Anomaly Detection in Time Series

Heraldo Borges[1], Reza Akbarinia[1], Florent Masseglia[1]

INRIA & LIRMM, Univ. Montpellier, Montpellier, France
{heraldo.pimenta-borges-filho,reza.akbarinia,florent.masseglia}@inria.fr

Abstract. Data mining has become an important task for researchers in the past few years, including detecting anomalies that may represent events of interest. The problem of anomaly detection refers to discovering the patterns that do not conform to expected behavior. This paper analyzes recent studies on the detection of anomalies in time series. The goal is to provide an introduction to anomaly detection and a survey of recent research and challenges. The article is divided into three main parts. First, the main concepts are presented. Then, the anomaly detection task is defined. Afterward, the main approaches and strategies to solve the problem are presented.

Keywords: Time series; Anomaly detection

1 Introduction

In many real-world applications including load demand forecasting [25], geography [10], human activity recognition [14], stock return [16] and others [6], the data is collected in the form of time series. Anomaly detection in this type of data refers to discovering any abnormal behavior within the data encountered in a specific time interval. Anomaly detection has been widely used in several application areas. For instance, cardiologists are interested in identifying anomalous parts of ECG signals to diagnose heart disorders [24]. Economists are interested in anomalous parts of share prices to analyze and build economic models[18]. Meteorologists are interested in anomalous parts of weather data to predict future consequences [26].

Consistent with [7], we can define an anomaly as a point in time where the system's behavior is unusual and significantly different from previous normal behavior. An anomaly can mean a change with positive consequences, such as an increase in the number of sales on a sales website, and a change with negative consequences, such as a change in the rotation frequency of a jet engine's turbine. In either case, whether positive or negative, the similarity is that anomalies must often be considered and carefully taken into account.

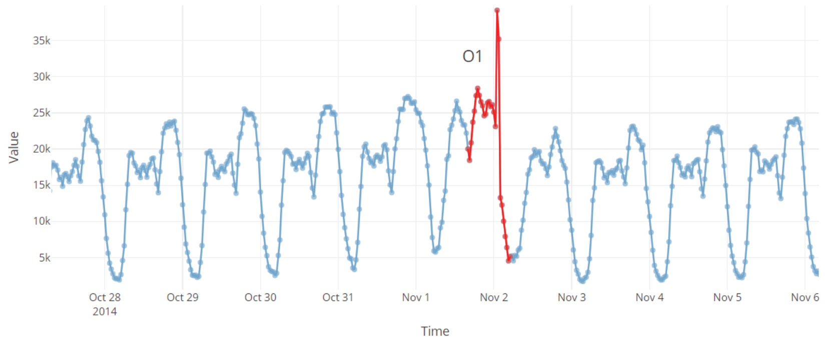
Anomalies can be temporal if the temporal sequence of data is relevant; i.e., the data maybe anomalous only in a specific temporal context. Temporal anomalies are often subtle and hard to detect in real applications. Detecting temporal anomalies in applications is valuable as they can allow reactions (*e.g.* serve as an early warning for problems with the underlying system, or decision support for adjustments in order to benefit from the anomaly).

This article aims to provide an organized overview of existing research to detect anomalies in time series. The objective is to provide an understanding of the problem of detecting anomalies and how existing techniques are related. The next section provides some definitions. In Section 3, a brief overview of time series anomaly detection techniques is reported.

2 Definitions

A time series is an ordered list of observations $X =< x_1, ..., x_n >$, where each value x_p is an observation collected at time p. Let $S_p = x_p, x_{p+1} + ... + x_{p+m-1}$ be the subsequence of size m, which starts at the position p of the time series X. Figure 1 presents an example of the time series, where $O1$ is an example of a subsequence.

Fig. 1. Time series



2.1 Time-series patterns

We briefly describe some main properties of time series important for anomaly detection methods.

We can say that a time series has a **trend** if the observed mean μ is not constant, but varies over time. The trend can have a linear behavior. The time series in Figure 2 has a positive trend over the years.

The **seasonality** is the recurrence of oscillations periodically. A time series is seasonal if it is influenced by seasonal aspects such as year, period, or year. Thus, there is always a fixed-term, where the oscillation reappears. Figure 3 shows a seasonal time series with annual house sale values.The seasonality can be observed, since the real estate market is generally not active at the beginning of the year and sales generally increase in the middle of the year, decreasing again at the end of the year.

Fig. 2. Sample time-series showing a constant trend

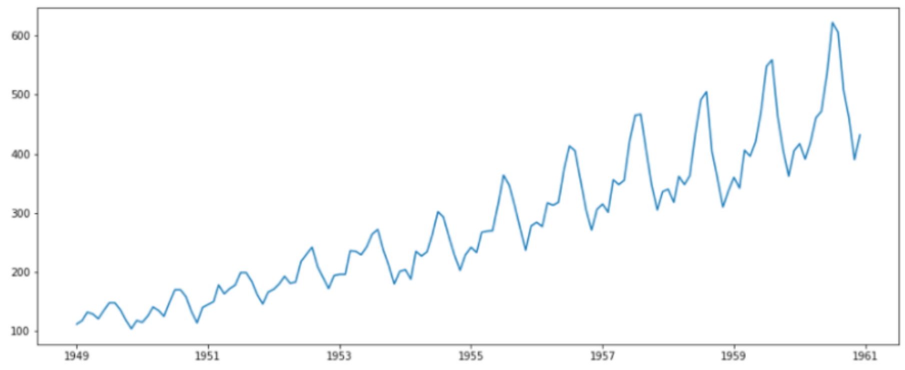
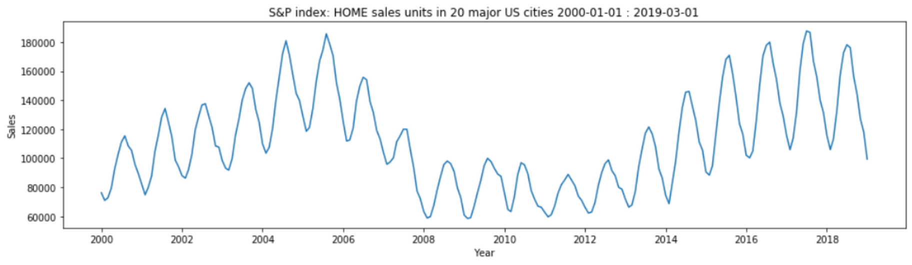


Fig. 3. Sample time-series showing yearly home sales



A **stationary** time series is one whose properties do not depend on the instant at which the series is observed [12]. Thus, time series that have trends, or seasonality, are not stationary, as these properties affect the value of the time series at different times. On the other hand, a series of white noise is stationary because it doesn't matter when it's observed, it looks the same at all times.

The given definition assumes that a stationary series will have the following characteristics:

1. The verification of *constant mean*, shown that there is no trend in the series.
2. Throughout the series period, there is a *constant autocorrelation*
3. The series does not show periodic movements, indicating that there is no *seasonality*

2.2 Anomalies

Several studies have attempted to define the condition of anomalous data. In [11], Hawkins defined anomaly as: "an observation that deviates so much from other observations that it raises the suspicion that a different mechanism generated it." In [23], Barnett and Lewis present another interpretation of this definition: "an outlier is an observation (or subset of observations) that appears to be inconsistent with the rest of this dataset." We can then generalize the definition of anomalies as described in [6] *"patterns in data that do not conform to a well-defined notion of normal behavior"*.

Activities such as a system failure, error in the capture of information by a sensor, or even earthquakes are examples of reasons for anomalies in the data. Even in different scenarios, all of these anomalies produce important information for analysts. The importance of analyzing and understanding these occurrences is an important point for detecting anomalies.

Anomaly types We can understand the problem of detecting anomalies for time series to identify outliers concerning some standard or usual signal. While there are many types of anomalies, let's focus on the most important types [6].

1. **Point anomaly**: A point anomaly happens when an observation or a set of several individual observations diverges from the other set of observations. One can observe a punctual discontinuity in a period of time outside the normality of the values. For example, a purchase with a very high amount within a set of credit card transactions. This type is closest to the concept presented in [11].
2. **Contextual anomaly**: An apparently normal observation may diverge when analyzed within a specific context, such as a period of time or a certain location. For example, low temperatures are normal in the winter period but are contextual anomalous when they occur in the summer, where high temperatures are expected.
3. **Collective anomalies**: It occurs when no individual observation is an anomaly in itself, but a group of these observations, when combined, exhibit behavior that diverges from the rest of the data. In the stock market, for example, the fall in the price of an asset does not deviate significantly from the normal range, but the combination of successive falls indicates a collective anomaly.

The contextualization and understanding of the type of anomaly to be identified can help choose the best detection model.Each approach is aimed at certain types of anomalies, having certain advantages and disadvantages according to the subjective definition of normal observations and anomalies in a given context.

Anomaly Detection Several works provide an overview of techniques for detecting outliers [1]. Anomaly detection is also known as outlier detection, event detection, novelty detection, drift detection, change point detection, failure detection, or misuse detection [6]. These different approaches and terms converge towards the same purpose: detecting abnormal patterns that deviate from the rest of the data, called anomalies or outliers.

Anomaly detection is a broad field that has been studied in the context of a large number of application domains, such as intrusion detection, fraud detection, failure detection in industry 3.0, system health monitoring, event detection in network sensors, and detection of disturbances in the ecosystem [27]. Due to the great diversity of types and techniques, it is necessary to understand the properties of anomalies:

1. Temporality: anomalies can be temporal when associated with some temporal information. These are usually anomalies found in time series as well as streaming data and medical data.
2. Labeling: Some datasets have a set of labeled anomalies, where the location of each instance in the dataset is indicated. In general, this set of labeled anomalies represents only a subset of the anomalies present. Annotated anomalies are used by supervised learning methods, where the method learns through examples to detect new anomalies.
3. Dimensionality: anomalies can present univariate data when represented in only one dimension or multivariate when presenting a set of variables. Multivariate data is generated, for example, on sensors.

Anomaly Detection in Time Series Detecting anomalies in temporal data differs from detecting non-temporal data. In non-temporal data, as in spatial data, it is possible to detect an anomaly by its location in less dense or more distant regions. Another way is to calculate the deviation from the anomalous observations to the rest of the data. In this domain, it is understood that the observations are independent of each other.

When we look at the temporal data sets, we can see that this premise is not true, as the observations are not completely independent of each other. Previous observations may influence new observations. Thus, the variation between values tends to be smooth and gradual, without major variations.

Take, for example, a list of ten values with the pricing of the value of an asset in the stock market, measured every one minute: \$12, \$15, \$16, \$18, \$17, \$42, \$43, \$18, \$19, \$18. Since these points are dependent on each other, we can see that the sudden increase to the value of \$42 could be an anomaly.

Anomaly detection methods for time series can be divided into two main categories [1]. There are methods that are based on time series prediction, which represent most of the statistical methods [27]. Another category are methods that are based on unusual forms of the time series. We will present some of these approaches in the next section.

3 Anomaly Detection Approaches For Time Series

In this section, we discuss various anomaly detection applications. We organize the approaches into three categories: statistics-based, cluster-based, and matrix profile-based.

3.1 Statistical Based Approaches

We present some stationary linear autoregressive models. Most of the methods used in time series are linked to linearity and stationarity. A process is stationary when its mean, variance and autocovariance are invariant with respect to time, such as, for example, the Auto-Regressive (AR) and Moving Averages (MA) and Auto-Regressive Moving Averages (ARMA) models.

Autoregressive Model (AR) In an autoregressive model, denoted by AR, we project the variable of interest based on the linear combination of a finite set of values before to the variable (independent variables) and an error value [13].

Thus, an autoregressive model of order p, $AR(p)$, can be written as:

$$Z_t = \varphi_1 Z_{t-1} + \varphi_2 Z_{t-2} + \ldots + \varphi_p Z_{t-p} + a_t \tag{1}$$

where a_t is white noise and the terms $Z_{t-1}, Z_{t-2}, \ldots, Z_{t-p}$ are terms that are independent of a_t. This model assumes that the current value of the series is a linear combination of the past p values of the series and a white noise a_t. We refer to Equation 1 as an **AR(p) model**, an autoregressive model of order p.

Autoregression-based techniques were extended to detect contextual anomalies in time series. Initially, the AR(p) model is fitted to the data. Then, for each Z_t instance, the residual of the instance is determined for calculating the anomaly score. This residual is the instance value that falls outside the regression model. Thus, the anomaly is scored as the difference between the estimated value and the residual value [6].

Moving Average Model (MA) Instead of using, as in Autoregressive Model (AR), the p observations prior to the variable of interest, a Moving Average Model (MA) considers the last q errors $\epsilon_t, \epsilon_{t-1}, \ldots, \epsilon_{t-q}$ prediction in a regression model. We refer to this model as an **MA (q)**, a moving average model of order q [13]:

$$Z_t = c + \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \ldots + \theta_q \epsilon_{t-q} \tag{2}$$

where the $\theta_1, \theta_2, \ldots, \theta_q$ are the parameters of the model, c is the expectation of Z_t (often assumed to equal 0), and the $\epsilon_t, \epsilon_{t-1}, \ldots, \epsilon_{t-q}$ are the white noise error terms.

It is important to note that this is a moving average model used to predict future values, unlike smoothing models, where average values are used to eliminate some of the randomnesses of the data. Therefore, we consider that each Z_t comes to be considered a weighted moving average of the latest forecast errors.

In the model of Equation 2, the moving average of the previous q observations, also called the MA of order q, is considered to calculate the current estimated value. The next step includes checking that the estimated value is within the predefined confidence band. Confidence band is the interval defined as a multiple of the standard deviation of the moving average of the previous period. If the value is greater than the maximum expected value (higher confidence band), it is then flagged as an anomaly.

Autoregressive Moving Average Model (ARMA) Autoregressive Moving Average (ARMA) models play a key role in time series modeling, being widely used for time series analysis and prediction. The ARMA model is a combination of the Autoregressive (AR) model, which describes the analytical component of the signal, and the Moving Average (MA) model, which describes the noise component of the signal. Its linear structure also presents a substantial simplification of linear prediction. Compared to pure AR or MA models, ARMA models provide the most efficient linear model in stationary time series, given the ability to model the unknown process with minimal parameterization [28].

An autoregressive and moving average (ARMA) model denoted by ARMA(p, q), combining the AR(p) and MA(q) models, can be written in a single equation in the form:

$$Z_t = c + \epsilon_t + \theta_1\epsilon_{t-1} + \theta_2\epsilon_{t-2} + \ldots + \theta_q\epsilon_{t-q} + \varphi_1 Z_{t-1} + \varphi_2 Z_{t-2} + \ldots + \varphi_p Z_{t-p} \quad (3)$$

Let Z_t be the sign Z at time t. We assume that Z_t linearly depends on the previous values $Z_{t-1}, \ldots, Z_{t-p}$ where p is the order of the autoregression. Thus, we have from AR that $\varphi_1, \ldots, \varphi_p$ are auto-regression parameters that can be learned from historical data and used to predict or find similar time series, c is a constant, and $\epsilon_t\ N(0, \sigma^2)$ is Gaussian noise. From MA we have $\theta_1, \ldots, \theta_q$ are learnable parameters of the model, μ is the expectation of Z_t and $\epsilon_{ti}\ N(0, \sigma^2)$ are the terms of Gaussian noise.

Using historical data, we can select p and q in ARMA(p,q) model and learn the model coefficients θ and ϕ based on which we can make a future prediction. A substantial deviation from the prediction result in a time-series anomaly. We can define substantial as two standard deviation from a moving average of Z. The model parameteters θ and ϕ are learned via Maximum Likelihood Estimation (MLE). Given the simplicity of the model it may be sufficient for many applications.

The ARMA model has the advantage of having a clear mathematical and statistical basis. However, there are some disadvantage, like the selection of the best values for p and q in order to find a better model for detecting anomalies [4]. If high values are used, the generated model will find a large number of false-negatives, thus, a low number of anomalies. In contrast, using small values for p and q, the generated model will identify a high number of false positives, that is, label a large number of observations as anomalies, which actually are not. There are some ways to adjust the model, such as the use of correlograms, the use of cross-validation [1] and the Box-Jenkins method [3].

3.2 Clustering-Based Approaches

Clustering-based approaches are used to group similar datasets into clusters. Although clustering and anomaly detection tasks are different in nature, several clustering-based anomaly detection techniques have been developed. During the clustering process, these approaches consider the identified observations and clusters and then detect the anomalies [6].

We can group these techniques into two groups based on different assumptions. In first, the dataset is grouped into similar data clusters. Instances that do not belong to any cluster are marked as anomalies. Several algorithms do not force all instances to belong to a cluster; thus, identified clusters are removed from the dataset, and residual instances are noted as anomalies. A disadvantage of these techniques is that they are not optimized to identify anomalies, as their initial objective is to identify clusters. Among these techniques, we have DBSCAN [9] as the most used.

In the second group of techniques, initially, the dataset into clusters, using a clustering algorithm. Then, the distance of each instance to the centroid of the nearest cluster is calculated. Instances that are further away from your centroid are marked as anomalies. A series of anomaly detection techniques that follow this approach have been proposed using different clustering algorithms, among these, the most notorious K-Means clustering [20].

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) Density-based spatial clustering of applications with noise (DBSCAN) is a data clustering non-parametric algorithm [9]. The method is significantly effective in identifying clusters of arbitrary shape and sizes, identifying and separating noise from the data, and detecting *"natural"* clusters and their arrangements within the data space, without any preliminary information about the groups.

Given a set of points in some space, DBSCAN groups points that are intimately very close together (points with many nearby neighbors), marking them as outliers (anomalies) points that are isolated in low-density regions (whose closest neighbors are very far away) [5]. Two important user-defined parameters are required: neighborhood distance epsilon (*eps*) and a minimum number of points *minpts*. For a given point, the points in the *eps* distance are called neighbors of that point. If the total number of points neighboring a point is greater than *minpts*, this group is called a cluster.

DBSCAN labels the data points in three categories: 1) core: the points that have at least *minpts* number of points in the *eps* distance; 2) border: those that are not core points but are the neighbors of core points; 3)outlier (anomalous): those that do not belong to any cluster. Emadi et al. [8] propose an algorithm based on DBSCAN to detect anomalies in Wireless Sensor Networks.

K-Means Clustering One of the most explored techniques for grouping data is the K-means [20]. It is a cluster based on centroids that partitions the dataset into k clusters of similar instances. We can define the k-means algorithm in a sequence of steps. Initially, the number K of clusters is defined. Then, the k centroids are initialized, which can be done by arbitrary choice. Then, for each object, the distance to the centroid of all clusters and connected to the nearest centroid is calculated. Then the centroids of the modified clusters are recalculated. Finally, the distance from the objects to the centroids is recalculated, updating the connections with the nearest ones. This last step is repeated until there are no more updates [17].

Originally, the k-means approach was not defined to work with time series. One of the possibilities to use k-means as anomaly detection in time series is through the use of the sliding window [19] approach. With this approach, a set of subsequences of equal lengths of the same time series is generated. In this set, the k-means algorithm is applied until it converges on the searched k clusters. To perform the anomaly search, the distance between each subsequence and its associated centroid is computed. This subsequence is marked as an anomaly if this distance value is greater than a defined threshold value δ. One of the biggest challenges of this approach is the correct parameterization of the k amounts of clusters.

3.3 Matrix Profile Technique

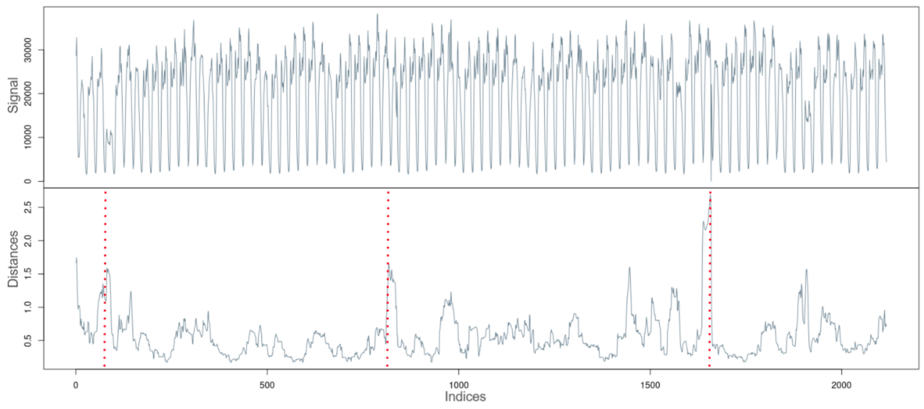
In 2016, Yeh et al. [29] published a novel technique to perform all-pairs-similarity-search on two time-series, producing two new series: the *Matrix Profile* and the *Matrix Profile Index*. The *Matrix Profile* is defined as a data structure containing the z-normalized Euclidean distances between each subsequence of the first series and its closest corresponding subsequence of the second time series. The Array Profile Index contains the index of the closest matching substring in the second series for each substring. By itself, the Matrix Profile can be used to detect anomalies in contexts where anomalies are defined by unique behavior [22]. In fact, in the Matrix Profile vector the anomalies can be detected in the points with high values, because the distance of the subsequences represented by these points to their closest matching subsequence is high.

In general, given two series of n real values, $\mathbf{S1} \in R^n$ and $\mathbf{S2} \in R^n$ and a subsequence length m, the Matrix Profile $\mathbf{M} \in R^{n-m+1}$ and a Matrix Profile Index $\mathbf{I} \in R^{n-m+1}$ are new series such that for each $i \in [0, n-m]$, I_i contains the index of the start of the subsequence of $S2$ with length m that best matches $S1_{i,m}$ and M_i contains the corresponding distance. In the case a *self-join* is performed where $S1 = S2$, an additional constraint is added to prevent *trivial matches*, where subsequences match themselves or nearby subsequences, called *exclusion zone*. The default distance measure used is the z-normalized Euclidean distance, which removes the effect of a changing data offset over time and thus focuses more on shape instead of amplitude. Typical causes of a changing offset are wandering baselines in sensors or natural phenomena (e.g., the gradual change in temperature throughout seasons) [22].

Figure 4 shows an example of the application of the matrix profile in a time series for the detection of anomalies. In this figure, the top image represents a time series, and the bottom image its matrix profile. The top 3 anomalies are marked in red, i.e., the three points where the matrix profile has high values.

Matrix profile is usually able to detect subsequences with unusual shapes in the data. It can be used for detecting point anomalies and also collective anomalies (*i.e.*, a sequence of abnormal points).

Fig. 4. Matrix Profile Anomaly Detection Example



In the rest of this section, we present the main algorithms for calculating the Matrix Profile.

STAMP The Matrix Profile was originally published together with the STAMP (Scalable Time Series Anytime Matrix Profile) [29], an anytime algorithm to calculate the Matrix Profile over a time series and the corresponding Index. Internally, STAMP uses a similarity search algorithm called MASS [21] that under z-normalized Euclidean iteratively calculates the *distance profile* of each subsequence, which is the distance of the subsequence to every subsequence, by using the Fast Fourier Transform (FFT).

The STAMP is outlined in Algorithm 1. In line 2, the length of T_B is extracted. In line 3, the matrix profile P_{AB} and matrix profile index I_{AB} are initialized. From lines 4 to line 6, the distance profiles D are calculated, using each subsequence $B[idx]$ in the time series T_B and T_A. The pairwise minimum for each element in D is performed with the paired element in P_{AB} (i.e., min($D[i]$, $P_{AB}[i]$) for $i = 0$ to length(D) -1). Then, as the minimum pair operations are performed, $I_{AB}[i]$ is updated with idx (when $D[i] \leq P_{AB}[i]$). Finally, the results, i.e., P_{AB} and I_{AB}, are returned in line 7. In this format, STAMP computes the matrix profile for the general similarity join. It is possible to change the algorithm to compute the self-similarity join matrix profile of a time series T_A, just by replacing T_B in line 2 with T_A, replace B with A in line 5, and ignore trivial matches in D when performing *ElementWiseMin* in line 6.

The overall complexity of the algorithm is $O(n^2 \log n)$ where n is the length of the time series. Since all subsequences are compared using the MASS algorithm, the $n \log n$ factor comes from the FFT subroutine.

Algorithm 1: STAMP (T_A, T_B, m)

```
  Input: Two time series, $T_A$ and $T_B$
  Interested subsequence length $m$
  Output: A matrix profile $P_{AB}$ and associated matrix profile index $I_{AB}$ of $T_A$
          join $T_B$
1 begin
2     $n_B \leftarrow Length(T_B)$
3     $P_{AB} \leftarrow$ infs, $I_{AB} \leftarrow$ zeros, idxes $\leftarrow 1 : n_B - m + 1$
4     for each idx in idxes do
5         $D \leftarrow$ MASS$(B[idx], T_A)$
6         $P_{AB}, I_{AB} \leftarrow$ ElementWiseMin$(P_{AB}, I_{AB}, D, idx)$
7     return $P_{AB}, I_{AB}$
```

STOMP The STOMP algorithm is similar to STAMP [15] in that it can be seen as highly optimized nested loop searches, with the repeated calculation of distance profiles as the inner loop. However, while STAMP must evaluate the distance profiles in random order (to allow its anytime behavior), STOMP performs an *ordered* search. It is by exploiting the locality of these searches that STOMP can reduce the time complexity by a factor of $O(\log n)$. STOMP uses the z-normalized Euclidean distance $d_{i,j}$, as shown below, of two time series subsequences $T_{i,m}$ and $T_{j,m}$ using their dot product, $QT_{i,j}$:

$$d_{i,j} = \sqrt{2m\left(1 - \frac{QT_{i,j} - m\mu_i\mu_j}{m\sigma_i\sigma_j}\right)} \tag{4}$$

Here m is the subsequence length, μ_i is the mean of $T_{i,m}$, μ_j is the mean of $T(j,m)$, σ_i is the standard deviation of $T_{i,m}$, and σ_j is the standard deviation of $T_{j,m}$. Note that $QT_{i,j}$ can be decomposed as:

$$QT_{i,j} = \sum_{k=0}^{m-1} T_{i+k}T_{j+k} \tag{5}$$

The time required to compute $d_{i,j}$ depends only on the time required to compute $QT_{i,j}$. To solve this problem, STOMP pre-computes and stores the means and standard deviation in $O(n)$ space and time, thus, it takes $O(1)$ to compute $d_{i,j}$ [30].

The pseudo-code of STOMP algorithm is shown in Algorithm 2. It begins in line 1 by computing the matrix profile length l. In line 2, it calculates the mean

Algorithm 2: STOMP (T, m)

```
   Input: A time series $T$ and a subsequence length $m$
   Output: Matrix profile $P$ and the associated matrix profile index $I$ of $T$
1  begin
2      $n \leftarrow$ Length$(T), l \leftarrow n - m + 1$
3      $\mu, \sigma \leftarrow$ ComputeMeanStd$(T, m)$
4      $QT \leftarrow$ SlidingDotProduct$(T[1:m], T)$, $QT_{first} \leftarrow QT$
5      $D \leftarrow$ CalculateDistanceProfile $(QT, \mu, \sigma)$
6      $P \leftarrow D, I \leftarrow$ ones // initialization
7      for $i$=2 to $l$ do
8          for $j$=$l$ downto 2 do
9              $QT[j] \leftarrow QT[j-1] - T[j-1] \times T[i-1] + T[j+m-1] \times T[i+m-1]$
10         $QT[1] \leftarrow QT_{first}[i]$
11         $D \leftarrow$ CalculateDistanceProfile $(QT, \mu, \sigma, i)$
12         $P, I \leftarrow$ ElementWiseMin $(P, I, D, i)$
13     return $P, I$
```

and standard deviation of every subsequence in T. Line 3 calculates the first dot product vector QT with the algorithm in TABLE I. Line 5 initializes the matrix profile P and matrix profile index I. The loop in lines 6-13 calculates the distance profile of every subsequence of T in sequential order, with lines 7-9 updating QT according to (5). Then update $QT[1]$ in line 10 is done with the pre-computed QT_first in line 3. Line 11 calculates distance profile D according to Equation (4). Finally, line 12 compares every element of P with D: if $D[j] < P[j]$, then $P[j] = D[j]$, $I[j] = i$.

The time complexity of STOMP is $O(n^2)$. Thus, it can achieve a $O(\log n)$ factor speedup over STAMP[15]. The $O(\log n)$ speedup makes little difference for small datasets, however, when considering the datasets with millions of data points, this $O(\log n)$ factor begins to produce a significant performance gain.

SCRIMP++ An extension of *STOMP* is proposed in [31]. *SCRIMP* is an anytime algorithm that computes the matrix profile algorithm combining the anytime component of *STAMP* with the speed of *STOMP*. The optimization of *SCRIMP++* is performed using the incremental calculation of the D diagonals of the scalar product of Equation 4, as follows:

$$Q_{i,j} = Q_{i-1,j-1} - t_{i-1}t_{j-1} + t_{i+m-1}t_{j+m-1} \tag{6}$$

Equation 6 presents the incremental approach, where the values of the diagonal cells can be calculated using the cell value previously calculated. This approach reduces the number of operations required for the new calculation.

Algorithm 3 shows the pseudo-code of *SCRIMP++*. Line 2 precomputes the means and standard deviations of all subsequences in T. In line 4, matrix profile P and matrix profile index I are initialized. In lines 6-16, the diagonals of the

Algorithm 3: The *SCRIMP++* Algorithm

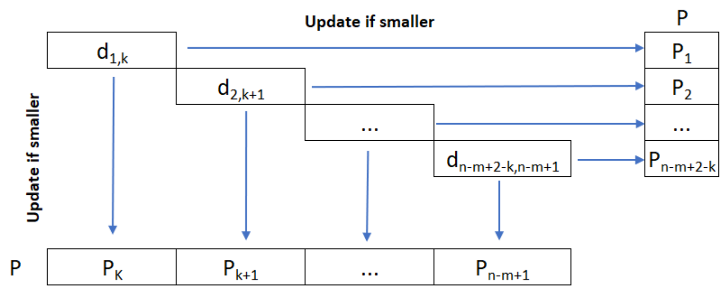
```
Input: A time series $T$ and a subsequence length $m$
Output: Matrix profile $P$ and matrix profile index $I$ of $T$
1  begin
2      $n \leftarrow$ Length($T$)
3      $\mu, \sigma \leftarrow ComputeMeanStd(T, m)$
4      $P \leftarrow inf, I \leftarrow$ ones
5      $Orders \leftarrow RandPerm(m/4 + 1 : n - m + 1)$
6      for $k$ in $Orders$ do
7          for $i$=1 to $n$-$m$+2-$k$ do
8              if $i = 1$ then
9                  $q \leftarrow DotProduct(T_{1,m}, T_{k,m})$
10             else
11                 $q \leftarrow q - t_1 t_{i+k-2} + t_{i+m-1} t_{i+k+m-2}$
12             $d \leftarrow CalculateDistance(q, \mu_i, \sigma_i, \mu_{i+k-1}, \sigma_{i+k-1})$
13             if $d < P_i$ then
14                 $P_i \leftarrow d, I_i \leftarrow i + k - 1$
15             if $d < P_{i+k-1}$ then
16                 $P_{i+k-1} \leftarrow d, I_{i+k-1} \leftarrow i$
17     return $P, I$
```

distance matrix are iteratively evaluated, being chosen in random order. Figure 5 shows an example of this process. Diagonal distance values, such as $d_{1,k}$, $d_{2,k}$, $\ldots$, $d_{n-m+2-k,n-m+1}$ are calculated one by one. If $d_{i,i+k-1}$, for any $i < n-m+1$, referenced by d in line 12, is less than P_i (line 13) or P_{i+k-1}(line 15), then the associated matrix profile and index values are updated. The iterative algorithm can be interrupted by the user to analyze the matrix profile and index values.

AAMP In many applications it is preferred to not normalize the time-series data because the anomalies can be deteted based on the point values, and not the shapes. AAMP [2] has been designed for such applications. It is an efficient algorithm for computing matrix profile with the pure (non-normalized) Euclidean distance. AAMP is executed in a set of iterations, such that in each iteration the distance of subsequences is computed incrementally. The time complexity of AAMP is $O(n \times (n - m))$ with small constants, where n is the time series length and m the subsequence length. The experiments reported in [2] show that the performance of AAMP is significantly better than that of STAMP and SCRIMP++ (an improved version of STOMP).

The main idea behind AAMP is that for computing the distance between subsequences it uses *diagonal sliding windows*, such that in each sliding window, the Euclidean distance is incrementally computed only between the subsequences that have a precise difference in their *start position.* Let $T_i = \langle t_i, t_{i+1}, \ldots, t_{i+m-1} \rangle$ and $T_j = \langle t_j, t_{j+1}, \ldots, t_{j+m-1} \rangle$ be two subsequences. The sliding windows in

Fig. 5. A SCRIMP++ iteration evaluates a randomly selected diagonal, thus updating the matrix profile



AAMP allow to use Equation 7 for incremental computation of the distance between subsequences T_i and T_j (denoted by $D_{i,j}$) by using the yet computed distance between subsequences T_{i-1} and T_{j-1} (denoted as $D_{i-1,j-1}$):

$$[H]D_{i,j} = \sqrt{D^2_{i-1,j-1} - (t_{i-1} - t_{j-1})^2 + (t_{i+m-1} - t_{j+m-1})^2} \tag{7}$$

Algorithm 4: AAMP algorithm

```
   Input: $T$: time series; $n$: length of time series; $m$: subsequence length
   Output: $P$: Matrix profile;
1  begin
2      for $i$=1 to $n$ do
3          P[i] = $\infty$ ;
4      for $k$=1 to $n$-$m$-1 do
5          $dist = Euc\_Distance(T_{1,m}, T_{k,m})$
6          if $dist < P[1]$ then
7              P[1] = dist;
8          if $dist < P[k]$ then
9              P[k] = dist;
10         for $i$=2 to $n - m + 1 - k$ do
11             $dist = \sqrt{(dist^2 - (t_{i-1} - t_{i-1+k})^2 + (t_{i+m-1} - t_{i+m+k-1})^2}$
12             if $dist < P[i]$ then
13                 P[i] = dist;
14             if $dist < P[i+k]$ then
15                 P[i+k] = dist;
```

Algorithm 4 shows the pseudo-code of AAMP. Initially, the algorithm sets all the values of the matrix profile to infinity (*i.e.*, maximum distance). Then, it performs $n - m - 1$ iterations using a variable k ($1 \leq k \leq n - m - 1$). In each iteration k, the algorithm compares each subsequence $T_{i,m}$ with the subsequence that is k positions far from it, *i.e.*, $T_{i,m+k}$. To do this, AAMP firstly computes the Euclidean distance of the first subsequence of the time series, *i.e.*, $T_{1,m}$, with the one that starts at position k, *i.e.*, $T_{k,m}$. This first distance computation is done using the normal formula of Euclidean distance. Then, in a sliding window, the algorithm incrementally computes the distance of other subsequences with the subsequences that are k position far from them, and this is done by using Equation 7 in $O(1)$. If the computed distance is smaller than the previous minimum distance that is kept in the matrix profile P, then it is updated with this new lower value.

4 Conclusion

In this paper, we presented a survey of anomaly detection methods in time series datasets. We firstly presented the main concepts related to anomalous data in different applications, and then defined the anomaly detection task. Afterwards, we described the important approaches for anomaly detection in three main categories: statistical based, clustering based, and matrix profile based.

References

1. Aggarwal, C.C.: Outlier Analysis. Springer International Publishing (2017). https://doi.org/10.1007/978-3-319-47578-3,
2. Akbarinia, R., Cloez, B.: Efficient matrix profile computation using different distance functions. CoRR **abs/1901.05708** (2019), http://arxiv.org/abs/1901.05708
3. Box, G.: Time series analysis : forecasting and control. John Wiley & Sons, Inc, Hoboken, New Jersey (2016)
4. Braei, M., Wagner, S.: Anomaly detection in univariate time-series: A survey on the state-of-the-art (2020)
5. Celik, M., Dadaser-Celik, F., Dokuz, A.S.: Anomaly detection in temperature data using DBSCAN algorithm. In: 2011 International Symposium on Innovations in Intelligent Systems and Applications. IEEE (Jun 2011). https://doi.org/10.1109/inista.2011.5946052, https://doi.org/10.1109/inista.2011.5946052
6. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: A survey. ACM Comput. Surv. **41** (07 2009). https://doi.org/10.1145/1541880.1541882
7. Chandola, V., Mithal, V., Kumar, V.: Comparative evaluation of anomaly detection techniques for sequence data. In: 2008 Eighth IEEE International Conference on Data Mining. IEEE (Dec 2008). https://doi.org/10.1109/icdm.2008.151, https://doi.org/10.1109/icdm.2008.151
8. Emadi, H.S., Mazinani, S.M.: A novel anomaly detection algorithm using DBSCAN and SVM in wireless sensor networks. Wireless Personal Communications **98**(2), 2025–2035 (Sep 2017). https://doi.org/10.1007/s11277-017-4961-1, https://doi.org/10.1007/s11277-017-4961-1

9. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining. p. 226–231. KDD'96, AAAI Press (1996)
10. Feyrer, J.: Trade and income—exploiting time series in geography. American Economic Journal: Applied Economics **11**(4), 1–35 (Oct 2019). https://doi.org/10.1257/app.20170616, https://doi.org/10.1257/app.20170616
11. Hawkins, D.M.: Identification of Outliers. Springer Netherlands (1980). https://doi.org/10.1007/978-94-015-3994-4, https://doi.org/10.1007/978-94-015-3994-4
12. Hyndman, R.: Forecasting : principles and practice. OTexts, Heathmont, Vic (2014)
13. Hyndman, R.: Forecasting : principles and practice. OTexts, Melbourne (2018)
14. Ignatov, A.: Real-time human activity recognition from accelerometer data using convolutional neural networks. Applied Soft Computing **62**, 915–922 (Jan 2018). https://doi.org/10.1016/j.asoc.2017.09.027, https://doi.org/10.1016/j.asoc.2017.09.027
15. Keogh, E., Kasetty, S.: Data Mining and Knowledge Discovery **7**(4), 349–371 (2003). https://doi.org/10.1023/a:1024988512476, https://doi.org/10.1023/a:1024988512476
16. Koijen, R.S., Lustig, H., Nieuwerburgh, S.V.: The cross-section and time series of stock and bond returns. Journal of Monetary Economics **88**, 50–69 (Jun 2017). https://doi.org/10.1016/j.jmoneco.2017.05.006, https://doi.org/10.1016/j.jmoneco.2017.05.006
17. Kumari, R., Sheetanshu, Singh, M.K., Jha, R., Singh, N.: Anomaly detection in network traffic using k-mean clustering. In: 2016 3rd International Conference on Recent Advances in Information Technology (RAIT). IEEE (Mar 2016). https://doi.org/10.1109/rait.2016.7507933, https://doi.org/10.1109/rait.2016.7507933
18. Lahmiri, S.: A variational mode decompoisition approach for analysis and forecasting of economic and financial time series. Expert Systems with Applications **55**, 268–273 (Aug 2016). https://doi.org/10.1016/j.eswa.2016.02.025, https://doi.org/10.1016/j.eswa.2016.02.025
19. Li, J., Izakian, H., Pedrycz, W., Jamal, I.: Clustering-based anomaly detection in multivariate time series data. Applied Soft Computing **100**, 106919 (Mar 2021). https://doi.org/10.1016/j.asoc.2020.106919, https://doi.org/10.1016/j.asoc.2020.106919
20. Likas, A., Vlassis, N., Verbeek, J.J.: The global k-means clustering algorithm. Pattern Recognition **36**(2), 451–461 (Feb 2003). https://doi.org/10.1016/s0031-3203(02)00060-2, https://doi.org/10.1016/s0031-3203(02)00060-2
21. Mueen, A., Zhu, Y., Yeh, M., Kamgar, K., Viswanathan, K., Gupta, C., Keogh, E.: The fastest similarity search algorithm for time series subsequences under euclidean distance (August 2017), http://www.cs.unm.edu/ mueen/FastestSimilaritySearch.html
22. Paepe, D.D., Hautte, S.V., Steenwinckel, B., Turck, F.D., Ongenae, F., Janssens, O., Hoecke, S.V.: A generalized matrix profile framework with support for contextual series analysis. Engineering Applications of Artificial Intelligence **90**, 103487 (Apr 2020). https://doi.org/10.1016/j.engappai.2020.103487, https://doi.org/10.1016/j.engappai.2020.103487
23. Pincus, R.: Barnett, v., and lewis t.: Outliers in statistical data. 3rd edition. j. wiley & sons 1994, XVII. 582 pp., £49.95. Biometrical Jour-

nal **37**(2), 256–256 (1995). https://doi.org/10.1002/bimj.4710370219, https://doi.org/10.1002/bimj.4710370219

24. Pourbabaee, B., Roshtkhari, M.J., Khorasani, K.: Deep convolutional neural networks and learning ECG features for screening paroxysmal atrial fibrillation patients. IEEE Transactions on Systems, Man, and Cybernetics: Systems **48**(12), 2095–2104 (Dec 2018). https://doi.org/10.1109/tsmc.2017.2705582, https://doi.org/10.1109/tsmc.2017.2705582
25. Qiu, X., Ren, Y., Suganthan, P.N., Amaratunga, G.A.: Empirical mode decomposition based ensemble deep learning for load demand time series forecasting. Applied Soft Computing **54**, 246–255 (May 2017). https://doi.org/10.1016/j.asoc.2017.01.015, https://doi.org/10.1016/j.asoc.2017.01.015
26. Soares, E., Costa, P., Costa, B., Leite, D.: Ensemble of evolving data clouds and fuzzy models for weather time series prediction. Applied Soft Computing **64**, 445–453 (Mar 2018). https://doi.org/10.1016/j.asoc.2017.12.032, https://doi.org/10.1016/j.asoc.2017.12.032
27. Thudumu, S., Branch, P., Jin, J., Singh, J.: A comprehensive survey of anomaly detection techniques for high dimensional big data. Journal of Big Data **7**(1) (Jul 2020). https://doi.org/10.1186/s40537-020-00320-x, https://doi.org/10.1186/s40537-020-00320-x
28. Wang, X., Ahn, S.H.: Real-time prediction and anomaly detection of electrical load in a residential community. Applied Energy **259**, 114145 (2020). https://doi.org/https://doi.org/10.1016/j.apenergy.2019.114145, https://www.sciencedirect.com/science/article/pii/S030626191931832X
29. Yeh, C., Zhu, Y., Ulanova, L., Begum, N., Ding, Y., Dau, H., Silva, D., Mueen, A., Keogh, E.: Matrix profile i: All pairs similarity joins for time series: A unifying view that includes motifs, discords and shapelets. In: 2016 IEEE 16th International Conference on Data Mining (ICDM). pp. 1317–1322. IEEE Computer Society, Los Alamitos, CA, USA (dec 2016). https://doi.org/10.1109/ICDM.2016.0179, https://doi.ieeecomputersociety.org/10.1109/ICDM.2016.0179
30. Zhu, Y., Zimmerman, Z., Senobari, N.S., Yeh, C.M., Funning, G., Mueen, A., Brisk, P., Keogh, E.: Matrix profile ii: Exploiting a novel algorithm and gpus to break the one hundred million barrier for time series motifs and joins. In: 2016 IEEE 16th International Conference on Data Mining (ICDM). pp. 739–748 (2016). https://doi.org/10.1109/ICDM.2016.0085
31. Zhu, Y., Yeh, C.C.M., Zimmerman, Z., Kamgar, K., Keogh, E.: Matrix profile xi: Scrimp++: Time series motif discovery at interactive speeds. In: 2018 IEEE International Conference on Data Mining (ICDM). pp. 837–846 (2018). https://doi.org/10.1109/ICDM.2018.00099