Large Scale Anomaly Detection in Mixed Numerical and Categorical Input Spaces

Carlos Eiras-Franco, David Martínez-Rego, Bertha Guijarro-Berdiñas, Amparo Alonso-Betanzos, and Antonio Bahamonde

***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 to track low likelihood values as anomalies. The main contribution of this method is that, to cope with the different nature of variables, we factorize the joint probability measure into two parts: the marginal density of the continuous variables and the conditional probability of the categorical variables given the continuous part of the feature vector, resulting in a model trained through a maximum likelihood objective function optimized with Stochastic Gradient Descent. This results in an effective and scalable algorithm. The comparison with other well-known anomaly detection algorithms over several datasets shows that ADMNC offers top level accuracy even in datasets that are out of reach for the most effective existing methods, and scales well to process very large datasets, which transforms it in a powerful tool for a problem growing in popularity that currently lacks suitable algorithms.**

***Index Terms*—Anomaly detection, Outlier detection, Logistic Regression, Gaussian Mixture Model, Machine Learning, Synthetic dataset generator.**

I. 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”[24]. Detecting anomalies is an old discipline for statisticians [18], under the name *outlier detection*. From those days on, these type of methods are becoming increasingly important. Anomaly detection is especially useful in practical situations where on the one hand the dataset is numerous and, on the other hand, it contains unexpected events that carry the most important information. Several challenges stand in the way of obtaining a general technique for anomaly detection: the increasing amount of domains in which this discipline has encountered application (detection of intrusions [30], surveillance [44], frauds [3], machine faults [21, 16, 20, 35]) adds high variability to the proposed solutions, while the scarcity of labeled data from real-world processes [11, 13] 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 [13, 19, 43]. 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 for a prior shape for this region through the fix of a specific distribution or geometric shape is usually a difficult task that can introduce a bias that prevents us from generating a meaningful model.

Another major difficulty, in which we focus on in this work, is the type of input data. The introduction of mixed numerical/categorical data can make the modeling of correlations between input variables a complicated issue. This has caused that many of the popular anomaly detection techniques: (a) can only deal with categorical or numerical data [45, 15, 2], (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 [23, 37].

Furthermore, the production of data has dramatically increased in recent years, at a much faster pace than computational power. This unbalance renders some of the most popular algorithms unable to deal with the amounts of data that users need to analyze. Algorithms with a quadratic spatial or time complexity are no longer suitable for this kind of analysis. This limitation has brought the focus towards the scalability of algorithms, spawning an active field known as big data learning [9, 31]. A successful approach so far 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 at exploring a strategy to model anomaly detection problems in which the data is numerous and contains 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 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 it 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 works towards 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 will allow the model to scale to large datasets, both in feature and sample sizes, making it an appealing option for Big Data applications. To demonstrate this, an implementation of the algorithm (from

Carlos Eiras-Franco, David Martínez-Rego, Bertha Guijarro-Berdiñas and Amparo Alonso-Betanzos work at the Research Center on Information and Communication Technologies (CITIC), Universidade da Coruña, 15071 A Coruña, Spain, e-mail: carlos.eiras.franco@udc.es

Antonio Bahamonde works at the Centro de Inteligencia Artificial, Universidad de Oviedo, 33204 Gijón, Spain

now on called Anomaly Detector for Mixed Numerical and Categorical inputs, ADMNC) in the popular cluster computing framework Apache Spark [1] is provided.[1]

Section II reviews the related work in this area. Section III presents the formal framework and Section IV introduces the parametric formulation of the problem. Section V reports a collection of experiments that show the properties of the proposed method on real datasets, as well as the definition of a synthetic dataset generator and further experiments with the resulting datasets. Section VI summarizes the main conclusions and future work.

II. 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 can 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 [29].

Anomaly detection approaches can be classified according to the nature of 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 [14]. Some anomaly detection models can only deal with categorical data [45, 15, 2]. The case of numeric variables has been treated mainly through statistical parametric and non-parametric models [40, 5], geometrical approximations [35] and using binary trees [34]. 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 [47], in which the set of outliers in a dataset is detected using a specially devised data structure called hypergraph and a local test for outliers based on a frequent itemset counting strategy. Another approach is presented in [25], where the problem is tackled using information theory concepts but, again, only categorical attributes are considered and numerical attributes need to be circumvented through discretization.
- *Metric-centered* techniques: This kind of methods define an anomaly as a point which lies in a low density region when compared to its neighborhood. They rely on a function that calculates the similarity between elements in the input space and so it can be extended to the mixed numerical/categorical case and other types of structured data [41] through a tailored similarity function. Local Outlier Factor (LOF) [10] can be considered the seminal work in this area. The basic criteria of these methods has also inspired subsequent improvements for high dimensional spaces [32] and improved density criteria such as [28]. LOCI (LOcal Correlation Integral) is a similar technique, improving LOF as it is able to detect outliers and also groups of outliers without user-required cut-offs [38]. These techniques present the following challenges: (a) devising effective similarity measures for mixed numerical/categorical input spaces and (b) scalability, since the similarity matrix needs to be computed before getting into the detection phase.
- *Mixed-criteria* techniques: This group of algorithms tackles the nature of numerical and categorical data separately, trying to design a criterion which encompasses the analysis of an element in both spaces. In this group we can classify LOADED [23, 37]. This algorithm blends in a single criterion categorical-categorical, categorical-numerical and numerical-numerical correlations using frequent itemsets concepts and local correlation matrices. Despite being the first attempt to handle categorical attributes that previous approaches ignore, the algorithm suffers for high execution times in high dimensional datasets, because although execution times scale linearly with the number of data points, it scales quadratically with the number of numerical attributes, and grows even more computationally expensive with the number of categorical attributes. Additionally, adaptations of supervised learning techniques like AdaBoost to this problem have also been presented [27] although its requirement of labeled samples prevents its use in the most common use cases.

In this research we focused on devising a probabilistic strategy able to solve anomaly detection problems where input elements pertain to an input space which mixes numerical and categorical variables. The proposed algorithm closely relates 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 should be able to tackle both large scale and high dimensionality problems.

III. Basic formulation

We aim at adjusting a probability measure function that best fit the data under normal conditions. Subsequently, when monitoring new data elements they are assigned a score and those whose score exceeds a pre-specified threshold would be considered as anomalies. Formally, the following expression has to be estimated:

$$P(x_1, x_2, x_3, \ldots, x_n) \qquad (1)$$

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

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

However, in many situations datasets present a mixture of categorical and numerical variables. In such a case, the model should take into account individually the nature of both types of variables. In this work, we heuristically propose the following factorization of the pdf under normal conditions:

$$P(\boldsymbol{y}|\boldsymbol{x})P(\boldsymbol{x}) \tag{2}$$

where $\boldsymbol{y}$ represents the categorical variables, while $\boldsymbol{x}$ stands for the set of numerical variables. With such a partition of the pdf, we can adopt a convenient technique for the estimation of the parameters of each part independently, while accounting for eventual interactions between the two parts. Below, on Section V we assess the adequacy of this approach with experiments on several datasets.

It was previously mentioned that an incorrect assumption about the shape of the underlying distribution could induce a bias that could harm the accuracy of the model. Nevertheless, we had to make such a decision, so we tried to adopt the most flexible model still computable in a closed form. We adopted a flexible parametric approach for both the conditioned probability of the categorical variables and the marginal probability of the set of numerical variables. Namely, in this work we used the following models for each part:

A. Numerical part

Mixture models are the most flexible parametric option for estimating this marginal. 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 Gaussian Mixture Models 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 perform a KMeans clustering on a small sample of the dataset. We then use the resulting centroids as the initial values for these means and we compute the empirical standard deviations of the clusters to obtain the initial diagonal covariance matrix. For the KMeans algorithm we use the default implementation included in Spark [4].

B. Categorical part

A binary translation was carried out for each of the N categorical variables. Then, $P(\boldsymbol{y}|\boldsymbol{x})$ was approximated by a Logistic Regression (LR) model [46]. From now on, $\boldsymbol{y}$ will represent the categorical binarized part of the input vector.

Namely, the LR model computes the conditional probability by means of

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

where $\Phi(\boldsymbol{x},\boldsymbol{y})$ is the so-called feature vector, which has the following form

$$\Phi(\boldsymbol{x},\boldsymbol{y}) = (\boldsymbol{x}\otimes\boldsymbol{y},\boldsymbol{y}\otimes\boldsymbol{y}) \tag{4}$$

where $\boldsymbol{x}$ represents again the numerical part of the elements (with an additional bias component, equal to 1), $\boldsymbol{y}$ is the transformation of the categorical variables to m indicator variables with a bias too, and $\otimes$ is the Kronecker product.

This feature function is inspired by works on probabilistic multilabel classification [22]. It should be noted that, by using this encoding, we account for: (a) *a priori* probability of each value of the categorical variables, (b) correlation of each categorical variable with the numerical part and (c) pairwise correlation between categorical variables.

It should be noted that the number of parameters $\boldsymbol{w}$ required is the same as the number of elements in Φ, which grows quadratically with the length of input vectors, which in turn can amount to a tendency to overfit the training examples and can slow down training.

To reduce the number of parameters we used a factorization approach. In symbols,

$$\begin{aligned}\Psi(\boldsymbol{x},\boldsymbol{y},\theta) &= -\langle \boldsymbol{w},\Phi(\boldsymbol{x},\boldsymbol{y})\rangle \\ &= -\left(\langle W_{xy}\boldsymbol{x},V_{xy}\boldsymbol{y}\rangle + \langle W_{yy}\boldsymbol{y},V_{yy}\boldsymbol{y}\rangle\right)\end{aligned} \tag{5}$$

where $\boldsymbol{\theta}$ is the set of parameter matrices W_{xy}, V_{xy}, W_{yy} and V_{yy} that map $\boldsymbol{x}$ and $\boldsymbol{y}$ to a k-dimensional subspace where their inner product yields the appropriate factor. The dimensions of the resulting matrices (namely $k\times|x|$, $|y|\times k$, $k\times|y|$ and $|y|\times k$) amount to a total of $k\times(|x|+3|y|)$ parameters to be learned which is significantly less than the original $|x|\times|y|+|y|\times|y|$ unless for large values of k.

Thus, in the following, we use, instead of Eq.3:

$$P(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{\theta}) = \frac{1}{1+e^{\Psi(\boldsymbol{x},\boldsymbol{y},\theta)}} \tag{6}$$

Thus, the learning process reduces 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 search for the optimal parameters.

IV. Maximum likelihood parameter estimation

Let D be a dataset of points $(\boldsymbol{x},\boldsymbol{y})$. Then, taking into account the expression (Eq. 2) for the factorization of the joint pdf, the data log-likelihood has the following form:

$$\log L(D) = \sum_{i=1}^{|D|}\log P(\boldsymbol{y}_i|\boldsymbol{x}_i,\boldsymbol{\theta}) + \sum_{i=1}^{|D|}\log P(\boldsymbol{x_i}) \tag{7}$$

It is important to note that the first part of the data log-likelihood is completely independent of the parameters of the second summand, so both optimization processes can be run in parallel, and exploit the structure of each problem separately. In the next sections we show how this can be achieved separately for each part and then combined in a unified algorithm.

Using the first term of (Eq. 7), we aim to obtain the parameters $\boldsymbol{\theta}^*$ that maximize

$$\boldsymbol{\theta}^* \quad = \quad \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\left\{\sum_{i=1}^{|D|}\log P(\boldsymbol{y}_i|\boldsymbol{x}_i,\boldsymbol{\theta}) - \nu\frac{\boldsymbol{\theta}^2}{2}\right\} \tag{8}$$

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

This is a well-known convex optimization problem that can be solved using an Stochastic Gradient Descent (SGD) algorithm [7].

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

$$\lambda_t = \frac{\lambda_0}{1 + \lambda_s(t-1)}. \tag{9}$$

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.

Additionally, to keep the values of the component of the parameters $\boldsymbol{\theta}$ bounded, after each updating, we ensure that every column of each matrix is in a ball with center the origin and radius B.

The implementation in Apache Spark parallelizes the mini-batch step in the SGD process, potentially accelerating the whole process if there are several computational units available.

V. Experimental settings and results

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

A. Methodology

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

Using this procedure we account for the fact that in real situations we typically do not have anomalous samples to train the 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. With this transformation, the obtained anomalies meet the definition presented in the Introduction.

To test the strengths of our algorithm, we compared the scores with those achieved by the following state-for-the-art algorithms. First, we considered two algorithms that make a differenced 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 SVM (OC-SVM) (with 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 [8], which makes them poor candidates for handling large amounts of data. Therefore, we also tested DOC-SVM [12], which is a distributed version of the same algorithm that can handle large datasets by splitting them, also implemented in Matlab[4]. Finally, we added to the test the recent iForest [34], 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 have typically 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-parameters values. For each fold, the algorithm is trained with the non-anomalous examples of the training set and evaluated on all the samples of the test set. The hyper-parameters explored are listed in Table I. The scores discussed in subsection V-B are the best average AUC obtained in the CV procedure.

TABLE I
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	$\nu \in \{0.1, 1, 10, 100, 1000\}$, $\lambda_s \in \{0.0001, 0.001, 0.01, 0.1, 1\}$ Number of gaussians $\in$ {2, 4}

[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 One-Class SVM that can use non-linear kernels with time complexity inferior to $\mathcal{O}(n^2)$.

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

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

In the following, we describe the datasets employed in the experiments:

1) Real datasets: As stated above, it is very difficult to come by real-world datasets with labeled anomalies. Thus, the datasets used are commonly employed for classification tasks, but they have been repurposed for anomaly detection. They were downloaded from the UCI Machine Learning Repository [33].

The description of datasets is reported in Table II. We distinguish two groups. In the first one there are small-medium size datasets: Arrhythmia (Arrhyth), German Credit (GC), and 3 versions of Abalone. These versions were built choosing different classes as anomalous and non-anomalous. So, Abalone 1-8 (Ab. 1), Abalone 9-11 (Ab. 9) and Abalone 11-29 (Ab. 11) were obtained by using, respectively, classes 1, 9 and 11 as non-anomalous and classes 8, 11 and 29 as anomalous.

TABLE II

REAL DATASETS USED FOR THE COMPARATIVE STUDY. THE *Anomaly ratio* IS THE QUOTIENT OF ANOMALOUS EXAMPLES OVER THE NUMBER OF EXAMPLES. IN PARENTHESIS, THE NUMBER OF NUMERICAL / CATEGORICAL FEATURES.

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)	286048	12 (10/2)	0.0096
KDD99 (full) (KDD)	4898431	41 (32/8)	0.8000
KDD99 (10%) (KDD10)	494021	41 (32/8)	0.8000
KDD99 (http) (KDDh)	623091	40 (32/7)	0.0065
KDD99 (smtp) (KDDs)	96554	40 (32/7)	0.0123
IDS	2071657	27 (8/19)	0.0333

The second group of datasets has a larger number of samples. They are versions of CoverType [6] and KDD99 [26] datasets. To transform CoverType (CT), instances of class 2 were assumed as 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 about it not accurately representing an intrusion detection task, those discrepancies have no effect in 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, what we pursue is identifying a minority set of instances that were generated by a different process than the rest. Besides, let us recall that this dataset has been used extensively for anomaly detection [11, 23, 37, 34, 27, 39]. For this work 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 done in similar studies [34], three additional datasets were obtained by transforming the full KDDCup99: (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; analogously (3) KDDCup99 (smtp) only contains smtp connections.

Finally, we used the IDS 2012 dataset [42], 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.

2) Synthetic dataset generator: In order to be able to exhaustively test the anomaly detection methods on data with diverse sizes and difficulty levels, we decided to create a synthetic dataset generator that could be parametrized. The data generated by this method was inspired on the data that would be created by a set of users interacting with a set of documents, although it was simplified to achieve a more general dataset. Therefore, each element of the dataset consists of a random binary vector, which symbolizes a bag-of-words representation of a document, and then another binary vector and a numerical vector which are generated from the existing random vector using a set of rules, which account for statistics regarding the viewers of said document. Note that in a dataset designed this way, the numerical variables depend on the binary ones, which is the opposite to our model in which the categorical variables are assumed to depend on the numerical ones. This is a design choice 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. Then, the generator creates two sets of random rules, one that will be used to produce a binary vector and the other to produce a numerical vector. Lastly, as many elements as requested by the user are generated for the dataset, each of which consists of a random binary vector together with the vectors resulting from the application of the mentioned sets of rules.

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

$$D = \{(\boldsymbol{u}_i, \boldsymbol{b}_i, \boldsymbol{n}_i) \mid i \in I\} \tag{10}$$

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

$$\boldsymbol{b}_{ij} = f_j(\boldsymbol{u}_i) \tag{11}$$

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

$$f_j(\boldsymbol{x}) : \{0,1\}^n \to 0, 1 = a \mid a \sim Be\left(\frac{\langle \boldsymbol{r}_j, \boldsymbol{x}\rangle}{|\boldsymbol{r}_j|}\right) \tag{12}$$

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

$$\boldsymbol{n}_{ij} \sim N\left(g_j((\boldsymbol{u}_i, \boldsymbol{b}_i)), \frac{1}{1 + g_j((\boldsymbol{u}_i, \boldsymbol{b}_i))}\right) \tag{13}$$

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

$$g_j(\boldsymbol{x}) : \{0,1\}^n \to \mathbb{R} = \frac{\langle \boldsymbol{s}_j, \boldsymbol{x}\rangle}{|\boldsymbol{s}_j|} \tag{14}$$

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 in 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 components randomly selected. 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 the 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 sets of datasets for our experiments, as described on Table III: on 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 on the second dataset we vary the number of elements to analyze the scalability of the different methods.

TABLE III
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	$\lvert u\rvert$	$\lvert b\rvert$	$\lvert n\rvert$	NV	Anomaly ratio
Synth1	$100 * 5^i$	20	10	100	4	0.5
Synth2	500	20	10	100	2^i	0.5

B. Results and discussion

The results obtained for the small-medium size datasets are shown in Table IV. It is hard to draw a conclusion from this table about which is the best algorithm. In fact, using the Nemenyi post-hoc test [17] with $\alpha = 0.05$ the scores achieved by ADMNC can not be significantly differentiated from any of the other algorithms; see Figure 1. Consequently, we set to expand on the study of these algorithms with further experiments.

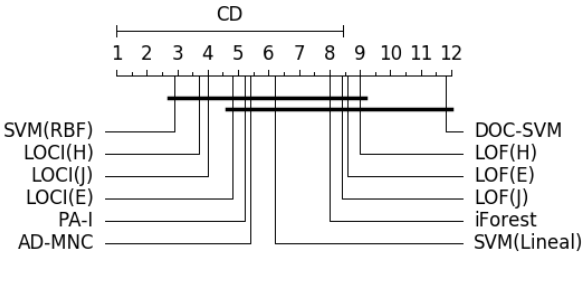

Fig. 1. Nemenyi test with the scores of Table IV. ADMNC is in the group of the best algorithms, although none is significantly better than the others

TABLE IV
Area under the ROC curve of the proposed approach (ADMNC) against 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**

A few larger datasets were also used. Here we have to realize that LOCI and LOF are quadratic algorithms regarding the number of examples. This makes them computationally very expensive, considerably more than the rest, and thus unable to manage large datasets. For this reason, LOF and LOCI were 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 V. 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 on their respective variants with only 10% of elements and those same values were used for all its variants. In any case, 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) requires quadratic (in terms of number of samples) memory space, which made 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 makes it fail when trying to split a large dataset and, consequently, it could not process any of these datasets. With these 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 Linear took several hours to compute, while ADMNC took just a few minutes. Even though they are implemented in different platforms, we set to illustrate this difference in scalability with our next experiment.

Since there is 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 test this and due to the

TABLE V

Area under the ROC curve of the proposed approach (ADMNC) against 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**

aforementioned lack of real datasets with the characteristics that we need, we used the synthetic datasets described in section V-A2 to allow us to control the size and difficulty of the dataset. The process used to generate these datasets is described in Section V-A2.

The results of testing the algorithms on the Synth1 family of datasets are shown on Figure 2 and 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 can not be compared, times are presented as the ratio between the time taken for processing 100 elements with that algorithm and the time taken for a given dataset size, which allows us to get an idea of the time complexity of each method. The time reported is the average execution time per fold for the algorithm using all the hyperparameter combinations described in Table I, except in cases when the execution time was too high, where just one hyperparameter combination was used and therefore the time 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 they 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 superlinear 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 approches linear when the dataset is large. Our method exhibits clearly sublinear complexity, which makes it the only candidate when dealing with very large datasets. Moreover, the complexity increment of ADMNC stops when the dataset reaches 12500 elements, since most of the complexity is due to the KMeans initialization step described in Section III-A which, once the dataset is large enough, works only with a fixed-size sample, therefore limiting the impact of the dataset size in the execution time. It is worth noting that, although the parallel implementation of our method potentially allows for additional speed-ups by 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.

We finally compared the performance of each algorithm on Synth2, a family of datasets in ascending order of difficulty.

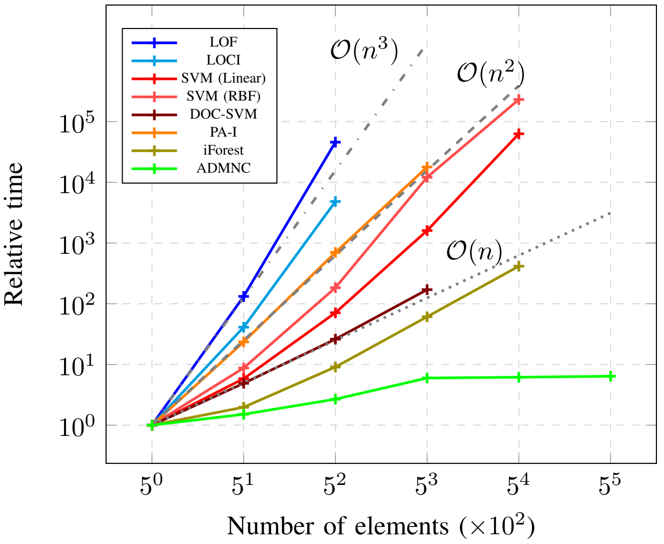

Fig. 2. Execution time for each algorithm on datasets of incrementing size (Synth1). Times are presented as a ratio of the time taken for a given execution and the time for the same algorithm on a 100 element dataset. Both axis are represented using a logarithmic scale.

Since the three variants of LOF and LOCI offered very similar results, only the best performer for each method is reported. Results shown in Figure 3 indicate that our method outperforms the rest of algorithms when the dataset is very complicated and 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 ones is no obstacle to the performance of ADMNC, even though it models the probability the other way around: the categorical variables depend on the numerical ones. It is also interesting to highlight that the methods that offer comparable AUROC to ADMNC (LOF, LOCI, SVM-L and SVM-R) all have time complexity $\mathcal{O}(n^2)$ or superior, which in turn makes their results unavailable for large datasets, leaving our method as the clearly superior option for those datasets.

VI. Conclusions and future work

In this paper, we present a new method for anomaly detection able to confront large datasets and high dimensionality scenarios and with the capability of dealing with data having both categorical and continuous variables. It constitutes an useful tool for an emerging problem that is currently lacking algorithms that can tackle it.

The approach presented uses a probabilistic perspective. The continuous part is modeled using a Gaussian Mixture while the categorical part is estimated using a Logistic Model that uses Maximum Likelihood approach optimized with an Stochastic Gradient Descent algorithm. Thus the whole method is scalable to large datasets, which is furthered by the parallel implementation provided.

Several experiments showed that this method obtains better or similar results than those of state-of-the-art anomaly detection methods for small-medium datasets and is able to obtain very good performance in datasets that are out of reach for other methods due to their computational demands. These

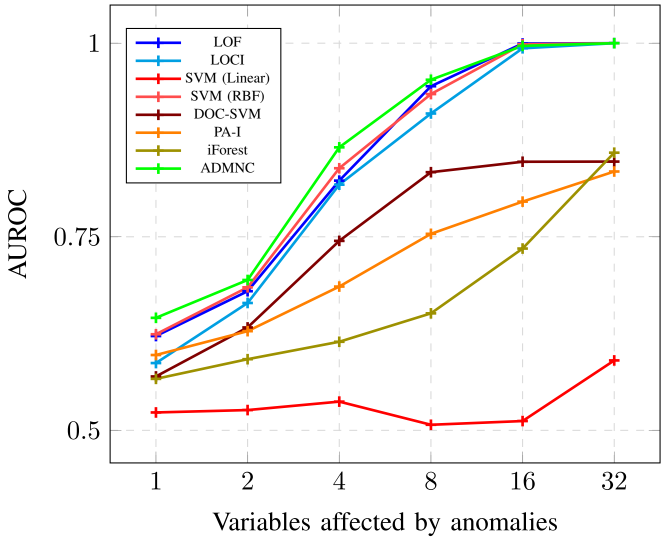

Fig. 3. Area under the ROC curve obtained on the Synth2 dataset. The number of variables affected by anomalies acts as a proxy for difficulty, a higher number indicates an easier dataset. X axis is represented using a logarithmic scale.

favorable results encourage further research on the capabilities of this method as new datasets become available in the future. To this avail we provide a working implementation of the algorithm in the popular framework Apache Spark.

In real world applications, and specially in the case of recent large datasets, the existence of missing data points can be relatively common. Thus, and as future work, we plan to extend our probabilistic model to be capable of confronting these situations. Also, we would like to explore in future works the interpretability of this model and the possibility of giving a justification to the user for each example labeled as an anomaly.

Acknowledgements

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.

References

[1] Apache Spark: Lightning-fast cluster computing. https://spark.apache.org/. Accessed: 2016-12-16.

[2] Akoglu, L., Tong, H., Vreeken, J., and Faloutsos, C. (2012). Fast and reliable anomaly detection in categorical data. In *Proceedings 21st ACM International Conference on Information and Knowledge Management, CKIM 2012*, New York, NY, USA. ACM.

[3] Aleskerov, E., Freisleben, B., and Rao, B. (1997). 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*, pages 220–226.

[4] Bahmani, B., Moseley, B., Vattani, A., Kumar, R., and Vassilvitskii, S. (2012). Scalable k-means++. *Proceedings of the VLDB Endowment*, 5(7):622–633.

[5] Bishop, C. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

[6] Blackard, J. A. and Dean, D. J. (1999). Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables. *Computers and electronics in agriculture*, 24(3):131–151.

[7] Bottou, L. and Bousquet, O. (2008). The tradeoffs of large scale learning. In Platt, J., Koller, D., Singer, Y., and Roweis, S., editors, *Advances in Neural Information Processing Systems*, volume 20, pages 161–168. NIPS Foundation (http://books.nips.cc).

[8] Bottou, L. and Lin, C.-J. (2007). Support vector machine solvers. *Large scale kernel machines*, 3(1):301–320.

[9] Bousquet, O. and Bottou, L. (2008). The tradeoffs of large scale learning. In *Advances in neural information processing systems*, pages 161–168.

[10] Breunig, M., Kriegel, H., Ng, R., and Sander, J. (2000). Lof: Identifying density-based local outliers. *SIGMOD Rec.*, 29(2):93–104.

[11] Campos, G. O., Zimek, A., Sander, J., Campello, R. J., Micenková, B., Schubert, E., Assent, I., and Houle, M. E. (2016). On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study. *Data Mining and Knowledge Discovery*, 30(4):891–927.

[12] Castillo, E., Peteiro-Barral, D., Berdiñas, B. G., and Fontenla-Romero, O. (2015). Distributed one-class support vector machine. *International journal of neural systems*, 25(07):1550029.

[13] Chandola, V., Banerjee, A., and Kumar, V. (2009). Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):15.

[14] Chandola, V., Banerjee, A., and Kumar, V. (2012). Anomaly detection for discrete sequences: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 24(5):823–839.

[15] Das, K. and Schneider, J. (2007). Detecting anomalous records in categorical datasets. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '07, New York, NY, USA. ACM.

[16] Das, S., Matthews, B. L., Srivastava, A. N., and Oza, N. (2010). 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*, KDD '10, pages 47–56, New York, NY, USA. ACM.

[17] Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7(Jan):1–30.

[18] Edgeworth, F. (1887). On discordant observations. *Phylosoph. Mag.*, 23(5):364–375.

[19] Emmott, A. F., Das, S., Dietterich, T., Fern, A., and Wong, W.-K. (2013). Systematic construction of anomaly detection benchmarks from real data. In *Proceedings*

of the ACM SIGKDD workshop on outlier detection and description, pages 16–21. ACM.

[20] Fernández-Francos, D., Martínez-Rego, D., O.Fontenla-Romero, and Alonso-Betanzos, A. (2013). Automatic bearing fault diagnosis based on one-class nu-svm. *Computers & Industrial Engineering*, 64(1):357–365.

[21] Fujimaki, R., Yairi, T., and Machida, K. (2005). An approach to spacecraft anomaly detection problem using kernel feature space. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 401–410. ACM.

[22] Ghamrawi, N. and McCallum, A. (2005). Collective multi-label classification. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*, pages 195–200.

[23] Ghoting, A., Otey, M., and Parthasarathy, S. (2004). Loaded: link-based outlier and anomaly detection in evolving data sets. In *Data Mining, 2004. ICDM '04. Fourth IEEE International Conference on*, pages 387–390.

[24] Hawkins, D. M. (1980). *Identification of outliers*, volume 11. Springer.

[25] He, Z., Deng, S., and Xu, X. (2005). An optimization model for outlier detection in categorical data. In Huang, D., X.P. Zhang, G., and Huang, editors, *Advances in Intelligent Computing*, volume 3644 of *Lecture Notes in Computer Science*, pages 400–409. Springer Berlin Heidelberg.

[26] Hettich, S. and Bay, S. (1999). KDD cup 1999 data. *The UCI KD Archive, Irvine, CA: University of California, Department of Information and Computer Science*.

[27] Hu, W., Hu, W., and Maybank, S. (2008). Adaboost-based algorithm for network intrusion detection. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 38(2):577–583.

[28] JeffreyXu, Y., Qian, W., Hongjun, L., and Aoying, Z. (2006). Finding centric local outliers in categorical/numerical spaces. *Knowledge and Information Systems*, 9(3):309–338.

[29] Khan, S. S. and Madden, M. G. (2014). One-class classification: taxonomy of study and review of techniques. *The Knowledge Engineering Review*, 29(03):345–374.

[30] Kumar, V. (2005). Parallel and distributed computing for cybersecurity. *IEEE Distributed Systems Online*, 6(10):1–10.

[31] LaValle, S., Lesser, E., Shockley, R., Hopkins, M. S., and Kruschwitz, N. (2011). Big data, analytics and the path from insights to value. *MIT sloan management review*, 52(2):21.

[32] Lazarevic, A. and Kumar, V. (2005). Feature bagging for outlier detection. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 157–166. ACM.

[33] Lichman, M. (2013). UCI machine learning repository. http://archive.ics.uci.edu/mll [Last accessed April 2017].

[34] Liu, F. T., Ting, K. M., and Zhou, Z.-H. (2012). Isolation-based anomaly detection. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 6(1):3.

[35] Martinez-Rego, D., Castillo, E., Fontenla-Romero, O., and Alonso-Betanzos, A. (2013). A Minimum Volume Covering Approach with a Set of Ellipsoids. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(12):2997–3009.

[36] Martínez-Rego, D., Fernández-Francos, D., O.Fontenla-Romero, and Alonso-Betanzos, A. (2015). Stream change detection via passive-aggressive classification and bernoulli CUSUM. *Information Sciences*, 305:130–145.

[37] Otey, M., Ghoting, A., and Parthasarathy, S. (2006). Fast distributed outlier detection in mixed-attribute data sets. *Data Mining and Knowledge Discovery*, 12(2-3):203–228.

[38] Papadimitrou, S.and Kitagawa, H., Gibbons, P., and Faloutsos, C. (2002). Loci: Fast outlier detection using the local correlation integral. Technical report irp-tr-02-09, Intel Research Laboratory.

[39] Sarasamma, S. T., Zhu, Q. A., and Huff, J. (2005). Hierarchical kohonenen net for anomaly detection in network security. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 35(2):302–312.

[40] Scholkopf, B., Platt, J., Shawe-Taylor, J., Smola, A., and Williamson, R. (2001). Estimating the support of a high-dimensional distribution. *Neural Computation*, 13(7):1443–1471.

[41] Schubert, E., Zimek, A., and Kriegel, H. (2012). Local outlier detection reconsidered: a generalized view on locality with applications to spatial, video, and network outlier detection. *Data Mining and Knowledge Discovery*, pages 1–48.

[42] Shiravi, A., Shiravi, H., Tavallaee, M., and Ghorbani, A. A. (2012). Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *computers & security*, 31(3):357–374.

[43] Singh, S., Tu, H., Donat, W., Pattipati, K., and Willett, P. (2009). Anomaly detection via feature-aided tracking and hidden markov models. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 39(1):144–159.

[44] Sodemann, A., Ross, M., and Borghetti, B. (2012). A review of anomaly detection in automated surveillance. *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, 42(6):1257–1272.

[45] S.Wu and Wang, S. (2011). 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*, volume 6871, pages 112–126.

[46] Walker, S. H. and Duncan, D. B. (1967). Estimation of the probability of an event as a function of several independent variables. *Biometrika*, 54(1-2):167–179.

[47] Wei, L., Qian, W., Zhou, A., Jin, W., and Yu, J. (2003). Hot: Hypergraph-based outlier test for categorical data. In Whang, K., Jongwoo, J., Shim, K., and Srivastava, J., editors, *Advances in Knowledge Discovery and Data Mining*, volume 2637 of *Lecture Notes in Computer Science*, pages 399–410. Springer Berlin Heidelberg.

Appendix - Best hyperparameters

This is the list of the best hyperparameter combination for each method for each dataset in the experiments reported in the paper. Hyperparameters 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 VI
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 VII
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 VIII
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 IX
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 X
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