Noname manuscript No.
(will be inserted by the editor)

Multiple Classifier Systems for Robust Classifier Design in Adversarial Environments

Battista Biggio · Giorgio Fumera · Fabio Roli

Received: date / Accepted: date

Abstract Pattern recognition systems are increasingly being used in *adversarial environments* like network intrusion detection, spam filtering and biometric authentication and verification systems, in which an adversary may adaptively manipulate data to make a classifier ineffective. Current theory and design methods of pattern recognition systems do not take into account the adversarial nature of such kind of applications. Their extension to adversarial settings is thus mandatory, to safeguard the security and reliability of pattern recognition systems in adversarial environments. In this paper we focus on a strategy recently proposed in the literature to improve the robustness of linear classifiers to adversarial data manipulation, and experimentally investigate whether it can be implemented using two well known techniques for the construction of multiple classifier systems, namely, bagging and the random subspace method. Our results provide some hints on the potential usefulness of classifier ensembles in adversarial classification tasks, which is different from the motivations suggested so far in the literature.

Keywords Adversarial classification · Multiple classifier systems · Robust classifiers · Linear classifiers

1 Introduction

Pattern recognition systems are increasingly being used in applications like biometric authentication and verification, intrusion detection in computer networks, spam filtering, Web page ranking and network protocol verification [33,46,32, 25,42,17,23], usually to discriminate between two pattern classes corresponding to a legitimate and a malicious behaviour. These applications are different from the ones considered in the standard pattern recognition theory, since they are characterised by the presence of a human *adversary* who generates malicious samples, and can adaptively manipulate data to avoid their detection. For example, the goal of biometric verification systems is to discriminate between genuine and impostor users, to allow or deny access to some protected resource. An impostor may try to be recognised as a genuine user by spoofing his fingerprints. Analogously, intrusion detection systems (IDSs) aim at discriminating between legitimate and intrusive network traffic, and hackers may camouflage their network packets so that they are mislabelled as legitimate. Likewise, spammers adopt several tricks to obfuscate their emails and get them past spam filters. In automatic Web page ranking pattern recognition systems can be used to automatically label or score Web pages according to predefined topics, for automatic ranking purpose. A malicious Webmaster may inflate the ranking of his Web site, for example, by manipulating the metadata of Web pages. In network protocol verification automatically recognising the protocol of network packets can be useful to improve the quality of service over a network. However, a malicious user may attempt to get a higher bandwidth by disguising the protocol in use.

The performance of intelligent data analysis systems [47], and in particular pattern recognition systems, may thus be undermined when they operate in adversarial environments. However, current theory and design methods of pattern recognition systems do not take into account the adversarial nature of such applications. Current methods and algorithms may thus exhibit vulnerabilities which can be exploited by an adversary to mislead them. This is a relevant issue that needs to be addressed, to allow the deployment of reliable

B. Biggio, G. Fumera, F. Roli
Dept. of Electrical and Electronic Eng., University of Cagliari
Piazza d'Armi, 09123 Cagliari, Italy
Tel.: +39-070-67557-76 (B. Biggio), -54 (G. Fumera), -79 (F. Roli)
Fax: +39-070-6755782
E-mail: {battista.biggio, fumera, roli}@diee.unica.it

and robust pattern recognition systems in many crucial applications.

The issue of *adversarial classification* has been raised only recently in the literature. Some theoretical issues have been addressed in the machine learning field, while researchers from various application-centred fields (mainly intrusion detection and spam filtering), focused on specific vulnerabilities of pattern recognition and machine learning systems in particular applications. However, the current research is still limited and fragmented, and a first attempt to unify the efforts in this field was made only in a recent NIPS workshop [38]. In brief, the main open issues are: identifying vulnerabilities of pattern recognition methods and algorithms to adversarial data manipulation, and developing methods for evaluating and improving their robustness in adversarial environments.

In this paper we focus on the use of multiple classifier systems (MCSs) to improve the robustness of pattern classifiers. During the past fifteen years MCSs became a state-of-the-art tool for the design of pattern classifiers, mainly because of their capability to improve accuracy with respect to an individual classifier. Some authors have recently argued that MCSs can also improve robustness in adversarial settings, since in principle more than one classifier has to be evaded to make the whole ensemble ineffective [46,29, 44,52]. However, this claim has never been investigated in depth, and remains questionable. Some opposite evidence was indeed provided in [45] for multi-modal biometric systems. Inspired by a strategy to improve the robustness of linear classifiers proposed in [35], in this paper we investigate the capability of a particular kind of MCS construction techniques to improve the robustness of linear base classifiers, based on a different rationale than the one argued so far. The strategy in [35] is based on keeping the feature weights as evenly distributed as possible, which forces the adversary to modify a larger number of feature values to evade the classifier. We argue that randomisation-based MCS construction techniques like bagging [9] and the random subspace method (RSM) [30] may produce such effect on the feature weights of linear classifiers, and empirically evaluate this behaviour on a spam filtering case study, extending preliminary results presented in [6,7].

The paper is structured as follows. An overview of the literature on adversarial classification is given in Sect. 2. In Sect. 3 we describe the robustness improvement strategy of [35], and discuss how MCSs can be exploited to implement it. In Sect. 4 we describe the method we used to evaluate robustness. Our experimental analysis is reported in Sect. 5.

2 Background

In this section we summarise the literature on adversarial classification, and in particular works which proposed the use of MCSs to improve robustness.

Theoretical works. Theoretical issues of adversarial classification have been addressed by few works in the machine learning field [17,2,40,13,37]. In [17] an analytical framework based on minimum risk theory was proposed, in which adversarial classification tasks were modelled as two-player games. Using this framework, the authors experimentally showed on a spam filtering task that an *adversary aware* classifier, designed by *anticipating* potential adversarial actions, can significantly outperform standard adversary unaware classifiers. In [2] some general issues about the security of machine learning systems in adversarial environments were discussed, and a taxonomy of attacks against them was developed. Some possible defence strategies were also sketched. A more specific issue was addressed in [40], namely, the computational complexity of reverse-engineering the classifier's decision function, for an adversary who can probe the classifier with "query" samples and get feedback on the assigned label. General frameworks for the evaluation of classifier performance in adversarial environments were proposed in [13,37].

While the above theoretical works pointed out the main issues of adversarial classification tasks, they did not provide practical methods to design robust classifiers in real applications. On the other hand, the majority of works published so far focused on very specific issues of individual applications, and their solutions can not be generalised to different contexts. Moreover, these works are mostly unrelated to theoretical ones, so that there is still a huge gap between theory and applications.

Application-specific works. Some works analysed the vulnerability of different classifiers used in IDSs [20,34] and in biometric verification systems [53,22]. Specific countermeasures were proposed for IDSs in [43,16], as well as methods to improve the robustness of specific classification algorithms. For instance, in [45] a modified version of the well known likelihood ratio rule was proposed, for multi-modal biometric verification tasks. In the spam filtering task, several works analysed the vulnerability of text classifiers against well known spammers' tricks aimed at getting spam emails misclassified as legitimate [26,41,31,24,35].

To our knowledge, only in [24,35] practical methods to improve classifier robustness were proposed. Both works focused on linear classifiers with Boolean features, whose robustness was measured in terms of the number of features that have to be modified to get a malicious sample misclassified as legitimate. For instance, this makes clearly sense in the case of text classifiers in spam filtering, where each feature is usually associated to a given word. Globerson

and Roweis [24] proposed a method to improve robustness against the so-called "feature deletion" attack, which consists in modifying a malicious sample to hide the presence of some attributes (e.g., "bad" words are often misspelled in spam emails). The method in [24] is based on a modification of the support vector machine learning algorithm. The strategy proposed in [35] will be explained in detail in Sect. 3.1, since it will be investigated in this work.

MCSs in adversarial classification tasks. MCSs have been firstly proposed in these tasks to improve classification accuracy, as in traditional (non-adversarial) classification problems [33,46,3,28,36]. A more specific motivation is that MCSs allow to deal in a natural way with heterogeneous features coming from different information sources, as in multi-modal biometric tasks [33,46]. A MCS architecture also allows to easily add new classifiers or detection modules to an existing system, which is a common practice in network intrusion detection and spam filtering tasks to counteract new kinds of attacks. Recently, it has also been argued that MCSs allow to improve classifier robustness in several adversarial classification tasks, based on the intuitive motivation that an adversary has to evade more than one classifier to make the whole ensemble ineffective [29, 52,44,46,47]. However, this claim has not been supported by any clear theoretical or empirical evidence so far. On the contrary, some empirical evidence against it was provided in [45], where it was shown that multi-modal biometric systems can be evaded by spoofing just one biometric trait.

The robustness of MCSs in adversarial classification problems has been also investigated in our previous works. Since adding new detection rules to a system in response to new attacks is a common practice in spam filtering and in network intrusion detection, in [5] we investigated whether adding classifiers to a given ensemble improves its robustness. In [4] we analysed randomisation strategies based on MCSs to prevent an adversary from gaining sufficient knowledge on a classifier to evade it. However, in these works we used the analytical model proposed by Dalvi et al. [17], which is based on unrealistic assumptions; thus, our results did not provide clear-cut conclusions. In [6] we provided some empirical evidence that a MCS architecture can be more robust than a single classifier architecture. Nevertheless, this work was limited to a logic OR of Boolean outputs of individual classifiers, which is not a standard MCS architecture. Lastly, in [7] we started investigating the use of bagging and the RSM to implement the strategy proposed in [35] for improving robustness. In this paper we extend our recent works [6,7] with a more thorough discussion on bagging and the RSM, and a more extensive experimental investigation.

3 Robust linear classifiers

In this section we describe the strategy proposed in [35] to improve the robustness of linear classifiers with Boolean features, and discuss how MCS construction methods like bagging and the RSM may be exploited to implement it.

3.1 Robust linear classifiers

Kolcz and Teo [35] proposed a strategy to improve the robustness of linear classifiers with Boolean features against manipulations of malicious samples aimed at getting them misclassified as legitimate at operation phase. From now on, we will denote such kind of manipulation as "attack", for short. The strategy proposed in [35] was targeted to scenarios in which classifier robustness can be related to the number of features that have to be modified in malicious samples to evade a classifier. A typical application scenario is a spam filtering task in which a text classifier is trained on Boolean features, each one denoting the presence or absence of a given word in an email. In this case it makes sense to evaluate robustness in terms of the number of words that a spammer has to add or to obfuscate in the original spam message to evade the classifier. In [35] it was observed that, if some features are highly discriminant on training samples, and the adversary knows them, he may manipulate his samples by modifying only the values of those features to evade the classifier. In practice, in applications like spam filtering and network intrusion detection, an adversary could guess the most discriminant features reasonably well. Based on this observation, and on the fact that in linear classifiers the most discriminant features are given the largest absolute weights, Kolcz and Teo [35] suggested that classifier robustness can be improved by avoiding to over-emphasise (under-emphasise) features which are highly (slightly) discriminant on training samples, since this would force the adversary to modify a higher number of feature values to evade the classifier. In other words, the absolute values of the feature weights should be distributed as evenly as possible. However, this may undermine the classifier accuracy on non-manipulated samples; thus, a trade-off between accuracy and robustness may be needed.

Among the different implementations of the strategy proposed in [35], we are interested in the one named *averaging*, which is based on a MCS approach. Let us first introduce some notation. We denote the decision function of a linear classifier as $f(\mathbf{x}) = \text{sign}\,(g(\mathbf{x})) \in \{-1,+1\}$, where $\mathbf{x} = (x_1,\ldots,x_n)$ is the n-dimensional feature vector of a given sample, $g(\mathbf{x}) = \sum_{i=i}^{n} w_i x_i + w_0$ is a linear discriminant function, $w_0, w_1, \ldots, w_n$ are the feature weights, to be set by a learning algorithm, and -1 and $+1$ are the labels of legitimate and malicious samples, respectively.

The *averaging* method consists in constructing a linear classifier whose discriminant function $g(\mathbf{x})$ is obtained by averaging the ones of L different linear classifiers $g_1(\mathbf{x})$, ..., $g_L(\mathbf{x})$. Such classifiers are obtained by running the chosen learning algorithm on the same training samples, but using L different, randomly selected subsets of the original feature set. This is obtained by setting to zero the values of non-selected features in all training samples. It is easy to see that the weights $w_1,\ldots,w_n$ of $g(\mathbf{x})$ equal the average of the corresponding weights of the L classifiers: $w_i = (1/L)\sum_{j=1}^{L} w_i^j$, $i = 1,\ldots,n$. Note that this increases the computational cost only at training phase, while the same cost of an individual linear classifier is incurred at operation phase.

The *averaging* method was borrowed from [51], where it was used to prevent feature overfitting and underfitting, which can happen when the training set is not sufficiently representative of the distribution of samples at operation phase. This is also the case of adversarial classification problems, which however were not considered in [51]. Quoting from [35], the rationale of this method is the following:

> By chance (due to randomness) in feature subset selection, highly indicative features are assigned to bags different from that of the less indicative features. Therefore, weights of the less indicative features will not be overwhelmed by the highly indicative ones during the modelling process.

3.2 MCS-based strategies to design robust linear classifiers

Kolcz and Teo [35] argued that the *averaging* method may produce a linear classifier whose weights are more evenly distributed than those of a single linear classifier trained with the same learning algorithm. The rationale was based on intuition, and was supported by some empirical results. However, a more thorough look at the *averaging* method reveals that it exhibits two interesting features, since it turns out to be very similar to a technique for ensemble construction which is well-known in the MCS field, i.e., the *random subspace method* (RSM) [30]. First, the RSM is known to be effective in improving classification accuracy with respect to a single classifier trained with the same learning algorithm. This makes it a stronger candidate as a method to reach a good trade-off between accuracy and robustness in adversarial classification tasks, provided that it is actually capable to produce more evenly distributed feature weights. Second, the RSM belongs to a family of well-known MCS construction techniques based on *randomisation*, whose other main representatives are bagging [9] and the random forest method [10]. This opens an interesting perspective on the potential usefulness of randomisation-based MCS techniques in adversarial classification tasks, besides the RSM. Indeed, since these techniques are rooted on the same underlying principle of the RSM, one may wonder whether they all may result not only in improving classification accuracy, but also in more evenly distributed weight values as a by-product, when applied to linear base classifiers, "naturally" implementing the strategy proposed in [35].

Randomisation-based techniques consist in constructing a MCS by training a given base classifier on different training sets, obtained by introducing some randomness in the original one. The first and most known technique of this kind is bagging [9]. It consists in training the individual classifiers on bootstrap replications of the original training set. Besides the RSM, which has been described above, another well-known method is the random forest [10]. It applies only to decision trees, and combines the idea of training set resampling and random feature subset selection. These techniques, and bagging in particular, have been extensively studied in the literature, in terms of why and under which conditions they are able to improve classification accuracy with respect to an individual base classifier, or the approximation error in regression problems, for bagging [9,18,27,21,12,11,30]. In particular, bagging is believed to work well for *unstable* base classifiers or regressors, whose decision or estimation function undergoes large changes for small perturbations of the training set. According to [9], bagging reduces such instability by reducing the variance component of the classification or estimation error. Other explanations have also been proposed; for instance, in [27] it was argued that bagging reduces the influence of outliers in the training set.

Despite all these works, to our knowledge the effect of randomisation-based techniques on the weight values of linear base classifiers was not considered by any author, even when the behaviour of bagging and the RSM method was specifically investigated for such kinds of classifiers [48,49]. Furthermore, although several theoretical models of bagging have been proposed [9,27,12,11], it turns out to be very difficult to exploit them to study its effect on the weight values of linear classifiers. Nevertheless, it is possible to provide at least an intuitive explanation of the potential side-effect of bagging of producing more evenly distributed weight vectors, similar to the intuitive motivation proposed in [35] for the RSM. When bagging is used, the training set of each base classifier is a bootstrap replication of the original training set. Therefore, each training sample may not appear in some bootstrap replications. One of the possible effects may be a reduction of the average weight of features which would be most discriminant on the whole training set, and an analogous increase of the average weight of least discriminant ones.

Since it is difficult to obtain analytical results to guide the analysis of randomisation-based methods, in this work we give an experimental analysis to get some hints on their behaviour, focusing on bagging and the RSM. We address

two main issues. The first one is to understand whether, and possibly under which conditions, they allow to produce more evenly distributed weight values than a single classifier. The second issue is to evaluate whether this allows to improve robustness, and what is the effect on the accuracy when a classifier is not under attack. Before presenting our experimental analysis, we report below the algorithms of bagging and the RSM, and describe in the next section the method we will use to evaluate robustness.

Algorithm 1 Bagging.

Input: A set of N training samples $T = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$, a learning algorithm $\mathscr{L}$, the ensemble size M.
Output: A classifier ensemble $\{f_1(\mathbf{x}), \ldots, f_M(\mathbf{x})\}$.

for $k = 1, \ldots, M$ **do**
  Construct a bootstrap replication T_k by randomly drawing with replacement N samples from T
  Train a classifier $f_k(\mathbf{x})$ by running $\mathscr{L}$ on T_k
end for
return $\{f_1(\mathbf{x}), \ldots, f_M(\mathbf{x})\}$

Algorithm 2 The Random Subspace Method.

Input: A set of n-dimensional feature vectors of training samples $T = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$, a learning algorithm $\mathscr{L}$, the ensemble size M, the feature subset size F.
Output: A classifier ensemble $\{f_1(\mathbf{x}), \ldots, f_M(\mathbf{x})\}$.

for $k = 1, \ldots, M$ **do**
  Randomly select a subset of F features, and construct a training set T_k by projecting the samples in T on such feature subset
  Train a classifier $f_k(\mathbf{x})$ by running $\mathscr{L}$ on T_k
end for
return $\{f_1(\mathbf{x}), \ldots, f_M(\mathbf{x})\}$

4 Robustness evaluation

Standard methods for classifier performance evaluation are based on estimating the generalisation capability using a set of samples collected at design phase, through cross-validation or analogous techniques. However, they can not provide information about the behaviour of a classifier under attack, for two main reasons. First, malicious samples belonging to training data may not have been subjected to adversarial modifications aimed at evading a classifier. For example, in standard biometric verification systems the training set does not include any *spoofed* trait, namely, fake biometric traits reproduced by an impostor to be recognised as a genuine user. This does not allow to assess the behaviour of such a system against a spoof attack. Second, even if the collected malicious samples include attacks (as it may happen in spam filtering and network intrusion detection tasks), they were not purposely crafted to evade the system under design, but the one which was operating when they were collected [35]. Therefore, they are not representative of the attacks that may be subsequently performed against the classifier under design. However, despite the relevance of adversarial classification tasks, no works in the literature have proposed practical robustness evaluation methods so far.

Evaluating classifier robustness in adversarial classification tasks is a complex issue, which is outside the scope of this paper, and is the subject of an ongoing work [13,37], also by the authors. In the following we describe the method used in our experiments, which is derived from a more general methodology we are currently developing, and tailored to the particular application scenario considered in this paper.

Given that the samples collected to design a classifier can not be considered representative of attacks observed at operation phase, our methodology to assess classifier robustness is based on *simulating* attacks carefully targeted against the classifier under design. More precisely, a classifier is first trained on the original training set, then its performance is evaluated on a *modified* testing set, in which *all* malicious samples have been modified to simulate the effect of an attack of interest. In the considered application scenario, the effect of attacks is simply to change the value of some features from 0 to 1, or vice versa. They can thus be simulated by directly modifying the feature vectors of malicious samples, without the need of manipulating the original samples (e.g., emails or network packets).

In the case of linear classifiers, it is of interest to evaluate robustness in a *worst-case* scenario. In this case the adversary is assumed to have complete knowledge of the classifier, namely, of the feature set and the decision function, and is always able to get a malicious sample misclassified as legitimate by modifying the minimum number of feature values. Another case of interest is the one in which the adversary has only an approximate knowledge of the classifier, as happens in many real cases. For instance, two well-known attacks against text classifiers used in spam filters are the so-called *good word insertion* (GWI) and *bad word obfuscation* (BWO). They respectively consist in modifying a spam email by *inserting* randomly chosen words which are not likely to appear in spam messages, and by *obfuscating* (e.g., by misspelling) typical "spammy" words. When evaluating the robustness of a classifier against these attacks, it is reasonable to assume that the adversary can guess a subset of the good and bad words used by the classifier, and their discriminant capability. An attack based on such partial knowledge can thus be simulated by modifying only the corresponding subset of features in malicious samples.

In the case of text classifiers for spam filtering it is also useful to evaluate robustness as a function of the *attack strength*,

namely, the maximum number of words (i.e., features) which can be modified (i.e., either *inserted* or *obfuscated*) in a spam email.

In our experiments we evaluate the robustness of linear classifiers considering both worst- and non-worst-case attack scenarios. The feature vectors of *all* malicious samples in the testing set are modified to simulate the effect of a given attack, by changing up to n_{MAX} feature values, which corresponds to the attack strength. The exact algorithms used to simulate the attacks are described below.

Worst-case attack. For a given feature vector $\mathbf{x}$ of a malicious sample, the features to modify (up to a given number n_{MAX}) to get a new feature vector $\mathbf{x}'$ are the ones which minimise the discriminant function $g(\mathbf{x}')$. This leads to the maximum decrease in performance for the given n_{MAX}. It is not difficult to see that, for linear classifiers with Boolean features, $\mathbf{x}'$ can be found as follows. First, the weights w_1, w_2, …, w_n have to be sorted in descending order of their absolute value, and the features have to be sorted accordingly. Note that this step has to be carried out only once, after classifier training. We denote the sorted weights and features respectively as $w_{(1)}, w_{(2)}, \ldots w_{(n)}$, and $x_{(1)}, x_{(2)}, \ldots, x_{(n)}$, where $|w_{(1)}| \geq |w_{(2)}| \geq \ldots \geq |w_{(n)}|$. Then, for $i = 1, 2, \ldots, n$, and until the number of modified feature values does not exceed n_{MAX}:

- if $x_{(i)} = 1$ and $w_{(i)} > 0$, $x_{(i)}$ is set to 0;
- if $x_{(i)} = 0$ and $w_{(i)} < 0$, $x_{(i)}$ is set to 1;
- otherwise, $x_{(i)}$ is left unchanged.

The exact procedure is reported as Algorithm 3.

Algorithm 3 Worst-case attack for a linear classifier with Boolean features.

Input: $\mathbf{x}$, the malicious sample to be modified; n_{MAX}, the maximum number of features which can be modified.
Output: $\mathbf{x}'$, the feature vector of the modified malicious sample.

$\mathbf{x}' \leftarrow \mathbf{x}$, $n_{\mathrm{mod}} \leftarrow 0$, $k \leftarrow 1$
while $n_{\mathrm{mod}} \leq n_{\mathrm{MAX}}$ and $k \leq n$ **do**
  if $x_{(k)} = 1$ **and** $w_{(k)} > 0$ **then**
    $x'_{(k)} \leftarrow 0$, $n_{\mathrm{mod}} \leftarrow n_{\mathrm{mod}} + 1$
  else if $x_{(k)} = 0$ **and** $w_{(k)} < 0$ **then**
    $x'_{(k)} \leftarrow 1$, $n_{\mathrm{mod}} \leftarrow n_{\mathrm{mod}} + 1$
  end if
  $k \leftarrow k + 1$
end while
return $\mathbf{x}'$

Non-worst-case attacks. We assume that the adversary devises his modifications to malicious samples based on an incomplete knowledge of the classifier's decision function, namely, on an approximation of the feature weights. The consequence is that he overestimates or underestimates the importance of some features, leading to a non-optimal choice of the features to modify. In practice, this corresponds to the case in which the adversary makes some educated guess on what the most discriminant features are. To simulate this scenario it is possible to use again Algorithm 3, but shuffling the order of features which were sorted according to descending absolute values of their weights. We considered two cases corresponding to different levels of the adversary's knowledge: swapping $n/2$ pairs of randomly chosen features in the sequence $x_{(1)}, x_{(2)}, \ldots, x_{(n)}$ (which we will refer to as "non-worst-case" attack), and shuffling them completely at random ("random" attack). Note that the latter corresponds to an adversary with no knowledge on the discriminant capability of the features.

5 Experimental results

The goal of our experiments is to investigate whether, and under which conditions, bagging and the RSM produce more evenly distributed weight values and exhibit a higher robustness than a single base classifier trained with the same learning algorithm. We used a spam filtering task as a case study. The experimental setup is described in Sect. 5.1, and the results are reported in Sects. 5.2 and 5.3.

5.1 Experimental setup

5.1.1 Data sets and classifiers

We used the benchmark TREC 2007 email corpus [15], publicly available at `http://plg.uwaterloo.ca/~gvcormac/treccorpus07`. It is made up of 75,419 real emails (25,220 legitimate and 50,199 spam messages), which were collected between April and July 2007.

Two kinds of linear classifiers with Boolean features were considered: text classifiers proposed in the spam filtering literature, and classifiers which can be used to automatically tune the linear decision function of the SpamAssassin filter.

We used support vector machines (SVMs) with linear kernel [19] and logistic regression (LR) [35] as text classifiers. The first 20,000 emails (in chronological order) of TREC 2007 were used as training set, and the next 20,000 emails as testing set. The bag-of-words feature model was used. We first extracted the features (words) from training emails using the tokenization method of SpamAssassin, and then selected $n = 20,000$ distinct features using a supervised feature selection approach based on the information gain criterion [39]. SVMs were trained using the libSVM software [14]; the C parameter of their learning algorithm was chosen among the values in $\{0.001, 0.01, 0.1, 1, 10, 100\}$, by maximising the performance measure (see below) through a 5-fold cross validation on training data. The LR classifier was trained using an online gradient descent algorithm [8].

In the experiments on the SpamAssassin filter we used its latest available version, 3.2.5. SpamAssassin consists of some hundred Boolean tests, each aimed at detecting a particular characteristic of spam or legitimate emails, like the presence of a typical spam word or an email header malformation produced by known automatic spam generation tools [50]. Some tests are associated to the output of a text classifier. The outcomes of tests are numerically coded as 1 and 0, respectively for 'True' and 'False'. An email is labelled as spam if the linear combination of the corresponding test outputs exceeds a given threshold, otherwise it is labelled as legitimate. SpamAssassin can thus be viewed as a linear classifier with Boolean features. Default values are provided for the threshold and the feature weights[1]. They are obtained by manually tuning the ones produced by a linear classifier, which is currently a perceptron[2]. Note that manual tuning is done by the SpamAssassin developers to improve robustness, given that the weights produced by learning algorithms are not considered reliable enough. For this reason, it is interesting to evaluate whether the strategy proposed in [35] to improve robustness of linear classifiers can be effectively exploited in the practical application scenario of SpamAssassin. In our experiments, we used again the SVM and LR linear classifiers, and also considered the default weights for comparison. Since about half of the SpamAssassin tests gave zero ('False') as output value for all the TREC 2007 emails, we disregarded them to reduce the computational complexity. The remaining number of tests (features) turned out to be $n = 549$. Since SpamAssassin includes a text classifier, we used the first 10,000 emails (in chronological order) of TREC 2007 to train it. The next 20,000 emails were used as training set for our linear classifiers, and the following 20,000 ones were used as testing set.

The performance of bagging and the RSM was evaluated for different values of their parameters, to investigate how they affect robustness and accuracy. Ensemble of 5, 10 and 20 base classifiers were considered. For the RSM, we considered feature subset sizes equal to 30%, 50% and 80% of the original feature set size. For bagging we also considered different training set sizes of the bootstrap replications (which is one of the variants proposed in the literature), equal to 30%, 50% and 100% of the original training set size. Since bagging and the RSM are respectively based on a random selection of training samples and feature subsets, we averaged the results over 5 runs of the experiments.

[1] For a detailed list of tests and corresponding weights, see `http://spamassassin.apache.org/tests_3_2_x.html`.

[2] `http://spamassassin.apache.org/full/3.0.x/dist/masses/README.perceptron`

5.1.2 Performance and weight evenness measures

Robustness was evaluated as described in Sect. 4, under the worst-case and the two non-worst-case scenarios of Sect. 4. Classification performance was evaluated with a measure derived from the area under the ROC curve (AUC). Since in tasks like spam filtering false positive (FP) errors are typically more harmful than false negative (FN) ones, the region of interest of the ROC curve is restricted to low FP rate values. As proposed in [35], we used a more informative measure than the AUC, defined as the area of the region of the ROC curve corresponding to FP rates in the range $[0, 0.1]$: $\mathrm{AUC}_{10\%} = \int_0^{0.1} \mathrm{TP}(\mathrm{FP})\mathrm{dFP}$. Note that $\mathrm{AUC}_{10\%} \in [0, 0.1]$.

To evaluate the weight evenness we used a measure proposed in [35]. It is defined as a function $F(k)$ given by the ratio of the sum of the k highest absolute weight values to the sum of all absolute weight values, for $k = 1, \ldots, n$:

$$F(k) = \frac{\sum_{i=1}^{k} |w_{(i)}|}{\sum_{j=1}^{n} |w_j|}, \tag{1}$$

where $|w_{(1)}|, |w_{(2)}|, \ldots, |w_{(n)}|$ denote the weights sorted in descending order of their absolute values, i.e., $|w_{(1)}| \geq |w_{(2)}| \geq \ldots \geq |w_{(n)}|$. Note that w_0 is not considered, since it is not associated to any feature. The most even weight distribution is given by identical weights, which corresponds to $F(k) = k/n$. The most uneven distribution is attained when only one weight is different from zero, and thus $F(k) = 1$ for each k value. Therefore, a flatter line corresponds to a more even weight distribution [35]. Since $F(k)$ is not a scalar, we chose to use a more concise, scalar measure given by:

$$E = \frac{2}{n-1} \left[n - \sum_{k=1}^{n} F(k) \right]. \tag{2}$$

It is easy to see that the range of E is $[0, 1]$, and that $E = 0$ and $E = 1$ correspond respectively to the most uneven and to the most even weight distribution.

5.2 Experimental results on text classifiers

We analyse first the classification performance of all the considered linear text classifiers, when they are not under attack, and the evenness of their weight distribution. Then we analyse their robustness across the different attack scenarios.

In Tables 1 and 2 we report the testing $\mathrm{AUC}_{10\%}$ for the LR and SVM base classifiers, when they are not under attack, and their weight evenness E (Eq. 2). We remind the reader that $\mathrm{AUC}_{10\%}$ ranges from 0 to 0.1, and that the results for MCSs were averaged over 5 repetitions, since the MCSs were built using different random feature subsets (RSM) or bootstrap replications (bagging).

Table 1 Average classification performance ($AUC_{10\%}$) and weight evenness (E) for the individual LR text classifier (first row) and the LR ensembles built by the RSM and bagging, when they are not under attack. In the row headers LR-X-Y-Z, X denotes the MCS technique, Y the ensemble size, and Z the feature subset size (for the RSM) or the training set size (for bagging).

classifier	$AUC_{10\%}$	E
LR	0.0994	0.309
LR-RSM-5-30	0.0996	0.258 ± 0.0043
LR-RSM-5-50	0.0996	0.308 ± 0.0031
LR-RSM-5-80	0.0996	0.325 ± 0.0016
LR-RSM-10-30	0.0996	0.311 ± 0.0037
LR-RSM-10-50	0.0996	0.33 ± 0.0023
LR-RSM-10-80	0.0996	0.338 ± 0.0023
LR-RSM-20-30	0.0996	0.335 ± 0.0029
LR-RSM-20-50	0.0996	0.341 ± 0.0029
LR-RSM-20-80	0.0996	0.342 ± 0.0029
LR-bag-5-30	0.0996	0.33 ± 0.0046
LR-bag-5-50	0.0997	0.337 ± 0.0037
LR-bag-5-100	0.0996	0.335 ± 0.0046
LR-bag-10-30	0.0997	0.341 ± 0.0046
LR-bag-10-50	0.0996	0.338 ± 0.002
LR-bag-10-100	0.0996	0.341 ± 0.0035
LR-bag-20-30	0.0997	0.343 ± 0.0047
LR-bag-20-50	0.0997	0.346 ± 0.0026
LR-bag-20-100	0.0997	0.345 ± 0.0029

Table 2 Average classification performance ($AUC_{10\%}$) and weight evenness (E) for the individual SVM text classifier (first row) and the SVM ensembles built by the RSM and bagging, when they are not under attack. See caption of Table 1 for the meaning of row headers.

classifier	$AUC_{10\%}$	E
SVM	0.0993	0.233
SVM-RSM-5-30	0.0993	0.192 ± 0.0018
SVM-RSM-5-50	0.0993	0.225 ± 0.004
SVM-RSM-5-80	0.0993	0.232 ± 0.0011
SVM-RSM-10-30	0.0994	0.232 ± 0.0034
SVM-RSM-10-50	0.0993	0.239 ± 0.0018
SVM-RSM-10-80	0.0994	0.237
SVM-RSM-20-30	0.0992	0.251 ± 0.0027
SVM-RSM-20-50	0.0993	0.248 ± 0.0012
SVM-RSM-20-80	0.0994	0.238
SVM-bag-5-30	0.0997	0.261 ± 0.006
SVM-bag-5-50	0.0997	0.258 ± 0.008
SVM-bag-5-100	0.0995	0.252 ± 0.0019
SVM-bag-10-30	0.0997	0.267 ± 0.0051
SVM-bag-10-50	0.0997	0.263 ± 0.0019
SVM-bag-10-100	0.0996	0.254 ± 0.0024
SVM-bag-20-30	0.0997	0.274 ± 0.0013
SVM-bag-20-50	0.0997	0.265 ± 0.0018
SVM-bag-20-100	0.0996	0.252 ± 0.0013

The $AUC_{10\%}$ values of the two individual base classifiers are similar and very high. Nevertheless, both bagging and the RSM almost always slightly outperformed the corresponding individual classifier. Moreover, bagging and the RSM also produced more evenly distributed weights than the corresponding individual classifiers, with some exception for the RSM, when only 5 classifiers were combined, or small feature subset sizes were used (30%). Note also that the weight evenness of the MCSs almost always increases as the ensemble size increases, while its behaviour as a function of the training size (for bagging) or feature subset size (for RSM) is not clear-cut.

Consider now the robustness under the different attack scenarios.

Worst-case attack. Figs. 1 and 2 show the robustness against a worst-case attack (these figures are best viewed in colour), in terms of $AUC_{10\%}$ as a function of the attack strength, namely, the maximum number of modified feature values. Note that the $AUC_{10\%}$ values for zero attack strength are the ones attained by the classifiers when they are not under attack, and are therefore the same as in Tables 1 and 2. Note also that the $AUC_{10\%}$ values drop to zero as the attack strength increases. This means that, after a given number of features has been modified, all spam emails are misclassified as legitimate. Although this is not the focus of this work, we point out that, in this kind of analysis, it is interesting for the designer of a pattern recognition system to analyse the behaviour of $AUC_{10\%}$ as the attack strength increases: the more graceful its decrease, the more robust the classifier.

Figs. 1 and 2 show that the behaviour of bagging and the RSM in terms of robustness, with respect to the corresponding base classifier, almost always agrees with the one hypothesised in [35] in terms of weight evenness. In particular, bagging always improved the robustness of both the LR and SVM base classifiers, besides providing more evenly distributed weights as discussed above. Similarly, the RSM slightly improved the robustness of the LR classifier, except when the ensemble size was 5, and its performance improved as the ensemble size increased. These are the same conditions under which the weight evenness of the RSM increased, and exceeded the one of the individual classifier. Finally, it can also be seen that the more evenly distributed the weight values of RSM and bagging, the higher their robustness. The only exception to this behaviour is that the RSM never improved the robustness of the SVM classifier, although it produced slightly more evenly distributed weights for larger ensemble sizes or feature subset sizes.

The above results show a rather clear correlation between the increase in weight evenness and the increase in robustness, which provides further support to the strategy suggested in [35]. Our results also suggest that both bagging and the RSM are able to provide more evenly distributed weights and to improve the robustness of an individual linear classifier, provided that their parameters are properly chosen. In particular, in the considered data set the RSM seemed more effective for large ensemble and feature subset sizes, while bagging benefited from small training set sizes.

Non-worst-case attacks. For the sake of brevity, we report in Fig. 3 only the results attained with an ensemble size of 10, when bagging was trained on 50% of the orig-

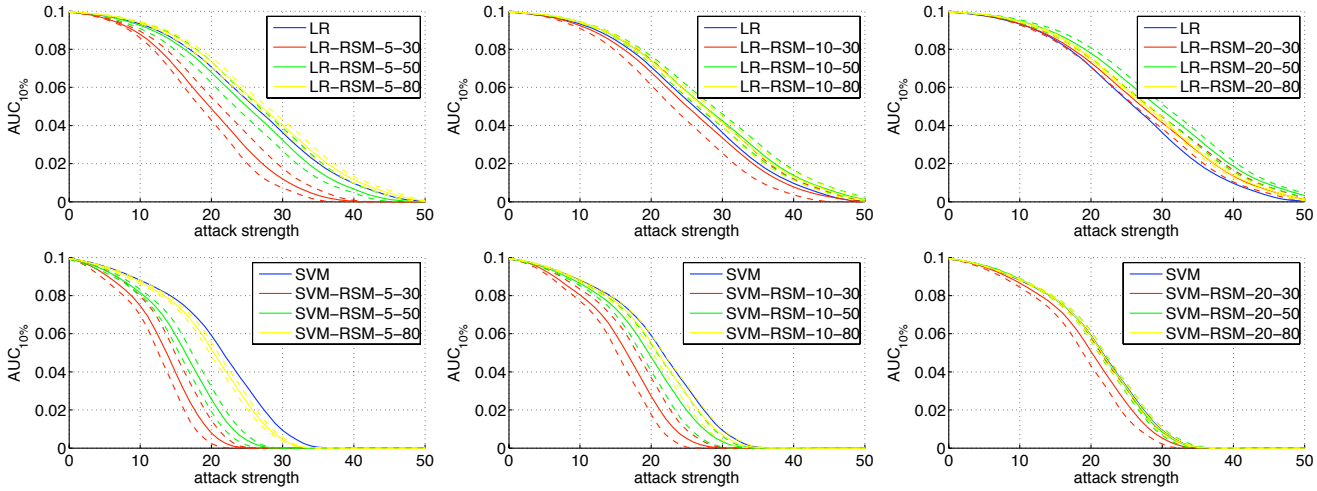

Fig. 1 Average $AUC_{10\%}$ (solid lines) with standard deviation (dashed lines), vs the attack strength, for the single LR (top) and SVM (bottom) text classifiers, and for their ensembles built with the RSM, against the worst-case attack. In the legends, X-RSM-Y-Z denotes the base classifier (X), the ensemble size (Y), and the percentage feature subset size (Z).

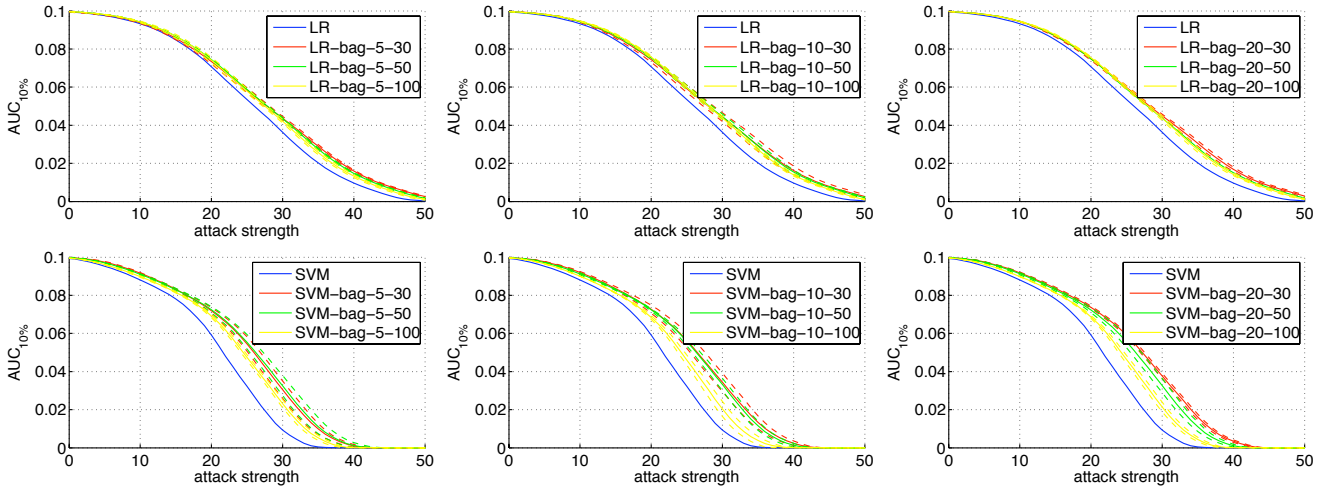

Fig. 2 Average $AUC_{10\%}$ (solid lines) with standard deviation (dashed lines), vs the attack strength, for the single LR (top) and SVM (bottom) text classifiers, and for their ensembles built with bagging, against the worst-case attack. See caption of Fig. 1 for the meaning of figure legends.

inal training set size, and the RSM was trained on subsets of 50% of the original feature set. For an easier comparison, we also report the corresponding results of the worst-case attack. The value of the weight evenness measure E is the same as above, since only the attack strategy is different. Note that in the considered cases the weight evenness was higher than the one of the individual classifier.

As expected, the "non-worst-case" attack scenario is less harmful for the classifiers than the worst-case attack. This can be seen from the higher $\text{AUC}_{10\%}$ values attained under the former attack, being equal the attack strength. Analogously, the "random" attack (simulating the case when the adversary has no knowledge on the relative discriminant capability of features) is the least harmful. For example, the $\text{AUC}_{10\%}$ of the individual LR classifier and of ensembles of LR classifiers is reduced to 0.08 by modifying less than 20 features (i.e., words in spam emails) in the worst-case, while more than 60 features have to be modified in the "non-worst-case" attack, and more than 400 features in the "random" attack.

From Fig. 3 it can be seen that both bagging and the RSM slightly improved the robustness of the LR base classifier (taking into account the larger variance than in the worst case attack), while the robustness of the SVM classifier is improved by both bagging and the RSM under the "random" attack, and by bagging only under the "non-worst-case" attack. As in the worst-case scenario, the relationship between classifier robustness and weight evenness agrees rather well with the one hypothesised in [35].

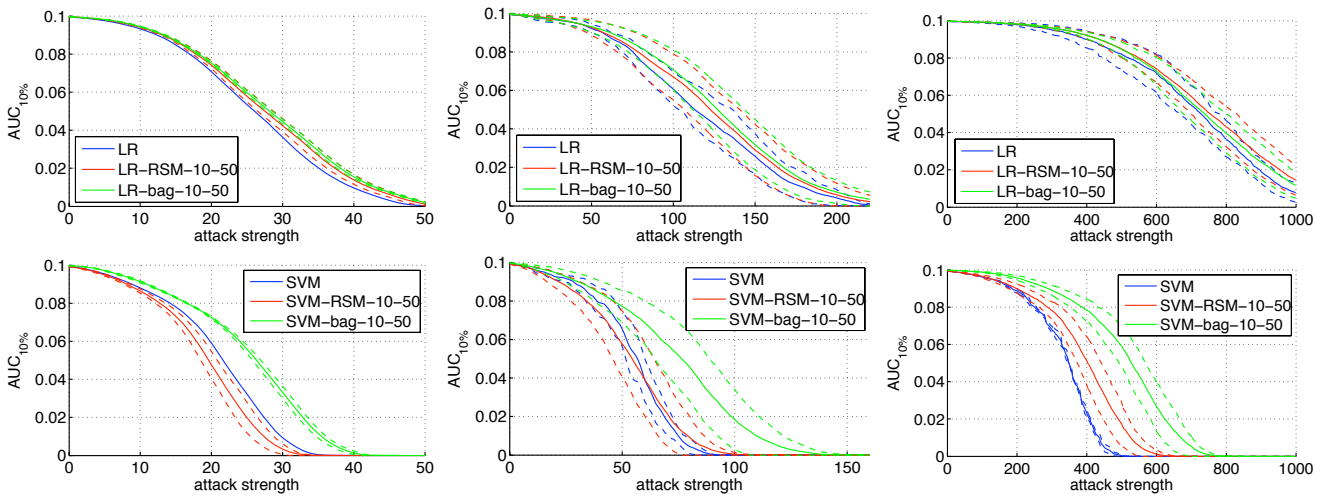

Fig. 3 Average $AUC_{10\%}$ (solid lines) with standard deviation (dashed lines), vs the attack strength, for the single LR (top) and SVM (bottom) text classifiers, and for their ensembles built with bagging and the RSM. Left: worst-case attack; middle: non-worst-case attack; right: random attack. See the caption of Figs. 1 and 2 for the meaning of figure legends.

Table 3 Average classification performance ($AUC_{10\%}$) and weight evenness (E) of SpamAssassin when it is not under attack, attained by the default weights (first row), the individual LR text classifier (second row), and the LR ensembles built by the RSM and bagging. The standard deviation is reported, when greater than 10^{-3}.

classifier	$AUC_{10\%}$	E
def. weights	0.0974	0.601
LR	0.0984	0.2
LR-RSM-5-30	0.0976	0.219 ± 0.025
LR-RSM-5-50	0.0979	0.222 ± 0.022
LR-RSM-5-80	0.0981	0.214 ± 0.011
LR-RSM-10-30	0.0971	0.269 ± 0.015
LR-RSM-10-50	0.0978	0.256 ± 0.016
LR-RSM-10-80	0.0983	0.213 ± 0.0076
LR-RSM-20-30	0.0977	0.287 ± 0.0073
LR-RSM-20-50	0.098	0.271 ± 0.011
LR-RSM-20-80	0.0982	0.231 ± 0.01
LR-bag-5-30	0.0977	0.19 ± 0.0033
LR-bag-5-50	0.098	0.195 ± 0.0023
LR-bag-5-100	0.0983	0.206 ± 0.004
LR-bag-10-30	0.0977	0.191 ± 0.0013
LR-bag-10-50	0.098	0.196 ± 0.0037
LR-bag-10-100	0.0983	0.206 ± 0.0012
LR-bag-20-30	0.0977	0.192 ± 0.0014
LR-bag-20-50	0.098	0.196
LR-bag-20-100	0.0983	0.206 ± 0.0013

Table 4 Average classification performance ($AUC_{10\%}$) and weight evenness (E) of SpamAssassin when it is not under attack, attained by the default weights (first row), the individual SVM text classifier (second row), and the SVM ensembles built by the RSM and bagging. The standard deviation is reported, when greater than 10^{-3}.

classifier	$AUC_{10\%}$	E
def. weights	0.0974	0.601
SVM	0.0983	0.147
SVM-RSM-5-30	0.0966 ± 0.0013	0.251 ± 0.069
SVM-RSM-5-50	0.0967 ± 0.0012	0.271 ± 0.03
SVM-RSM-5-80	0.0978	0.186 ± 0.063
SVM-RSM-10-30	0.0952 ± 0.0017	0.394 ± 0.027
SVM-RSM-10-50	0.097 ± 0.0012	0.314 ± 0.029
SVM-RSM-10-80	0.098	0.177 ± 0.046
SVM-RSM-20-30	0.0968	0.434 ± 0.025
SVM-RSM-20-50	0.0973	0.365 ± 0.021
SVM-RSM-20-80	0.0982	0.207 ± 0.058
SVM-bag-5-30	0.0983	0.126 ± 0.0092
SVM-bag-5-50	0.0983	0.142 ± 0.0081
SVM-bag-5-100	0.0985	0.15 ± 0.0077
SVM-bag-10-30	0.0984	0.137 ± 0.0079
SVM-bag-10-50	0.0985	0.154 ± 0.0072
SVM-bag-10-100	0.0984	0.159 ± 0.0043
SVM-bag-20-30	0.0984	0.143 ± 0.0082
SVM-bag-20-50	0.0984	0.154 ± 0.0062
SVM-bag-20-100	0.0985	0.162 ± 0.0037

5.3 Experimental results on SpamAssassin

Tables 3 and 4 show the $AUC_{10\%}$ and weight evenness values when SpamAssassin is not under attack. The individual LR and SVM classifiers exhibited a similar $AUC_{10\%}$ value, while LR produced more evenly distributed weights. Bagging performed slightly worse than the single LR classifier and slightly better than the SVM one, and did not always improve their weight evenness. The RSM often performed worse than the corresponding individual classifier (sometimes, significantly worse), but always produced more evenly distributed weights.

The weight evenness tends to increase as the feature subset size of RSM decreases (which is opposite to the behaviour observed in previous experiments), and as the training set size of bagging increases. The weight evenness tends to increase also as the ensemble size increases, as in the previous experiments. The behaviour of the RSM in this respect

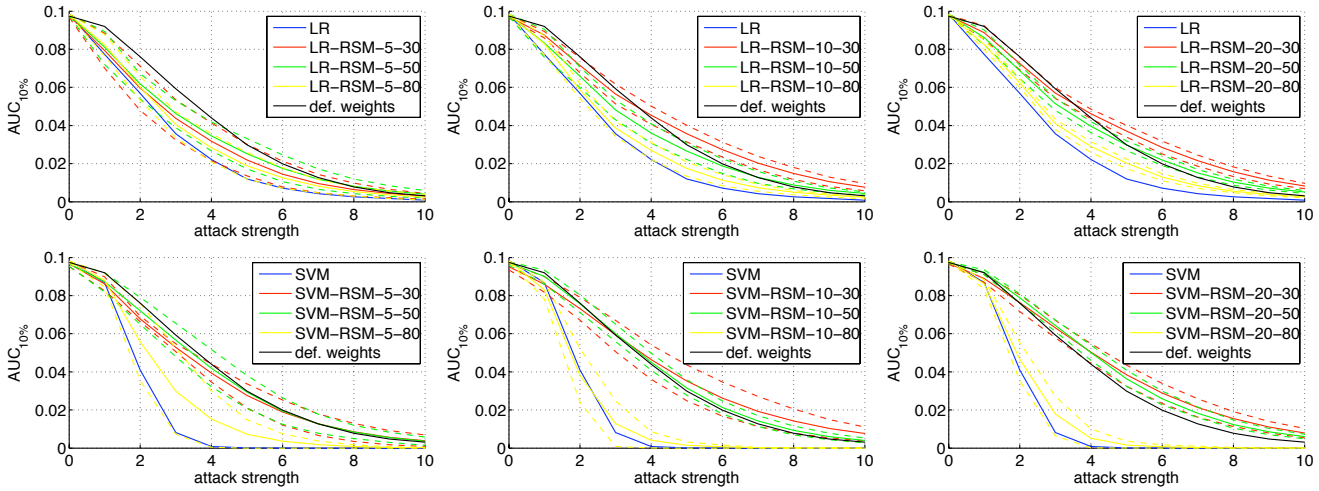

Fig. 4 Average $AUC_{10\%}$ (solid lines) with standard deviation (dashed lines) of SpamAssassin, vs the attack strength, attained by the default weights and by weights produced by the single LR (top) and SVM (bottom) text classifiers, and by their ensembles built with the RSM, against the worst-case attack. See caption of Fig. 1 for the meaning of figure legends.

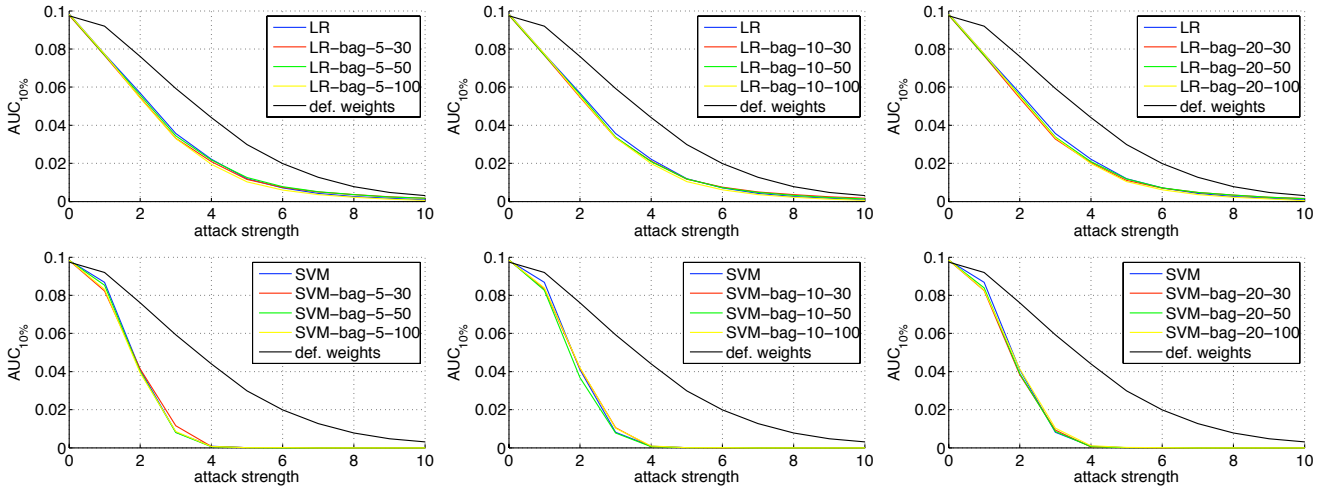

Fig. 5 Average $AUC_{10\%}$ (solid lines) with standard deviation (dashed lines) of SpamAssassin, vs the attack strength, attained by the default weights and by weights produced by the single LR (top) and SVM (bottom) text classifiers, and by their ensembles built with bagging, against the worst-case attack. See caption of Fig. 1 for the meaning of figure legends.

can be explained with the fact that, among SpamAssassin tests, only the nine ones associated to its text classifier exhibit a very high discriminant capability. Hence, the base classifiers are likely to assign high absolute weight values only to these tests. On the contrary, when using small subsets of features to train the base classifiers, some of them are likely to include few (or none) of the features (tests) associated to the text classifier. Consequently, the other features get higher absolute weight values. Accordingly, the average values of the feature weights of the individual classifier of the ensemble are likely to be more evenly distributed.

Consider finally the default weight values of SpamAssassin. They attain a worse classification performance than the LR and SVM individual classifier, as well as most of their ensembles. However, they are also much more evenly distributed than any set of weight produced by the considered learning algorithms. The criteria used by the SpamAssassin developers to manually tune the weights appear therefore coherent with the strategy proposed in [35], and seem to push the trade-off between classification performance (without attacks) and robustness to attacks (namely, weight evenness) in favour of the latter, much more than the considered learning algorithms.

Worst-case attack. Robustness against the worst-case attack is shown in Figs. 4 and 5. It is easy to see that the robustness of both bagging and the RSM follows a behaviour which is in good agreement with their weight evenness, as hypothesised in [35]. RSM-based ensembles are almost al-

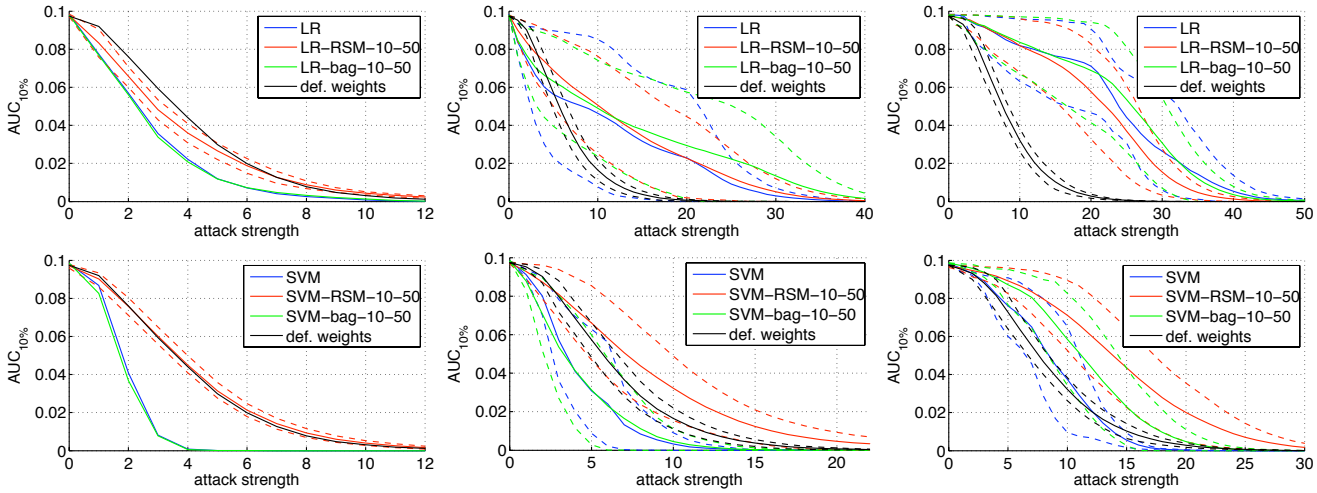

Fig. 6 Average $AUC_{10\%}$ (solid lines) with standard deviation (dashed lines) of SpamAssassin, vs the attack strength, attained by the default weights and by weights produced by the single LR (top) and SVM (bottom) text classifiers, and by their ensembles built with the RSM and bagging. Left: worst-case attack; middle: non-worst-case attack; right: random attack. See caption of Fig. 1 for the meaning of figure legends.

ways more robust than the corresponding base classifier, and their robustness increases under the same conditions under which their weight evenness increases. Analogously, the robustness of bagging-based ensembles is very similar to the one of the corresponding base classifier, as the evenness of their weights. Contrary to text classifiers, the RSM turned out to be more effective than bagging in improving the robustness of SpamAssassin, at least for the ensemble parameters considered here. Finally, it is interesting to note that, while the default SpamAssassin weights exhibit a much higher robustness than the individual LR and SVM classifiers, and of their ensembles built with bagging, the RSM attained the same robustness for high attack strengths, and sometimes even a better robustness. This result is promising under the viewpoint of the reliability of pattern recognition systems in adversarial classification tasks.

Non-worst-case attacks. As in Sect. 5.2 we only consider ensembles of 10 classifiers, trained on 50% of the original features for the RSM, and on 50% of the training set size for bagging. In Fig. 6 the robustness of the different classifiers considered is shown, as well as the default SpamAssassin weights, for the "non-worst-case" and "random" attacks. The same considerations of Sect. 5.2 apply here, about classifier robustness under the different scenarios. Fig. 6 shows that the only significant improvement in robustness is attained by the RSM with the SVM classifier. Note that the RSM applied to SVMs provided the highest robustness improvement also under the worst-case attack. The relationship between the weight evenness and robustness is thus confirmed by these results also in non-worst-case attack scenarios. Note finally that in the "non-worst-case" attack both the individual LR and SVM classifiers attain a similar robustness as the default SpamAssassin weights, while they attain an even better robustness in the "random" attack scenario.

To sum up, the RSM turned out to be quite effective in improving both the accuracy and robustness of the LR and SVM base classifiers, when they were used to set the weights of the SpamAssassin's tests. Instead, bagging turned out to be not better than the corresponding base classifier in this task. These behaviour is somewhat opposite to the one observed in text classifiers. A possible explanation is that the RSM is much more effective than bagging in forcing the evenness of weight distributions, when few features exhibit a relatively high discriminant capability. This is indeed the case of the SpamAssassin tests, as pointed out above, while in the case of text classifiers (where bagging was slightly more effective than the RSM), the feature weights of the base classifiers were relatively more evenly distributed. Nevertheless, also these experiments showed a good agreement between the observed behaviour of the weight evenness and the robustness of bagging and the RSM, and the behaviour hypothesised in [35].

6 Conclusions

The use of pattern recognition systems in adversarial environments poses new challenges to researchers in the pattern recognition field, since current theory and design methods do not take into account the possibility of actively manipulating data in an adversarial way, to make a classifier ineffective. In this work, after introducing adversarial classification and giving an overview of the (still limited) literature to make the reader aware of the main issues in this new research field, we focused on the issue of how to design pattern classifiers which are robust to manipulations of ma-

licious samples made at operation phase. In particular, we considered one of the very few practical strategies proposed so far to improve classifier robustness, which is not tied to a specific application and a specific attack [35]. Such strategy was proposed for linear classifiers with Boolean features, for contexts in which robustness can be measured in terms of the number of feature values which have to be modified to get a malicious sample misclassified as legitimate. It consists in keeping the feature weights in the discriminant function as evenly distributed as possible, with the rationale that this forces an adversary to modify a large number of feature values to evade the classifier. In this paper we investigated the possibility of implementing this strategy using the well known RSM and bagging randomisation-based MCS construction techniques. This was implicitly envisaged in [35], and in the case of linear classifiers it has the advantage of not increasing the computational complexity at operation phase.

In this paper we gave only an experimental investigation, given that it turned out to be not straightforward to analytically evaluate the effect of randomisation-based MCS construction techniques on the evenness of weight distributions of linear classifiers. Our experiments were carried out on a case study related to a spam filtering task, and encompassed the two kinds of linear classifiers which have been considered so far in the spam filtering literature, as well as classifiers which are used in real spam filters: text classifiers in which features correspond to words in emails, and classifiers used to set the weights of the decision function of the SpamAssassin filter.

While the original goal of MCSs is to improve classification accuracy with respect to a single base classifier, we found evidence that the RSM and bagging can also produce more evenly distributed weights as a side effect, and that this often results in improving robustness under attack. The extent of this effect turned out to depend on the the training set size for bagging, the feature subset size for the RSM, and the ensemble size for both. Although the effect of these parameters could not be clearly explained by our results, some patterns seemed to emerge. The RSM seems more effective than bagging in improving robustness, if a very small number of features exhibit a very high discriminant capability, as in the case of the SpamAssassin filter. A possible explanation for the RSM is the one given in [35]: since in the RSM the ensemble members are trained on subsets of the original feature set, the most discriminant features are likely to be not used by several individual classifiers. Their average weights will thus be lower than in a single classifier trained on all features, while the average weights of the other features will be higher, which implies a more even weight distribution. We add to this explanation that bagging is less likely to produce this effect, since its individual classifiers are trained on the same features. On the other hand, we observed that bagging performs better than the RSM when the weights of the original feature set are more evenly distributed, as happened in text classifiers.

Furthermore, we found that the weight evenness of both the RSM and bagging tends to increase as the ensemble size increases. Finally, the behaviour of the weight evenness produced by the RSM, as the feature subset size of individual classifiers increases, was somewhat contradictory across the two kinds of linear classifiers: it increased for text classifiers, and decreased in the SpamAssassin filter. For bagging, the weight evenness increased as the training set size of individual classifiers increases, but only on text classifiers.

To sum up, our results show that randomisation-based MCS construction techniques can be useful to improve the robustness of linear classifiers with Boolean features in adversarial environments, and provide some hints to better understanding their behaviour in such context. These results suggest a further investigation on this issue, given the relevance that MCSs have gained in the design of pattern recognition systems.

Acknowledgements This work was partly supported by a grant awarded to B. Biggio by Regione Autonoma della Sardegna, PO Sardegna FSE 2007-2013, L.R. 7/2007 “Promotion of the scientific research and technological innovation in Sardinia”.

References

1. The Apache SpamAssassin Project, http://spamassassin.apache.org/.
2. M. Barreno, B. Nelson, R. Sears, A. D. Joseph, and J. D. Tygar. Can machine learning be secure? In *ASIACCS '06: Proc. 2006 ACM Symp. on Information, computer and communications security*, pages 16–25, New York, NY, USA, 2006. ACM.
3. J. A. Benediktsson, J. Kittler, and F. Roli, editors. *Multiple Classifier Systems, 8th International Workshop (MCS 2009)*, volume 5519 of *Lecture Notes in Computer Science*. Springer, 2009.
4. B. Biggio, G. Fumera, and F. Roli. Adversarial pattern classification using multiple classifiers and randomisation. In *12th Joint IAPR Int. Workshop on Structural and Syntactic Pattern Recognition (SSPR 2008)*, volume 5342 of *LNCS*, pages 500–509, 2008. Springer-Verlag.
5. B. Biggio, G. Fumera, and F. Roli. Evade hard multiple classifier systems. In O. Okun and G. Valentini, editors, *Supervised and Unsupervised Ensemble Methods and Their Applications*, volume 245 of *Studies in Computational Intelligence*, pages 15–38. Springer Berlin / Heidelberg, 2009.
6. B. Biggio, G. Fumera, and F. Roli. Multiple classifier systems for adversarial classification tasks. In Benediktsson et al. [3], pages 132–141.
7. B. Biggio, G. Fumera, and F. Roli. Multiple classifier systems under attack. In N. E. Gayar, J. Kittler, and F. Roli, editors, *MCS*, Lecture Notes in Computer Science. Springer, 2010 (in press).
8. C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 1 edition, October 2007.

9. L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.
10. L. Breiman. Random forests. *Machine Learning*, 45:5–32, 2001.
11. P. Bühlmann and B. Yu. Analyzing bagging. *Annals of Statistics*, 30(4):927–961, 2002.
12. A. Buja and W. Stuetzle. The effect of bagging on variance, bias, and mean squared error. Tech. rep., AT&T Labs-Research, 2000.
13. A. A. Cárdenas and J. S. Baras. Evaluation of classifiers: Practical considerations for security applications. *AAAI Workshop on Evaluation Methods for Machine Learning*, Boston, MA, USA, 2006.
14. C.-C. Chang and C.-J. Lin. LibSVM: a library for support vector machines, http://www.csie.ntu.edu.tw/ cjlin/libsvm/, 2001.
15. G. V. Cormack. Trec 2007 spam track overview. In E. M. Voorhees and L. P. Buckland, editors, *TREC*, volume Special Publication 500-274. National Institute of Standards and Technology (NIST), 2007.
16. G. F. Cretu, A. Stavrou, M. E. Locasto, S. J. Stolfo, and A. D. Keromytis. Casting out demons: Sanitizing training data for anomaly sensors. *Security and Privacy, IEEE Symp. on*, 0:81–95, 2008.
17. N. Dalvi, P. Domingos, Mausam, S. Sanghai, and D. Verma. Adversarial classification. In *Tenth ACM SIGKDD Int. Con. on Knowledge Discovery and Data Mining (KDD)*, pages 99–108, Seattle, 2004.
18. P. Domingos. Why does bagging work? a bayesian account and its implications. In *Proc. 3rd Int. Conf. on Knowledge Discovery and Data Mining*, pages 155–158, 1997.
19. H. Drucker, D. Wu, and V. N. Vapnik. Support vector machines for spam categorization. *IEEE Trans. on Neural Networks*, 10(5):1048–1054, 1999.
20. P. Fogla, M. Sharif, R. Perdisci, O. Kolesnikov, and W. Lee. Polymorphic blending attacks. In *USENIX-SS'06: Proc. 15th Conf. on USENIX Security Symposium*, 2006. USENIX Association.
21. J. H. Friedman and P. Hall. On bagging and nonlinear estimation. *J. of Statistical Planning and Inference*, 137(3):669 – 683, 2007. Special Issue on Nonparametric Statistics and Related Topics: In honor of M.L. Puri.
22. J. Galbally-Herrero, J. Fierrez-Aguilar, J. D. Rodriguez-Gonzalez, F. Alonso-Fernandez, J. Ortega-Garcia, and M. Tapiador. On the vulnerability of fingerprint verification systems to fake fingerprint attacks. In *Proc. IEEE Intl. Carnahan Conf. on Security Technology, ICCST*, pages 130–136, 2006.
23. F. Gargiulo, L. I. Kuncheva, and C. Sansone. Network protocol verification by a classifier selection ensemble. In Benediktsson et al. [3], pages 314–323.
24. A. Globerson and S. T. Roweis. Nightmare at test time: robust learning by feature deletion. In W. W. Cohen and A. Moore, editors, *ICML*, volume 148 of *ACM Int. Conf. Proc. Series*, pages 353–360. ACM, 2006.
25. P. Graham. A plan for spam, http://paulgraham.com/spam.html, 2002.
26. J. Graham-Cumming. How to beat an adaptive spam filter. In *MIT Spam Conference*, Cambridge, MA, USA, 2004.
27. Y. Grandvalet. Bagging equalizes influence. *Machine Learning*, 55:251–270, 2004.
28. M. Haindl, J. Kittler, and F. Roli, editors. *Multiple Classifier Systems, 7th Int. Workshop, MCS 2007, Prague, Czech Republic, May 23-25, 2007, Proceedings*, volume 4472 of *Lecture Notes in Computer Science*. Springer, 2007.
29. S. Hershkop and S. J. Stolfo. Combining email models for false positive reduction. In *KDD '05: Proc. 11th ACM SIGKDD Int. Conf. on Knowledge Discovery in Data Mining*, pages 98–107, 2005. ACM.
30. T. K. Ho. The random subspace method for constructing decision forests. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 20(8):832–844, 1998.
31. Z. Jorgensen, Y. Zhou, and M. Inge. A multiple instance learning strategy for combating good word attacks on spam filters. *J. of Machine Learning Research*, 9:1115–1146, 2008.
32. R. A. Kemmerer and G. Vigna. Intrusion detection: A brief history and overview (supplement to Computer magazine). *Computer*, 35:27–30, 2002.
33. J. Kittler, M. Hatef, R. P. Duin, and J. Matas. On combining classifiers. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 20(3):226–239, 1998.
34. M. Kloft and P. Laskov. A 'poisoning' attack against online anomaly detection. In Laskov and Lippmann [38].
35. A. Kolcz and C. H. Teo. Feature weighting for improved classifier robustness. In *6th Conf. on Email and Anti-Spam (CEAS)*, 2009.
36. L. I. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms*. Wiley, Hoboken, N. J., 2004.
37. P. Laskov and M. Kloft. A framework for quantitative security analysis of machine learning. In *AISec '09: Proc. 2nd ACM Workshop on Security and Artificial Intelligence*, pages 1–4, 2009. ACM.
38. P. Laskov and R. Lippmann, editors. *Neural Information Processing Systems (NIPS) Workshop on Machine Learning in Adversarial Environments for Computer Security, http://mls-nips07.first.fraunhofer.de*, 2007.
39. D. D. Lewis. An evaluation of phrasal and clustered representations on a text categorization task. In *SIGIR '92: Proc. 15th annual Int. ACM SIGIR Conf. Research and development in Information Retrieval*, pag. 37–50, New York, NY, USA, 1992.
40. D. Lowd and C. Meek. Adversarial learning. In A. Press, editor, *Proc. 11th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD)*, pages 641–647, 2005.
41. D. Lowd and C. Meek. Good word attacks on statistical spam filters. In *2nd Conf. on Email and Anti-Spam (CEAS)*, 2005.
42. T. A. Meyer and B. Whateley. Spambayes: Effective open-source, bayesian based, email classification system. In *1st Conf. on Email and Anti-Spam (CEAS)*, 2004.
43. R. Perdisci, D. Dagon, W. Lee, P. Fogla, and M. Sharif. Misleading worm signature generators using deliberate noise injection. In *Security and Privacy, IEEE Symp. on*, pages 15 pp.–31, 2006.
44. R. Perdisci, G. Gu, and W. Lee. Using an ensemble of one-class svm classifiers to harden payload-based anomaly detection systems. In *Int. Conf. on Data Mining (ICDM)*, pages 488–498. IEEE Computer Society, 2006.
45. R. N. Rodrigues, L. L. Ling, and V. Govindaraju. Robustness of multimodal biometric fusion methods against spoof attacks. *J. Vis. Lang. Comput.*, 20(3):169–179, 2009.
46. A. A. Ross, K. Nandakumar, and A. K. Jain. *Handbook of Multibiometrics*. Springer, 2006.
47. D. B. Skillicorn. Adversarial knowledge discovery. *IEEE Intelligent Systems*, 24:54–61, 2009.
48. M. Skurichina and R. P. W. Duin. Bagging for linear classifiers. *Pattern Recognition*, 31:909–930, 1998.
49. M. Skurichina and R. P. W. Duin. Bagging, boosting and the random subspace method for linear classifiers. *Pattern Analysis and Applications*, 5(2):121–135, 2002.
50. H. Stern. A survey of modern spam tools. In *5th Conf. on Email and Anti-Spam (CEAS)*, 2008.
51. C. Sutton, M. Sindelar, and A. McCallum. Feature bagging: Preventing weight undertraining in structured discriminative learning. IR 402, University of Massachusetts, 2005.
52. T. Tran, P. Tsai, and T. Jan. An adjustable combination of linear regression and modified probabilistic neural network for anti-spam filtering. In *Int. Conf. on Pattern Recognition (ICPR08)*, pages 1–4, 2008.
53. U. Uludag and A. K. Jain. Attacks on biometric systems: A case study in fingerprints. In *Proc. SPIE-EI 2004, Security, Steganography and Watermarking of Multimedia Contents VI*, pages 622–633, 2004.