Lagrangian-based Pattern Extraction for Edge Computing in the Internet of Things

Roonak Rezvani, Shirin Enshaeifar, Payam Barnaghi
Centre for Vision, Speech and Signal Processing (CVSSP)
University of Surrey
{r.rezvani, s.enshaeifar, p.barnaghi}@surrey.ac.uk

***Abstract*—Edge computing can improve the scalability and efficiency of IoT systems by performing some of the analysis and operations on the nodes or on intermediary edge devices. This will reduce the energy consumption, data transmission load and latency by shifting some of the processes to the edge devices. In this paper, we introduce a pattern extraction method which uses both the Lagrangian Multiplier and the Principal Component Analysis (PCA) to create patterns from raw sensory data. We have evaluated our method by applying a clustering method on constructed patterns. The results show that by using our proposed Lagrangian-based pattern extraction method, the existing clustering algorithms perform more accurately - by up to 20% higher compared with the state-of-the-art methods, especially in dealing with dynamic real-world data. We have conducted our evaluations based on synthetic and real-world data sets and have compared the results to the existing state-of-the-art approaches. We also discuss how the proposed methods can be embedded into the edge computing devices in IoT systems and applications.**

***Index Terms*—Internet of Things, edge computing, pattern extraction, Lagrangian Multiplier**

I. Introduction

The rapid advancement of the Internet of Things (IoT) has resulted in a unique opportunity to collect and communicate large volumes of real-world observation and measurement data. IoT has led to the development of smart systems which can be used to enhance user experience and autonomously interact with the surrounding environment [1]. The large volume of data and time consuming computing processes lead to serious challenges in the development of new applications in different fields such as smart homes, smart health-care and smart cities. To tackle this issue, scalable and adaptive methods to analyse large volume and dynamic data streams are needed. Furthermore, collected data streams often need to be pre-processed and filtered for further understanding and obtaining actionable information.

An IoT system can include the Edge and the Cloud layers; the Edge layer (i.e. IoT sensors, IoT gateway and access points) and the Cloud layer (i.e. Internet connection and the Cloud) [2]. IoT sensors collect huge amount of data and transfer the data to the Cloud for further analysis (See Figure 1). However, transferring huge amount of data is a bottleneck in IoT systems. Edge computing [3] appears as a promising solution to provide storage and processing services near the IoT devices (i.e. sensors). In edge computing the aim is to process data in edge layer rather than the Cloud which helps to improve the performance of the IoT system.

Pattern extraction and representation methods are some of the pre-processing steps which help users to explore the data and understand the underlying structure for extracting meaningful information and detecting anomalies [4]. In the field of stream data (i.e. data gathered from sensors), pre-processing methods aim to overcome challenges in data streams which are volume, heterogeneity and being multivariate.

There have been some works to address these challenges in the area of time-series and data streams pre-processing. Some of these existing researches are Discrete Fourier Transform (DFT) [5], Discrete Wavelet Transform (DWT) [6], Singular Value Decomposition (SVD) [7], Piece-wise Aggregate Approximation (PAA) [8], Adaptive Piece-wise Constant Approximation (APCA) [9], Symbolic Aggregate approXimation (SAX) [10], Extended Symbolic Aggregate approXimation (ESAX) [11], Principal Component Analysis (PCA) [12] and Blocks of Eigenvalues Algorithm for Time-series Segmentation (BEATS) [13].

This work introduces a new pre-processing method in the field of sensor data analysis which can be useful for recognising the structure of data and detecting patterns. In the proposed method, we apply Lagrangian Multiplier and Principal Component Analysis (PCA) to aggregate data streams and extract patterns as a representation method to deal with the quantity and quality of data streams. After that, we use Gaussian Mixture Models (GMM) for clustering the aggregated data and evaluate the performance of the proposed technique. In this paper, the experiments are on recorded real-world sensor data for validation purposes. We suggest that to integrate the proposed method into an IoT system based on edge computing to address the problem of transferring and processing workload in sensory data analysis.

Our proposed pattern extraction method provides high performance and capability for clustering dynamic data and our evaluation results show a Silhouette Coefficient of 0.69 which is around 20% higher compared with other solutions (e.g. using only PCA as pre-processing method).

The organisation of this paper is as follows. Section 2 includes the related work. Section 3 describes the proposed method. Section 4 discusses the results of the experiments on real-world and synthesised data and Section 5 provides a conclusion.

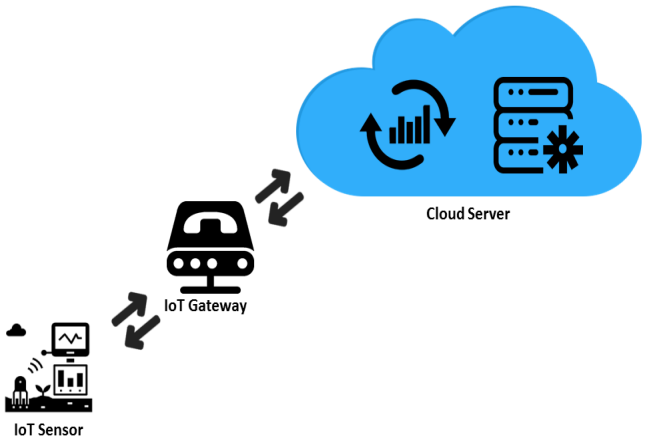


Fig. 1: An overview of a distributed IoT system

II. Related Work

One of the issues in time-series and IoT data streams analysis is the volume of data which is large and expensive in storage and processing time. There have been several suggestions to tackle the issue. Discrete Fourier Transform (DFT) is one of the methods which shows time-series in the frequency space as a finite number of sine or cosine waves that are represented with Fourier coefficients [8].
A time-series $\mathbf{x} = [x_1, \ldots, x_n]$ can be shown as n numbers of Fourier coefficients $\bar{\mathbf{x}}_f$ [5]:

$$\bar{\mathbf{x}}_f = \frac{1}{\sqrt{n}} \sum_{t=0}^{n-1} x_t e^{-j2\pi f \frac{t}{n}} \tag{1}$$

where j is $\sqrt{-1}$. Discrete Wavelet Transform (DWT) is another method which is an inner product of the time-series with the scaled wavelet $\psi(x)$. In this method, time-series $\mathbf{x} = [x_1, \ldots, x_n]$ can be represented as a series:

$$\mathbf{x} = \sum < \psi(x), x > \psi(x) \tag{2}$$

where $\psi(x)$ is the wavelet from the Haar transformation which is the n^{th} derivative of below function [14]:

$$\theta(x) = \begin{cases} 1 & \text{if } 0 \leqslant x < 1 \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

$$\psi(x) = \begin{cases} 1 & \text{if } 0 \leqslant x < \frac{1}{2} \\ -1 & \text{if } \frac{1}{2} \leqslant x < 1 \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

In other words, the Haar transformation applies averaging and differentiation operations on a time-series [6]. However, in transformation based methods for representation, coefficients are global features and computing most of the geometric and mathematical features (i.e. local features) is not possible, although this is important in data analysis methods.
Singular Value Decomposition (SVD) is another method which is data-driven as rather than using sine and cosine waves it uses the input data to compute a decomposition as shown below [7].

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. Note that the diagonal elements of matrix $\mathbf{\Lambda}$ are the eigenvalues of $\mathbf{X}$ and r is the rank of $\mathbf{X}$ [7].

There are other methods besides the transformation based methods, where we can divide time-series data into equal-sized segments. One of the methods in this area is Piece-wise Aggregate Approximation (PAA) which represents each segment with its mean value. Given a time-series like $\mathbf{x} = [x_1, \ldots, x_n]$, PAA will represent it as a vector $\bar{\mathbf{x}} = [\bar{x}_1, \ldots, \bar{x}_w]$ where $w < n$ [8]:

$$\bar{x}_i = \frac{w}{n} \sum_{j=\frac{n}{w}(i-1)+1}^{\frac{n}{w}i} x_j \tag{6}$$

Adaptive Piece-wise Constant Approximation (APCA) is another method in this field which represents time-series with mean values of segments with different length which is based on the activity in the segments (i.e. segments with low amount of activity have long length) [9]. Given a time-series $\mathbf{x} = [x_1, \ldots, x_n]$, the APCA representation is given as:

$$\bar{X} = \{(\bar{x}_1, xr_1), \ldots, (\bar{x}_w, xr_w)\} \tag{7}$$

where $\bar{x}_i$ is the mean value of i^{th} segment and xr_i is the right end point [9]. However, PAA and APCA cannot preserve the shape of each segment (i.e. segments with different shape could have the same mean values).
Another form of representation for time-series data is using symbolic representations. For instance, Symbolic Aggregate approXimation (SAX) uses PAA to reduce the length of data and after that, with the Gaussian distribution assumption, it represents the PAA vector as a sequence of symbols [10]. SAX divides the area under a Gaussian curve into equiprobable areas and maps each one to a symbol [10]. These symbols are used to represent each fragment of the data.
However, as SAX uses PAA to reduce the dimensions, it has the short-come of information loss. Extended Symbolic Aggregate Approximation (ESAX) tried to overcome the issues in SAX with adding more information in representing each segment [11]. In ESAX, each segment besides mean and the symbolic representation has two more added attributes; the great value and the minimum value of the segment.
Blocks of Eigenvalues Algorithm for Time-series Segmentation (BEATS) is another method which divides time-series into 8×8 observations blocks and then transforms each block into a matrix and after that by applying Discrete Cosine Transform (DCT) and eigenvalues, it represents the time-series with an eigenvector of the quantised DCT matrix [13].

Principal Component Analysis (PCA) is a well-known method for analysing multivariate time-series which represent data in a lower dimensional space. In practice, PCA performs SVD (i.e. $\mathbf{X} = \mathbf{U} \times \mathbf{\Lambda} \times \mathbf{U}^T$) where matrix $\mathbf{U}$ contains the principal components and matrix $\mathbf{\Lambda}$ contains the corresponding variances which have been maximised during the PCA procedure [12]. PCA has been used as a representation method after segmentation which can help the clustering and analysing of data as it reduces the dimension and volume of data while preserving its critical information [15].

In the next section, we describe our algorithm for representing and pre-processing multivariate time-series data using Lagrangian Multipliers and applying PCA. This creates a flexible and scalable method which is also suitable for dynamic real-world data streams.

III. Methodology

A multivariate time-series can be shown as a matrix $N \times D$ where N is the number of observations over time and D is the number of variables (i.e. dimensions). We first apply the Lagrangian Multiplier to scale each dimension between $(-1, +1)$. We then apply PCA over blocks of s observations[1] to extract the most common pattern. These steps are explained in the following sections.

A. Aggregating using Lagrangian Multiplier

The Lagrangian Multiplier is a method to maximise or minimise a function in relation to equality constraints. In our proposed method, we use this transformation to scale each dimension of multivariate time-series into $(-1, +1)$. In other words, we seek to find a vector for each dimension which maximises the dot product of the vector and the dimension subject to the vector being a unity.

For the Lagrangian method, we define a function $L(\mathbf{x}, \lambda) = E(\mathbf{x}) - \lambda g(\mathbf{x})$ where $g(\mathbf{x}) = 0$ is the equality constrain and $E(\mathbf{x})$ is the function we want to find its extrema (i.e. maxima or minima) point that satisfies the constraint. As a result, the extrema has to satisfy the equations $\frac{\mathrm{d}L}{\mathrm{d}\lambda} = 0$ and $\frac{\mathrm{d}L}{\mathrm{d}\mathbf{x}} = 0$. We represent the multivariate time-series as a matrix $\mathbf{M} = [\mathbf{m}_1, \ldots, \mathbf{m}_D]$ where $\mathbf{m}_i$ is the i^{th} variable of the time-series. $E(\mathbf{x})$ is the dot product of $\mathbf{m}_i$ and the unity vector $\vec{\mathbf{x}} = [x_1, \ldots, x_N]$ which we want to maximise it by finding the right unity vector. $g(\mathbf{x})$ is the constrain related to the $\vec{\mathbf{x}}$ being unity (i.e. $\|\vec{\mathbf{x}}\| = 1$) so $g(\mathbf{x}) = \sum_{j=1}^{N} x_j^2 - 1 = 0$.

The Lagrangian Multiplier function is shown below:

$$L(\mathbf{x}, \lambda) = \vec{\mathbf{x}} \cdot \mathbf{m}_i - \lambda g(\mathbf{x})$$
$$L(\mathbf{x}, \lambda) = \sum_{j=1}^{N} x_i m_{j,i} - \lambda (\sum_{j=1}^{N} x_j^2 - 1) \tag{8}$$

where for solving the equation $L(\mathbf{x}, \lambda) = 0$:

$$\begin{cases} \frac{\partial L}{\partial x_j} = m_{j,i} - \lambda 2 x_j = 0 & \text{for } \; j = 1, \ldots, N \\ \\ \frac{\partial L}{\partial \lambda} = -(\sum_{j=1}^{N} x_j^2 - 1) = 0 \end{cases} \tag{9}$$

and therefore:

$$x_j = \frac{1}{2\lambda} m_{j,i} \quad \text{for } \; j = 1, \ldots, N \tag{10}$$

which shows that the $\vec{\mathbf{x}}$ is having a constant ratio to $\mathbf{m}_i$. As a result:

$$\vec{\mathbf{x}} = \frac{\mathbf{m}_i}{\|\mathbf{m}_i\|} \tag{11}$$

meaning that for each dimension we have a unit vector.

B. PCA for Length Reduction

We divide the matrix $\mathbf{M}_{N \times D}$ row-wise with step size s. In other words, $\mathbf{M}$ will be divided into $\mathbf{M}_1, \ldots, \mathbf{M}_w$ where w is the number of matrices after the dividing procedure (i.e. $w = \frac{N}{s}$). We then perform PCA over each $\mathbf{M}_i$ and take the highest principal component as the representative of $\mathbf{M}_i$. For PCA, we perform Singular Value Decomposition (SVD) for each matrix M_i:

$$\underset{s \times D}{\mathbf{M}_i} = \underset{s \times L}{\mathbf{U}} \times \underset{L \times L}{\mathbf{\Lambda}} \times \underset{L \times D}{\mathbf{V}^T} \tag{12}$$

where L is the number of largest selected singular values. To reduce the length, we take the first column of $\mathbf{V}$ which includes the highest principal component as the representation of $\mathbf{M}_i$ and it is corresponding to the highest variance.

The next section of this paper describes the evaluation and performance of the proposed representation method using a data analysis technique which is clustering[2]. Clustering is one of the solutions to uncover patterns which helps users to explore the data and understand the underlying structure to extract meaningful information and detect anomalies [4]. For clustering, we use the Multivariate Gaussian Mixture Models (GMM) which is a model-based clustering method and has been used in existing works [4].

IV. Evaluation

To evaluate our proposed method, we applied it to two different data sets: a synthetic data set and a real-world air-pollution data set. We used Multivariate Gaussian Mixture Models (GMM) for clustering to capture the capability of our representation method.

[1]The parameter s will be selected according to the sampling frequency and the desired aggregation criteria

[2]Clustering is a technique which gathers similar data into one group without prior knowledge of the groups.

A. GMM for Clustering

Multivariate Gaussian Mixture Models (GMM) will fit K Multivariate Gaussian Mixture Models to data. The method has two phases; one is the training phase and the other is the mapping phase. In the training phase, the model learns the GMM parameters and in the mapping phase it maps each data to one of the clusters.

We can define Multivariate GMM as a mixture of K Multivariate Gaussian distributions (i.e. it is a weighted average of K Gaussian distributions):

$$f(\mathbf{m}_i|m, K, \theta) = \sum_{k=1}^{K} p_k \phi(\mathbf{m}_i|a_k) \tag{13}$$

where $\mathbf{m}_i$ is a vector of D dimensions, p_k is mixing proportion, $\phi(\mathbf{m}_i, a_k)$ is the Gaussian density of $\mathbf{m}_i$ with parameter $a_k = (\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ where $\boldsymbol{\mu}_k$ is the mean vector of k^{th} mixture model and $\boldsymbol{\Sigma}_k$ is the variance matrix of k^{th} mixture model and $\theta = (p_1, \ldots, p_k, a_1, \ldots, a_k)$ is the set of parameters for the GMM. At the training phase, an Expectation-Maximisation (EM) algorithm is used to learn the GMM parameters θ.

In the Expectation-step (E-step), we compute $w_{i,k}$ for each $\mathbf{m}_i$ and each mixture model:

$$w_{i,k} = \frac{p_k \phi(\mathbf{m}_i|a_k)}{\sum_{j=1}^{K} p_j \phi(\mathbf{m}_i|a_j)} \tag{14}$$

which is the conditional probability that k^{th} mixture model contains $\mathbf{m}_i$. In the Maximisation-step (M-step), we use $w_{i,k}$ and the data items to update the GMM parameters.

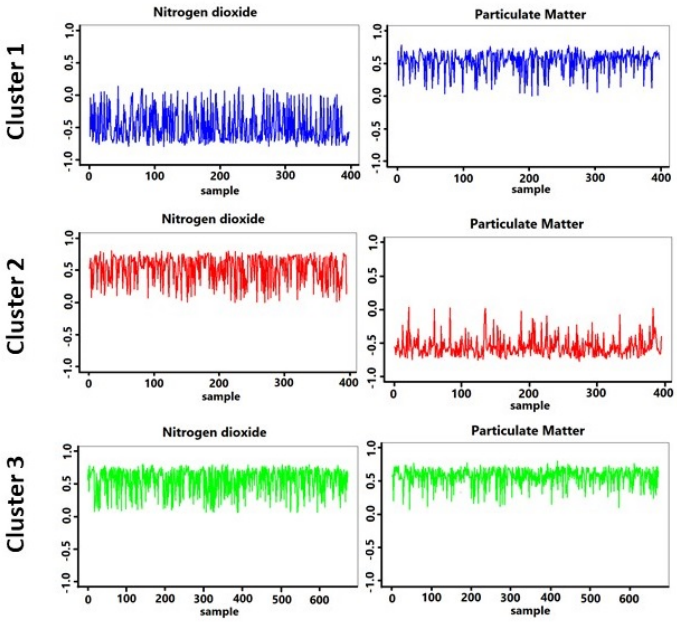
$$\begin{aligned} p_k^{new} &= \frac{N_k = \sum_{i=1}^{N} w_{i,k}}{N} \\ \mu_k^{new} &= (\frac{1}{N_k}) \sum_{i=1}^{N} w_{i,k} \cdot \mathbf{m}_i \\ \boldsymbol{\Sigma}_k^{new} &= (\frac{1}{N_k}) \sum_{i=1}^{N} w_{i,k} \cdot (\mathbf{m}_i - \boldsymbol{\mu}_k^{new})(\mathbf{m}_i - \boldsymbol{\mu}_k^{new})^t \end{aligned} \tag{15}$$

At the mapping phase, we compute the probability of each data point being in each cluster (i.e. mixture model) and map them to the cluster with the maximum probability value.

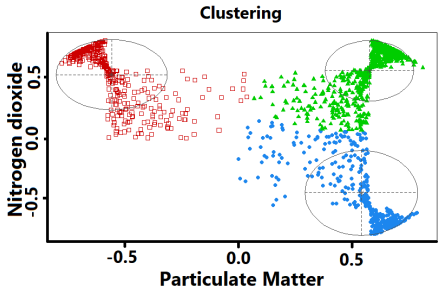
B. Using Synthetic Data

To generate synthetic data sets, we used a multivariate Gaussian distribution and constructed a time-series with 2400 samples over time and four-dimensions which have three different Gaussian distributions with equal covariance matrix and three different mean vectors. Each Gaussian distribution contains 800 samples. We also added white Gaussian noise with signal to noise ratio (SNR) of 0.01.

We applied our method to the data with noise with step size (s) equal to 12 and evaluated the performance via the Silhouette Coefficient and for the data without noise we did not apply the PCA step. This decision was based on the fact that the data was generated with a multivariate Gaussian distribution and applying PCA would have distorted the GMM results.



(a) Notice the different patterns of observation from each cluster



(b) The output of clustering algorithm after applying Lag and PCA on real-time air pollution data

Fig. 2: The clustering result

TABLE I: Performance of the propose method averaged over 50 sets of synthetic data

Data	Silhouette Coefficient
Without Noise	0.87
With Noise	0.47

The Silhouette Coefficient assesses the similarity measure of an observation within its own cluster compared to other clusters. The Silhouette Coefficient ranges between -1 and $+1$; the higher the coefficient the better the clustering performance. The calculated Silhouette Coefficients for the synthetic data is presented in Table I. For rigour, we have generated 50 different sets of synthetic data and the final coefficients were averaged over all the generated data sets. As shown, the Silhouette Coefficient for the noise-free synthetic data is significantly higher than the one for noisy data with SNR = 0.01. As expected, higher noise affected the performance of the proposed method.

TABLE II: Clustering results for real-world Data

Method	Silhouette Coefficient	Ratio
Lag + PCA + GMM	0.69	4.09
Raw + GMM	0.46	2.25
Lag + GMM	0.457	2.25
PCA + GMM	0.395	2.05

C. Using Real-world Data

To illustrate the performance of our proposed method for real-world applications, we selected air quality data from the CityPulse project's open data set[3]. We used the air quality observations data for a period of two months which were recorded every five minutes (i.e. 12 samples per hour). The data has two dimensions; Nitrogen-dioxide (NO_2) and Particulate Matter (PM).

We set the step size as 12 ($s = 12$) which contains observations for an hour. We cluster the data in three different clusters based on the air-quality index for air-pollution assessments (i.e. low risk, medium risk and high risk). See Figure 2 for the clustering result. To evaluate our proposed method, we compared it with existing solutions. We applied the GMM clustering to the raw data, to the data after applying only the Lagrangian Multiplier and also to the data after applying only PCA.

To provide numerical assessment, we calculated Silhouette Coefficient and also the ratio of average distance between clusters to average distance within clusters[4]. Note that we have used the ratio as the Lagrangian transformation scale the data and this affects the distance measures for different scenarios. Therefore, to be able to provide a fair and consistent comparison, we calculate the ratio.

The results are shown in Table II. The Silhouette Coefficient for our proposed method is 0.69 which shows higher performance compared with other methods. The ratio of average distance between clusters to average distance within clusters is higher in our proposed method which means the samples are closer within each cluster and they are well-separated from other clusters. Figure 3 illustrates how the proposed model is embedded into an edge computing architecture for an IoT system.

V. Conclusion

We introduced a new method for extracting patterns from multivariate IoT data streams. The method uses Lagrangian Multiplier to scale the data and then it uses Principal Component Analysis (PCA) to reduce the length of the data (i.e. extract the most useful features from the data). To evaluate the performance and efficiency of the method, we use the existing Multivariate Gaussian Mixture Models (GMM) to cluster the data sets. The method was assessed using both synthetic and real-world data sets and has shown that it outperforms the state-of-the-art methods by up to 20%.

[3]http://iot.ee.surrey.ac.uk:8080/datasets.html

[4]The higher the distance between and smaller the distance within clusters the better the clustering performance, so ratio of a high performance clustering should be high [16]

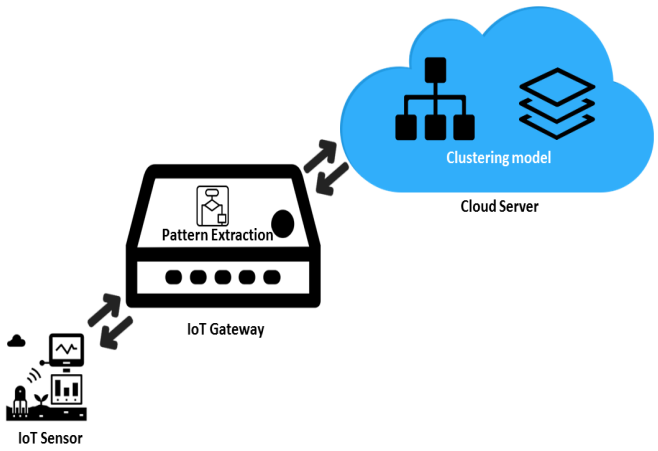


Fig. 3: The edge based architecture for the proposed method

The proposed method utilises techniques that can be run and integrated into an edge computing device. This allows to perform the pattern extraction in the edge layer of an IoT system and transfer the reduced sized data and patterns to the Cloud for further analysis. The outputs of the clustering step can be used to identify the structure of data, analyse existing patterns and detect anomalies. In our proposed method, we have used a fixed number of observations as step size. Our future work will focus on providing adaptive and dynamic selection of the step size. We also plan to apply our method on data sets with higher dimensions.

Acknowledgment

This work is supported by the European Commissions Horizon 2020 (EU H2020) for the IoTCrawler project (http://iotcrawler.eu/) under contract number: 779852.

References

[1] S. Enshaeifar, P. Barnaghi, S. Skillman, A. Markides, T. Elsaleh, S. T. Acton, R. Nilforooshan, and H. Rostill, "The internet of things for dementia care," *IEEE Inter. Comp.*, vol. 22, no. 1, pp. 8–17, 2018.

[2] H. Li, K. Ota, and M. Dong, "Learning iot in edge: deep learning for the internet of things with edge computing," *IEEE Network*, vol. 32, no. 1, pp. 96–101, 2018.

[3] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computinga key technology towards 5g," *ETSI white paper*, vol. 11, no. 11, pp. 1–16, 2015.

[4] S. Aghabozorgi, A. S. Shirkhorshidi, and T. Y. Wah, "Time-series clustering–a decade review," *Information Sys.*, vol. 53, pp. 16–38, 2015.

[5] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos, *Fast subsequence matching in time-series databases*. ACM, 1994, vol. 23, no. 2.

[6] K.-P. Chan and W.-C. Fu, "Efficient time series matching by wavelets," in *icde*. IEEE, 1999, p. 126.

[7] D. Wu, A. Singh, D. Agrawal, A. El Abbadi, and T. R. Smith, "Efficient retrieval for browsing large image databases," in *Proceedings of the fifth international conference on Information and knowledge management*. ACM, 1996, pp. 11–18.

[8] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra, "Dimensionality reduction for fast similarity search in large time series databases," *Knowledge and information Systems*, vol. 3, no. 3, pp. 263–286, 2001.

[9] ——, "Locally adaptive dimensionality reduction for indexing large time series databases," *ACM Sigmod Record*, vol. 30, no. 2, pp. 151–162, 2001.

[10] J. Lin, E. Keogh, S. Lonardi, and B. Chiu, "A symbolic representation of time series, with implications for streaming algorithms," in *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*. ACM, 2003, pp. 2–11.

[11] Y. Jiang, T. Lan, and D. Zhang, "A new representation and similarity measure of time series on data mining," in *2009 International Conference on Computational Intelligence and Software Engineering*. IEEE, 2009, pp. 1–5.
[12] H. Yoon, K. Yang, and C. Shahabi, "Feature subset selection and feature ranking for multivariate time series," *IEEE transactions on knowledge and data engineering*, vol. 17, no. 9, pp. 1186–1198, 2005.
[13] A. Gonzalez-Vidal, P. Barnaghi, and A. F. Skarmeta, "Beats: Blocks of eigenvalues algorithm for time series segmentation," *IEEE Transactions on Knowledge and Data Engineering*, 2018.
[14] Z. R. Struzik and A. Siebes, "The haar wavelet transform in the time series similarity paradigm," in *European Conference on Principles of Data Mining and Knowledge Discovery*. Springer, 1999, pp. 12–22.
[15] J. Abonyi, B. Feil, S. Nemeth, and P. Avra, "Principal component analysis based time series segmentation: A new sensor fusion algorithm," *preprint*, 2004.
[16] D. S. Wilks, "Cluster analysis," in *International geophysics*. Elsevier, 2011, vol. 100, pp. 603–616.