Bagging Classifiers for Fighting Poisoning Attacks in Adversarial Classification Tasks

Battista Biggio, Igino Corona, Giorgio Fumera,
Giorgio Giacinto, and Fabio Roli

Dept. of Electrical and Electronic Engineering, University of Cagliari
Piazza d'Armi, 09123 Cagliari, Italy
{battista.biggio,igino.corona,fumera,giacinto,roli}@diee.unica.it
http://prag.diee.unica.it

Abstract. Pattern recognition systems have been widely used in *adversarial classification* tasks like spam filtering and intrusion detection in computer networks. In these applications a malicious adversary may successfully mislead a classifier by "poisoning" its training data with carefully designed attacks. Bagging is a well-known ensemble construction method, where each classifier in the ensemble is trained on a different bootstrap replicate of the training set. Recent work has shown that bagging can reduce the influence of outliers in training data, especially if the most outlying observations are resampled with a lower probability. In this work we argue that poisoning attacks can be viewed as a particular category of outliers, and, thus, bagging ensembles may be effectively exploited against them. We experimentally assess the effectiveness of bagging on a real, widely used spam filter, and on a web-based intrusion detection system. Our preliminary results suggest that bagging ensembles can be a very promising defence strategy against poisoning attacks, and give us valuable insights for future research work.

1 Introduction

Security applications like spam filtering, intrusion detection in computer networks, and biometric authentication have been typically faced as two-class classification problems, in which the goal of a classifier is to discriminate between "malicious" and "legitimate" samples, e.g., spam and legitimate emails. However, these tasks are quite different from traditional classification problems, as intelligent, malicious, and adaptive adversaries can manipulate their samples to mislead a classifier or a learning algorithm. In particular, one of the main issues to be faced in a so-called *adversarial classification* task [6] is the design of a *robust* classifier, namely, a classifier whose performance degrades as gracefully as possible under attack [6,12]. Adversarial classification is attracting a growing interest from the pattern recognition and machine learning communities, as witnessed by a recent workshop held in the context of the NIPS 2007 conference [1], and by a special issue of *Machine Learning* [13] entirely dedicated to this topic.

[1] `http://mls-nips07.first.fraunhofer.de`

In this work we consider a specific class of attacks named *causative attacks* in [2,1], and *poisoning* attacks in [11], in which the adversary is assumed to control a subset of samples that will be used to train or update the classifier, and he carefully designs these samples to mislead the learning algorithm. For instance, in intrusion detection skilled adversaries may inject poisoning patterns to mislead the learning algorithm which infers the profile of legitimate activities [11] or intrusions [15,4]; while in spam filtering adversaries may modify spam emails by adding a number of "good words", i.e., words which are likely to appear in legitimate emails and not in spam, so that when a user reports them as spam to the filter, it becomes more prone to misclassify legitimate emails as spam [14].

We argue that the problem of designing robust classifiers against poisoning attacks can be formulated as a problem in which one aims to reduce the influence of outlier samples in training data. The main motivation is that the adversary aims to "deviate" the classification algorithm from learning a correct model or probability distribution of training data, and typically can control only a small percentage of training samples. If this were not true, namely, poisoning samples were similar to other samples within the same class (or even to novel samples which represent the normal evolution of the system), their effect would be negligible. Since bagging, and in particular weighted bagging [19], can effectively reduce the influence of outlying observations in training data [9], in this work we experimentally investigate whether bagging ensembles can be exploited to build robust classifiers against poisoning attacks. We consider two relevant application scenarios: a widely deployed text classifier in spam filtering [16], and a basic version of HMM-Web, an Intrusion Detection System (IDS) for web applications [5].

The paper is structured as follows: in Sect. 2 we review related works, in Sect. 3 we highlight the motivations of this work, in Sect. 4 we describe the problem formulation and the considered applications, in Sect. 5 we present our experiments, and, eventually, in Sect. 6 we draw conclusions and sketch possible future work.

2 Background

The aim of this section is twofold. We first discuss works which investigated the effectiveness of bagging ensembles in the presence of outliers (Sect. 2.1). Then, we shortly review works related to poisoning attacks (Sect. 2.2).

2.1 Bagging in the Presence of Outliers

Bagging, short for *bootstrap aggregating*, was originally proposed in [3] to improve the classification accuracy over an individual classifier, or the approximation error in regression problems. The underlying idea is to perturb the training data by creating a number of bootstrap replicates of the training set, train a classifier on each bootstrap replicate, and aggregate their predictions. This allows to reduce the variance component of the classification or estimation error (in regression) (e.g., [3,7]). Indeed, bagging has shown to be particularly successful when applied to "unstable" classifiers (i.e., classifiers whose predictions vary significantly when small variations in training data occur) like decision trees and neural networks. Other explanations to the effectiveness of bagging were also proposed; in particular, in [9] it was argued that bagging equalizes the influence of training samples, namely, it reduces the influence of outlier samples in training data. This was also experimentally verified on a simple task in [10], and exploited in [8] to develop outlier resistant PCA ensembles.

To further reduce the influence of the most outlying observations in training data, *weighted* bagging was proposed in [19,18]. The rationale behind this approach is to resample the training set by assigning a probability distribution over training samples, in particular, lower probability weights to the most outlying observations. The method can be summarised as follows. Given a training set $T_n = \{\mathbf{x}_i, y_i\}_{i=1}^n$, and a set of probability weights $w_1, \ldots, w_n$, for which it holds $\sum_{i=1}^n w_i = 1$:

1. create m bootstrap replicates of T_n by sampling $(\mathbf{x}_i, y_i)$ with probability w_i, $i = 1, \ldots, n$;
2. train a set of m classifiers, one on each bootstrap replicate of T_n;
3. combine their predictions, e.g., by majority voting, or averaging.

Note that this corresponds to the standard bagging algorithm [3] when $w_i = 1/n$, $i = 1, \ldots, n$, and the majority voting is used as combining rule.

The set of weights $w_1, \ldots, w_n$ was estimated in [19,18] using a kernel density estimator. Since kernel density estimation can be unreliable in highly dimensional feature spaces, the authors exploited a *boosted* kernel density estimate, given by

$$f(\mathbf{x}_i) = \sum_{j=1}^{n} \frac{w_j}{(2\pi)^{d/2}\sigma^d} k(\mathbf{x}_i, \mathbf{x}_j)\,, \tag{1}$$

where $k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\gamma ||\mathbf{x}_i - \mathbf{x}_j||^2\right)$ is a Gaussian kernel, and the set of weights $w_1, \ldots, w_n$ is iteratively estimated as follows. Initially, all samples are equally weighted, i.e., $w_i = 1/n,\ i = 1, \ldots, n$. Each weight is then iteratively updated according to $w_i^{(k+1)} = w_i^{(k)} + \log\left(f^{(k)}(\mathbf{x}_i)/g^{(k)}(\mathbf{x}_i)\right)$, where k represents current iteration, and $g(\mathbf{x}_i)$ is the "leave-one-out" estimate of $f(\mathbf{x}_i)$, given by

$$g(\mathbf{x}_i) = \sum_{j=1}^{n} \frac{w_j}{(2\pi)^{d/2}\sigma^d} k(\mathbf{x}_i, \mathbf{x}_j) I(j \neq i)\,, \tag{2}$$

where $I(j \neq i)$ equals 0 (1) only when $j = i$ $(j \neq i)$. Once convergence or a maximum number of iterations is reached, the final weights are inverted and normalized as

$$w_i = \frac{1}{w_i^{(k)}} \Big/ \sum_{j=1}^{n} \frac{1}{w_j^{(k)}}\,, \tag{3}$$

so that weights assigned to outlying observations exhibit lower values.

2.2 Works on Poisoning Attacks

Poisoning attacks were investigated in the field of adversarial classification, mainly considering specific applications. According to a taxonomy of potential attacks against machine learning algorithms proposed in [2,1], they can be more generally referred to as *causative* attacks, and can be exploited either to increase the false positive rate (i.e., the percentage of misclassified legitimate samples) or the false negative rate at operation phase. They were thus further categorised as *availability* or *integrity* attacks.

Poisoning attacks were devised against spam filters [14] (based on adding "good words" to spam emails, as described in Sect. 1) and simple online IDSs [2,11]. Instead, in [17] a countermeasure against them was proposed; in particular, the framework of Robust Statistics was exploited to reduce the influence of poisoning attacks in training data (which were implicitly considered "outliers").

3 Motivations of This Work

The aim of this section is to further clarify the scope of this work. As mentioned in Sect. 1, we argue that poisoning attacks can be regarded as outlying observations with respect to other samples in training data. The reason for that is twofold:

1. since the goal of a poisoning attack is to "deviate" the classification algorithm from learning a correct model or probability distribution of one of the two classes (or both), poisoning attack samples have to be different from other samples within the same class;
2. since the adversary is likely to control only a small percentage of training data in real applications, each poisoning sample should be able to largely deviate the learning process.

In addition, we also point out that several defence strategies implicitly deal with poisoning attacks as they were outliers, e.g., [17]. Besides this, as discussed in Sect. 2, a number of works highlighted that bagging (and in particular weighted bagging) can reduce the influence of outliers in training data. Thus, in this work we experimentally investigate whether bagging and weighted bagging can be successfully exploited to fight poisoning attacks in two different adversarial classification tasks, namely, spam filtering and intrusion detection. We also point out that comparing bagging and weighted bagging with the defence strategies mentioned in Sect. 2 is out of the scope of this work, as we are only considering a preliminary investigation.

4 Problem Formulation and Application Scenarios

In this section we briefly describe the problem formulation related to the two case studies considered in this work, namely, spam filtering and web-based intrusion detection. For the spam filtering task, we considered a text classifier proposed in [16], which is currently adopted in several spam filters, including SpamAssassin (`http://spamassassin.apache.org`), SpamBayes ((`http://spambayes.sourceforge.net`), and BogoFilter (`http://bogofilter.sourceforge.net`). It estimates the probability for an email to be spam mainly based on its textual content. We tested the effectiveness of this classifier, as well as that of bagging ensembles, against a poisoning attack proposed in [14], aimed at generating a higher false positive rate at operation phase (i.e., a causative availability attack [1]). The rationale is to modify spam emails by adding "good words" without making them appear as legitimate emails (e.g., using white text on a white background [2]), so that users still report them as spam to the filter, increasing the probability for legitimate emails including those words to be classified as spam.

For the intrusion detection task, we focused on *web applications*. Web applications are largely employed in simple websites, and in security-critical environments like medical, financial, military and administrative systems. A web application is a software program which generates informative content in real time, e.g., an HTML page, based on user inputs (queries). Cyber-criminals may divert the expected behaviour of a web application by submitting malicious queries, either to access confidential information, or to cause a denial of service (DoS). In our experiments we considered a simplified version of HMM-Web, a state-of-the-art IDS for web applications based on Hidden Markov Models (HMMs) [5]. We devised a poisoning attack against this classifier, aimed at allowing more intrusions at operation phase (i.e., a causative integrity attack [1]), as inspired by [2,11]. To this aim, we generate attack queries with (1) a different structure with respect to legitimate queries, and (2) portions of structures similar to intrusive sequences. This attack turned out to be very effective in practice, as it degrades the classifier's performance when very few poisoning attacks are injected into the training set. On the contrary, we noted that the same classifier was very robust to the injection of random sequences or of the intrusive ones.

5 Experiments

We start describing the experimental setup for the spam filtering and web-based intrusion detection tasks. In both experiments performance was evaluated as proposed in [12]. In particular, we computed the area under the ROC curve (AUC) in the region corresponding to FP rates in $[0, \mathrm{K}]$: $\mathrm{AUC_K} = 1/\mathrm{K} \int_0^{\mathrm{K}} \mathrm{TP(FP)dFP} \in [0, 1]$, where K denotes the maximum allowed FP rate. This measure may be considered as more informative than the AUC since it focuses on the performance around practical operating points (corresponding to low FP rates).

Spam filtering. Our experiments in spam filtering were carried out on the publicly available TREC 2007 email corpus [3], which is made up of 25,220 legitimate and 50,199 spam emails. The first 10,000 emails (in chronological order) were used as training set, while the remaining 65,419 were used as testing set.

[2] `http://www.virusbtn.com/resources/spammerscompendium/index`

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

The SpamAssassin filter was used to extract a set of distinct tokens from training emails, which turned out to be 366,709. To keep a low computational complexity, we selected a subset of 20,000 tokens (with the information gain criterion), and used them as feature set. In particular, each email was represented by a Boolean feature vector, whose values denoted either the absence (0) or presence (1) of the corresponding tokens in the given email. We exploited the text classifier proposed in [16] to build bagging and weighted bagging ensembles. We considered 3, 5, 10, 20, and 50 as the number of base classifiers, and the simple average to combine their outputs. For weighted bagging, we set the γ parameter of the Gaussian kernel as the inverse of the number of features (i.e., $0.5E^{-4}$), since the latter corresponds to the maximum value of the distance between two samples. Moreover, we performed a further experiment by varying $\gamma \in \{1E^{-3}, 1E^{-5}, 1E^{-6}\}$ besides the default value, to study the effect of this parameter on the robustness of weighted bagging. To keep the kernel density estimation computationally negligible, we computed an estimate of $f(\mathbf{x})$ and $g(\mathbf{x})$ (see Sect. 2.1) by considering a randomly chosen subset of 50 training spam emails (instead of the whole set). We point out that this did not significantly affect the estimation of the probability weights. Poisoning attack samples were created as follows. First, a set of spam emails $\mathcal{S}$ was randomly sampled (with replacement) from the training set; then, a number of randomly chosen "good words" (chosen among an available set of good words) was added to each spam in $\mathcal{S}$; and, finally, all spam emails in $\mathcal{S}$ were added to the training set. As in [14], we investigated the worst case scenario in which the adversary is assumed to know the whole set of "good words" used by the classifier (which includes all tokens extracted from legitimate emails). In our experiments we noted that the adversary is required to add up to about $5,000$ randomly chosen "good words" to each spam to make the classifier completely useless at 20% poisoning (i.e., using 2,500 poisoning emails). We thus evaluated the performance of the considered classifiers by varying the fraction of poisoning attacks in $[0, 0.2]$ with steps of 2%.

Web-based intrusion detection. We experimented with a dataset which reflected real traffic on a production web server employed by our academic institution. We collected 69,001 queries towards the principal web application, in a time interval of 8 months. We detected 296 intrusive attempts among them. The first 10,000 legitimate queries (in chronological order) were used as training set, while the remaining 58,705 legitimate queries and the intrusive queries were used as test set. Each web application query q has the form $a_1 = v_1 \& a_2 = v_2 \& \ldots \& a_n = v_n$, where a_i is the i-th attribute, v_i is its corresponding value, and n is the number of attributes of q. We encoded each query as the sequence of attributes and their values. [4] The HMM was trained using the Baum-Welch algorithm, to exploit the underlying structure of legitimate sequences, and consequently detect intrusions by assigning them a lower likelihood. To build a simple and effective model, we initialized the HMM with two states: one associated to the emission of symbols in even positions, and the other

[4] The whole data set is available at `http://prag.diee.unica.it/pra/system/files/dataset_hmm_mcs2011.zip`

associated to the emission of symbols in odd positions. The emission probability of each symbol was initialized as its relative frequency in even or odd positions, depending on the state. The state transition matrix was randomly initialized. Similarly to spam filtering, we carried out experiments also considering bagging and weighted bagging, with 3, 5, 10, 20 HMMs per ensemble, and the simple average as combining rule. In order to apply the kernel density estimator used in weighted bagging (remind that we deal with sequences of non-fixed length), we first extracted all possible bigrams (i.e. contiguous subsequences of length two) from legitimate and intrusive sequences. Then, we represented each sequence as a Boolean feature vector, in which each value denotes either the absence (0) or presence (1) of the corresponding bigram in the given sequence. The length of each feature vector (total number of bigrams) turned out to be $N = 205$. As in spam filtering, the default value of γ has been computed as the inverse of the cardinality of the feature space, i.e., $1/N \approx 5E^{-3}$, and $f(\mathbf{x})$ and $g(\mathbf{x})$ were estimated using a subset of 50 training samples. The sensitivity of weighted bagging to γ was further studied by varying $\gamma \in \{5E^{-4}, 2.5E^{-3}\}$. The poisoning attacks against the HMM were built exploiting the rationale described in the Sect. 4. In particular, poisoning sequences contained only bigrams which were not present in legitimate sequences, but which might have been present in intrusive sequences. As this attack turned out to be very effective, we evaluated the performance of the considered classifiers by varying the fraction of poisoning attacks in $[0, 0.02]$ with steps of 0.2%.

5.1 Experimental Results

In this section we report the results for spam filtering (Fig. 1) and web-based intrusion detection (Fig. 2). We assessed performance using the $AUC_{10\%}$ measure in the spam filtering task (as in [12]), and $AUC_{1\%}$ in the intrusion detection task (since FP rates higher than 1% are unacceptable in this application). Results are averaged over 5 repetitions, as poisoning samples were randomly generated. We do not report standard deviation values as they turned out to be negligible.

First, note that AUC values decreased for increasing percentage of poisoning, as expected. When no poisoning attack is performed (0%), all classifiers behaved similarly, and, in particular, bagging and weighted bagging only slightly outperformed the corresponding single classifiers. Under attack, instead, bagging and weighted bagging significantly outperformed the single classifiers. In particular, the performance improvement was marked when the injected amount of poisoning attacks significantly affected the single classifier's performance (see, for instance, 8-10% of poisoning for spam classifiers).

Increasing the ensemble size of bagging classifiers turned out to significantly improve the performance under attack only in the spam filtering task (Fig. 1, left). The underlying reason could be that bagging can effectively drop the variance of the classification error by increasing the ensemble size (as mentioned in Sect. 2.1, and shown in [7]); thus, increasing the ensemble size may be effective only when poisoning attacks introduce a substantial variance in the classification

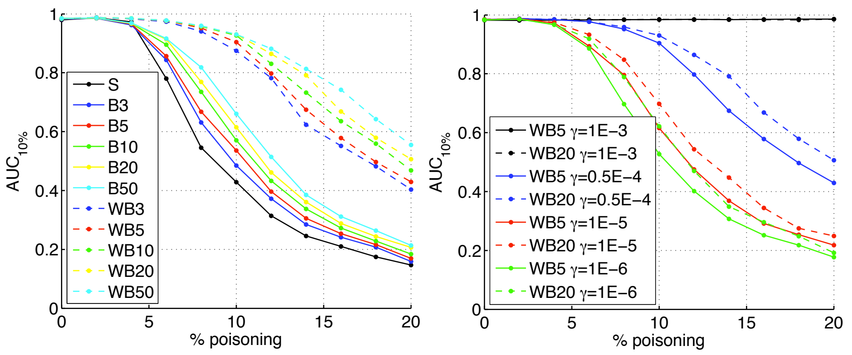

Fig. 1. *Left:* performance of the spam classifier (S), bagging (B), and weighted bagging with default γ (WB) against percentage of poisoning attacks in training data, for different ensemble sizes (3,5,10,20,50). *Right:* performance of WB with ensemble sizes of 5 and 20 against percentage of poisoning attacks in training data, for different γ.

error (whereas this may be not true when the error is highly biased). This aspect can be a promising research direction to investigate.

We focus now on weighted bagging, which significantly improved the performance over standard bagging in both experiments, as expected. This is clearly due to the use of a kernel density estimator, which basically imputes outliers in training data, and reduces their influence. To investigate the effectiveness of weighted bagging more in depth, we considered different values of the γ parameter, as explained in the previous section. The rationale was to alter the performance of the kernel density estimator. From Fig. 1 (right) one can immediately note that a value of $\gamma = 10^{-3}$ in the spam filtering task allowed weighted bagging to completely *remove* poisoning attacks from training data (the performance did not decrease). On the other hand, for $\gamma = 10^{-6}$ performance was very similar to that of standard bagging. Similar results were obtained in the intrusion detection task. Fig. 2 (left) shows that the higher γ, the more gracefully the performance of weighted bagging decreased. It is worth noting that weighted bagging can worsen performance with respect to standard bagging, even in absence of poisoning, if the weights assigned by the kernel density estimator to samples in the same class exhibit a large variance. The reason is that this leads to obtain a set of bootstrap replicates of the training set which do not reflect the correct probability distribution of training samples.

To sum up, standard bagging can provide a significant improvement in performance over an individual classifier, in particular against some kinds of poisoning attacks. The effectiveness of weighted bagging is strictly related to the capability of estimating a reliable set of weights, namely, on the capability of the kernel density estimator to correctly impute the outlying observations. However, when this happens (as in our experiments) weighted bagging can provide a great performance improvement. Besides this, when using a good kernel density estimator the adversary is required to spend more "effort" to build a poisoning

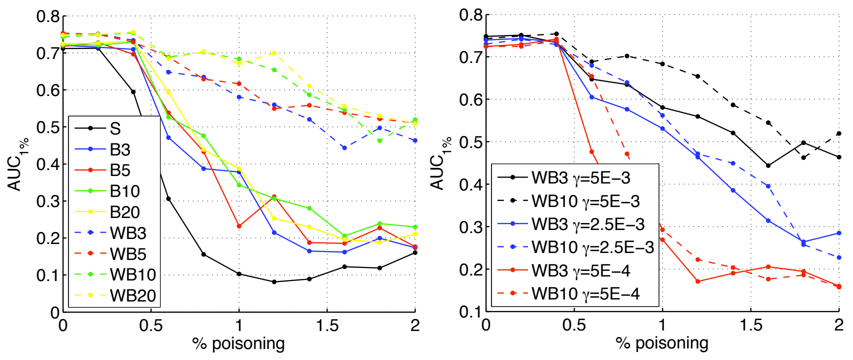

Fig. 2. *Left:* performance of the HMM-web classifier (S), bagging (B), and weighted bagging with default γ (WB) against percentage of poisoning attacks in training data, for different ensemble sizes (3,5,10,20). *Right:* performance of WB with ensemble sizes of 3 and 10 against percentage of poisoning attacks in training data, for different γ.

attack which misleads weighted bagging. For instance, in the case of spam filtering the adversary would be required to add only a few "good words" to each spam email, so that poisoning emails are not easily distinguishable from others. Consequently, he would be required to control a much larger percentage of training data, which may be not feasible in practice.

6 Conclusions and Future Work

In adversarial environments, like spam filtering and intrusion detection in computer networks, classifiers must not only be accurate, but also robust to poisoning attacks, i.e., to the deliberate injection of malicious noise in the training set. In this preliminary study we experimentally showed, for two relevant applications, that bagging ensembles may be a *general*, effective technique to address the problem of poisoning attacks, regardless the base classification algorithm. These results give us valuable insights for future research work. First, we plan to theoretically investigate the effectiveness of bagging against poisoning attacks. Second, we aim to study more general methods for better estimating the set of resampling weights. Lastly, we want to investigate what categories of poisoning attacks can be effectively tackled by increasing the ensemble size (as this emerged as an open problem from our experiments).

Acknowledgements. The authors would like to thank the anonymous reviewers for useful insights and suggestions. This work was partly supported by a grant from Regione Autonoma della Sardegna awarded to B. Biggio and I. Corona, PO Sardegna FSE 2007-2013, L.R.7/2007 "Promotion of the scientific research and technological innovation in Sardinia".

References

1. Barreno, M., Nelson, B., Joseph, A., Tygar, J.: The security of machine learning. Machine Learning 81, 121–148 (2010)
2. 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 Comm. Sec. (ASIACCS 2006), NY, USA pp. 16–25 (2006)
3. Breiman, L.: Bagging predictors. Machine Learning 24(2), 123–140 (1996)
4. Chung, S.P., Mok, A.K.: Advanced allergy attacks: Does a corpus really help? In: Kruegel, C., Lippmann, R., Clark, A. (eds.) RAID 2007. LNCS, vol. 4637, pp. 236–255. Springer, Heidelberg (2007)
5. Corona, I., Ariu, D., Giacinto, G.: Hmm-web: a framework for the detection of attacks against web applications. In: Proc. 2009 IEEE Int'l Conf. Comm. (ICC 2009), NJ, USA, pp. 747–752 (2009)
6. Dalvi, N., Domingos, P.: Mausam, S. Sanghai, and D. Verma. Adversarial classification. In: Proc. 10th ACM SIGKDD Int'l Conf. Knowledge Disc. and Data Mining (KDD), USA, pp. 99–108 (2004)
7. Fumera, G., Roli, F., Serrau, A.: A theoretical analysis of bagging as a linear combination of classifiers. IEEE TPAMI 30(7), 1293–1299 (2008)
8. Gabrys, B., Baruque, B., Corchado, E.: Outlier resistant PCA ensembles. In: Gabrys, B., Howlett, R.J., Jain, L.C. (eds.) KES 2006. LNCS (LNAI), vol. 4253, pp. 432–440. Springer, Heidelberg (2006)
9. Grandvalet, Y.: Bagging equalizes influence. Machine Learning 55, 251–270 (2004)
10. Hall, P., Turlach, B.: Bagging in the presence of outliers. In: Scott, D. (ed.) Mining and Modeling Massive Data Sets In Science, Engineering, and Business, CSS, vol. 29, pp. 536–539 (1998)
11. Kloft, M., Laskov, P.: Online anomaly detection under adversarial impact. In: Proc. 13th Int'l Conf. Artificial Intell. and Statistics (AISTATS), pp. 405–412 (2010)
12. Kolcz, A., Teo, C.H.: Feature weighting for improved classifier robustness. In: 6th Conf. Email and Anti-Spam (CEAS), CA, USA (2009)
13. Laskov, P., Lippmann, R.: Machine learning in adversarial environments. Machine Learning 81, 115–119 (2010)
14. Nelson, B., Barreno, M., Chi, F.J., Joseph, A.D., Rubinstein, B.I.P., Saini, U., Sutton, C., Tygar, J.D., Xia, K.: Exploiting machine learning to subvert your spam filter. In: Proc. 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats (LEET 2008), CA, USA, pp. 1–9 (2008)
15. Robinson, G.: A statistical approach to the spam problem. Linux J (2001), `http://www.linuxjournal.com/article/6467`
16. Perdisci, R., Dagon, D., Lee, W., Fogla, P., Sharif, M.: Misleading worm signature generators using deliberate noise injection. In: Proc. 2006 IEEE Symp. Sec. and Privacy (S&P 2006), USA (2006)
17. Rubinstein, B.I., Nelson, B., Huang, L., Joseph, A.D.: Antidote: understanding and defending against poisoning of anomaly detectors. In: Proc. 9th ACM Internet Meas. Conf. IMC 2009, pp. 1–14 (2009)
18. Segui, S., Igual, L., Vitria, J.: Weighted bagging for graph based one-class classifiers. In: Proc. 9th Int. Workshop on MCSs. LNCS, vol. 5997, pp. 1–10. Springer, Heidelberg (2010)
19. Shieh, A.D., Kamm, D.F.: Ensembles of one class support vector machines. In: Benediktsson, J.A., Kittler, J., Roli, F. (eds.) MCS 2009. LNCS, vol. 5519, pp. 181–190. Springer, Heidelberg (2009)