Parallel Computing TEDA for High Frequency Streaming Data Clustering

Xiaowei Gu[1], Plamen P. Angelov[1], German Gutierrez[2], Jose Antonio Iglesias[2], and Araceli Sanchis[2]

[1] Data Science Group, School of Computing and Communications, Lancaster University, LA1 4WA, UK
{x.gu3,p.angelov}@lancaster.ac.uk
[2] CAOS Research Group, Computer Science Department, Carlos III University of Madrid, 28918 Leganes, Spain
{ggutierr,jiglesia,msam}@inf.uc3m.es

Abstract. In this paper, a novel online clustering approach called *Parallel_TEDA* is introduced for processing high frequency streaming data. This newly proposed approach is developed within the recently introduced TEDA theory and inherits all advantages from it. In the proposed approach, a number of data stream processors are involved, which collaborate with each other efficiently to achieve parallel computation as well as a much higher processing speed. A fusion center is involved to gather the key information from the processors which work on chunks of the whole data stream and generate the overall output. The quality of the generated clusters is being monitored within the data processors all the time and stale clusters are being removed to ensure the correctness and timeliness of the overall clustering results. This, in turn, gives the proposed approach a stronger ability of handling shifts/drifts that may take place in live data streams. The numerical experiments performed with the proposed new approach *Parallel_TEDA* on benchmark datasets present higher performance and faster processing speed when compared with the alternative well-known approaches. The processing speed has been demonstrated to fall exponentially with more data processors involved. This new online clustering approach is very suitable and promising for real-time high frequency streaming processing and data analytics.

Keywords: High frequency streaming data, TEDA, Parallel computation, Clustering, Real time

1 Introduction

The amount and scale of the data streams grow rapidly because of the more mature technologies and lower prices of various electronic devices as well as wider distributed sensor networks. As the world has already entered the Era of Big Data, data-intensive technologies have been extensively used in the developed economies and numerous international organizations and companies are now much more frequently getting involved in Internet-based activities. All of these have largely increased the demand for developing advanced intensive data processing/analysis methodologies and technologies.

Traditional clustering methods more often share several common problems [1–8], for example, they require the number of clusters [2, 3] or thresholds [4–6] or radii [7]. Most of the existing methods are offline [9], require iterative calculations. The minority of online, incremental, one pass approaches [10] still require user-defined parameters like radii, thresholds which makes them not fully autonomous and unsupervised. Very recently, completely autonomous approaches were proposed [11, 12], however, they do not consider the big data, high frequency, parallel computation aspects. In this paper, we propose *Parallel_TEDA* approach which is addressing precisely these aspects. It is memory- and computation-efficient. Indeed, a single data processor might also have difficulties in handling data samples which arrive in a very high speed.

Some published algorithms [13–16] try to improve computation efficiency by dividing the streaming data chunk by chunk and processing them separately. Then the algorithms fuse the individual clustering results together. However, such algorithms [13–16] lack the ability to detect the evolutions and transitions of the clusters, which sometimes lead to the fusion of non-related clusters [17].

TEDA (Typicality and Eccentricity Data Analysis) was introduced in [18–20] as a fully data-driven and prior-assumptions-free framework for data analysis. Moreover, TEDA supports the recursive form of calculation and is parameter-free. Therefore, TEDA is a suitable algorithm for real-time streaming data processing.

In this paper, we propose a novel online clustering approach for big data streams called *Parallel_TEDA*, which is developed on the basis of TEDA [18–20]. As a result, the proposed approach inherits all its advantages, including no requirement of user input or assumptions, regardless the shape of clusters, recursive computation, self-adaptation to the data pattern. Meanwhile, the idea of parallel processing [13–16] is involved to make it suitable for high frequency data streams processing. Within the *Parallel_TEDA* framework, a number of data streams processors and a fusion center are involved to collaborate with each other to process the high frequency data stream. As a result, the processing speed is largely improved because of its parallel computation structure. Within the proposed approach, the high frequency data stream is split into data chunks, which are passed to different processors. Once the data samples are being processed by the processors, they are discarded to improve the memory-efficiency and only the key information is stored. The fusion center automatically gathers all the key information from the individual processors and fuses together into the final clustering results. Because of the computationally lean recursive expression, the calculations are very fast and no iterations or search is involved which makes possible for real time processing of high frequency data streams. Moreover, compared with the traditional parallel processing techniques [13–16], the proposed *Parallel_TEDA* is able to follow the ever changing data pattern and successfully avoids the problem of meaningless fusion because of advantages of TEDA.

Additionally, we use the time tag, age [21] to monitor the quality of the existing clusters stored in the data stream processors and enhance its ability to self-evolve. Thus, *Parallel_TEDA* is able to follow the changes of the data pattern and avoid the accumulation of error. Because of the time tag, when the *Parallel_TEDA* approach is dealing with multi-data streams, there is no need to wait for processing the data streams sequentially one by one. Once, there is any free processor, the next data stream can be processed

seamlessly. In addition, several data streams can be processed together at the same time because of the multi-processors structure.

The remainder of this paper is organized as follows: Section II introduces the theoretical basis of the TEDA framework. The details of the proposed *Parallel_TEDA* approach are described in section III. In section IV, the algorithm of the proposed *Parallel_TEDA* approach is presented. Section V is for numerical experiments as well as analysis and the conclusions are presented in section VI.

2 Theoretical basis of the TEDA framework

Typicality and Eccentricity based Data Analytics (TEDA) was introduced recently [18–20] as a non-parametric, assumption-free methodology for extracting information from data. In its nature, it is close to statistical learning; however, traditional statistical methods assume random variables and try to approximate these *prior* assumptions with the experimental data. TEDA is free from the range of assumptions made in traditional probability theory and statistical learning and is entirely data-driven. No assumption is made for data (in)dependence, number, nature (determinative or stochastic). The main quantities (measures) of mutual (ensemble) properties of the data, namely, typicality can be seen as similar to probability and membership function in fuzzy sets, but it is objective and assumptions and parameters free.

2.1 Cumulative Proximity

Cumulative proximity [18, 19, 22] is the sum of distances from a particular data sample to all the available data samples. The cumulated proximity of $\boldsymbol{x}_i$ to all other data samples is expressed as:

$$\pi_k(\boldsymbol{x}_i) = \sum_{l=1}^{k} d^2(\boldsymbol{x}_i, \boldsymbol{x}_l);\ \ i = 1, 2, \ldots, k;\ \ k > 1 \tag{1}$$

where $d(\boldsymbol{x}_i, \boldsymbol{x}_l)$ denotes any distance measure between $\boldsymbol{x}_i$ and $\boldsymbol{x}_l$. For clarity, in this paper, we use the most commonly used Euclidean type of distance, namely $d(\boldsymbol{x}_i, \boldsymbol{x}_l) = \|\boldsymbol{x}_i - \boldsymbol{x}_l\|$.

2.2 Standardized Eccentricity

Eccentricity and its standardised form [18–20] are derived by normalising the cumulative proximity and are very useful for anomaly detection, image processing, fault detection, etc.

The eccentricity of $\boldsymbol{x}_i$ is calculated as [18, 19]

$$\xi_k(\boldsymbol{x}_i) = \frac{2\pi_k(\boldsymbol{x}_i)}{\sum_{l=1}^{k} \pi_k^i(\boldsymbol{x}_l)};\ i = 1, 2, \ldots, k;\ k > 1 \tag{2}$$

Standardized eccentricity is derived directly from the eccentricity [18, 19].

$$\varepsilon_k(\boldsymbol{x}_i) = k\xi_k^i(\boldsymbol{x}_i) = \frac{2k\pi_k(\boldsymbol{x}_i)}{\sum_{l=1}^{k}\pi_k(\boldsymbol{x}_l)};\ i = 1, 2, \ldots, k;\ k > 1 \tag{3}$$

It is obvious that the sums of the eccentricity and standardized eccentricity are 2 and $2k$, respectively. Compared with eccentricity, standardized eccentricity is a more convenient measure because when k tends to be infinity, its mean value does not drop fast, namely more stable than eccentricity. Using it, the well-known *Chebyshev inequality* [24] obtains a very elegant form.

2.3 Typicality

Typicality is the main measure introduced within TEDA [18–20] . It can be considered as a form of centrality or the normalized inverse cumulative proximity [20]. The typicality of $\boldsymbol{x}_i$ $(j = 1, 2, \ldots, k;\ k > 1)$ is [20]:

$$\tau_k(x_i) = \frac{\frac{1}{\pi_k(\boldsymbol{x}_i)}}{\sum_{l=1}^{k}\frac{1}{\pi_k(\boldsymbol{x}_l)}}) = [\pi_k(\boldsymbol{x}_i)[\sum_{l=1}^{k}(\pi_k(\boldsymbol{x}_l))^{-1}]]^{-1} \tag{4}$$

Typicality represents spatial distribution pattern of the data and resembles the probability distribution function [19].

2.4 Recursive Form of Calculation

It is very important in the context of working with high frequency live data streams to have one pass type of algorithms and recursive calculations. In this case, all quantities considered within the TEDA framework, namely, the cumulative proximity, standardized eccentricity and typicality can all be updated recursively. With Euclidean distance, the cumulated proximity of new coming data sample $\boldsymbol{x}_{k+1}$ at the $(k+1)^{th}$ time instance is recursively calculated as follows [18, 22]:

$$\pi_{k+1}(\boldsymbol{x}_{k+1}) = (k+1)(\left\|\boldsymbol{x}_{k+1} - \boldsymbol{\mu}_{k+1}\right\|^2 + X_{k+1} - \left\|\boldsymbol{\mu}_{k+1}\right\|^2)$$

$$\text{where } \boldsymbol{\mu}_{k+1} = \tfrac{k}{k+1}\boldsymbol{\mu}_k + \tfrac{1}{k+1}\boldsymbol{x}_{k+1}; \boldsymbol{\mu}_1 = \boldsymbol{x}_1 \tag{5}$$

$$X_{k+1} = \tfrac{k}{k+1}X_k + \tfrac{1}{k+1}\left\|\boldsymbol{x}_{k+1}\right\|^2;\ X_1 = \left\|\boldsymbol{x}_1\right\|^2$$

The corresponding sum of cumulated proximity can be updated as follows [20]:

$$\sum_{j=1}^{k+1}\pi_{k+1}(\boldsymbol{x}_j) = \sum_{j=1}^{k+1}\sum_{l=1}^{k+1}(\boldsymbol{x}_j - \boldsymbol{x}_l)^T(\boldsymbol{x}_j - \boldsymbol{x}_l) = 2(k+1)^2(X_{k+1} - \left\|\boldsymbol{\mu}_{k+1}\right\|^2) \tag{6}$$

Using equations (6) and (5), the recursive form of the standardized eccentricity can be obtained as:

$$\varepsilon_{k+1}(\boldsymbol{x}_{k+1}) = \frac{\left\|\boldsymbol{x}_{k+1} - \boldsymbol{\mu}_{k+1}\right\| + X_{k+1} - \left\|\boldsymbol{\mu}_{k+1}\right\|^2}{X_{k+1} - \left\|\boldsymbol{\mu}_{k+1}\right\|^2} = 1 + \frac{\left\|\boldsymbol{x}_{k+1} - \boldsymbol{\mu}_{k+1}\right\|}{X_{k+1} - \left\|\boldsymbol{\mu}_{k+1}\right\|^2} \tag{7}$$

The recursive update of the typicality $\tau_k(\boldsymbol{x}_j)$ can obviously be achieved using equation (5).

Before and independently of TEDA, the clustering quality and monitoring parameter, *age*, was introduced in 2005 [21]

2.5 Cluster Age

Since the proposed approach is for live streaming data, the data pattern of the stream will potentially change along with time elapse, as a result, the old existing clusters may not be able to represent well the ensemble properties of the data samples following a possible shift or drift [22, 23]. Cluster age [22] allows to decide whether a cluster is outdated. Cluster age [21] is an accumulated relative time tag which allows to decide whether a cluster is outdated and is expressed as follows:

$$A_k^c = k - \frac{\sum_{l=1}^{S_k^c} I_l^c}{S_k^c};\ c = 1, 2, ...C;\ S_k^c \geq 1; k > 1 \tag{8}$$

where the sub-label c denotes the c^{th} cluster and C is the number of existing clusters in the clustering result; S_k^c is the support of the c^{th} cluster at the time instance k (number of data samples associated with it).

2.6 Chebyshev Inequality

The well-known *Chebyshev inequality* [24] describes the probability of the distance between a certain data sample $\boldsymbol{x}_i$ and the mean value to be larger than n times the standard deviation, namely $n\sigma$. For Euclidean type of distance it has the following form:

$$P(\|\boldsymbol{x}_i - \boldsymbol{\mu}_k\| \leq n^2\sigma^2) \geq 1 - \frac{1}{n^2} \tag{9}$$

In TEDA, this condition is expressed regarding standardized eccentricity and in a more elegant form [19]:

$$P(\varepsilon_k(\boldsymbol{x}_i) \leq 1 + n^2) \geq 1 - \frac{1}{n^2} \tag{10}$$

That means, we can directly check the value of the standardized eccentricity, ε_k and see if it is less than 10 for the "3σ" case.

3 The proposed approach Parallel_TEDA

First of all, let us denote the data stream in the Hilbert space R^d as$\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k, \ldots\}$, where the index k indicates the time instance at which the k^{th} data sample arriving. The continuously arriving data samples are divided into chunks and sent to different data stream processors (assuming there are processors and the chunk size is K) according to the time instance of arrival, namely:

$$\mathbf{X} = \begin{bmatrix} \underbrace{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots \boldsymbol{x}_K}_{chunk1 \rightarrow processor1} \\ \underbrace{\boldsymbol{x}_{K+1}, \boldsymbol{x}_{K+2}, \ldots \boldsymbol{x}_{2K}}_{chunk2 \rightarrow processor2} \\ \vdots \\ \underbrace{\boldsymbol{x}_{(N-1)K+1}, \boldsymbol{x}_{(N-1)K+2}, \ldots \boldsymbol{x}_{NK}}_{chunkN \rightarrow processorN} \\ \underbrace{\boldsymbol{x}_{NK+1}, \boldsymbol{x}_{NK+2}, \ldots \boldsymbol{x}_{(N+1)K}}_{chunk(N+1) \rightarrow processor1} \\ \underbrace{\boldsymbol{x}_{(N+1)K+1}, \boldsymbol{x}_{(N+1)K+2}, \ldots \boldsymbol{x}_{(N+2)K}}_{chunk(N+2) \rightarrow processor2} \\ \vdots \end{bmatrix} \tag{11}$$

For each data chunk, once it is sent to the data stream processor, it will be processed separately and the current processing result will only be influenced by the previous data chunks processed within the same data stream processor and will also influence the future output of the processor. Therefore, for the remainder of this section, all the basic elements of *Parallel_TEDA* approach introduced are considered to be within the same particular data stream processor.

The collection of data samples entered the i^{th} data processor are denoted as $\mathbf{X}^i = \{\boldsymbol{x}_1^i, \boldsymbol{x}_2^i, \ldots, \boldsymbol{x}_K^i\}$ $(i = 1, 2, \ldots, N)$, and the number of samples, which can also be viewed as time instance, is k and will keep growing with time. The number of data samples processed by each processor is considered to be the same.

Parallel_TEDA approach is divided into two stages. The first stage is the clusters and parameters updating within the individual data stream separately. We consider this stage to be a separate processing. The second stage is for merging the separate clustering results of all existing data streams together, called clusters fusion. The architecture of the proposed approach is presented in Fig. 1.

In the remainder of this section, the proposed approach will be described in detail.

3.1 Separate Processing Stage

In this stage, all the data chunks separated from the data stream are processed separately in their corresponding processors. For the i^{th} data stream processor, at each time instance a new data sample arrives, denoted as $\boldsymbol{x}_{k+1}^i$, their standardized eccentricity per cluster is calculated using equation (7). Then, we monitor $\varepsilon_{k+1}^{i,c}(\boldsymbol{x}_{k+1}^i)$; $c = 1, 2, \ldots, C^i$, here C^i denotes the number of existing clusters in the i^{th} processors.

For every cluster , the standardized eccentricities of all its members sum to $2S_k^{i,c}$, and the average standardized eccentricity is $\varepsilon_{average} = 2$. Combining the Chebyshev conditions in terms of the standardised eccentricity (equation (10)) [19], we use $n = 2$ to achieve a balance between sensitivity to anomalous data and tolerance to the intra-cluster variance, namely, $\varepsilon_o = 5$. The condition for associating a data sample with a particular cluster can be expressed as follows:

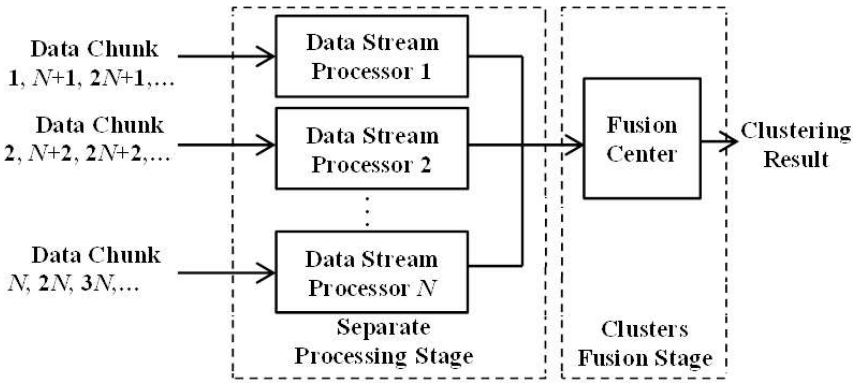


Fig. 1: The architecture of *Parallel_TEDA*

$$\text{IF } (\varepsilon^{i,c}_{k+1}(\boldsymbol{x}^i_{k+1}) \leq \varepsilon_o) \quad \text{THEN } (\boldsymbol{x}^i_{k+1} \in c) \tag{12}$$

For each new data sample $\boldsymbol{x}^i_{k+1}$, if the condition to a particular existing cluster is met, it is associated with this cluster. If χ^i_{k+1} is associated with two or more clusters, it is associated with the cluster that satisfies the following condition:

$$c_{selected} = argmin^{C_i}_{c=1}\varepsilon^{i,c}_{k+1}(\boldsymbol{x}^i_{k+1}) \tag{13}$$

Once, the cluster that $\boldsymbol{x}^i_{k+1}$ should be assigned to is decided, the corresponding mean (center) $\boldsymbol{\mu}^{i,c_{selected}}_{k+1}$, the mean of the scalar product $X^{i,c_{selected}}_{k+1}$, age $A^{i,c_{selected}}_{k+1}$ and support $S^{i,c_{selected}}_{k+1}$ of the cluster need to be updated accordingly:

$$\begin{aligned}
\boldsymbol{\mu}^{i,c_{selected}}_{k+1} &\leftarrow \left(\frac{S^{i,c_{selected}}_k}{S^{i,c_{selected}}_k+1}\boldsymbol{\mu}^{i,c_{selected}}_k + \frac{1}{S^{i,c_{selected}}_k+1}\boldsymbol{x}^i_{k+1}\right)\\
X^{i,c_{selected}}_{k+1} &\leftarrow \left(\frac{S^{i,c_{selected}}_k}{S^{i,c_{selected}}_k+1}X^{i,c_{selected}}_k + \frac{1}{S^{i,c_{selected}}_k+1}\left\|\boldsymbol{x}^i_{k+1}\right\|^2\right)\\
A^{i,c_{selected}}_{k+1} &\leftarrow \left(k+1-\frac{S^{i,c_{selected}}_k(k-A^{i,c_{selected}}_k)+k+1}{S^{i,c_{selected}}_k+1}\right)\\
S^{i,c_{selected}}_{k+1} &\leftarrow (S^{i,c_{selected}}_k+1)
\end{aligned} \tag{14}$$

In contrast, if there is no cluster that meets the condition (equation (12)) for a particular data sample, $\boldsymbol{x}^i_{k+1}$, the new data sample $\boldsymbol{x}^i_{k+1}$ forms a new cluster. The parameters of the new cluster are then initialised as follows:

$$C_i \leftarrow (C_i+1);\ \boldsymbol{\mu}^{i,C_i}_{k+1} \leftarrow \boldsymbol{x}^i_{k+1};\ \ X^{i,C_i}_{k+1} \leftarrow \left\|\boldsymbol{x}^i_{k+1}\right\|^2;\ \ A^{i,C_i}_{k+1} \leftarrow 0;\ \ S^{i,C_i}_{k+1} \leftarrow 1 \tag{15}$$

For the existing clusters which do not have new member at the $(k+1)^{th}$ time instance, their age should be updated using equation (16) All other parameters do not change.

$$A_{k+1}^{i,C_i} \leftarrow (A_k^{i,C_i}+1), c \in Other \tag{16}$$

After the parameters updating, before the processor handles the next input data sample, every existing cluster needs to be checked whether it is already out of date according to its age using the following ageing condition:

$$\text{IF } (A_{k+1}^{i,C_i} > \mu_A^i + \sigma_A^i) \text{ AND } (A_{k+1}^{i,C_i} > K) \text{ THEN (The } c^{th} \text{ cluster is out of date)} \tag{17}$$

where μ_A^i is the average value of the ages of all clusters within the processor, σ_A^i is standard deviation of all clusters'ages within the i^{th} processors.

Once, the processor detects a stale cluster, the stale cluster is removed automatically because it may have adverse influence on the future clustering results. After the stale clusters cleaning operation, the data stream processor will be ready for the next data sample and begin a new round of data processing and parameters updating. Based on the needs of users, the clustering results can be viewed and checked at any time. Once requested by users, all the processors will send the existing clusters and the corresponding parameters stored in their memories to the fusion center, and the fusion center will fuse all the clustering results together and give the final output. However, this user inference is entirely optional. The algorithm is designed to work fully autonomously and will perform fusion anyway unless specifically prompted not to.

3.2 Clusters Fusion Stage

In this stage, the clustering results from all the data stream processors will be fused together to generate the final output of the proposed *Parallel_TEDA* approach.

Once we get all the clusters from all the processors and re-denote their parameters as: centers $\boldsymbol{\mu}_j$, means of scalar product X_j and supports S_j, $j = 1, 2, \ldots, C_0 (C_0 = \sum_{i=1}^{N} C_i)$, the fusion process will start. The clusters fusion stage begins from the cluster with the smallest support and ends with the largest one. For each cluster, starting from its closest neighboring cluster to the farthest cluster away from it, we check the condition to decide whether this cluster should be merged with the neighboring cluster:

$$\begin{gathered}\text{IF } (\varepsilon_l(\boldsymbol{\mu}_j) \leq \varepsilon_o) \text{ AND } (\varepsilon_j(\boldsymbol{\mu}_l) \leq \varepsilon_o) \\ \text{THEN (The } j^{th} \text{ and } l^{th} \text{ clusters should merge together)}\end{gathered} \tag{18}$$

where $\boldsymbol{\mu}_j$ and $\boldsymbol{\mu}_l$ are the centers of the j^{th} and l^{th} clusters, ε_i and ε_j are calculated using the following equations:

$$\begin{aligned}\varepsilon_l(\boldsymbol{\mu}_j) &= \frac{S_l^2\left\|(\boldsymbol{\mu}_j-\boldsymbol{\mu}_l)\right\|^2+(S_l+1)(S_lX_l+\left\|\boldsymbol{\mu}_j\right\|^2)-\left\|\boldsymbol{\mu}_j+S_l\boldsymbol{\mu}_l\right\|^2}{(S_l+1)(S_lX_l+\left\|\boldsymbol{\mu}_j\right\|^2)-\left\|\boldsymbol{\mu}_j+S_l\boldsymbol{\mu}_l\right\|^2} \\ \varepsilon_j(\boldsymbol{\mu}_l) &= \frac{S_j^2\left\|(\boldsymbol{\mu}_l-\boldsymbol{\mu}_j)\right\|^2+(S_j+1)(S_jX_j+\|\boldsymbol{\mu}_l\|^2)-\left\|\boldsymbol{\mu}_l+S_j\boldsymbol{\mu}_j\right\|^2}{(S_j+1)(S_jX_j+\|\boldsymbol{\mu}_l\|^2)-\left\|\boldsymbol{\mu}_l+S_j\boldsymbol{\mu}_j\right\|^2}\end{aligned} \tag{19}$$

If the merging condition is met, the two clusters merge together:

$$\begin{gathered}\boldsymbol{\mu}_{new} \leftarrow (\frac{S_j}{S_j+S_l}\boldsymbol{\mu}_j + \frac{S_l}{S_j+S_l}\boldsymbol{\mu}_l); \quad X_{new} \leftarrow (\frac{S_j}{S_j+S_l}X_j + \frac{S_l}{S_j+S_l}X_l); \\ S_{new} \leftarrow (S_j + S_l); \quad C_0 \leftarrow (C_0 - 1)\end{gathered} \tag{20}$$

After the merging, there may be some trivial clusters (with small support) left satisfying the following condition:

$$\text{IF } (S_i < \frac{\sum_{j=1}^{C_o} S_j}{5C_o}) \quad \text{THEN (Merge the } i^{th}\text{cluster with the nearest larger cluster)} \tag{21}$$

where the nearest larger cluster is determined as follows:

$$c_{merge} = \text{argmin}_{l=1}^{C_o}(\|\boldsymbol{\mu}_i - \boldsymbol{\mu}_l\|); \;\; l \neq i; \;\; S_l \geq \frac{\sum_{j=1}^{C_o} S_j}{5C_o} \tag{22}$$

Then, equation (20) is used to merge the minor cluster with the larger one.

4 Parallel_TEDA Procedure

In this section, the proposed *Parallel_TEDA* approach is summarized in a form of pseudo-code in two parts according to the structure given in Fig.1.

The first part of this approach is executed in every processor independently, namely the separate processing stage. The second part of the proposed algorithm is executed in the fusion center, namely clusters fusion stage.

1. Parallel_TEDA Algorithm Part 1:

- **While** the data sample of the i^{th} data stream is available (or until nor interrupted):
  - If ($k = 1$) Then:
    $C_i \leftarrow 1; \quad \boldsymbol{\mu}_k^{i,C_i} \leftarrow \boldsymbol{x}_k^i; \quad X_k^{i,C_i} \leftarrow \left\|\boldsymbol{x}_k^i\right\|^2; \quad A_k^{i,C_i} \leftarrow 0; \quad S_k^{i,C_i} \leftarrow 1;$
  - Else:
    1. Calculate $\varepsilon_{k+1}^{i,c}(\boldsymbol{x}_k^i), \;\; c = 1, 2, \ldots, C_i$ using equations (5) and (7);
    2. If (Association Condition is met (equation (12)) Then:
       - Find the target cluster $c_{selected}$ using equations (12) and (13);
       - Update $\boldsymbol{\mu}_{k+1}^{i,c_{selected}}$, $X_{k+1}^{i,c_{selected}}$, $S_{k+1}^{i,c_{selected}}$, $A_{k+1}^{i,c_{selected}}$ using equation (14);
       - Update the ages of other clusters using equation (16);
    3. Else
       - Add a new cluster and its corresponding parameters using equation (15);
       - Update the ages of other clusters using equation (16);
    4. End If
  - End If
- **End While**

2. Parallel_TEDA Algorithm Part 2:

- **While** the existing clusters exhibit the potential of merging
  - Calculate $\varepsilon_l(\boldsymbol{\mu}_j)$ and $\varepsilon_j(\boldsymbol{\mu}_l)$ using equations (19).
  - If (Merging Condition is met) Then:
    - Merge the two clusters using equations (20);
- **End While**
- Merge the minor clusters with the nearest larger clusters using equations (21), (22) and (20)

5 Numerical Experiments

In this section, several numerical experiments using benchmark datasets from [25] are conducted to study the performance of the newly proposed *Parallel_TEDA* approach. The details of the benchmark datasets are given in Table 1.

Table 1: Benchmark Dataset Description

DataSet	Details				
	Number of Data Samples	Number of Clusters	Number of Attributes	Maximum Cluster Size	Minimum Cluster Size
A1	3000	20	2	150	150
A2	5250	35	2	150	150
A3	7500	50	2	150	150
S1	5000	15	2	350	300
S2	5000	15	2	350	300

The *Parallel_TEDA* approach was developed into a software which run within MATLAB R2015a. The performance was evaluated on a PC with processor 3.60 GHz2, and 8 GB RAM. The first experiment is conducted to verify the correctness and effectiveness of the *Parallel_TEDA* approach. Datasets A1 and A2 are used together to testify the ability of the parallel computation as well as handling multi-data streams. In this experiment, 11 data stream processors are involved and the data chunk size is set to be 250. The processing procedure of the data chunks of the three datasets is given in Fig. 2. The time-varying clustering results of each process cycle are presented in Fig. 3.

As it is clearly shown in Fig. 3, the proposed approach can successfully follow the changes of the data pattern including the shift leading to creation of new clusters (evolving the structure). Parallel processors automatically discard the stale clusters which cannot represent adequately the latest ensemble properties of the data. Moreover, it also shows that the proposed approach is capable to seamlessly handle multiple data streams.

The dataset A3 is used to investigate the relationships between execution time and the number of processors as well as the chunk size. The amount of time consumed (in seconds) for processing the dataset A3 are listed in Table 2 with different number of

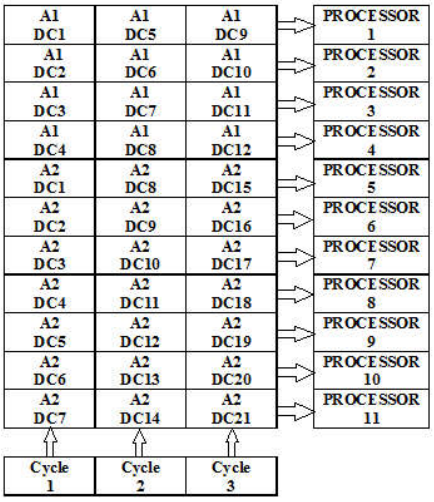


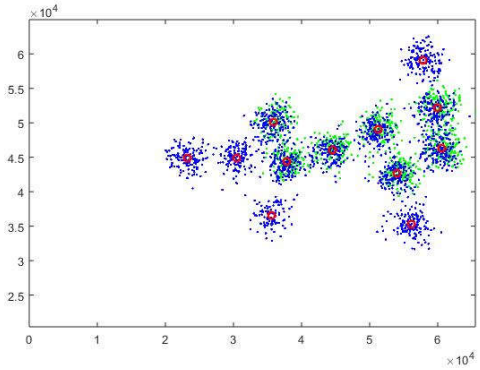
Fig. 2: The processing procedure of first experiment (DC is short for data chunk)

processors and chunk sizes. The amount of the time consumed by the two processing stages are presented separately for easier analysis. The relationship between average processing time and the number of processors is additionally depicted in Fig. 4 for visual clarity.

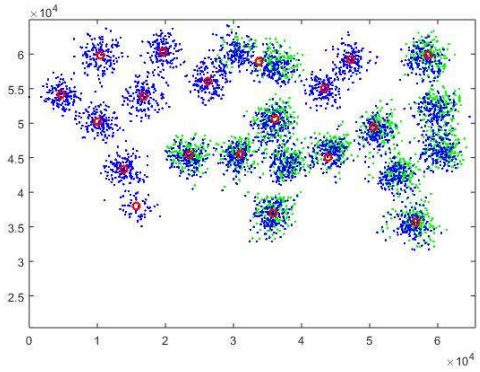
Table 2 and Fig.4 show that, the processing speed becomes higher and the time required, respectively, becomes lower when more processors are involved. It is also interesting to notice that, when more processors are involved in the task, there will be an approximately exponential decrease of the execution time used by the first stage of the proposed approach. However, the fusion center needs time to fuse all the clusters together as there will be more clusters generated by individual processors. Correspondingly, the execution time of stage 2 of the proposed approach will increase with the larger chunk size because the larger chunk size will lead to more clusters generated in the processors in each process cycle, which in turn, also cost more time in processing the future data chunk as well as in fusing clustering results together. Nonetheless, the number of processors has the most significant influence on the overall processing speed. Combining the two stages of the approach, there is still an approximately exponential decrease of the processing time (see Fig. 4)

In order to further study the performance of the proposed approach, the well-known subtractive [6, 7] and ELM clustering approaches [10] were used for comparison purpose.

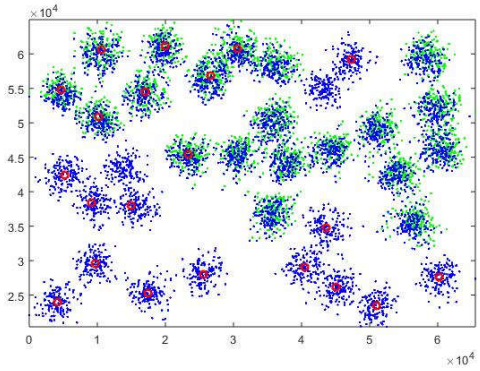
Since *Parallel_TEDA* approach is for live data streams, no processed data samples will be kept in memory, we consider the following alternative measures as indicators of clustering performances: 1) Number of clusters, C_0; 2) Maximum support of the clusters, S_{max}; 3) Minimum support of the clusters, S_{min}; and 4) Execution time, t_{exe}.



(a) Process cycle 1



(b) Process cycle 2



(c) Process cycle 3

Fig. 3: The time-varying clustering result (The green dots are the data samples from A1 dataset, blue dots are the ones from A2 dataset and red circles are centers of the clusters).

Table 2: Execution Time Study (in seconds)

Chunk Size	Stages	Number of Processors						
		2	3	4	5	6	7	8
100	1	0.5746	0.4093	0.3137	0.2517	0.2097	0.1911	0.1715
	2	0.0088	0.0111	0.0105	0.0237	0.0115	0.0289	0.0132
200	1	0.6018	0.4276	0.3141	0.2698	0.2213	0.2018	0.1773
	2	0.0091	0.0118	0.0125	0.0149	0.0135	0.0158	0.0248
300	1	0.6359	0.4468	0.3317	0.2900	0.2308	0.2018	0.1829
	2	0.0100	0.0140	0.0166	0.0127	0.0162	0.0175	0.0215
400	1	0.6764	0.4739	0.3447	0.291	0.2389	0.2159	0.1916
	2	0.0222	0.0148	0.0140	0.0186	0.0166	0.0308	0.0389
500	1	0.7438	0.5316	0.3721	0.3047	0.2602	0.2196	0.1996
	2	0.0114	0.0133	0.0178	0.019	0.0418	0.0523	0.0550

For a better comparison, in the following experiments, *Parallel_TEDA* approach will not discard old clusters, and each time these approaches only handle one data stream. We use 4 processors in the proposed *Parallel_TEDA* approach in comparative experiments and the chunk size is set to 200 data samples/points. The comparison results are presented in Table 3. As it is shown in Table 3, compared with the two comparative clustering approaches, *Parallel_TEDA* approach exhibits more accurate clustering results. Note that, both subtractive and ELM clustering approaches require the radius (r or R, respectively) to be pre-defined. Moreover, its choice heavily influences the result (see Table 3). The proposed *Parallel_TEDA* approach does not require such parameter to be pre-defined. Yet, it is able to identify the correct number of clusters and is the fastest of them. The subtractive clustering approach can be comparable with the proposed approach in terms of clustering accuracy, but its performance heavily relies on the proper user input. In addition, it is offline and iterative. Moreover, even though only 4 processors are involved, *Parallel_TEDA* algorithm is already the fastest among the three. Apparently, the proposed *Parallel_TEDA* approach has clear advantages because of the high performance and the potential for even higher processing speed.

6 Conclusions

In this paper, we proposed a novel real-time clustering approach for high frequency streaming data processing, called *Parallel_TEDA*. This approach inherits the advantages of the recently introduced TEDA theoretical framework and has the ability of parallel computation due to its multi-processors structure. In addition, it can successfully follow the drifts and/or shifts in the data pattern. Within the TEDA framework, the streaming data samples are divided into a number of data chunks and sequentially sent to the data processors for clustering. Each generated cluster is additionally assigned a time tag for online cluster quality monitoring. The fusion center gathers the clustering results from each data processor and fuses them together to obtain the overall output. Numerical experiments show the superior performance of the proposed approach as well as the

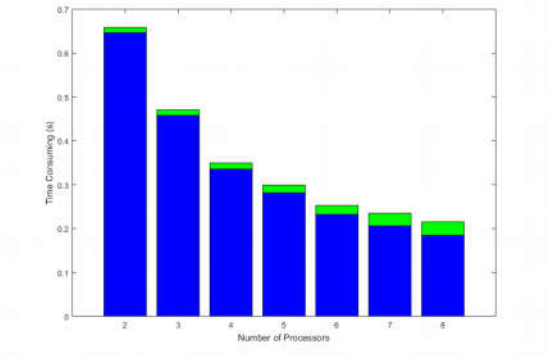


Fig. 4: The relationship between processing time and number of processors (The blue bars represent the time consumed in stage 1 and the green ones represent the time consumed in stage 2)

potential for an even higher processing speed. This approach will be a promising tool for further applications in online high frequency data processing and analysis.

7 Acknowledgements

The second author would like to acknowledge the partial support through The Royal Society grant IE141329/2014 *Novel Machine Learning Paradigms to Address Big Data Streams*. The third, fourth, and fifth authors would like to acknowledge the support by the Spanish Goverment under the project TRA2013-48314-C3-1-R and the project TRA2015-63708-R.

References

1. Fukunaga, K., Hostetler, L.: The estimation of the gradient of a density function, with applications in pattern recognition. IEEE Transactions on Information Theory, vol. 21(1), pp. 32-40 (1975).
2. MacQueen, J.: Some methods for classification and analysis of multi-variate observations. In: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics, Berkeley, pp. 281-297 (1967).
3. Bezdek, J. C., Ehrlich, R. and Full, W.: FCM: The fuzzy c-means clustering algorithm. Computers & Geosciences, vol.10(2), pp.191-203 (1984).
4. Johnson, S.: Hierarchical clustering schemes. Psychometrika, Vol. 32(3), pp. 241-254 (1967).
5. de Oliveira, J. V. and Pedrycz, W.(eds.). Advances in Fuzzy Clustering and Its Applications. New York: Wiley (2007).
6. Yager, R. and Filev, D.: Generation of fuzzy rules by mountain clustering. Journal of Intelligent & Fuzzy Systems, vol.2(3), pp. 209-219 (1994).
7. Chiu, S.L.: Fuzzy model identification based on cluster estimation. Journal of Intelligent & Fuzzy Systems, vol.2(3), pp. 1064-1246 (1994).

Table 3: Performance Comparison (PTE denotes *Parallel_TEDA*; SuC denotes Subtractive clustering)

Clustering Approaches	User Input	Data set	Measures			
			C_0	S_{max}	S_{min}	t_{exe}
PTE	**None**	A1	**20**	**166**	**135**	**0.1388**
SuC	R=0.3		9	560	152	0.3728
	R=0.1		31	153	27	0.3445
ELM	R=600		4	2532	2	0.8820
	R=200		63	914	1	5.6551
PTE	**None**	A2	**35**	**176**	**125**	**0.2741**
SuC	R=0.3		11	712	166	0.6811
	R=0.1		35	157	143	0.6954
ELM	R=600		9	3583	1	1.7233
	R=200		105	914	1	14.8939
PTE	**None**	A3	**50**	**176**	**125**	**0.4956**
SuC	R=0.3		13	884	298	1.1141
	R=0.1		50	165	140	1.1323
ELM	R=600		4	3583	1	2.7453
	R=200		132	818	1	28.0227
PTE	**None**	S1	**15**	**359**	**294**	**0.3372**
SuC	R=0.3		10	670	323	0.6281
	R=0.1		15	355	296	0.6114
ELM	R=10000		4	2532	2	0.8820
	R=3000		63	914	1	5.6551
PTE	**None**	S2	**15**	**359**	**286**	**0.3288**
SuC	R=0.3		10	833	319	0.6313
	R=0.1		15	351	289	0.6215
ELM	R=10000		13	1700	1	2.4259
	R=3000		102	358	1	14.9862

8. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: 2nd International Conference on Knowledge Discovery and Data Mining. Portland, Oregan, vol. 96(34), pp. 226-231 (1996).
9. Wang, C., Lai, J., Huang D. and Zheng, W.: SVStream: A support vector-based algorithm for clustering data streams. IEEE Transactions on Knowledge and Data Engineering, vol.25(6), pp.1410–1424 (2013).
10. Baruah, R. and Angelov, P.: Evolving local means method for clustering of streaming data. IEEE Congress on Computational Intelligence, Brisbane, Australia, pp.2161–2168 (2012).
11. Hyde, R. and Angelov, P.: A fully autonomous data density based clustering technique. In: IEEE Symposium on evolving and autonomous learning systems, Orlando, USA pp. 116–123 (2014).
12. Angelov, P., Gu, X., Gutierrez, G., Iglesias, J. A. and Sanchis, A.: Autonomous data density based clustering method. In: 2016 IEEE World Congress on Computational Intelligence. Vancouver, Canada, to appear (2016).
13. Guha, S., Mishra, N., Motwani, R. and O'Callaghan, L.: Clustering data streams. In: Proceedings of the Annual Symposium on Foundations of Computer Scuebce (FOCS), Redondo

Beach, CA, pp.359–366 (2000).
14. Guha, S., Meyerson, A., Mishra, N., Motwani, R. and O'Callaghan, L.: Clustering data streams: Theory and practice. IEEE Transactions on Knowledge and Data Engineering, vol. 15(3), pp. 515-528 (2003).
15. Aggarwal, C., Han, J., Wang, J. and Yu, S.: A framework for clustering evolving data streams. In: proceedings of the International Conference on Very Large Data Bases, Berlin, Germany, pp. 81–92 (2003).
16. Comode, G., Muthukrishnan, S. and Zhang, W.: Conquering the divide: Continuous clustering of distributed data streams. In: Proceedings of the International Conference on Data Engineering, Istanbul, pp.1036–1045 (2007).
17. Gama, J., Rodrigues, P., and Sebastio, R.: Evaluating algorithms that learn from data streams. In: Proceedings of the ACM Symposium on Applied Computing, Hawaii, pp.1496–1500 (2009).
18. Angelov, P.. Outside the Box: An Alternative Data Analytics Framework. Journal of Automation, Mobile Robotics and Intelligent Systems, 8(2), pp. 53–59 (2014).
19. Angelov, P.: Typicality distribution function- a new density-based data analytics tool. In: IEEE International Joint Conference on Neural Networks (IJCNN), Killarney, pp. 1–8 (2015).
20. Angelov, P., Gu, X., Kangin, D. and Principe, J.: TEDA: Typicality based Empirical Data Analytics. Submitted to Information Sciences (2016).
21. Angelov, P. and Filev, D.: Simpl_eTS: A simplified method for learning evolving Takagi-Sugeno fuzzy models. In: IEEE International Conference on Fuzzy Systems, Reno, USA, pp. 1068–1073 (2005).
22. Angelov, P.: Autonomous Learning Systems from Data Stream to Knowledge in Real Time. West Sussex, United Kingdom: John Wiley & Sons, Ltd. (2012).
23. Lughofer, E. and Angelov, P.: Handling drifts and shifts in on-line data streams with evolving fuzzy systems. Applied Soft Computing Journal, vol.11(2), pp.2057–2068 (2011).
24. Saw, J., Yang, M. and Mo, T.: Chebyshev inequality with estimated mean and variance. The American Statistician, vol.38(2), pp. 130-132 (1984).
25. Clustering Datasets - University of Eastern Finland, last access: 12 May, 2016. `http://cs.joensuu.fi/sipu/datasets/`