Anomaly Detection by Bagging

Tomáš Pevný

Tomas.Pevny@agents.fel.cvut.cz
Agent Technology Center, Department of Computers,
Czech Technical University in Prague

Abstract. Many contemporary domains, e.g. network intrusion detection, fraud detection, etc., call for an anomaly detector processing a continuous stream of data. This need is driven by the high rate of their acquisition, limited resources for storing them, or privacy issues. The data can be also non-stationary requiring the detector to continuously adapt to their changes. A good detector for these domains should therefore have a low training and classification complexity, on-line training algorithm, and, of course, a good detection accuracy.
This paper proposes a detector trying to meet all these criteria. The detector consists of multiple weak detectors, each implemented as a one-dimensional histogram. The one-dimensional histogram was chosen because it can be efficiently created on-line, and probability estimates can be efficiently retrieved from it. This construction gives the detector linear complexity of training with respect to the input dimension, number of samples, and number of weak detectors. Similarly, the classification complexity is linear with respect to number of weak detectors and the input dimension.
The accuracy of the detector is compared to seven anomaly detectors from the prior art on the range of 36 classification problems from UCI database. Results show that despite detector's simplicity, its accuracy is competitive to that of more complex detectors with a substantially higher computational complexity.

Keywords: anomaly detection, on-line learning, ensemble methods, large data

Abstract.

1 Introduction

The goal of an anomaly detector is to find samples, which in some sense deviate from the majority. It finds application in many important fields, such as network intrusion detection, fraud detection, monitoring of health, environmental and industrial processes, data-mining, etc. These domains frequently need a detector with low complexity of training and classification, which can efficiently process large number of samples, ideally in real-time. These requirements implicate that the detector should be trained on-line, which is also important for domains,

where the data cannot be stored, or the data are non-stationary and the detector needs to be updated continously.

This paper describes a detector (further called ADBag), which has a provably linear complexity of training with respect to the number of training samples n, and their dimension d. The classification has a low complexity which scales linearly with dimension d. The detector consists of an ensemble of k weak detectors, each implemented as a one-dimensional histogram with b bins. The one-dimensional histogram was chosen because it can be created online in a single pass over the data, and probability estimates can be efficiently retrieved from it. Since each weak detector process only one-dimensional data, the input space $\mathbb{R}^d$ is reduced to $\mathbb{R}$ by a projection on a randomly generated vector w. Projection vectors w create the diversity among weak detectors, which is important for the success of the ensemble approach. As will be explained in more detail in Section 3, ADBag can be related to a naïve Parzen window estimator [20], as each weak detector provides an estimate of the probability of a sample. According to [26], Parzen window estimator gives frequently better results than more complex classifiers[1].

ADBag's accuracy is experimentally compared to the selected prior art algorithms on 36 problems downloaded from the UCI database [6] listed in the classification problems with numerical attributes. Although there is no single dominating algorithm, ADBag's performance is competitive to others, but with a significantly smaller computational and storage requirements. On a dataset with millions of features and samples it is demonstrated that ADBag can efficiently handle large-scale data.

The paper is organized as follows. The next section briefly reviews relevant prior art, shows the computational complexity, and discusses issues related to the on-line learning and classification. ADBag is presented in Section 3. In Section 4, it is experimentally compared to the prior art and its efficiency is demonstrated on a large-scale dataset [17]. Finally, Section 5 concludes the paper.

2 Related work

A survey on anomaly and outlier detection [3] contains plenty of methods for anomaly and outlier detection. Below, those relevant to us or otherwise important are reviewed. We remark that ADBag falls into the category of model-based detectors, since every weak detector essentially creates a very simple model.

2.1 Model-based detectors

Basic model-based anomaly detectors assume the data follow a known distribution. For example principal component transformation based detector [25] with complexity of training of $O(nd^3)$ assumes a multi-variate normal distribution.

[1] Parzen window estimator is not suitable for real-time detection, since the complexity of obtaining an estimate of pdf depends linearly on the number of observed samples n.

Although multi-variate normal distribution rarely fits the real data, as will be seen later, this detector frequently provides good results.

On the contrary, One-Class Support Vector Machine [23] (OC-SVM) does not assume anything about the distribution of the data. It finds the smallest area, where $1 - \nu$ fraction of data are located (ν is parameter of the method specifying desired false-positive rate). This is achieved by projecting the data to high-dimensional (feature) space, and then finding hyperplane best separating the data from the origin. It has been noted that when OC-SVM is used with a linear kernel, it introduces bias to the origin [26]. This problem is removed by using Gaussian kernel. Support Vector Data Description algorithm [26] (SVDD) removes the bias in OC-SVM by replacing the separation hyperplane by a sphere encapsulating most of the data. It has been showed that if OC-SVM and SVDD are used with a Gaussian kernel, they are equivalent [23]. Due to this fact, SVDD is omitted in the comparison section. The complexity of the training of both methods is super-linear with respect to the number of samples, n, being $O(n^3 d)$ in the worst-case.

Recently proposed FRAC [19] aimed to bridge the gap between supervised and unsupervised learning. FRAC is an ensemble of models, each estimating one feature on basis of others (for data of a dimension d, FRAC uses d different models). The rationale behind this is that anomalous samples exhibit different dependencies among features, which can be detected from prediction errors modeled by histograms. FRAC's complexity depends on the algorithm used to implement models, which can be large, considering that a search for a possible hyper-parameters needs to be done. Because of this, an ordinary linear least-square regression is used leading to the complexity $O(nd^4)$. It is stated that FRAC is well suited for an on-line setting, but it might not be straightforward. For real-valued features, every update changes all models together with distribution of their errors. Consequently, to update the histograms of their errors, all previously observed samples are required. This means that unless some simplifying assumptions are accepted, the method cannot be used in an online settings.

Generally, the complexity of the classification of model-based detectors is negligible in comparison to the complexity of training. Yet for some methods this might be difficult to control. An example is OC-SVM, where the classification complexity depends on the number of support vectors, which is a linear function of the number of training samples n.

The on-line training of all above detectors is generally difficult. Since there is no closed-form solution for the on-line PCA, the on-line version of the PCA detector does not exist either. The on-line adaptation of OC-SVM is discussed in [11], but the solution is an approximation of the solution returned by the batch version. The exact on-line version of SVDD is described in [27], but the algorithm requires substantial bookkeeping, which degrades its usability in real-time applications. Moreover, the bookkeeping increases the storage requirements, which are not bounded anymore.

2.2 Distance-based detectors

Distance based detectors use all available data as a model. All data are usually presented in a single batch and the outliers are found within. Thus, it is assumed that the majority of samples comes from one (nominal) class. The lack of training phase makes the adaptation to on-line settings easy — new samples are just added to the set of already observed samples. Notice that this increases the complexity of the classification phase, which is a linear function of the number of samples n.

A k-nearest neighbor [12] (KNN) is a popular method to identify outliers inspired by the corresponding method from classification. It ranks samples according to their distance to k^{th}- nearest neighbor. KNN has been criticized for not being able to detect outliers in data with clusters of different density [2]. The local outlier factor [2] (LOF) solves this problem by defining the outlier score as a fraction of sample's distance to its k^{th}-nearest neighbor and the average of the same distance of all its k nearest neighbors. True inliers have score around one, while outliers have score much greater. The prior art here is vast and it is impossible to list all, hence we refer to [29] for more.

The complexity of the classification phase of nearest-neighbor based detectors is driven by the nearest-neighbor search, which is with a naïve implementation $O(n)$ operation. To alleviate, more efficient approaches have been adopted based on bookkeeping [22], better search structures like KD-trees, or approximate search. Nevertheless, the complexity of all methods depends in some way on the number of training samples n.

2.3 Ensembles in outlier detection

The ensemble approach has been so far little utilized in the anomaly detection. A significant portion of the prior art focuses on a unification of scores [8,24], which is needed for diversification by using different algorithms [18]. A diversification by a random selection of sub-spaces has been utilized in [14].

ADBag's random projection can be considered as a modification of the sub-space method. Yet, the important difference is that the random projection relates all features together and not just some of them, as the sub-space method does. Also, all previous work use heavy-weighted detectors, while ADBag uses very simple detector, which gives it its low complexity.

2.4 Random projections

Random projections have been utilized mainly in distance-based outlier detection schemes to speedup the search. De Vries et al. [5] uses the property that random projections approximately preserve L_2 distances among set of points [10]. Thus instead of performing the k^{th}-NN search in a high-dimensional space in the LOF, the search is conducted in the space of reduced dimension but on a larger neighborhood, which is then refined by the search in the original dimension.

Similarly, Pham et al. [21] use random projections to estimate distribution of angles between samples, which has been proposed in [13] as a good anomaly criterium.

Unlike both above schemes, ADBag is a model-based detector that does not rely on the notion of distance or angles, but on the notion of probability density. ADBag brings random projections to the extreme, by projecting the input space into a single dimension, which extremely simplifies the complexity of all operations over it.

3 Algorithm description

ADBag is an ensemble of equal, weak detectors. Every weak detector within the ensemble is a histogram in one-dimensional space $\mathbb{R}$, which is created by projection the input space $\mathbb{R}^d$ to a vector w. Projection vectors are generated randomly during the initialization of a weak histograms before any data have been observed. Let $h_i(x) = \hat{p}_i(x^{\mathrm{T}} w_i)$ denotes the output of i^{th} weak detector ($\hat{p}_i$ is empirical probability distribution function (pdf) of data projected on w_i). ADBag's output is an average of negative logarithms of output of all weak detectors. Specifically,

$$f(x) = -\frac{1}{k} \sum_{i=1}^{k} \log h_i(x). \tag{1}$$

The rest of this section describes ADBag in detail, and it also explains the design choices. Subsection 3.1 describes strategies to generate projection vectors w_i. Following Subsection 3.2 points to issues related to on-line creation of histograms and presents a method adopted from the prior art. Finally, Subsection 3.3 explains the rationale behind the aggregation function and ADBag itself. The section finishes with a paragraph discussing ADBag's hyper-parameters and their effect on the computational complexity.

3.1 Projections

Projection vectors are generated randomly at the initialization of weak detectors. In all experiments presented in Section 4, their elements were generated according to the normal distribution with zero mean and unit variance, which was chosen due to its use in the proof of Johnson-Lindenstrauss (JL) lemma [10] (JL lemma shows that L_2 distances between points in the *projected space* approximates the same quantity in the *input space*). Other probabilities to generate random vectors w certainly exists. According to [15] it is possible to use sparse vector w, which is interesting, as it would allow the detector to elegantly deal with missing variables.

3.2 Histogram

Recall that one of the most important requirements was that the detector should operate (learn and classify) over data-streams, which means that the usual ap-

Algorithm 1: Algorithm constructing approaximation of the probability distribution of the data $\{x_1, \ldots, x_n\}$ projected on the vector w.

initialize $\mathbf{H} = \{\}$, $z_{\min} = +\infty$, $z_{\max} = -\infty$, and $w \sim \mathcal{N}(0, \mathbf{1}^d)$;

for $j \leftarrow 1$ **to** n **do**

  $z = x_j^{\mathrm{T}} w$;

  $z_{\min} = \min\{z_{\min}, z\}$;

  $z_{\max} = \min\{z_{\max}, z\}$;

  if $\exists(z_i == z)$ **then**

    $m_i = m_i + 1$;

    continue

  else

    $\mathbf{H} = \mathbf{H} \cup \{z, 1\}$

  end

  if $|\mathbf{H}| > b$ **then**

    Sort pairs in $\mathbf{H}$ such that $z_1 < z_2 < \ldots < z_{b+1}$;

    Find i minimizing $z_{i+1} - z_i$;

    Replace pairs (z_i, m_i), (z_{i+1}, m_{i+1}), by the pair

$$\left(\frac{z_i m_i + z_{i+1} m_{i+1}}{m_i + m_{i+1}}, m_i + m_{i+1} \right)$$

end

$\mathbf{H} = \mathbf{H} \cup \{(z_{\min}, 0), (z_{\max}, 0)\}$;

Sort pairs in $\mathbf{H}$ such that $z_{\min} < z_1 < z_2 < \ldots < z_{\max}$;

proaches, such as equal-area, or equal-width histograms cannot be used. The former requires the data to be available in a single batch, while the latter requires the bounds of the data to be known in beforehand. To avoid these limitations and have a bound resources, an algorithm proposed in [1] is adopted, despite it does not guarantee the convergence to the true pdf. A reader interested in this problem should look to [16] and to references therein. In the experimental section, ADBag with equal-area histogram created in the batch training is compared to the ADBag with the on-line histogram with the conclusion that both offer the same performance.

The chosen on-line histogram approximates the distribution of data in a set of pairs $\mathbf{H} = \{(z_1, m_1), \ldots, (z_b, m_b)\}$, where $z_i \in \mathbb{R}$ and $m_i \in \mathbb{N}$, and b is an upper bound on the number of histogram bins. The algorithm maintains pairs (z_i, m_i), such that every point z_i is surrounded by m_i points, of which half is to the left and half is to the right to z_i. Consequently, the number of points in the interval $[z_i, z_{i+1}]$ is equal to $\frac{m_i + m_{i+1}}{2}$, and the probability of point $z \in (z_i, z_{i+1})$ is estimated as a weighted average.

The construction of $\mathbf{H}$ is described in Algorithm 1. It starts with $\mathbf{H} = \{\}$ being an empty set. Upon receiving a sample, $z = x^{\mathrm{T}} w$, it looks if there is a pair (z_i, m_i) in $\mathbf{H}$ such that z is equal to z_i. If so, the corresponding

Algorithm 2: Algorithm returning approximate of probability density in point x projected on the vector w.

$\mathbf{H} = \mathbf{H} \cup \{(z_{\min}, 0), (z_{\max}, 0)\}$;
Sort pairs in $\mathbf{H}$ such that $z_{\min} < z_1 < z_2 < \ldots < z_{\max}$;
$z = x^{\mathrm{T}} w$;
if $\exists (i | z_i < z \leq z_{i+1})$ **then**
  return $\frac{z_i m_i + z_{i+1} m_{i+1}}{2M(z_{i+1} - z_i)}$;
else
  return 10^{-10}
end
;

count m_i is increased by one. If not, a new pair $(z, 1)$ is added to $\mathbf{H}$. If size of $\mathbf{H}$ exceeds the maximal number of bins b, the algorithm finds two closest pairs (z_i, m_i), (z_{i+1}, m_{i+1}), and replaces them with an interpolated pair $\left(\frac{z_i m_i + z_{i+1} m_{i+1}}{m_i + m_{i+1}}, m_i + m_{i+1}\right)$. Keeping z_i sorted makes all above operations efficient.

The estimation of the probability density in point $z = x^{\mathrm{T}} w$ is described in Algorithm 2. Assuming pairs in $\mathbf{H}$ are sorted according to z_i (the sorting is explicitly stated, but as has been mentioned above, for efficiency $\mathbf{H}$ should be sorted all the time), an i such that $z_i < z \leq z_{i+1}$ is found first. If such i exists, then the density in z is estimated as $\frac{z_i m_i + z_{i+1} m_{i+1}}{2M(z_{i+1} - z_i)}$, where $M = \sum_{i=1}^{b} m_i$. Otherwise, it is assumed that z is outside the estimated region and 10^{-10} is returned.

3.3 Aggregation of weak detectors

ADBag's output on a sample $x \in \mathbb{R}^d$ can be expressed as

$$\begin{aligned} f(x) &= -\frac{1}{k} \sum_{i=1}^{k} \log \hat{p}_i(x w_i^{\mathrm{T}}) \\ &= -\log \left(\prod_{i=1}^{k} \hat{p}_i(x w_i^{\mathrm{T}}) \right)^{\frac{1}{k}} \\ &\sim -\log p(x w_1, x w_2, \ldots, x w_k), \end{aligned} \tag{2}$$

where $\hat{p}_i$ denotes empirical marginal pdf along the projection w_i, and $p(x w_1, x w_2, \ldots, x w_k)$ denotes the joint pdf.

The equation shows that ADBag's output is inversely proportional to the joint probability of the sample under the assumption that marginals on projection vectors w_i and w_j are independent $\forall i, j \in k$, $i \neq j$ (used in the last line of Equation (2)). Similarly, the output can be viewed as a negative log-likehood of

the sample, meaning that less likely the sample is, the higher value of anomaly it gets.

The independence of xw_i^{T} and xw_j^{T} for $i \neq j$ assumed in last equation in (2) is questionable and in reality it probably does not hold. Nevertheless, the very same assumption is made in the naïve bayes classifier, which despite that it is almost always violated, gives results competitive to more sophisticated classifiers. Zhang [28] explains this phenomenon from a theoretical point of view and gives conditions, under which effects of conditional dependencies cancel out making naïve bayes equal to the bayes classifier. These conditions depend on the probability distribution of both classes and they are difficult to be verified in practice, since they require the exact knowledge of conditional dependencies among features. Nevertheless, due to ADBag's similarity to the Parzen window classifier, the similar argumentation might explain ADBag's performance.

Another line of thoughts can relate ADBag to a PCA based detector [25]. If dimension d is sufficiently high, then projection vectors w_i and w_j, $i \neq j$ are approximately orthogonal. Assuming again the independence of xw_i^{T} and xw_j^{T}, the projected data are orthogonal and uncorrelated, which are the most important properties of Principal Component Analysis (PCA).

3.4 Hyper-parameters

ADBag is controlled by two hyper-parameters: the number of weak detectors k and the number of histogram bins b within every detector. Both parameters are very predictable in a way they influence the accuracy.

Generally speaking, the higher the number of weak detectors, the better the accuracy. Nevertheless, after certain threshold, adding more detectors does not significantly improve the accuracy. In all experiments in Section 4.2, we set $k = 150$. Afterward investigation of the least k at which ADBag has the accuracy above 99% of the maximum has found that this k was most of the time well below 100.

The number of histogram bins was set to $b = [\sqrt{n}]$, where n is the number of samples. The rationale behind this choice is the following. If the number of samples $n \to +\infty$, and $b = \sqrt{n}$, then the equal area histogram converges to the true probability distribution function. In practice, b should be set according to available resources and expected number of samples. Interestingly, investigation on a dataset with millions of features and samples revealed that the effect of b on the accuracy is limited and small values of b are sufficient (see Section 4.3 for details).

Both hyper-parameters influence the computational complexity of ADBag's training and classification. It is easy to see that both complexities depend at most linearly on both parameters.

4 Experiments

ADBag's accuracy was evaluated and compared to the state of the art on 36 problems from UCI database [6] listed under the category "classification problems with numerical attributes without missing variables".

The following algorithms were chosen due to their generality and acceptance by a community: PCA based anomaly detector [25], OC-SVM [23], FRAC [19], KNN [12], and LOF [2]. Although these algorithms were not designed for real-time on-line learning, they were selected because they provide a good benchmark.

For every problem, the class with the highest number of samples were used as a nominal class and all samples from remaining classes were used as representatives of anomalous class. In one repetition of the experiment, 75% of nominal samples was used for training and the rest was used for testing. The samples from the anomalous class were not sampled — all of them were always used. The data have been always normalized to have zero mean and unit variance on the training part of the nominal class. The distance-based algorithms (KNN and LOF) used the training data as a data to which distance of classified sample was calculated. Every experiment was repeated 100 times.

To avoid problems with a tradeoff between false positive and false negative rate, the area under the ROC curve (AUC) is used as a measure of the quality of the detection. This measure is frequently used for comparisons of this kind.

The matlab package used in the evaluation process is available at `http://agents.fel.cvut.cz/~pevnak`.

4.1 Settings of hyper-parameters

LOF and KNN method both used $k = 10$ (recall that in both methods, k denotes the number of samples determining the neighborhood), as recommended in [2].

OC-SVM with a Gaussian kernel is cumbersome to use in practice, since its two hyper-parameters (width of the Gaussian kernel γ and expected false positive rate ν) have unpredictable effect on the accuracy (note that anomalous samples are not available during training). Hence, the following heuristic has been adopted. A false positive rate on the grid

$$(\nu, \gamma) \in \left\{ \left(0.01 \cdot 2^i, \frac{10^j}{d} \right) | i \in \{0, \dots, 5\}, j \in \{-3, \dots, 3\} \right\},$$

has been estimated by five-fold cross-validation (d is the input dimension of the problem). Then, the lowest ν and γ (in this order) with estimated false positive rate below 1% has been used. If such combination of parameters does not exist, the combination with the lowest false positive rate has been used. The choice of 1% false positive rate is justified by training on samples from the nominal class only, where no outliers should be present. The reason for choosing lowest ν and γ is that a good generalization is expected. The choice of the parameters is probably not optimal for maximizing AUC, but it shows the difficulty of using algorithms with many hyper-parameters. SVM implementation has been taken from the libSVM library [4].

FRAC detector used ordinary linear least-square estimators, which in this setting does not have any hyper-parameters. The PCA detector based on principal component transformation used all components with eigenvalues greater than 0.01.

ADBag used $k = 150$ weak detectors, and $b = \sqrt{n}$ bins in the histogram (n is the number of samples from nominal class). As will be discussed later, 150 weak detectors is for most problems unreasonably high, but it was used to avoid the poor performance due to their low number. The same holds for b.

4.2 Experimental results

Table 1 shows average AUCs of compared detectors on all 36 tested problems. The first thing to notice is that there is no single detector dominating the others. Every detector excels in at least one problem, and it is inferior in other. Therefore the last row in Table 1 shows the average rank of a given detector over all problems (calculated only for unsupervised detectors with batch learning). According to this measure, KNN, and the proposed ADBag detector with equal-area histogram provide overall the best performance, and they are on average equally good.

The result shows that increased complexity of the detector does not necessarily lead to a better performance, which can be caused by more complicated setting of parameters to be tuned.

Column captioned $k_{\min}$ in Table 1 shows a sufficient number of weak detectors, determined as the least k providing AUC higher then $0.99 \cdot \text{AUC}$ on 150 detectors. For all problems, the sufficient number of weak detectors is well below the maximum number of 150. For many problems, $k_{\min}$ is higher than the dimension of the input space. This shows that diverse set of random projections provides a different views on the problem leading to the better accuracy.

For almost all problems, the accuracy *increases* with the number of projections. The only exception is "musk-2" dataset, which is not unimodal, as the plot of first two principal components reveals three clusters of data. Contrary, the "spect-heart" is actually a difficult problem even for the supervised classification, as the Adaboost [7] algorithms achieves only 0.62 AUC.

Investigation of problems on which ADBag is worse than the best detector shows that ADBag performs poorly in cases, where the support of the probability distribution of nominal class is not convex, and it encapsulates the support of the probability distribution of the anomalous class. We believe that these cases are rare in very high-dimensional problems, for which ADBag is designed.

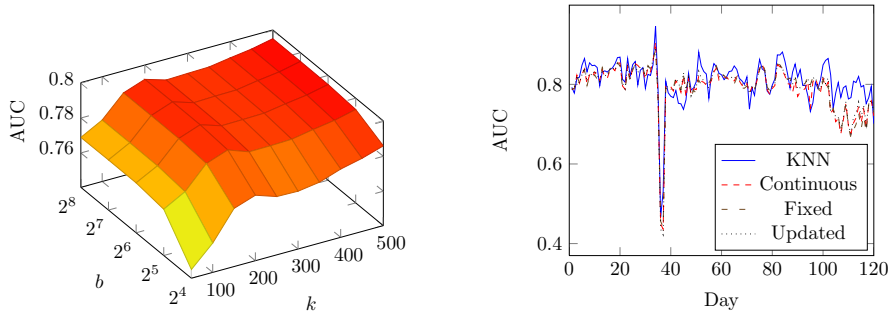
Finally, comparing accuracies of batch and on-line versions (Table 1), it is obvious that the on-line version of ADBag is no worse than the batch version. This is important for an efficient processing of large data and application in non-stationary problems [9].

	batch unsupervised methods						online			
dataset	FRAC	PCA	KNN	LOF	SVM	ADBag	ADBag	d	n	$k_{\min}$
abalone	0.44	0.46	0.46	0.50	**0.60**	0.59	0.59	8	1146	10
blood-transfusion	0.41	0.53	**0.56**	0.46	0.53	**0.56**	0.55	4	428	2
breast-cancer-wisconsin	0.94	**0.96**	0.95	0.95	0.90	**0.96**	**0.96**	30	268	20
breast-tissue	0.87	0.90	0.91	0.92	**0.94**	**0.94**	**0.94**	9	16	18
cardiotocography	0.71	0.72	0.73	0.80	**0.87**	0.82	0.82	21	1241	41
ecoli	0.93	0.97	**0.98**	**0.98**	**0.98**	**0.98**	**0.98**	7	107	12
gisette	—	0.78	0.78	**0.83**	0.74	0.74	0.75	5000	2250	78
glass	0.64	**0.68**	0.67	**0.68**	0.55	0.67	0.65	9	57	10
haberman	**0.67**	0.61	**0.67**	0.66	0.33	0.64	0.64	3	169	28
ionosphere	0.96	0.96	**0.97**	0.95	0.93	0.96	0.96	34	169	39
iris	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	4	38	4
isolet	0.87	0.98	**0.99**	**0.99**	**0.99**	**0.99**	0.98	617	180	40
letter-recognition	**0.99**	**0.99**	0.98	0.98	0.79	0.96	0.95	16	610	53
libras	0.66	**0.88**	0.72	0.75	0.72	0.82	0.81	90	18	79
madelon	0.51	0.51	0.52	0.52	**0.53**	0.51	0.51	500	975	59
magic-telescope	0.83	0.80	0.83	**0.84**	0.69	0.73	0.73	10	9249	27
miniboone	0.62	0.54	0.77	0.54	—	**0.83**	**0.83**	50	70174	19
multiple-features	0.77	**1.00**	0.99	**1.00**	0.99	0.99	0.99	649	150	12
musk-2	0.83	0.79	0.79	**0.84**	0.18	0.53	0.42	166	4186	1
page-blocks	**0.96**	**0.96**	**0.96**	0.91	0.86	0.95	0.94	10	3685	12
parkinsons	0.71	0.68	0.61	0.67	**0.85**	0.74	0.73	22	110	73
pendigits	**1.00**	**1.00**	0.99	**1.00**	0.99	0.99	0.99	16	858	6
pima-indians	0.72	0.72	**0.75**	0.68	0.64	0.73	0.73	8	375	32
sonar	**0.67**	0.66	0.53	0.62	0.65	0.63	0.63	60	83	99
spect-heart	0.31	0.27	0.22	0.23	**0.68**	0.32	0.31	44	159	1
statlog-satimage	0.81	0.97	**0.99**	**0.99**	0.85	**0.99**	**0.99**	36	1150	21
statlog-segment	0.99	**1.00**	0.99	0.99	0.98	0.99	0.99	19	248	10
statlog-shuttle	0.91	0.98	**1.00**	**1.00**	0.65	0.92	0.92	8	34190	9
statlog-vehicle	**0.98**	**0.98**	0.89	0.94	0.60	0.85	0.86	18	164	69
synthetic-control-chart	0.97	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	60	75	9
vertebral-column	0.85	0.88	0.88	0.89	0.87	**0.95**	0.90	6	150	20
wall-following-robot	0.75	0.68	**0.78**	0.74	0.63	0.70	0.68	24	1654	59
waveform-1	0.89	0.90	0.90	0.89	**0.93**	0.91	0.92	21	1272	67
waveform-2	0.81	**0.82**	0.81	0.80	0.76	0.78	0.80	40	1269	111
wine	0.93	**0.95**	**0.95**	0.93	0.91	0.93	0.93	13	53	63
yeast	0.72	0.72	0.72	0.71	0.67	**0.74**	**0.72**	8	347	33
Average rank	4.0	3.2	**3.1**	3.2	4.3	**3.1**				

Table 1: Average unnormalized area under ROC curve calculated from 100 repetitions (higher is better). The best performance for a given problem are bold-faced. The last row shows the average rank of the algorithm from all 36 problems (lower is better).

4.3 URL dataset

A URL dataset [17] contains 2.4 million samples with 3.2 million features, hence it is a good dataset where ADBag's power can be demonstrated. Each sample in the dataset contains sparse features extracted from an URL. The class of benign samples contains random URL (obtained by visiting `http://random.yahoo.com/bin/ryl`). The class of malicious samples was obtained by extracting links in spam e-mails. URL's were collected during 120 days. The original work used the dataset in a supervised binary classification scenario obtaining accuracy around 98%. Here, we use the dataset in the anomaly detection scenario utilizing samples from the benign class for training.



(a) ADBag with respect to k and b.

(b) KNN and versions of ADBag.

Fig. 1: Left figure shows AUC of updated ADBag on URL dataset for different number of weak detectors k, and histogram bins b. Right figure shows AUC of KNN detector in reduced space, Continuously updated ADBag (Continuous), ADBag trained on the first day (Fixed), and ADBag trained every day (Updated) with respect to days.

ADBag was evaluated with different number of weak detectors $k \in \{50, 100, 150, \ldots, 500\}$, and different number of bins $b \in \{16, 32, 64, 128, 256\}$. A three strategies to train ADBag were investigated.

Fixed ADBag was trained on benign samples from the day zero only, which has never been used for evolution. *Continuous* ADBag was trained on all samples from benign class up to the day before the testing data came from. This means that if Continuous ADBag was evaluated on data from day l, the training used benign samples from days $0, 1, 2, \ldots, l-1$ (Continuous ADBag was trained in the on-line manner). Finally, *Updated* ADBag was trained on benign samples from the previous day than the data for evaluation.

Note that the prior art used in the previous section cannot be used directly for a benchmarking purposes, because it cannot handle these data. In order to have a method that ADBag can be compared to, a strategy from [5] has been adopted

and used with the KNN detector (the best according to results in previous subsection). Specifically, a random projection matrix $W \in \mathbb{R}^{3.2 \cdot 10^6, 500}$, $W_{ij} \sim N(0, 1)$ has been created and all samples were projected in this new, k-dimensional space. Remark that due to Johnson-Lindenstrauss lemma, L_2 distances between points should be approximately preserved, hence the KNN method should work without modification. The KNN method was executed with 10 and 20 nearest neighbors, finding the former being better.

Missing features were treated as zeros, which allows to efficiently handle not-yet-seen variables by adding new row(s) to projection matrix W. This strategy has been used for both ADBag and KNN methods.

Figure 1b shows AUCs of KNN and three variants of ADBag in every day in the data set. According to the plot, in some days ADBag detectors were better, whereas in other days it was vice versa. The average difference between the KNN and ADBag retrained every day was about 0.02, which is negligible difference considering that Continuous ADBag was approximately 27 times faster. Interestingly, all three versions of ADBag provided similar performance, meaning that the benign samples were stationary. This phenomenon has been caused by the fact that the distribution of all URLs on the internet has not changed considerably during 120 days, when benign samples were acquired.

Figure 1a shows, how the accuracy of Continuous ADBag changes with the number of weak detectors k, and the number of histogram bins b. As expected, higher values yields to higher accuracy, although competitive results can be achieved for $k = 200$ and $b = 32$. This suggests that b can have a low value, which does not have to be proportional to the number of samples n.

Continuos ADBag's average time of update and classification per day for the most complex setting with $k = 500$ and $b = 256$ was 5.86s. The average classification time for the KNN detector with $k = 500$ and 10 nearest neighbors was 135.57. Both times are without projection of data to a lower-dimensional space, which was done separately. This projection took 669.25s for 20,000 samples. These numbers show that ADBag is well suited for efficient detection of anomalous events on large-scale data. Its accuracy is competitive to the state of the art methods, while its running times are order of magnitude lower. Running times were measured on Macbook air equipped with a 2-core 2Ghz Intel core i7 processor and 8Gb of memory.

5 Conclusion

This paper has proposed an anomaly detector with bounded computational and storage requirements. This type of detector is important for many contemporary applications requiring to process large data, which are beyond capabilities of traditional algorithms, such as one-class support vector machine or nearest-neighbor based methods.

The detector is built as an ensemble of weak detectors, where each weak detector is implemented as a histogram in one dimension. This one dimension is obtained by projecting the input space to a randomly generated projection

vector. Projection vectors are generated randomly, which simultaneously creates the needed diversity between weak detectors.

The accuracy of the detector was compared to five detectors from the prior art on 36 classification problems from UCI datasets. According to the results, the proposed detector and the nearest-neighbor based detector provide overall the best performance. It was also demonstrated that the detector can efficiently handle dataset with millions of samples and features.

The fact, that the proposed detector is competitive to established solutions is especially important, if one takes its small computational and memory requirements into the account. Moreover, the detector can be trained on-line on a data-streams, which open doors to its application in non-stationary problems.

With respect to experimental results, the proposed detector represents an interesting alternative to established solutions, especially if large data needs to be efficiently handled. It would be interesting to investigate the impact of sparse random projections on the accuracy, as this will increase the efficiency and enable the detector to be applied on data with missing features.

6 Acknowledgments

This work was supported by the Grant Agency of Czech Republic under the project P103/12/P514.

References

1. Y. Ben-Haim and E. Tom-Tov. A streaming parallel decision tree algorithm. *The Journal of Machine Learning Research*, 11:849–872, 2010.
2. M. M. Breunig, H.P. Kriegel, R.T. Ng, and J. Sander. Lof: identifying density-based local outliers. *SIGMOD Rec.*, 29(2):93–104, 2000.
3. V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Computing Surveys (CSUR)*, 41(3):1–58, 2009.
4. C. C. Chang and C. J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.
5. T. de Vries, S. Chawla, and M. E. Houle. Finding local anomalies in very high dimensional space. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 128–137. IEEE, 2010.
6. A. Frank and A. Asuncion. UCI machine learning repository, 2010. `http://archive.ics.uci.edu/ml`.
7. Y. Freund and R. E. Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. In *Computational learning theory*, pages 23–37. Springer, 1995.
8. J. Gao and P.-N. Tan. Converting output scores from outlier detection algorithms into probability estimates. In *Data Mining, 2006. ICDM'06. Sixth International Conference on*, pages 212–221. IEEE, 2006.
9. E. Hazan and C. Seshadhri. Efficient learning algorithms for changing environments. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 393–400. ACM, 2009.

10. W. B. Johnson and J. Lindenstrauss. Extensions of lipschitz mappings into a hilbert space. *Contemporary mathematics*, 26(189-206):1, 1984.
11. J. Kivinen, A. J. Smola, and R.C. Williamson. Online Learning with Kernels. *IEEE Transactions on Signal Processing*, 52(8):2165–2176, 2004.
12. E. Knorr and R. T. Ng. Finding intensional knowledge of distance-based outliers. In *Proceedings of the International Conference on Very Large Data Bases*, pages 211–222, 1999.
13. H.-P. Kriegel and A. Zimek. Angle-based outlier detection in high-dimensional data. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 444–452. ACM, 2008.
14. A. Lazarevic and V. Kumar. Feature bagging for outlier detection. In *Conference on Knowledge Discovery in Data: Proceeding of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, volume 21, pages 157–166, 2005.
15. P. Li. Very sparse stable random projections for dimension reduction in l α ($0 < \alpha \leq 2$)) norm. *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '07*, page 440, 2007.
16. X. Lin. Continuously maintaining order statistics over data streams. In *Proceedings of the eighteenth conference on Australasian database-Volume 63*, pages 7–10. Australian Computer Society, Inc., 2007.
17. J. Ma, L. K Saul, S. Savage, and G. M. Voelker. Identifying suspicious urls: an application of large-scale online learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 681–688. ACM, 2009.
18. H. V. Nguyen, H. H. Ang, and V. Gopalkrishnan. Mining outliers with ensemble of heterogeneous detectors on random subspaces. In *Database Systems for Advanced Applications*, pages 368–383. Springer, 2010.
19. K. Noto, C. Brodley, and D. Slonim. Frac: a feature-modeling approach for semi-supervised and unsupervised anomaly detection. volume 25, pages 109–133. Springer US, 2012.
20. E. Parzen. On estimation of a probability density function and mode. *The annals of mathematical statistics*, 33(3):1065–1076, 1962.
21. N. Pham and R. Pagh. A near-linear time approximation algorithm for angle-based outlier detection in high-dimensional data. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 877–885. ACM, 2012.
22. D. Pokrajac, A. Lazarevic, and J. L. Latecki. Incremental Local Outlier Detection for Data Streams. In *2007 IEEE Symposium on Computational Intelligence and Data Mining*, pages 504–515. IEEE, 2007.
23. B. Schölkopf, J. C. Platt., J. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Comput.*, 13(7):1443–1471, 2001.
24. E. Schubert, R. Wojdanowski, A. Zimek, and H.-P. Kriegel. On evaluation of outlier rankings and outlier scores. In *Proceedings of the 12th SIAM International Conference on Data Mining (SDM), Anaheim, CA, 2012*, pages 1047–1058, 2012.
25. M. L. Shyu. A novel anomaly detection scheme based on principal component classifier. Technical report, DTIC Document, 2003.
26. D. M. J. Tax and R. P. W. Duin. Support vector data description. *Mach. Learn.*, 54(1):45–66, 2004.
27. D.M.J. Tax and P. Laskov. Online svm learning: from classification to data description and back. In *Neural Networks for Signal Processing, 2003. NNSP'03. 2003 IEEE 13th Workshop on*, pages 499–508, 2003.

28. H. Zhang. The optimality of naive bayes. In V Barr and Z Markov, editors, *Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference*. AAAI Press, 2004.
29. A. Zimek, E. Schubert, and H.-P. Kriegel. A survey on unsupervised outlier detection in high-dimensional numerical data. *Statistical Analysis and Data Mining*, 5(5):363–387, 2012.