Information Sciences 487 (2019) 115–127

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Large scale anomaly detection in mixed numerical and categorical input spaces

Carlos Eiras-Franco[a,*], David Martínez-Rego[a], Bertha Guijarro-Berdiñas[a], Amparo Alonso-Betanzos[a], Antonio Bahamonde[b]

[a] Research Center on Information and Communication Technologies (CITIC), Universidade da Coruña, A Coruña 15071, Spain
[b] Centro de Inteligencia Artificial, Universidad de Oviedo, Gijón 33204, Spain

ARTICLE INFO

Article history:
Received 20 March 2018
Revised 11 February 2019
Accepted 3 March 2019
Available online 4 March 2019

Keywords:
Anomaly detection
Outlier detection
Scalability
Big data
Mixed data
Synthetic dataset generator

ABSTRACT

This work presents the ADMNC method, designed to tackle anomaly detection for large-scale problems with a mixture of categorical and numerical input variables. A flexible parametric probability measure is adjusted to input data, allowing low likelihood values to be tracked as anomalies. The main contribution of this method is that, to cope with the variable nature of the variables, we factorize the joint probability measure into two parts, namely, the marginal density of the continuous variables and the conditional probability of the categorical variables given the continuous part of the feature vector. The result is a model trained through a maximum likelihood objective function optimized with stochastic gradient descent that yields an effective and scalable algorithm. Compared with other well-known anomaly detection algorithms over several datasets, ADMNC is observed to both offer top level accuracy in datasets that are out of reach for the most effective existing methods and to scale up well to processing very large datasets. This makes it a powerful tool for solving a problem growing in popularity that currently lacks suitable scalable algorithms.

1. Introduction

An anomaly or outlier can be defined as “an observation which deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism” [22]. Detecting anomalies is an old discipline for statisticians [17], denominated *outlier detection*. Since those days, this type of method has become increasingly important. Anomaly detection is especially useful in practical situations where the dataset is both numerous and contains unexpected events that carry the most important information. Several challenges stand in the way of developing a general technique for anomaly detection: the growing number of domains of application (detection of intrusions [30], surveillance [46], fraud [2], machine faults [14,19,35]) adds high variability to the proposed solutions, while the scarcity of labeled data from real-world processes [9,11] makes it difficult to test the generalization of new solutions. As a consequence, the data available in practice for building the model is usually unlabeled [11,18,45]. In addition, the regions of the input space that are likely to contain only non-anomalous elements can be very complex in nature. Therefore, deciding a prior shape for this region through the choice

* Corresponding author.
E-mail address: carlos.eiras.franco@udc.es (C. Eiras-Franco).

https://doi.org/10.1016/j.ins.2019.03.013
0020-0255/

of a specific distribution or geometric shape is a potentially difficult task that can introduce bias, preventing the generation of a meaningful model.

Another major difficulty, on which we focus here, is the type of input data. The introduction of mixed numerical/categorical data can complicate the modeling of correlations between input variables. This has meant that many of the popular anomaly detection techniques: (a) can only deal with categorical or numerical data [1,13,47], (b) leave to the practitioner the responsibility of dealing with this issue through (non-formal) bespoke processes, or (c) introduce heuristic criteria to deal with mixed nature data [21,39].

Furthermore, the quantity of data produced has dramatically increased in recent years, at a much faster pace than computational power. This imbalance renders some of the most popular algorithms incapable of dealing with the amounts of data that users need to analyze. Algorithms with quadratic spatial or time complexity are no longer suitable for this kind of analysis. This limitation has swung the focus to the scalability of algorithms, spawning an active field known as big data learning. One successful approach has come from the use of new cluster computing frameworks and techniques that, paired with new algorithms with reduced complexity, help bridge the gap between computing power and processing needs.

With this research, we aimed to explore a strategy to model anomaly detection problems in which the data are numerous and contain categorical and numerical input variables. We adopted a probabilistic view of the problem and dealt with each kind of variable individually in order to approximate the joint probability measure function with a parametric model. This approach differs from the state of the art in this field in that, instead of departing directly from an heuristic concept of *outlierness* in mixed categorical/numerical spaces, it starts from a formal formulation of the problem in terms of a joint probability measure function approximation and adopts a parametric structure that makes this feasible to compute. This makes the proposed algorithm both theoretically and technically sound. Additionally, by splitting the model into two smaller parts, the computational requirements are reduced, which enhances the scalability of the algorithm. The whole model is trained through a maximum likelihood objective function optimized with stochastic gradient descent. Therefore, the algorithm lends itself well to parallel computation, which allows the model to scale up to large datasets, both in terms of features and sample sizes, making it an appealing option for big data applications. To demonstrate this, an implementation of the algorithm (denominated Anomaly Detector for Mixed Numerical and Categorical inputs, ADMNC) in the popular cluster computing framework Apache Spark is provided.[1]

Section 2 reviews related work in this area. Section 3 presents the formal framework and Section 4 introduces the parametric formulation of the problem. Section 5 reports a collection of experiments that show the properties of the proposed method in real datasets, as well as the definition of a synthetic dataset generator and further experiments with the resulting datasets. Section 6 summarizes the main conclusions and future work.

2. Related work

Numerous anomaly detection techniques have been developed, either from an application-specific or a more general-purpose point of view. Anomaly detection application domains impose restrictions which dramatically determine the design of the algorithms. Consequently, the research in this area has yielded only a few general algorithms in recent years [28].

Anomaly detection approaches can be classified according to the nature of **the** input data. We have assumed that each instance can be described using a set of attributes. These can be of different types, such as binary, categorical or numerical. The nature of the attributes determines the applicability of anomaly detection techniques. Different statistical models and algorithms have been designed for numerical and categorical data [12]. Some anomaly detection models can only deal with categorical data [1,13,47], whereas numerical variables have been treated mainly through statistical parametric and non-parametric models [4,42], geometrical approximations [35], using binary trees [33] and autoencoder neural networks [20,23,38]. In addition, there have been numerous efforts to deal with the problem of mixed numerical/categorical anomaly detection. Current approaches in this last group can be classified in one of the following abstract strategies:

- *Categorical space* techniques. These algorithms build an anomaly detection model specially devised for categorical variables and transform any numerical variable into a categorical space through a previous discretization phase. In this group we can find HOT [48], in which the set of outliers in a dataset is detected using a specially devised data structure called a hypergraph and a local test for outliers based on a frequent itemset counting strategy, and OutRank [37] that detects anomalies using random walks on an adjecency graph. Another approach consists in tackling the problem using information theory concepts [24,49] but, again, only categorical attributes are considered and numerical attributes need to be circumvented through discretization.
- *Metric-centered* techniques. These methods define an anomaly as a point which lies in a low density region in comparison with its neighborhood. They rely on a function that calculates the similarity between elements in the input space and so can be extended to the mixed numerical/categorical case and other types of structured data [43] through a tailored similarity function. LOF (Local Outlier Factor) [8] can be considered the seminal work in this area. The basic criteria of these methods has also inspired subsequent improvements for high dimensional spaces [31] and improved density criteria such as JeffreyXu et al. [27]. LOCI (LOcal Correlation Integral) is a similar technique that improves on LOF, as it is able to detect outliers and also groups of outliers without user-required cut-offs [40]. These techniques present

[1] The implementation of ADMNC in Apache Spark is available for download at http://github.com/eirasf/ADMNC/.

challenges in (a) devising effective similarity measures for mixed numerical/categorical input spaces and (b) scalability, since the similarity matrix needs to be computed before moving on to the detection phase.

- *Mixed-criteria* techniques. This group of algorithms tackles the nature of numerical and categorical data separately, by trying to design a criterion which encompasses the analysis of an element in both spaces. Solutions that tackle this problem using adaptations of supervised learning techniques like AdaBoost have been described [26], although its requirement for labeled samples prevents its use in the most common use cases. In this group we can also classify LOADED [21], which blends categorical-categorical, categorical-numerical and numerical-numerical correlations in a single criterion using frequent itemset concepts and local correlation matrices. This algorithm has the drawback of high execution times for high dimensional datasets, because although its computational complexity scales linearly with the number of data points, it scales quadratically with the number of numerical attributes, and grows even more computationally costly with the number of categorical attributes. Although a more efficient version named RELOADED was introduced in [39], this still requires more computational effort than more recent methods like ODMAD [29], which first identifies elements with an anomalous categorical part by finding unusual values or combinations of values, and then, for the elements not deemed as anomalies, computes the pairwise similarity of their numerical part to that of points sharing the same categorical values. This takes into account the relationship of the numerical part with each of the categorical values, but disregards any possible dependence on a combination of categorical values. Moreover, its computational complexity, even though less than that of the aforementioned RELOADED, still scales exponentially with the number of categorical attributes, limiting its use to datasets with few such variables. Another algorithm in this category is POD [50], but its reliance on k-nearest neighbor distances entails a computational complexity that renders it unsuitable for large datasets. There have also been efforts to provide a statistical foundation to this problem, like MITRE [34], which uses a generalized linear model with additional latent variables to model correlations and error. Large magnitudes of the error are then used to identify anomalies. Despite being a sound model, its inference is computationally costly [16], particularly when the number of correlations modeled is high. Moreover, the method relies on the user providing the domain knowledge to identify explanatory and dependent variables; the alternative of assuming all variables to be dependent would drive the execution time up. These characteristics limit its scalability. Finally, the use of deep belief networks for anomaly detection has been proposed in a method named MIXMAD [16] but, although this method scales well in terms of number of variables, its computational complexity makes it unfit for processing large datasets.

In this research we focused on devising a probabilistic strategy able to solve anomaly detection problems, where input elements belong to an input space which mixes numerical and categorical variables. The proposed algorithm is closely related to *mixed-criteria* techniques in the sense that correlations between categorical and numerical variables are explicitly modeled. In addition, the method overcomes the scalability problems of previous approaches, and can tackle both large-scale and high-dimensional problems.

3. Basic formulation

We aim to obtain the best fit of a probability measure function for the data under normal conditions. In subsequent monitoring of new data elements, these are assigned a score and those whose score does not reach a pre-specified threshold are considered anomalies. Formally, given a dataset $\mathcal{D} = \left\{\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{|\mathcal{D}|}\right\}$, we need to estimate $\mathrm{P}(\boldsymbol{x})$.

If we have an homogeneous set of variables, this problem can be reduced to probability density function (pdf) parameter learning. For instance, if all the variables under normal conditions can be well represented by a Gaussian, we can directly elicit the moments of a complete Gaussian from a dataset by maximizing the likelihood of the data, or alternatively, follow a Bayesian approach with an adequate prior.

However, in many situations, datasets have a mix of categorical and numerical variables. For ease of reference we rewrite the dataset as

$$\mathcal{D} = \left\{(\boldsymbol{x}_0, \boldsymbol{y}_0) \ldots, (\boldsymbol{x}_{|\mathcal{D}|}, \boldsymbol{y}_{|\mathcal{D}|})\right\},$$

where $\boldsymbol{x}_i$ is the numerical or continuous component, and $\boldsymbol{y}_i$ is the categorical or nominal counterpart of the ith instance of the dataset.

In this case, the model should individually take into account the nature of the two types of variables. In this work, we propose the following heuristic factorization of the pdf:

$$P(\boldsymbol{y}, \boldsymbol{x}) = P(\boldsymbol{y}|\boldsymbol{x})P(\boldsymbol{x}). \tag{1}$$

With this partition of the pdf, we can adopt a suitable technique for estimating the parameters of each part independently, while accounting for possible interactions between the two parts.

It was previously mentioned that an incorrect assumption about the shape of the underlying distribution could induce bias that could harm the accuracy of the model. Nevertheless, since we had to make that decision, we tried to adopt the most flexible model that was still computable in a closed form. We heuristically adopted a flexible parametric approach for both the conditioned probability of the categorical variables and the marginal probability of the numerical variables. In Section 5 we test the adequacy of these heuristic assumptions with results obtained in experiments on several datasets.

Below we describe the models used for each part of the proposed factorization.

3.1. Numerical part

Mixture models are the most flexible parametric option for estimating this marginal. The Gaussian mixture model (GMM) is the first appealing option due to its closed form parameter update formulas. In particular, we used the existing implementation of GMM available in Apache Spark, which uses the expectation-maximization algorithm to induce the maximum-likelihood model for the given set of samples. Since GMM is very sensitive to the initial values of the means of the Gaussians, we first performed KMeans clustering on a small sample from the dataset. We then used the resulting centroids as the initial values for these means and computed the empirical standard deviations of the clusters to obtain the initial diagonal covariance matrix. For the KMeans algorithm we used the default implementation included in Spark [3].

3.2. Categorical part

The categorical part of each element in the dataset can be assumed, without any loss of generality, to be a binary vector

$$\boldsymbol{y} = (y^0, \ldots, y^k), \quad y^j \in \{0, 1\}$$

which can be obtained by one-hot encoding of the categorical variables of the element.

With this in mind, we can estimate $P(\boldsymbol{y}|\boldsymbol{x})$ as

$$P(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{w}) = \prod_{j=0}^{k} P(Y = y^j | (\boldsymbol{x}, \boldsymbol{m}_j), \boldsymbol{w}) \tag{2}$$

where $\boldsymbol{m}_j$ is just the one-hot representation of "j", and $\boldsymbol{w}$ is a parameter to be learned from the dataset using a logistic regression (LR) model. Specifically, the LR model computes the conditional probability by means of

$$P(Y = y^j | (\boldsymbol{x}, \boldsymbol{m}_j), \boldsymbol{w}) = \frac{1}{1 + e^{(2y^j - 1)\langle \boldsymbol{w}, (\boldsymbol{x}, \boldsymbol{m}_j)\rangle}} \tag{3}$$

where the notation $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ represents the dot product of $\boldsymbol{x}$ and $\boldsymbol{y}$. Thus, the learning process is reduced to obtaining the optimal parameters for both the LR model (which approximates the conditional probability of the categorical part of the elements) and the mixture model (for the marginal pdf of the numerical part of the elements). In the next section we show how a maximum likelihood strategy can effectively carry out the optimal parameter search.

4. Maximum likelihood parameter estimation

Let D be a dataset of points $(\boldsymbol{x}, \boldsymbol{y})$. Taking into account the expression (Eq. (1)) for the factorization of the joint pdf, the data log-likelihood has the following form:

$$\log L(D) = \sum_{i=1}^{|D|} \log P(\boldsymbol{y}_i | \boldsymbol{x}_i, \boldsymbol{w}) + \sum_{i=1}^{|D|} \log P(\boldsymbol{x}_i) \tag{4}$$

It is important to note that, since the first part of the data log-likelihood is completely independent of the parameters of the second addend, both optimization processes can be run in parallel and so exploit the structure of each problem separately. This can be achieved separately for each part and then combined in a unified algorithm.

Using the first term of (Eq. (4)), we aimed to obtain the parameters $\boldsymbol{w}^*$ that maximize

$$\boldsymbol{w}^* = \underset{\boldsymbol{w}}{\operatorname{argmax}} \left\{ \sum_{i=1}^{|D|} \log P(\boldsymbol{y}_i | \boldsymbol{x}_i, \boldsymbol{w}) - \nu \frac{\boldsymbol{w}^2}{2} \right\} \tag{5}$$

where ν is a hyper-parameter that controls an optional regularization term that is proportional to the norm of the parameter vector, added to cope with possible overfitting issues.

This is a well-known convex optimization problem that can be solved using a stochastic gradient descent (SGD) algorithm [6].

For the implementation of SGD in our experiments, at each iteration t, we updated the learning rate, as it is usually done, according to this formula:

$$\lambda_t = \frac{\lambda_0}{1 + \lambda_s (t - 1)} \tag{6}$$

Therefore, controlling the constants λ_0 and λ_s it is possible to tune the convergence of the algorithm, which is crucial in order to obtain a good model.

The implementation in Apache Spark parallelizes the minibatch step in the SGD process, potentially accelerating the whole process if there are several computational units available. The code is available for download at http://github.com/eirasf/ADMNC.

5. Experimental settings and results

In this Section we describe a set of experiments comparing our algorithm with other state-of-the-art methods in terms of both accuracy and time. The first subsection describes methodological issues related to how we measured the scores of the anomaly detection algorithms and the datasets and algorithms used in the experiments. We also describe a synthetic dataset generator and introduce the synthetic datasets used in our experiments. We then report performance results obtained with real world datasets and next make a comparison of the scalability of all algorithms. Finally, we report the results of experiments that show the effect of the complexity of the dataset on the results obtained by each algorithm.

5.1. Methodology

To measure the performance of the different methods we tried to simulate a real world environment. Thus, the learning algorithms were trained using only non-anomalous samples. The models thus obtained were then tested with a mixture of anomalous and non-anomalous samples. The performance scores were measured computing the area under the ROC (*Receiving Operator characteristic*) curve (AUC).

We account for the fact that in real situations we typically do not have anomalous samples to train learning algorithms. Additionally, for experimental purposes we may use datasets with an arbitrary fraction of anomalies, which is useful given the scarcity of datasets. Therefore, we were able to use binary classification tasks, selecting one of the classes as anomalous, as is common practice [16,29,34,38]. With this transformation, the obtained anomalies comply with the definition given in the Introduction.

To test the strength of our algorithm, we compared the scores with those achieved by the following state-of-the-art algorithms. First, we considered two algorithms that make a differential treatment of numerical and categorical variables: the well-known LOF and LOCI algorithms using Euclidean, Jaccard and Hamming distances, for which we used a Matlab implementation[2]. Additionally, we compared the results with other anomaly detection algorithms that do not differentiate between numerical and categorical variables. For this purpose we selected one-class support vector machine (OC-SVM) (with a radial basis function—RBF— and linear kernels), for which we used the Matlab interface of LibSVM.[3] It is worth noting that the complexity of this family of algorithms approaches quadratic time regarding the number of samples in favorable cases [7]; this makes them poor candidates for handling large amounts of data. Therefore, we also tested DOC-SVM [10], which is a distributed version of the same algorithm that can handle large datasets by splitting them, also implemented in Matlab.[4] Finally, we included in testing the recent iForest [33], implemented in R[5]; and PA-I [36],[6] also written in Matlab. For those algorithms that do not make a distinction between categorical and numerical variables, the categorical variables in the datasets were transformed using one-hot encoding.

The algorithms used for comparisons typically have several hyper-parameters. To find the best combination for each dataset, we performed a cross-validation (CV) with 5 folds for each possible set of hyper-parameter values. For each fold, the algorithm was trained with the non-anomalous examples from the training set and evaluated on all the samples of the test set. The hyper-parameters explored are listed in Table 1. The scores discussed in Section 5.2 are the best average AUC obtained in the CV procedure.

Below we describe the datasets employed in the experiments.

5.1.1. Real datasets

As stated above, it is very difficult to come by real-world datasets with labeled anomalies. Thus, the datasets used are those commonly employed for classification tasks, but re-purposed for anomaly detection. They were downloaded from the UCI Machine Learning Repository [32].

The datasets are reported in Table 2, where we distinguish between two groups. The first group includes small-medium sized datasets: Arrhythmia (Arrhyth), German Credit (GC), and 3 versions of Abalone. These versions were built choosing different classes as anomalous and non-anomalous; thus, Abalone 1–8 (Ab. 1), Abalone 9–11 (Ab. 9) and Abalone 11–29 (Ab. 11) were obtained using, respectively, classes 1, 9 and 11 as non-anomalous and classes 8, 11 and 29 as anomalous.

The second group of datasets, with a larger number of samples, contains versions of CoverType [5] and KDD99 [25] datasets. To transform CoverType (CT), instances of class 2 were assumed to be normal, while instances of class 4 were selected as anomalies. With respect to KDDCup99 it should be noted that, although there has been some criticism that it does not accurately represent an intrusion detection task, those discrepancies have no impact on the validity of the dataset for our purposes. Although the anomaly condition of the elements that are labeled as such may be disputed with the argument that the dataset constitutes a biased sample, our aim is to identify a minority set of instances that were

[2] https://github.com/jeroenjanssens/lof-loci-occ.

[3] https://www.csie.ntu.edu.tw/~cjlin/libsvm/#matlab.

[4] To the best of the authors knowledge, there are no other implementations of OC-SVM that can use non-linear kernels with a time complexity inferior to $\mathcal{O}(n^2)$.

[5] https://sourceforge.net/projects/iforest/.

[6] We tried to include LOADED in the comparison but despite our best efforts we could not find an implementation; also, the code obtained by strictly following the description provided by the authors resulted in an algorithm that performed very poorly.

Table 1
Hyper-parameters explored for each algorithm in the experiments reported in this section.

Algorithm	Hyper-parameters range
LOF	P $\in$ {0.01, 0.03, 0.05}, K $\in$ {2, 3, 5, 10}, (only Jaccard and Hamming) $\lambda \in$ {0.1, 0.3, 0.5, 0.7, 0.9}
LOCI	$\alpha \in$ {0.1, 0.3, 0.5}, (only Jaccard and Hamming) $\lambda \in$ {0.1, 0.3, 0.5, 0.7, 0.9}
OC-SVM	$\nu \in$ {0.01, 0.05, 0.1, 0.3}, (only RBF) $\gamma \in$ {0.01, 0.05, 0.1, 1, 3, 10}
iForest	rFactor $\in$ {0.01, 0.1, 0.5, 0.8, 1}, Row Samples $\in$ {0.01, 0.025, 0.05, 0.1}
PA-I	$\sigma \in$ {1, 2, 3, 4, 5}, C $\in$ {0.01, 0.025, 0.05, 0.1}, R $\in$ {0.97, 0.99}
ADMNC	$\nu \in$ {0.1, 1, 10, 100, 1000}, $\lambda_0 = 1$ $\lambda_s \in$ {0.0001, 0.001, 0.01, 0.1, 1} Number of gaussians $\in$ {2, 4}

Table 2
Real datasets used for the comparative study. The *Anomaly ratio* is the quotient of anomalous examples over the number of examples. The numbers of numerical / categorical features are in parentheses.

Dataset	# Samples	# Features(N/C)	Anomaly ratio
Arrhythmia	420	278 (271/7)	0.4357
German Credit (GC)	1000	20 (7/13)	0.3000
Abalone 1–8 (Ab. 1)	4177	10 (7/3)	0.3368
Abalone 9–11 (Ab. 9)	4177	10 (7/3)	0.3167
Abalone 11–29 (Ab. 11)	4177	10 (7/3)	0.3464
CoverType (CT)	286,048	12 (10/2)	0.0096
KDD99 (full) (KDD)	4,898,431	41 (32/8)	0.8000
KDD99 (10%) (KDD10)	494,021	41 (32/8)	0.8000
KDD99 (http) (KDDh)	623,091	40 (32/7)	0.0065
KDD99 (smtp) (KDDs)	96,554	40 (32/7)	0.0123
IDS	2,071,657	27 (8/19)	0.0333

generated by a different process than the rest. Note also that this dataset has been used extensively for anomaly detection [9,21,26,33,39,41]. For this research we transformed KDD99 into an anomaly detection dataset by assuming that attacks of any class are anomalies. To cope with the lack of labeled large datasets, as has been done in similar studies [33] three additional datasets were obtained by transforming the full KDDCup99. Thus, (1) KDDCup99 (10%) is the reduced dataset available at the UCI Repository, which contains only 10% of the instances, (2) KDDCup99 (http) is the result of filtering the full dataset to keep only http connections, and, analogously, (3) KDDCup99 (smtp) only contains smtp connections.

Finally, we used the IDS 2012 dataset [44], which covers the same domain as KDD99 but solving its weak points. Again, we consider any sort of attack as an anomaly, as opposed to normal traffic.

5.1.2. Synthetic dataset generator

In order to be able to exhaustively test the anomaly detection methods on datasets of diverse sizes and difficulty levels, we decided to create a synthetic dataset generator that could be parametrized. Previous works in the field, which have stated the need for such a publicly available generator to provide the data on which to perform experiments, have resorted to devising their own ad-hoc generators [29,34], a situation which complicates comparisons across studies. Our configurable generator (available for download at http://github.com/eirasf/ADMNC/) offers that capability to data science practitioners in that it can be used by researchers in the field to construct benchmarks.

The data generated by this method, while inspired by the data that would be created by a set of users interacting with a set of documents, was simplified to achieve a more general dataset. Each element of the dataset consists of a random binary vector, which symbolizes a bag-of-words representation of a document. Another binary vector and a numerical vector are generated from the existing random vector using a set of rules that account for statistics regarding the viewers of said document. Note that in a dataset designed this way, the numerical variables depend on the binary variables, which is the opposite assumption of our model that the categorical variables depend on the numerical variables. This design choice is intended to test the reliability of our model in detecting dependencies between the variables.

To obtain the dataset first we must choose the size of the vectors. The generator then creates two sets of random rules, one used to produce a binary vector and the other used to produce a numerical vector. Lastly, generated for the dataset are as many elements as requested by the user, each of which consists of a random binary vector together with the vectors resulting from the application of the mentioned sets of rules.

Table 3
Families of synthetic datasets used for the comparative study. *NV* represents the number of variables affected by each anomaly. In Synth1 and Synth2, the number of samples and NV vary, respectively, with the values $i \in |0, 5|$.

Dataset	# Samples	$\|u\|$	$\|b\|$	$\|n\|$	NV	Anomaly ratio
Synth1	$100 * 5^i$	20	10	100	4	0.5
Synth2	500	20	10	100	2^i	0.5

In the equations, the generated dataset D is described as a set of vectors over a set of indices

$$D = \{(\boldsymbol{u}_i, \boldsymbol{b}_i, \boldsymbol{n}_i) \mid i \in I\} \tag{7}$$

where $\boldsymbol{u}_i$ is a binary vector generated at random with uniform probability for each component and where the jth component in $\boldsymbol{b}_i$ is generated from $\boldsymbol{u}_i$ by a function f_j

$$\boldsymbol{b}_{ij} = f_j(\boldsymbol{u}_i) \tag{8}$$

which assigns 1 with a probability proportional to the fraction of conditions of the rule $\boldsymbol{r}_j$ satisfied by $\boldsymbol{u}_i$

$$f_j(\boldsymbol{x}) : \{0,1\}^n \to 0, 1 = a \mid a \sim Be\left(\frac{\langle \boldsymbol{r}_j, \boldsymbol{x}\rangle}{|\boldsymbol{r}_j|}\right) \tag{9}$$

where $Be(x)$ is a Bernouilli distribution with probability x. The jth component in $\boldsymbol{n_i}$ is sampled from a normal distribution whose mean and standard deviation are dictated by a function g_j

$$\boldsymbol{n}_{ij} \sim N\left(g_j((\boldsymbol{u}_i, \boldsymbol{b}_i)), \frac{1}{1 + g_j((\boldsymbol{u}_i, \boldsymbol{b}_i))}\right) \tag{10}$$

which simply indicates the fraction of conditions in rule $\boldsymbol{s}_j$ that the concatenation of $\boldsymbol{u}_i$ and $\boldsymbol{b}_i$ meets:

$$g_j(\boldsymbol{x}) : \{0,1\}^n \to \mathbb{R} = \frac{\langle \boldsymbol{s}_j, \boldsymbol{x}\rangle}{|\boldsymbol{s}_j|} \tag{11}$$

The rule sets R and S are randomly generated at the beginning of the generation process and kept constant for all elements. R must hold a vector $\boldsymbol{r}_j$ for each component in $\boldsymbol{b}$ and, analogously, S must contain as many rules as components are desired in $\boldsymbol{n}$. Each rule simply consists of a binary vector as long as $\boldsymbol{u}$ and $(\boldsymbol{u}_i, \boldsymbol{b}_i)$, respectively, indicating which components are affected by the rule.

After D is generated, a fraction of the elements are turned into anomalies by altering a number of its randomly selected components. Binary components are altered by flipping their value, while numerical components are incremented with a value randomly sampled from a standard normal distribution. When the dataset is constructed this way, the number of variables affected by an anomaly acts as a proxy for dataset difficulty: intuitively, the fewer components are altered by an anomaly, the more difficult it is to spot it.

With this methodology, we created two datasets for our experiments, as described on Table 3: in the first dataset we left the number of variables affected by an anomaly as a parameter, to study the effect of dataset difficulty on the algorithms; while in the second dataset we varied the number of elements to analyze the scalability of the different methods.

5.2. Results and discussion

The results obtained for the small-medium sized datasets are shown in Table 4. It is hard to draw a conclusion from this table regarding the best algorithm. In fact, using the Nemenyi post-hoc test [15] with $\alpha = 0.05$, the scores achieved by ADMNC cannot be significantly differentiated from those for any of the other algorithms; see Fig. 1. Consequently, expanded on the study of these algorithms with further experiments.

A few larger datasets were also used. Here the fact that LOCI and LOF are quadratic algorithms regarding the number of examples makes them computationally far more costly than the other algorithms, and thus unable to manage large datasets. LOF and LOCI were therefore excluded from this comparison. Therefore, ADMNC only had to compete with algorithms that make no distinction between categorical and numerical variables.

The results obtained with these larger datasets are shown in Table 5. To overcome computational difficulties, we performed these experiments using only 2 folds instead of 5. In addition, for all algorithms the best parameters for KDD and IDS were determined for their respective variants with only 10% of elements and those same values were used for all the variants. Four of the algorithms struggled with the two largest datasets: (1) the implementation of iForest in R could not handle the memory requirements of KDD or IDS, (2) PA-I took more than 10 hours to explore a single hyper-parameter combination with KDD, (3) OC-SVM (RBF) required quadratic memory space (in terms of number of samples), which made

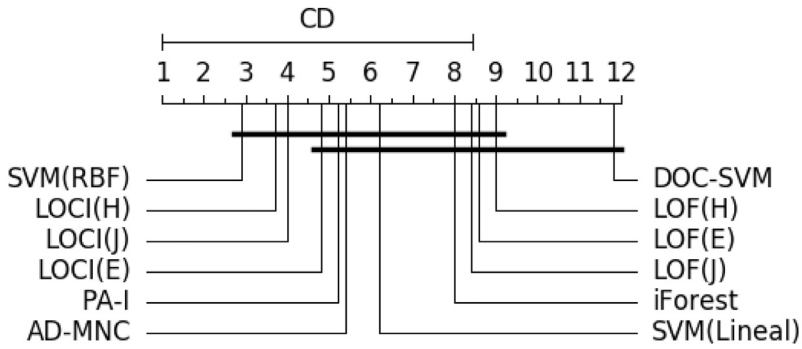

Fig. 1. Nemenyi test with the scores for Table 4. ADMNC is in the group of the best algorithms, although no algorithm is significantly better than the others.

Table 4
AUC of the proposed approach (ADMNC) compared with LOF and LOCI algorithms (using Euclidean (E), Hamming (H) and Jaccard (J) distances), OC-SVM using linear (SVM-L) and RBF (SVM-R) kernels, DOC-SVM, iForest and PA-I. The best results for each dataset are highlighted in boldface.

	Arrhyth	GC	Ab. 1	Ab. 9	Ab. 11
LOF (E)	0.6670	0.5847	0.6936	0.6029	0.5927
LOF (H)	0.6983	0.5646	0.6936	0.6029	0.5927
LOF (J)	0.7010	0.5681	0.6936	0.6029	0.5927
LOCI (E)	0.6735	0.5917	0.8524	0.6756	0.7155
LOCI (H)	0.7141	0.5709	**0.8526**	0.6856	0.7155
LOCI (J)	0.7144	0.5663	0.8512	**0.6874**	0.7159
SVM-L	0.6794	0.5697	0.7944	0.6140	0.7670
SVM-R	**0.7479**	**0.6452**	0.8121	0.6756	0.7448
DOC-SVM (RBF)	0.6530	0.5419	0.5561	0.5748	0.5502
iForest	0.7133	0.5792	0.6519	0.5966	0.5984
PA-I	0.6932	0.6216	0.8498	0.6511	0.7113
ADMNC	0.6140	0.6276	0.8453	0.6120	**0.7930**

Table 5
AUC of the proposed approach (ADMNC) compared with OC-SVM using linear (SVM-L) and RBF (SVM-R) kernels, iForest, and PA-I for large datasets. In all cases "-" indicates that results could not be obtained due to excessive time and/or memory requirements.

	CT	KDD10	KDD	KDDh	KDDs	IDS
SVM-L	0.9975	0.8712	0.8806	0.9139	0.9959	0.7300
SVM-R	0.9988	0.9965	–	0.9961	0.9859	–
iForest	0.9652	–	–	–	–	–
PA-I	**0.9989**	–	–	–	0.9903	–
ADMNC	0.9763	**0.9968**	**0.9975**	**0.9993**	**0.9972**	**0.9254**

handling the full KDD or IDS datasets impossible, and (4) even though the distributed nature of DOC-SVM allows it to process arbitrarily large datasets, the reliance on Java of its Matlab implementation meant it failed when trying to split a large dataset and, consequently, it could not process any of these datasets. With those methods unable to handle large datasets, OC-SVM with a linear kernel and ADMNC rendered results very favorable to our method. Additionally, the parallel implementation of ADMNC made handling large datasets much easier. It is worth noting that, for the largest datasets OC-SVM-L took several hours to compute, while ADMNC took just a few minutes. Even though they are implemented in different platforms, we illustrate this difference in scalability with our next experiment.

Since there was great disparity in the computational costs of the tested algorithms, we performed additional experiments to more thoroughly assess their scalability in terms of dataset size. To be able to adequately run this test and due to the aforementioned lack of real datasets with the necessary characteristics, we used the synthetic datasets described in Section 5.1.2 to allow us to control the size and difficulty of the dataset. The process used to generate these datasets is described in Section 5.1.2.

The results of testing the algorithms on the Synth1 family of datasets, shown in Fig. 2, show that our method is clearly superior in terms of scalability of the dataset size. Since the compared algorithms are implemented in diverse platforms and, therefore, their absolute times cannot be compared, times are presented as the ratio between the time taken to process 100 elements with that algorithm and the time taken for a given dataset size, which allows us to get an idea of

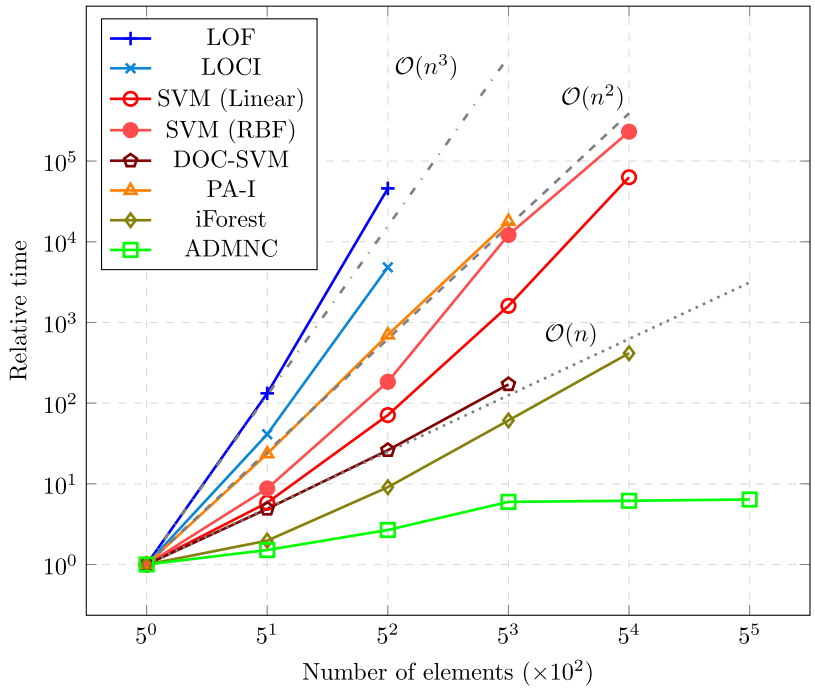

Fig. 2. Execution time for each algorithm on datasets of incrementing size (Synth1). Times are presented as a ratio of the time taken for a given execution and the time for the same algorithm on a 100-element dataset. Both axes are represented using a logarithmic scale.

the time complexity of each method. The time reported is the average execution time per fold for the algorithm using all the hyper-parameter combinations described in Table 1, except when the execution time was too high, when just one hyper-parameter combination was used and where the time therefore corresponds to a single execution. Even with this simplification, for some methods the absolute times were unmanageable for the largest versions of the dataset, so times could only be measured for the smaller versions. While the times of LOF and LOCI approach cubic complexity, PA-I displays quadratic complexity, OC-SVM exhibits super-linear complexity that approaches quadratic when the dataset is large, and DOC-SVM presents linear complexity, although its current implementation does not allow the use of large datasets. The time complexity of iForest approaches linear when the dataset is large. Our method exhibits clearly sub-linear complexity, which makes it the only candidate for very large datasets. Moreover, the ADMNC complexity increment is stopped when the dataset reaches 12,500 elements. This is because most of the complexity is due to the KMeans initialization step described in Section 3.1 which, once the dataset is large enough, works only with a fixed-size sample, therefore limiting the impact of dataset size on execution time. It is worth noting that, although the parallel implementation of our method potentially allows for additional speed-up using more computing cores, the scalability shown in these experiments does not stem from the addition of more computing cores. All experiments reported in this paper were executed using 12 computing cores on a single machine.

Finally, we compared the performance of each algorithm on Synth2, a family of datasets with an ascending order of difficulty. Since the three variants of LOF and LOCI offered very similar results, only the best performer for each method is reported. Results shown in Fig. 3 indicate that our method outperforms the rest of algorithms when the dataset is very complicated, whereas as the difficulty decreases the results even out. It is worth noting that the fact that the dataset is constructed with numerical variables depending on the binary variables is no obstacle to the performance of ADMNC, even though it models the probability the other way around, that is, the categorical variables depend on the numerical variables. It is also interesting to highlight that the methods with a comparable AUC to ADMNC (LOF, LOCI, SVM-L and SVM-R) all have time complexity $\mathcal{O}(n^2)$ or superior, which makes their results unavailable for large datasets. This leaves our method as the clearly superior option for those datasets.

6. Conclusions and future work

In this paper, we present a new scalable method for anomaly detection capable of handling large datasets and high dimensionality scenarios and of dealing with data having both categorical and continuous variables. It constitutes an useful tool for an emerging problem that currently lacks capable algorithms.

The approach presented uses a probabilistic perspective. The continuous part is modeled using a Gaussian mixture model, while the categorical part is estimated using a logistic model that uses a maximum likelihood approach optimized with a

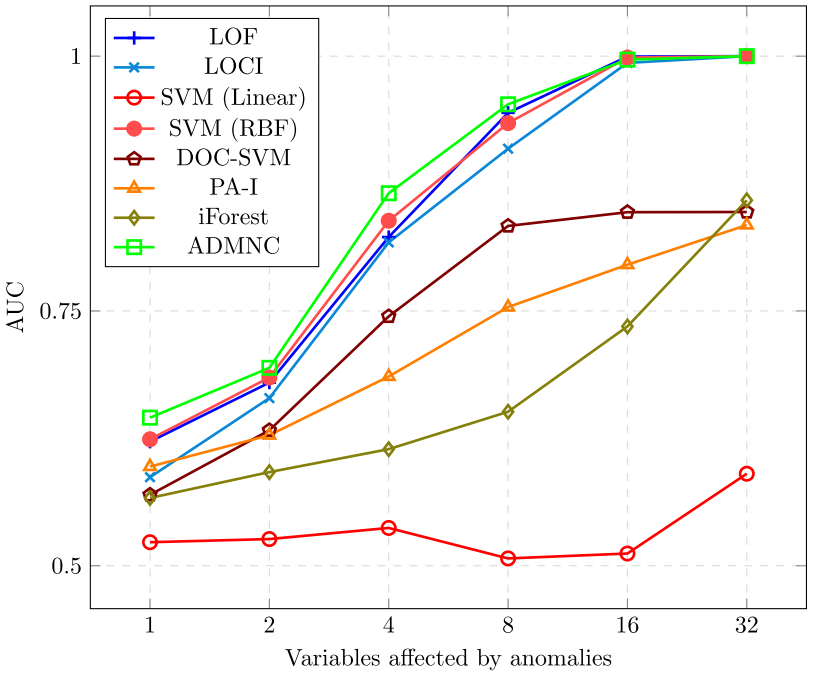

Fig. 3. Area under the ROC curve obtained on the Synth2 dataset. The number of variables affected by anomalies acts as a proxy for difficulty (a higher number indicates an easier dataset). The X axis is represented using a logarithmic scale.

stochastic gradient descent algorithm. The whole method is thus scalable to large datasets and further enhanced by parallel implementation.

Several experiments show that this method obtains better or similar results than those of state-of-the-art anomaly detection methods for small-medium datasets and that it performs well with datasets that are beyond the reach of other methods because of their computational demands. These favorable results would point to the viability of further research on the capabilities of this method as new datasets become available in the future. To facilitate further research we provide a working implementation of the algorithm in the popular Apache Spark framework.

In real world applications, and especially in the case of recent large datasets, the existence of missing data points is relatively common. Thus, and as future work, we plan to extend our probabilistic model to deal with these situations. We will also explore the interpretability of this model and the possibility of justifying each example labeled as an anomaly to users.

Acknowledgments

This research has been financially supported in part by the Spanish Ministerio de Economía y Competitividad (research projects TIN 2015-65069-C2, both 1-R and 2-R), by the Xunta de Galicia (Grants GRC2014/035 and ED431G/01) and the European Union Regional Development Funds.

Appendix: Best hyper-parameters

This is the list of the best hyper-parameter combination for each method for each dataset in the experiments reported in the paper. Hyper-parameters for Synth1 are not listed since they are not relevant for the execution time, which is the only measure reported for that dataset family.

Table 6
Best hyper-parameters for LOF.

	E (**K, P**)	H (**K, P, λ**)	J (**K, P, λ**)
Arrhyth	10, 0.01	10, 0.01, 0.9	10, 0.01, 0.7
GC	10, 0.01	10, 0.01, 0.3	10, 0.01, 0.1
Ab. 1	10, 0.01	10, 0.01, 0.3	10, 0.01, 0.3
Ab. 9	10, 0.01	10, 0.01, 0.3	10, 0.01, 0.3
Ab. 11	10, 0.01	10, 0.01, 0.3	10, 0.01, 0.3
Synth1-100	5, 0.01	3, 0.01, 0.9	5, 0.01, 0.5
Synth1-500	10, 0.01	10, 0.01, 0.9	10, 0.01, 0.7
Synth1-2500	Single test repeating values above		

Table 7
Best hyper-parameters for LOCI.

	LOCI		
	E (α)	H (α, λ)	J (α, λ)
Arrhyth	0.3	0.5, 0.9	0.3, 0.5
GC	0.3	0.1, 0.5	0.1, 0.1
Ab. 1	0.1	0.1, 0.7	0.1, 0.3
Ab. 9	0.3	0.1, 0.9	0.1, 0.7
Ab. 11	0.5	0.5, 0.7	0.5, 0.3
Synth1-100	0.3	0.1, 0.9	0.1, 0.9
Synth1-500	0.1	0.5, 0.9	0.3, 0.1
Synth1-2500	Single test repeating values above		

Table 8
Best hyper-parameters for SVM.

	Linear (ν)	RBF (γ, ν)	DOC-SVM(γ, ν)
Arrhyth	0.3	1, 0.1	1
GC	0.01	1, 0.01	3
Ab. 1	0.01	1, 0.3	5
Ab. 9	0.05	10, 0.1	1
Ab. 11	0.3	3, 0.05	1
CT	0.3	1, 0.3	4
KDD	0.1	10, 0.01	2
Synth1-100	0.3	0.01, 0.1	0.01, 0.3
Synth1-500	0.01	0.01, 0.01	0.01, 0.3
Synth1-2500	0.01	0.05, 0.01	Same values
Synth1-12500	Single test repeating values above		
Synth1-62500	Same values		*

Table 9
Best hyper-parameters for PA-I and iForest.

	PA-I			iForest	
	σ	C	R	rF	rS
Arrhyth	1	0.01	0.97	1	0.2
GC	3	0.01	0.97	1	1
Ab. 1	5	0.01	0.97	0.1	0.01
Ab. 9	1	0.05	0.97	1	0.5
Ab. 11	1	0.01	0.97	0.8	0.01
CT	4	0.025	0.97	1	0.01
KDD	2	0.05	0.97	0.1	0.5
Synth1-100	4	0.05	0.97	0.5	0.025
Synth1-500	4	0.01	0.97	0.8	0.01
Synth1-2500	4	0.01	0.97	1	0.01
Synth1-12500	4	0.01	0.97	1	0.025

Table 10
Best hyper-parameters for ADMNC.

	ν	λ_s	# gaussians
Arrhyth	1	1	4
GC	0.1	0.001	4
Ab. 1	100	0.01	4
Ab. 9	10	0.001	4
Ab. 11	0.1	0.001	4
CT	0.1	0.0001	4
KDD	1	0.1	2
IDS	1	0.1	4
Synth1-100	0.1	1	4
Synth1-500	1	1	4
Synth1-2500	10	1	4
Synth1-12500	10	0.001	4
Synth1-62500	100	0.1	4
Synth1-312500	1	0.1	4

Supplementary materials

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.ins.2019.03.013.

References

[1] L. Akoglu, H. Tong, J. Vreeken, C. Faloutsos, Fast and reliable anomaly detection in categorical data, in: Proceedings 21st ACM International Conference on Information and Knowledge Management, CKIM 2012, ACM, New York, NY, USA, 2012.
[2] E. Aleskerov, B. Freisleben, B. Rao, CARDWATCH: a neural network based database mining system for credit card fraud detection., in: Proceedings of the IEEE Conference on Computational Intelligence for financial engineering, 1997, pp. 220–226.
[3] B. Bahmani, B. Moseley, A. Vattani, R. Kumar, S. Vassilvitskii, Scalable k-means++, Proc. VLDB Endowment 5 (7) (2012) 622–633.
[4] C. Bishop, Pattern Recognition and Machine Learning (Information Science and Statistics), Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.
[5] J.A. Blackard, D.J. Dean, Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables, Comput. Electron. Agric. 24 (3) (1999) 131–151.
[6] L. Bottou, O. Bousquet, The tradeoffs of large scale learning, in: J. Platt, D. Koller, Y. Singer, S. Roweis (Eds.), Advances in Neural Information Processing Systems, 20, NIPS Foundation, 2008, pp. 161–168. (http://books.nips.cc).
[7] L. Bottou, C.-J. Lin, Support vector machine solvers, Large Scale Kernel Mach. 3 (1) (2007) 301–320.
[8] M. Breunig, H. Kriegel, R. Ng, J. Sander, Lof: identifying density-based local outliers, SIGMOD Rec. 29 (2) (2000) 93–104.
[9] G.O. Campos, A. Zimek, J. Sander, R.J. Campello, B. Micenková, E. Schubert, I. Assent, M.E. Houle, On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study, Data Min. Knowl. Discov. 30 (4) (2016) 891–927.
[10] E. Castillo, D. Peteiro-Barral, B.G. Berdiñas, O. Fontenla-Romero, Distributed one-class support vector machine, Int. J. Neural Syst. 25 (07) (2015) 1550029.
[11] V. Chandola, A. Banerjee, V. Kumar, Anomaly detection: a survey, ACM Comput.Surv. (CSUR) 41 (3) (2009) 15.
[12] V. Chandola, A. Banerjee, V. Kumar, Anomaly detection for discrete sequences: a survey, IEEE Trans. Knowl. Data Eng. 24 (5) (2012) 823–839.
[13] K. Das, J. Schneider, Detecting anomalous records in categorical datasets, in: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, in: KDD '07, ACM, New York, NY, USA, 2007.
[14] S. Das, B.L. Matthews, A.N. Srivastava, N. Oza, Multiple kernel learning for heterogeneous anomaly detection: algorithm and aviation safety case study, in: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, in: KDD '10, ACM, New York, NY, USA, 2010, pp. 47–56.
[15] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (January) (2006) 1–30.
[16] K. Do, T. Tran, S. Venkatesh, Energy-based anomaly detection for mixed data, Knowl. Inf. Syst. (2018) 1–23.
[17] F. Edgeworth, On discordant observations, Phylos. Mag. 23 (5) (1887) 364–375.
[18] A.F. Emmott, S. Das, T. Dietterich, A. Fern, W.-K. Wong, Systematic construction of anomaly detection benchmarks from real data, in: Proceedings of the ACM SIGKDD Workshop on Outlier Detection and Description, ACM, 2013, pp. 16–21.
[19] D. Fernández-Francos, D. Martínez-Rego, O.Fontenla-Romero, A. Alonso-Betanzos, Automatic bearing fault diagnosis based on one-class nu-svm, Comput. Ind. Eng. 64 (1) (2013) 357–365.
[20] U. Fiore, F. Palmieri, A. Castiglione, A. De Santis, Network anomaly detection with the restricted boltzmann machine, Neurocomputing 122 (2013) 13–23.
[21] A. Ghoting, M. Otey, S. Parthasarathy, Loaded: link-based outlier and anomaly detection in evolving data sets, in: Data Mining, 2004. ICDM '04. Fourth IEEE International Conference on, 2004, pp. 387–390.
[22] D.M. Hawkins, Identification of Outliers, 11, Springer, 1980.
[23] S. Hawkins, H. He, G. Williams, R. Baxter, Outlier detection using replicator neural networks, in: International Conference on Data Warehousing and Knowledge Discovery, Springer, 2002, pp. 170–180.
[24] Z. He, S. Deng, X. Xu, An optimization model for outlier detection in categorical data, in: D. Huang, G. X.P. Zhang, Huang (Eds.), Advances in Intelligent Computing, Lecture Notes in Computer Science, 3644, Springer Berlin Heidelberg, 2005, pp. 400–409.
[25] S. Hettich, S. Bay, KDD Cup 1999 Data, The UCI KD Archive, Irvine, CA: University of California, Department of Information and Computer Science, 1999.
[26] W. Hu, W. Hu, S. Maybank, Adaboost-based algorithm for network intrusion detection, IEEE Trans. Syst. Man Cybern.Part B (Cybernetics) 38 (2) (2008) 577–583.
[27] Y. JeffreyXu, W. Qian, L. Hongjun, Z. Aoying, Finding centric local outliers in categorical/numerical spaces, Knowl. Inf. Syst. 9 (3) (2006) 309–338.
[28] S.S. Khan, M.G. Madden, One-class classification: taxonomy of study and review of techniques, Knowl. Eng. Rev. 29 (03) (2014) 345–374.
[29] A. Koufakou, M. Georgiopoulos, A fast outlier detection strategy for distributed high-dimensional data sets with mixed attributes, Data Min. Knowl. Discov. 20 (2) (2010) 259–289.
[30] V. Kumar, Parallel and distributed computing for cybersecurity., IEEE Distrib. Syst. Online 6 (10) (2005) 1–10.
[31] A. Lazarevic, V. Kumar, Feature bagging for outlier detection, in: Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining, ACM, 2005, pp. 157–166.
[32] M. Lichman, UCI machine learning repository, 2013. http://archive.ics.uci.edu/ml [Last accessed April 2017].
[33] F.T. Liu, K.M. Ting, Z.-H. Zhou, Isolation-based anomaly detection, ACM Trans. Knowl. Discov.Data (TKDD) 6 (1) (2012) 3.
[34] Y.-C. Lu, F. Chen, Y. Wang, C.-T. Lu, Discovering anomalies on mixed-type data using a generalized student-*t* based approach, IEEE Trans. Knowl. Data Eng. 28 (10) (2016) 2582–2595.
[35] D. Martinez-Rego, E. Castillo, O. Fontenla-Romero, A. Alonso-Betanzos, A minimum volume covering approach with a set of ellipsoids, Pattern Anal. Mach. Intell. IEEE Trans. 35 (12) (2013) 2997–3009.
[36] D. Martínez-Rego, D. Fernández-Francos, O.Fontenla-Romero, A. Alonso-Betanzos, Stream change detection via passive-aggressive classification and bernoulli CUSUM, Inf. Sci. 305 (2015) 130–145.
[37] H. Moonesignhe, P.-N. Tan, Outlier detection using random walks, in: Null, IEEE, 2006, pp. 532–539.
[38] M. Nicolau, J. McDermott, et al., Learning neural representations for network anomaly detection, IEEE Trans. Cybern. (2018) 1–14. 99
[39] M. Otey, A. Ghoting, S. Parthasarathy, Fast distributed outlier detection in mixed-attribute data sets, Data Min. Knowl. Discov. 12 (2–3) (2006) 203–228.
[40] S. Papadimitrou, H. Kitagawa, P. Gibbons, C. Faloutsos, LOCI: Fast Outlier Detection using the LOcal Correlation Integral, Technical report IRP-TR-02-09, Intel Research Laboratory, 2002.
[41] S.T. Sarasamma, Q.A. Zhu, J. Huff, Hierarchical kohonenen net for anomaly detection in network security, IEEE Trans. Syst. Man Cybern.Part B (Cybernetics) 35 (2) (2005) 302–312.
[42] B. Scholkopf, J. Platt, J. Shawe-Taylor, A. Smola, R. Williamson, Estimating the support of a high-dimensional distribution, Neural Comput. 13 (7) (2001) 1443–1471.
[43] E. Schubert, A. Zimek, H. Kriegel, Local outlier detection reconsidered: a generalized view on locality with applications to spatial, video, and network outlier detection, Data Min. Knowl. Discov. (2012) 1–48.
[44] A. Shiravi, H. Shiravi, M. Tavallaee, A.A. Ghorbani, Toward developing a systematic approach to generate benchmark datasets for intrusion detection, Comput.Secur. 31 (3) (2012) 357–374.

[45] S. Singh, H. Tu, W. Donat, K. Pattipati, P. Willett, Anomaly detection via feature-aided tracking and hidden markov models, IEEE Trans. Syst. ManCybern.-Part A 39 (1) (2009) 144–159.
[46] A. Sodemann, M. Ross, B. Borghetti, A review of anomaly detection in automated surveillance, IEEE Trans. Syst. Man Cybern.Part C 42 (6) (2012) 1257–1272, doi:10.1109/TSMCC.2012.2215319.
[47] S.Wu, S. Wang, Parameter-free anomaly detection for categorical data., in: Proceedings of the 7th International Conference on Machine Learning and Data Mining, MLDM 2011. Lecture notes in Computer Science, 6871, 2011, pp. 112–126.
[48] L. Wei, W. Qian, A. Zhou, W. Jin, J. Yu, Hot: hypergraph-based outlier test for categorical data, in: K. Whang, J. Jongwoo, K. Shim, J. Srivastava (Eds.), Advances in Knowledge Discovery and Data Mining, Lecture Notes in Computer Science, 2637, Springer Berlin Heidelberg, 2003, pp. 399–410.
[49] S. Wu, S. Wang, Information-theoretic outlier detection for large-scale categorical data, IEEE Trans. Knowl. Data Eng. 25 (3) (2013) 589–602.
[50] K. Zhang, H. Jin, An effective pattern based outlier detection approach for mixed attribute data, in: Australasian Joint Conference on Artificial Intelligence, Springer, 2010, pp. 122–131.