# A New Pattern Representation Method for Time-Series Data

Roonak Rezvani, Payam Barnaghi, *Senior Member, IEEE*, and Shirin Enshaeifar, *Member, IEEE*

**Abstract**—The rapid growth of Internet of Things (IoT) and sensing technologies has led to an increasing interest in time-series data analysis. In many domains, detecting patterns of IoT data and interpreting these patterns are challenging issues. There are several methods in time-series analysis that deal with issues such as volume and velocity of IoT data streams. However, analysing the content of the data streams and extracting insights from dynamic IoT data is still a challenging task. In this paper, we propose a pattern representation method which represents time-series frames as vectors by first applying Piecewise Aggregate Approximation (PAA) and then applying Lagrangian Multipliers. This method allows representing continuous data as a series of patterns that can be used and processed by various higher-level methods. We introduce a new change point detection method which uses the constructed patterns in its analysis. We evaluate and compare our representation method with Blocks of Eigenvalues Algorithm (BEATS) and Symbolic Aggregate approXimation (SAX) methods to cluster various datasets. We have evaluated our algorithm using UCR time-series datasets and also a healthcare dataset. The evaluation results show significant improvements in analysing time-series data in our proposed method.

**Index Terms**—Lagrangian multiplier, data analytics, aggregation, data representation, change point detection

## 1 Introduction

The rapid growth of connected devices and networks generates massive amount of data. Some of this data is generated by Internet of Things (IoT) technologies which capture information from the physical environment. IoT is pushing the boundaries between the physical and digital world and allows collecting continuous observations and measurements and also provides means for actuation and interaction with the physical world. IoT data is often represented as time-series which is a collection of observations in a time domain [1].

Analysing time-series data can be beneficial in developing effective methods for processing the observations and gaining insight into relationships and hidden structures of the data. It is also important to identify various patterns in time-series data and gather information about how these patterns change or co-relate over time. However, there are several challenges in analysing IoT data including high dimensionality, volume, scalability, noise and measurements errors, computational costs, heterogeneity, diversity of sensor technologies and variations in the quality of data. Due to the applications of time-series data in different domains such as biology, industry and finance, there have been several works on the development of transformation and representation methods for time-series data. One of the key steps in processing time-series data is often dimensionality reduction and applying spatial methods to transform the data from time domain to other domains [2] such as Discrete Fourier Transform (DFT) [3], Discrete Wavelet Transform (DWT) [4] and Singular Value Decomposition (SVD) [5].

There are other types of representation techniques such as Piecewise Linear Segmentation [6], Piecewise Aggregate Approximation (PAA) [2] and Adaptive Piecewise Constant Approximation (APCA) [7] that use simple statistical properties to reduce the size of time-series data. There are also methods that represent time-series data in symbolic or latent space models; e.g., Symbolic Aggregate approXimation (SAX) [8] and Blocks of Eigenvalues Algorithm for Time-series Segmentation (BEATS) [1]. However, in this paper, our purpose is not just to reduce the size of time-series data. The main objective is to identify and analyse the emerging patterns from the time-series data.

Normalised time-series data can be represented by a Gaussian model with mean of zero ($\mu = 0$) and standard deviation of one ($\sigma = 1$) [8]. Methods such as SAX use this Gaussian property to represent the data based on the symbolic representations. However, the existing segmentation and representation methods for time-series data have limitations in handling noisy data and do not adapt well to data and concept drifts. In this paper, we introduce a time-series data segmentation and pattern representation method which capture the useful information of data. We explore the spatial and temporal correlations of data in small time segments. We represent the time-series as blocks of segments and later on as eigenvectors of those segments.

Detecting change points in time-series data is also one of the key issues in analysing continuous data points. Several approaches have been proposed in this area including CUmulative SUM method (CUSUM) [9], [10], Bayesian method [11]

- *The authors are with the Centre for Vision, Speech, and Signal Processing (CVSSP), the Department of Electronic and Electrical Engineering, University of Surrey, GU2 7XH, Guildford, United Kingdom, and also with the Care Research and Technology Centre, UK Dementia Research Institute (UK DRI), W12 0NN, London, United Kingdom.*
  *E-mail: {r.rezvani, p.barnaghi, s.enshaeifar}@surrey.ac.uk.*

*Manuscript received 23 Oct. 2018; revised 2 Dec. 2019; accepted 10 Dec. 2019. Date of publication 23 Dec. 2019; date of current version 3 June 2021.*
*(Corresponding author: Roonak Rezvani.)*
*Recommended for acceptance by D. J. Cook.*
*Digital Object Identifier no. 10.1109/TKDE.2019.2961097*

and Kullback-Leibler Importance Estimation Procedure (KLIEP) [12].

The main contributions of this paper include a novel method for representing time-series data and developing a method for creating patterns based on eigenvector space model and also change point detection by using entropy models applied to the blocks of segments created by our pattern representation method. We have evaluated our work by applying clustering methods on a set of common datasets and have compared our work with some of the key state-of-the-art works in this domain. The evaluation results show that our proposed method provides better results in clustering time-series data (up to 20 percent higher than the state-of-the-art methods) and is also capable of detecting changes in time-series data with a higher performance (around 37 percent higher accuracy and around 10 percent higher reliability compared with a baseline method) which can be found in Section 5.

The remainder of this paper is organised as follows: Section 2 describes the related work. Section 3 details the proposed method and explains the mathematical details of the algorithm. Section 4 describes the change point detection. Section 5 presents the performance evaluations compared with other representation and change point detection techniques using a set of standard datasets and discusses the results of experiments. Section 6 describes the complexity analysis of the proposed method in representation and change point detection. Section 7 concludes the paper and discusses future work.

## 2 Related Work

### 2.1 Time-Series Representation

One of the major challenges in analysing time-series data is the volume of data points which makes data analysis complex and expensive in terms of storage and processing time. There are several solutions to tackle this problem. In most of the existing work, segmentation has been used as a pre-processing step. Segmentation is transforming a time-series of length $n$ to a sequence of $l$ piecewise segments ($l < n$) [1]. After segmentation, the representation methods are an important step to process segments of time-series data. Time-series data representation requires two key considerations: minimising the true distance measure of representation with original data, and preserving the key characteristics of the data while representing the data in lower dimensions [13].

One solution is to represent each segment as a line which connects the two endpoints. In other words, the representation of segments $(p_i, \ldots, p_j)$ as lines is connecting each $p_i$ to $p_j$ [14]. Keogh *et al.* [6] represent each segment with a line and a weight related to its importance with a bottom-up algorithm which uses merging of the sequences to get to the required number of sequences. They also introduce a weighted euclidean distance measure which includes the weights of segments in the measurement (Eq. (2)).

The euclidean distance measure is shown below where $\mathbf{x} = [x_1, \ldots, x_n]$ and $\mathbf{y} = [y_1, \ldots, y_n]$ are sequences of length $n$

$$D(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}. \quad (1)$$

In the weighted euclidean distance measure, $\mathbf{x}$ and $\mathbf{y}$ sequences have slices in which each slice is in a segment of the data, and each segment has weight as shown below:

$$D'(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{j \in \text{segments}} W_j D(\mathbf{x}_j, \mathbf{y}_j)}, \quad (2)$$

where $\mathbf{x}_j$ and $\mathbf{y}_j$ are the slices of $\mathbf{x}$ and $\mathbf{y}$ in the $j$th segment of the method and $W_j$ is the weight of the segment. In the simple line-based method some information will be lost; for example, if local maximum or minimum points are not endpoints, their information will be lost. There can also be a big change in the shape of the time-series within each segment, which will not be represented using a simple linear model. Shatkay *et al.* [15] deal with the above mentioned problems, by breaking the time-series into extremum points (excluding the small local extremums). They present their solution by using sliding windows, shifting a window through a time-series and cutting at the point where the mean absolute error passes a threshold. Each segment is represented by a function which can be a linear regression, a line which is an approximation of the curve, or interpolation through the endpoint of each sequence or in other words, a line which approximates the curve with including the endpoints [15]. There are existing works in introducing the streaming version of this technique [16]. Duvignau *et al.* [17] implemented a new Piecewise Linear Approximation (PLA) method in streaming paradigm with introducing a "singleton stream" contains only one value and using the number of points in each segment rather than storing the entire timestamps. Using the singleton stream reduces the generation of more unnecessary data when there are not enough data points.

Another type of representation for segments is to use the mean value [2], [7]. In [2], time-series of length $n$ ($\mathbf{x} = [x_1, \ldots, x_n]$) are represented as vectors of $w$-dimensions; where $w$ ($w < n$) is the number of equal sized segments. The representation is $\mathbf{x} = [\bar{x}_1, \ldots, \bar{x}_w]$ where

$$\bar{x}_i = \frac{w}{n} \sum_{j=\frac{n}{w}(i-1)+1}^{\frac{n}{w}i} x_j. \quad (3)$$

This method is called Piecewise Aggregate Approximation (PAA). PAA is simple and fast to compute, and for distance measure, the weighted euclidean distance measure or euclidean distance measure can be used. In [7], Keogh *et al.* discuss an arbitrary lengths representation method which is called Adaptive Piecewise Constant Approximation (APCA). Given a time-series $\mathbf{x} = [x_1, \ldots, x_n]$, the APCA representation is given as

$$\bar{X} = \{(\bar{x}_1, xr_1), \ldots, (\bar{x}_w, xr_w)\}, \quad (4)$$

where $\bar{x}_i$ is the mean value of $i$th segment and $xr_i$ is the right endpoint. One of the advantages of this method is the ability to place one segment in the areas of low activity and more segments in the areas of high activity. This technique outperforms the original PAA, and it also creates a better approximation for time-series. However, PAA and APCA have a major drawback which is the lack of ability to preserve the shape of time-series; in other words, two segments with different shapes can have similar mean values [18].

Other representation methods use transformation models such as transforming to frequency domain, the number of waves passing a point in a certain time. In these methods, the main purpose is to preserve the frequency coefficients which have the most ability to differentiate the transformed sequence from other sequences. Discrete Fourier Transform (DFT) is one of the methods which represents time-series sequences as a finite number of sine or cosine waves which can be shown via a set of complex numbers that are called Fourier coefficients [2]. One of the main advantages of representing time-series in a frequency domain, besides data compression, is the ability to use the DFT coefficients to construct the original time-series [2]. The DFT results can also handle the euclidean distance measure by data mining algorithms such as clustering and classification. However, DFT cannot capture an online event in an ad-hoc manner at a particular time [18]. To deal with this problem, one can use Discrete Wavelet Transform (DWT) and the Haar transform [19] which provides a quick approximation. It also preserves the euclidean distance measures between sequences in both time and frequency domains [4]. Wavelet transform is a technique to divide data into different frequency components based on two variables: frequency and time. The Wavelet transform is a time-frequency localisation to represent the frequency behaviour of a signal locally in time [20].

Another transformation method is Singular Value Decomposition (SVD) which is a common model in text and data analytics. SVD is a data-dependent operation because unlike DFT which takes cosine and sine as kernel functions, SVD computes the kernel function from the input data [5]. The definition of SVD is given below [21]:

**Theorem 1.** *Given an $n \times m$ matrix $\mathbf{X}$ we can represent it as*

$$\mathbf{X} = \mathbf{U} \times \mathbf{\Lambda} \times \mathbf{V}^T, \tag{5}$$

*where $\mathbf{U}$ is a column-orthonormal $n \times r$ matrix, $\mathbf{\Lambda}$ is a diagonal $r \times r$ matrix, and $\mathbf{V}$ is a column-orthonormal $m \times r$ matrix.*

The diagonal elements of matrix $\mathbf{\Lambda}$ are the eigenvalues of $\mathbf{X}$ and $r$ is the rank of $\mathbf{X}$ which is the number of the largest eigenvalues of $\mathbf{X}$. SVD can be used to reduce dimensions of multivariate time-series data. However, one of the problems with SVD is the cost of updates. We need to recalculate the entire operation when there are new elements in the time-series [5]. SVD also is not efficient for large datasets [13]. Another form of representation for time-series data is using symbolic forms. One of the well-known methods in this area is Symbolic Aggregate approXimation (SAX) which discretises a time-series into a set of symbolic sequences [8]. SAX uses PAA as an intermediate representation before creating symbolic representations. SAX normalises data to have a mean of zero and a standard deviation of one. It then reduces the length of time-series data using PAA. After that, with the assumption of a Gaussian distribution of normalised data [22], SAX uses breakpoints to produce $a$ equally probable areas under the Gaussian curve [8].

Breakpoints:

Breakpoints are a sorted list of numbers $B = \beta_1, \ldots, \beta_{a-1}$ such that the area under a Gaussian curve from $\beta_i$ to $\beta_{i+1}$ is $\frac{1}{a}$ ($\beta_0$ and $\beta_a$ are defined as $-\infty$ and $\infty$). $a$ is the number of symbols.

To represent a PAA representation as a sequence of symbols which is called a SAX word, SAX performs as follows: a sequence (PAA representation) $\mathbf{c} = [c_1, \ldots, c_w]$ will be represented as a word $\hat{\mathbf{c}} = [\hat{c}_1, \ldots, \hat{c}_w]$ where

$$\hat{c}_i = \alpha_j \text{ iff } \beta_{j-1} \leqslant c_i < \beta_j, \tag{6}$$

and $\alpha_j$ is the $j$th element of the SAX alphabets, $1 \leqslant j \leqslant a$ [8]. SAX similar to PAA does not consider the segments trends; in other words, two segments with same mean values have same symbol representation [1].

The Gaussian assumption is also a drawback in applying SAX to dynamic data streams. To deal with this problem, there have been works to extend SAX; e.g., improving the breakpoints using k-means clustering [23]. aSAX, adaptive SAX, uses small number of PAA representations (first represents normalised time-series as PAA) as training samples and the alphabet size $a$ as number of clusters and the breakpoints for Gaussian curve as initialisation of clusters [23]. There are also several other extensions of SAX, including ESAX (Extended SAX) which is to tackle the issue of segments trends [24]. This method adds maximum and minimum to each PAA segment besides its mean value. Other extensions include SAX Trend Distance ($SAX_TD$) [24] and SAX with Standard Deviation ($SAX_{SD}$) [25]. $SAX_TD$ uses the start and endpoints of each PAA segment for improvement and $SAX_{SD}$ adds the standard deviation of each PAA segment to it.

SAX introduces a distance measure with lower bounds euclidean distance [8]. This distance measure is defined as

$$\text{MINDIST}(\hat{\mathbf{q}}, \hat{\mathbf{c}}) = \sqrt{\frac{n}{w}} \sqrt{\sum_{i=1}^{w} dist(\hat{q}_i, \hat{c}_i)^2}, \tag{7}$$

where $\hat{\mathbf{q}}$ and $\hat{\mathbf{c}}$ are the SAX representations of time-series segments $\mathbf{q}$ and $\mathbf{c}$ and $w$ is the reduced length of these segments representations. The $dist()$ function computes the distance of two symbols using a lookup function $cell(r, c)$ which is defined as

$$cell(r, c) = \begin{cases} 0 & \text{if } |r - c| \leq 1 \\ \beta_{max(r,c)-1} - \beta_{min(r,c)} & \text{otherwise} \end{cases}. \tag{8}$$

The $dist(\hat{q}_i, \hat{c}_i)$ is equal to look up function $cell(r, c)$ when $\hat{q}_i = \alpha_r$ and $\hat{c}_i = \alpha_c$ [8]. An issue with the above described methods is using the normalising and re-scaling measures. To construct a segment representation without the normalisation step, the Blocks of Eigenvalues Algorithm for Time-series Segmentation (BEATS) divides the time-series into equal sized blocks and then uses Discrete Cosine Transform (DCT) and eigenvalues to represent the segments [1]. Discrete Cosine Transform (DCT) is a frequency transformation which takes cosine as its kernel function [1].

BEATS divides time-series data into windows of 64 observations and transforms each segment into a matrix $M$ of $8 \times 8$. It then applies DCT and transforms $\mathbf{M}$ using $\mathbf{D} = \mathbf{UMU}^T$, where

$$U_{i,j} = \begin{cases} \frac{\sqrt{2}}{2} & \text{if } i, j = 1 \\ cos(\frac{\pi}{n}(i-1)(j-\frac{1}{2})) & \text{if } i, j > \frac{1}{2} \end{cases}. \tag{9}$$

After that it uses $\mathbf{Z}$ (the standard quantisation matrix of DCT [26]) to quantise the $\mathbf{D}$ matrix ($\mathbf{Q} = \frac{\mathbf{D}}{\mathbf{Z}}$), it then extracts

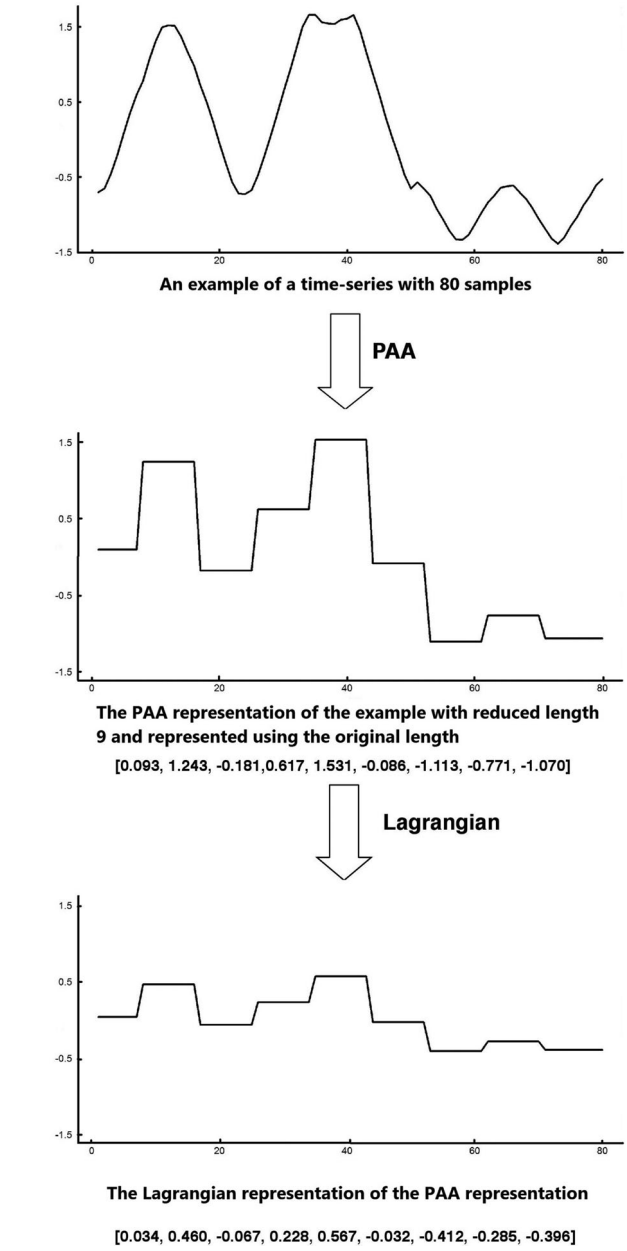



Fig. 1. An example of the proposed method.

the upper-left $4 \times 4$ matrix of it and computes the eigenvalues and removes the duplicate values [1].

### 2.2 Change Point Detection

In time-series analysis change detection is another key issue. A change detection method tries to find abrupt changes in the properties of time-series data [27]. The change point could have different interpretations in various methods. In some methods, the change point detection identifies a point in which the stationary properties change or in some method identifies a region that the change occurs. CUmulative SUM method (CUSUM) is a well-known algorithm. CUSUM is a sequential analysis technique [9] which uses a parameter of the probability distribution, mean, as a measurement to detect the change [9]. It sums samples sequentially, and when the value exceeds a threshold, it indicates a change point [9]. Another common change detection method is Kullback-Leibler Importance Estimation Procedure (KLIEP) that uses dissimilarity measure between two segments of time-series [12]. The dissimilarity measure which has been used in KLIEP is Kullback-Leibler (KL) divergence [12]

$$KL[p(x)||p'(x)] = -\int p'(x)\log\frac{p(x)}{p'(x)}, \tag{10}$$

where $p(x)$ and $p'(x)$ are probability distributions of two successive segments of time-series. Bayesian methods are also used for change detection in time-series data [27]. The Bayesian change detection method is based on the assumption that a time-series can be divided into segments that each have a probability distribution [11]. The process is based on Bayes theorem, and it estimates the posterior distribution using a Bayesian estimation method [11]. Bayesian change point detection method uses a supplementary variable ($r_t$) to show the time passes after the previous change point. At each time this variable can be set to zero or increase by one. The distribution of $r_t$ can be represented as below [27]:

$$P(r_t|x_{1:t}) = \frac{\sum_{r_{t-1}} P(r_t|r_{t-1})P(x_t|r_{t-1}, x_t^{(r)})P(r_{t-1}, x_{1:t-1})}{\sum_{r_t} P(r_t, x_{1:t})}, \tag{11}$$

where $x_t^{(r)}$ is the data corresponding to $r_t$, $P(r_t|r_{t-1})$ is the prior distribution in Bayes theorem which is the probability distribution without further evidence, $P(x_t|r_{t-1}, x_t^{(r)})$ is the likelihood function and $P(r_{t-1}, x_{1:t-1})$ is the recursive module for joint distributions in Bayes theorem [11].

In the next section, we describe our algorithm that represents the data as unit vectors. Unlike the existing methods such as SAX, it also does not need the distribution assumption. In representation algorithms, distance measure is one of the requirements which has been considered in our method.

## 3 Methodology

This section describes our proposed method that allows a time-series of length $n$ to be represented as a unit vector of $w$ ($w < n$) dimensions. We first reduce the length of the time-series using PAA and then by applying the Lagrangian multiplier[1] change the PAA representation to a unit vector. In mathematical optimisation, one can use the idea of Lagrangian multiplier for finding the local maxima and minima of a function related to equality constraints. This method allows solving the optimisation problems without finding parametric equations for the functions. Using the Lagrangian multiplier in our method has two key advantages: reducing the length of the time-series and showing the differences between different time-series without scaling problem by using unit vectors. We use an example to explain our proposed method. The example uses a time-series of length 80 as shown in Fig. 1.

1. Lagrangian is a reformulation of the classical mechanics which describes the motion of point particles. To compute the trajectory of a system of particles, the path point particles follow, one can use the Lagrangian multiplier. In the Newtonian law the equation of motion is given based on force; however, Lagrangian system uses kinetic energy (the energy which a system possesses because of the particles' motion) and potential energy (the energy which a system possesses due to the position of the particles relatives to other objects).

### 3.1 Length Reduction via PAA

PAA reduces the length of the time-series by splitting it into $w$ equal sized windows, represented by their means. A time-series $\mathbf{c}$ of length $n$ will be represented in a $w$-dimensional space with a vector $\vec{\mathbf{c}} = [\bar{c}_1, \ldots, \bar{c}_w]$. The $i$th element of $\vec{\mathbf{c}}$ is calculated using

$$\bar{c}_i = \frac{w}{n} \sum_{j=\frac{n}{w}(i-1)+1}^{\frac{n}{w}i} c_j. \tag{12}$$

This vector representation will be a reduced data representation (shown in Algorithm 1). We divide our example into a sequence of nine equal sized windows, and each window has been represented using its mean value. The PAA representation of our example is: [0.093, 1.243, -0.181, 0.617, 1.531, -0.086, -1.113, -0.771, -1.070].

**Algorithm 1.** PAA Algorithm

**Input:** $w, \mathbf{c}$
**Output: paa**
$w \leftarrow$ number of windows
$n \leftarrow$ length of a sample
$\mathbf{c} = [c_1, \ldots, c_n]$ a sample from dataset
$stepFloat \leftarrow \frac{n}{w}$
$step \leftarrow \text{integer}(\lceil stepFloat \rceil)$
$sectionStart \leftarrow 1$
$j \leftarrow 1$
**while** $sectionStart \leqslant n - step$ **do**
  **section** $\leftarrow [c_{sectionStart}, \ldots, c_{sectionStart+step-1}]$
  $paa_j \leftarrow \text{mean}(\textbf{section})$
  $sectionStart \leftarrow \text{integer}(j \cdot stepFloat)$
  $j \leftarrow j + 1$
**end while**

### 3.2 Aggregation via Lagrangian Multiplier

The Lagrangian multiplier allows us to maximise or minimise a function subject to equality constraints. In this method, we would like to find extrema, maximum or minimum, of function $E(\mathbf{u}) = E(u_1, \ldots, u_w)$. If we have no constraints, then the extrema must satisfy the $\triangledown E = 0$ system of equations which is equivalent to $\frac{\partial E}{\partial u_i} = 0$ for all $i$ elements in the time-series.

However, our goal is to find the extrema subject to a single constraint $g(\mathbf{u}) = 0$. In other words, we want to find the extrema points that satisfy the constraint. For this purpose, in Lagrangian multiplier method, we define a new function $L(\mathbf{u}, \lambda) = E(\mathbf{u}) - \lambda g(\mathbf{u})$. We then find the extrema of $L$ with respect to both $\mathbf{x}$ and $\lambda$ ($\frac{\mathrm{d}L}{\mathrm{d}\lambda} = 0$ and $\frac{\mathrm{d}L}{\mathrm{d}\mathbf{u}} = 0$). We find a unit vector $\vec{\mathbf{u}} = [u_1, \ldots, u_w]$ for each PAA representation $\vec{\mathbf{c}}$ which maximises the dot product $(\vec{\mathbf{u}} \cdot \vec{\mathbf{c}})$. The constraint is

$$\|\vec{\mathbf{u}}\| = \sqrt{\sum_{i=1}^{w} u_i^2} = 1 \tag{13}$$

$$g(\mathbf{u}) = \sum_{i=1}^{w} u_i^2 - 1 = 0. \tag{14}$$

The Lagrangian function is shown below:

$$\begin{aligned} L(\mathbf{u}, \lambda) &= \vec{\mathbf{u}} \cdot \vec{\mathbf{c}} - \lambda g(\mathbf{u}) \\ L(\mathbf{u}, \lambda) &= \sum_{i=1}^{w} u_i \bar{c}_i - \lambda \left( \sum_{i=1}^{w} u_i^2 - 1 \right). \end{aligned} \tag{15}$$

To solve this equation, we set $\triangledown L$ equal to zero

$$\begin{cases} \frac{\partial L}{\partial u_i} = \bar{c}_i - \lambda 2 u_i = 0 & \text{for } \; i = 1, \ldots, w \\ \frac{\partial L}{\partial \lambda} = -(\sum_{i=1}^{w} u_i^2 - 1) = 0 \end{cases}. \tag{16}$$

And then by solving the equations, we obtain

$$u_i = \frac{1}{2\lambda} \bar{c}_i \quad \text{for } \; i = 1, \ldots, w, \tag{17}$$

which means $\vec{\mathbf{u}}$ is proportional to $\vec{\mathbf{c}}$ and they are in the same direction. As a result, by normalising $\vec{\mathbf{c}}$ we will have a unit vector with the following properties:

$$\vec{\mathbf{u}} = \frac{\vec{\mathbf{c}}}{\|\vec{\mathbf{c}}\|}. \tag{18}$$

Therefore, $\vec{\mathbf{u}}$ is a unit vector projection of PAA representation $\vec{\mathbf{c}}$. As a result, we have a unit vector representation $\vec{\mathbf{u}}$ for each time-series $\vec{\mathbf{c}}$ (shown in Algorithm 2). The PAA representation of our example has been normalised and the unit vector projection is shown below:

[0.034, 0.460, -0.067, 0.228, 0.567, -0.032, -0.412, -0.285, -0.396]

**Algorithm 2.** Lagrangian Representation Algorithm

**Input: paa**
**Output: u**
$w \leftarrow$ length of sample after PAA
**paa** $= [paa_1, \ldots, paa_w]$ PAA representation of a sample
$\mathbf{u} = [u_1, \ldots, u_w]$ unit vector variable
$g(u_1, \ldots, u_w) \leftarrow u_1^2 + \cdots + u_w^2 - 1$
$f(\textbf{paa}, \mathbf{u}) \leftarrow paa_1 u_1 + \cdots + paa_w u_w$
$L(\textbf{paa}, \mathbf{u}, \lambda) \leftarrow f(\textbf{paa}, \mathbf{u}) - \lambda g(\mathbf{u})$
**for** $i$ from 1 to $w$ **do**
  $partialDer_i = paa_i - (\lambda 2 u_i)$
**end for**
$partialDer_\lambda = -u_1^2 - \cdots - u_w^2 + 1$
$u \leftarrow$ Solve $(partialDer_1 = 0, \ldots, partialDer_w = 0, partialDer_\lambda = 0)$

For a better understanding of Lagrangian multiplier, we show the process by providing an example:

**Example 3.2.1.** Let a three dimensional vector $\vec{\mathbf{v}}$ be defined as follow: $\vec{\mathbf{v}} = [3, -5, 4]$. We want to find a unit vector $\vec{\mathbf{u}} = [x, y, z]$ which maximises the $\vec{\mathbf{v}} \cdot \vec{\mathbf{u}}$ product. The magnitude of a unit vector is one. So

$$\sqrt{x^2 + y^2 + z^2} = 1 \Rightarrow x^2 + y^2 + z^2 = 1,$$

is our constraint. $\vec{v} \cdot \vec{u}$ is

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} \cdot \begin{bmatrix} 3 \\ -5 \\ 4 \end{bmatrix} = 3x - 5y + 4z,$$

which we want to maximise. The Lagrangian equation of this example is

$$L(x, y, z, \lambda) = 3x - 5y + 4z - \lambda(x^2 + y^2 + z^2 - 1).$$

We now solve the $\triangledown L = 0$ by setting each partial derivative of this expression equal to zero

$$\begin{aligned}
&\frac{\partial}{\partial x}(3x - 5y + 4z - \lambda(x^2 + y^2 + z^2 - 1)) = 3 - 2\lambda x = 0\\
&\frac{\partial}{\partial y}(3x - 5y + 4z - \lambda(x^2 + y^2 + z^2 - 1)) = -5 - 2\lambda y = 0\\
&\frac{\partial}{\partial z}(3x - 5y + 4z - \lambda(x^2 + y^2 + z^2 - 1)) = 4 - 2\lambda z = 0\\
&\frac{\partial}{\partial \lambda}(3x - 5y + 4z - \lambda(x^2 + y^2 + z^2 - 1))\\
&\quad = -x^2 - y^2 - z^2 + 1 = 0.
\end{aligned}$$

Solving for $x$, $y$ and $z$ in the first three equations above, give us

$$x = 3 \cdot \frac{1}{2\lambda} y = -5 \cdot \frac{1}{2\lambda} \; z = 4 \cdot \frac{1}{2\lambda},$$

so

$$\vec{\mathbf{u}} = \frac{1}{2\lambda} \begin{bmatrix} 3 \\ -5 \\ 4 \end{bmatrix} = \frac{1}{2\lambda} \vec{\mathbf{v}}.$$

This means $\vec{\mathbf{u}}$ is proportional to $\vec{\mathbf{v}}$. For the maximisation, the unit vector is in the same direction as $\vec{\mathbf{v}}$ and the result of Lagrangian multiplier process which is shown below:

$$\vec{\mathbf{u}}_{max} = \frac{\vec{\mathbf{v}}}{\|\vec{\mathbf{v}}\|}.$$

So, in our example $\lambda$ is

$$\left(\frac{3}{2\lambda}\right)^2 + \left(\frac{-5}{2\lambda}\right)^2 + \left(\frac{4}{2\lambda}\right)^2 = 1 \; \lambda = \frac{5\sqrt{2}}{2}.$$

And $\vec{\mathbf{u}}_{max}$ which is the result of Lagrangian multiplier is

$$\vec{\mathbf{u}}_{max} = \frac{1}{5\sqrt{2}} \begin{bmatrix} 3 \\ -5 \\ 4 \end{bmatrix}.$$

The outcome of the Lagrangian multiplier applied to PAA segmentation constructs our pattern representation model. We then use the constructed patterns to process and analyse the time-series. Since the constructed patterns are represented as vectors, we can use various analysis method to process and extract information from the underlying time-series data. In the following section, we discuss change point detection using the result of our Lagrangian multiplier algorithm. In Section 5, we also describe an evaluation of the algorithm for clustering a set of common time-series datasets.

## 4 Change Points Detection

In this section, we describe a method for change point detection using our Lagrangian multiplier based representation method. As mentioned in Section 2, the change point is an abrupt difference in the parameters of time-series, which is useful in learning the structure of the data [27]. Entropy is the amount of information one can expect to get if an event happens. Moreover, entropy and probability distribution have a relationship; when the probability of the data is high, the amount of new information we get is little, and the entropy is small. However, entropy is sensitive to noise. For this reason, before using an entropy measure, we apply the Singular Spectrum Analysis (SSA) technique on the Lagrangian based representations to reduce the noise. SSA decomposes a sequence into an additive set of independent sequences. SSA has been used for extracting trends and noises in signal processing [28]. The SSA method builds a Hankel matrix from the sequence and then by applying SVD, decomposes the matrix into a sum of matrices which can be reconstructed into sequences. The sum of these resulting sequences is equal to the original sequence. A Hankel matrix is a matrix with equal elements on the anti-diagonals.

After applying SSA, we take the sequence and change the representation of it in which each row represent a sliding window from the data matrix. We then obtain the maximum and minimum of the values in the matrix and compute the value range ($\max - \min$) and divide it into equal sized ranges. After that, we change the matrix elements into their relevant range. By computing the probability distribution of each element and their joint probabilities, we compute the mutual information between each adjacent pair of rows and compare them together to find the change points. The following describes the above mentioned steps in more details (see Algorithm 4). We also show the algorithm with an example in Fig. 2.

### 4.1 Singular Spectrum Analysis

We now describe the SSA method in more details.

#### 4.1.1 Step 1. Embedding

Let $\vec{\mathbf{u}} = [u_1, \ldots, u_w]$ be the sequence which has been obtained from our Lagrangian representation method then in embedding step, $\mathbf{u}$ is mapped to trajectory matrix $\mathbf{M} \in \mathbb{R}_{l \times k}$ where its first column is a segment from the $\mathbf{u}$ and its second column is the one-step lagged version of the first and $l$ $(1 < l < w)$ is the window length and $k = w - l + 1$. The trajectory matrix of $\vec{\mathbf{u}}$ is [28]

$$\begin{aligned}
\mathbf{M} &= [\mathbf{m}_1 : \mathbf{m}_2 : \ldots : \mathbf{m}_k]\\
&= \begin{bmatrix}
u_1 & u_2 & u_3 & \ldots & u_k\\
u_2 & u_3 & u_4 & \ldots & u_{k+1}\\
u_3 & u_4 & u_5 & \ldots & u_{k+2}\\
\vdots & \vdots & \vdots & \ddots & \vdots\\
u_l & u_{l+1} & u_{l+2} & \ldots & u_w
\end{bmatrix},
\end{aligned} \tag{19}$$

where $\mathbf{m}_i = [u_i, \ldots, u_{i+l-1}]^T$ $(1 \le i \le k)$ where $k$ is the number of segments and $\mathbf{M}$ is a Hankel matrix which means that it has equal elements on the anti-diagonals.

#### 4.1.2 Step 2. Singular Value Decomposition

Singular Value Decomposition is a factorisation of the Hankel matrix $\mathbf{M}$ as $\mathbf{M}_{l \times k} = \mathbf{U} \cdot \mathbf{\Sigma} \cdot \mathbf{V}^T$, where $\mathbf{U}$ is an $l \times l$ unitary matrix, $\mathbf{\Sigma}$ is a $l \times k$ rectangular diagonal matrix with

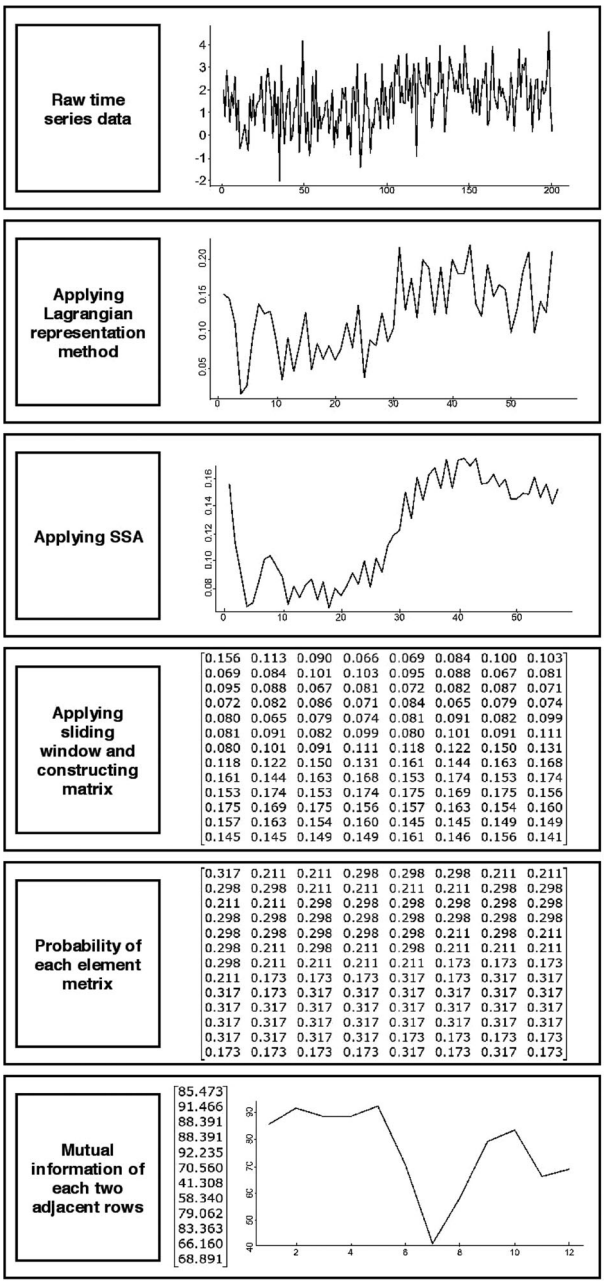



Fig. 2. An example of the proposed method for change detection.

non-negative numbers on the diagonal which are the square roots of the non-zero eigenvalues of $\mathbf{M} \cdot \mathbf{M}^T$, and $\mathbf{V}$ is an $k \times k$ unitary matrix. It can be proven that the trajectory matrix $\mathbf{M}$ can be expressed as the summation of $d$ matrices which called elementary matrices $\mathbf{M} = \mathbf{E_1} + \mathbf{E_2} + \cdots + \mathbf{E_d}$ where $d$ is the number of eigenvalues of $\mathbf{M} \cdot \mathbf{M}^T$ in decreasing order $(\delta_1, \ldots, \delta_d)$ and $\mathbf{E_j} = \sqrt{\delta_j} \cdot \mathbf{u}_j \cdot \mathbf{v}_j^T$ for $j = 1, 2, \ldots, d$ where $\mathbf{u}_j$ is the corresponding eigenvector, and the vector $\mathbf{v}_j$ is obtained from $\mathbf{v}_j = \mathbf{M}^T \cdot \mathbf{u}_j / \sqrt{\delta_j}$ [28]. The plot of eigenvalues in decreasing order is called the Singular Spectrum, and the last matrices represent noise in the sequence [28].

#### 4.1.3 Step 3. Reconstruction (Diagonal Averaging)

In this step, each elementary matrix $\mathbf{E_j}$ is transformed into a sequence of length $w$ which called principal component by applying diagonal averaging method. Diagonal averaging algorithm is as follows:

Let $\mathbf{E_j} = [E_{qs}]_j$, $1 \le j \le d$, $1 \le q \le l$ and $1 \le s \le k$. The principal component $\mathbf{p}_j = [p_h]_j$ where $0 \le h < w$, corresponding to this elementary matrix is [28]:
for $0 \le h < \min(l,k) - 1$

$$p_h = \frac{1}{h+1} \sum_{m=1}^{h+1} N_{m,h-m+2}, \tag{20a}$$

for $\min(l,k) - 1 \le h < \max(l,k)$

$$p_h = \frac{1}{\min(l,k)} \sum_{m=1}^{\min(l,k)} N_{m,h-m+2}, \tag{20b}$$

for $\max(l,k) \le h < w$

$$p_h = \frac{1}{w-h} \sum_{m=h-\max(l,k)+2}^{w-\max(l,k)+1} N_{m,h-m+2}. \tag{20c}$$

It can be shown that the squared norm of each elementary matrix is equal to corresponding eigenvalue and the squared norm of the trajectory matrix is the sum of the squared norms of the elementary matrices [28]. To smooth the sequence, we divide the principal components in two groups: dominant and noise. We only take the combination of dominant principal components as the new sequence. As the two first principal components have the highest eigenvalues, we take them as the dominant group.

### 4.2 Matrix Construction

After applying SSA and obtaining a smoother sequence, we represent the data as a matrix by applying sliding windows. The matrix has 8 columns so the window length is equal to 8, and we take slide of the window as half of the window length (see Algorithm 3). Obviously, the other size matrices can also be used here. However, we use a matrix with 8 columns to explain our approach and to keep the complexity of the SVD low (more experiments on choosing the window size has been done in Evaluation section). After applying the sliding window, we have a matrix $\mathbf{Q} = [Q_{ij}]$ which is $m \times 8$. We get the maximum and minimum numbers of this matrix elements and calculate the range of the element values ($r = \max - \min$). We then divide $r$ into four equal sized ranges with length of $\alpha = r/4$ (Eq. (21))

$$\begin{aligned} r &= r_1 + r_2 + r_3 + r_4 \\ &= [\min, \min + \alpha) + [\min + \alpha, \min + 2\alpha) \\ &\quad + [\min + 2\alpha, \min + 3\alpha) + [\min + 3\alpha, \max]. \end{aligned} \tag{21}$$

After that, we take each element of matrix $\mathbf{Q}$ and convert it to its corresponding range number and construct $\mathbf{Q}' = [Q'_{ij}]$

$$Q'_{ij} = \begin{cases} 1 & \text{if } Q_{ij} \in [\min, \min + \alpha) \\ 2 & \text{if } Q_{ij} \in [\min + \alpha, \min + 2\alpha) \\ 3 & \text{if } Q_{ij} \in [\min + 2\alpha, \min + 3\alpha) \\ 4 & \text{if } Q_{ij} \in [\min + 3\alpha, \max] \end{cases}. \tag{22}$$

### 4.3 Probability and Joint Probability Calculation

We then calculate the probability distribution of each element in matrix $\mathbf{Q}'$. The probability of each element is the number of times that an element occurs in the matrix $\mathbf{Q}'$ divided by the total number of the elements in the matrix. For calculating the joint probability of each pair of elements in matrix $\mathbf{Q}'$, we consider the number of times that the pairs occur in a row and divide it by the total number of rows in matrix $\mathbf{Q}'$.

**Algorithm 3.** Sliding Window Algorithm

**Input:** $\mathbf{x}, window, slide$
**Output:** $[\mathbf{row}_1, \ldots, \mathbf{row}_{rows}]$

```
window ← window length
slide ← slide length
x = [x_1, ..., x_w] input
rows ← ⌈(w−window+1)/slide⌉
sectionStart ← 1
j ← 1
while j ⩽ rows do
    row_j ← [x_sectionStart, ..., x_(sectionStart+window−1)]
    sectionStart ← sectionStart + slide − 1
    j ← j + 1
end while
```

### 4.4 Mutual Information Calculation

Mutual information measures the relationship between two events. In other words, it measures how much information one event communicates about the other one. We calculate the mutual information of each pair of adjacent rows in matrix $\mathbf{Q}' = [\mathbf{q}'_1, \ldots, \mathbf{q}'_m]^T$. The formal definition of mutual information is given by [29]

$$I(\mathbf{q}'_i, \mathbf{q}'_{i+1}) = \sum_{x \in \mathbf{q}'_i} \sum_{y \in \mathbf{q}'_{i+1}} P(x,y) \log \frac{P(x,y)}{P(x)P(y)}, \tag{23}$$

in which, $P(x)$ is the probability of element $x$ in row $\mathbf{q}'_i$, $P(y)$ is the probability of element $y$ in row $\mathbf{q}'_{i+1}$ and $P(x,y)$ is the joint probability of elements $x$ and $y$.

### 4.5 Change Points Detection

For finding the change points, we find the triangle fluctuations, a downward trend following an upward trend or an upward trend following a downward trend, among every three adjacent mutual information which means among $I(\mathbf{q}'_{i-1}, \mathbf{q}'_i)$, $I(\mathbf{q}'_i, \mathbf{q}'_{i+1})$ and $I(\mathbf{q}'_{i+1}, \mathbf{q}'_{i+2})$ for $i = 1, \ldots, (m-2)$ where $m$ is the number of rows in matrix $\mathbf{Q}'$. For each triangle fluctuation, we can say the change point is among the $(i-1)$th, $i$th and $(i+1)$th rows where each row has 8 elements and each element is the Lagrangian representation of a window in the PAA step of our proposed representation and aggregation method. We take the middle row as the possible range for change point, which is the $i$th row. So we find 8 elements, all in a row, that each of them has the possibility of being a change point.

As an example, we show one of the samples from the dataset with a change in standard deviation in Fig. 3. We show the real change in the sample, and we also show the detected change points ranges in the mutual information line plot which shows that the detected change points ranges are 5th, 8th and 10th rows and we show every 8 elements of each row in the last line plot. Overall, the method detected 24 samples as possible change points which is 12 percent (or $\frac{3}{25}$) of the raw samples.

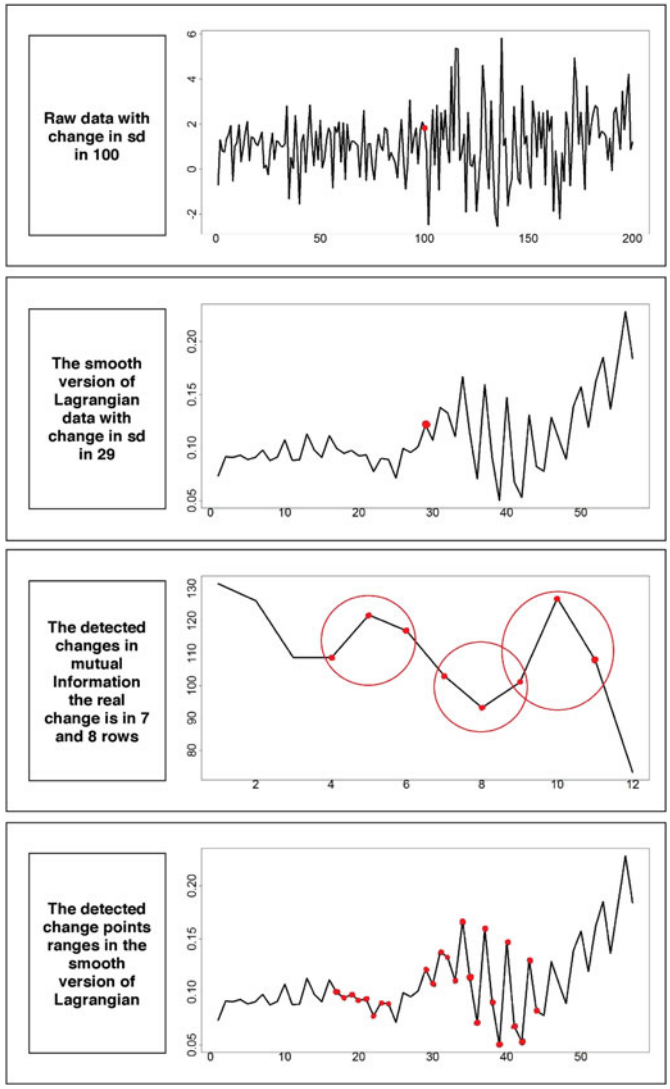



Fig. 3. An example of the change detection.

## 5 Evaluation

To evaluate our method, we perform clustering based on a set of common datasets. We have introduced some of the common algorithms in Section 2. In this section, we compare our algorithm with the state-of-the-art methods that represent an improvement in comparison with elementary ones. The codes for the existing methods have been accessed from the authors' public repositories, and when they were not available, we used R and Python (Numpy and Sympy libraries have been used) to program them. We perform each technique using several datasets to demonstrate the type of problems for which our method performs better than the existing solutions. To evaluate our proposed change point detection method, we compared our method with one of the widespread change points detection methods using three different datasets.

**Algorithm 4.** Change Point Detection Algorithm

**Input:** **lag**
**Output:** *all possible* **r**

$w \leftarrow$ length of sample after PAA
**lag** $= [lag_1, \ldots, lag_w]$ Lagrangian representation of a sample
**ssa** $= [ssa_1, \ldots, ssa_w] \leftarrow SSA(lag)$
$\mathbf{M} \leftarrow SlidingWindow(\mathbf{ssa}, window = 8, slide = 4)$
$n \leftarrow 8 \times$ rows($\mathbf{M}$)
$min \leftarrow \min(\mathbf{M})$
$max \leftarrow \max(\mathbf{M})$
$l = \frac{(max - min)}{4}$
**for** $i$ from 1 to 4 **do**
  **for** all elements of $\mathbf{M}$ as $m$ **do**
    **if** $min < m < min + l$ **then**
      $m' \leftarrow i$
    **end if**
  **end for**
  $min \leftarrow min + l$
**end for**
**for** all elements of $\mathbf{M'}$ as $M'$ **do**
  $probabilty(M') = \frac{\text{number of } M' \text{ in } \mathbf{M'}}{n}$
**end for**
**for** all pairs of elements of $\mathbf{M'}$ as $(M', N')$ **do**
  $probabilty(M', N') = \frac{\text{number of } (M', N') \text{ in rows of } \mathbf{M'}}{\text{rows of } \mathbf{M'}}$
**end for**
**for** all pair rows of $\mathbf{M'}$ as $(\mathbf{r}, (\mathbf{r}+\mathbf{1}))$ **do**
  $mutualInfo(\mathbf{r}, (\mathbf{r}+\mathbf{1})) = \sum_{M' \in \mathbf{r}} \sum_{N' \in (\mathbf{r}+\mathbf{1})} (probabilty_{(M',\ N')}) \times \log\left(\frac{probabilty(M', N')}{probabilty(M') \times probabilty(N')}\right)$
**end for**
**for** all quartet of rows in $\mathbf{M'}$ as $((\mathbf{r}-\mathbf{1}), \mathbf{r}, (\mathbf{r}+\mathbf{1}), (\mathbf{r}+\mathbf{2}))$ **do**
  **if** $(mutualInfo((\mathbf{r}-\mathbf{1}), \mathbf{r}), mutualInfo(\mathbf{r}, (\mathbf{r}+\mathbf{1})), mutualInfo((\mathbf{r}+\mathbf{1}), (\mathbf{r}+\mathbf{2}))$ is a triangle fluctuation **then**
    **r** is a possible row for change point
  **end if**
**end for**

### 5.1 Datasets

The datasets are from the UCR Time-series Classification Archive [30] and all of them have training and testing samples. For number of windows in PAA for each dataset, we used the reduced length in [1] and in some of the experiments we used different number of windows to compare the results. We have also evaluated our pattern representation method using additional real-world dataset in a healthcare application, and using different window sizes. The results of these experiments are shown in Appendices D and C, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TKDE.2019.2961097. In the SAX algorithm, we set $\alpha$ as 10 for all the datasets. The UCR datasets[2] that have been used in experiments are described below:

#### 5.1.1 Arrow Heads

The Arrow heads dataset has 211 time-series of length 251, which have been clustered into three classes. This dataset consists of the conversion of the shapes of the projectile points, arrow heads, to time-series [31]. Projectile points can be divided into different classes based on their location. The reduced length for this dataset in [1] is 72. So in the PAA we divide the samples of this dataset in 72 equal sized windows.

2. http://www.cs.ucr.edu/~eamonn/time_series_data/

#### 5.1.2 Lightning 7

The Lightning 7 dataset has 143 samples of length 319, which is the transient electromagnetic events associated with lightning. This data has been gathered with a sample rate of 50 MHz for 800 microseconds and after that a Fourier transformation and some collapsing and smoothing have been performed on the input data to produce series of length 637. This dataset has seven classes which describe the ways of lightning production. With information from [1], we reduced the length of each time-series to 96 windows.

#### 5.1.3 Coffee

The Coffee dataset contains 56 time-series with length of 286, which have been clustered into two classes. In this dataset, the data has been gathered using Fourier transform infrared spectroscopy which is an alternative to wet chemical methods for verification of different coffee species: Arabica and Robusta [32]. Using [1], we reduced the length of each sample into 84 windows.

#### 5.1.4 Ford A

This dataset has 1320 time-series of length 500, which has two classes. This data has been generated for classification competition and the clustering is based on existence of a symptom which has been measured with engine noise [1].

We changed the length of each time-series using PAA with 111 equal sized windows.

#### 5.1.5 Proximal Phalanx Outline Age Group

This dataset has 605 samples of 80 observations. The dataset has been gathered from Children's Hospital in Los Angeles, and the information is collected from radiography images to test the hands' and bones' outline detection and to validate if they are useful in bone age prediction [33]. This dataset has information about the outline of Proximal Phalanx of hand, the bone closest to the hand, and it has been used to predict the subject age group (There are three different age groups) [33]. And we reduced the number of observations to nine.

### 5.2 Clustering

Clustering is an unsupervised machine learning method which deals with finding a structure for a collection of unlabelled data and groups the data into different clusters. A cluster is a collection of data which is similar but dissimilar to the data from other clusters. In other words, clustering minimises the within-group-object similarity and maximises between-group-object dissimilarity. With clustering, we can find patterns in our time-series data. After transforming our data using the proposed method, we applied two different clustering algorithms. First, the centroid based algorithm, k-means, to cluster the samples. Second, the connectivity based algorithm, hierarchical agglomerative clustering with complete linkage method, in which the distance between two clusters is the maximum distance between their objects.

In our evaluation, we have a dataset of $n$ samples $\{\mathbf{u_1}, \ldots, \mathbf{u_n}\}$. Each sample is the unit vector representation of a time-series and has $w$ dimensions ($\mathbf{u_i} = [u_{i1}, \ldots, u_{iw}]$ for $i = 1, \ldots, n$). In clustering, one of the important elements we should consider is distance measure. In our evaluation, we used two different distance measures:

1) Euclidean distance: First we compute the average vector $\vec{\mathbf{m}} = [m_1, \ldots, m_w]$ of all the $t$ samples

$$m_i = \frac{\sum_{k=1}^{l} u_{ki}}{t} \quad \text{for} \quad i = 1, \ldots, w. \tag{24}$$

Then we calculate the cosine similarity (the cosine of the angle between two vectors) between each sample and $\vec{\mathbf{m}}$ (see Eq. (25)). After that, we have a vector of $w$ dimensions which is our clustering input and we perform our clustering algorithms based on euclidean distance.

2) Cosine dissimilarity: Our $n$ samples are our inputs, but in the clustering algorithms, we use cosine dissimilarity (which is one minus the cosine similarity) to compute the distance between our samples

$$\text{Similarity} = \frac{\vec{\mathbf{x}}\vec{\mathbf{y}}}{\|\vec{\mathbf{x}}\|\|\vec{\mathbf{y}}\|} = \frac{\sum_{i=1}^{w} x_i y_i}{\sum_{i=1}^{w} x_i \sum_{i=1}^{w} y_i}. \tag{25}$$

To determine the number of clusters, we used a fixed number in our clustering algorithms because we were aware of the number of clusters of datasets. For evaluating our cluster quality, we used the Silhouette coefficient. The Silhouette coefficient is a measure of how similar an object is to its own cluster compared to other clusters. Ranges from -1 to 1 and a higher value means that the object has been clustered well. The Silhouette coefficient of object $i$ is defined as

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}, \tag{26}$$

where $a(i)$ is average distance between $i$ and all other objects within the cluster and $b(i)$ is the lowest average distance of object $i$ to all objects in any other clusters which $i$ is not a member. We use this value to compare the performance of different clustering algorithms.

TABLE 1
The Silhouette Coefficient of Each Method Using Each Dataset as Input

(a) Using euclidean distance.

| Model \Dataset | Arrow Head | Lightning 7 | Coffee | Ford A | Proximal |
|---|---|---|---|---|---|
| **Lagrangian_HC** | 0.61 | 0.59 | 0.73 | 0.51 | 0.74 |
| **Lagrangian_Kmeans** | 0.67 | 0.57 | 0.69 | 0.56 | 0.62 |
| **BEATS_HC** | 0.58 | 0.24 | 0.25 | 0.35 | 0.4 |
| **SAX$_{SD}$_HC** | 0.44 | 0.26 | 0.25 | 0.00 | 0.48 |
| **Raw Data_HC** | 0.49 | 0.13 | 0.29 | 0.02 | 0.43 |
| **Raw Data_Kmeans** | 0.47 | 0.12 | 0.33 | 0.05 | 0.46 |

(b) Using cosine dissimilarity.

| Model \Dataset | Arrow Head | Lightning 7 | Coffee | Ford A | Proximal |
|---|---|---|---|---|---|
| **Lagrangian_HC** | 0.66 | 0.33 | 0.45 | 0.04 | 0.43 |
| **Lagrangian_Kmeans** | 0.67 | 0.27 | 0.47 | 0.10 | 0.61 |
| **BEATS_HC** | 0.58 | 0.24 | 0.25 | 0.35 | 0.4 |
| **SAX$_{SD}$_HC** | 0.44 | 0.26 | 0.25 | 0.00 | 0.48 |
| **Raw Data_HC** | 0.49 | 0.13 | 0.29 | 0.02 | 0.43 |
| **Raw Data_Kmeans** | 0.47 | 0.12 | 0.33 | 0.05 | 0.46 |

We compare our algorithm with $\mathrm{SAX}_{SD}$ and BEATS. We chose $\mathrm{SAX}_{SD}$ as it also uses PAA as an intermediate step and represents time-series data in a finite domain. BEATS has been chosen as it provides the highest results compared with the other state-of-the-art algorithms discussed in Section 2. The results of clustering experiments on the datasets are shown in Table 1. The result of Silhouette coefficient with using the euclidean distance and average vector have been shown in Table 1a, and the results of applying cosine dissimilarity have been shown in Table 1b. Regarding BEATS, we only used the hierarchical clustering for BEATS and $\mathrm{SAX}_{SD}$ because of their higher performance. For BEATS, we used cosine dissimilarity, and for $\mathrm{SAX}_{SD}$, we used *MINDIST* [8]. We also compared our proposed algorithm with the clustering algorithms applied to the original data.

For the Arrow Head dataset, our proposed method outperforms BEATS and $\mathrm{SAX}_{SD}$ in both hierarchical and K-means clustering methods. In Table 1a with using the hierarchical clustering, our method performs around 3 percent better than BEATS and 17 percent better than $\mathrm{SAX}_{SD}$. And in Table 1b, the performance is 8 percent higher than BEATS and 22 percent higher than $\mathrm{SAX}_{SD}$. For the Lightning 7 dataset, our method performs better in all the clustering methods. However, it works better in Table 1a by around 31 and 35 percent higher. In Table 1b, it only outperforms the existing methods by 9 percent.

In the Coffee dataset, we experience the same type of results as the Lightning 7 dataset, which shows more improvements in Table 1a (around 44–48 percent). However, with around 20–22 percent better performance in Table 1b, our proposed method shows its significant improvements. In the Proximal dataset, the results are a little different with a huge improvement in Table 1a with the hierarchical clustering which is around 34 percent better than BEATS and a small 3 percent improvement in Table 1b with the hierarchical clustering compared to BEATS. However, $\mathrm{SAX}_{SD}$ in this dataset performs better than our algorithm with a small difference that can occur due to the number of alphabets.

The only dataset which our proposed method does not perform better in all the clustering methods is Ford A dataset. In this dataset, our method does not provide higher performance as shown in Table 1b. However, it outperforms $\mathrm{SAX}_{SD}$ in all the clustering methods. However, as shown in Table 1a, our method performs around 20 percent better than BEATS. As shown in Table 1, our proposed method outperforms both hierarchical and K-means clustering methods. We have also performed further evaluation on the performance of the clustering method in Appendix B, available in the online supplemental material. Our proposed method uses a Lagrangian multiplier to aggregate the time-series and represents them as unit vectors and preserves the important information of time-series in a smaller length. In the next section, we discuss how these properties allow our proposed method to provide an effective approach for change detection in time-series. There are also further experiments using our proposed method on a real-world higher dimensional dataset, which are reported in Appendix D, available in the online supplemental material.

### 5.3 Change Detection Method

To evaluate the change point detection method, we have generated datasets which have specific change points. We have generated three datasets with 200 observations in each sample which have normal distribution, and there is a change point in 100th timestamp of each sample.

In dataset $A$, the initial 100 timestamps of each sample have normal distribution with mean of 1 and standard deviation of 1, $\mathcal{N}(1,1)$, and the remaining 100 observations have normal distribution with mean of 2 and standard deviation of 1, $\mathcal{N}(2,1)$. Each sample of the dataset $B$ has normal distribution with mean of 0 and standard deviation of 1, $\mathcal{N}(0,1)$ for the initial 100 observations and mean of 0 and standard deviation of 2 for the remaining observations, $\mathcal{N}(0,4)$. And the dataset $C$ has normal distribution with changes in both mean and standard deviation values at each 100th observation. For a better presentation of datasets

$$
\begin{aligned}
A &= \{x_i | f(x_i) = \mathcal{N}(1,1) \text{ for } i = 1,\ldots,100 \ \ \& \ \ f(x_i) \\
&\quad = \mathcal{N}(2,1) \text{ for } i = 101,\ldots,200\} \\
B &= \{x_i | f(x_i) = \mathcal{N}(0,1) \text{ for } i = 1,\ldots,100 \ \ \& \ \ f(x_i) \\
&\quad = \mathcal{N}(0,4) \text{ for } i = 101,\ldots,200\} \\
C &= \{x_i | f(x_i) = \mathcal{N}(1,1) \text{ for } i = 1,\ldots,100 \ \ \& \ \ f(x_i) \\
&\quad = \mathcal{N}(2,4) \text{ for } i = 101,\ldots,200\}.
\end{aligned}
$$

We compare our change detection method with CUmulative SUM method (CUSUM) [9], [10] and the Bayesian change detection method [11] which are two of the well-known and widespread change point detection techniques. To evaluate the influence of our representation technique in detecting change points, we applied CUSUM and the Bayesian method on both raw datasets (each sample with 200 observations) and Lagrangian multiplier based representation of the datasets. With raw datasets as input for CUSUM: for dataset $A$ (with change in mean value) only in 561 samples of 1000 samples, there was a change point; in dataset $B$, CUSUM performed lower and could not find the true change points in any of the samples; and in dataset $C$, only in 38 of 1000 samples, the true change points were detected using CUSUM. However, by applying our proposed representation method before CUSUM, provides a significant improvement in the change point detection ability of CUSUM. For dataset $A$, the number of samples with detection of true change point improved around 43 percent and became 990 out of 1000 samples. In dataset $C$ the improvement was around 60 percent, and it became 642 samples. In dataset $B$ the improvement was not significant, and the number of samples with detection of true change point was 5. For the Bayesian change detection method, the difference between using the original data and the Lagrangian represented data is not significant and only in dataset $C$ there is a 4 percent improvement. This could be due to high sensitivity of this method on the location of change points.

To compare the proposed change detection method with the methods mentioned above and since our method finds a row or a vector of 8 elements as a possible change point range, for a fair comparison, we consider the matrix which has been constructed in our change point detection method, as the input of CUSUM and the Bayesian methods. As a result, CUSUM and the Bayesian method find rows as change points, too. In other words, to evaluate our change point detection method, we first applied our Lagrangian representation method on the datasets with the number of windows equals to 57 ($w = 57$). Particularly, we aggregated each sample ($\mathbf{x} = [x_1, \ldots, x_{200}]$) of datasets into a Lagrangian representation ($\mathbf{x}' = [x'_1, \ldots, x'_{57}]$). To reduce the impact of noise, we then applied SSA and made each Lagrangian representation smoother. We then apply the sliding window algorithm with a fixed window length equals to 8 and a slide equals to half of the window length (4) which constructed matrices with 13 rows and 8 columns. After matrix construction, we apply our method, CUSUM and the Bayesian method to each matrix. We then check if the true change point is among the elements which have been detected as possible change points in our proposed method, CUSUM and the Bayesian method. The true change point happens in 100th timestamp ($x_{100}$) of each sample and it aggregates into 29th timestamp ($x'_{29}$) after applying the Lagrangian representation method. After matrix construction, this change point ($x'_{29}$) appears in two rows of the matrix (rows 7 and 8).

In dataset $A$, the number of samples with real change point that have been detected by our method is 800 out of 1000. In the dataset $B$, this number is 696 and in dataset $C$ this number is 759 out of 1000 samples.

TABLE 2
Results of Applying the Proposed Method and CUSUM and Bayesian on Datasets

| Dataset | The Proposed Method | | | | CUSUM | | | | Bayesian | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | recall | precision | F1 score | accuracy | recall | precision | F1 score | accuracy | recall | precision | F1 score | accuracy |
| **A** | 80% | 18.28% | 29.77% | 65.68% | 100% | 9.29% | 17.00% | 11.24% | 63.5% | 54.9% | 58.93% | 91.95% |
| **B** | 69.6% | 17.52% | 28.00% | 67.45% | 61.2% | 13.55% | 22.19% | 60.99% | 44.4% | 17.3% | 24.62% | 72.29% |
| **C** | 75.9% | 18.26% | 29.44% | 66.93% | 100% | 9.62% | 17.56% | 14.64% | 64.2% | 27.04% | 38.05% | 81% |

To evaluate the proposed method, we used accuracy, the degree to which the result of a calculation conforms to the correct value, recall which is the ability of the model to find all the relevant cases, precision which is the proportion of relevant cases among the retrieved cases by the model and F1 score, the combination of recall and precision. To calculate the value of recall, we need to know True Positive (TP) and False Negative (FN). TP is the amount of data points classified as positive or, in our case, the change points which are true change points. FN is the number of points that the method identifies as negative that actually are positive (see Eq. (27))

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{27}$$

For calculation of precision rather than TP we use False Positive (FP) counts which refers to the data points that the method incorrectly detected as the change points while they are not. The equation is

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}. \tag{28}$$

F1 score calculation is as follow:

$$\text{F1} = 2 \times \frac{(\text{precision} \times \text{recall})}{(\text{precision} + \text{recall})}. \tag{29}$$

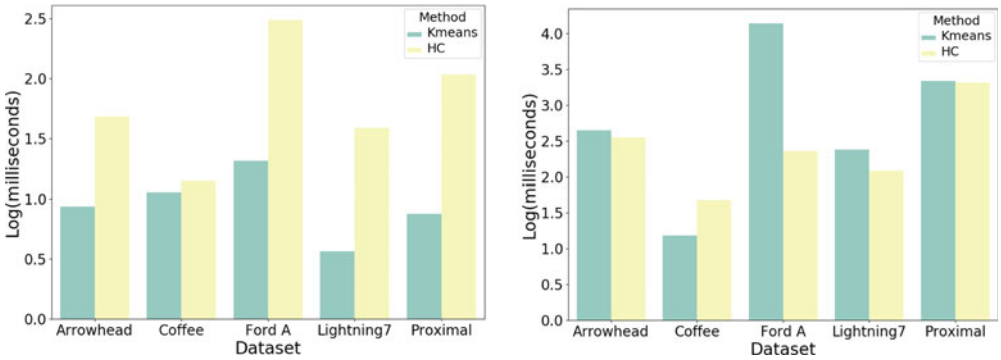
For accuracy, we also use True Negative (TN) which is number of data points that are negative and defined correctly as negative. (see Eq. (30))

$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}. \tag{30}$$

The results of applying the proposed change detection method, CUSUM and the Bayesian method on datasets have been shown in Table 2. It can be seen that the CUSUM detected all the change points in datasets $A$ and $C$ and the recall is 100 percent in these datasets but because of the lower precision than the proposed method its ability to detect only the relevant cases is lower than our method. However, our proposed method with lower recall in $A$ and $C$ datasets has higher precision, F1 score and accuracy compare to CUSUM and it is more relevant and useful in change detection. In dataset $B$, the proposed method performs even better and it also has higher recall (69.6 percent compare to 61.2 percent) and because of the higher accuracy (67.45 percent) and F1 score (28.00 percent), it is more useful and reliable. The result of the Bayesian change detection method is the probability that each sample represent a change point. To compare the results of the Bayesian change point detection with other methods, we chose a threshold of 0.8. This indicates that for the points that the algorithm provides an 80 percent or higher probability of being a change point, we consider it as a detected change point. As shown in Table 2, the Bayesian method performs well in dataset $A$ that includes change in the mean value; however, the recall value is around 16 percent less than our proposed algorithm. In dataset $B$ with changes in the standard deviation, our proposed method performs higher than the Bayesian method, and in dataset $C$, our proposed method has higher recall (11 percent). Our proposed method works better on datasets with changes in standard deviation rather than mean value. However, it finds a range of possible timestamps for change point which extends the applicability of our solutions to the scenarios which an approximation of the change point is sufficient.

### 5.4 Processing Time

The representation, pattern creation and abstraction methods for time-series data often work with statistical measurements, symbolic representations, or stream processing. These methods have disadvantages such as normalisation as pre-processing step, not preserving the shape of time-series and not being efficient for large datasets. In Section 1, we have mentioned IoT data analysis as one of the motivations of this work. In IoT applications, memory is an important requirement to consider. In our proposed pattern representation method, as the process is applied to the segments, there is no need to store large volumes of data. In other words, one of the advantages of the proposed method is the suitability and low memory usage, which make it appropriate to run on the edge devices. Using a computer with an Intel Core i7-6700 Processor, 32 GB RAM Memory, Windows 7 operating system and PyCharm 2018.1.3 IDE, we have measured the average processing time of the Lagrangian representation for a time-series in Ford A dataset, as it has the longest time-series among the datasets which is around 400-500 milliseconds. We have also shown the relative comparison of the processing time of each clustering algorithm applied to each dataset in Figs. 4 and 5. In Fig. 5, we show the processing time in milliseconds and outside the log space which makes the representation of the differences larger. For FordA dataset, the processing time is much higher because of the larger volume of data it has compared to other datasets. In the proposed change detection method, the average processing time for each time-series in all datasets is around 45-65 milliseconds. We carried out further analysis to measure the delay in identifying the change points. With further analysis, we found out that in our experiments, in average, we will have five samples delay before detecting the change point. The algorithm uses a moving window and each new value in time-series data is used to update the value and checked for the change point detection. In cases of dynamic or




(a) Using Euclidean distance. (b) Using cosine dissimilarity.

Fig. 4. Running time in log(milliseconds) for the clustering algorithms using the Lagrangian representations.

very slow-changing data streams, the delay in the number of data points before detecting the change point could change. We have also carried out further experiments with various window sizes to show the effect of the chosen window size in our change point detection. The results of these experiments are reported in Appendix A, available in the online supplemental material.

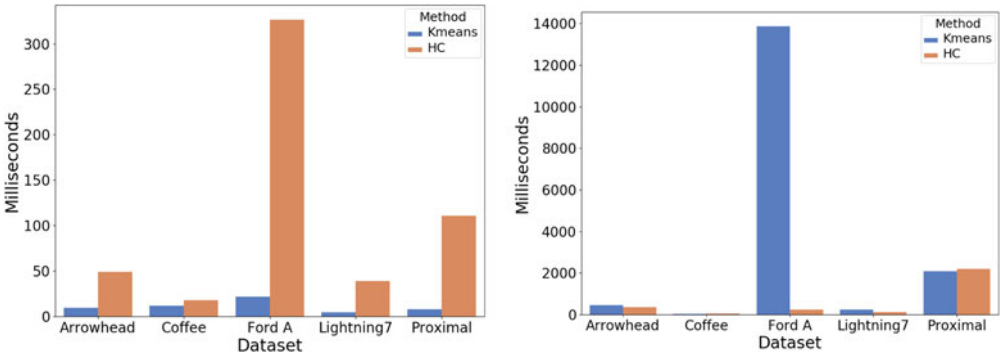
## 6 Complexity Analysis

The time complexity is represented as a function of the length of time-series ($N$). There are different steps in Lagrangian based representation and the one which has the most impact on the time complexity is solving the system of equations from the Lagrangian Multiplier technique. However, regarding the discussion in 3.2 section, this step is only normalisation of the PAA vector representation of the time-series which has the time complexity of $\mathcal{O}(N)$. PAA method and other steps of the proposed representation method have the time complexity of $\mathcal{O}(N)$. So, the time complexity of the proposed method is $\mathcal{O}(N)$.

For the time complexity of the change point detection algorithm, we should consider the time complexity of SSA which contains SVD. SVD is the most time consuming step is SSA. SSA only uses the Partial SVD (i.e., uses the $k$ components of each matrix) and the matrix used to decompose in SVD is Hankel matrix which makes the matrix-vector multiplications faster. There are different implementations for SVD with different time complexities. In our implementation of the proposed change points detection method, we used the SSA function from the "Rssa" package in R which is the fastest implementation of SSA [34]. In this implementation, they used Fast Fourier Transform (FFT) with complexity of $\mathcal{O}(kN \log(N))$ where $k$ is the number of components from SVD. The "Rssa" package uses Lanczos-based Partial SVD with the complexity of $\mathcal{O}(kN \log(N))$ [34]. Lanczos is an iterative algorithm which finds $k$ most useful eigenvalues and eigenvectors of a Hermitian matrix, a matrix that is equal to its own conjugate transpose matrix. The package also uses the FFT for reconstruction step. With these changes in implementation, it reduces the time complexity of SSA from $\mathcal{O}(N^3)$ to $\mathcal{O}(kN \log(N) + k^2N)$. So, the overall time complexity of the change points detection algorithm will be $\mathcal{O}(N + kN \log(N) + k^2N + N^2 + N^3)$ where $\mathcal{O}(N^3)$ is the time complexity of computing the joint probability and $\mathcal{O}(N^2)$ is the time complexity of computing the probability matrix and there are also Lagrangian representation, sliding window, range construction, computing the mutual information of each adjacent pair of rows and finding the triangle fluctuations steps which have time complexity of $\mathcal{O}(N)$. As $\mathcal{O}(N^3)$ is the dominating term, the time complexity of change point detection is $\mathcal{O}(N^3)$.

## 7 Conclusion

In this paper, we introduced a new method for aggregating and representing time-series data. We use Piecewise Aggregate Approximation (PAA) to reduce the length of time-series data. We then use Lagrangian multiplier to represent time-series as unit vectors. The proposed method preserves the key information in a lower dimensional vector. Our method, unlike PAA which only represents data in a sequence of




(a) Using Euclidean distance. (b) Using cosine dissimilarity.

Fig. 5. Running time in milliseconds for the clustering algorithms using the Lagrangian representations.

continuous numbers, provides a representation of the patterns in time-series. We have shown that our representation is more efficient in comparison with other representation methods. The results of the Lagrangian multiplier provide vector representations which can be used to analyse patterns and changes in time-series data. We describe our model by applying time-series clustering and also by extending the work to develop a change point detection algorithm.

For change point detection, we used mutual information and entropy. Because entropy is sensitive to noise, we first use Singular Spectrum Analysis (SSA) technique to make our Lagrangian representation smoother. After that, we use a matrix construction and an entropy model to find a range for change points. Compared to the-state-of-the-art algorithms, our method shows significant improvements in clustering time-series data. It offers around 20 percent higher performance than Blocks of Eigenvalues Algorithm for Time-series Segmentation (BEATS) and Symbolic Aggregate approXimation (SAX) methods. For change point detection method assessment, we have shown that our proposed method provides higher accuracy compared with CUmulative SUM method (CUSUM) which is around 37 percent higher. In comparison with the Bayesian method, our proposed method outperforms in dataset $B$ but in other datasets it only shows higher sensitivity compared to the Bayesian method.

In our evaluation, we used fixed values for window length and sliding windows. The future work will focus on adaptive window size selection and extending the solution to multi-variate and non-stationary time-series data. We will also work on using the constructed patterns in time-series forecasting and change point prediction in dynamic time-series datasets.

## Acknowledgments

This work was supported by Care Research and Technology Centre at the UK Dementia Research Institute and TIHM for Dementia project. The work was also partially supported by the European Commissions Horizon 2020 (EU H2020) IoT-Crawler project (http://iotcrawler.eu/) under contract number: 779852.

## References

[1] A. Gonzalez-Vidal, P. Barnaghi, and A. F. Skarmeta, "BEATS: Blocks of eigenvalues algorithm for time-series segmentation," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 11, pp. 2051–2064, Nov. 2018.

[2] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra, "Dimensionality reduction for fast similarity search in large time-series databases," *Knowl. Inf. Syst.*, vol. 3, pp. 263–286, 2001.

[3] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos, "Fast subsequence matching in time-series databases," *ACM SIGMOD Rec.*, vol. 23, no. 2, pp. 419–429, 1994.

[4] K.-P. Chan and W.-C. Fu, "Efficient time-series matching by wavelets," in *Proc. 15th Int. Conf. Data Eng.*, 1999, Art. no. 126.

[5] D. Wu, A. Singh, D. Agrawal, A. El Abbadi, and T. R. Smith, "Efficient retrieval for browsing large image databases," in *Proc. 5th Conf. Inf. Knowl. Manage.*, 1996, pp. 11–18.

[6] E. J. Keogh and M. J. Pazzani, "An enhanced representation of time-series which allows fast and accurate classification, clustering and relevance feedback," in *Proc. 4th Int. Conf. Knowl. Discovery Data Mining*, 1998, pp. 239–243.

[7] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra, "Locally adaptive dimensionality reduction for indexing large time-series databases," *ACM SIGMOD Rec.*, vol. 30, no. 2, pp. 151–162, 2001.

[8] J. Lin, E. Keogh, S. Lonardi, and B. Chiu, "A symbolic representation of time-series, with implications for streaming algorithms," in *Proc. 8th ACM SIGMOD Workshop Res. Issues Data Mining Knowl. Discovery*, 2003, pp. 2–11.

[9] E. S. Page, "Continuous inspection schemes," *Biometrika*, vol. 41, no. 1/2, pp. 100–115, 1954.

[10] L. Liu, F. Tsung, and J. Zhang, "Adaptive nonparametric CUSUM scheme for detecting unknown shifts in location," *Int. J. Prod. Res.*, vol. 52, no. 6, pp. 1592–1606, 2014.

[11] R. P. Adams and D. J. MacKay, "Bayesian online change point detection," *Stat*, vol. 1050, p. 19, 2007.

[12] S. Liu, M. Yamada, N. Collier, and M. Sugiyama, "Change point detection in time-series data by relative density-ratio estimation," *Neural Netw.*, vol. 43, pp. 72–83, 2013.

[13] X. Wang, A. Mueen, H. Ding, G. Trajcevski, P. Scheuermann, and E. Keogh, "Experimental comparison of representation methods and distance measures for time-series data," *Data Mining Knowl. Discovery*, vol. 26, no. 2, pp. 275–309, 2013.

[14] T. Pavlidis and S. L. Horowitz, "Segmentation of plane curves," *IEEE Trans. Comput.*, vol. C-23, no. 8, pp. 860–870, Aug. 1974.

[15] H. Shatkay and S. B. Zdonik, "Approximate queries and representations for large data sequences," in *Proc. 12th Int. Conf. Data Eng.*, 1996, pp. 536–545.

[16] B. Havers, R. Duvignau, H. Najdataei, V. Gulisano, A. C. Koppisetty, and M. Papatriantafilou, "DRIVEN: A framework for efficient data retrieval and clustering in vehicular networks," in *Proc. IEEE 35th Int. Conf. Data Eng.*, 2019, pp. 1850–1861.

[17] R. Duvignau, V. Gulisano, M. Papatriantafilou, and V. Savic, "Streaming piecewise linear approximation for efficient data management in edge computing," in *Proc. 34th ACM/SIGAPP Symp. Appl. Comput.*, 2019, pp. 593–596.

[18] Y. Fathy, P. Barnaghi, and R. Tafazolli, "Large-scale indexing, discovery, and ranking for the Internet of Things (IoT)," *ACM Comput. Surveys*, vol. 51, no. 2, 2018, Art. no. 29.

[19] T.-C. Fu, "A review on time-series data mining," *Eng. Appl. Artif. Intell.*, vol. 24, no. 1, pp. 164–181, 2011.

[20] I. Daubechies, *Ten Lectures on Wavelets*, vol. 61. Philadelphia, PA, USA: SIAM, 1992.

[21] G. H. Golub and C. F. van Loan, *Matrix Computations*. Baltimore, MD, USA: The Johns Hopkins Univ. Press, 1996.

[22] R. J. Larsen *et al.*, *An Introduction to Mathematical Statistics and its Applications*, vol. 2. Englewood Cliffs, NJ, USA: Prentice-Hall, 1986.

[23] N. D. Pham, Q. L. Le, and T. K. Dang, "Two novel adaptive symbolic representations for similarity search in time-series databases," in *Proc. 12th Int. Asia-Pacific Web Conf.*, 2010, pp. 181–187.

[24] Y. Sun, J. Li, J. Liu, B. Sun, and C. Chow, "An improvement of symbolic aggregate approximation distance measure for time-series," *Neurocomputing*, vol. 138, pp. 189–198, 2014.

[25] C. T. Zan and H. Yamana, "An improved symbolic aggregate approximation distance measure based on its statistical features," in *Proc. 18th Int. Conf. Inf. Integr. Web-Based Appl. Services*, 2016, pp. 72–80.

[26] A. C. Bovik, *The Essential Guide to Image Processing*. New York, NY, USA: Academic, 2009.

[27] S. Aminikhanghahi and D. J. Cook, "A survey of methods for time-series change point detection," *Knowl. Inf. Syst.*, vol. 51, no. 2, pp. 339–367, 2017.

[28] N. Golyandina, V. Nekrutkin, and A. A. Zhigljavsky, *Analysis of Time Series Structure: SSA and Related Techniques*. London, U.K. /Boca Raton, FL, USA: Chapman and Hall/CRC, 2001.

[29] C. E. Shannon and W. Weaver, "The mathematical theory of communication," University of Illinois Press, 1998.

[30] Y. Chen *et al.*, "The UCR time-series classification archive," Jul. 2015. [Online]. Available: www.cs.ucr.edu/eamonn/time_series_data/

[31] L. Ye and E. Keogh, "Time-series shapelets: A new primitive for data mining," in *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2009, pp. 947–956.

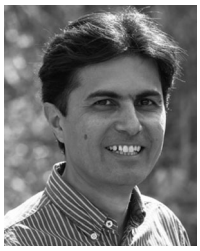
[32] R. Briandet, E. K. Kemsley, and R. H. Wilson, "Discrimination of Arabica and Robusta in instant coffee by fourier transform infrared spectroscopy and chemometrics," *J. Agr. Food Chem.*, vol. 44, no. 1, pp. 170–174, 1996.

[33] L. M. Davis, "Predictive modelling of bone ageing," PhD dissertation, University of East Anglia, 2013.

[34] N. Golyandina and A. Zhigljavsky, *Singular Spectrum Analysis for Time-Series*. Berlin, Germany: Springer, Jan. 2013.

[35] A. Strehl and J. Ghosh, "Cluster ensembles—A knowledge reuse framework for combining multiple partitions," *J. Mach. Learn. Res.*, vol. 3, no. Dec., pp. 583–617, 2002.
[36] S. Enshaeifar *et al.*, "Health management and pattern analysis of daily living activities of people with dementia using in-home sensors and machine learning techniques," *PloS One*, vol. 13, no. 5, 2018, Art. no. e0195605.

**Roonak Rezvani** is currently working toward the PhD degree in machine learning in the Centre for Vision, Speech and Signal Processing (CVSSP), University of Surrey, Guildford, United Kingdom. Her research interests include Internet of Things (IoT), big data analytics, machine learning, and time-series data processing.



**Payam Barnaghi** is professor of machine intelligence with the Department of Electronic and Electrical Engineering and a member of the Centre for Vision, Speech, and Signal Processing (CVSSP), University of Surrey. He is technical lead of the Department of Health/NHS TIHM for dementia Project and Healthy Home programme lead and deputy director of Care Research and Technology Centre, UK Dementia Research Institue (UK DRI). His research interests include machine learning, Internet of Things, adaptive algorithms, information search, and retrieval and their applications in healthcare. He is a senior member of the IEEE and a fellow of the Higher Education Academy.

**Shirin Enshaeifar** is a lecturer in machine learning and deputy technical lead of the Technology Integrated Health Management (TIHM) Project, Department of Electronic and Electrical Engineering, University of Surrey. Her research interests include biomedical signal processing, machine learning, and data analysis with application in brain-computer interface (BCI), technology enabled healthcare services, and Internet of Things (IoT). She is a member of the IEEE.