Adversarial classification: An adversarial risk analysis approach

Roi Naveiro-Flores
Instituto de Ciencias Matemáticas
ICMAT-CSIC Campus Cantoblanco UAM
C/ Nicolás Cabrera, 13-15
28049 Madrid - Spain
roi.naveiro@icmat.es

Alberto Redondo
Instituto de Ciencias Matemáticas
ICMAT-CSIC Campus Cantoblanco UAM
C/ Nicolás Cabrera, 13-15
28049 Madrid - Spain
alberto.redondo@icmat.es

Fabrizio Ruggeri
CNR-IMATI
Via Alfonso Corti 12
20133 Milano - Italy
fabrizio@mi.imati.cnr.it

David Ríos Insua
Instituto de Ciencias Matemáticas
ICMAT-CSIC Campus Cantoblanco UAM
C/ Nicolás Cabrera, 13-15
28049 Madrid - Spain
david.rios@icmat.es

Abstract

Classification problems in security settings are usually contemplated as games in which one or more adversaries try to fool a classifier so as to obtain a benefit. Most attempts to such adversarial classification cases have focused on game theoretical approaches with strong underlying common knowledge assumptions, which are not realistic in security domains. We provide and discuss an alternative adversarial risk analysis framework which we illustrate with an example.

1 Introduction

Dalvi et al. [2004] provided an approach to enhance classification algorithms so as to deal with cases in which an adversary is present in the problem class they designated Adversarial Classification. They consider a case in which the classifier (*C*) first assumes that the data is untainted and computes her optimal classifier; then, the adversary (*A*) deploys his optimal attack against such classifier; subsequently, *C* implements the optimal classifier against this attack, and so on. As pointed out by the authors, a very strong assumption is made: all parameters of both players are known to each other. Although standard in game theory, this common knowledge assumption is actually unrealistic in security scenarios.

Stemming from Dalvi et al. [2004], there has been a large literature in the field, as reviewed e.g. in Biggio et al. [2014] or Li and Vorobeychik [2014]. Subsequent approaches expanding and developing this pioneering framework have mainly focused on analysing possible attacks over classification algorithms and assessing the security of such algorithms against those attacks, see e.g. Lowd and Meek [2005], Barreno et al. [2006], Zhou et al. [2012] or Biggio et al. [2014].

Despite that much work has been performed in classifier security assessment, only a few methods have been proposed aimed at modifying classification algorithms to be robust in adversarial environments. Most of them have focused on application-specific domains, for instance Kołcz and Teo [2009] on spam detection, with few general approaches to tackle this problem. Vorobeychik and Li [2014] address the problem studying the impact of randomization schemes over different classifiers against adversarial attacks, and propose an optimal randomization scheme as the best defense. Other general 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 assumption there mentioned. For instance, Kantarcıoğlu et al. [2011] treat the adversarial classification problem as a Stackelberg game in which both players know each other payoff functions.

In this paper we present an alternative novel framework for adversarial classification, based on Adversarial Risk Analysis (ARA), mitigating standard common knowledge assumptions. ARA, described in Rios Insua et al. [2009], is an emergent paradigm supporting decision makers (DM) who confront adversaries in problems with random consequences that depend on the actions of all participants. It 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' decision-making problems; however, the uncertainty about their probabilities and utilities is propagated within them, leading to the corresponding random optimal adversarial decisions which provide the required forecasting distributions. ARA thus operationalizes the Bayesian approach to games, as in Kadane and Larkey [1982], facilitating a procedure to forecast the adversarial decisions. Compared with the usual game theoretic approaches, ARA does not assume the standard common knowledge hypothesis, Hargreaves-Heap and Varoufakis [2004], according to which agents share information about their utilities and probabilities.

2 Adversarial Risk Analysis Classification

We introduce here ACRA, our ARA framework for adversarial classification not requiring strong common knowledge assumptions usual in earlier attempts. There are two agents relevant in our problem, a classifier C (she) and an adversary A (he). There are two types of objects that may be received by the classifier, respectively denoted as malicious $(+)$ or innocent $(-)$. Such type is designated by Y. The distribution X over the object features depends on the type $(X|Y)$. The attacker chooses an attack a which applied to the features x leads to $x' = a(x)$, actually observed by the classifier. A general transformation leading 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 just study cases in which the attacker does not deal with innocent instances ($y = -$), i.e. we restrict our attention to so-called integrity violation attacks. Huang et al. [2011] or Barreno et al. [2006] provide taxonomies of attacks against pattern classifiers.

Upon observing x', the classifier needs to determine the class y of the object. Her guess $y_c = c(x')$ provides her with a utility $u_c(y_c, y)$. She aims at maximising expected utility. The attacker also aims at maximising his expected utility by 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 classification decision. In doing this, we shall need to forecast the attacker's actions. For that we shall need to consider A's problem. As we lack common knowledge, we shall model our uncertainty about A's beliefs and preferences and obtain her optimal random attack, thus obtaining the required forecasting distribution.

2.1 The classifier problem

We start by formulating the problem faced by C as a Bayesian game, as in Kadane and Larkey [1982]. The decision to be made by A appears as random to the classifier, since she does not know how the adversary will modify the data he receives. C aims at maximizing her expected utility. Her utility u_c depends on the actual label y and her classifying decision $c(x') = y_c$. Suppose we are capable of assessing from the classifier: 1) $p_C(y)$ and $p_C(x|y)$, i.e. her beliefs about the class distribution and feature distribution given the class. We could estimate this distributions using the training data, which is clean by assumption; 2) $p_C(x'|a, x)$, her beliefs about the transformation results. We shall consider only deterministic transformations, i.e. $p_C(x'|a, x) = I(x' = a(x))$, where I is the indicator function; 3) $u_C(y_C, y)$, C's utility when she classifies as y_C an instance whose actual label is y; 4) $p_C(a|x, y)$, which describes C's beliefs about A's action, given x and y.

In addition, we assume that the classifier is able to compute the set $\mathcal{A}(x)$ of possible attacks over a given instance x. Thus, 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'. Then, she should choose the class

y_C with maximum posterior expected utility. In our context, this problem is equivalent to finding $c(x') = \arg\max_{y_C} \sum_{y\in\{+,-\}} u_C(y_C, y)p_C(y|x')$, that can be shown to be equivalent to (see section A.1 of the Appendix):

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

where $p_C(a_{x\to x'}|x, +)$, refers to the probability that A will execute the attack that transforms x into x', when he receives $(x, y = +)$. All the ingredients required in the expression above are standard, except for $p_C(a_{x\to x'}|x, y)$, as it demands strategic thinking by C. To facilitate such forecast, we now consider the adversary problem.

2.2 The adversary problem

We assume the adversary aims at modifying x to maximize his expected utility by making C classify malicious instances as innocent. In this case, C's decision appears as random to A. Suppose that we have available from the adversary: 1) $p_A(x'|a, x)$, which describes his beliefs about the transformation results. As for 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, which has some implementation cost; 3) $p_A(c(x')|x')$, which describes A's beliefs about the classifier decision when she observes 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 shall denote its density by $f_A(p|a(x))$, with expectation $p^A_{a(x)}$. The adversary would then choose, among all the attacks $a \in \mathcal{A}(x)$, that maximising his expected utility

$$\begin{aligned} a^*(x, y) &= \arg\max_a \int \Bigg[u_A(c(a(x)) = +, y, a)\, p + \\ &+ u_A(c(a(x)) = -, y, a)\, (1 - p) \Bigg] f_A(p|a(x))dp \end{aligned} \tag{2}$$

Since we assumed that A does not change the data when $y = -$, we only consider the case in which $y = +$. Then, problem (2) above can be shown to be equivalent to

$$a^*(x, y) = \arg\max_a \Bigg[u_A(+, +, a) - u_A(-, +, a) \Bigg] p^A_{a(x)} + u_A(-, +, a).$$

Once with it, we would go back to the classifier problem and plug the required estimate in it. However, C does not know the involved elements u_A and $p^A_{a(x)}$ from the adversary. Suppose we may model her uncertainty by a random utility function U_A and a random expectation $P^A_{a(x)}$. Then, given x, we may solve for the random optimal attack

$$A^*(x, +) = \arg\max_a \Big([U_A(+, +, a) - U_A(-, +, a)]\, P^A_{a(x)} + U_A(-, +, a) \Big). \tag{3}$$

The classifier just needs to 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 $p_C(a_{x\to x'} \mid x, +)$, one will typically use simulation, by drawing K samples $\big(U^k_A(y_C, +, a), P^{A,k}_{a(x)}\big)$, $k = 1, \ldots, K$, finding $A^*_k(x, +)$ using (3), and approximating

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

Of the required random elements, we model the random utilities using $U_A(y_c, y, a) = \exp(\rho\,(Y_{y_c y} - B))$ where $Y_{y_c y}$, modelled in turn through gamma distributions; represent A's gain when C's decision is y_C and the actual label is y, B is the random cost of implementing a particular attack, and ρ is the random risk proneness coefficient. We model the random expectation $P^A_{a(x)}$ assuming that it is beta-distributed with mean equal to the probability that the classifier assigns to the object being malicious, and adjustable variance var in order to adapt the knowledge that the classifier could have about her opponent. A detailed description about how each of these elements are modelled, is provided in section A.2 of the Appendix.

3 Application Example

In this section, we show the results of the application of the ACRA machinery to the domain of spam filtering, focusing on one good word insertion (1-GWI) attacks. We test the proposed approach against the utility sensitive naive Bayes (NB) classifier using the Spambase Data Set from the UCI Machine learning repository, Lichman [2013]. Specifics about implementation details and the data and parameters are provided in sections A.3 and A.4 of the Appendix, respectively[1].

We ran ten experiments for each of ten values of k, the adjustable parameter used to adapt the knowledge that the classifier could have about her opponent, from 0 to 0.9. We used four different utility matrices for the classifier, one of them being the 0/1 utility matrix. For the rest, we choose utility 1 for correctly classified instances and negative utility -1 for classifying a spam email as legit. The penalty for classifying a non-spam email as spam was set, respectively, to -1, -5 and -10. We use several performance metrics such as accuracy, average utility, false positive rate (FPR) and false negative rate (FNR). The results of the utility sensitive NB algorithm on the original untampered test set are referred to as NB-Plain. The results of the ACRA algorithm and utility sensitive NB on the attacked test set are referred to as ACRA and NB-Tainted, respectively. Specific details about how the test set was attacked are provided in section A.4 of the Appendix. As shown in Figures 1 and 2 of section A.6 of the Appendix, the presence of an adversary significantly degrades NB performance both in accuracy and expected utility. Another interesting fact is that ACRA performance decreases as k increases. This is to be expected since as k grows, the knowledge the classifier has about the adversary decreases. Provided that the classifier assumptions about the adversary behaviour matches the actual behaviour, the performance of the classifier will degrade with k. A major contribution of this work is precisely that of providing some parameters to tune so as to adapt the knowledge that the classifier could have about her opponent.

Something surprising about Figures 1 and 2 is 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 have plotted FPR and FNR in Figures 3 and 4 of the Appendix, respectively. Now, false positive and false negative rates grow with k, as we anticipated before. Observe that ACRA presents smaller FPR and FNR than NB-Tainted as expected. Nevertheless, ACRA has a lower FPR than NB-Plain but higher FNR, except for high negative values of $u_C(+,-)$, where ACRA has also lower FNR. The reason for this is that high negative values of $u_C(+,-)$ penalize more the false positives, thus increasing the FNR. This increase in FNR is more pronounced in NB-Plain than in ACRA. Nevertheless the decrease in FPR with $u_C(+,-)$ is greater within ACRA than in NB-Plain. The fact that the presence of an adversary systematically decreases the FPR was also observed by Dalvi et al. [2004]. The reason for this is that the adversary is very unlikely to apply the identity attack to a spam email, because the cost difference between such attack and 1-GWI attacks is negligible in terms of utility gain. Consequently, for a given non spam email that NB-Plain classifies as positive i.e. has a high $p_C(x|+)$, ACRA will be giving 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, since filtering out a non-spam email is highly undesirable compared to letting a spam email get through the user.

4 Discussion

Adversarial classification is a very important problem area with many applications in security settings. The pioneering work of Dalvi et al. [2004] has framed most of the approaches to the problem within the standard game theoretic framework, in spite of the required common knowledge assumptions actually questioned by the above mentioned pioneers. This has motivated us to focus on ARA perspective on the problem. The framework is general and may be extended in several ways. In our examples we have used NB as initial classifier, but we could use others, or even a mixture of them. We have only considered exploratory attacks, but we could extend the approach to other types of attacks. In addition, we have considered targeted attacks by a single attacker. We could combine untargeted attacks with targeted attacks by several attackers.
As presented above, ACRA may turn to be extremely heavy from a computational point of view. We provide several computational enhancements in the Appendix.

[1]For the sake of reproducibility, we have open source the code and data used for the application example. It is available at `https://github.com/roinaveiro/NIPS17_ACRA_spam_experiment.git`

A Appendix

A.1 The classifier problem

As mentioned in section 2.1, the classifier should choose the class y_C with maximum posterior expected utility. In our context, this problem is equivalent to finding

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

Expanding the previous expression, we have that

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

As we only consider integrity-violation attacks, we have that $p_C(a|x',-) = I(a = \textit{id})$, where I is the indicator function and *id* stands for the identity attack, that of leaving x unchanged. Using $p_C(x'|a,x) = I(x' = a(x))$, we finally have that the problem to be solved by the classifier is

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

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

A.2 Modeling $U_A(y_c,+,a)$ and $P^A_{a(x)}$.

Of the random elements required to solve (3), 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 the gain from C's decision. If we adopt the notation $Y_{y_C y}$ to represent A's gain when C's decision is y_C and the actual label is y, we may use:

- $-Y_{++} \sim Ga(\alpha_1, \beta_1)$ with $\frac{\alpha_1}{\beta_1} = -d$, d being the expected gain for A and variance α_1/β_1^2 as perceived. We assume that the utility obtained by the adversary when C classifies a truly malicious instance as malicious, is negative.
- $Y_{-+} \sim Ga(\alpha_2, \beta_2)$ with $\frac{\alpha_2}{\beta_2} = e$, the expected gain for A, and variance α_2/β_2^2 as perceived. We assume that the utility obtained by the adversary when C classifies a malicious instance as innocent, is positive.
- $Y_{+-} = Y_{--} = \delta_0$, the degenerate distribution at 0. We assume that the adversary does not perceive 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, we would have that the random utility would be (strategically equivalent to)

$$U_A(y_c, y, a) = \exp(\rho\,(Y_{y_c y} - B)),$$

with, say, $\rho \sim U[a_1, a_2]$, the random risk proneness coefficient.

On the other hand, the random expectation $P^A_{a(x)}$ is not easy to assess. 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 class of problems. We illustrate here just the next stage in such hierarchy for our situation. First, in expression (1) to be solved by the classifier, the adversary does not know the terms in the expected utility. By assuming uncertainty over them through random distributions and utilities $P^A_C(+), P^A_C(-), P^A_C(x|+), P^A_C(x'|-), U^A_C(y_C, +), U^A_C(y_C, -), P^A_C(a_{x\to x'}|x, +)$, he would get the corresponding random optimal decision replacing the corresponding elements. Observe that this requires the assessment of $P^A_C(a_{x\to x'}|x, +)$ (what the classifier believes that the adversary 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 hierarchy. One would typically stop at a level in which no more information is reasonably available. At that stage, one could use a non-informative prior over the involved probabilities and utilities.

At a first stage of the hierarchy, we could focus on characterising the adversary's analysis of the classifier through $P^A_{a(x)}$, the adversary (random) expected probability that the classifier declares an instance as malicious when she observes $x' = a(x)$. A relevant heuristic would be to base such assessment on the probability $Pr_C(c(x') = +|x') = r$ that the classifier assigns to the object being malicious when she observes x', and then adding some uncertainty around it. This could be implemented by making

$$P^A_{a(x)} \sim \beta e(\delta_1, \delta_2), \tag{4}$$

with mean $\frac{\delta_1}{\delta_1+\delta_2} = r$ and variance $\frac{(\delta_1\delta_2)}{(\delta_1+\delta_2)^2(\delta_1+\delta_2+1)} = var$ as perceived, leading to

$$\delta_1 = \left(\frac{1-r}{\text{var}^2} - \frac{1}{r}\right) r^2, \qquad \delta_2 = \delta_1\left(\frac{1}{r} - 1\right). \tag{5}$$

Specifics would depend on case studies.

A.3 Implementation Details

We illustrate in this section how to obtain the required elements to the application of the ACRA approach to the example of section 3 and highlight other implementation details. The data available refers to m emails characterised according to the *bag-of-words* features representation, with binary features indicating the presence (1) or not (0) of n relevant words in a given dictionary. Additionally, a label indicates whether the corresponding message is spam $(+)$ or not $(-)$. An email is, therefore, assimilated with a vector of 0's and 1's of dimension n, together with a label. We consider as only possible attack strategy the good word insertion (GWI) attack, limited to adding at most one word. This entails converting at most one of the 0's in the originally received message into a 1. Then, we have that:

- 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, let us designate by $J(x')$ the indices of the features with value 1 in x'. Then, 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 $\mathcal{X}' = \{x', x'_j : \forall j \in J(x')\}$.

A.3.1 Classifier elements

The elements required in the classifier problem include:

- $p_C(y)$ and $p_C(x|y)$. They are standard if we just consider, as we do here, exploratory attacks. We could thus use our favourite probabilistic classifier to come out with them.
- $u_C(y_C, y)$ is also standard.

The final component, $p_C(a_{x \to x'}|x, y)$, has a clear strategic value and we use an ARA approach to approximate it.

A.3.2 Adversary elements

The adversary's random utilities follow the general arguments in Section A.1. We also need to assess $P^A_{a(x)}$. We use the proposed heuristic (4). For a given a, the adversary may compute $a(x)$. If $J(a(x))$ is the set of indices with feature value 1 in $a(x)$ then

$$r_a = \frac{p_C(a(x)|+)p_C(+)}{\sum_{y \in \{+,-\}} p_C(a(x)|y)p_C(y)}, \tag{6}$$

is the probability of C believing that the observation $a(x)$ has label $+$, when she receives $a(x)$. Then, for such attack a, we could make $\frac{\delta_1^a}{\delta_1^a + \delta_2^a} = r_a$ and $\frac{(\delta_1^a \delta_2^a)}{(\delta_1^a + \delta_2^a)^2(\delta_1^a + \delta_2^a + 1)} = var$, and solve for δ_1^a and δ_2^a, as in (5).

A.3.3 Implementation

The ingredients above allow us to implement the proposed simulation-optimisation scheme for adversarial classification. Recall that for a given x' that C receives, the classifier aims at providing the label y_C of the original message.

Given x' (the received message) and var (the adjustable variance), assume C has a procedure to compute:

- The set $\mathcal{X}'$ of all possible emails that may lead to the observed x'.
- Estimate the required probabilities to compute r_a (6).

We also need a routine to generate from the random utility function, as follows:

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

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

We, then, proceed as follows

1. PREPROCESSING 1.
   Train your favourite algorithm to estimate $p_C(y)$ and $p_C(x|y)$ (assuming that the training set has not been tainted).

2. PREPROCESSING 2.
   For each $x \in \mathcal{X}'$, we must assess $p_C(a_{x \to x'}|x, +)$. We do it by simulation.
   Compute $\mathcal{A}(x)$. For $a \in \mathcal{A}(x)$:

```
    • Compute $a(x)$ and $r_a$ using (6).
    • Using $r_a$ and $var$, compute $\delta_1^a$, $\delta_2^a$.
  For $k = 1, ..., K$
    • For $a \in \mathcal{A}(x)$
      – Generate $U^k(y_C, +, a)$, $P_a^{Ak} \sim \beta e(\delta_1^a, \delta_2^a)$ .
      – Compute $\psi^k(a) = \left[U_A^k(+,+,a) - U_A^k(-,+,a)\right] P_a^{Ak} + U_A^k(-,+,a)$.
    • Compute $a_k^* = \arg\max_{a \in \mathcal{A}(x)} \psi^k(a)$.
  Then, approximate
```

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

```
3. OPERATION. (While PREPROCESSING 1 or 2 is not updated)
   Read $x'$
   Solve
```

$$\begin{aligned} c(x') &= \arg\max_{y_C} \left[u(y_C,+) p_C(+) \sum_{x \in \mathcal{X}'} p_C(a_{x \to x'}|x,+) p_C(x|+) dx + \right. \\ &+ \left. u(y_C,-) p_C(x'|-) p_C(-) \right] \end{aligned}$$

```
   Output $c(x')$.
```

A.4 Data and Parameters

For the application example of section 3, we use the Spambase Data Set from the UCI Machine learning repository, Lichman [2013]. This dataset consists of 4601 emails, out of which 1813 are spam (a 39.4% spam prevalence). For each email, the database contains information about 54 relevant words. We use the *bag-of-words* representation with binary features indicating the presence (1) or not (0) of such words, i.e. each email is assimilated with a vector of 0's and 1's of dimension 54.

We first divide the dataset into training and test sets, respectively taking 75% and 25% of the set for training and testing purposes. Each of the results presented is an average over 10 experiments performed under different training-test splits.

We trained a naive Bayes classifier using the training data, that remains unaltered by assumption. To test the ACRA algorithm against NB on tampered data, we need to simulate attacks over the test set. For this purpose, we solved the attacker's problem (2) for each email in the test set removing the uncertainty that is not present from the attacker's point of view. We restricted ourselves to one good word insertion (1-GWI) class of attacks: the attacker can add, at most, one word.

The adversary's required parameters were fixed to:

- $-U_A(+,+,a) \sim Ga(\alpha_1, \beta_1)$ with $E[-U_A(+,+,a)] = 5$ and *Var*$[-U_A(+,+,a)] = 0.1$. Therefore, $\alpha_1 = 250$, $\beta_1 = 0.02$.
- $U_A(-,+,a) \sim Ga(\alpha_2, \beta_2)$ with $E[U_A(-,+,a)] = 5$ and *Var*$[U_A(-,+,a)] = 0.1$. Therefore, $\alpha_1 = 250$, $\beta_1 = 0.02$.
- $U_A(-,-,a)$ and $U_A(+,-,a)$ are 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 of attack a, and $\alpha \sim U[0,1]$ is a realization of the uniform distribution.
- The random risk proneness coefficient was set to $\rho \sim U[0,1]$.
- In order to get $P_{a(x)}^A$ for a particular attack a, we need to generate from a beta distribution (4) whose mean value r is equal to $Pr_C(c(a(x)) = +|a(x))$. We force the beta distribution to be concave, and consequently its variance is bounded from above by $\min\left(\frac{r^2(1-r)}{1+r}, \frac{r(1-r)^2}{2-r}\right)$. We fix the adjustable variance var to be $k \in [0,1]$ times its upper bound.

A.5 Computational Enhancements

As presented in the body of the paper, ACRA may turn out to be extremely heavy from a computational point of view. Note that:

- $\mathcal{X}'$ is typically of combinatorial nature. Sometimes we may reduce its size and complexity by introducing constraints on the adversary behaviour. We may store the sets as required once we have computed them for the first time.
- The estimation of the probabilities $p_C(a_{x\to x'}|x,+)$ requires undertaking a computationally demanding simulation.
- The summation to obtain the classifier expected utilities is over $\mathcal{X}'$, which, as we said, may be large. Moreover, recall that each of the terms in the summation requires one of the earlier probabilities which, as we said, are very demanding computationally.

We present now several suggestions that are useful in coping with such computational burden. Note first that (1) may be reformulated as setting $c(x') = +$ if and only if

$$u_C(+,+)p_C(+)\sum_{x\in\mathcal{X}'} p_C(a_{x\to x'}|x,+)p_C(x|+) + u_C(+,-)p_C(x'|-)p_C(-) >$$
$$u_C(-,+)p_C(+)\sum_{x\in\mathcal{X}'} p_C(a_{x\to x'}|x,+)p_C(x|+) + u_C(-,-)p_C(x'|-)p_C(-),$$

or, equivalently, setting $c(x') = +$, if and only if

$$\sum_{x\in\mathcal{X}'} p_C(a_{x\to x'}|x,+)p_C(x|+) > t \tag{7}$$

with

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

One possible way to proceed with the simulation would be to use Monte Carlo (MC) simulation in evaluating (7). If $\{x_n\}$ is a sample of size N from $p_C(x|+)$, such condition would become

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

Note though that we should take into account the inherent uncertainty in this MC approximation. We could use an estimate of the uncertainty in the MC estimate; typically, through an estimate Δ of the standard deviation. We would declare then that $c(x') = +$, if

$$I - 2\Delta > t.$$

As a final comment, we could test the above condition sequentially, before reaching the maximum sample size N. By making I_m and Δ_m depend on the the sample size m, we would check sequentially

$$I_m - 2\Delta_m > t.$$

Note that I_m and Δ_m could be defined sequentially based on I_{m-1}, Δ_{m-1} and $\hat{p_C}(a_{x_m\to x'|x_m,+})$.

The final considerations refer to the estimation of $p_C(a_{x\to x'}|x,+)$ which entails a simulation. First, to speed up computations, we could use a relatively small MC size K and assume a Dirichlet-multinomial model with prior $Dir(1,1,...,1)$ over the attacks in $\mathcal{A}(x)$ (the set of all possible attacks over email x). Then, approximate

$$\widehat{p_C}(a_{x\to x'}\,|\,x,+) \simeq \frac{\#\{a_k^* = a_{x\to x'}\} + 1}{K + card(\mathcal{A}(x))}.$$

Finally, 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 the approximation

$$\widehat{I} = \frac{1}{N}\sum_n \psi(x_n, x'),$$

and similarly for Δ in the sequential rule above.

A combination of the above approaches alleviates the computational burden and largely makes ACRA computationally feasible, specially taking into account that many of the above proposals are amenable to parallelization.

In addition, note that we go through one simulation stage and one optimisation stage. It might be possible to perform the whole process in a single stage based on the augmented probability simulation approach in Bielza et al. [1999].

A.6 Figures

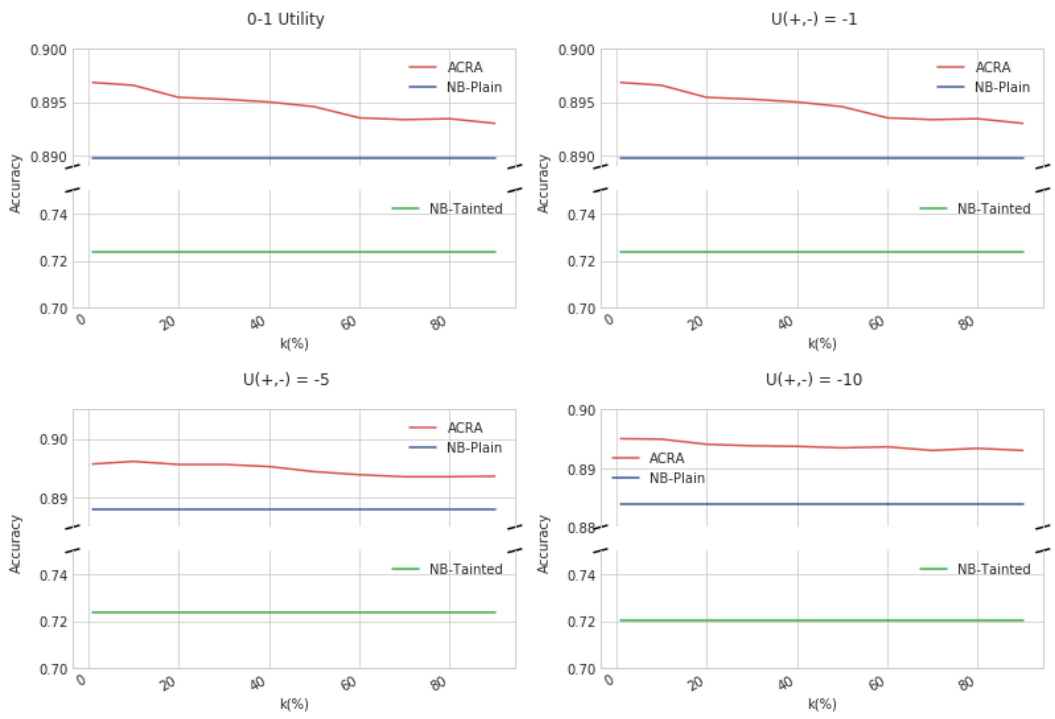

Figure 1: Accuracy versus k for different utility matrices.

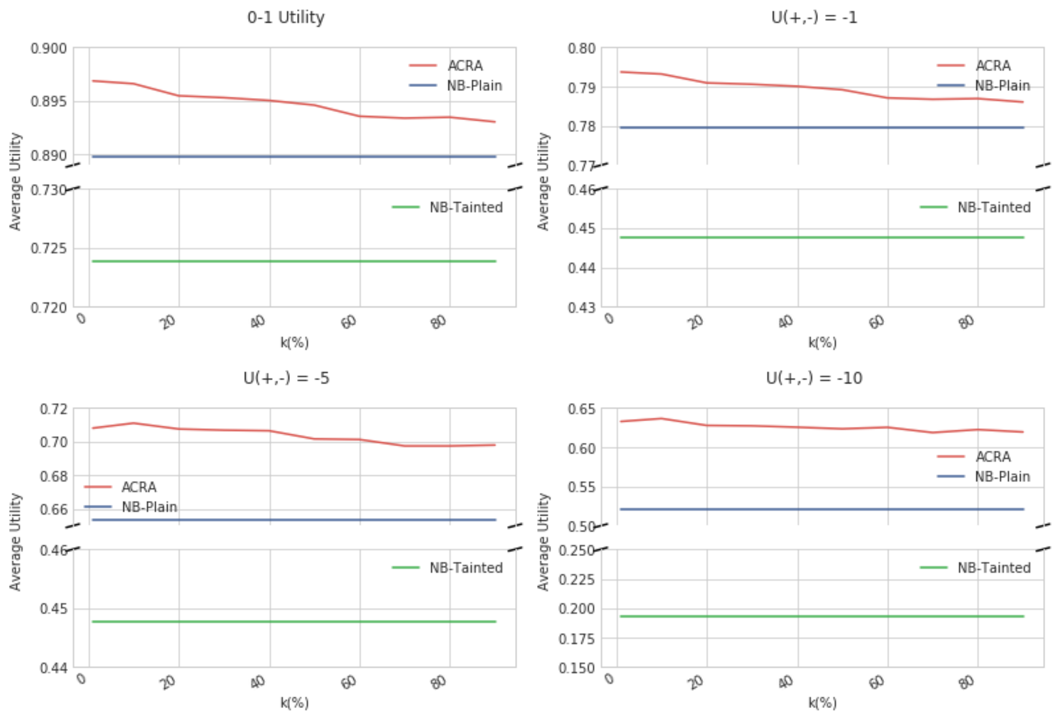

Figure 2: Average utility gain versus k for different utility matrices.

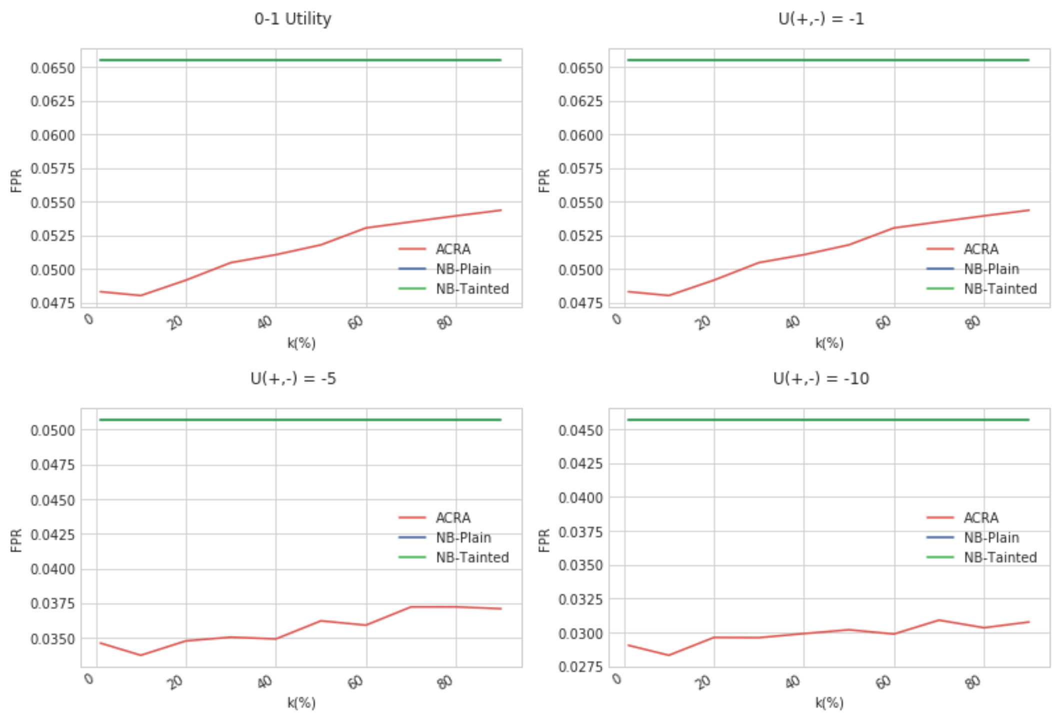

Figure 3: False positive rates versus k for different utility matrices. Note that FPR are the same for NB-Plain and NB-Tainted, because the adversary is not modifying negative instances.

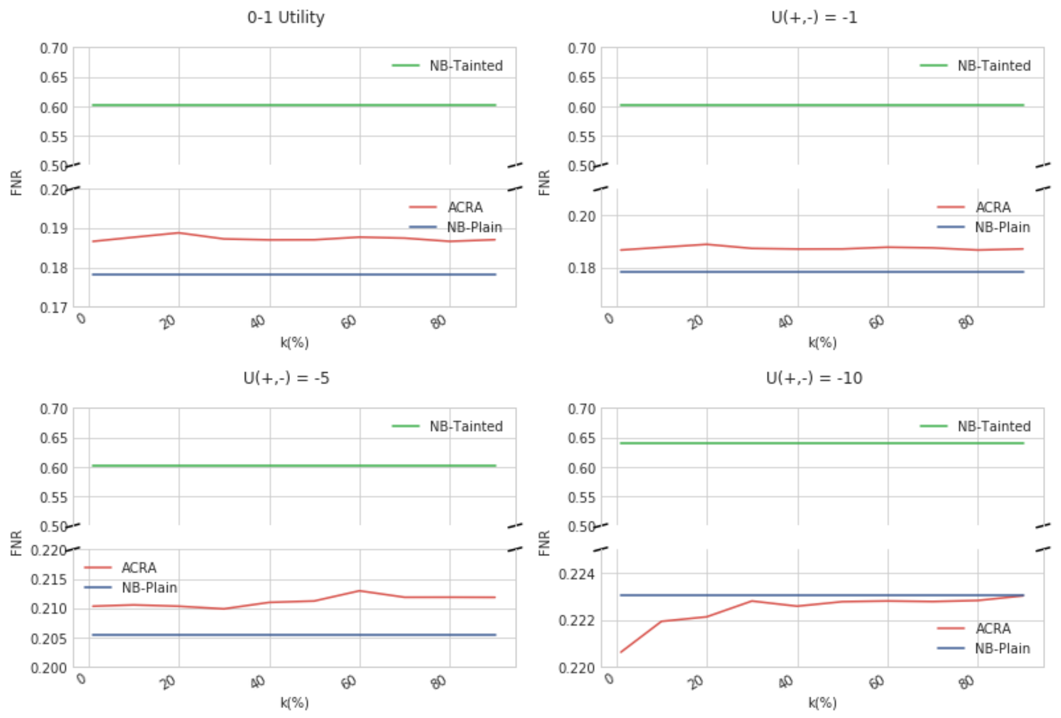

Figure 4: False negative rate versus k for different utility matrices.

References

N. Dalvi, P. Domingos, Mausam, S. Sumit, and D Verma. Adversarial Classification. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pages 99–108, New York, NY, USA, 2004. ACM. ISBN 1-58113-888-1.

B. Biggio, G. Fumera, and F. Roli. Security evaluation of pattern classifiers under attack. *IEEE Transactions on Knowledge and Data Engineering*, 26(4):984–996, April 2014. ISSN 1041-4347.

Bo Li and Yevgeniy Vorobeychik. Feature cross-substitution in adversarial classification. In *Advances in neural information processing systems*, pages 2087–2095, 2014.

D. Lowd and C. Meek. Adversarial Learning. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, KDD '05, pages 641–647, New York, NY, USA, 2005. ACM. ISBN 1-59593-135-X.

Marco Barreno, Blaine Nelson, Russell Sears, Anthony D. Joseph, and J. D. Tygar. Can Machine Learning Be Secure? In *Proceedings of the 2006 ACM Symposium on Information, Computer and Communications Security*, ASIACCS '06, pages 16–25, New York, NY, USA, 2006. ACM. ISBN 1-59593-272-0.

Yan Zhou, Murat Kantarcioglu, Bhavani Thuraisingham, and Bowei Xi. Adversarial Support Vector Machine Learning. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1059–1067. ACM, 2012.

Aleksander Kołcz and Choon Hui Teo. Feature Weighting for Improved Classifier Robustness. 2009.

Yevgeniy Vorobeychik and Bo Li. 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, Richland, SC, 2014. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 978-1-4503-2738-1.

Murat Kantarcıoğlu, Bowei Xi, and Chris Clifton. Classifier Evaluation and Attribute Selection Against Active Adversaries. *Data Mining and Knowledge Discovery*, 22(1):291–335, 2011.

Rios Insua, J. Rios, and D. Banks. Adversarial Risk Analysis. *Journal of the American Statistical Association*, 104(486):841–854, 2009.

Joseph B Kadane and Patrick D Larkey. Subjective probability and the theory of games. *Management Science*, 28(2):113–120, 1982.

Shaun Hargreaves-Heap and Yanis Varoufakis. *Game Theory: A Critical Introduction*. Routledge, 2004.

Ling Huang, Anthony D. Joseph, Blaine Nelson, Benjamin I.P. Rubinstein, and J. D. Tygar. Adversarial Machine Learning. In *Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence*, AISec '11, pages 43–58, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-1003-1.

M. Lichman. UCI machine learning repository, 2013. URL `http://archive.ics.uci.edu/ml`.

Jack PC Kleijnen. Regression metamodels for simulation with common random numbers: comparison of validation tests and confidence intervals. *Management Science*, 38(8):1164–1185, 1992.

Concha Bielza, Peter Müller, and David Ríos Insua. Decision analysis by augmented probability simulation. *Management Science*, 45(7):995–1007, 1999.