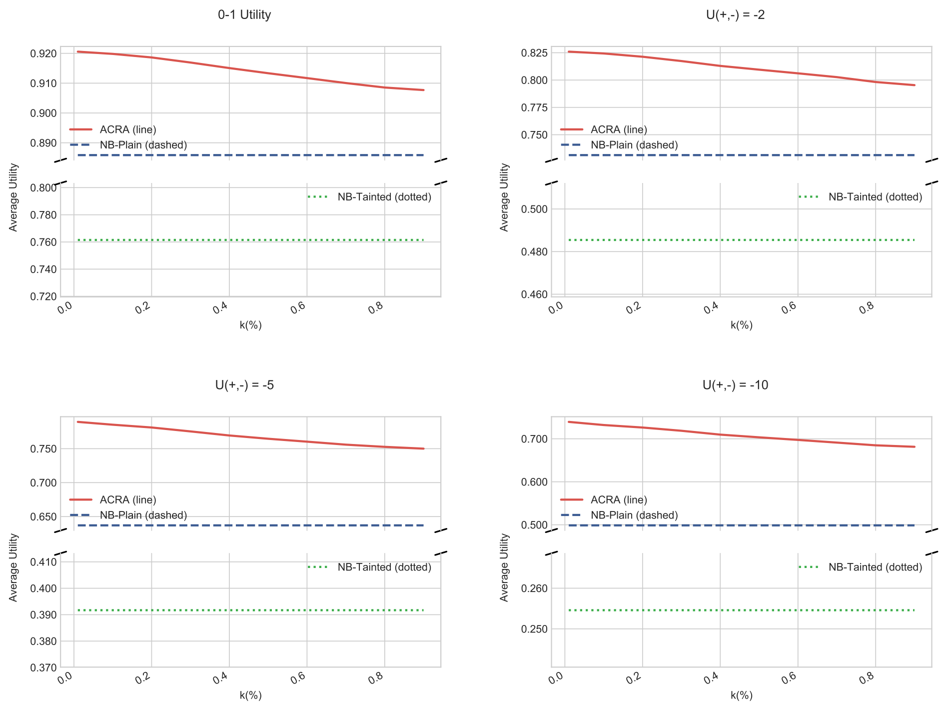


Figure 2: Average utility gain versus k for different utility matrices.

the knowledge that C has about A. Provided that the assumptions of C about A's behavior are correct, the classifier performance will degrade with k. One of our contributions here is that of providing parameters that may be tuned so as to adapt the knowledge that the classifier could have about her opponent. Note in Figures 1 and 2 that ACRA beats NB-Plain in accuracy as well as in utility: ACRA performs better on tainted data than utility sensitive NB on untainted data.

To better understand this result, we plotted FPR and FNR in Figures 3 and 4, re-

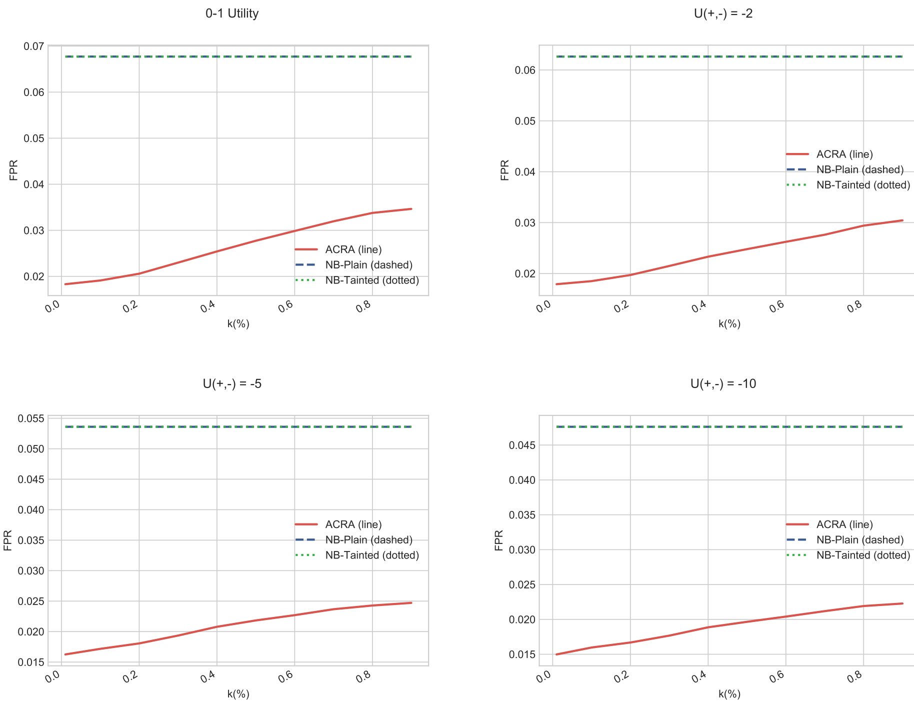


Figure 3: Average false positive rate versus k for different utility matrices.

spectively. FPR coincides for NB-Plain and NB-Tainted as the adversary is not modifying innocent instances. Both FPR and FNR grow with k for ACRA: the more the classifier knows about the adversary strategy, the better she protects, and lower FPR and FNR are attained. In addition, the increase of $u_C(+,-)$ raises the cost of false positives, reducing FPR at the expense of increasing FNR. Regarding comparison of different algorithms, observe that false negatives undermine the performance of NB on tampered data. In contrast, ACRA seems more robust, presenting smaller FPR and FNR than NB-Tainted. ACRA has also significantly lower FPR than NB-Plain, causing overall performance to raise up. The reason for this is that the adversary is very unlikely to apply the identity attack to a spam, as the cost difference between such attack and 1-GWI attacks is negligible in terms of utility gain. Then, for a legitimate email that NB-Plain classifies as positive, i.e. has high $p_C(x|+)$, ACRA will give a very low weight to $p_C(x|+)$, thus reducing the probability of classifying such email as spam. Reducing FPR is crucial in spam detection, as filtering a non-spam is typically more undesirable than letting spam reach the user. Surprisingly, ACRA also

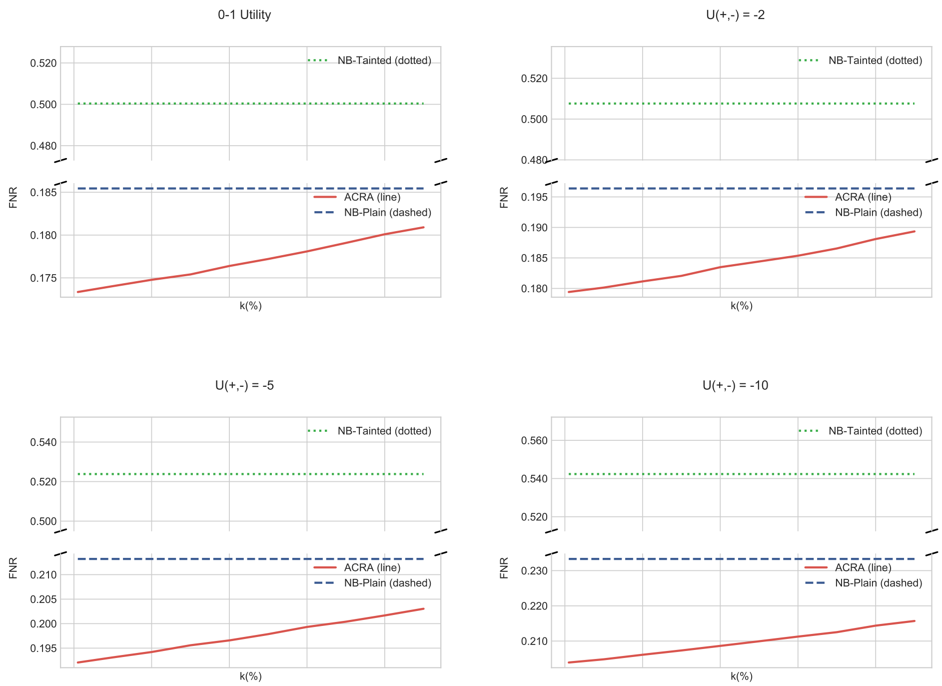


Figure 4: Average false negative rate versus k for different utility matrices.

has lower FNR than NB-Plain, specially for low values of k, although this improvement is not as remarkable as that in FPR. Both effects together produce ACRA to outperform utility sensitive NB both in tainted and untainted data: we enhance the classifier performance by taking advantage of the information we may have about the adversary, a core idea underlying ACRA.

4 Computational issues

The raw version of ACRA as presented above, may turn out to be extremely heavy computationally in some application domains.

Indeed, note first that $\mathcal{X}'$, the set of possible originating instances, is of combinatorial nature in certain applications. For instance, if we consider k-GWI attacks in spam filtering, the size of $\mathcal{X}'$ is $\mathcal{O}(n^k)$, being n the number of words that the adversary has access to. We may reduce its size and complexity through case-specific constraints about the adversary behavior. For example, in GWI we may reduce n assuming that some words cannot be modified by the adversary, or limit the number of words inserted, either explicitly, or implicitly through penalizing the insertion of additional words. We may store for later usage the sets once we have computed them for the first time.

Moreover, the estimation of probabilities $p_C(a_{x \to x'}|x, +)$ requires undertaking a computationally demanding simulation in the inner For loop in the Estimate $p_C(a_{x \to x'}|x, +)$ function (Appendix). For each simulation iteration, we need to consider the whole set $\mathcal{A}(x)$, which may be large, and solve the corresponding optimization problem. Finally, the summation to obtain C's expected utilities while operating is over $\mathcal{X}'$, which may be large. Moreover, each of the terms in the summation requires one of the earlier probabilities which, as we said, is computationally demanding.

We present now several suggestions that aid in alleviating such computational burden. Note first that the optimization problem (1) may be reformulated as setting $c(x') = +$ if and only if $\sum_{x \in \mathcal{X}'} p_C(a_{x \to x'}|x, +) p_C(x|+) > t$, where

$$t = \frac{\Big[u_C(-,-) - u_C(+,-)\Big] p_C(x'|-) p_C(-)}{\Big[u_C(+,+) - u_C(-,+)\Big] p_C(+)}.$$

One way to proceed with the left hand side summation would be through a Monte Carlo (MC) approximation. Should $\{x_n\}$ be a sample of size N from $p_C(x|+)$, the condition would be approximated through

$$I = \frac{1}{N} \sum_n p_C(a_{x_n \to x'}|x_n, +) I(x_n \in \mathcal{X}') > t. \tag{5}$$

A potential problem with this is that $p_C(x|+)$ for $x \in \mathcal{X}'$ is generally small, and a standard MC sample might contain no points in $\mathcal{X}'$. We could use importance sampling to mitigate this issue, Owen and Zhou (2000). We may use the restriction of $p(x|+)$ to $\mathcal{X}'$ as importance distribution. Let $\tilde{p}(x|+)$ be the probability distribution defined by

$$\tilde{p}(x|+) = \frac{p(x|+)}{Q} \cdot I(x \in \mathcal{X}')$$

with $Q = \sum_{x \in \mathcal{X}'} p(x|+)$. Then,

$$\sum_{x \in \mathcal{X}'} p_C(a_{x \to x'}|x, +) p_C(x|+) = \sum_{x \in \mathcal{X}'} \frac{p_C(a_{x \to x'}|x, +) p_C(x|+)}{\tilde{p}(x|+)} \tilde{p}(x|+) = Q \sum_{x \in \mathcal{X}'} p_C(a_{x \to x'}|x, +) \tilde{p}(x|+).$$

Now, if $\{x_n\}$ is a sample of size N from $\tilde{p}(x|+)$, we could approximate our target by

$$\tilde{I} = \frac{Q}{N} \sum_{n=1}^{N} p_C(a_{x_n \to x'}|x_n, +). \tag{6}$$

We should take though into account the inherent uncertainty in the above MC approximations when checking inequality (5). An estimate $\tilde{\Delta}$ of the standard deviation would be

$$\tilde{\Delta} = \sqrt{\frac{\sum_{i=1}^{N} Q^2 p_C^2(a_{x_i \to x'}|x_i, +) - N\tilde{I}^2}{N-1}}.$$

We would then declare $c(x') = +$ if $\tilde{I} - 2\tilde{\Delta} > t$. Observe that we could test this condition sequentially, before reaching the maximum size N allowed by our computational budget. By making $\tilde{I}_m$ and $\tilde{\Delta}_m$ depend on the sample size m, we would check sequentially whether

$$\tilde{I}_m - 2\tilde{\Delta}_m > t, \tag{7}$$

and eventually stop, with $\tilde{I}_m$ and $\tilde{\Delta}_m$ defined sequentially, since

$$\tilde{I}_m = \frac{(m-1)\tilde{I}_{m-1} + p_C(a_{x_m \to x'}|x_m, +)}{m}, \quad \tilde{\Delta}^2_m = \frac{(m-2)\tilde{\Delta}^2_{m-1} + Q^2 p^2_C(a_{x_m \to x'}|x_m, +) + (m-1)\tilde{I}^2_{m-1} - m\tilde{I}^2_m}{m-1}.$$

Another consideration refers to the estimation of $p_C(a_{x \to x'}|x, +)$. First, to speed up computations, we could use a relatively small MC size K in our function ESTIMATE $p_C(a_{x \to x'}|x, +)$. To avoid getting null probabilities, we could adopt a Dirichlet-multinomial model with non-informative prior $Dir(1, 1, ..., 1)$ over the probabilities of the attacks in $\mathcal{A}(x)$ and approximate

$$\widehat{p}_C(a_{x \to x'} \,|\, x, +) \simeq \frac{\#\{a^*_k = a_{x \to x'}\} + 1}{K + |(\mathcal{A}(x))|}.$$

Moreover, we could use a regression metamodel, Kleijnen (1992), based on approximating in detail $p_C(a_{x \to x'}|x, +)$ at a few pairs (x, x'), as allowed by our computational budget, fit a regression model $\psi(x, x')$ to $(x, x', \widehat{p}_C(a_{x \to x'}|x, +))$ and use it to replace $p_C(a_{x \to x'}|x, +)$ in the above expressions.

Last, but not least, ACRA is amenable of parallelization at least at two different levels. Observe first that the terms in summation (6) are independent. To improve performance, we may compute several batches of those terms in parallel by running different processes at different nodes of a cluster with multi-core processing. Whenever we use the sequential approach in (7), this parallelization strategy would require a master node that checks such condition periodically when the computation of any batch of terms finishes in the corresponding worker node. Finally, recall that computing terms in summation (6) entails a simulation. We could accelerate the computation of each simulation by running in parallel different processes for different batches of MC samples. Both parallelization strategies could be combined by sending different simulations to different nodes in a cluster, and parallelizing each simulation within the cores of each node.

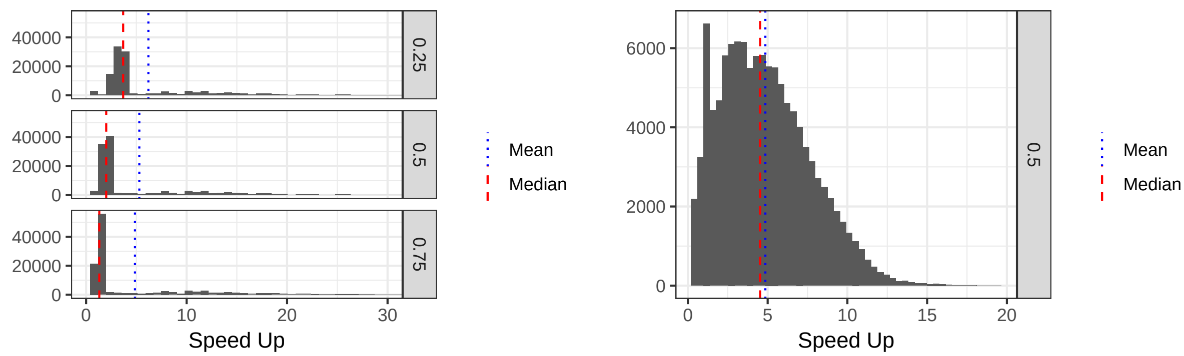
The combination of the above approaches alleviates the computational burden and largely makes ACRA computationally feasible as we show next.

5 Application

We illustrate several of the proposed enhancements with the example in Section 3.4: we test the ACRA framework using MC simulation (MC ACRA) with importance sampling and sequential strategy (7) against the raw algorithm (ACRA), as well as its parallelization. We tried different MC sample sizes, measuring them as proportions of the cardinal of $\mathcal{X}'$: e.g. an MC sample size of 0.5 corresponds to considering $|\mathcal{X}'|/2$ values. We fixed $k = 0.1$, and used the 0/1 utility. Table 1 shows the average performance metrics of both algorithms over 100 experiments. As the sample size increases, accuracy also increases. Nevertheless, we get fairly good results for relatively small sample sizes. For instance, with a 0.25 sample size we already beat NB on tainted data in accuracy and get comparable results to those obtained by NB on the original untampered test set. With 0.5 sample size we outperform NB on clean data. To compare execution times, we computed the speed up (the quotient between execution times of ACRA and MC ACRA over instances of all 100 experiments). Figure 5a presents the speed up histogram, Table 2 shows mean and median speed ups

	Size	Accuracy	FPR	FNR
ACRA	1.00	0.919	$1.87 \cdot 10^{-2}$	$1.77 \cdot 10^{-1}$
MC ACRA	0.75	0.912	$3.20 \cdot 10^{-2}$	$1.74 \cdot 10^{-1}$
MC ACRA	0.50	0.905	$2.70 \cdot 10^{-2}$	$1.99 \cdot 10^{-1}$
MC ACRA	0.25	0.885	$2.09 \cdot 10^{-2}$	$2.60 \cdot 10^{-1}$
NB-Plain	-	0.886	$6.77 \cdot 10^{-2}$	$1.85 \cdot 10^{-1}$
NB-Tainted	-	0.761	$6.77 \cdot 10^{-2}$	$5.00 \cdot 10^{-1}$

Table 1: Comparison between MC ACRA, raw ACRA and NB.



(a) (b)

Figure 5: Speed up histograms.

for each MC size. As expected, the median speed up is close to the inverse of the MC size, e.g. when size is 0.5, MC ACRA performs approximately twice faster than ACRA. Nevertheless, the speed up distributions (Figure 5a) are skewed to the right suggesting that MC ACRA performs much faster on average. This is due to the sequential rule in (7): for some instances, condition (7) is reached in a few iterations and consequently, over those instances MC ACRA performs much faster than ACRA.

We have also tested the first parallelization approach in Section 4, computing in parallel the terms in the MC approximation (6). We used a 16 core processor for this purpose. We

Size	Mean	Median
0.25	6.20	3.69
0.50	5.30	2.00
0.75	4.86	1.31

Table 2: Mean and median speed ups.

performed 100 experiments fixing $k = 0.1$, MC size to 0.5 and 0/1 utility. The histogram of speed ups is in Figure 5b. In this case, both the mean (4.856) and median (4.530) are close. We are not using the sequential approach in (7) and consequently, extreme values do not occur. Nevertheless, we obtain a huge improvement in time performance, almost 5 times faster both in mean and median.

	Dataset	Size	Accuracy	FPR	FNR
MC ACRA	UCI	0.5	0.904	$3.69 \cdot 10^{-2}$	$1.87 \cdot 10^{-1}$
NB-Plain	UCI	-	0.887	$6.56 \cdot 10^{-2}$	$1.87 \cdot 10^{-1}$
NB-Tainted	UCI	-	0.724	$6.56 \cdot 10^{-2}$	$6.01 \cdot 10^{-1}$
MC ACRA	Enron-Spam	0.5	0.824	$1.32 \cdot 10^{-1}$	$3.05 \cdot 10^{-1}$
NB-Plain	Enron-Spam	-	0.721	$2.83 \cdot 10^{-1}$	$2.68 \cdot 10^{-1}$
NB-Tainted	Enron-Spam	-	0.534	$2.83 \cdot 10^{-1}$	1.00
MC ACRA	Ling-Spam	0.5	0.958	$3.90 \cdot 10^{-2}$	$5.68 \cdot 10^{-2}$
NB-Plain	Ling-Spam	-	0.957	$4.00 \cdot 10^{-2}$	$5.75 \cdot 10^{-2}$
NB-Tainted	Ling-Spam	-	0.800	$4.00 \cdot 10^{-2}$	1.00

Table 3: Comparison between MC ACRA and NB under 2-GWI attacks.

The combination of all the approaches will induce considerable improvements rendering ACRA largely feasible, as we illustrate with the results of an experiment under 2-GWI attacks in different databases[2] in Table 3. As in previous examples, the results presented

[2] Besides the UCI Spam Data Set, we used the Enron-Spam Data Set at `https://www.cs.cmu.edu/~enron` and the Ling-Spam Data Set at

are averages over 100 experiments performed under different train-test splits. Observe that MC ACRA 0.5 consistently beats NB, both in clean and tainted test sets.

6 Discussion

Adversarial classification is an increasingly important problem area with many security applications. The pioneering work of Dalvi et al. (2004) has framed most of later approaches to the problem within the standard game theoretic context, in spite of the unrealistic required CK assumptions, actually questioned by the own authors. This motivated us to focus on an ARA perspective to the problem presenting ACRA, a general framework for adversarial classification that mitigates such assumption. Our framework is general in the sense that application-specific assumptions are kept to a minimum. In addition, we have provided computational enhancements that have significantly improved ACRA performance, allowing us to solve large problems and use it in operational settings.

Our framework may be extended in several ways. First of all, in our examples we have used NB as the basic classifier in the preprocessing phase. We could use other probabilistic classifiers, or even a mixture of them. We have only considered exploratory attacks, but we could extend the approach to take into account attacks over the training data, also called poisoning attacks, Biggio et al. (2012). In addition, we have just considered the case in which the attacker performs intentional attacks. In some problems, there could be, in addition, random attacks. The proposed framework could be extended to take those into account as well as to the case in which there are several attackers. We have concentrated on binary classification problems, but the extension to multi-label classification is relevant. Finally, we have illustrated the approach in spam detection, but other security areas like

http://csmining.org/index.php/ling-spam-datasets.html malware detection or fraud detection are truly important.

Note that in ACRA we go through a simulation stage to forecast attacks and an optimization stage to determine optimal classification. The whole process might be performed in a single stage, possibly based on augmented probability simulation, Bielza et al. (1999).

References

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

Bielza, C., Müller, P., and Rios Insua, D. (1999). Decision analysis by augmented probability simulation. *Management Science*, 45:995–1007.

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

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

Chen, C.-H. (2015). *Handbook of Pattern Recognition and Computer Vision*. World Scientific.

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

Efron, B. and Hastie, T. (2016). *Computer Age Statistical Inference*, volume 5. Cambridge University Press.

Fan, J., Feng, Y., Jiang, J., and Tong, X. (2016). Feature Augmentation via Nonparametrics and Selection (FANS) in high-dimensional classification. *Journal of the American Statistical Association*, 111:275–287.

French, S. and Rios Insua, D. (2000). *Statistical Decision Theory*. Wiley.

Hargreaves-Heap, S. and Varoufakis, Y. (2004). *Game Theory: A Critical Introduction*. Routledge.

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

Kadane, J. B. and Larkey, P. D. (1982). Subjective probability and the theory of games. *Management Science*, 28:113–120.

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

Kleijnen, J. P. (1992). Regression metamodels for simulation with common random numbers: comparison of validation tests and confidence intervals. *Management Science*, 38:1164–1185.

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

Li, B. and Vorobeychik, Y. (2014). Feature cross-substitution in adversarial classification. In *Advances in Neural Information Processing Systems*, pages 2087–2095.

Lichman, M. (2013). UCI Machine Learning Repository. http://archive.ics.uci.edu/ml.

Lowd, D. and Meek, C. (2005a). Adversarial learning. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, KDD '05, pages 641–647.

Lowd, D. and Meek, C. (2005b). Good word attacks on statistical spam filters. In *Proceedings of the Second Conference on Email and Anti-Spam (CEAS)*, volume 2005.

Owen, A. and Zhou, Y. (2000). Safe and effective importance sampling. *Journal of the American Statistical Association*, 95:135–143.

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

Raiffa, H. (1982). *The Art and Science of Negotiation*. Harvard University Press.

Rios, J. and Rios Insua, D. (2012). Adversarial Risk Analysis for Counterterrorism Modeling. *Risk analysis*, 32:894–915.

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

Song, Y., Kołcz, A., and Giles, C. L. (2009). Better naive bayes classification for high-precision spam detection. *Software: Practice and Experience*, 39:1003–1024.

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

Zeager, M. F., Sridhar, A., Fogal, N., Adams, S., Brown, D. E., and Beling, P. A. (2017). Adversarial learning in credit card fraud detection. In *Systems and Information Engineering Design Symposium (SIEDS), 2017*, pages 112–116. IEEE.

Zhou, X., Iversen Jr, E. S., and Parmigiani, G. (2005). Classification of missense mutations of disease genes. *Journal of the American Statistical Association*, 100:51–60.

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

Appendix

Routine 1

function GENERATE $U^k(y_C, +, a)$
  Generate $B^k \sim B$, $\rho^k \sim U[a_1, a_2]$
  Generate $Y^k_{++} \sim -Ga(\alpha_1, \beta_1)$, $Y^k_{-+} \sim Ga(\alpha_2, \beta_2)$
  $U^k(y_C, y, a) = \exp(\rho^k(Y^k_{y_C,y} - B^k))$
  return $U^k(y_C, y, a)$
end function

where k designates a generic sample instance in the Monte Carlo scheme.

Routine 2

function ESTIMATE $p_C(a_{x \to x'}|x, +)$
  for $x \in \mathcal{X}'$ **do**

Compute $\mathcal{A}(x)$

for $a \in \mathcal{A}(x)$ **do**

Compute $a(x)$ and r_a using (4)

Using r_a and var, compute δ_1^a, δ_2^a as in (3)

end for

for $k = 1, 2, \ldots, K$ **do**

for $a \in \mathcal{A}(x)$ **do**

Generate $U^k(y_C, +, a)$, $P_a^{Ak} \sim \beta e(\delta_1^a, \delta_2^a)$

Compute $\psi^k(a) = \left[U_A^k(+,+,a) - U_A^k(-,+,a)\right] P_a^{Ak} + U_A^k(-,+,a)$

end for

Compute $a_k^* = \arg\max_{a \in \mathcal{A}(x)} \psi^k(a)$

end for

Approximate

$$\widehat{p}_C(a_{x \to x'} \mid x, +) \simeq \frac{\#\{a_k^* = a_{x \to x'}\}}{K}$$

Store $\widehat{p}_C(a_{x \to x'} \mid x, +)$

end for

end function