

# Multiple classifier systems under attack

Battista Biggio, Giorgio Fumera, and Fabio Roli

Dept. of Electrical and Electronic Eng., Univ. of Cagliari  
Piazza d'Armi, 09123 Cagliari, Italy  
[{battista.biggio,fumera,roli}@diee.unica.it](mailto:{battista.biggio,fumera,roli}@diee.unica.it)  
WWW home page: <http://prag.diee.unica.it>

**Abstract.** In adversarial classification tasks like spam filtering, intrusion detection in computer networks and biometric authentication, a pattern recognition system must not only be accurate, but also *robust* to manipulations of input samples made by an adversary to mislead the system itself. It has been recently argued that the robustness of a classifier could be improved by avoiding to overemphasize or underemphasize input features on the basis of training data, since at operation phase the feature importance may change due to modifications introduced by the adversary. In this paper we empirically investigate whether the well known bagging and random subspace methods allow to improve the robustness of linear base classifiers by producing more uniform weight values. To this aim we use a method for performance evaluation of a classifier under attack that we are currently developing, and carry out experiments on a spam filtering task with several linear base classifiers.

## 1 Introduction

In *adversarial classification* tasks like spam filtering, intrusion detection in computer networks and biometrics [1–4], the goal of a pattern recognition system is to discriminate between two classes, which can be named “legitimate” and “malicious”, while an intelligent adversary manipulates samples to mislead the system itself. Adversarial classification problems are therefore non-stationary, which implies that a pattern recognition system should be designed by taking into account not only its accuracy (usually evaluated from a set of training samples) but also its robustness, namely the capability of undergoing an accuracy degradation as low as possible when it is under attack. Very few works addressed so far the problem of devising practical methods to improve robustness. Recently, in [7] it was suggested that a more robust classifier could be obtained by avoiding to give features too much or too little emphasis during classifier training, and a similar approach was suggested in [6]. This allows to design robust classifiers against attacks in which the adversary exploits some knowledge on the classification function (e.g., the most discriminant features), as we discuss in the next section.

It is well known that one of the main motivations for the use of multiple classifier systems (MCSs) is the improvement of classification accuracy with respect

to a single classifier. Recently, MCSs have also been applied to adversarial classification tasks based only on intuitive motivations, although there is no clear evidence so far that they can be also useful to improve robustness. Our aim is to investigate whether and under which conditions MCSs allow to improve the robustness of a pattern recognition system in adversarial classification tasks, with respect to a single classifier architecture. In this work we will focus on two of the most known methods for constructing MCSs, namely bagging [8] and the random subspace method (RSM) [9]. The reason is that we argue that these methods could result in avoiding to give features too much or too little emphasis with respect to a single classifier during classifier training, which is the strategy suggested in [7] to improve robustness. Accordingly, we first experimentally investigate whether this assumption holds in practice, and then, whether this allows bagging and RSM to produce more robust classifiers. Our experimental analysis was carried out on a spam filtering task, using the well known and widespread open source anti-spam filter *SpamAssassin* [10].

In Sect. 2 we survey related works. The method to assess the robustness is explained in Sect. 4.1. The experiments are reported in Sects. 4 and 5.

## 2 Related works

MCSs are currently being used in several adversarial classification tasks, like multimodal biometric systems [1, 4], intrusion detection in computer systems [2], and spam filtering [11–13]. Two practical reasons are that in many of such tasks several heterogeneous feature subsets are available, and they can be easily exploited in a MCS architecture where each individual classifier is trained on a different feature subset [1]; moreover, in tasks like intrusion detection it is often necessary to face never-seen-before attacks, which can be easily done with a MCS architecture by adding new classifiers. For instance, MCSs are used in commercial spam filters which use heterogeneous information sources to label incoming e-mails (like the e-mail’s text, its header, the attached images if any, etc.). Another motivation was proposed by several authors: using many classifiers would improve robustness because it would require the adversary to evade all the individual classifiers, or at least more than one of them, to evade the whole system [4, 2, 11]. However, we point out that this motivation is only based on intuition, and its validity has never been evaluated. Accordingly, understanding whether and how MCSs actually allow to improve robustness is still an open question.

In our past works we addressed this problem under several viewpoints. Since adding new detection rules to a system in response to new attacks is a common practice in spam filtering and in intrusion detection in computer systems, in [14] we investigated whether adding classifiers to a given ensemble can improve its robustness. In [15] we analysed a randomization strategy based on MCSs to improve robustness by preventing an adversary from gaining too much knowledge on a classifier. However in these works we used an analytical model for adversarial classification problems proposed in [5], which is based on unrealis-

tic assumptions, and thus our results could not be exploited to devise practical methods to design robust classifiers. In [16] we provided an empirical evidence<sup>1</sup> that a MCS architecture can be more robust than a single classifier architecture. This work was however limited to a non-standard MCS architecture (a logic OR of Boolean outputs of individual classifiers).

Among the few practical methods proposed so far in the literature for improving classifier robustness, the ones in [6, 7] are particularly interesting since they are not limited to a specific task. In [7] it was pointed out that assessing the performance of a classifier during its design phase is likely to provide an *optimistic* estimate of the performance at operation phase, since training samples cannot contain any attack targeted to the classifier under design. In particular, if some features have a high discriminant capability on training samples and the adversary is aware of that, he could manipulate his samples at operation phase to modify the values of such features. The consequence is that their discriminant capability becomes lower at operation phase. For instance, text classifiers based on the bag of words feature model (in which terms occurrence or frequency are used as features) are widely used in spam filters. However spammers can guess the most discriminant terms that characterize spam and legitimate emails, and indeed two real and widely used attack strategies consist in camouflaging “spammy” terms (e.g., by misspelling them) to avoid their detection, and in adding “legitimate” terms not related to the spam message. The strategy proposed in [7] to improve robustness is hence to avoid to give features too much or too little emphasis during classifier training, to protect the classifier against such kinds of attacks. Several versions of this strategy were devised for linear classifiers, resulting in learning algorithms whose goal is a trade-off between attaining a high training accuracy and forcing the feature weights to be as much uniform as possible. A seemingly different strategy was proposed in [6], where a SVM-like learning algorithm was developed to make a linear classifier based on Boolean features robust against the modification of the most important features. Interestingly, it turns out that the effect of the proposed algorithm is similar to the strategy of [7], namely it results in producing more uniform weight values.

### 3 Motivations of this work

The strategies in [6, 7] are based on the intuition that, to improve robustness against attacks based on some knowledge on the relevance of each feature in the classifier’s decision function, it can be useful to prevent the learning algorithm to overemphasize or underemphasize features. In the case of linear classifiers this strategy can be implemented by forcing the feature weights to be as uniform as possible. In this paper we investigate whether this effect can be obtained by some known MCS construction methods. We focus in particular on the well known bagging and RSM. In RSM, base classifiers are trained on randomly chosen feature subsets. Thus each feature may not be used by some base classifiers. In the particular case of linear classifiers, we can say that the most discriminant

---

<sup>1</sup> Reported analytical results turned out to be wrong (see the Erratum below).

features (on the training set) have a zero weight when they are not used, so their average weight across base classifiers could be lower than in a classifier trained on the whole feature set. Analogously, the average weight of less discriminant features could be higher in average, since their importance can be higher if they are used in base classifiers where the most discriminant features do not appear. When bagging is used, the training set of each base classifier is a bootstrap replicate of the original training set. Therefore, each training sample could not appear in some bootstrap replicates. One of the possible effects could be a reduction of the average weight of most discriminant features and an increase of the average weights of less discriminant ones. As mentioned above, our goal is to experimentally investigate whether and how this happens, and whether this results in a higher robustness with respect to an individual classifier, against attacks based on some knowledge on the feature discriminant capability. We focus in particular on a linear combination of linear classifiers, which can be easily analysed. Indeed the discriminant function of such a linearly combined set of classifiers can be written as:

$$g(\mathbf{x}) = \frac{1}{K} \sum_{k=1}^K \left( \sum_{i=1}^n w_i^k x_i + w_0^k \right) = \sum_{i=1}^n w_i^{\text{avg}} x_i + w_0^{\text{avg}} \quad (1)$$

which is still a linear discriminant function in feature space, where  $K$  is the number of base classifiers,  $n$  the number of features of each base classifier and  $w_0^{\text{avg}} \dots w_n^{\text{avg}}$  are the weights assigned by the MCS to the input features, each one computed by averaging the correspondent  $K$  weights of the base classifiers. Therefore, our goal is to understand whether the weights  $w_0^{\text{avg}} \dots w_n^{\text{avg}}$  obtained by bagging and by RSM are more uniform than the weights of a single linear classifier trained on the whole feature set and on all the available training samples, and whether this results in a higher robustness under attack with respect to an individual classifier.

## 4 Experimental setup

We first describe the method used in this paper to evaluate the robustness of a classifier, and then the data set and the classifiers used in the experiments.

### 4.1 A method to assess the robustness of classifiers under attack

Ideally, the robustness of a classifier against a given attack should be evaluated on samples corresponding to such attack. However training samples cannot contain attacks devised against the classifier which is being designed, as pointed out in [7], and it could be very difficult to construct real attacks. Accordingly, we are currently developing a methodology to evaluate classifier's robustness, whose basic idea is to evaluate the accuracy of a classifier on *artificial* samples obtained by modifying the feature vectors of the available malicious testing samples with the aim of *simulating* the effect of a given attack of interest. Our methodology

is an extension of the method used in [7] to evaluate the proposed strategies to improve robustness (see Sect. 2).

The modifications to feature vectors of testing samples can be made with different criteria, depending on the specific application, the classifier and the attack to simulate. Nevertheless, in general it is useful to evaluate robustness under attacks of different strength [17, 6, 7]. The attack strength on a given sample can be measured as the distance in the feature space between the original sample and the one camouflaged by the attacker. We point out that the distance in feature space was used in [5, 18] to measure the adversary’s effort in modifying a given sample. The underlying rationale is that the more modifications an adversary makes to a sample, the more is the required effort. In particular, in the case of binary features a straightforward distance measure is the Hamming distance, which amounts to the number of features modified to “camouflage” a malicious sample.

One of the conditions under which it can be interesting to evaluate robustness is a “worst case” attack, in which the adversary is assumed to exactly know the classifier’s decision function. Let us denote with  $A(\mathbf{x})$  the modification of a feature vector  $\mathbf{x}$ , with  $m$  the maximum distance in the feature space between the original and modified feature vector  $\mathbf{x}'$  according to a given metric  $d(\cdot, \cdot)$  (namely, the maximum attack strength), and with  $g(\mathbf{x})$  the discriminant function of the considered classifier, with the convention that the decision function is  $f(\mathbf{x}) = \text{sign } g(\mathbf{x}) \in \{-1, +1\}$ , and that  $+1$  and  $-1$  are the labels respectively of the malicious and legitimate class. For any given malicious pattern  $\mathbf{x}$ , the worst case attack is the one which modifies  $\mathbf{x}$  to the pattern  $\mathbf{x}'$  which decreases most the value of the discriminant function  $g(\mathbf{x}')$ , under the constraint  $d(\mathbf{x}, \mathbf{x}') \leq m$ . This can be written as the solution of a constrained optimization problem:

$$A(\mathbf{x}) = \operatorname{argmin}_{\mathbf{x}'} g(\mathbf{x}'), \text{ s.t. } d(\mathbf{x}, \mathbf{x}') \leq m. \quad (2)$$

In the case when the features are Boolean (and thus  $m$  corresponds to the maximum number of features which can be modified),  $d(\cdot, \cdot)$  is the Hamming distance and the discriminant function is linear ( $g(\mathbf{x}) = \sum_{i=1}^n w_i x_i + w_0$ ), it is easy to see that the solution can be found as follows. First the absolute values of the weights  $|w_1|, \dots, |w_n|$  must be sorted in decreasing order. Then the features must be considered in that order, and the values of up to  $m$  of them must be switched either from 1 to 0, if the corresponding weight is positive, or from 0 to 1, if the weight is negative.

## 4.2 Data set and base classifiers

Our experiments were carried out on a spam filtering task. We used the TREC 2007 e-mail corpus, publicly available at <http://plg.uwaterloo.ca/~gvcormac/treccorpus07> and made up of 75,419 real e-mails (25,220 legitimate and 50,199 spam messages). We considered the first, second and third sequence of 10,000 e-mails (in chronological order) denoted in the following as  $D_1$ ,  $D_2$  and  $D_3$ , to build the training and testing sets as described below. The performance measure

adopted to evaluate accuracy and robustness is the portion of the area under the ROC curve corresponding to false positive (FP) error rates between 0 and 10%, denoted as  $AUC_{10\%}$ . Since the area under the whole ROC curve takes on values in  $[0, 1]$ ,  $AUC_{10\%}$  takes on values in  $[0, 0.1]$ . This measure was suggested in [7], to take into account the fact that in adversarial classification tasks (and especially in security applications) FP errors are typically much more harmful than false negative ones, and thus the classifier's operating point is required to have a low FP rate. We carried out two sets of experiments, with two different linear classifiers used in spam filtering tasks.

**Experiment 1.** In the first set of experiments we used two text classification algorithms proposed in the spam filtering literature, namely, support vector machine (SVM) [19] and logistic regression (LR) [7]. The e-mails in  $D_1$  and  $D_2$  were used to build respectively the training and testing set. We used the *bag of words* feature model, which is a common choice in text classification and spam filtering tasks. We first constructed a *vocabulary*, namely the set of distinct terms appearing in training e-mails (to this aim we used the anti-spam filter *SpamAssassin* [10], see below), which turned out to be 366,709. Each training and testing e-mail was then represented as a feature vector of the same size, in which each component was associated to a vocabulary term, and was set to 1 if that term occurred in the e-mail, to zero otherwise. The LR classifier was trained by maximising the classification accuracy through a gradient descent algorithm. Since the original feature set was too large for SVMs, we selected 20,000 features from training e-mails using the information gain criteria. The  $C$  parameter of the SVM learning algorithm was chosen among the values  $\{0.001, 0.01, 0.1, 1, 10, 100\}$  by maximising the  $AUC_{10\%}$  through a 5-fold cross validation on the training set. In the experiments we compared a single classifier built using LR and SVM, and MCSs built using bagging and RSM, for different ensemble sizes (3, 5 and 10), different fractions of randomly selected features from the original feature set for RSM (20%, 50%, 80%), and different training set sizes for bagging (20% and 100% of the original one). The robustness was then evaluated using the method described in Sect. 4.1 on the testing set.

**Experiment 2.** The second set of experiments was carried out using the popular and widespread open source anti-spam filter *SpamAssassin* [10] (version 3.2.5). It is made up of some hundred Boolean “tests” on the e-mail content, each one aimed at detecting a particular characteristic of spam or legitimate e-mails (for instance the presence of a typical spam word, or a known e-mail's header malformation indicating that it has been forged by an automatic software, as typically done by spammers [20]). Denoting with  $\mathbf{x} = (x_1, \dots, x_n) \in \{0, 1\}^n$  the tests' outputs, the decision function of the whole filter can be written as  $f(\mathbf{x}) = \text{sign}(\sum_{i=1}^n w_i x_i + w_0)$  where  $\{w_0, \dots, w_n\} \in \mathbb{R}^{n+1}$  is a set of weight values, and  $\text{sign}(t) = +1$  if  $t \geq 0$ ,  $-1$  elsewhere. An e-mail is classified as spam (legitimate) if  $f(\mathbf{x}) = +1$  ( $f(\mathbf{x}) = -1$ ). In our experiments, we used only the tests whose value was not zero for at least one e-mail of the data set, which turned out to be 549. Default weight values  $w_0, \dots, w_n$  are provided by

SpamAssassin developers<sup>2</sup>. These values were derived by manually adjusting the values obtained by a perceptron trained over a huge amount of data, with the aim of increasing the robustness of the spam filter. Note that default weights of tests associated to characteristics of spam (legitimate) e-mails are positive (negative). We used SpamAssassin as a linear classifier, and carried out the same experiments described above for text classifiers. Namely, we computed the weight values using an individual LR and SVM classifier, and a MCS obtained using bagging and RSM, on the same base classifiers. These classifiers were also compared to the individual classifier with the default weight values, namely the standard SpamAssassin. We point out that some SpamAssassin tests (features) are based on the outputs of a text classifier, which must be previously trained. We trained it using the  $D_1$  e-mail subset. To compute the weights with the LR and SVM classifiers we used the e-mails of the  $D_2$  subset as training set. The LR and SVM classifiers were trained as in the previous experiments. Robustness was evaluated on the e-mails in  $D_3$ . Results are reported in Sect. 5.

## 5 Experimental results

The results of our experiments are reported in terms of  $AUC_{10\%}$  as a function of the attack strength  $m$ . Since the features are Boolean, we measured the attack strength  $m$  as the maximum number of features which can be modified. According to this measure, a classifier is more robust than another if it exhibits a higher  $AUC_{10\%}$  value for the same value of  $m$ . Note that the true positive (TP) classification rate of a classifier decreases as the attack strength  $m$  increases, and so does the  $AUC_{10\%}$  value. As a limit case, beyond some  $m$  value all the modified malicious testing samples are labelled as legitimate. Consequently, the TP rate equals zero for any FP value and the corresponding  $AUC_{10\%}$  equals zero as well. We also report a measure of the evenness of the feature weights assigned by the classifiers, which was proposed in [7], defined as the ratio of the sum of the top  $K$  absolute weight values to the sum of all the  $n$  absolute weight values, for  $K = 1, \dots, n$ :

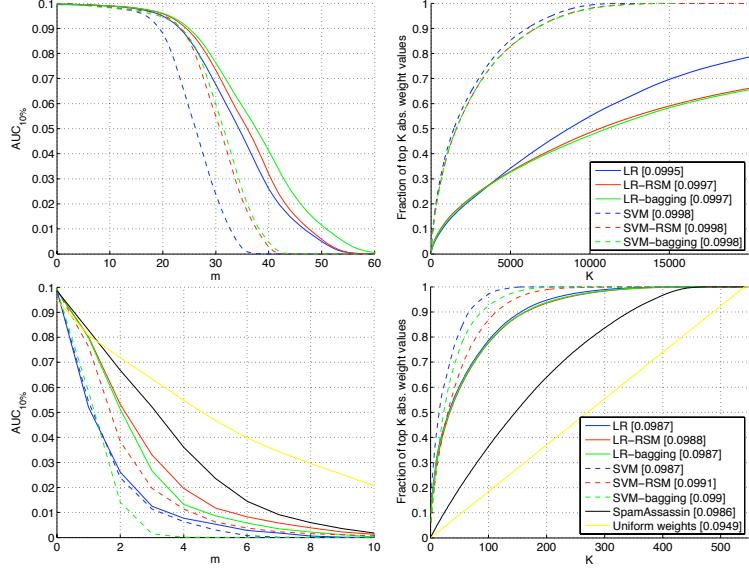
$$F(K) = \left( \sum_{i=1}^K |w_{(i)}| \right) / \left( \sum_{i=1}^n |w_{(i)}| \right) \quad (3)$$

where  $|w_{(1)}|, \dots, |w_{(n)}|$  are the absolute weight values sorted in decreasing order. The weight  $w_0$  is disregarded since it does not affect the  $AUC_{10\%}$  value. When all weights are equal,  $F(K) = K$ , while as the weight distribution become uneven,  $F(K)$  approaches 1 for any  $K$  value.

**Experiment 1.** The results obtained with text classifiers are shown in Fig. 1. We found that the robustness of MCSs significantly increased for increasing ensemble size with RSM (especially for a low feature subset size), while no significant changes were observed with bagging. This result provides some support the one reported by the authors in [14]. Due to lack of space, only the results for the

---

<sup>2</sup> [http://spamassassin.apache.org/tests\\_3\\_2\\_x.html](http://spamassassin.apache.org/tests_3_2_x.html)



**Fig. 1.**  $AUC_{10\%}(m)$  (left) and  $F(K)$  (right) for single LR and SVM classifiers, and for bagging and RSM with LR and SVM as base classifiers. Top: text classifiers (exp. 1); bottom: SpamAssassin (exp. 2). For the latter, results obtained with default and uniform weights are also reported. The  $AUC_{10\%}$  value at  $m = 0$  is reported also in the legend. Results for the MCSs are averaged over 5 runs (standard deviation is not reported since it is negligible). Note that the  $AUC_{10\%}$  drops to zero when the attack strength becomes so high that all the modified malicious testing samples are labelled as legitimate, and thus the TP rate equals zero for any FP value.

largest considered ensemble size (10) are reported. Similarly, the robustness of the RSM method significantly increased as the size of feature subsets increased, while no significant difference was observed with bagging for different training set sizes. Due to lack of space, only results corresponding to the highest considered feature subset size in RSM (80%) and training set size in bagging (100%) are reported. Let us compare bagging and RSM with a single classifier. First, Fig. 1 shows for  $m = 0$  (namely, on the original testing set) the  $AUC_{10\%}$  value of bagging and RSM was nearly the same as the one of the corresponding individual classifiers trained on the original training set. This means that bagging and RSM did not improve the performance of the individual classifiers. However, for  $m > 0$  (namely, when the classifiers are under attack) both bagging and RSM allowed to obtain more even weight distributions with respect to the individual classifiers, and also exhibited a higher robustness. This result supports our main hypothesis, namely that bagging and RSM result in more uniform weight values than a single classifier (Sect. 3). This result is interesting also because it shows that although MCSs did not improve classification accuracy with respect to individual classifiers, they nevertheless allowed to improve robustness. It can also

be noted that in this case using a MCS did not increase the computational cost at classification phase, since the discriminant function of the MCS was linear as the one of the single classifier.

**Experiment 2.** The results obtained with the SpamAssassin filter as a linear classifier are reported in Fig. 1. On the basis of the results of the previous experiments, we considered only ensembles of 10 base classifiers, and feature subsets of 80% of the whole feature set for RSM. We also report the results obtained by SpamAssassin as a single classifier, both using its default weight values and using uniform weight values, equal to either +1 (for features associated to characteristics of spam e-mails, see above) or to -1. As expected, SpamAssassin with uniform weights attains the highest robustness (see Fig. 1, bottom). However, it does not exploit any information about the discriminant capability of the features, and therefore it exhibits the worst performance on the original testing set without attacks ( $AUC_{10\%}$  for  $m = 0$ ). An interesting result is that weights obtained by the LR and SVM classifiers slightly outperformed the default SpamAssassin weights for  $m = 0$ , while the default values exhibited a higher robustness for  $m > 0$ , namely when the classifiers were under attack. This provides evidence that the default weights, which were manually set by SpamAssassin developers, are actually capable to improve its robustness under attack. In particular, we observed that the LR and SVM learning algorithms assigned higher weights to the most discriminant features (on training samples), which turned out to be the ones related to the text classifier included in SpamAssassin. This guaranteed to achieve higher performances when the filter was not under attack, but also undermined its robustness under attack. The corresponding default weight values were indeed lower than the ones assigned by the LR and SVM classifiers. As in the previous experiments, Fig. 1 also shows that the use of MCSs did not improve the classification accuracy when the classifiers were not under attack, but it allowed to improve the robustness when the classifiers were under attack, with the only exception of bagging with the SVM base classifier. It is interesting to note that the default SpamAssassin weights exhibited a higher robustness than bagging and RSM, besides than single classifiers.

## 6 Conclusions

While MCSs are mainly used to improve classification accuracy, inspired by [6, 7] we argued that methods like bagging and RSM could also result in improving robustness in adversarial classification tasks against attacks based on some knowledge of the classifier's discriminant function, due to a potential side effect consisting in giving more uniform weights to the features with respect to a single classifier. Reported experiments performed on a real spam filtering task, using text classifiers and a real spam filter, provided evidence that this intuition is correct. In particular, our experiments showed that, even in cases when bagging and RSM did not improve the performance of a single classifier when they are not under attack, they turned out to be significantly more robust under attack. These results provide a first sound motivation to the application of MCSs in

adversarial classification tasks, other than the intuitive and qualitative considerations that have motivated their use so far, and thus open a new and relevant area of research for MCSs.

## References

1. Kittler, J., Hatef, M., Duin, R.P., Matas, J.: On combining classifiers. *IEEE Trans. Pattern Analysis and Machine Intelligence* **20**(3) (1998) 226–239
2. Perdisci, R., Gu, G., Lee, W.: Using an ensemble of one-class svm classifiers to harden payload-based anomaly detection systems. In: Int. Conf. Data Mining (ICDM), IEEE Computer Society (2006) 488–498
3. Giacinto, G., Roli, F., Didaci, L.: Fusion of multiple classifiers for intrusion detection in computer networks. *Patt. Rec. Lett.* **24** (2003) 1795–1803
4. Ross, A.A., Nandakumar, K., Jain, A.K.: *Handbook of Multibiometrics*. Springer (2006)
5. Dalvi, N., Domingos, P., Mausam, Sanghai, S., Verma, D.: Adversarial classification. In: ACM Int. Conf. Knowledge Discovery and Data Mining (2004) 99–108
6. Globerson, A., Roweis, S.T.: Nightmare at test time: robust learning by feature deletion. In: Int. Conf. Machine Learning (2006) 353–360
7. Kolcz, A., Teo, C.H.: Feature weighting for improved classifier robustness. In: Sixth Conf. Email and Anti-Spam (CEAS), Mountain View, CA, USA (2009)
8. Breiman, L.: Bagging predictors. *Machine Learning* **24**(2) (1996) 123–140
9. Ho, T.K.: The random subspace method for constructing decision forests. *IEEE Trans. Pattern Analysis and Machine Intelligence* **20**(8) (1998) 832–844
10. The Apache SpamAssassin Project, <http://spamassassin.apache.org/>
11. Hershkop, S., Stolfo, S.J.: Combining email models for false positive reduction. In: Proc. ACM Int. Conf. Knowledge Discovery in Data Mining, ACM (2005) 98–107
12. Lynam, T.R., Cormack, G.V., Cheriton, D.R.: On-line spam filter fusion. In: Proc. Int. ACM SIGIR Conf. on Res. and Dev. Inf. Retr., ACM (2006) 123–130
13. Tran, T., Tsai, P., Jan, T.: An adjustable combination of linear regression and modified probabilistic neural network for anti-spam filtering. In: Int. Conf. Patt. Rec. (2008) 1–4
14. Biggio, B., Fumera, G., Roli, F.: Evade hard multiple classifier systems. In: Okun, O., Valentini, G., eds.: *Supervised and Unsupervised Ensemble Methods and their Applications*. Vol. 245 *Studies in Comp. Int.* Springer-Verlag, (2009) 15–38
15. Biggio, B., Fumera, G., Roli, F.: Adversarial pattern classification using multiple classifiers and randomisation. In: Joint IAPR Int. Workshop on Structural and Syntactic Pattern Recognition, Vol. 5342 LNCS, Springer-Verlag (2008) 500–509
16. Biggio, B., Fumera, G., Roli, F.: Multiple classifier systems for adversarial classification tasks. In: *Multiple Classifier Systems*. Vol. 5519 LNCS, Springer (2009) 132–141
17. Barreno, M., Nelson, B., Sears, R., Joseph, A.D., Tygar, J.D.: Can machine learning be secure? In: Proc. 2006 ACM Symp. Information, computer and communications security (ASIACCS 06), New York, NY, USA, ACM (2006) 16–25
18. Lowd, D., Meek, C.: Adversarial learning. In: Proc. 11th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (2005) 641–647
19. Drucker, H., Wu, D., Vapnik, V.N.: Support vector machines for spam categorization. *IEEE Trans. Neural Networks* **10**(5) (1999) 1048–1054
20. Stern, H.: A survey of modern spam tools. In: 5th Conf. Email and Anti-Spam (CEAS), Mountain View, CA, USA (2008)