Parameterless Semi-Supervised Anomaly Detection in Univariate Time Series

Oleg Iegorov ✉ and Sebastian Fischmeister

University of Waterloo, Canada
{oiegorov,sfischme}@uwaterloo.ca

Abstract. Anomaly detection algorithms that operate without human intervention are needed when dealing with large time series data coming from poorly understood processes. At the same time, common techniques expect the user to provide precise information about the data generating process or to manually tune various parameters.
We present SIM-AD: a semi-supervised approach to detecting anomalies in univariate time series data that operates without any user-defined parameters. The approach involves converting time series using our proposed Sojourn Time Representation and then applying modal clustering-based anomaly detection on the converted data. We evaluate SIM-AD on three publicly available time series datasets from different domains and compare its accuracy to the PAV and RRA anomaly detection algorithms. We conclude that SIM-AD outperforms the evaluated approaches with respect to accuracy on trendless time series data.

Keywords: Anomaly Detection, Time Series, Motifs, Modal Clustering

1 Introduction

Time series data arise when any data generating process is observed over time. These data are used in diverse fields of research, such as intrusion detection for cyber-security, medical surveillance, economic forecasting, fault detection in safety-critical systems, and many others. One of the main tasks performed on time series data is anomaly detection (AD). Anomalies appear when the underlying process deviates from its normal behavior. For this reason, anomalies can help analysts better understand the data generating process or safely steer it towards nominal operation. As Aggarwal points out in [5], virtually all AD algorithms train a model of the normal patterns in the data and then compute anomaly scores for new observations based on the deviation from these patterns.

With the ever-growing rate of data generation, data mining (DM) and machine learning (ML) tasks need to be performed with minimal human intervention, and anomaly detection is no exception [37]. This task usually requires a combination of expertise in the target application domain as well as in ML and DM algorithms. Experts are rare, and their work is costly. Companies thus need AD tools that do not require manual parameter tweaking and that can work

on raw, unprocessed data without jeopardizing accuracy of the results. Consequently, the demand for parameterless solutions has recently given rise to the field of automated ML (AutoML) [24].

There exist numerous algorithms for anomaly detection in time series data; for example, the surveys in [12] and [20] point to dozens of AD approaches aiming at time series data. Indeed, different application domains typically have their own AD problem formulations that require dedicated techniques. An anomaly detection problem can be described with a set of largely independent *facets*, some of which we list below. The underlined facets define the anomaly detection problem that we address in this paper:

- time series: univariate/multivariate
- anomaly detection type: unsupervised/semi-supervised/supervised
- parameters: none/training/anomaly detection
- anomaly type: subsequence/whole time series
- anomaly score: binary/numeric
- anomaly detection mode: online/offline

This way, we are interested in *online* anomaly detection in *univariate* time series data. Moreover, we expect at least one time series capturing normal (i.e., non-anomalous) system behavior to be available for training: this scenario is often referred to as *semi-supervised* anomaly detection in the literature [12,13,32]. Also, we want to detect anomalous *subsequences* and have *binary* yes/no anomaly scores for subsequences in the target time series. Finally, we want both the training and anomaly detection stages to be completely *parameterless*.

One of the application domains that can benefit from our problem formulation is safety-critical embedded systems used in automotive, avionics, medical device, and other mission-critical industries. Anomaly detection in such contexts is a crucial task since a system malfunction can cause catastrophic damages or worse put human lives in danger. As an example, consider an embedded system running a water treatment plant [18]. The hardware setup consists of various sensors and actuators, each generating a stream of univariate time series data. A malfunctioning system component can incur damage to the facility (e.g., by causing a tank overflow) or contaminate treated water [1]. It is possible to collect data during normal system operation and train a system model on these data. However, the large amount of generated data makes any manual pre-processing or visual analysis of the data complicated or not feasible. Therefore, a parameterless training on unprocessed data is much preferable. Another requirement is to detect anomalies in an online mode allowing preventative measures to be taken immediately. Finally, it is equally important to detect momentary and slowly-evolving malfunctions, both of which can be captured by anomalous subsequences in a time series data stream. Chapter 7 in [12] provides other examples of application domains where our problem formulation is relevant.

In this paper, we propose a solution to the anomaly detection problem defined above: the SIM-AD approach. Since our goal is to make anomaly detection parameterless, we have to assume that all the knowledge about the normal

[1] `https://www.theregister.co.uk/2016/03/24/water_utility_hacked`

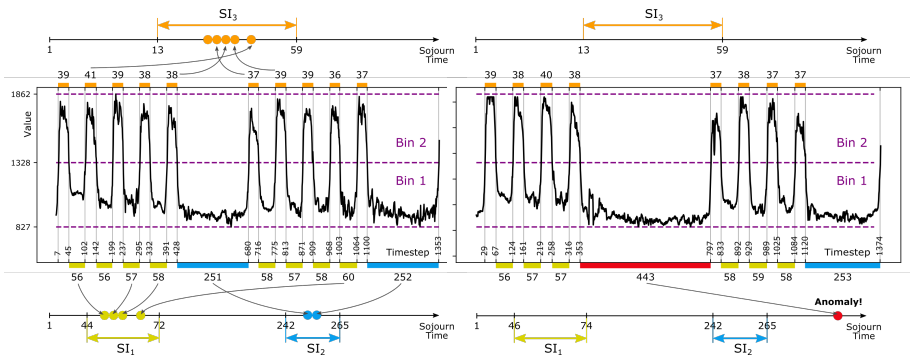
operation of the target system is captured in a single or multiple time series available for training. In data mining, one obtains knowledge about a system by mining data patterns [21]. Therefore, we focus on mining patterns from time series data generated by the target system during normal operation and then look for the mined patterns in the new data that may capture anomalous system behavior. With SIM-AD, we represent a time series pattern as a typical number of timesteps that the time series spends (or *sojourns*) in a value range. To facilitate using SIM-AD, we have developed a web application available at `https://sim-ad.herokuapp.com/`.

The rest of the paper is organized as follows. In Sections 2 and 3, we provide an overview of SIM-AD and review the related work. In Sections 4 and 5, we present the main components of SIM-AD, including our new time series representation and the SIM algorithm. We then compare the accuracy of SIM-AD with the RRA and PAV anomaly detectors in Section 6.

2 Terminology and overview

A univariate time series T of length N is a sequence of N measurements, or *observations*, of some variable collected chronologically: $T = \langle t_1, t_2, \ldots, t_N \rangle$. Observations are usually made at regular time intervals. A subsequence s of a time series T is a contiguous sampling of observations $t_i, \ldots, t_{i+m}$ of length $m < N$, where i is an arbitrary position, such that $1 \leq i \leq N - m + 1$.

With SIM-AD, we view a time series T as a sequence of temporal *states*. A state can be one of two types. A *motif-state* is a state that the system visits multiple times; it corresponds to a time series pattern, or a *motif* [31], i.e., a subsequence of T that looks similar to one or more other subsequences of T. A *nonmotif-state*, on the other hand, is visited only once; it corresponds to a subsequence of T that does not look similar to any other subsequence of T. Detection of motif-states corresponds to the problem of motif discovery, which is an active area of research in the data mining community [40]. Unfortunately, mining motifs of all possible lengths in raw time series poses a scalability problem [17]. A common approach to tackle this problem involves transforming the original time series T into some representation that has fewer data points than T [7]. SIM-AD uses a new time series representation which we call Sojourn Time Representation (STR). An STR of a time series T is a sequence of *sojourn times*. A sojourn time (ST) represents the number of timesteps that T sojourns in one of two contiguous and non-overlapping value ranges called Bin 1 and Bin 2 (see Fig. 1). *With SIM-AD, we assume that the first and last observations of a motif of a time series T correspond to consecutive intersection points between T and the line separating Bin 1 and Bin 2.* By making this assumption, we can map a motif or a nonmotif of a time series T to an ST in the STR of T. However, the same motif in T may correspond to slightly different STs in the STR of T due to random fluctuations in real-world time series data. Moreover, having a group of STs corresponding to the same motif in T and given a new ST, we would like to be able to decide whether the new ST belongs to that group; this way,



(a) Training stage (b) Anomaly detection stage

Fig. 1: Overview of SIM-AD (the shown time series are excerpts from the Power dataset introduced in Section 6.1)

we could conclude whether a subsequence of some time series T' corresponds to one of the states of T. We propose the Sojourn Interval Miner (SIM) algorithm that addresses the two problems mentioned above by clustering STs and finding outlier limits around the clusters. SIM mines *sojourn intervals* from the STR of a time series T. A sojourn interval (SI) is a pair of integers defining the outlier limits around a cluster of STs. Conceptually, an SI plays the role of an STR-equivalent of a time series state. Finally, given a set of SIs and a test time series T', SIM-AD detects anomalies in T' by finding subsequences of T' whose STs fall outside of SIs mined from the training time series T.

Fig. 1 presents an overview of the training and anomaly detection stages of SIM-AD on a pair of excerpts from the Power time series introduced in Section 6.1. During the training stage (Fig. 1a), SIM-AD first computes the STR of the training time series T using Bin 1 and Bin 2. It then clusters STs from each bin independently, obtaining three clusters: two (blue and yellow) for STs from Bin 1 and one (orange) for the STs from Bin 2. This way, SIM-AD represents T as a sequence of three motif-states. For example, a subsequence between timesteps 237 and 295 belongs to one motif-state while a subsequence between timesteps 428 and 680 belongs to another motif-state. Next, SIM-AD mines an SI for each cluster of STs in T. For example, $\text{SI}_3 = [13, 59]$ denotes a range of lengths of subsequences in a test time series falling into Bin 2 that will map to the motif-state represented by the orange STs extracted from the training time series T. Finally, SIM-AD extracts STs from the test time series T' and finds a single ST whose value of 443 falls outside of SIs mined from T (Fig. 1b). Therefore, SIM-AD reports the corresponding subsequence of T' (highlighted in red) as an anomaly.

3 Related work

Practical importance and challenging nature of anomaly detection have led to extensive scientific research, with numerous approaches being proposed and employed in industrial settings [13,32,5]. One of the causes of the proliferation of AD methods is the variability in data types and data sources. As we mentioned in Section 2, SIM-AD uses discrete sojourn times to model time series data. This way, SIM-AD converts the problem of anomaly detection in time series data into the problem of anomaly detection in non-temporal data. Therefore, we review the related work in anomaly detection for both types of data. Also, we only discuss works whose problem formulation has the largest overlap with ours.

It is common to see the terms *anomaly*, *outlier*, *novelty*, or *discord* used interchangeably in anomaly detection literature. In this work, we distinguish anomalies from outliers as in [32]: outlier subsequences contaminate normal time series data, and the goal is to cope with their presence during the training stage. We abstain from using the term discord since it was introduced in the context of unsupervised anomaly detection [26]. Finally, we do not make any distinction between anomalies and novelties.

3.1 Anomaly detection in time series data

Regression-based anomaly detection techniques, such as Long Short-Term Memory (LSTM) networks [23,19], use a window of w consecutive observations t_i, t_{i+1}, ..., t_{i+w-1} to predict the values of the subsequent n observations t_{i+w}, ..., $t_{i+w+n-1}$. They then declare observations whose predicted value significantly deviates from the true value as anomalies. These methods require substantial user involvement in preprocessing raw data and tuning various parameters.

Discord mining algorithms, e.g., RRA [36], address the problem of unsupervised anomaly detection in univariate time series data. The term *discord* denotes the most unusual subsequence within a time series. Early discord mining algorithms [26] require the user to specify the length of discords. Later works allow for mining variable-length discords [36,41] but still require the user to choose the time series discretization parameters. This way, discord mining algorithms have good accuracy when the approximate length of motifs is chosen correctly [37]. Moreover, these algorithms output a ranked list of discords, and the user must choose a threshold on the length of this list to distinguish discords from normal subsequences.

Similarly to SIM-AD, segmentation-based anomaly detectors, e.g., PAV [14] and Gecko [35], segment a time series and consider each segment as a state. They then either learn a Finite State Automaton (FSA) of the mined states [35] or simply memorize the frequency of each state [14] and then detect anomalies as segments that do not match any state in the learned FSA or that appear infrequently. Unlike SIM-AD, the Gecko algorithm requires a database of similar time series for training [12] and takes the minimum length of a segment as a parameter.

Another relevant anomaly detection technique uses Numenta's Hierarchical Temporal Memory (HTM) [6]. This method addresses the problem of online and unsupervised anomaly detection in univariate time series data streams. Based on our understanding of the HTM theory, and confirmed by Numenta's engineers [2], this technique assumes that the first timestamps of motif occurrences are known.

3.2 Anomaly detection in non-temporal data

Statistical anomaly detection techniques fit a mathematical model to the given data instances and then apply an inference test to determine whether a new instance belongs to this model. Instances that have a low probability to be generated from the learned model are declared as anomalies [13]. Parametric approaches assume that normal data are generated from a parametric distribution fully defined by parameters θ [32]. These parameters are then estimated from the training data. Nonparametric approaches, e.g., the ones based on Kernel Density Estimation (KDE) [38] (also called parzen window estimation), do not make assumptions about the distribution from which data are sampled. However, they require a threshold on probability density that separates normal instances from the anomalous ones [15].

Nearest neighbor (NN) techniques detect anomalies by considering distances between data instances. The distances are used either directly, by assigning anomaly scores to instances based on the distance to their kth nearest neighbor [33], or indirectly, by computing local densities of data instances and then declaring instances with smaller densities as anomalies [10]. In both cases, NN approaches require the user to specify the size of the local neighborhood.

Semi-supervised clustering techniques group normal data instances into clusters and then detect anomalies based on the distances between new data instances and the clusters. Some approaches adopt distance-based clustering that requires the number of clusters [16] or a similarity threshold [22]. Moreover, distance-based clustering assumes that clusters have a particular shape [28]. In contrast, density-based clustering does not require the number of clusters nor does it make assumptions about the shape of clusters [28]. However, this type of clustering requires the user to choose a smoothing factor. In DBSCAN-like clustering [11], the smoothing factor is expressed as the minimum cluster size, while in modal clustering [30] it is the kernel bandwidth that is used to compute the KDE of the data. An advantage of modal clustering is that the kernel bandwidth can be estimated automatically using, for example, the Improved Sheather-Jones plug-in rule [8]. We look closer at modal clustering in Section 5.1. Regardless of the underlying algorithm, clustering-based anomaly detection techniques need an anomaly threshold to determine whether a new data instance belongs to some cluster or constitutes an anomaly.

[2] `https://discourse.numenta.org/t/3141`

4 Sojourn Time Representation (STR)

Transforming a time series T into a representation having fewer data points than T is often a prerequisite for scalable motif discovery. At the same time, we want to reduce the size of T without requiring any parameters. Common time series representations, e.g., PLR [25] and SAX [29], require the user to set at least one parameter, such as the length of a sliding window. This motivated us to propose Sojourn Time Representation (STR): a new type of time series representation, similar to the clipped representation [34], that we construct from a time series in a parameterless way.

An STR of a time series T is the run-length encoding of a sequence T_d, where T_d is the result of discretization of T into two bins, Bin 1 and Bin 2. Although the number of bins can be considered as a hard-coded parameter, we justify using two discretization bins in Section 6.5. We define Bin 1 and Bin 2 using non-overlapping intervals $[v_1, v_2)$ and $[v_2, v_3]$ correspondingly, where v_1 is the minimum value in T, v_3 is the maximum value in T, and v_2 is the median value among the unique values in T. This way, $T_d = \langle d_1, d_2, \ldots, d_N \rangle$, where $d_i = 1$ if $t_i \in [v_1, v_2)$ and $d_i = 2$ if $t_i \in [v_2, v_3]$, for $\forall i = 1, 2, \ldots, N$. Next, we apply run-length encoding to T_d. As a result, we obtain a sequence of tuples (r, b), where r refers to the run length, that is, the number of consecutive elements in T_d having the same value, and b refers to the bin number. We call this run-length encoding of T_d the Sojourn Time Representation of T, STR(T). Also, we refer to the run lengths in STR(T) as the *sojourn times* of T with respect to Bin 1 and Bin 2. As an example, the first few elements of the STR of the time series shown in Fig. 1a are $\langle (39, 2), (56, 1), (41, 2), (56, 1), (39, 2), ... \rangle$.

5 Sojourn Interval Miner (SIM)

The SIM algorithm lies at the core of SIM-AD and consists of two steps briefly mentioned in Section 2: clustering sojourn times and mining sojourn intervals given an STR of a time series T. The general problem addressed by SIM is the following: given a set of integers J (i.e., sojourn times extracted from training data) and a new integer j' (i.e., a sojourn time extracted from test data), find whether j' is anomalous with respect to all $j \in J$.

5.1 Parameterless modal clustering of sojourn times

In Section 3.2, we explained why modal clustering is the most relevant approach to clustering outlier-contaminated univariate data: it is possible to make it parameterless without assuming that data are sampled from some exponential family. We next consider modal clustering in detail and then show how we use it for anomaly detection.

In modal clustering, clusters correspond to densely-populated regions of the sample space [30]. A probability density function (PDF) is estimated from a given data sample using a nonparametric technique and then clusters are formed

around the modes in the estimated PDF (hence the name *modal* clustering). The most popular nonparametric approach to density estimation is Kernel Density Estimation (KDE) [8,38]. In KDE, an estimate $\hat{f}$ of the true PDF f at a point x is computed by placing a kernel function K, usually a Gaussian of particular variance h, on each observation X_i ($i \in 1, \ldots, n$) and then summing up the values of all kernels at point x:

$$\hat{f}(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - X_i}{h}\right). \tag{1}$$

Each mode in the estimated PDF $\hat{f}$ then defines a cluster, and observations are assigned to clusters based on the distance to their nearest mode.

Note that the variance of Gaussian kernels, h, commonly called the *bandwidth*, is a parameter that must be estimated separately or chosen manually. Its value greatly influences the number and location of modes in $\hat{f}$ [38]. There exist various methods to estimate h from a data sample, the most popular being the rule-of-thumb and plug-in estimators. These methods find a value of h that maximizes the estimation accuracy of $\hat{f}$, usually based on the asymptotic mean integrated squared error (AMISE). Computing the AMISE requires approximating the functionals of the unknown density f. Most of the bandwidth estimation methods approximate the functionals of f assuming that f is normal, which is usually not the case. One exception is the Improved Sheather-Jones (ISJ) plug-in rule [8]. This method does not use the normal reference rule and thus is completely data-driven. ISJ was shown to accurately estimate densities of unimodal and multimodal distributions from samples of various sizes. Moreover, the numerical procedure used in ISJ is fast when implemented using the Discrete Cosine Transform [8]. We, therefore, use the ISJ method to automatically estimate the bandwidth h from Equation 1. Interestingly, the literature on modal clustering does not focus on the problem of estimating h [30], leaving it to be set manually or using a rule-of-thumb approach which is known to significantly oversmooth multimodal densities. Instead, these works consider multivariate data and focus on finding the modes of $\hat{f}$ in multidimensional spaces, which is a non-trivial task.

Gaussian KDE is known to be sensitive to outliers, resulting in the appearance of "spurious" modes in $\hat{f}$ centered at outlier data instances [9,8]. This behavior is undesirable when the goal is to minimize the AMISE of $\hat{f}$ with respect to the true density f. Therefore, various methods have been proposed to make KDE more robust to outliers by smoothing out the tails of $\hat{f}$ [8]. We, however, use this peculiarity of the Gaussian KDE as a feature. It allows to get more accurate clustering results by assigning outliers to their own, distinct clusters instead of "stretching out" the existing clusters to accommodate the outliers.

5.2 Mining sojourn intervals

Once we grouped the STs from a set J into a set of clusters C, how can we decide whether a new ST $j' \notin J$ is anomalous with respect to J? We provide

an answer to this question using the property of the Gaussian kernel density estimator mentioned above: if an ST j' is an anomaly, then the KDE of $\{J \cup j'\}$ will have a mode centered at j'. Indeed, the convex shape of the Gaussian kernel guarantees that the KDE of $\{J \cup j'\}$ will have a "bump" around j' when the distance from j' to the closest cluster $c \in C$ is sufficiently large given the kernel bandwidth h estimated from J.

When we perform online anomaly detection with SIM-AD, we must analyze STs of the incoming time series T' in real-time. We avoid computing the KDE of $\{J \cup j'\}$ for each ST j' from T' by finding sojourn intervals (SIs) around clusters C during the training stage. Given a cluster $c \in C$, we apply a binary search to find the closest ST g, such that the KDE of $\{c \cup g\}$ has a bump on g, using the bandwidth h estimated from normal STs J. We perform this search on both sides of the cluster c and define an SI using the found pair of outlier STs g_1 and g_2. This allows us to detect anomalies by simply comparing the values of the incoming STs to the SIs and reporting the subsequences of T' that correspond to STs that fall outside of the SIs.

6 Experiments

We compare the accuracy of SIM-AD, PAV, and RRA anomaly detectors on three publicly-available time series datasets previously used in the anomaly detection literature. We evaluate the accuracy of the detectors with a pair of F-scores.

6.1 Datasets

Power The dataset contains 35,040 measurements of power consumption of a Dutch research facility during the entire year of 1997 [4]. The measurements capture the aggregate power consumption during fifteen-minute time intervals. The dataset was used for anomaly detection in [26,36,39]. There are 44 normal and 8 anomalous weeks in the dataset. Fig. 2 shows power consumption during a normal week and an anomalous week. During a normal week, we can observe five consecutive peaks and valleys corresponding to the consumption on workdays, while the longer valleys are observed on weekends when relatively little power was consumed.

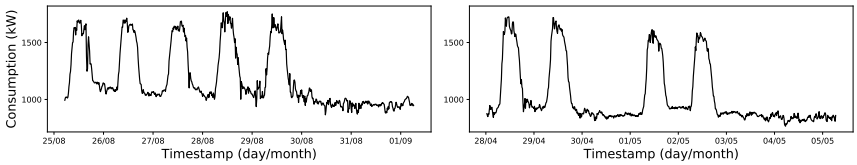


Fig. 2: A normal week (left) and an anomalous week (right) in the Power dataset

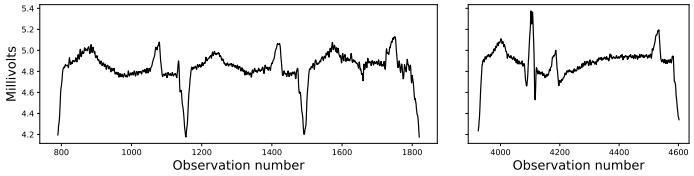


Fig. 3: Three normal heartbeats (left) and an anomalous heartbeat (right) in the ECG dataset

ECG The electrocardiogram (ECG) dataset from the MIT-BIH Arrhythmia Database [1] was previously used for anomaly detection in [26,36,39]. The dataset contains 21,600 measurements of electrical potential between two points on the body surface of a patient during 55 heartbeats. Individual heartbeats have a similar shape, but their duration varies slightly. There are three anomalous heartbeats in the dataset reported by a cardiologist [26]. Fig. 3 shows examples of three normal consecutive heartbeats and an anomalous heartbeat.

LIT101 Finally, we consider time series data generated by the LIT101 sensor of the Secure Water Treatment (SWaT) testbed [3] (version 0). Datasets from this testbed were used for anomaly detection in [19,27]. SWaT is a scaled down version of a real-world industrial water treatment plant [18] producing five gallons/minute of double-filtered water. It consists of 51 sensors and actuators that control its six-stage filtration process. The LIT101 sensor measures the raw water tank level (in millimeters). The data collected from the testbed comprise eleven days of continuous operation. The researchers who had built the SWaT testbed ran it normally during the first seven days and then launched attacks on different components during the remaining four days. This way, the LIT101 dataset consists of a normal time series with 496,800 measurements and an attack time series with 449,919 measurements. A total of 36 attacks were launched during the last four days of SWaT operation. Only five attacks targeted the LIT101 sensor. Fig. 4 shows a part of the normal LIT101 time series, where the raw water tank was filled and emptied seven times, as well as two anomalous parts during the attacks on the LIT101 sensor. Some of the other 31 attacks also affected the LIT101 measurements. Moreover, the time series sometimes contains

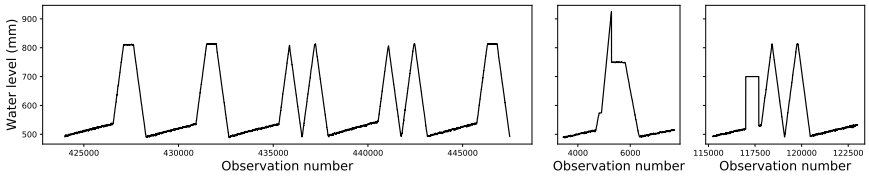


Fig. 4: Normal (left) and two anomalous (center and right) subsequences from the LIT101 dataset

unusual subsequences before or after the launched attack. We manually annotate these unusual parts but do not mark them as anomalies. This way, if an anomaly detector reports one of these parts of the attack time series as an anomaly, we ignore it and do not consider it as a true positive or as a false positive.

6.2 Algorithms

To the best of our knowledge, the Pattern Anomaly Value (PAV) and Rare Rule Anomaly (RRA) anomaly detectors are the only existing algorithms that support parameterless AD in time series data. However, unlike SIM-AD, both of them aim at unsupervised AD and both approaches require the user to choose a threshold on the number of reported anomalies. We next explain how we use these detectors in our experiments.

RRA is a discord-mining algorithm [36]. Its implementation is freely available as part of the GrammarViz 3.0 tool [37,2]. RRA mines variable-length discords and requires setting three SAX-discretization parameters: sliding window size w, PAA word size p, and alphabet size a. We set these parameters using the semi-automated method from [37]. This method requires choosing a *parameter learning interval*, that is, a training subsequence that captures the normal behavior of the generative process. The user must also set the ranges of acceptable parameter values. The authors of RRA propose setting the range for p from 2 to 50, for a from 2 to 15, and for w from 10 to the doubled length of time series motifs. We followed these guidelines and set the upper value of the range for w to 500. We also configured RRA to use the Re-Pair grammar inference algorithm, EXACT numerocity reduction strategy, and set the normalization threshold to 0.01.

PAV mines *linear patterns* in univariate time series data. If observations are equally spaced in time (which is true for our experimental data), then a linear pattern is simply the value difference between consecutive observations. PAV reports the n linear patterns with the smallest number of occurrences as anomalies, where n is a user-specified threshold.

6.3 Accuracy metrics

We evaluate the accuracy of anomaly detectors using a pair of F-scores, where F-score $= 2 \cdot \mathrm{P} \cdot \mathrm{R}/(\mathrm{P} + \mathrm{R})$, P is the precision, and R is the recall. The first F-score measures the classification accuracy while the second one measures the coverage accuracy of a detector. We next explain why two F-scores are necessary and how we calculate precision and recall in both cases.

The classification F-score (F-class) measures how well the detected intervals classify the anomalous parts of a time series. The detected intervals that overlap at least one anomalous interval are true positives. This way, we compute precision as $\mathrm{P} = \mathrm{TP}/\mathrm{D}$, and recall as $\mathrm{R} = \mathrm{TP}/\mathrm{A}$, where TP is the number of true

positives, D is the number of detected intervals, and A is the number of anomalous intervals. F-class, however, does not take into account the lengths of the detected intervals. Indeed, a detector which reports an interval that covers the entire time series will have a perfect F-class = 1, since this interval necessarily overlaps all anomalous intervals.

The coverage F-score (F-cover) measures how well the detected intervals cover the anomalous intervals. In this case, a true positive is a timestamp of an anomalous interval covered by one or more detected intervals. We thus have precision P = TP/C and recall R = TP/A, where TP is the number of true positives, C is the number of timestamps in all detected intervals, and A is the number of timestamps in all anomalous intervals. F-cover alone, however, does not say how many anomalies were detected. Indeed, a detector that reports an interval that fully covers a long anomalous interval but leaves shorter anomalies uncovered will still have a large F-cover. Therefore, we need both F-scores to evaluate the accuracy of an anomaly detector.

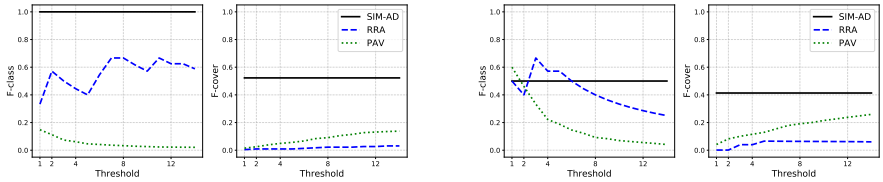
6.4 Results

We report the accuracies of the SIM-AD, RRA, and PAV anomaly detectors applied on the three datasets presented in Section 6.1. For each dataset, we show classification and coverage F-scores of the detectors on two plots. The X-axis on these plots indicates the number of top discords for RRA and the number of smallest unique frequencies of linear patterns for PAV that the user chooses as an anomaly threshold. SIM-AD has constant F-class and F-cover since its number of detected anomalies does not depend on any threshold.

As a preliminary step, we split the Power and ECG datasets into a training set and a test set, such that the training set does not include any labeled anomalies. The LIT101 dataset comes already partitioned into such two sets. We use the entire training set as a parameter learning interval for the RRA detector and run the PAV detector only on the test set.

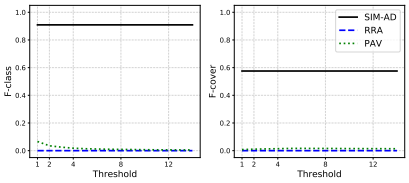
Power SIM-AD dominates other detectors both in classification and in coverage accuracies (Fig. 5a). RRA performs much worse on the Power dataset than it has been reported in [36]. In fact, the parameter learning method returns clearly not optimal values of $w = 10$, $p = 2$, $a = 7$ when the entire training set is used as the parameter learning interval. Finally, the anomalies reported by PAV appear to be random. Indeed, PAV aims at detecting *point anomalies* [12] (i.e., unusual spikes/dips in observed values), while the anomalies in the Power time series are of *collective* type [12] (i.e., unusual sequences of observed values).

ECG Both RRA and PAV detectors show better classification F-score than SIM-AD for some values of the threshold. For the RRA detector, the parameter learning method returns a very low value for w, and the top discords do not match the ones reported in [36], where the learning interval was chosen manually. SIM-AD detects only one out of three anomalies and does not report any false



(a) Power dataset

(b) ECG dataset



(c) LIT101 dataset

Fig. 5: Accuracy results for the SIM-AD, RRA, and PAV anomaly detectors

positives. The inferior classification F-score of SIM-AD on this dataset is due to a "noisy" trailing subsequence in the ECG time series where the value range of the heartbeats changes drastically. Incidentally, removing the noisy subsequence from the training set allows SIM-AD to achieve a perfect F-class of 1. Although having an inferior F-class, SIM-AD covers the anomalous intervals in this dataset much better than the other two detectors, as shown by the F-cover plot in Fig. 5b.

LIT101 SIM-AD outperforms both the RRA and PAV detectors on this dataset. Our approach detects all five anomalies and a single short false positive (Fig. 5c). In the case of the RRA detector, none of the top 24 discords overlap anomalous subsequences in the attack LIT101 dataset. Similarly to the Power dataset, the parameter selection method learned a very small window length $w = 10$, and the detector failed to combine observations from short windows into longer patterns. Although PAV correctly detects all five anomalies starting from the threshold of 1, it also returns numerous false positives that bring its F-scores down.

6.5 Discussion

Experimental results show that SIM-AD outperforms both the RRA and PAV anomaly detectors on two out of three considered time series datasets. We can explain the inferior accuracy of SIM-AD on the ECG dataset by the presence of a trend towards the end of the training time series. Indeed, SIM-AD assumes that a motif's occurrences are not shifted in the value range. Notice that without additional knowledge about a system, our approach has no way of knowing whether two subsequences mapped to different sojourn intervals correspond to the same, although shifted in value, motif. We can summarize this limitation in the following way: SIM-AD performs poorly on time series data having a trend.

In future work, we will address this limitation by automatically fitting the line that defines discretization bins to the time series trend.

Another limitation of SIM-AD concerns time series motifs that map to a *sequence* of sojourn intervals. As an example, consider the weekly motif in the Power dataset (see Fig. 1). The motif consists of a sequence of peaks and valleys corresponding to a sequence of sojourn intervals mined by SIM-AD: $\langle \mathrm{SI}_3, \mathrm{SI}_1, \mathrm{SI}_3, \mathrm{SI}_1, \mathrm{SI}_3, \mathrm{SI}_1, \mathrm{SI}_3, \mathrm{SI}_1, \mathrm{SI}_3, \mathrm{SI}_2 \rangle$. The current version of our approach does not mine such composite motifs, and we leave this task as future work.

Why did we choose to use two discretization bins to compute the STR of a time series? Notice that identical subsequences will have identical STRs, hence, will be represented with the same set of SIs. If similar subsequences have small variations in them, their STRs will be different, but their set(s) of SIs are likely to be the same. Indeed, the more occurrences a motif has in a time series, the more STs it produces, the more robust to variations the mined SIs become. It is possible, however, that SIM-AD maps structurally-different subsequences that spend similar amounts of time in Bin 1 and Bin 2 to the same set of SIs. If one of these subsequences corresponds to an instance of a motif while the other one does not, the anomaly detection stage of SIM-AD will incorrectly report both subsequences as instances of the same motif, i.e., it will produce a false negative. We plan to address this issue in the future work by recursively mining fine-grained structure of subsequences mapped to the same set of SIs.

The computational complexity of the training stage of SIM-AD includes the time needed to (a) construct an STR of the training time series; (b) find the optimal bandwidth of the KDE kernel; (c) compute the KDE of the sojourn times. The first step has linear complexity with respect to the length of the training time series, while the other two operations depend on the length of the STR and have efficient implementations that use Discrete Cosine Transform [8] and Fast Fourier Transform [38]. Indeed, training each of the considered time series datasets took SIM-AD less than a second on a single core of an Intel Core i5-6200U CPU. The anomaly detection stage of SIM-AD needs to run only two relational operators on each incoming observation. Therefore, it is possible to perform anomaly detection on data streams in real-time.

7 Conclusion

In this paper, we addressed the problem of parameterless anomaly detection in univariate time series data. To this end, we proposed and evaluated SIM-AD: a semi-supervised anomaly detection approach that does not require any parameters. At the core of SIM-AD lies a modal clustering-based anomaly detection approach that uses kernel density estimation and the Improved Sheather-Jones plug-in bandwidth estimator. We showed that SIM-AD outperforms the relevant anomaly detection algorithms on trendless time series data. In future work, we plan to address the three limitations of SIM-AD mentioned in Section 6.5.

References

1. ECG dataset, http://www.cs.ucr.edu/~eamonn/discords/mitdbx_mitdbx_108.txt, (2nd column)
2. Grammarviz 3.0, https://grammarviz2.github.io/grammarviz2_site/
3. LIT101 dataset, https://itrust.sutd.edu.sg/itrust-labs_datasets/
4. Power dataset, http://www.cs.ucr.edu/~eamonn/discords/power_data.txt
5. Aggarwal, C.C.: Outlier analysis. In: Data mining. pp. 237–263. Springer (2015)
6. Ahmad, S., Lavin, A., Purdy, S., Agha, Z.: Unsupervised real-time anomaly detection for streaming data. Neurocomputing **262**, 134–147 (2017)
7. Bettaiah, V., Ranganath, H.S.: An analysis of time series representation methods: data mining applications perspective. In: Proceedings of the 2014 ACM Southeast Regional Conference. pp. 16:1–16:6 (2014)
8. Botev, Z., Grotowski, J., Kroese, D.: Kernel density estimation via diffusion. The annals of Statistics **38**(5), 2916–2957 (2010)
9. Breiman, L., Meisel, W., Purcell, E.: Variable kernel estimates of multivariate densities. Technometrics **19**(2), 135–144 (1977)
10. Breunig, M.M., Kriegel, H.P., Ng, R.T., Sander, J.: LOF: identifying density-based local outliers. In: Proceedings of the International Conference on Management of Data. vol. 29, pp. 93–104. ACM (2000)
11. Campello, R.J., Moulavi, D., Zimek, A., Sander, J.: Hierarchical density estimates for data clustering, visualization, and outlier detection. ACM Transactions on Knowledge Discovery from Data (TKDD) **10**(1), 5:1–5:51 (2015)
12. Chandola, V.: Anomaly detection for symbolic sequences and time series data. Ph.D. thesis, University of Minnesota (2009)
13. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: A survey. ACM computing surveys (CSUR) **41**(3), 15 (2009)
14. Chen, X., Zhan, Y.: Multi-scale anomaly detection algorithm based on infrequent pattern of time series. Journal of Computational and Applied Mathematics **214**(1), 227–237 (2008)
15. Chow, C.: Parzen-window network intrusion detectors. In: Proceedings of the 16 th International Conference on Pattern Recognition (ICPR). pp. 385–388. (2002)
16. Clifton, D., Bannister, P., Tarassenko, L.: A framework for novelty detection in jet engine vibration data. In: Key engineering materials. vol. 347, pp. 305–310 (2007)
17. Gao, Y., Lin, J.: HIME: discovering variable-length motifs in large-scale time series. Knowl. Inf. Syst. **61**(1), 513–542 (2019)
18. Goh, J., Adepu, S., Junejo, K.N., Mathur, A.: A dataset to support research in the design of secure water treatment systems. In: Proceedings of the 11th International Conference on Critical Information Infrastructures Security (CRITIS). pp. 88–99. Springer (2016)
19. Goh, J., Adepu, S., Tan, M., Lee, Z.S.: Anomaly detection in cyber physical systems using recurrent neural networks. In: 18th IEEE International Symposium on High Assurance Systems Engineering (HASE). pp. 140–145 (2017)
20. Gupta, M., Gao, J., Aggarwal, C.C., Han, J.: Outlier detection for temporal data: A survey. IEEE Trans. Knowl. Data Eng. **26**(9), 2250–2267 (2014)
21. Han, J., Pei, J., Kamber, M.: Data mining: concepts and techniques. Elsevier (2011)
22. He, Z., Xu, X., Deng, S.: Discovering cluster-based local outliers. Pattern Recognition Letters **24**(9-10), 1641–1650 (2003)
23. Hundman, K., Constantinou, V., Laporte, C., Colwell, I., Soderstrom, T.: Detecting spacecraft anomalies using LSTMs and nonparametric dynamic thresholding.

In: Proceedings of the 24th ACM International Conference on Knowledge Discovery & Data Mining (KDD). pp. 387–395. ACM (2018)
24. Hutter, F., Kotthoff, L., Vanschoren, J.: Automated Machine Learning. Springer (2019)
25. Keogh, E., Chu, S., Hart, D., Pazzani, M.: Segmenting time series: A survey and novel approach. In: Data mining in time series databases, pp. 1–21. (2004)
26. Keogh, E., Lin, J., Fu, A.: HOT SAX: efficiently finding the most unusual time series subsequence. In: Fifth IEEE International Conference on Data Mining (ICDM). pp. 226–233 (2005)
27. Kravchik, M., Shabtai, A.: Detecting cyber attacks in industrial control systems using convolutional neural networks. In: Proceedings of the 2018 Workshop on Cyber-Physical Systems Security and Privacy. pp. 72–83. ACM (2018)
28. Kriegel, H.P., Kroger, P., Sander, J., Zimek, A.: Density-based clustering. Wiley Interdiscip. Rev. Data Min. Knowl. Discov. **1**(3), 231–240 (2011)
29. Lin, J., Keogh, E., Wei, L., Lonardi, S.: Experiencing SAX: a novel symbolic representation of time series. Data Min. Knowl. Discov. **15**(2), 107–144 (2007)
30. Menardi, G.: A review on modal clustering. International Statistical Review **84**(3), 413–433 (2016)
31. Patel, P., Keogh, E.J., Lin, J., Lonardi, S.: Mining motifs in massive time series databases. In: Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM). pp. 370–377 (2002)
32. Pimentel, M.A., Clifton, D.A., Clifton, L., Tarassenko, L.: A review of novelty detection. Signal Processing **99**, 215–249 (2014)
33. Ramaswamy, S., Rastogi, R., Shim, K.: Efficient algorithms for mining outliers from large data sets. In: Proceedings of the International Conference on Management of Data. vol. 29, pp. 427–438. ACM (2000)
34. Ratanamahatana, C., Keogh, E., Bagnall, A.J., Lonardi, S.: A novel bit level time series representation with implication of similarity search and clustering. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining. pp. 771–777. Springer (2005)
35. Salvador, S., Chan, P.: Learning states and rules for detecting anomalies in time series. Appl. Intell. **23**(3), 241–255 (2005)
36. Senin, P., Lin, J., Wang, X., Oates, T., Gandhi, S., Boedihardjo, A., Chen, C., Frankenstein, S.: Time series anomaly discovery with grammar-based compression. In: Proceedings of the 18th International Conference on Extending Database Technology (EDBT). pp. 481–492 (2015)
37. Senin, P., Lin, J., Wang, X., Oates, T., Gandhi, S., Boedihardjo, A.P., Chen, C., Frankenstein, S.: Grammarviz 3.0: Interactive discovery of variable-length time series patterns. ACM Transactions on Knowledge Discovery from Data (TKDD) **12**(1), 10 (2018)
38. Silverman, B.W.: Density Estimation for Statistics and Data Analysis. Chapman & Hall, London (1986)
39. Singh, A.: Anomaly Detection for Temporal Data using Long Short-Term Memory (LSTM). Master's thesis, KTH Information and Communication Technology, Sweden (2017)
40. Torkamani, S., Lohweg, V.: Survey on time series motif discovery. Wiley Interdiscip. Rev. Data Min. Knowl. Discov. **7**(2) (2017)
41. Wang, X., Lin, J., Patel, N., Braun, M.: Exact variable-length anomaly detection algorithm for univariate and multivariate time series. Data Min. Knowl. Discov. **32**(6), 1806–1844 (2018)