Time series anomaly detection

Jacopo Gobbi
University of Trento
194438
jacopo.gobbi@studenti.unitn.it

Kateryna Konotopska
University of Trento
198234
kateryna.konotopska@studenti.unitn.it

ABSTRACT

Anomaly detection is a recurrent topic in various fields, while it is done for different reasons, such as cyber-security or industrial applications, there exist a common objective: find a deviation from the common rule, peculiar in such a way that we might even declare it is not part of what we have been observing until now. While solving this problem might seem obvious to humans, developing a scalable system which retain expressivity while preserving resources is prone to difficulties, like finding a general purpose distance measure, or minimizing algorithm complexity. In this paper we test three kinds of anomaly detection algorithms on time series data: our proposed SOM based solution and two state-of-the-art models from literature: a detector built on an ensemble of vectors and a model which makes use of symbolization based pattern analysis.

Keywords

anomaly detection; time series

1. INTRODUCTION

Given the potential scale of modern systems, the requisites for a working solution - for any problem - have never been more demanding; good results must be provided in a timely fashion, with minimal memory footprint and taking into consideration possible faults of the many sub-systems that could be feeding data to our own.

Anomaly detection is no different, for a modern business, such as Amazon, not having an automatic way of finding outliers is obviously unfeasible, given that those could be frauds or indicate malfunctions on a system level.

We focus on time series data, which is made of samples that have a specific order, given by the time of their appearance; this poses another problem, while samples in non time series data have a meaning of their own (imagine dog pictures vs cat pictures), our data points might not be much informative on their own, requiring some kind of domain knowledge or tuning to understand what makes sense semantically; here the anomaly is usually not a single point, but an anomalous pattern or window of values.

This report aims to test and compare three different approaches to solve this problem, both on artificial and real data, univariate and multivariate; the first approach is a SOM [Kohonen and Honkela, 2007] based classifier that tries to capture the different patterns in the data and scores samples by how un-patterny the window they are contained in is. The second approach is LODA, an ensemble of - extremely - weak classifiers which models the joint probability of the data in an approximate way in order to mark as anomalies windows of samples deemed too improbable. The third and last model is the time series Bitmaps algorithm, introduced by [Lin et al., 2003], based on the probability distribution of discretized patterns found in the data.

2. RELATED WORK

Researchers have been tackling anomaly detection with many different tools, ranging from data-centered solutions, like k-nearest neighbour, to peculiar neural models, like variational auto-encoders [Kingma and Welling, 2013]; while a full and detailed survey is of course out of the scope of our report, we try to address the most pertinent with our work.

Rule-based expert systems [Ligeza, 2006] faded away with data becoming more and more abundant; nowadays, most scalable solutions are now based on building some sort of model or gathering significant statistics that can help in discriminating anomalous observations.

To do that, some models attempt at modeling the joint probability of the data distribution, while others try to find rules, hyper-planes, or more complex shapes that separate normal data from exceptions, like [Scholkopf et al., 2001]; others embed the input data to smaller spaces, and exploit those to remove noise or facilitate classification, this is the case for PCA based models [Shyu et al., 2003] or the whole plethora of auto-encoders based architectures, like [Malhotra et al., 2015], [Malhotra et al., 2016], [Park et al., 2017].

Clustering, whilst not directly related to anomaly detection, has been used for this purpose, backed by the intuition that ordinary data points would usually be grouped together, while abnormal ones should be far away from the others, given some kind of distance measure and scale; works of this type span from simple, using k-means [Gaddam et al., 2007], to more complex like db-scan [Celik et al., 2011].

Regarding self-organizing maps (SOMs), there exist many kinds of adaptations, especially for time dependant data, among those we note: Kangas' Model [Kangas et al., 1990], which simply recombines the current input with past inputs by a scaling factor, the TKM-Temporal Kohonen Map [Chappell and Taylor, 1993], which merges the result of each representant with its value from the previous step, the RSOM-Recurrent SOM [Koskela et al., 1998], here a temporal smoothing rule acts on the difference between the input and the representant(s), this memory is also taken into account when updating the representants in the map.

Other, more complex designs exists, like [Shahreza et al., 2011], which considers the different representants in the SOM as particles to adjust in order to solve an optimization problem using particle swarm optimization.

To conclude on SOMs, the very large majority of SOM based approaches determines anomalies using a form of error related to how dissimilar a sample is to its representant, in our proposal we try to also include a score based on representants frequencies, which reuses results from the map update step to keep track of trends in the usage of representants in a cheap way while maintaining the smoothness with which the update propagates.

Ensemble methods have also been put to test for this task (altough not much), isolation forests [Liu et al., 2008] take a set of isolation trees built on unlabeled data, each tree is grown until leafs contain a single sample, more anomalous samples will likely position themselves in more deeply placed leaf; [Kashef, 2018] use cooperative clustering to find anomalies through collective scoring; LODA [Pevný, 2016], uses weak classifiers (simple vectors) to approximate the density of the data by keeping track, using histograms, of the result of dot products between samples and its vectors.

The field of deep learning has also contributed to solving this problem, from simple fully connected classifiers to recurrent variational denoising autoencoders, their feature extraction and embedding capabilities have been exploited to distinguish the extraordinary from the normal. While more classical architectures operate as simple classifiers, autoencoders usually operate on the reconstruction error, on the intuition that if we the network is used to encode/compress normal data, it will struggle with anomalies. Discretization approaches: works based on discretization in anomaly detection domain are [Zhang et al., 2016], [Truong and Anh, 2015], [Morgan et al., 2007], [Sajjipanon and Ratanamahatana, 2009] and others; these methods use a fixed symbol representations of continuous values and use them for further clustering/pattern investigation.

3. PROBLEM STATEMENT

Anomaly detection, despite being a difficult task, is not defined by overly complex formulas or by a clear optimization problem to solve; informally, we are looking for something that deviates from what is otherwise considered normal, we are thus looking for what breaks expectations in some way (and for this reason this task can be related to novelty detection).

Marginally more formal definitions exist in literature, such as:

- ”An observation which appears to be inconsistent with the remainder of that set of data” (Barnett and Lewis, 1984).
- ”Anomalies are patterns in data that do not conform to a well defined notion of normal behaviour” (Chandola et al., 2009).

By now it should be clear that we are looking for data points or patterns that do not comply with the rest of our data; to do this, we may face many challenges, among which we can find: the scalability of our solutions, concept drift (related to change in the distribution over time), seasonality (slightly related to concept drift, but more about recurring trends) and noise. In a more pragmatical definition, given a sequence of n elements (for time series, a set for time independent data), we must return a score for each of these elements, with higher scores representing anomalies.

Different classes of outlier detection can be distinguished, in **unsupervised anomaly detection**, unlabeled data is fed to the detector, expecting that anomalies should be noticed given that the majority of the data is instead ordinary; a dataset of known regular samples is instead used for training the model in **semi-supervised detection**, later to be run on a test set. The **supervised** category focuses instead on training a classifier between normal and anomalous instances, given labeled data, the main drawback of this method is given by class imbalance and the fact that our system might over-fit to only consider the shown categories of anomalies as outliers.

Moreover, **on-line anomaly detectors** have the further constraint of having no dataset, but data points that are delivered in a stream, this poses the additional challenge of not having some kind of global view of the data, only partial; related works are [Tan et al., 2011], [Spinosa et al., 2009] and LODA [Pevný, 2016].

4. SOLUTION

We compare our SOM based proposal with LODA and a library from the industry [LinkedIn, 2017], the first two models were implemented by us, and the implementation can be found at [Gobbi and Konotopska, 2018]; LODA has been implemented as the alternating histograms version. Although some optimization has been carried out, our implementations could be optimized more, but we preferred to keep things more simple for the sake of clarity.

As a general rule, these models receive as input a window of samples, in the univariate case the window is a vector of data points ranging from time t to $(t + window\ size - 1)$; for the multivariate case, given n variables, the input vector is built by concatenating the n windows of each variable.

4.1 SMOOTH-SOM: SOM with smooth frequencies

A self-organizing map, or Kohonen map, is a model that aims to produce a low dimensional and discrete representation of an otherwise larger and usually continuous input space. To do that, a certain number of representants is initialized and, more or less abstractly, they are organized in such a way that a distance or a similarity measure based on the topology of this organization can be defined, usually done by considering them to be positioned in a two dimensional grid or more refined shapes, such as toroidals.

As a similarity measure we use a Gaussian function, which decreases in value in a smooth way as we get further from a certain representant; the similarity for a two dimensional map with first dimension x and second dimension y can be defined as:

$$s(rep, rep') = \exp(\frac{-(x-x')^2}{\sigma^2}) * \exp(\frac{-(y-y')^2}{\sigma^2})$$

With x and y being the first and second coordinate of rep, and x', y' the coordinates of the rep', σ helps in regulating the rate of change of the similarity. For more dimensions, this generalizes to:

$$s(rep, rep') = \prod_i \exp(\frac{-(c_i - c_i')^2}{\sigma^2}) = \exp\frac{-\sum_i (c_i - c_i')^2}{\sigma^2}$$

Starting from an empty map, with randomly initialized representants (vectors of length equal to the size of the window

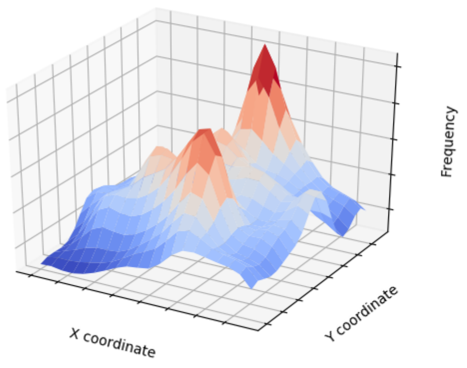

Figure 1: Example of the topology of frequencies using smooth counter updates.

of data point we are considering), we add one data point at a time, finding its representant and updating the map accordingly.

To find the representant for a data point we simply iterate over all representants, picking the one minimizing distance, in our case the normalized Euclidean distance; once that is done, the map is updated, changing every element by a magnitude dependant on its similarity (distance in the 2D map) with the representant, scaled by a constant. Formally, the update function for a single element is defined as:

$$rep_i(t+1) = rep_i(t) + s(rep_i(t), rep_{dp}(t)) * \alpha * (dp - rep_i(t))$$

Where dp is the data point, which is in our case a window of samples (so technically just a vector) and α is the scaling factor.

Moreover, we keep track of the frequency of appearance by having a 2D grid of counters, which gets smoothly updated by using the already computed Gaussian similarity between the chosen representant and the rest of the elements:

$$count_i(t+1) = count_i(t) + s(rep_i(t), rep_{dp}(t))$$

This basically updates by 1 the counter of the representant, and propagates smaller values in the rest of the map; the important difference between this and a +0/+1 approach is that such continuity preserves the concept of similarity between neighbours by smoothly distributing the increment.

Each data point is simultaneously added and scored as an anomaly, deciding on the combination of two scores: the distance measure with the representant, and the negative log of the normalized frequency counter of the representant; higher values will indicate anomalies; we also employ a periodical decay on the counters to allow better management of concept drift.

This model requires the following parameters to be defined: **x** and **y**, which are the dimensions of the map and directly define the number of representants; **the size of each representant**, which is its length as a vector and thus the windows of samples we are considering; the length of the **decay period**, which is how often we rescale the counters of the map, the **decay factor** (which we kept at 0.5), which is a value between 0 and 1 used for decaying; σ and α as

Algorithm 1 SOM anomaly detection

```
    Input data point $dp$, $\alpha$, $\sigma$
    Output anomaly score
 1: procedure ADD DATA POINT
 2:     $rep, distance \leftarrow SOM.get\_best\_representant(dp)$
 3:     $g\_sim \leftarrow gaussian\_sim(SOM.x, SOM.y, rep.x, rep.y, \sigma)$
 4:     $SOM.update\_reps(dp, g\_sim, \alpha)$
 5:     $SOM.counters += g\_sim$
 6:     $freq\_score = -\log \frac{SOM.counter_{rep.x,rep.y}}{\sum_i^{SOM.x} \sum_j^{SOM.y} (SOM.counter_{i,j})}$
 7:     $SOM.decay\_counter \leftarrow SOM.decay\_counter + 1$
 8:     if $SOM.decay\_counter = SOM.decay\_period$ then
 9:         $SOM.decay\_counter \leftarrow 0$
10:         $SOM.counters *= SOM.decay\_factor$
11:     end if
12:     $return\ distance * freq\_score$
13: end procedure
```

precedently explained.

Given n windows of samples and r representants, having d dimensions, the complexity of this model is $O(n*r*d)$, which is linear with respect to the data points, this is almost a requirement for all algorithms, since anything beyond linear extremely reduces scalability for big data; a favorable characteristic of this solution is that it can be entirely implemented in matrix operations, which makes it fast and easily parallelizable.

4.2 LODA

LODA [Pevný, 2016] is an ensemble based anomaly detector which operates by approximating the joint probability of the distribution, the ensemble is comprised of a set of one dimensional histograms obtained by projecting a selected part of the input space into randomly generated vectors, the histograms are then used to compute an anomaly score by taking the negative log of the approximated probability:

$$f(x) = -\frac{1}{k} \sum_i^k \log p_i(x^T w_i)$$

Where x is the input vector (our window of samples), p_i is the probability estimation of histogram i, w_i is the *ith* projection vector and k is the total number of histograms.

As far as we know, LODA is the first model to demonstrate that a set of very weak classifiers can act as a strong anomaly detector, moreover, it can deal with missing variables and rank features according to their contribution to the anomaly score, while these two features have not been experimented with, we can appreciate how such a simple model delivers strong results and such useful characteristics.

Part of the strength of LODA is related to its low number of parameters and the fact that the number of histograms, bins for each histogram, and dimensions over which we are randomly projecting are backed by sound mathematics.

Number of histograms The number of histograms is determined by stopping at the minimum k such that adding more histograms would result in a reduction of variance below a threshold τ (we use a τ equal to 0.01, as suggested by the author of LODA), so, given $f_k(x)$ being the anomaly score of LODA while having k histograms, we define the *kth* reduction of variance as (note that n is the total number of samples):

$$\sigma_k = \frac{1}{n}\sum_{i}^{n} \mid f_{k+1}(x_i) - f_k(x_i) \mid$$

We then look for the k such that:

$$\underset{k}{\text{argmin}} \frac{\sigma_k}{\sigma_1} <= \tau$$

with σ_1 being used as a normalization factor to make the obtained value less problem-dependent, thus allowing τ to not be a parameter but a set value.

Histograms bins The number of equi-width bins is determined ad hoc for each histogram (given that they project on different features, the number of optimal bins may vary), using the method proposed by [Birge and Rozenholc, 2006], based on picking the number of bins b which maximized the penalized maximum likelihood:

$$\underset{b}{\text{argmax}} \sum_{i}^{b} n_i \log \frac{bn_i}{N} - \left[b - 1 + (\log b)^{2.5}\right]$$

with N being the number of total samples, ni the samples mapped to bin b_i.

Vectors initialization Each vector, one for each histogram, is randomly initialized by having $\sqrt{d}$ non-zero elements sampled from $N(0,1)$; this is because it has been shown that both choices have preserving properties with respect to the $L2$ distance of the input space, [Li, 2007] [Johnson and Lindenstrauss, 1984].

To deal with concept drift LODA employs floating window histograms and alternating histograms; we choose to implement and test the latter, given its more robustness to clustered anomalies [Pevný, 2016]. While running, the old histogram set is used for classification, while the new one will be constructed and will be replacing the old one after a certain interval; this interval, (along with the window size over samples, if we are dealing with time series data), is the only parameter needed by LODA.

To synthesize, once we have built our histograms either by accumulating a first batch of data or on a training set, we project the input data point over all projection vectors, compute the probability p_i given by each histogram based on the result of $x^T w_i$, and then average over the negative log of these values to obtain an anomaly score; if we reached the end of the decay interval we build new histograms using the last n data points we have classified, where n is equal to the length of the interval.

As SMOOTH-SOM (SSOM), this model is built upon very simple and efficient operations, moreover, its complexity is $O(n{*}k{*}\sqrt{d})$, which is lower than SOM, but practically equal if we are not exploiting the sparseness of our vectors, which might be worth only for very large values of d.

4.3 Time Series Bitmaps

The main idea behind Bitmaps algorithms is the investigation of patterns in sequential data. In order to use this method the dataset must contain a finite range of values. Since the time series domain is usually of continuous nature, a discretization is applied, using for example the Symbolic Aggregate approximation (SAX) of [Lin et al., 2003].

To analyze the patterns of every discretized sub-sequence of the dataset the counts of all possible combinations of the alphabet values of length **k** are computed. k is a parameter defined by the user and defines the length of patterns(combinations) to analyze.

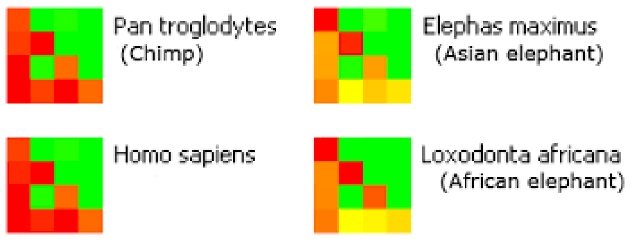

Figure 2: Bitmaps of DNA sequences of four animals. Taken from [Wei et al., 2005]

Those counts are then fitted into a matrix, called the Bitmap matrix, containing k^d counts, where k^d is the number of all the possible combinations of k values of range **d**, the size of the alphabet. These matrices are easy to analyze even by visual inspection, by coloring based on the values of counts, as can be seen from the example in figure 2, which shows how unrelated species have visually discernible matrices.

To compute the distance between two sub-sequences a distance between their bitmaps is computed. It is defined as the sum of the squared distances between each pair of frequencies:

$$dist(a,b) = \sum_{i}^{n}\sum_{j}^{n}(a_{ij} - b_{ij})^2$$

where a and b are the corresponding bitmaps, n is the dimension of the square matrices.

aa	ab	ba	bb
ac	ad	bc	bd
ca	cb	da	db
cc	cd	dc	dd

Table 1: Example of bitmap patterns for alphabet {a,b,c,d}, k=2, d=4; each cell of the bitmap contains counts of these patterns

Anomaly detection is performed as described in [Wei et al., 2005]: two sliding windows are used to detect anomalies by sliding them across the dataset. The latter window, called **lag window**, is used to look ahead for anomalous patterns and the former one, called **lead window**, is a kind of memory window, used by the algorithm to predict the future. After applying discretization to the two windows, their bitmaps distance is used as the anomaly score.

For each new step of sliding window only two cells of each bitmap are updated: a pattern counters increments of a new data point and pattern counters decrements of an old data point. Therefore, the time complexity is $O(4*n)$, linear in the length of the dataset.

The quality of the results depends on the discretization precision, sliding windows dimensions, and the length of the sub-sequence patterns to analyze, introducing a low number of parameters. Their recommended values are described in [Wei et al., 2005].

The open-source implementation of this algorithm: LinkedIn Luminol [LinkedIn, 2017] python library was used.

5. EXPERIMENTS

We experimented on the aforementioned algorithms in a supervised manner, by splitting data in training and test sets, trying to be proportionate with the number of anomalies contained in the two divisions more than the total number of data points. An exception is made for the dataset Riccione, for which we took about one fifth of the data as train set, due to its large size.

Although we are doing supervised detection, we tried to keep the supervision as limited as possible, meaning that while these models are technically classifiers, we do not provide label information during training (as we would, for example, do for neural networks in order to back propagate on errors on an instance basis); we simply run a set of experiments with randomized parameters, in order to choose which combination works best. These architectures can in fact work in a non-supervised manner, but some of the parameters would need domain knownledge.

However, in the results section we report both results on the test sets (by supervised detection, after tuning on train data) and an average on all the runs on the train sets, which can indeed be considered a form of unsupervised detection task, with no knownledge on parameters.

Beyond the parameters described in the previous section, we also experimented with normalization and - limitedly - with smoothing; their impact on performance is reported later, while here we - extremely briefly - describe the smoothing technique used.

The Savitzky Golay Filtering [Savitzky and Golay, 1964] is a type of low-pass filter which is often used to clear the spectral lines in noisy spectrometric data; if we were naive, we could use some very simple form of linear combination of the neighbours of our sample (data at $t-1$, $t+1$, etc.), like moving window averaging. This works fine when our data acts linearly, but ends up intrudcing some bias if the underlying function has a second derivative with value different from zero.

The purpose of the Savitzky Golay technique is to take into consideration higher moments by introducing coefficients able to preserve those; to do that, a polynomial is fitted to the data with least-square error, not by ordinary means (like gradient descent, etc.) but by finding a particular set of coefficients (by solving a system of linear equations) such that using those to linearly combine the points part of the current window would have the same effect of attempting to fit the window by use of polynomial least-squares. The level of fitness depends on the order parameter, the higher, the (usually) more fitting.

5.1 Data

We performed our experiments on three artificial univariate datasets, two univariate real datasets and a multivariate real dataset, which is the largest in terms of data points and time period.

5.1.1 Artificial data

Taken from the NUMENTA anomaly benchmark repository [Numenta, 2018], the three datasets (one of which is used for tuning) have the same length of 4032 samples, each of them has a number of anomalies equal to 403 (total number of indexes flagged as anomalies/part of an anomalous pattern) which represent an interval of lack of regularity in what is an otherwise fairly periodic and predictable data.

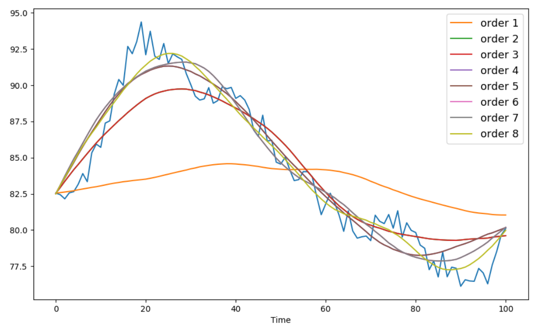

Figure 3: Different orders of smoothing applied to a window of the taxi dataset.

5.1.2 Real data

Taxi Each sample represents the number of people taking a taxi in New York at a certain moment; while the original dataset from the NYC Taxi and Limousine Commission was more fine grained, we use a version which aggregates those values every 30 minutes.

Five anomaly intervals are present: (1) during the NYC marathon, (2) Thanksgiving, (3) Christmas, (4) New Years Eve, (5) a snow storm; resulting in 1035 indices flagged as anomalies out of 10320.

Machine Temperature This dataset consists of readings from a sensor monitoring the temperature of an industrial machine, three anomalies are present: a scheduled shutdown, and two real, unexpected situations. Out of 22695 data points, 2268 are considered as anomalous.

Riccione Readings from 24 different sensors from a purification plant comprised of three main lanes, those values represent: flow rate, solids currently in a lane, oxygen, ammonia, nitrates, oxygen being injected and pressure on different valves. Anomalies here are fairly heterogeneous, spacing from sensor re-calibrations, to probes erroneous readings, valves malfunctioning, or the load of the plant being watered down due to heavy rain.

The data was provided by the company in charge of the plant; given its large size, we aggregated values in 30 minutes buckets, resulting in 22206 data points for each variable, out of which 1828 are anomalies. The provided labeling is quite poor, so we do not expect performance to carry over from the tuning set to test set.

5.2 Performance metrics

To measure the performance of the tested algorithms we proceeded as following: given a dataset comprised of n data points, we flag any data point part of intervals that were marked as anomaly as class 1, and the rest as class 0, essentially transforming our problem in a binary classification problem.

For parameter tuning, we use the parameters that resulted in the best **ROC** score [Fawcett, 2006] on the train set. Moreover, to help the reader get a more general idea on the performance of each algorithm, given the anomaly scores obtained by using the best parameters, we look for the thresholds that would respectively lead to the best F_1 and $F_{0.1}$ scores, using those we then compute and report the following metrics: **recall**, **precision**, **false positive rate**, **total**

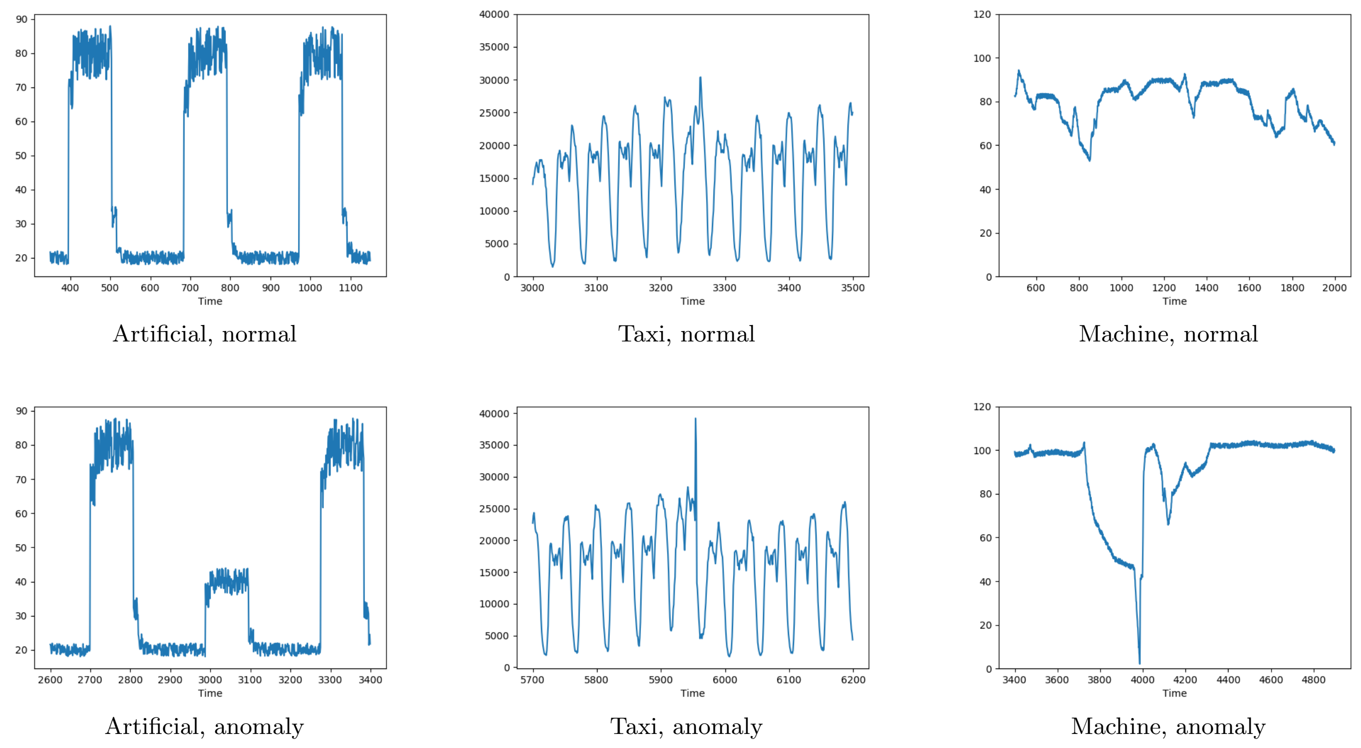

Figure 4: Examples of ordinary and anomalous behaviours.

anomalies correctly labeled.

5.3 Results

Results have been more or less stable, with LODA being the most trustable in general; we believe this is due to two reasons, **(1)** while simple, the algorithm is backed by sound mathematics, **(2)** its low number of parameters, which allows for an easier tuning.

Algorithm	Dataset	ROC	Parameters
SSOM	Art.1	0.973	smoothing=1, norm=1, dimension=10, wsize=301 , sigma=1, update_weight=0.2, decay_period=600
LODA	Art.1	0.997	smoothing=0, norm=0, wsize=225, memory=1400
Bitmap	Art.1	0.608	smoothing=45, norm=0, chunk_size(k)=7, precision(d)=8, lag_win_size=1500, lead_win_size=1500
SSOM	Art.2	0.977	smoothing=1, norm=0, dimension=10, wsize=301 , sigma=1, update_weight=0.2, decay_period=600
LODA	Art.2	0.988	smoothing=0, norm=0, wsize=225, memory=1400
Bitmap	Art.2	0.592	smoothing=45, norm=0, chunk_size(k)=7, precision(d)=8, lag_win_size=1500, lead_win_size=1500
SSOM	Taxi	0.919	smoothing=5, norm=0, dimension=8, wsize=175, sigma=5, update_weight=0.001, decay_period=1400
LODA	Taxi	0.972	smoothing=5, norm=0, wsize=125, memory=1600
Bitmap	Taxi	0.700	smoothing=8, norm=1, chunk_size(k)=4, precision(d)=4, lag_win_size=2000, lead_win_size=40
SSOM	Machine	0.695	smoothing=45, norm=0, dimension=2, wsize=275, sigma=3, update_weight=0.001, decay_period=400
LODA	Machine	0.933	smoothing=1, norm=0, wsize=301, memory=800
Bitmap	Machine	0.840	smoothing=0, norm=1, chunk_size(k)=5, precision(d)=2, lag_win_size=1500, lead_win_size=1200
SSOM	Riccione	0.570	smoothing=1, norm=1, dimension=9, wsize=4801 , sigma=3, update_weight=0.1, decay_period=800
LODA	Riccione	0.565	smoothing=0, norm=1, wsize=1801, memory=600
Bitmap	Riccione	0.487	smoothing=5, norm=0, chunk_size(k)=12, precision(d)=14, lag_win_size=1600, lead_win_size=600

Algorithms and their ROC scores on test datasets, with parameters. Smoothing=0 means no smoothing, while otherwise it refers to the smoothing order. Norm=1 means that normalization (on a window level) has been applied.

SSOM has provided competitive results, and we are satisfied with it, moreover, its memory footprint is the lowest among the three models, with LODA having a slightly bigger one, and Bitmap having the highest memory requirements by a big margin, depending on parameters.

In terms of speed, Bitmap is usually the fastest, with

Model	SET	F1	FPR	R	P	#L	#CP
SSOM	Art.1	0.80	0.06	0.67	0.99	403	403
LODA	Art.1	**0.89**	0.00	0.80	0.99	403	324
Bitmap	Art.1	0.24	0.61	0.89	0.14	403	360
SSOM	Art.2	0.82	0.05	0.99	0.70	403	403
LODA	Art.2	**0.86**	0.03	0.94	0.79	403	378
Bitmap	Art.2	0.46	0.03	0.37	0.59	403	151
SSOM	Taxi	0.74	0.22	0.99	0.60	414	414
LODA	Taxi	**0.85**	0.04	0.84	0.87	414	347
Bitmap	Taxi	0.50	0.36	0.70	0.39	414	288
SSOM	Mach.	0.41	0.00	0.26	0.98	1134	296
LODA	Mach.	0.45	0.07	0.50	0.41	1134	567
Bitmap	Mach.	**0.52**	0.10	0.72	0.41	1134	818
SSOM	Ricc.	**0.18**	0.35	0.51	0.12	31200	15897
LODA	Ricc.	0.14	0.99	0.99	0.08	31200	31200
Bitmap	Ricc.	0.14	0.99	0.99	0.08	31200	31200

Table 2: F_1 results, #L stands for number of anomalies (indexes flagged as 1) in the test set, #CP for the number of correctly predicted anomalies. P stands for precision, R for recall, and FPR for the false positive rate.

Model	SET	F0.1	FPR	R	P	#L	#CP
SSOM	Art.1	0.67	0.05	0.99	0.67	601	403
LODA	Art.1	**0.99**	0.00	0.80	0.99	324	324
Bitmap	Art.1	0.14	0.52	0.77	0.14	2220	313
SSOM	Art.2	0.71	0.02	0.54	0.71	304	217
LODA	Art.2	**0.80**	0.03	0.94	0.79	476	378
Bitmap	Art.2	0.59	0.03	0.37	0.60	253	151
SSOM	Taxi	**0.99**	0.00	0.43	0.99	179	179
LODA	Taxi	0.98	0.00	0.30	0.99	125	125
Bitmap	Taxi	0.75	0.00	0.03	0.99	12	12
SSOM	Mach.	**0.95**	0.00	0.26	0.98	301	296
LODA	Mach.	0.41	0.07	0.50	0.41	1380	567
Bitmap	Mach.	0.90	0.00	0.08	0.99	87	87
SSOM	Ricc.	**0.23**	0.01	0.04	0.23	31200	1152
LODA	Ricc.	0.08	0.99	0.99	0.08	31200	31200
Bitmap	Ricc.	0.08	0.99	0.99	0.08	31200	31200

Table 3: $F_{0.1}$ results, both this table and Table 3 refer to results obtained by using the parameters that maximized the ROC score on the training set.

SSOM and LODA trailing behind; however our implementation has still room for optimization, while Luminol, the library implementing the algorithm, is provided by LinkedIn.

We have run our experiments in offline supervised detection mode, but we also provide results on online unsupervised detection by reporting the average ROC score of the tuning trials on each dataset, for each algorithm, in table 4. While these may not be perfectly indicative of performances, given that tuning runs are done with random parameters, they add some insight on how these models perform in a context of zero knownledge of parameters and data received in streaming.

The Riccione dataset had overall poor results, while part of the reason is the lower number of tuning runs we made (given the size of the data), we believe those low scores are mostly due to the very poor labeling that has been provided with the dataset. Given that it is a multivariate (24 variables) time series, we could not hope to correct ourselves the labeling, because of its size and our lack of domain knowledge.

6. CONCLUSION

Parameters still pose a challenge for the design of a general purpose algorithm, while they allow the user to better express its needs, they are ultimately a burden to tune, and should be kept at a minimum.

The trade-off between what our algorithm should do (requirements such as online detection, memory, complexity) and how little prior information it needs (parameters) is what ultimately constitutes the difficulties in their design, especially in this task.

As an example, the requirements on false positives and recall are arbitrarily different based on the specific task, it is still not possible to provide an algorithm which accommodates all needs by determining a threshold on its own, due to the fact that the definition of anomaly does not have a set limit on how improbable an outlier should be, such an algorithm would inevitably have some assumptions or bias. A threshold could perhaps be determined by introducing a human-computer loop as suggested by [Battiti and Campigotto, 2010]; by periodically proposing to a user a set of possible anomalies, and letting him decide what and what should not have been defined as so.

Future work Doing this project, especially during the phase of gathering the scientific literature and while implementing LODA, led to conceiving many ideas, which unluckily we could not develop further due to lack of time.

Model	SET	avg. ROC
SSOM	Art.	0.40
LODA	Art.	**0.75**
Bitmap	Art.	0.55
SSOM	Taxi	0.50
LODA	Taxi	**0.78**
Bitmap	Taxi	0.43
SSOM	Mach.	0.69
LODA	Mach.	**0.83**
Bitmap	Mach.	0.55
SSOM	Ricc.	0.48
LODA	Ricc.	0.45
Bitmap	Ricc.	**0.54**

Table 4: Average ROC score on randomized tuning runs on the train sets, this should offer some insight on how these algorithms act in a data streaming, no knowledge on parameters context.

First, we appreciated the approach used by LODA to effectively get rid of parameters (number of histograms, number of bins) with an elegant, mathematically proven solution; which is something that our SSOM model could indeed use; we could, in fact, decide on the number of representants by stopping when the return on adding more representants gets too low, as LODA does.

Another improvement for our proposal would be moving on from grid shaped topologies to toroidals or more complex shapes, like the star shaped SOM from [Come et al., 2010].

One advantage of SOM over the other two architectures is that it retains information on the shape of the data, allowing for further analysis by simply visually inspecting the most used representants; our addition of smooth frequencies could also permit our model to have some sort of generative properties by extending the collected statistics by also taking note of representant transitions, i.e. representant chosen at time t has coordinates $x = 4$ and $y = 1$, while representant at time $t + 1$ has coordinates $x = 2$ and $y = 9$, so we increment that transition counter.

It is not far fetched to say that this could very easily become a Markov model of some order, with the order depending on how far we want to keep track of transitions. Noise could be used both on the representants and on the choice of transition, as an example, if we are to transition to the representant with coordinates x, y, we transition instead to random variables K, J, of distribution N(x, S) and N(y, S) respectively, where S is a parameter directly tied to the amount of noise we want; this is made possible thanks to the fact that neighbours in a SOM are similar.

Another possibility would be rethinking the way data points are labeled as anomalies: first, instead of considering data as two classes, ordinary (0) and anomalous (1), we should consider each data point has having a score; then, given index i of a point indicating the most severe moment of an anomaly window, we could assign it a score of 1.0, and then smoothly decrease that score while passing it to surrounding samples ($i - 1$, $i - 2$, etc); this would allow for a more sensible scoring for two reasons: **(1)** labeling made by humans, especially on deciding when an anomalous period starts and ends, is made in a way that makes sense to them, and it is a bit fuzzy, **(2)** with classical labeling, if we have an anomaly at moment t, and an outlier detector declares an anomaly at time $t - 1$, we would see this as a full mistake, while it could be that the model detected that something was out of normal before it became clear to the person that has done the labeling.

7. REFERENCES

[Battiti and Campigotto, 2010] Battiti, R. and Campigotto, P. (2010). Reactive search optimization : Learning while optimizing . an experiment in interactive multi-objective optimization.

[Birge and Rozenholc, 2006] Birge, L. and Rozenholc, Y. (2006). How many bins should be put in a regular histogram. *ESAIM: Probability and Statistics*, 10:24âĂŞ45.

[Celik et al., 2011] Celik, M., Dadaser-Celik, F., and Dokuz, A. (2011). Anomaly detection in temperature data using dbscan algorithm.

[Chappell and Taylor, 1993] Chappell, G. J. and Taylor, J. G. (1993). The temporal kohÃÿnen map. *Neural Networks*, 6(3):441 – 445.

[Come et al., 2010] Come, E., Cottrell, M., Verleysen, M., and Lacaille, J. (2010). Self organizing star (SOS) for health monitoring. In *ESANN 2010, 18th European Symposium on Artificial Neural Networks, Bruges, Belgium, April 28-30, 2010, Proceedings*.

[Fawcett, 2006] Fawcett, T. (2006). An introduction to roc analysis. *Pattern Recogn. Lett.*, 27(8):861–874.

[Gaddam et al., 2007] Gaddam, S. R., Phoha, V. V., and Balagani, K. S. (2007). K-means+id3: A novel method for supervised anomaly detection by cascading k-means clustering and id3 decision tree learning methods. *IEEE Trans. on Knowl. and Data Eng.*, 19(3):345–354.

[Gobbi and Konotopska, 2018] Gobbi, J. and Konotopska, K. (2018). Anomaly detection on time series. https://github.com/fruttasecca/anomaly-detection-project.

[Johnson and Lindenstrauss, 1984] Johnson, W. and Lindenstrauss, J. (1984). Extensions of lipschitz maps into a hilbert space. 26:189–206.

[Kangas et al., 1990] Kangas, J. A., Kohonen, T. K., and Laaksonen, J. T. (1990). Variants of self-organizing maps. *IEEE Transactions on Neural Networks*, 1(1):93–99.

[Kashef, 2018] Kashef, R. F. (2018). Ensemble-based anomaly detetction using cooperative learning. In Anandakrishnan, A., Kumar, S., Statnikov, A., Faruquie, T., and Xu, D., editors, *Proceedings of the KDD 2017: Workshop on Anomaly Detection in Finance*, volume 71 of *Proceedings of Machine Learning Research*, pages 43–55. PMLR.

[Kingma and Welling, 2013] Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *CoRR*, abs/1312.6114.

[Kohonen and Honkela, 2007] Kohonen, T. and Honkela, T. (2007). Kohonen network. 2:1568.

[Koskela et al., 1998] Koskela, T., Varsta, M., Heikkonen, J., and Kaski, K. (1998). Temporal sequence processing using recurrent som. In *Knowledge-Based Intelligent Electronic Systems, 1998. Proceedings KES '98. 1998 Second International Conference on*, volume 1, pages 290–297 vol.1.

[Li, 2007] Li, P. (2007). Very sparse stable random projections for dimension reduction in lÎś ($0 < \alpha 2$) norm. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '07, pages 440–449, New York, NY, USA. ACM.

[Ligeza, 2006] Ligeza, A. (2006). *Logical Foundations for Rule-Based Systems*. STUDIES IN COMPUTATIONAL MATHEMATICS. Springer.

[Lin et al., 2003] Lin, J., J. Keogh, E., Lonardi, S., and Chiu, B. (2003). A symbolic representation of time series, with implications for streaming algorithms.

[LinkedIn, 2017] LinkedIn (2017). Anomaly detection and correlation library. https://github.com/linkedin/luminol.

[Liu et al., 2008] Liu, F. T., Ting, K. M., and Zhou, Z.-H. (2008). Isolation forest. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, ICDM '08, pages 413–422, Washington, DC, USA. IEEE Computer Society.

[Malhotra et al., 2016] Malhotra, P., Ramakrishnan, A., Anand, G., Vig, L., Agarwal, P., and Shroff, G. (2016). Lstm-based encoder-decoder for multi-sensor anomaly detection. *CoRR*, abs/1607.00148.

[Malhotra et al., 2015] Malhotra, P., Vig, L., Shroff, G., and Agarwal, P. (2015). Long short term memory networks for anomaly detection in time series.

[Morgan et al., 2007] Morgan, I., Liu, H., Turnbull, G., and Brown, D. (2007). Time discretisation applied to anomaly detection in a marine engine. In *Proceedings of the 11th International Conference, KES 2007 and XVII Italian Workshop on Neural Networks Conference on Knowledge-based Intelligent Information and Engineering Systems: Part I*, KES'07/WIRN'07, pages 405–412, Berlin, Heidelberg. Springer-Verlag.

[Numenta, 2018] Numenta (2018). The numenta anomaly benchmark. https://github.com/numenta/NAB.

[Park et al., 2017] Park, D., Hoshi, Y., and Kemp, C. C. (2017). A multimodal anomaly detector for robot-assisted feeding using an lstm-based variational autoencoder. *CoRR*, abs/1711.00614.

[Pevný, 2016] Pevný, T. (2016). Loda: Lightweight on-line detector of anomalies. *Machine Learning*, 102(2):275–304.

[Sajjipanon and Ratanamahatana, 2009] Sajjipanon, P. and Ratanamahatana, C. A. (2009). A novel fractal representation for dimensionality reduction of large time series data. In *Proceedings of the 13th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, PAKDD '09, pages 989–996, Berlin, Heidelberg. Springer-Verlag.

[Savitzky and Golay, 1964] Savitzky, A. and Golay, M. J. E. (1964). Smoothing and differentiation of data by simplified least squares procedures. *Analytical Chemistry*, 36:1627–1639.

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

[Shahreza et al., 2011] Shahreza, M. L., Moazzami, D., Moshiri, B., and Delavar, M. (2011). Anomaly detection using a self-organizing map and particle swarm optimization. *Scientia Iranica*, 18(6):1460 – 1468.

[Shyu et al., 2003] Shyu, M.-L., Chen, S.-C., Sarinnapakorn, K., and Chang, L. (2003). A novel anomaly detection scheme based on principal component classifier. In *IEEE Foundations and New Directions of Data Mining Workshop, in conjunction with ICDM'03*, pages 171–179.

[Spinosa et al., 2009] Spinosa, E. J., de Leon F. de Carvalho, A. P., and Gama, J. a. (2009). Novelty detection with application to data streams. *Intell. Data Anal.*, 13(3):405–422.

[Tan et al., 2011] Tan, S. C., Ting, K. M., and Liu, T. F. (2011). Fast anomaly detection for streaming data. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Two*, IJCAI'11, pages 1511–1516. AAAI Press.

[Truong and Anh, 2015] Truong, C. D. and Anh, D. T. (2015). An efficient method for motif and anomaly detection in time series based on clustering. *Int. J. Bus. Intell. Data Min.*, 10(4):356–377.

[Wei et al., 2005] Wei, L., Kumar, N., Lolla, V. N., Keogh, E. J., Lonardi, S., and Ratanamahatana, C. (2005). Assumption-free anomaly detection in time series. In *SSDBM*.

[Zhang et al., 2016] Zhang, P., Xiao, Y., Zhu, Y., Feng, J., Wan, D., Li, W., and Leung, H. (2016). A new symbolization and distance measure based anomaly mining approach for hydrological time series:. 13:26–45.