Evade Hard Multiple Classifier Systems

Battista Biggio and **Giorgio Fumera** and **Fabio Roli** [1]

Abstract. Multiple classifier systems are widely used in security applications like biometric personal authentication, spam filtering, and intrusion detection in computer networks. Several works experimentally showed their effectiveness in these tasks. However, their use in such applications is motivated only by intuitive and qualitative arguments. In this work we give a first possible formal explanation of why multiple classifier systems are harder to evade, and therefore more secure, than a system based on a single classifier. To this end, we exploit a theoretical framework recently proposed to model adversarial classification problems. A case study in spam filtering illustrates our theoretical findings.

1 Introduction

The effectiveness of multiple classifier systems has been proven in several real applications [4, 3]. Various authors showed that using classifier ensembles can allow to improve the detection capability in security applications like biometric authentication, and intrusion detection in computer networks. The classifier ensemble approach is also used in commercial and open source spam filters. A short survey of the literature on these applications is reported in section 2. For the purposes of this work, it is important to note that the use of multiple classifier systems have been also proposed to improve the so called "hardness of evasion" of a security system, namely, to increase the effort that the attacker has to do to evade the system [6]. However, the main motivations proposed so far in favour of the classifier ensemble approach in security applications are indeed based on qualitative and intuitive arguments, besides the experimental evidence. In particular, to our knowledge, no work tried to develop formal arguments to analyse the improvements attainable by multiple classifier systems on the hardness of evasion.

The aim of this work is to make a first step in this direction. We consider a theoretical framework recently proposed in [1] to model *adversarial classification* problems, namely, problems in which a system based on machine learning techniques is used against an *adaptive* adversary which can modify malicious patterns to evade the system. This is clearly the case of many security applications, including the ones mentioned above. This framework is summarised in section 3.1. In section 3.2 we use it to model the particular case in which the system is made up of an ensemble of classifiers, and new classifiers can be added to the system in response to adversarial actions aimed at evading it. This allow us to give a possible formal explanation of why multiple classifiers are harder to evade than a single classifier. These findings are then experimentally investigated in section 4 with a case study in spam filtering, using the SpamAssassin open source spam filter and a real corpus of legitimate and spam e-mails.

[1] Dept. of Electrical and Electronic Engineering, University of Cagliari, Piazza d'Armi, 09123 Cagliari, Italy, e-mails: {battista.biggio, fumera, roli}@diee.unica.it

2 Previous works on multiple classifiers for security applications

The use of classifier ensembles to improve the detection accuracy or the hardness of evasion has been recently proposed by several authors in different security applications [4, 3, 6, 2, 7]. It also turns out that the design of well-known spam filters like SpamAssassin (`http://spamassassin.apache.org`) and intrusion detection systems (IDSs) like Snort (`http://www.snort.org/`) implicitly follows the approach based on combining an ensemble of detectors.

SpamAssassin and Snort are based on a similar rationale. They consist of a set of classification rules ("test") in the if-then form, which analyse different characteristics of input patterns (respectively e-mails and network packets) to detect the presence of "signatures" denoting a malicious origin of the pattern. Each test outputs a score which denotes the "likelihood" that the pattern is malicious. Tests are often unrelated to each other, in the sense that they are based on unrelated characteristics of the input pattern. In some cases, they are focused on specific signatures of known attacks. In the case of SpamAssassin, the scores provided by the tests which *fired* (namely, the ones whose antecedent part is true for the processed e-mail) are summed up. The final decision about the input pattern (to be labelled either as malicious or as innocent) is taken by thresholding the sum of the scores. [2] This architecture makes easy to add new tests (i.e., binary classification rules) or delete old ones, to keep SpamAssassin and Snort updated and to customise them to the specific traffic of the network on which they operate. We point out that SpamAssassin and Snort can be considered as instances of classifier ensembles. Each rule can indeed be viewed as a single classifier based on a specific set of features, whose score is then fused with the ones of the other classifiers through the well known sum rule [4]. This kind of ensemble-like modular architecture is used also in commercial spam filters and IDSs.

The classifier ensemble approach has been explicitly investigated in the literature of IDSs, as an alternative to the approach based on a "monolithic" classifier [2, 6]. The ensemble architecture proposed in [2] and [6] is similar: it consists in fusing classifiers each trained on a distinct feature representation of patterns. The main motivations are derived from the field of classifier ensembles. It is indeed known that, if different sets of heterogeneous and loosely correlated features are available, as happens for IDSs, combining the outputs of different classifiers trained on different feature sets can be more effective than

[2] Actually Snort rules give a boolean output, and the decision function is a logic OR between all the outputs. Nevertheless, this is equivalent to having a binary score, say 0 and 1, and to threshold the sum of such scores with a value between 0 and 1.

designing a single classifier in the feature space made up of all the available features (see for instance [4]). Moreover, if the overall number of features is large, such a single classifier would be more prone than a classifier ensemble to the so-called curse of dimensionality problem. In [2] it was also pointed out that the ensemble approach "reflects the behaviour of network security experts, who usually look at different traffic statistics in order to produce reliable attack signatures". The above arguments qualitatively support the use of classifier ensembles to improve the effectiveness of an IDS, i.e. to attain both low false alarm rates and high attack detection rates. In [6] it was also pointed out that classifier ensembles can allow to improve the hardness of evasion. The reason is that fusing classifiers that work on different feature spaces "forces the attacker to devise a mimicry attack that evades multiple models of normal traffic at the same time, which is intuitively harder than evading just one model".

Classifier ensembles have been very investigated also in biometric applications, in which they can be straightforwardly exploited in several ways [7]. For instance, it is intuitive that using different biometric traits (like face, fingerprints, and speech) the recognition accuracy of a system as well as its hardness of evasion can be improved. With regard to the hardness of evasion, the use of multiple classifiers using different biometric traits strongly discourages fraudulent attempts to deceive personal identity verification systems. In fact, deceiving a multi-modal biometric system would require the construction of different kinds of fake biometric traits, which is a very challenging task. However, no work theoretically analysed the hardness of evasion of biometric systems that use the classifier ensemble approach.

As can be seen from the above overview, so far the hardness of evasion of security systems based on MCSs is not motivated theoretically, but is rather intuitively suggested by the fact that the MCS architecture naturally allows adding new classifiers (e.g., new filtering rules in SpamAssassin) in response to adversarial actions aimed at evading it, and experimental evidences show that adding new classifiers, using different features, makes more difficult for the attacker to evade the system. Experience and intuition also suggest the designer of these security systems that the characteristics which allow detecting malicious patterns can be very different and heterogeneous, and can change over time due to new tricks used by spammers and hackers to defeat spam filters and IDSs. Also this motivates the heuristic strategy commonly used to increase the hardness of evasion by adding new classifiers using different input features. In the next section we will address this open issue, providing a first theoretical justification of the classifier ensemble approach as tool for improving the hardness of evasion.

3 Why are multiple classifiers harder to evade?

In the literature, the first attempt to formalise a scenario in which an adaptive adversary tries to defeat a system based on machine learning techniques was made in [1]. The authors formalised this class of problems under the statistical framework of the minimum risk theory, as a two-class classification problem in which a classifier has to discriminate between positive (malicious) and negative (innocent) instances (e.g., spam and legitimate e-mails, genuine or impostor users in access control systems, attack or normal packets in computer network traffic, etc.), while the adversary can modify either training or testing instances to defeat the classifier. Assuming that both the classifier and the adversary are guided by utility and cost functions, the model proposed in [1] allows to show how the adversary should compute the optimal strategy for modifying instances, and how the classifier should take this into account by modifying its decision function accordingly. This can also allow to evaluate the effectiveness of a given strategy followed by the classifier to improve its detection capability and its hardness of evasion. This analytical framework is summarised in section 3.1. In section 3.2 we will exploit it to analyse the case when the system is made up of an ensemble of classifiers working on different feature sets, and the decision function is obtained by thresholding the sum of the scores provided by each classifier. This classifier architecture is commonly used in spam filters, IDSs and biometric authentication systems, as described in section 2. Our aim is to give a support based on a formal model to the strategy used to improve the detection capability and the hardness of evasion in these kinds of systems, based on adding new classifiers each working on homogeneous features, different from the ones used by other classifiers.

3.1 A theoretical framework for adversarial classification

When machine learning or pattern recognition techniques are used in applications like spam filtering, intrusion detection, biometric authentication, etc., their task can be formalised as a two-class classification problem. Denoting with y the class label, instances belong either to a positive ($y = +$) or to a negative class ($y = -$). The positive class is made up of malicious instances, while innocent or legitimate instances belong to the negative class. Instances are considered as realisations of a random variable X, and are represented as vectors of N feature values which are random variables themselves, $(X_1, \ldots, X_i, \ldots, X_N)$. A realisation of such random variable is denoted as $x = (x_1, \ldots, x_i, \ldots, x_N)$, where x_i is a possible value for feature X_i. It is assumed that instances are generated i.i.d. according to a given distribution $P(X)$, which can be rewritten as $P(X) = P(X|+)P(+) + P(X|-)P(-)$. The set of all possible realisations of X defines the feature space $\mathcal{X}$. In [1] it is assumed that the adversary can modify only positive instances at the operation phase, to make them being misclassified as legitimate by the classifier, but it can not modify any negative instance nor positive instances belonging to the training set. We point out that this assumption is true for several real applications. For instance, in the spam filtering task, where instances correspond to e-mails, spammers can modify only positive instances (their own e-mails) to evade the anti-spam filter, but they can not modify legitimate e-mails or any training instance, given that the training process is usually carried out *offline*, over a hand-labelled corpora of e-mails. There are however some cases, like IDSs trained *online*, in which the adversary can modify training instances: the model in [1] can not be directly applied to these cases.

In [1] it is then assumed that the classifier and the adversary act according to given utility and cost functions. Denoting with $y_C(x)$ the label given to the instance x by the classifier, the utility function of the classifier is represented as $U_C(y_C, y)$. It is reasonable to assume that such utility is positive for correctly classified instances and negative for misclassified ones, that is, $U_C(+,+) > 0$, $U_C(-,-) > 0$, $U_C(+,-) \leq 0$ and $U_C(-,+) \leq 0$. The cost for the classifier is assumed to be due to measuring each feature. The cost for measuring the i-th feature is denoted as V_i. Finally, it should be taken into account that the adversary could modify any positive instance x using some function $\mathcal{A}(x)$. For example, in spam filtering the function $\mathcal{A}(x)$ could be implemented by adding words or using synonyms in the spam mails. It follows that the expected utility for the classifier is given by

$$U_C = \sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}} P(x,y)[U_C(y_{\mathcal{C}}(\mathcal{C}(\mathcal{A}(x))),y) - \sum_{i=1}^{N} V_i], \quad (1)$$

where $\mathcal{Y} = \{+,-\}$ and $P(x,y)$ is the joint probability of pattern x being generated with true label y (note that, by the above assumptions, $\mathcal{A}(x) = x$ if $y = -$, namely for each negative instance).

Similarly, denoting with $U_A(y_c, y)$ the utility function of the adversary, it is assumed that it is positive for positive instances misclassified by the classifier as legitimate ($U_A(-,+) > 0$), negative for correctly classified positive instances ($U_A(+,+) \leq 0$), and zero for negative instances ($U_A(-,-) = U_A(+,-) = 0$) whatever the label assigned by the classifier, namely, the adversary utility is not affected by the classification of negative instances. The cost for the adversary is that faced for modifying an instance x according to the function $\mathcal{A}(x)$ defined above. It is assumed that the cost is given by $W(x,\mathcal{A}(x)) = \sum_{i=1}^{N} W_i(x,\mathcal{A}(x))$, being W_i the cost for modifying the i-th feature. Of course, $W_i = 0$ if the i-th feature is not changed, $W_i > 0$ otherwise. The expected utility for the adversary is thus

$$U_{\mathcal{A}} = \sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}} P(x,y)[U_A(y_{\mathcal{C}}(\mathcal{C}(\mathcal{A}(x))),y) - W(x,\mathcal{A}(x))]. \quad (2)$$

Using the above model, the adversarial classification problem is formulated as a game between classifier and adversary, in which the two players make one move at a time. A move by the classifier consists in choosing a decision function $y_{\mathcal{C}}(\cdot)$, taking into account both the training set and any knowledge it may have about the strategy $\mathcal{A}(\cdot)$ defined in the previous move by the adversary. Analogously, a move by the adversary consists in choosing a strategy $\mathcal{A}(\cdot)$, taking into account the available knowledge about the decision function chosen by the classifier at the previous move. Although game theory could in principle be applied to find the optimal sequence of moves by both players according to their utility and cost functions, it was shown in [1] that this is computational intractable, and anyway it requires the knowledge of the distribution $P(x,y)$, which is unrealistic. Therefore, a simplified single-shot version of the game was considered in [1]. Initially, classifier constructs a decision function using its training set, assuming that training instances are untainted (i.e., $\mathcal{A}(x) = x$). Then the adversary chooses his strategy $\mathcal{A}(\cdot)$ with the aim of maximising his expected utility (given by eq. 2), assuming perfect knowledge of the decision function chosen by the classifier and of its utility and cost functions. Finally, classifier moves again by choosing a new decision function, taking into account the move by the adversary and assuming to know his utility and cost functions. We point out that the optimal adversary strategy consists in defining, for each positive instance x, a modification x' which maximises the corresponding summand of the adversary expected utility in eq. 2, that is:

$$\mathcal{A}(x) = \underset{x'\in\mathcal{X}}{\text{argmax}}[U_A(y_{\mathcal{C}}(\mathcal{C}(x'),+) - W(x,x')]. \quad (3)$$

Given the above definition of the adversary utility and cost functions, it is easy to see that the adversary will change a given instance x, only if it is correctly classified by the classifier as positive, and if there is any instance $\mathcal{A}(x) \neq x$ which is misclassified by the classifier as negative, and the modification cost $W(x,\mathcal{A}(x))$ is lower than the utility gain $U_A(-,+) - U_A(+,+)$. Otherwise, it turns out that the best strategy is to leave the instance x unchanged, namely $\mathcal{A}(x) = x$.

In [1] the above framework was applied to find the optimal adversary and classifier strategies, when the classifier is a Naive Bayes, under the standard assumption of game theory that both players have perfect knowledge of each other, namely each one knows the utility and cost functions of the other one, and the adversary also knows the classification algorithm and the training set used by the classifier. The corresponding adversarial classification system was then experimentally evaluated on a spam filtering task, quantitatively showing that the classifier effectiveness significantly degrades if the adversarial nature of this task is not taken into account, while an adversary-aware classifier can perform significantly better.

3.2 Multiple Classifier Systems for Adversarial Classification

The framework for adversarial classification proposed in [1] applies also to the case we are interested in, namely a system made up of an ensemble of N classifiers working on different feature sets, whose decision function consists in thresholding the sum of the scores provided by each classifier. To this aim, denoting with $s_i(x_i)$ the score provided by the i-th classifier for the instance x represented in the i-th feature space, and with t the decision threshold, it is sufficient to define the decision function as

$$y_{\mathcal{C}}(x) = \begin{cases} +, & \text{if } \sum_{i=1}^{N} s_i \geq t, \\ -, & \text{if } \sum_{i=1}^{N} s_i < t, \end{cases} \quad (4)$$

and to consider V_i and $W_i(x,\mathcal{A}(x))$ as the cost incurred respectively by the classifier for measuring the i-th feature set of the instance x and by the adversary for modifying it.

We now define the classifier strategy against the adversary as the addition of one or more classifiers to the previous ensemble, based on different feature sets. The rationale of this strategy is intuitive, as pointed out in section 2: taking into account different characteristics, or views, of the same instances, it should be easier to discriminate positive from negative ones, and at the same time it should be more difficult for the adversary to evade all of them. We would like to formally analyse this rationale, in light of the framework in [1]. Analogously to [1], we assume that initially the classifier is trained on a given training set and operates on untainted instances, namely $\mathcal{A}(x) = x$. Then the adversary reacts by devising a strategy $\mathcal{A}(x)$, which is likely to decrease the classifier effectiveness. Next, the classifier adds some new classifiers to the previous ensemble. The adversary can in turn react again by devising a new strategy to defeat the new version of the MCS, and so on.

Let us now define the adversary strategy, namely the optimal way in which the adversary should choose the function $\mathcal{A}(x)$ against a given ensemble of N classifiers. To this aim, we assume that the adversary knows the feature set used by each of the individual classifiers, the score $s_i(x), i = 1,\ldots,N$, provided by each individual classifier for any positive instance x, and the threshold t. We also assume that the cost $W_i(x,\mathcal{A}(x))$ for modifying the i-th feature set is equal to the absolute difference between the corresponding scores $s_i(x)$ and $s_i(\mathcal{A}(x))$. This means that the total cost $W(x,\mathcal{A}(x))$ equals the Manhattan distance in the N-dimensional score space between the corresponding score vectors. This is a reasonable assumption without any specific knowledge about the nature of features. It is reasonable to assume that the higher the score reduction the adversary would like to attain, the more the modifications he has to make. Thus for getting a lower score he has to face a higher cost. We point out that a similar assumption about the cost of modifying an instance in a given feature space was made in [5]: in that work, the cost was assumed to be a function of the distance between x and $\mathcal{A}(x)$ in the feature space. The optimal strategy of the adversary against an ensemble of N classifiers, defined in eq. 3 for the general case, can be rewritten as follows:

$$\mathcal{A}(x) = \begin{cases} x' \neq x, & \text{if } \exists x' : y_{\mathcal{C}}(x') = -, \Delta U_A > W(x, x'), \\ & x' = \text{argmax}_{x'' \in \mathcal{X}} \Delta U_A - W(x, x''), \\ x, & \text{otherwise}, \end{cases} \tag{5}$$

where ΔU_A is given by $U_A(-,+) - U_A(+,+)$. Under the above assumptions, the above optimal strategy can be rephrased as finding, for any given instance x which is correctly classified as positive by the classifier, namely $\sum_{i=1}^{N} s_i(x) \geq t$, an instance $\mathcal{A}(x)$ which is misclassified as negative by the classifier, namely $\sum_{i=1}^{N} s_i(\mathcal{A}(x)) < t$, and for which the utility gain ΔU_A exceeds the cost for making the modification, which by the above assumptions is given by:

$$W(x, \mathcal{A}(x)) = \sum_{i=1}^{N} |s_i(x) - s_i(\mathcal{A}(x))|. \tag{6}$$

If no such instance can be found, then x is left unchanged. It is not difficult to see that the minimum cost the adversary has to incur so that the modified instance is misclassified as negative equals the difference between the total score given to x by the classifier and the decision threshold t: $\sum_{i=1}^{N} s_i(x) - t$.

It is now possible to give a formal explanation, according to the above framework, of the reasons why the classifier ensemble approach, and in particular adding new classifiers (using new features) to a given ensemble, can be effective in the considered kind of applications. We consider the simplest case in which the previous N classifiers and the decision threshold s are left unchanged. A consequence of adding M new classifiers ($M \geq 1$) is that the score of any positive pattern could increase. In particular, considering the optimal strategy $\mathcal{A}(x))$ against N classifiers, if $\mathcal{A}(x) \neq x$, as seen above the modified instance $\mathcal{A}(x))$ evades the classifier, namely $s^N(x) = \sum_{i=1}^{N} s_i(\mathcal{A}(x)) < t$. However, this is no more guaranteed if one ore more new classifiers are added. Indeed the new score $s^{N+M}(x)$ will be given by the sum of the previous one and of that of the new classifiers, $s^N(x) + \sum_{i=N+1}^{N+M}(x)$, which could exceed t. In other words, the optimal strategy of the adversary against N classifiers could be less effective, in the sense that it could allow to evade the classifier for a smaller set of instances, if new classifiers are added to the ensemble. This means that the detection capability of the classifier has improved. Moreover, the cost to evade the classifier could increase, just because the score of any positive pattern could increase. For some positive patterns x correctly classified by the new classifier ensemble, such increase of the score from $s^N(x)$ to $s^{N+M}(x)$ could make the difference $s^{N+M}(x) - t$ larger than the utility, $U_A(-,+)$, that the adversary would gain by modifying x so that it is misclassified as negative. This implies that there could be some positive instances that the adversary can afford to modify to evade N classifiers, but not to evade $N + M$ classifiers. This means that the classifier has become harder to evade.

The above analysis gives a first theoretical support to the arguments proposed in favour of the classifier ensemble approach in many works related to security applications, mentioned in section 2. In the next section we will apply the above framework to a case study related to the spam filtering task, using a real spam filter and a real corpus of spam and legitimate e-mails, to quantitatively evaluate the improvement in the effectiveness and in the hardness of evasion attainable using the classifier ensemble approach.

4 A case study in spam filtering

In this section we apply the theoretical framework of section 3.2 to a case study involving a real spam filter, SpamAssassin. This is a well known and widely used spam filter, and is particularly suitable to our aim since it is an open source software. The architecture of SpamAssassin (as well as the one of most commercial spam filters), summarised in section 2, fits the one considered in section 3.2. This allows us to analyse the effect of adding new filtering modules (in other words, new classifiers based on different feature sets) on the detection capability and on the hardness of evasion of this spam filter. For our experiments we used the latest version of SpamAssassin (3.2.4) and the configuration named "bayes+net", which includes all the available filtering modules (in particular, a Naive Bayes text classifier) comprising the optional ones (Razor, Pyzor, etc.). Each module has its own scoring system, and scores can be continuous valued or discrete. SpamAssassin is deployed with a predefined scoring system for each module, and a predefined detection threshold t set to 5. We chose to use these predefined settings, although the scoring system of each rule and the value of t can be modified by users. The only module which includes a trainable classification algorithm is the one based on the Naive Bayes text classifier: we trained this classifier as described below.

Our experiments were carried out on the publicly available TREC 2007 e-mail data set.[3] It is made up of 75,419 real e-mail messages, received by a mail server between April 2007 and July 2007, and contains 25,220 legitimate and 50,199 spam messages. For training the Naive Bayes classifier used by SpamAssassin, we used the first 10,000 messages of the data set in chronological order. This training set was made up of 1,969 legitimate e-mails and 8,031 spam e-mails. The remaining 65,419 e-mails were used as test set.

For the purpose of our experiments, the main problem in using a real spam filter made up by a large number of different modules is that it is very difficult to define for each of them the possible modifications that the adversary (namely, a spammer) can make to evade the filter. Moreover, this also makes difficult to define commensurable modification costs for the adversary, for modules using heterogeneous features.[4] Nevertheless, taking into account that in the theoretical framework of section 3.2 the modification cost for modifying the feature vector representation of instances corresponding to any given module was defined as the absolute distance between the scores provided by that module before and after the modification, we decided to avoid the explicit definition of the possible modifications. We simply assume that the adversary can make any modification which reduces arbitrarily the score provided by each module. So the only constraint on the adversary is that the cost faced for modifying any e-mail has to be lower than the utility gained by getting that e-mail misclassified by the filter, as explained in section 3.2. We point out that this simplifying assumption is totally in favour of the adversary, not of the classifier, since we are not setting any constraint on the actual modifications which can be made on spam e-mails.

In our experiments we considered the utility functions $U_A(y_{\mathcal{C}}, y)$

[3] `http://plg.uwaterloo.ca/~gvcormac/treccorpus07/`

[4] This turned out to be possible in the experiments reported in [1] on the same application, because the experiments were made using a single Naive Bayes text classifier: in that work, the modifications were defined as adding dictionary words or using synonyms.

and $U_C(y_C, y)$ reported below:

$$U_A = \begin{cases} U_A(+,+) = 0 \\ U_A(-,+) = 1,5 \\ U_A(+,-) = 0 \\ U_A(-,-) = 0 \end{cases} \quad U_C = \begin{cases} U_C(+,+) = 1 \\ U_C(-,+) = -10 \\ U_C(+,-) = -1 \\ U_C(-,-) = 1 \end{cases} \tag{7}$$

Note that two different utility functions for the adversary were considered, differing only in the value of $U_A(-,+)$, which is the utility gained by the adversary when he evades the filter. This value affects the capability of adversary of evading the filter, since it equals the maximum cost the it can afford for modifying e-mails (we remind the reader that an e-mail x can be changed to any e-mail x', only if $W(x,x') \leq U_A(-,+) - U_A(+,+)$). Finally, we assumed that the cost V_i faced by the MCS for measuring the feature vector of the i-th classifier is zero (such cost is just a negative constant added to the expected value of the utility function in the framework of section 3.2).

The version of SpamAssassin we used is made up of 619 filtering rules each based on different characteristics (features) of e-mails (the list of the rules can be found at `http://spamassassin.apache.org/tests_3_2_x.html`). Since this number is rather high, we did not add rules one at a time. Instead we subdivided them into n disjoint subsets $S_1, \ldots, S_n$, and added at each step all the rules of a given subset. For the purposes of these experiments we chose $n = 6$. The number of rules for each subset was set to 119 for S_1 and to 100 for all the other subsets. In the real SpamAssassin filter new rules are usually added in response to new spammers' tricks which are identified in real spam e-mails. Accordingly, it would have been reasonable to subdivide the rules taking into account their chronological order. Unfortunately the time in which each rule was introduced is not reported in the SpamAssassin documentation. So we resort to make a random subdivision. To make experiments easily reproducible, we sorted the rules alphabetically according to their names as listed in `http://spamassassin.apache.org/tests_3_2_x.html`. The only exception was the rule related to the Naive Bayes classifier: since it is known that text classifiers are used in spam filters since several years, we included this rule in the first subset, S_1.

To evaluate how the addition of filtering modules affects the detection rate and the hardness of evasion of SpamAssassin using the above framework, the experiments were carried according to the above procedure.

```
1. $R \leftarrow \emptyset, \mathcal{A}^0(x) = x$
2. For $k = 1, \ldots, n$:
  2.1 $R \leftarrow R \bigcup S_k$
  2.2 Evaluate the expected utility of the classifier and of the adversary, when the classifier uses the rules in $R$ and the adversary uses the strategy $\mathcal{A}^{k-1}(x)$ which was optimal for the previous set of rules
  2.3 Compute the optimal adversarial strategy $\mathcal{A}^k(x)$ against rules in $R$
  2.4 Evaluate the expected utility of the classifier and of the adversary, when the classifier uses the rules in $R$ and the adversary uses the corresponding optimal strategy $\mathcal{A}^k(x)$
```

The above experimental set up can be phrased as follows. At each step, we first evaluate the performance of the classifier and the adversary after a new set of rules is added to the classifier, and the adversary uses the strategy which was optimal against the previous set of rules (in the first step, this means that the adversary does not modify his instances). This simulates what happens in real cases, as soon as a spam filer is updated. Then the optimal strategy of the adversary against the new set of rules is computed, and the performances of the classifier and the adversary are evaluated again. This simulates what happens when spammers devise new tricks to evade the last version of a spam filter.

The optimal adversarial strategy $\mathcal{A}^k(x)$ at each step was computed as follows, according to section 3.2 and to the above assumption about how the adversary can modify his instances. Denoting $S_1 \bigcup \ldots \bigcup S_k$ as R, for any positive instance x correctly classified by the filter (namely $y_C(x) = +$, or equivalently $\sum_{i \in R} s_i(x) \geq t$), we compute the set of feasible values s'_i for the scores of rules in R which would correspond to an instance x' classified as negative (namely $\sum_{i \in R} s'_i < t$), such that the corresponding cost $W(x,x') = \sum_{i \in R} |s'_i - s_i(x)|$ is minimum and is lower than the utility gain. If such scores can be found, then we assume that the adversary evades the filters by modifying x, otherwise it is assumed that the adversary can not afford to modify x to evade the filter.

The results are shown in figures 1 and 2, for both utility functions considered for the adversary, in terms of the expected utility of the adversary and of the classifier as a function of the number of rules used in SpamAssassin. The results in the top left graph refer to the case in which the adversary does not modify his instances. As one could expect, the expected utility of the classifier increases as the number of rules increases while the opposite happens for the adversary. In other words, this means that adding new filtering modules based on different features allows to improve the detection capability. The only exception is when passing from 419 to 519 rules. This can be due to the fact that after adding rules to SpamAssassin it would be better to reweight all of them, although this was not made in our experiments for the sake of simplicity. The bottom left graph shows what happens when the adversary uses the optimal strategy against each set of rules. The expected utility of the adversary significantly improves with respect to the previous case. The one of the classifier still increases as the number of rules increases, but obviously attains lower values than in the previous case. However, it is worth noting that the improvement attained by the adversary, reported in the top right graph from top, tends to decrease as the number of rules increases. Similarly, the decrease in classifier's expected utility tends to be higher for lower number of rules. The reason is that the modification cost the adversary has to face to evade the classifier increases as the number of rules increases, until it exceeds the utility gain for some positive instances, making it no more convenient to modify them. This is a clear evidence that adding new filtering modules based on different features allows to improve not only the classifier's discriminant capability, but also its hardness of evasion. Consider finally the bottom right graph, corresponding to the case when the classifier adds new rules, and the adversary uses the strategy which was optimal against the *previous* set of rules. For lower number of rules (up to 319), the expected utility of the adversary is between the ones of the first two graphs: this is reasonable, because it is now trying to evade only *some* of the rules used by the classifier. However, for higher number of features its expected utility is even worse than the one it attained *without* trying to evade any rule. The expected utility of the classifier is instead very close to the one it attained when the adversary did not try to evade any rule. This means that the addition of new rules allowed to compensate the tricks introduced by the adversary to evade the previous rules. In other words, most spam e-mails which evaded the previous version of the filter were detected by the new rules.

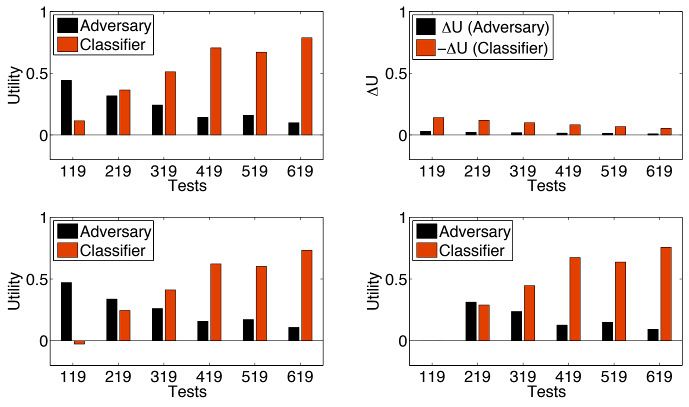

Figure 1. Expected utility for the adversary and the classifier, as a function of the number of rules used by the classifier, when $U_A(-, +) = 1$. Top left: the adversary does not modify his instances. Bottom left: the adversary uses the optimal strategy against the classifier. Top right: gain and loss in expected utility attained respectively by the adversary and the classifier, when passing from the situation in the top left graph to that in the bottom left one, for each number of rules used by the classifier. Bottom right: for each set of rules, the adversary uses the optimal strategy against the *previous* set of rules.

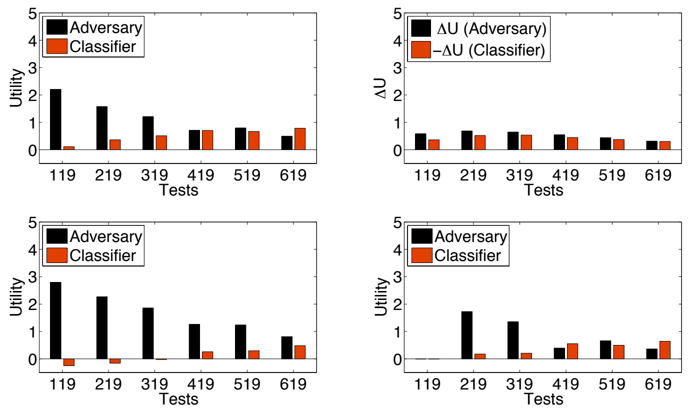

Figure 2. Expected utility for the adversary and the classifier, as a function of the number of rules used by the classifier, when $U_A(-, +) = 5$. See caption of figure 1 for the other details.

The behaviour of the expected utility in figures 1 and 2 is similar, with the obvious difference that the expected utility of the adversary is higher in the graphs of figure 2 than in figure 1, since it can afford a higher cost to modify instances (the opposite happens for the classifier). These experimental results on a real case study give thus a quantitative confirmation to the theoretical explanation given in section 3.2 on the effectiveness of the classifier ensemble approach in improving both the detection capability and the hardness of evasion of a security system like a spam filter.

ACKNOWLEDGEMENTS

We would like to thank Nilesh Dalvi and Mausam for providing us the code used in [1].

REFERENCES

[1] Nilesh Dalvi, Pedro Domingos, Mausam, Sumit Sanghai, and Deepak Verma, 'Adversarial classification', in *Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 99–108, Seattle, (2004).

[2] Giorgio Giacinto, Fabio Roli, and Luca Didaci, 'Fusion of multiple classifiers for intrusion detection in computer networks', *Pattern Recognition Letters*, **24**, 1795–1803, (2003).

[3] Michal Haindl, Josef Kittler, and Fabio Roli, eds. *Multiple Classifier Systems, 7th International Workshop, MCS 2007, Prague, Czech Republic, May 23-25, 2007, Proceedings*, volume 4472 of *Lecture Notes in Computer Science*. Springer, 2007.

[4] Josef Kittler, Mohamad Hatef, Robert P.W. Duin, and Jiri Matas, 'On combining classifiers', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **20**(3), 226–239, (1998).

[5] Daniel Lowd and Christopher Meek, 'Adversarial learning', in *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, ed., ACM Press, Chicago, IL., (2005).

[6] Roberto Perdisci, Guofei Gu, and Wenke Lee, 'Using an ensemble of one-class svm classifiers to harden payload-based anomaly detection systems', in *International Conference on Data Mining (ICDM)*, pp. 488–498. IEEE Computer Society, (2006).

[7] Arun A. Ross, Karthik Nandakumar, and Anil K. Jain, *Handbook of Multibiometrics*, Springer Publishers, 2006.