A Method for Autonomous Data Partitioning

Xiaowei Gu[1], Plamen P. Angelov[1,2], and José C. Príncipe[3]

[1]School of Computing and Communications, Lancaster University, Lancaster, LA1 4WA, UK;

[2]Technical University, Sofia, 1000, Bulgaria (Honorary Professor);

[3]Computational NeuroEngineering Laboratory, Department of Electrical and Computer Engineering, University of Florida, USA.

E-mail: x.gu3@lancaster.ac.uk, p.angelov@lancaster.ac.uk, principe@cnel.ufl.edu

Abstract- In this paper, we propose a fully autonomous, local-modes-based data partitioning algorithm, which is able to automatically recognize local maxima of the data density from empirical observations and use them as focal points to form shape-free data clouds, i.e. a form of Voronoi tessellation. The method is free from user- and problem- specific parameters and *prior* assumptions. The proposed algorithm has two versions: *i)* offline for static data and *ii)* evolving for streaming data. Numerical results based on benchmark datasets prove the validity of the proposed algorithm and demonstrate its excellent performance and high computational efficiency compared with the state-of-art clustering algorithms.

Keywords- autonomous, data partitioning, local modes, Voronoi tessellation.

1. Introduction

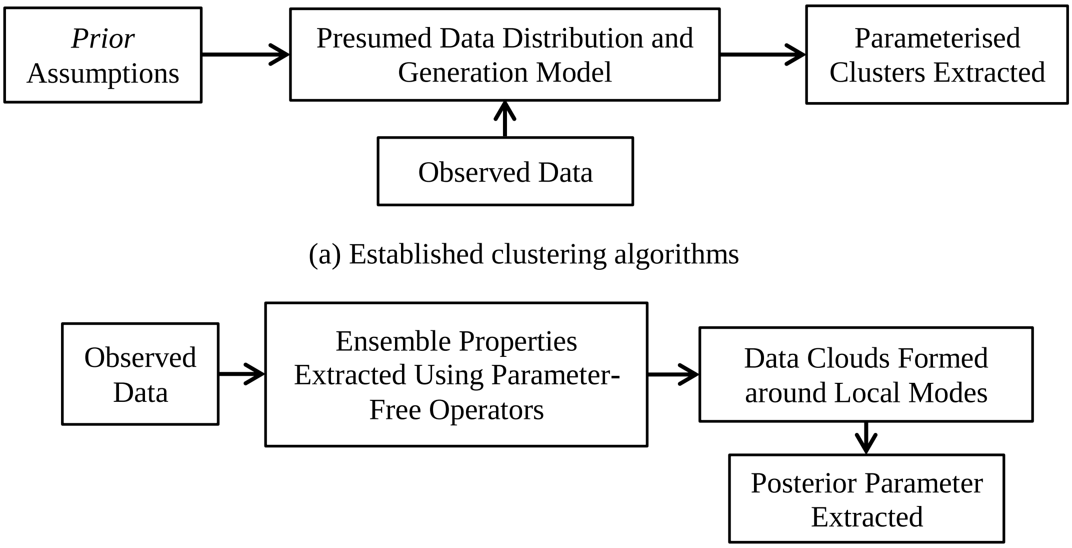
Clustering aims at identifying the intrinsic data patterns. The main task of clustering methods is to group similar data samples together and separate them from the less similar ones. As a common unsupervised machine learning technique for statistical data analysis, clustering has long been considered as one of the most effective tools for information extraction and summarization because it does not require labelling [34]. Clustering is widely used in many fields of study including pattern recognition, image analysis, biology, natural language processing, etc., and thus, is a hot topic in the field of data analytics [34],[50]. Nonetheless, established clustering methods require proper parameter settings that can be very different across datasets, and their performance is heavily relying on the *prior* knowledge and assumptions. However, *prior* knowledge in real situations is usually very limited and assumptions are seldom met.

In this paper, we propose a fully autonomous local-modes-based free-shape data partitioning method named “autonomous data partitioning (ADP)”. Local modes are akin of the peaks of the local data density. The proposed method is free from user- and problem-specific parameters and *prior* assumptions. It also utilizes parameter-free operators to disclose the underlying data distribution and ensemble properties of the empirically observed data using the natural distance metric. Based on these, the local modes representing the local maxima of the data density can be automatically determined. The proposed approach uses these local modes to partition the data space in shape-free data clouds [3],[4] (a kind of Voronoi tessellation [41]). This objectively represents the real data distribution unlike the desirable/expected (often subjectively) preferences represented by traditional clusters. The comparison between the established clustering approaches and the proposed ADP algorithm is presented in Fig. 1.

The proposed ADP algorithm has an offline version and an evolving version. Its offline version is based on the ranks of the data samples in terms of their densities and mutual distances instead of the commonly used means and variances. Ranks are very important but other approaches avoid them because they are nonlinear and discrete operators. Therefore, the offline version is more stable and effective in partitioning static datasets. The evolving version is also generic and fully data-driven. It can continue the offline partitioning process with the newly arrived data, but can also start without initial conditions.

Numerical experiments based on benchmark datasets demonstrate that the proposed algorithm is able to achieve excellent clustering performance very quickly without user- or problem- specific *prior* knowledge. More importantly, the ADP algorithm performs much better on the larger size and high dimensional datasets on which the current clustering algorithms struggle.

The remainder of this paper is organized as follows. The related work is reviewed in section 2. Section 3 summarizes the theoretical basis of the proposed ADP approach. The ADP approach is introduced in section 4. The numerical examples are presented in section 5. Section 6 provides a detailed analysis and discussions based on section 5 and section 7 concludes the paper.


Prior Assumptions
Presumed Data Distribution and Generation Model
Parameterised Clusters Extracted
Observed Data
Observed Data
Ensemble Properties Extracted Using Parameter-Free Operators
Data Clouds Formed around Local Modes
Posterior Parameter Extracted

(a) Established clustering algorithms

(b) The ADP algorithm

Fig.1. Comparison between the current clustering algorithms and the proposed ADP algorithm

2. Related Work

As it was stated in the first section, the current clustering algorithms need various types of free parameters to be predefined by users. These user inputs include the number of clusters/centroids [10],[33],[39],[50], threshold [18],[32],[49],[53], the type and parameters of the kernel function [17],[19],[21],[43], radius of the clusters [12],[20],[21],[35], net size [34],[42], etc. They can significantly influence the performance and efficiency of the algorithms. Pre-setting the suitable parameters for clustering algorithms is always a challenging task and requires a certain amount of *prior* knowledge. In real situations, however, *prior* knowledge is very limited and not enough for users to decide the best inputs for these algorithms in advance. Without carefully chosen parameters, the efficiency as well as the effectiveness of the clustering algorithms is lower than reported.

One popular alternative method to overcome the problem is to pre-define parametric penalty function and iteratively achieve the optimal clustering result by satisfying the constraints imposed by the penalty function [25],[48],[54]. However, this kind of methods also raises the questions about the objectiveness and effectiveness of the penalty function. In addition, the computational efficiency of these methods deteriorates rapidly with the increase of the scale of the data.

To avoid the requirement of user inputs, a number of so-called “nonparametric” clustering techniques were proposed within the last decade [11],[27],[36]. However, instead of being really nonparametric, these clustering techniques, in fact, tune a number of parameters in advance and fix them for different problems.

The other problem that these clustering approaches [11],[12],[18],[19],[21],[27],[36],[40],[53] currently have is that they simplify the real data representation by assuming that the data follows a specific distribution (i.e. most often a Gaussian). However, this *prior* assumption oversimplifies the real problems, and often leads to unnatural clustering results. Lifting these assumptions is possible with information theoretic [31] and spectral clustering [38], but at the expense of much higher computational cost.

In comparison to the state-of-the-art approaches, the proposed ADP method has the following two unique properties:

1) it is data-driven and free from user- and problem- specific parameters;

2) it does not impose a data generation model on the empirical observations.

3. Theoretical Basis

The proposed ADP approach is proposed within the recently introduced Empirical Data Analysis (EDA) framework [5],[6],[8]. EDA resembles statistical learning in its nature but is free from the range of assumptions made in traditional probability theory and statistical learning approaches [5],[6],[8]. EDA measures play an instrumental role in the proposed ADP approach for extracting the ensemble properties from the observed data (see Fig. 1(b)), and frees the ADP algorithm from the requirement to make *prior* assumptions on the data generation model and user- and problem-specific parameters (see Fig. 1(a)). Most importantly, they guarantee objectiveness of the partitioning results.

Firstly, we define the data set/stream in the Euclidean data space $\mathbf{R}^N$ as $\left\{\boldsymbol{x}\right\}_K=\left\{\boldsymbol{x}_1,\boldsymbol{x}_2,...,\boldsymbol{x}_K\right\}$, $\boldsymbol{x}_k=\left[x_{k,1},x_{k,2},...,x_{k,N}\right]^{\mathrm{T}}\in\mathbf{R}^N$, where $k=1,2,...,K$ is an order index. We, further, consider that some data samples within the data set/stream can have the same value, i.e. $\exists\boldsymbol{x}_k=\boldsymbol{x}_j,\quad k\neq j$. The set of sorted unique data samples is denoted as $\left\{\boldsymbol{u}\right\}_{L_K}=\left\{\boldsymbol{u}_1,\boldsymbol{u}_2,...,\boldsymbol{u}_{L_K}\right\}$, $\boldsymbol{u}_j=\left[u_{j,1},u_{j,2},...,u_{j,N}\right]^{\mathrm{T}}$($j=1,2,...,L_K$), $\left\{\boldsymbol{u}\right\}_{L_K}\subseteq\left\{\boldsymbol{x}\right\}_K$, $L_K\leq K$ with the corresponding occurrence $\left\{f\right\}_{L_K}=\left\{f_1,f_2,...,f_{L_K}\right\}$, $\sum_{k=1}^{L_K}f_k=1$.

In this paper, we use the natural metric of the space of samples (i.e. Euclidean distance) for clarity, however, various types of distances within the data space, $\mathbf{R}^N$ can be considered as well. In the remainder of this paper, all the derivations are conducted at the K^{th} time instance except when specifically mentioned.

In this section, we will summarize the nonparametric EDA measures that we use in ADP:

i) local density, D [6],[8];

ii) global density, D^G [6],[8];

and their recursive implementations. They will be briefly reviewed next to make the paper self-contained.

3.1 Local Density

Local density [6] is a measure within EDA framework identifying the main local mode of the data distribution and is derived empirically from all the observed data samples in a parameter-free way. The local density, D of the data sample $\boldsymbol{x}_i$ is expressed as follows ($k=1,2,...,K;\ L_K>1$) [6],[8]:

$$D_K\left(\boldsymbol{x}_k\right)=\frac{\sum_{j=1}^{K}\sum_{l=1}^{K}d^2\left(\boldsymbol{x}_j,\boldsymbol{x}_l\right)}{2K\sum_{l=1}^{K}d^2\left(\boldsymbol{x}_k,\boldsymbol{x}_l\right)} \tag{1}$$

where $d\left(\boldsymbol{x}_k,\boldsymbol{x}_l\right)$ is the distance between data samples $\boldsymbol{x}_k$ and $\boldsymbol{x}_j$; the coefficient 2 is used in the denominator for normalization (because each distance is counted twice in the numerator).

For the case of Euclidean distance, the calculation of $\sum_{l=1}^{K} d^2\left(\boldsymbol{x}_k,\boldsymbol{x}_l\right)=\sum_{l=1}^{K}\left\|\boldsymbol{x}_k-\boldsymbol{x}_l\right\|^2$ and $\sum_{j=1}^{K}\sum_{l=1}^{K} d^2\left(\boldsymbol{x}_j,\boldsymbol{x}_l\right)= \sum_{j=1}^{K}\sum_{l=1}^{K}\left\|\boldsymbol{x}_j-\boldsymbol{x}_l\right\|^2$ can be simplified by using the mean of $\left\{\boldsymbol{x}\right\}_K$, $\boldsymbol{\mu}_K$ and the average scalar product, X_K as [6],[8]:

$$\sum_{l=1}^{K}\left\|\boldsymbol{x}_k-\boldsymbol{x}_l\right\|^2=K\left(\left\|\boldsymbol{x}_k-\boldsymbol{\mu}_K\right\|^2+X_K-\left\|\boldsymbol{\mu}_K\right\|^2\right) \tag{2}$$

$$\sum_{j=1}^{K}\sum_{l=1}^{K}\left\|\boldsymbol{x}_j-\boldsymbol{x}_l\right\|^2=2K^2\left(X_K-\left\|\boldsymbol{\mu}_K\right\|^2\right) \tag{3}$$

$\boldsymbol{\mu}_K$ and X_K can be updated recursively as [4]:

$$\boldsymbol{\mu}_k=\frac{k-1}{k}\boldsymbol{\mu}_{k-1}+\frac{1}{k}\boldsymbol{x}_k;\quad \boldsymbol{\mu}_1=\boldsymbol{x}_1;\quad k=1,2,...,K \tag{4}$$

$$X_k=\frac{k-1}{k}X_{k-1}+\frac{1}{k}\left\|\boldsymbol{x}_k\right\|^2;\ X_1=\left\|\boldsymbol{x}_1\right\|^2;\ k=1,2,...,K \tag{5}$$

The recursive calculation of $\boldsymbol{\mu}_K$ and X_K allows for “one pass” evaluation, thus, ensures computation-efficiency because only the key aggregated/summarized information has to be kept in memory.

Combining (1)-(5), we can re-formulate the local density, D in a recursive form:

$$D_K\left(\boldsymbol{x}_k\right)=\frac{1}{1+\frac{\left\|\boldsymbol{x}_k-\boldsymbol{\mu}_K\right\|^2}{X_K-\left\|\boldsymbol{\mu}_K\right\|^2}} \tag{6}$$

From (6) we can see that when the Euclidean distance is used, the local density, D behaves as a **Cauchy function,** while **there is no *prior* assumption about the type of the distribution involved**. Note that $0<D_K\left(\boldsymbol{x}_k\right)\le 1$ and the closer $\boldsymbol{x}_k$ is to the main local mode, the higher the value of $D_K\left(\boldsymbol{x}_k\right)$ is.

3.2 Global density

The global density, D^G estimated at the unique data sample $\boldsymbol{u}_k$ is a weighted sum of its local density by the corresponding occurrence, f_k ($k=1,2,...,L_K;\ L_K>1$), expressed as follows [6],[8]:

$$D_K^G\left(\boldsymbol{u}_k\right)=\frac{f_i}{\sum_{j=1}^{L_K} f_j}D_K\left(\boldsymbol{u}_k\right)=\frac{f_k}{1+\frac{\left\|\boldsymbol{u}_k-\boldsymbol{\mu}_K\right\|^2}{X_K-\left\|\boldsymbol{\mu}_K\right\|^2}} \tag{7}$$

It is clear that it behaves locally as a Cauchy function, but has multiple local modes/peaks which depend on f_k (if $L_K=K$, global density reduces to the local density but with a smaller amplitude). The largest of the peaks is called the global peak. The global density can directly disclose the data pattern without any further pre-processing. This property can be very useful in real cases where the units of measurement are fixed and data samples measured at different indices are likely to share the same values. The reason we do not stop with D^G but move further with partitioning is that although it is fully automatic and objective, D^G often reveals too many local peaks.

4. The Autonomous Data Partitioning Algorithm

In this section, we will describe the proposed ADP algorithm for further refining the local modes of the data set/stream. The local modes are defined as the local maxima of the global data density and are constructed directly from samples. These local modes play a key role in partitioning the data space into shape-free data clouds [3],[4] by aggregating data samples around them and forming naturally a Voronoi tessellation [41]. We will present two independent algorithms for the two versions, *i)* offline and *ii)* evolving.

4.1 Offline ADP Algorithm

The offline ADP algorithm works with the global density, D^G of the observed data samples. It is based on the ranks of the data samples in terms of their global densities and mutual distributions. Its main procedure is described as follows:

Stage 1: Rank order the samples with regards to the distance to the global mode

We start by organizing the unique data samples $\{\boldsymbol{u}\}_K$ into an indexing list, denoted by $\{\boldsymbol{u}^*\}_K$ based on their mutual distances and values of the global density, D^G .

Firstly, the global densities of the unique data samples, $D_K^G(\boldsymbol{u}_i)$ ($i=1,2,...,L_K$) are calculated using (7). The unique data sample with the highest global density is then selected as the first element of the indexing list $\{\boldsymbol{u}^*\}_{L_K}$:

$$\boldsymbol{u}^{*1} = \underset{j=1,2,...,L_K}{\arg\max}\left(D_K^G\left(\boldsymbol{u}_j\right)\right) \tag{8}$$

We set $\boldsymbol{u}^{*1}$ as the first reference point: $\boldsymbol{u}^{*r} \leftarrow \boldsymbol{u}^{*1}$ and remove $\boldsymbol{u}^{*1}$ from $\{\boldsymbol{u}\}_{L_K}$. Then, by selecting out the unique data sample which is nearest to $\boldsymbol{u}^{*r}$, the second element of $\{\boldsymbol{u}^*\}_{L_K}$ denoted by $\boldsymbol{u}^{*2}$ is identified and it is set as the new reference point ($\boldsymbol{u}^{*r} \leftarrow \boldsymbol{u}^{*2}$) and, is also removed from $\{\boldsymbol{u}\}_{L_K}$. The process is repeated until $\{\boldsymbol{u}\}_{L_K}$ becomes empty, and the rank ordered list $\{\boldsymbol{u}^*\}_{L_K}$ is finally derived. Based on this list, we can rank the global density of the unique data samples as: $\left\{D_K^G\left(\boldsymbol{u}^*\right)\right\} =$ $\left\{D_K^G\left(\boldsymbol{u}^{*1}\right), D_K^G\left(\boldsymbol{u}^{*2}\right), ..., D_K^G\left(\boldsymbol{u}^{*L_K}\right)\right\}$.

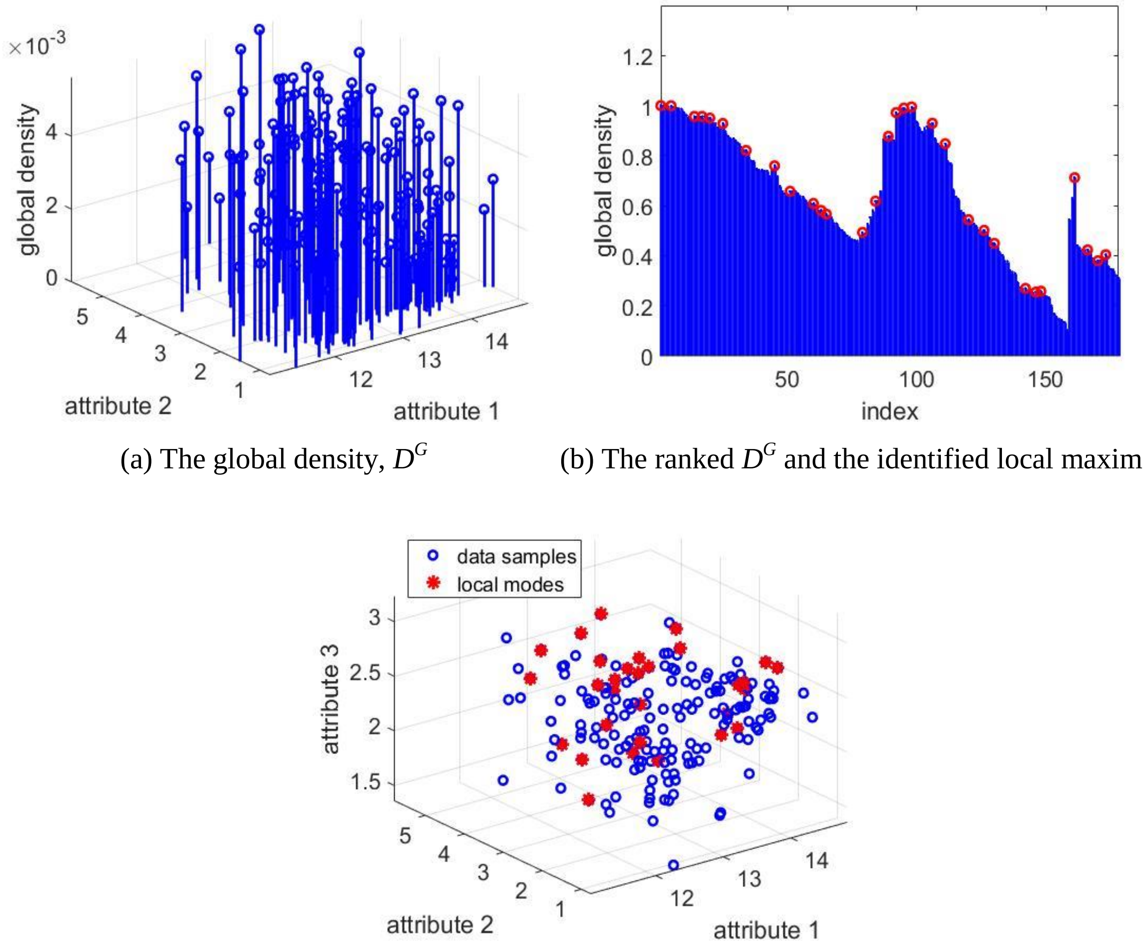
An example using the wine dataset [1] is shown in Fig. 2. Fig. 2(a) shows the global density of the wine dataset, Fig. 2(b) shows the ranked global density.

Stage 2: Detecting local maxima (local modes)

Based on the list $\{\boldsymbol{u}^*\}_{L_K}$ and the ranked global density $\left\{D_K^G\left(\boldsymbol{u}^*\right)\right\}$, we can identify all the data samples with the local maxima of D^G directly as follows:

$$\begin{gathered} IF\ \left(\operatorname{sgn}\left(D_K^G\left(\boldsymbol{u}^{*j}\right) - D_K^G\left(\boldsymbol{u}^{*(j+1)}\right)\right) = 1\right)\ AND\ \left(\operatorname{sgn}\left(D_K^G\left(\boldsymbol{u}^{*(j-1)}\right) - D_K^G\left(\boldsymbol{u}^{*j}\right)\right) = -1\right) \\ THEN\ \left(\boldsymbol{u}^{*j}\ is\ a\ local\ maximum\ of\ D^G\right) \end{gathered} \tag{9}$$

where $\operatorname{sgn}(x)=\begin{cases}1 & x>0\\ 0 & x=0\\ -1 & x<0\end{cases}$ is the sign function; we denote the collection of data samples with the local maxima of D^G as $\{\boldsymbol{u}^{**}\}_{L_K^*}=\{\boldsymbol{u}^{**j} \mid j=1,2,...,L_K^*\}$ ($L_K^*<L_K$). The local maxima (peaks) identified from the Wine dataset [1] are marked by red circles in Fig. 2(b). The locations of the local maxima in the data space are depicted in Fig. 2(c).



(a) The global density, D^G

(b) The ranked D^G and the identified local maxima

(c) The locations of local maxima in the data space

Fig.2. Illustrative example based on the wine dataset

Stage 3: Forming data clouds

The local peaks identified from the indexing list, namely, $\{\boldsymbol{u}^{**}\}_{L_K^*}$, are then used to attract the data samples $\{\boldsymbol{x}\}_K$ that are closer to them using a min operator:

$$\mathit{winning\ cloud}=\underset{j=1,2,...,L_K^*}{\arg\min}\left(\left\|\boldsymbol{x}_i-\boldsymbol{u}^{**j}\right\|\right);\ i=1,2,...,K;\ L_K^*>1 \tag{10}$$

By assigning all the data samples to the nearest local maxima, a number of Voronoi tessellations [41] are naturally formed and data clouds are built around the local maxima [3],[4]. More importantly, the process is free from any threshold.

After the data clouds are formed, the actual centre (mean) $\boldsymbol{\mu}^j$ and the standard deviation σ^j ($j=1,2,...,L_K^*$) per data cloud and the support, S^j, namely, the number of data samples within the data cloud can be calculated easily. Note that, all this procedure is post factum, i.e. it is determined bottom up from the data without a *prior* assumption, except the selection of the metric.

Stage 4: Filtering local modes

The data clouds formed in the previous stage may contain some less representative ones; therefore, in this stage, we filter the initial Voronoi tessellations and combine them into larger, more meaningful data clouds.

The global densities at the data clouds centres $\{\boldsymbol{\mu}\}_{L_K^*}$ are firstly calculated as follows ($j=1,2,...,L_K^*$):

$$D_K^G\left(\boldsymbol{\mu}^j\right)=\frac{S^j}{1+\dfrac{\left\|\boldsymbol{\mu}^j-\boldsymbol{\mu}_K\right\|^2}{X_K-\left\|\boldsymbol{\mu}_K\right\|^2}} \tag{11}$$

where S^j is the support of the data cloud.

In order to identify the centres with the local maxima of the global density, we introduce the following three objectively derived quantifiers of the data pattern:

$$i)\ \eta_K^c=2\sum_{p=1}^{L_K^*-1}\sum_{q=p+1}^{L_K^*}\left\|\boldsymbol{\mu}^p-\boldsymbol{\mu}^q\right\|\Big/L_K^*\left(L_K^*-1\right) \tag{12}$$

$$ii)\ \gamma_K^c=\frac{\sum\limits_{\boldsymbol{x},\boldsymbol{y}\in\{\boldsymbol{\mu}\}_{L_K^*},\boldsymbol{x}\neq\boldsymbol{y},\ \|\boldsymbol{x}-\boldsymbol{y}\|\leq\eta_K^c}\left\|\boldsymbol{x}-\boldsymbol{y}\right\|}{M_\eta} \tag{13}$$

$$iii)\ \lambda_K^c=\frac{\sum\limits_{\boldsymbol{x},\boldsymbol{y}\in\{\boldsymbol{\mu}\}_{L_K^*},\boldsymbol{x}\neq\boldsymbol{y},\ \|\boldsymbol{x}-\boldsymbol{y}\|\leq\gamma_K^c}\left\|\boldsymbol{x}-\boldsymbol{y}\right\|}{M_\lambda} \tag{14}$$

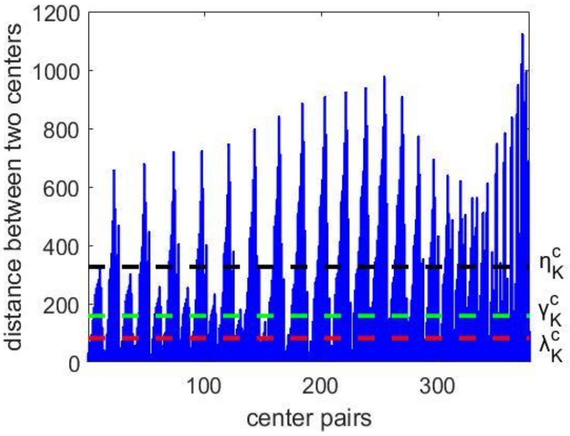


Fig.3. Plot of η_K^c , γ_K^c and λ_K^c on the wine dataset.

η_K^c is the average Euclidean distance between any pair of existing local modes. γ_K^c is the average Euclidean distance between any pair of existing local modes within a distance less than η_K^c , and M_η in (13) is the number of such local mode pairs. λ_K^c is the average Euclidean distance between any pair of existing local modes within a distance less than γ_K^c , M_γ in (14) is the number of such local mode pairs.

Note that η_K^c, γ_K^c and λ_K^c are not problem- specific and are parameter-free. The relationship between η_K^c, γ_K^c and λ_K^c is depicted in Fig. 3 using the wine dataset [1] again, where one can see that λ_K^c is a smaller value compared with η_K^c and γ_K^c which are obtained as the average distance of two very close data cloud centres.

The quantifier λ_K^c can be viewed as the estimation of the distances between the strongly connected data clouds condensing the local information from the whole dataset. Moreover, instead of relying on a fixed threshold, which may frequently fail, η_K^c, γ_K^c and λ_K^c are derived from the dataset objectively and conforming with the concept of a mode. Experience has shown that they provide meaningful tessellations, regardless of the distribution of the data.

Each centre $\boldsymbol{\mu}^j$ ($j=1,2,...,L_K^*$) is compared with the centres of neighbouring data clouds in terms of the global density:

$$IF\left(D_K^G\left(\boldsymbol{\mu}^j\right)=\max\left(\left\{\left\{D_K^G\left(\boldsymbol{\mu}\right)\right\}_n^j, D_K^G\left(\boldsymbol{\mu}^j\right)\right\}\right)\right)\ THEN\left(\boldsymbol{\mu}^j\ is\ one\ of\ the\ local\ maxima\right) \tag{15}$$

where $\left\{D^G\left(\boldsymbol{\mu}\right)\right\}_n^j$ is the collection of global densities of the neighbouring centres, which satisfy the following condition:

$$IF\left(\left\|\boldsymbol{\mu}^i-\boldsymbol{\mu}^j\right\|\leq\frac{\lambda_K^c}{2}\right)\ THEN\left(\boldsymbol{\mu}^i\ is\ neighbouring\ \boldsymbol{\mu}^j\right) \tag{16}$$

The criterion of neighbouring range is defined in this way because two centres with the distance smaller than γ_K^c can be considered to be potentially relevant in the sense of spatial distance; λ_K^c is the average distance between the centres of any two potentially relevant data clouds. Therefore, when the condition (16) is satisfied, both $\boldsymbol{\mu}^i$ and $\boldsymbol{\mu}^j$ are highly influencing each other and, the data samples within the two corresponding data clouds are strongly connected. Therefore, the two data clouds are considered as neighbours. This criterion also guarantees that only small-size (less important) data clouds that significantly overlap with large-size (more important) ones will be removed during the filtering operation.

After the filtering operation (stage 4), the data cloud centres with local maximum global densities denoted by$\left\{\boldsymbol{\mu}^*\right\}_{L_K^{**}}=\left\{\boldsymbol{\mu}^{*j}\mid j=1,2,...,L_K^{**},\ L_K^{**}\leq L_K^*\right\}$ are obtained. Then, $\left\{\boldsymbol{\mu}^*\right\}_{L_K^{**}}$ are used as local modes for forming data clouds in stage 3 and are filtered in stage 4.

Stages 3 and 4 are repeated until all the distances between the existing local modes exceed $\frac{\lambda_K^c}{2}$. Finally, we obtain the remaining centres with the local maxima of D^G, denoted by $\left\{\boldsymbol{\mu}^o\right\}$, and use them as the local modes to form data clouds using (10).

After the data clouds are formed, the corresponding centres, standard deviations, supports, members and other parameters of the formed data clouds can be extracted post factum. The final partitioning result of the wine dataset [1] is depicted in Fig. 4, where the dots in different colours stand for data samples from different data clouds, the black asterisks stand for the centres of the data clouds.

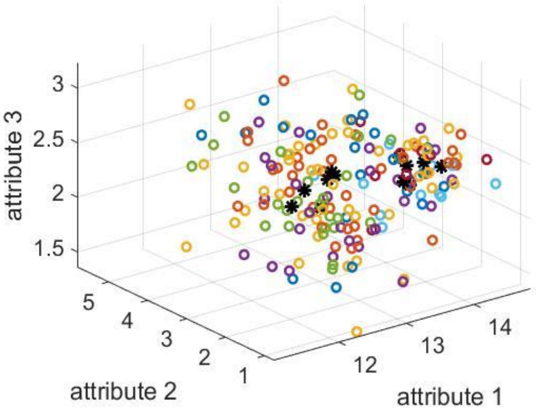


Fig. 4. The partitioning result of the wine dataset ("o" in different colours denote data samples of different data clouds, "*" denote local modes).

The main procedure of the proposed ADP algorithm (offline version) is presented in the following pseudo code.

Offline ADP Algorithm

```
Input: The static dataset $\{x\}_K$;
Algorithm Begins
  i. Calculate $D_K^G(\boldsymbol{u}_i)$ ($i=1,2,...,L_K$) using (7) for all unique data samples;
  ii. Find the unique data sample $\boldsymbol{u}^{*1}$ with global maximum of $D^G$ using (8);
  iii. Send $\boldsymbol{u}^{*1}$ into $\{\boldsymbol{u}^*\}_{L_K}$ and $D_K^G(\boldsymbol{u}^{*1})$ into $\{D_K^G(\boldsymbol{u}^*)\}$ and delete $\boldsymbol{u}^{*1}$ from $\{\boldsymbol{u}\}_{L_K}$;
  iv. $\boldsymbol{u}^{*r} \leftarrow \boldsymbol{u}^{*1}$;
  v. While $\{\boldsymbol{u}\}_{L_K} \neq \phi$
      * Find the unique data sample which is nearest to $\boldsymbol{u}^{*r}$;
      * Send the data sample and the corresponding $D^G$ to $\{\boldsymbol{u}^*\}_{L_K}$ and $\{D_K^G(\boldsymbol{u}^*)\}$, respectively;
      * Delete the data sample from $\{\boldsymbol{u}\}_{L_K}$;
      * Set the data sample as the next $\boldsymbol{u}^{*r}$;
  vi. End While
  vii. Filter $\{\boldsymbol{u}^*\}_{L_K}$ and $\{D_K^G(\boldsymbol{u}^*)\}$ using (9) and obtain $\{\boldsymbol{u}^{**}\}_{L_K^*}$ as local modes of data clouds;
  viii. While $\{\boldsymbol{u}^{**}\}_{L_K^*}$ are not fixed
      * Use $\{\boldsymbol{u}^{**}\}_{L_K^*}$ to form the data clouds from $\{x\}_K$ using (10);
      * Obtain the new centres $\{\boldsymbol{\mu}\}_{L_K^*}$ and support $\{S\}_{L_K^*}$ of the data clouds;
      * Calculate $D_K^G(\boldsymbol{\mu}^j)$( $j=1,2,...,L_K^*$ ) using (11);
```

```
        * Find the local maxima of $D_K^G\left(\boldsymbol{\mu}^j\right)$ ( $j=1,2,...,L_K^*$ ) using (15);
        * Select $\left\{\boldsymbol{\mu}^*\right\}_{L_K^{**}}$ with local maxima $D_K^G\left(\boldsymbol{\mu}^j\right)$ as the new local modes.
        * $\left\{\boldsymbol{u}^{**}\right\}_{L_K^*} \leftarrow \left\{\boldsymbol{\mu}^*\right\}_{L_K^{**}}$;
        * $L_K^* \leftarrow L_K^{**}$;
  ix. End While
  x. $\left\{\boldsymbol{\mu}^o\right\} \leftarrow \left\{\boldsymbol{u}^{**}\right\}_{L_K^*}$;
  xi. Build the data clouds with $\left\{\boldsymbol{\mu}^o\right\}$ using (10).
```

Algorithm Ends

Output: **Data clouds formed around $\left\{\boldsymbol{\mu}^o\right\}$.**

4.2 Evolving ADP Algorithm

The proposed evolving ADP algorithm works with the local density, D of the streaming data. This algorithm is able to start “from scratch”, i.e. with a single sample. In addition, a hybrid between the evolving and the offline versions is also possible.

The main procedure of the evolving algorithm is as follows.

Stage 1: Initialization

We select the first data sample within the data stream as the first local mode. The proposed algorithm then starts to self-evolve its structure and update the parameters based on the arriving data samples.

Stage 2: System structure and meta-parameters update

For each newly arriving data sample at the current time instance $k \leftarrow k+1$, denoted as $\boldsymbol{x}_k$, the global meta-parameters $\boldsymbol{\mu}_k$ and X_k are updated with $\boldsymbol{x}_k$ firstly using (4) and (5). The local density at $\boldsymbol{x}_k$ and the centres of all the existing data clouds, $D_k\left(\boldsymbol{x}_k\right)$ and $D_k\left(\boldsymbol{\mu}_k^i\right)$ ($i=1,2,...,C_k$) are calculated using (6); here, we use C_k as the number of existing local modes at the k^{th} time instance.

Then, the following condition [4] is checked to decide whether $\boldsymbol{x}_k$ will form a new data cloud:

$$\begin{gathered} IF\left(D_k\left(\boldsymbol{x}_k\right) > \max_{i=1}^{C_k}\left(D_k\left(\boldsymbol{\mu}_k^i\right)\right)\right) OR \left(D_k\left(\boldsymbol{x}_k\right) < \min_{i=1}^{C_k}\left(D_k\left(\boldsymbol{\mu}_k^i\right)\right)\right) \\ THEN\ \left(\boldsymbol{x}_k\ becomes\ a\ new\ focal\ point\right) \end{gathered} \tag{17}$$

If the condition is met, a new data cloud is added with $\boldsymbol{x}_k$ as its local mode ($C_k \leftarrow C_k+1$, $\boldsymbol{\mu}_k^{C_k} \leftarrow \boldsymbol{x}_k$ and $S_k^{C_k} \leftarrow 1$).

Otherwise, the existing local mode closest to $\boldsymbol{x}_k$ is found, denoted as $\boldsymbol{\mu}_k^n$. Then, the following condition is checked before $\boldsymbol{x}_k$ is assigned to the data cloud formed around $\boldsymbol{\mu}_k^n$:

$$IF\left(\left\|\boldsymbol{x}_k-\boldsymbol{\mu}_k^n\right\| \le \frac{\eta_k^c}{2}\right) THEN\left(\boldsymbol{x}_k\ is\ assigned\ to\ \boldsymbol{\mu}_k^n\right) \tag{18}$$

However, it is not computationally efficient to calculate η_k^c at each time when a new data sample arrives. Since the average distance between all the data samples η_k^d is approximately equal to η_k^c, $\eta_k^c \approx \eta_k^d$, η_k^c can be replaced as:

$$\eta_k^c \approx \eta_k^d = \sqrt{\frac{\sum_{i=1}^{k}\sum_{l=1}^{k}\left\|\boldsymbol{x}_i - \boldsymbol{x}_l\right\|^2}{k^2}} = \sqrt{2\left(X_k - \left\|\boldsymbol{\mu}_k\right\|^2\right)} \tag{19}$$

If the condition (18) is satisfied, then $\boldsymbol{x}_k$ is associated with the nearest existing local mode $\boldsymbol{\mu}_k^n$ and the meta-parameters of $\boldsymbol{\mu}_k^n$ are updated as follows:

$$S_k^n \leftarrow S_k^n + 1 \tag{20}$$

$$\boldsymbol{\mu}_k^n \leftarrow \frac{S_k^n - 1}{S_k^n}\boldsymbol{\mu}_k^n + \frac{1}{S_k^n}\boldsymbol{x}_{k+1} \tag{21}$$

If the condition (18) is not satisfied, then $\boldsymbol{x}_k$ starts a new data cloud: $C_k \leftarrow C_k + 1$, $\boldsymbol{\mu}_k^{C_k} \leftarrow \boldsymbol{x}_k$ and $S_k^{C_k} \leftarrow 1$.

The local modes and supports of other data clouds that do not get the new data sample stay the same for the next processing cycle. After the update of the system structure and the meta-parameters, the algorithm is ready for the next data sample.

Stage 3: Forming data clouds

When there are no more data samples, the identified local modes (renamed as $\left\{\boldsymbol{\mu}^o\right\}$) are used to build data clouds using (10). The parameters of these data clouds can be extracted post factum.

The main procedure of the proposed ADP algorithm (evolving version) is presented in the following pseudo code.

Evolving ADP Algorithm

```
Input: The data stream $\{x\}_K$;
Algorithm Begins
  A. While the new data sample $\boldsymbol{x}_k$ of the data stream is available (or until interrupted)
        i. If ($k=1$) Then
           1. $\boldsymbol{\mu}_1 \leftarrow \boldsymbol{x}_1$;   $X_1 \leftarrow \|\boldsymbol{x}_1\|^2$;   $C_1 \leftarrow 1$;   $\boldsymbol{\mu}_1^1 \leftarrow \boldsymbol{x}_1$;   $S_1^1 \leftarrow 1$;
        ii. End If
        iii. If ($k>1$) Then
           1. $k \leftarrow k+1$;
           2. Update $\boldsymbol{\mu}_k$ and $X_k$ with $\boldsymbol{x}_k$ using (4) and (5);
           3. If (Condition (17) is met) Then
              * $C_k \leftarrow C_k + 1$;   $S_k^{C_k} \leftarrow 1$;   $\boldsymbol{\mu}_k^{C_k} \leftarrow \boldsymbol{x}_k$;
           4. Else
              * Find the nearest local mode $\boldsymbol{\mu}_k^n$;
              * If (Condition (18) is met) Then
```

```
                    - $C_k \leftarrow C_k$;   $S_k^n \leftarrow S_k^n + 1$;   $\boldsymbol{\mu}_k^n \leftarrow \frac{S_k^n - 1}{S_k^n} \boldsymbol{\mu}_k^n + \frac{1}{S_k^n} \boldsymbol{x}_k$;
                * Else
                    - $C_k \leftarrow C_k + 1$;   $S_k^{C_k} \leftarrow 1$;   $\boldsymbol{\mu}_k^{C_k} \leftarrow \boldsymbol{x}_k$;
                * End If
            5. End If
        iv. End If
    B. End While
    C. Build the data clouds with $\{\boldsymbol{\mu}^o\}$ using (10).
Algorithm Ends
Output: Data clouds formed around $\{\boldsymbol{\mu}^o\}$.
```

4.3 Handling the Outliers

After the data clouds are formed by all the identified local modes, one may notice some data clouds with support equal to 1, which means that there is no sample associated with these data clouds except for the local modes. This kind of local modes are considered to be outliers and they are assigned to the nearest normal data cloud using (10) and the meta-parameters of the data clouds that receive new members are updated using (20) and (21). Nonetheless, we have to stress that these abnormal local modes are ignored from the partitioning results; they can still be kept in memory in case new data samples arrive.

5. Numerical Examples

In this section, we will study the performance of the proposed algorithms. All the numerical experiments are conducted with Matlab2015a on a PC with dual core i7 processor with clock frequency 3.4GHz each and 16GB RAM. The links to the code of the ADP algorithm and benchmark data/image sets we use in the numerical examples of this paper are given in Appendix.

5.1 Experiments on Numerical Datasets

In this subsection, a number of benchmark datasets are used in the performance evaluation as tabulated in Table 1. During the experiments, we assume that we do not have any *prior* knowledge about the benchmark datasets. The following well-known algorithms are used for comparison:

i) MS: Mean-shift clustering algorithm [19];

ii) SUB: Subtractive clustering algorithm [18];

iii) DBS: DBScan clustering algorithm [21];

iv) SOM: Self-organizing map algorithm [34];

v) ELM: Evolving local means clustering algorithm [20];

vi) DP: Density peaks clustering algorithm [43];

*vii)*NMM: Nonparametric mixture model based clustering algorithm [11];

*viii)*NMI: Nonparametric mode identification based clustering algorithm [36];

ix) CEDS: Clustering of evolving data streams algorithm [29].

Table 1. Details of the Benchmark Datasets for Evaluation

Abbreviation	**Dataset**	**N_A** [b]	**N_S** [c]	**N_C** [d]
PI	PIMA [47]	8	768	2
BA	Banknote Authentication [37]	4	1372	2
S1	S1 [23]	2	5000	15
S2	S2 [23]	2	5000	15
CA	Cardiotocography [9]	22	2126	3
PB	Pen-Based Handwritten Digits Recognition [2]	16	10992	10
ST	Steel Plates Faults [14]	27	1941	7
MU	Multiple Features [30]	649	2000	10
OD	Occupancy Detection [16] [a]	5	20560	2
MA	MAGIC Gamma Telescope [13]	10	19020	2
LE	Letter Recognition [24]	16	20000	26

[a] The time stamps in the original dataset have been removed; [b] Number of attributes; [c] Number of samples; [d] Number of classes.

The free parameters and *prior* assumptions required by the algorithms are listed in Table 2. The free parameter settings used in the experiments by the algorithms are also presented in this table. Due to the very limited *prior* knowledge during the experiments, the values of these free parameters as listed in Table 2 are determined by the recommendations and/or the experimental settings in the published literature to maximize performance of the datasets. In contrast, our approach does not utilize any of this knowledge, as it is self-organizing and autonomous.

In order to objectively compare the performance of different algorithms, we consider the following measures:

i) Number of data clouds/clusters (C), which should be equal or larger than the number of classes in the dataset;

ii) Calinski Harabasz index (CH) [15], which is used to estimate the optimal number of clusters; the higher the Calinski Harabasz index is, the better the clustering result is;

iii) Mean Silhouette coefficient (SI) [44], which is an indication of how well each sample lies within its cluster. The value range of this index is from -1 to 1. Mean Silhouette coefficient should also be as high as possible.

iv) Time: The execution time (in seconds), which directly indicates the computational complexity and should be as small as possible.

Table 2. Comparison of User Inputs and/or *Prior* Assumptions between Different Algorithms

Algorithm	**Free Parameter(s)**	***Prior* Assumption**	**Parameter Setting**
ADP	none	none	no need
MS	*i)* bandwidth, *p* *ii)* kernel function type	Gaussian distribution	*i)* *p*= 0.15 [20] *ii)* Gaussian kernel
SUB	initial cluster radius, *r*	Gaussian distribution	*r*= 0.3 [18]
DBS	*i)* cluster radius, *r* *ii)* minimum number of data samples within the radius, *p*	Gaussian distribution	*i)* the value of the knee point of the sorted *p*-dist graph *ii)* *p*=4 [21]
SOM	net size	none	12×12 [42]

ELM	initial cluster radius, *r*	Gaussian distribution	*r*=0.15 [20]
DP	*i)* minimum distance, *ρ* *ii)* local density value, *δ*	Gaussian distribution	*i)* relatively high, *ρ* *ii)* high, *δ* [43]
NMM	*i) prior* scaling parameter *ii)* kappa coefficient	Gaussian distribution	predefined [11] [a]
NMI	grid size	Gaussian distribution	predefined [36]
CEDS	*i)* microCluster radius, *r* *ii)* decay factor, *ω* *iii)* min microCluster threshold, *φ*	none	*i) r*= 0.15 *ii) ω*=500 *iii) φ*=1 [29]

[a] The parameters are fixed in advance

The quality measures of performance of the algorithms in terms of C, CH, SI and time based on the benchmark datasets listed in Table 1 are tabulated in Table 3, where we further bold the top three of each performance measure in the experiments for visual clarity.

Ideally, the number of data clouds/clusters generated should be equal or close to the number of classes (ground truth) in the dataset. However, in real situations, especially in the cases of large-scale and/or high dimensional datasets, data samples from different classes are more often mixed with each other, and data samples from the same class may spread into different locations far away from each other in the data space. The best way to deal with such datasets is to partition/cluster the data samples into smaller data clouds/clusters, and combine them later. Therefore, in this paper, we only require the numbers of data clouds/clusters in the clustering results to be close to the ground truth but larger than the number of classes, but not excessively large [28]. Therefore, we consider the clustering result with C meeting the following condition as a valid one:

$$\mathrm{N_C} \le \mathrm{C} \le \mathrm{N_S}/a \tag{22}$$

In this paper, we consider $a=3$ as a generic value that is not user- or problem-dependent. The rationale is that it indicates an extreme case when each cluster, on average, has only <3 members. In this case, the clustering/partitioning result provides too many trivial clusters and is not understandable for users. If $\mathrm{C} \le \mathrm{N_C}$, it implies that the clustering algorithm fails to separate the data samples from different classes.

Therefore, we additionally add an extra column titled “V” to Table 3 indicating the Validity of the clustering results, where “√” denotes that the clustering result is valid, “×” denotes the invalid one.

Table 3. Numerical Experiment Results on the Numerical Datasets

	Algorithm	**C**	**CH**	**SI**	**Time**	**V**
PI	ADP-o [a]	21	**601.6263**	**0.4673**	**0.16**	√
	ADP-e [b]	22	**572.2715**	0.3470	**0.18**	√
	MS	474			0.31	×
	SUB	9	256.7434	0.1002	0.52	√
	DBS	4	51.9112	-0.0117	**0.06**	√
	SOM	144	310.3476	0.3654	3.15	×
	ELM	14	31.8829	-0.3338	0.91	√
	DP	3	**294.9684**	0.2895	1.69	√
	NMM	3	243.4787	0.2416	107.04	√
	NMI	7	278.691	**0.5322**	6.95	√
	CEDS	768			0.78	×
BA	ADP-o	28	**1128.1432**	**0.5616**	**0.19**	√
	ADP-e	27	932.6723	0.4937	0.28	√
	MS	24	488.2148	0.3650	**0.09**	√
	SUB	14	**1068.8898**	**0.5311**	0.99	√

	DBS	48	352.5907	0.2138	**0.18**	√
	SOM	144	**1125.6221**	**0.5828**	4.31	×
	ELM	1	/	/	54.69	×
	DP	2	723.7514	0.3877	2.57	√
	NMM	4	787.2835	0.3718	170.46	√
	NMI	20	690.5714	0.4894	6.74	√
	CEDS	120	163.6131	0.1709	5.67	×
S1	ADP-o	15	**22675.2540**	**0.8803**	**1.09**	√
	ADP-e	84	6178.3721	**0.5949**	**0.99**	√
	MS	13	/	/	0.03	×
	SUB	10	/	/	2.62	×
	DBS	32	**14877.9431**	**0.5851**	**2.24**	√
	SOM	144	**14891.8742**	0.5412	11.88	√
	ELM	1	/	/	0.78	×
	DP	2	/	/	4.08	×
	NMM	6	/	/	814.18	×
	NMI	6	/	/	15.44	×
	CEDS	21	1285.7427	0.2913	11.85	×
S2	ADP-o	18	**12109.9581**	**0.7609**	**1.05**	√
	ADP-e	65	**4813.5951**	**0.5554**	**0.97**	√
	MS	13	/	/	0.05	×
	SUB	10	/	/	2.39	×
	DBS	35	3406.5872	0.2992	**2.22**	√
	SOM	144	**9575.3768**	**0.5215**	11.47	√
	ELM	1	/	/	0.77	×
	DP	2	/	/	5.11	×
	NMM	9	/	/	958.76	×
	NMI	4	/	/	16.81	×
	CEDS	26	1324.1078	0.2399	12.96	√
CA	ADP-o	71	**393.01630**	**0.3699**	**0.47**	√
	ADP-e	45	**447.0620**	0.3088	**0.46**	√
	MS	3	41.7123	0.3353	**0.18**	√
	SUB	73	235.8645	0.1459	5.77	√
	DBS	11	24.3466	0.0377	0.52	√
	SOM	144	330.5136	**0.3361**	7.12	×
	ELM	2	/	/	0.48	×
	DP	2	/	/	2.41	×
	NMM	3	**445.4341**	**0.4161**	257.86	√
	NMI	175	272.1408	0.1288	40.35	×
	CEDS	77	272.1408	0.1288	6.55	√
PB	ADP-o	79	**1057.9771**	**0.3821**	**6.22**	√
	ADP-e	92	**967.5478**	**0.3206**	**2.22**	√
	MS	8493	/	/	156.55	×
	SUB	187	382.6055	0.0113	82.98	√
	DBS	38	385.3319	-0.0780	**12.95**	√
	SOM	144	864.3771	0.2954	48.27	√
	ELM	9	/	/	16.78	×
	DP	3	/	/	12.53	×
	NMM	41	**980.6707**	**0.3190**	7883.89	√
	NMI	4316	/	/	2331.22	×
	CEDS	1	/	/	2466.47	×
ST	ADP-o	13	**15349.4503**	**0.8507**	**0.28**	√

	ADP-e	34	11745.7686	0.6613	**0.35**	√
	MS	1			0.78	×
	SUB	1			0.85	×
	DBS	14	3910.0867	**0.7920**	**0.34**	√
	SOM	144	**30043.6256**	0.4979	7.84	×
	ELM	1553			1.43	×
	DP	2			2.48	×
	NMM	2			77.99	×
	NMI	6			12.35	×
	CEDS	18	**24574.0909**	**0.7859**	7.06	×
MF	ADP-o	63	**618.6775**	**0.3474**	**2.96**	√
	ADP-e	78	**414.2603**	**0.2513**	**1.19**	√
	MS	4			2.59	×
	SUB	1994			282.27	×
	DBS	5			1.51	×
	SOM	144	**422.4845**	**0.2598**	233.13	×
	ELM	1			0.48	×
	DP	2			4.28	×
	NMM	1			722.11	×
	NMI	1			1892.13	×
	CEDS	90	229.8913	-0.0274	**47.97**	√
OD	ADP-o	18	**34653.4935**	**0.7608**	18.11	√
	ADP-e	131	**21530.3617**	0.3573	**4.03**	√
	MS	20	4291.6125	0.5733	**0.10**	√
	SUB	9	19878.6811	0.3408	**16.35**	√
	DBS	208	1995.0598	-0.6614	36.73	√
	SOM	144	**81402.2929**	0.5776	51.05	√
	ELM	1			1.17	×
	DP	2	5495.9202	**0.6418**	48.45	√
	NMM	4	8017.4665	0.5216	2617.14	√
	NMI	15	10922.5114	**0.7368**	396.67	√
	CEDS	13	1555.1093	0.0898	42.22	√
MA	ADP-o	47	**1430.4657**	**0.4120**	**17.68**	√
	ADP-e	380	643.6832	0.2081	**3.86**	√
	MS	1472	16.2135	-0.4401	48.58	×
	SUB	8	**1730.7881**	-0.1783	31.54	√
	DBS	15	17.8876	-0.4656	37.15	√
	SOM	144	1257.2603	**0.2344**	72.12	√
	ELM	25	334.6816	-0.0155	**28.55**	√
	DP	1			44.68	×
	NMM	4	**2381.0536**	**0.5746**	2486.65	√
	NMI	1578	19.4133	-0.1805	6050.48	×
	CEDS	54	406.1227	-0.2991	5976.02	√
LE	ADP-o	235	433.4874	**0.3045**	**21.99**	√
	ADP-e	242	414.5848	**0.2793**	**4.28**	√
	MS	7620			224.92	×
	SUB	153	471.2221	0.2026	154.50	√
	DBS	51	94.7283	-0.3547	**42.10**	√
	SOM	144	**622.6495**	**0.2897**	99.38	√
	ELM	9			17.55	×
	DP	2			53.55	×
	NMM	52	**481.4270**	0.0237	15682.64	√

NMI	14526			5768.78	×
CEDS	43	**569.9774**	0.0376	13109.01	√

[a] The ADP algorithm-offline version; [b] The ADP algorithm-evolving version.

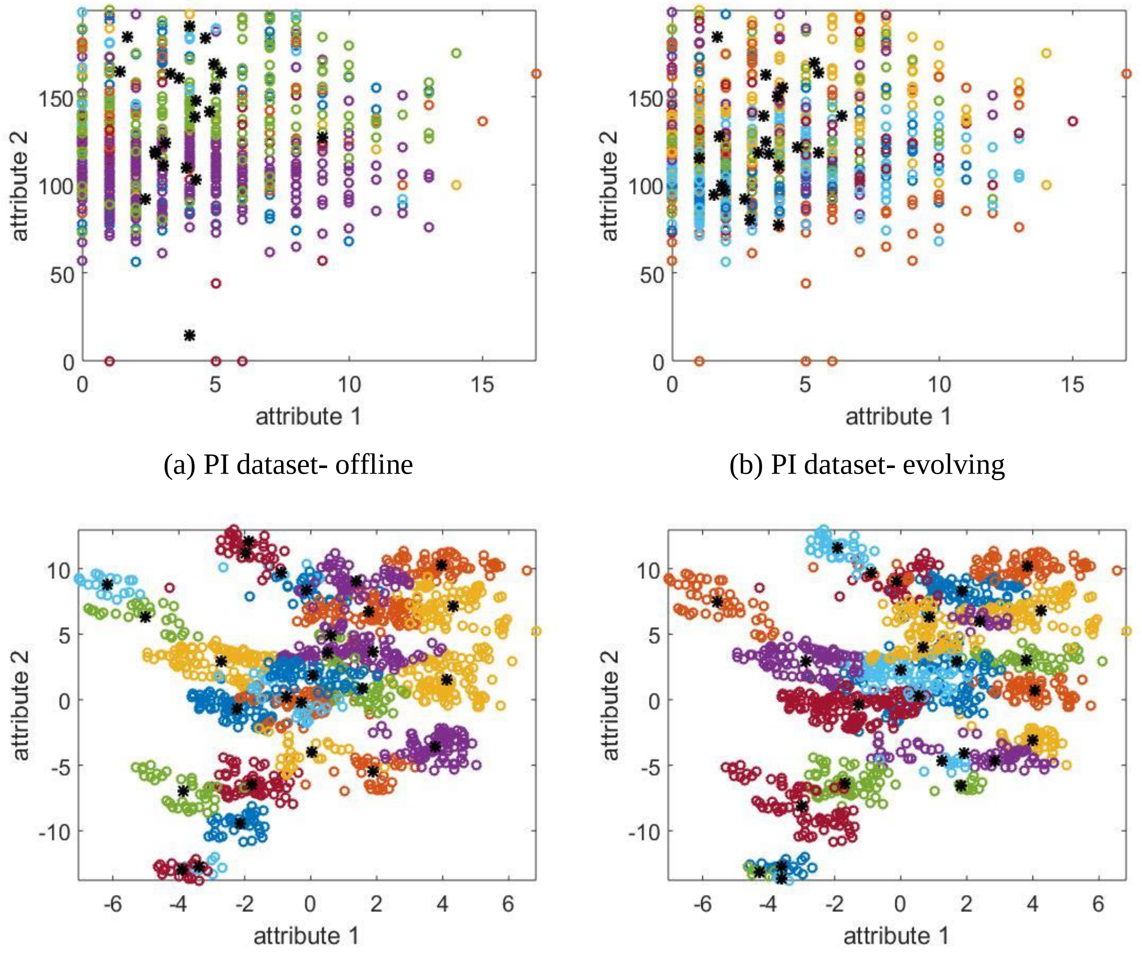
In the experiments, for the high dimensional datasets (N>20, N=N$_A$), we normalize the data via the following equation, which converts the Euclidean distance between data samples into a cosine dissimilarity [28]:

$$\boldsymbol{x}_{normalized} = \boldsymbol{x} / \|\boldsymbol{x}\| \tag{23}$$

We apply the following feature re-scaling operation on the low dimensional datasets (N<20) for the MS , ELM and CEDS algorithms [20],[29]:

$$x^i_{normalized} = \frac{x^i - x^i_{\min}}{x^i_{\max} - x^i_{\min}} \tag{24}$$

where i denotes the i^{th} dimension of the data, $i = 1, 2, ..., N$; $x^i_{\max}$ and $x^i_{\min}$ are the maximum and minimum values of the i^{th} attribute of the data. Note that, the Calinski Harabasz indexes of the results obtained by these two algorithms are calculated based on de-normalized results.



(a) PI dataset- offline

(b) PI dataset- evolving

(c) BA dataset- offline

(d) BA dataset- evolving

Fig. 5. The data partitioning results ("o" in different colours denote data samples of different data clouds, "*" denote local modes)

The partitioning results (both offline and evolving) of the PI and BA datasets are presented in Fig. 5 as additional illustrations; here only the first two dimensions of the results are presented for visual clarity.

5.2 Experiments on Image Datasets

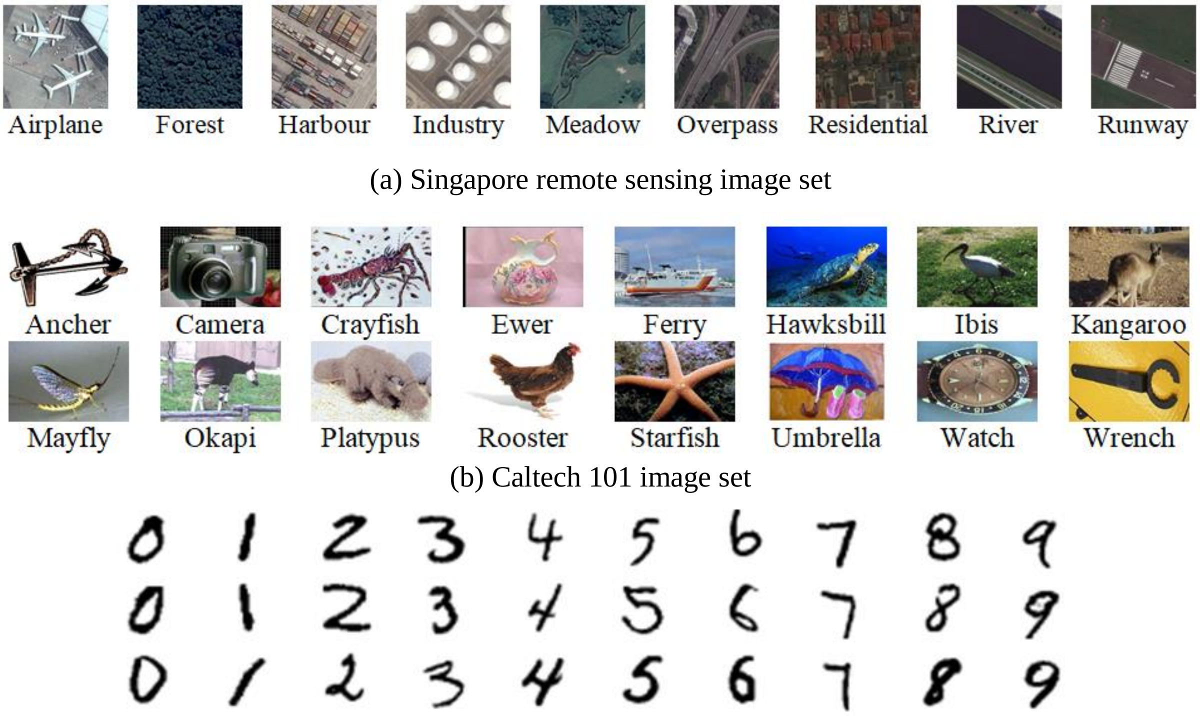
As it was stated in section 1, clustering techniques are widely used in image analysis. In this subsection, we also conduct several experiments on image clustering. The details of the benchmark image sets used in this subsection are tabulated in Table 4.

Singapore image set [26] is a recently introduced benchmark dataset for remote sensing scene classification. Caltech 101 image set [22] is widely used as a benchmark for object recognition. MNIST image set [35] is the most widely used large scale dataset for handwritten digits recognition. Examples of these images are given in Fig. 6.

Table 4. Details of the Benchmark Image Sets for Evaluation

Abbreviation	Image Set	R[a]	N_A	N_S	N_C
SIG	Singapore [26]	256×256×3	4096	1086	9
CAL	Caltech 101 [22]	Roughly 300×200×3	4096	9144	102
MNI	MNIST [35]	28×28×1	784	70000	10

[a] Resolution.



(a) Singapore remote sensing image set

(b) Caltech 101 image set

(c) MNIST image set

Fig. 6. Illustrative examples of the images used in the experiments

Images, especially, the large-size RGB ones, are significantly unstructured by their nature compared with numerical data. For example, pixels themselves do not have a spatial meaning. Instead of clustering the images based on the values of the pixels directly, one commonly used approach is to extract global features from the images and conduct image clustering based on the global features [7].

Therefore, in this paper, for the Singapore and Caltech 101 image sets, we use the 1×4096 dimensional activations of the pre-trained VGG-VD-16 convolutional neural network [46] from the first fully connected layer [52] as the feature vectors of the images for clustering. The clustering results of the two image sets produced by the ten clustering algorithms based on the respective feature vectors are tabulated in Table 5. In the experiments of this subsection, equation (23) is used to normalize the feature vectors due to the very high dimensionality of the problems.

Due to the wide variety of semantic contents and complex textural information contained in the images, separating images of different classes is very difficult. It is of great importance for a clustering/partitioning algorithm to be able to demonstrate strong separation ability. Therefore, for the image clustering problems, we involve an additional clustering quality measure, Purity (PU), which is calculated based on the result and the ground truth indicating the separation ability [20]:

$$\mathrm{PU} = \sum_{i=1}^{C} S_D^i \Big/ K \tag{25}$$

where S_D^i is the number of data samples with the dominant class label in the i^{th} cluster. The higher purity the clustering result has, the stronger separation ability the clustering algorithm exhibits.

Table 5. Image Clustering based on Feature Vectors

	Algorithm	**C**	**CH**	**SI**	**PU**	**Time**	**V**
SIG	ADP-o	161	**15.9752**	**0.1698**	**0.9871**	**3.54**	√
	ADP-e	97	**21.4609**	**0.1460**	**0.9678**	**18.20**	√
	MS	1085				19.33	×
	SUB	1086				198.26	×
	DBS	2				**0.95**	×
	SOM	144	**17.1072**	**0.1529**	**0.9853**	473.30	√
	ELM	804				126.70	×
	DP	2				3.93	×
	NMM	No result generated after 10 hours					×
	NMI	1086				17242.22	×
	CEDS	1086				400.17	×
CAL	ADP-o	1083	**12.3015**	**0.0785**	**0.8372**	**289.22**	√
	ADP-e	441	**24.6957**	**0.0567**	**0.8074**	**2362.12**	√
	MS	9094				2511.28	×
	SUB	9139				7770.47	×
	DBS	40				71.00	×
	SOM	144	**64.8208**	**0.1435**	**0.7730**	4630.92	√
	ELM	1110	3.3999	-0.1791	0.4389	**4205.27**	√
	DP	24				123.07	×
	NMM	No result generated after 10 hours					×
	NMI	No result generated after 10 hours					×
	CEDS	No result generated after 10 hours					×

For the MNIST image set, due to the much simpler structure and semantic contents of the handwritten digit images, we can conduct the image clustering by using the pixels directly. In the following experiment, we convert each image from a 28×28 pixel matrix into a 1×784 pixel vector, and use the pixel vectors as the input to the clustering algorithms. The clustering results on the MNIST image set are tabulated in Table 6. However, as both the cardinality and dimensionality of this image set are very high, the computation- and memory-efficiency of the offline and incremental algorithms deteriorate dramatically due to the iterative learning process. Therefore, in this experiment, we only involve the clustering/data partitioning algorithms, which are non-iterative and “one pass”, namely, the evolving version of ADP, ELM and CEDS.

Table 6. Image Clustering based on Pixel Values

	Algorithm	**C**	**CH**	**SI**	**PU**	**Time**	**V**
MNI	ADP-e	4569	**31.7081**	**0.0636**	**0.9547**	**9603.45**	√
	ELM	3				18.11	×
	CEDS	No result generated after 10 hours					×

6. Analysis and Discussion

In this section, we will analyse the performance of the proposed algorithm, and compare it with 8 other well-known algorithms based on the numerical examples presented in section IV.

i) MS algorithm [19]

MS algorithm is very fast when the scale and dimensionality of the dataset is low. However, its calculation speed decreases quickly in processing large scale and high dimensional datasets. The quality of its clustering results varies dramatically. Without *prior* knowledge, it produced invalid clustering results on many datasets. This is due to its gradient nature making it highly dependent on initial guess and being prone to fall into local minima [19].

ii) SUB algorithm [18]

The calculation efficiency of the SUB algorithm [18] is also largely dependent on the scale and dimensionality of the dataset. It is very inefficient with large-scale and high-dimensional datasets. The clustering results also vary a lot. This algorithm is able to perform high quality clustering on low-dimensional and large-scale datasets. In other cases, however, it failed to give valid and/or useful clustering results.

iii) DBS algorithm [21]

DBS algorithm [21] is an efficient incremental online algorithm. The results it produced generally contain smaller number of clusters. However, the quality of its clustering results is very low. One may also notice that, DBScan algorithm is not effective in handling high-dimensional and large-scale datasets.

iv) SOM algorithm [34]

SOM algorithm [34] requires the size of its net to be pre-fixed and, thus, always produces results with the same number of clusters. The pre-fixed net size enabled the algorithm to perform high quality clustering on high- dimensional and large-scale datasets. However, its calculation efficiency is much lower and it failed to give useful clustering results on small-size datasets.

v) ELM algorithm [20]

ELM algorithm [20] was exhibiting high quality clustering performance on small-scale datasets. However, it did not give any useful clustering results on low-dimensional and large-scale datasets. In complex problems, the algorithm failed to separate the data samples of different classes.

vi) DP algorithm [43]

DP algorithm [43] does not require any *prior* knowledge in advance, however, during the operation, users need to make choices based on a decision graph generated from the data, namely, to choose one of the rectangles corresponding to the proper minimum distance between centres and local density value. Different choices can lead to very different results and there could be thousands of rectangles generated from datasets with huge size and high dimensionality, which makes it impossible for users to select.

Moreover, using the recommended selection, the algorithm failed to separate the data samples of different classes for high-dimensional and large-scale datasets.

vii) NMM algorithm [11]

NMM algorithm [11] is one of the so-called “nonparametric” algorithms in the comparison despite having a number of pre-defined parameters and coefficients.

This algorithm is based on the *prior* assumption of data distribution being Gaussian. This algorithm can give very good clustering results on datasets with Gaussian or similar distributions. However, this algorithm failed to give useful clustering results on many other datasets. In addition, its computation efficiency is low.

viii) NMI algorithm [36]

NMI algorithm [36] is also a so-called "nonparametric" approach. Similarly to the mixture model clustering algorithm [11], it has a number of pre-defined parameters, i.e. grid size, interval between two grids, and it assumes that the data has a Gaussian distribution. This algorithm is very accurate when the datasets are small and the structure is simple.

However, it provided invalid results in processing large-scale and high dimensional datasets. Moreover, its computation efficiency is also largely influenced by the size and dimensionality of the data.

ix) CEDS algorithm [29]

CEDS [29] is a recently introduced algorithm for streaming data clustering. This algorithm can follow the changing data pattern of the data stream and group the samples into arbitrary shaped clusters. Nonetheless, based on the recommended experimental settings, this algorithm is only effective on lower dimensional and/or smaller size datasets, and it frequently fails on complex problems. Its computation efficiency is also very low on large-scale problems.

x) ADP algorithm

As a real algorithm that is free from user- and problem- specific parameters, the proposed ADP algorithm is able to consistently provide high quality clustering results without any user inputs or *prior* assumptions. Both versions (offline and evolving) are highly efficient computationally and they are very effective in handling datasets with different scale and dimensionality. The indexes measured from the clustering results are all highly ranked compared with the other 9 comparative algorithms used for comparison. From the rank in Tables 3, 5 and 6 one can see that the proposed algorithm is always ranked in the top 3. The computation time is also within the fastest. It is critically important to notice that the proposed algorithm is autonomous, user- and problem- parameter free. In addition, it can evolve its structure to follow the changing data pattern, while others are not. In addition, from the numerical examples one can notice that, the proposed algorithm exhibits even higher effectiveness and efficiency in handling large-scale and higher dimensional datasets compared with other algorithms.

The strong performance of the proposed ADP algorithm comes from a fundamentally different data processing approach based on rank operators. Rank operators are normally avoided in clustering because they are non-linear operators, and so most clustering algorithms prefer the linear mean operator. We believe that the specificity of the rank operator plays a central role in the creation of more parsimonious partitions, specifically when augmented with local mode definitions that are parameter free. Although computationally more demanding, on line rank updates are still practical as we show in our work. For the offline version, the ADP algorithm identifies prototypes from the data samples based on their ranks in terms of the data densities and mutual distances instead of the commonly used means and variances, and use the prototypes to aggregate data samples around them forming Voronoi tessellations [41]. For the evolving version of the ADP algorithm, it has a more flexible evolving structure compared with other online approaches due to its prototype-based nature. In addition, it replaces the pre-defined threshold, which is commonly used in other online approaches, with a dynamically changing threshold derived from the data. Therefore, the ADP algorithm is able to obtain a more stable, effective and objective partitioning compared with other approaches.

However, we have to admit that, although the data partitioning results obtained by the offline version of the ADP algorithm are not influenced by the order of the input data because of the ranking operations, there is no guarantee that the partitioning results can converge to a locally optimal solution because of other greedy steps in the algorithmic procedure (multiple peaks, etc.). Alternatively, there is no known iterative process for minimizing an objective function involved in the ADP algorithm (in both the offline and evolving versions). Nonetheless, by involving an iterative process to minimizing the objective function in a similar way as described in [45], the ADP algorithm can also converge to the locally optimal partitions, but this is out of the scope of this paper.

7. Conclusion and Future Work

In this paper, a novel algorithm for data partitioning, named autonomous data partition (ADP), was introduced. Both, the offline and evolving versions of the ADP algorithm are entirely data-driven, autonomous and require no user- and problem- specific input. Using nonparametric operators, the proposed algorithm is able to identify the local modes representing the local maxima of the density based on the empirically observed data samples. It partitions the data space into the shape- and parameter-free data clouds. Compared with the well-known algorithms, the proposed approach has the following significant advantages:

i) It is free from *prior* assumptions and user- and problem- specific parameters;

ii) It is able to conduct high-quality clustering in a short time without the need of *prior* knowledge.

Numerical experiments conducted with the benchmark datasets demonstrate the validity of the proposed algorithm and also show its advantages compared with the alternative well-known algorithms. Moreover, the advantages of the ADP algorithm are even more pronounced on larger size, higher dimensional, complex problems.

As future work, we will study the local optimality and convergence of the ADP algorithm. We will also apply the proposed algorithm to more complex problems, i.e. remote sensing scene image analysis, high frequency trading, etc., and study the underlying data patterns behind them.

Appendix

		Web Link
Codes	ADP Algorithm	http://empiricaldataanalytics.org/downloads.html
	Pre-trained VGG-VD-16 convolutional neural network [46]	http://www.vlfeat.org/matconvnet/pretrained/
Datasets	PIMA [47]	https://archive.ics.uci.edu/ml/datasets/pima+indians+diabetes
	Banknote Authentication [37]	https://archive.ics.uci.edu/ml/datasets/banknote+authentication
	S1 [23]	http://cs.joensuu.fi/sipu/datasets/
	S2 [23]	http://cs.joensuu.fi/sipu/datasets/
	Cardiotocography [9]	https://archive.ics.uci.edu/ml/datasets/cardiotocography
	Pen-Based Handwritten Digits Recognition [2]	https://archive.ics.uci.edu/ml/datasets/Pen-Based+Recognition+of+Handwritten+Digits
	Steel Plates Faults [14]	http://archive.ics.uci.edu/ml/datasets/steel+plates+faults
	Multiple Features [30]	https://archive.ics.uci.edu/ml/datasets/Multiple+Features
	Occupancy Detection [16]	https://archive.ics.uci.edu/ml/datasets/Occupancy+Detection+
	MAGIC Gamma Telescope [13]	https://archive.ics.uci.edu/ml/datasets/magic+gamma+telescope
	Letter Recognition [24]	https://archive.ics.uci.edu/ml/datasets/letter+recognition
	Singapore [26]	http://icn.bjtu.edu.cn/Visint/resources/Scenesig.aspx
	Caltech 101 [22]	http://www.vision.caltech.edu/Image_Datasets/Caltech101/
	MNIST [35]	http://yann.lecun.com/exdb/mnist/

Acknowledgment

This work was partially supported by The Royal Society grant IE141329/2014 “Novel Machine Learning Paradigms to address Big Data Streams”.

Reference

[1] S. Aeberhard, D. Coomans, and O. de Vel, “Comparison of classifiers in high dimensional settings,” *Dept. Math. Statist., James Cook Univ., North Queensland, Australia, Tech. Rep 92-02*, 1992.

[2] F. Alimoglu and E. Alpaydin, “Methods of combining multiple classifiers based on different representations for pen-based handwritten digit recognition,” in *Proceedings of the Fifth Turkish Artificial Intelligence and Artificial Neural Networks Symposium*, 1996, pp. 1–8.

[3] P. Angelov and R. Yager, “A new type of simplified fuzzy rule-based system,” *Int. J. Gen. Syst.*, vol. 41, no. 2, pp. 163–185, 2011.

[4] P. Angelov, Autonomous learning systems: from data streams to knowledge in real time. John Wiley & Sons, Ltd., 2012.

[5] P. Angelov, “Outside the box: an alternative data analytics framework,” *J. Autom. Mob. Robot. Intell. Syst.*, vol. 8, no. 2, pp. 53–59, 2014.

[6] P. P. Angelov, X. Gu, J. Principe, and D. Kangin, “Empirical data analysis - a new tool for data analytics,” in *Proceedings of IEEE International Conference on Systems, Man, and Cybernetics*, 2016, pp. 53–59.

[7] P. Angelov and P. Sadeghi-Tehran, “Look-a-Like: A Fast Content-Based Image Retrieval Approach Using a Hierarchically Nested Dynamically Evolving Image Clouds and Recursive Local Data Density,” *Int. J. Intell. Syst.*, vol. 32, no. 1, pp. 82–103, 2016.

[8] P. Angelov, X. Gu, and D. Kangin, “Empirical data analytics,” *Int. J. Intell. Syst.*, vol. 32, no. 12, pp. 1261–1284, 2017.

[9] D. Ayres-de-Campos, J. Bernardes, A. Garrido, J. Marques-de-Sa, and L. Pereira-Leite, “SisPorto 2.0: A program for automated analysis of cardiotocograms,” *J. Matern. Fetal. Med.*, vol. 9, no. 5, pp. 311–318, 2000.

[10] J. C. Bezdek, R. Ehrlich, and W. Full, “FCM: The fuzzy c-means clustering algorithm,” *Comput. Geosci.*, vol. 10, no. 2–3, pp. 191–203, 1984.

[11] [D. M. Blei and M. I. Jordan, “Variational inference for Dirichlet process mixtures,” *Bayesian Anal.*, vol. 1, no. 1 A, pp. 121–144, 2006.

[12] D. Birant and A. Kut, “ST-DBSCAN: An algorithm for clustering spatial-temporal data,” *Data Knowl. Eng.*, vol. 60, no. 1, pp. 208–221, 2007.

[13] R. K. Bock, A. Chilingarian, M. Gaug, F. Hakl, T. Hengstebeck, M. Jiřina, J. Klaschka, E. Kotrč, P. Savický, S. Towers, A. Vaiciulis, and W. Wittek, “Methods for multidimensional event classification: A case study using images from a Cherenkov gamma-ray telescope,” *Nucl. Instruments Methods Phys. Res. Sect. A Accel. Spectrometers, Detect. Assoc. Equip.*, vol. 516, no. 2–3, pp. 511–528, 2004.

[14] M. Buscema, “Metanet*: The theory of independent judges.,” *Subst. Use Misuse*, vol. 33, no. 2, pp. 439–461, 1998.

[15] T. Caliński and J. Harabasz, “A dendrite method for cluster analysis,” *Commun. Stat. Methods*, vol. 3, no. 1, pp. 1–27, 1974.

[16] L. M. Candanedo and V. Feldheim, “Accurate occupancy detection of an office room from light, temperature, humidity and CO2 measurements using statistical learning models,” *Energy Build.*, vol. 112, pp. 28–39, 2016.

[17] Y. Cheng, “Mean Shift, Mode Seeking, and Clustering,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 17, no. 8, pp. 790–799, 1995.

[18] S. L. Chiu, “Fuzzy model identification based on cluster estimation.,” *Journal of intelligent and Fuzzy systems*, vol. 2, no. 3. pp. 267–278, 1994.

[19] D. Comaniciu and P. Meer, “Mean shift: A robust approach toward feature space analysis,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 5, pp. 603–619, 2002.

[20] R. Dutta Baruah and P. Angelov, “Evolving local means method for clustering of streaming data,” *IEEE Int. Conf. Fuzzy Syst*., pp. 10–15, 2012.

[21] M. Ester, H. P. Kriegel, J. Sander, and X. Xu, “A density-based algorithm for discovering clusters in large spatial databases with noise,” in *Proceedings of International Conference on Knowledge Discovery and Data Mining*, 1996, vol. 96, pp. 226–231.

[22] L. Fei-Fei, R. Fergus, and P. Perona, “Learning generative visual models from few training examples: an incremental Bayesian approach tested on 101 object categories,” *Comput. Vis. Image Underst.*, vol. 106, no. 1, pp. 59–70, 2007.

[23] P. Fränti and O. Virmajoki, “Iterative shrinking method for clustering problems,” *Pattern Recognit.*, vol. 39, no. 5, pp. 761–775, 2006.

[24] P. W. Frey and D. J. Slate, “Letter recognition using Holland-style adaptive classifiers,” *Mach. Learn.*, vol. 6, no. 2, pp. 161–182, 1991.

[25] B. J. Frey and D. Dueck, “Clustering by passing messages between data points,” *Science*, vol. 315, no. 5814, pp. 972–976, 2007.

[26] J. Gan, Q. Li, Z. Zhang, and J. Wang, “Two-level feature representation for aerial scene classification,” *IEEE Geosci. Remote Sens. Lett.*, vol. 13, no. 11, pp. 1626–1630, 2016.

[27] S. Ghosh, L. Sigal, and E. B. Sudderth, “Nonparametric Clustering with Distance Dependent Hierarchies,” in *UAI*, 2014, pp. 260–269.

[28] X. Gu, P. Angelov, D. Kangin, and J. Principe, “Self-organised direction aware data partitioning algorithm,” *Inf. Sci. (Ny).*, vol. 423, pp. 80–95, 2018.

[29] R. Hyde, P. Angelov, and A. R. MacKenzie, “Fully online clustering of evolving data streams into arbitrarily shaped clusters,” *Inf. Sci. (Ny).*, vol. 382–383, pp. 96–114, 2017.

[30] A. K. Jain, R. P. W. Duin, and J. Mao, “Statistical pattern recognition: a review,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 1, pp. 4–37, 2000.

[31] R. Jenssen, D. Erdogmus, K. E. Hild, J. C. Principe, and T. Eltoft, “Information cut for clustering using a gradient descent approach,” *Pattern Recognit.*, vol. 40, no. 3, pp. 796–806, 2007.

[32] S. C. Johnson, “Hierarchical clustering schemes,” *Psychometrika*, vol. 32, no. 3, pp. 241–254, 1967.

[33] S. Krinidis and V. Chatzis, “A robust fuzzy local information c-means clustering algorithm,” *IEEE Trans. Image Process.*, vol. 19, no. 5, pp. 1328–1337, 2010.

[34] T. Kohonen, Self-organizing maps. Berlin: Springer, 1997.

[35] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2323, 1998.

[36] J. Li, S. Ray, and B. G. Lindsay, “A nonparametric statistical approach to clustering via mode identification,” *J. Mach. Learn. Res.*, vol. 8, no. 8, pp. 1687–1723, 2007.

[37] V. Lohweg, J. L. Hoffmann, H. Dörksen, R. Hildebrand, E. Gillich, J. Hofmann, and J. Schaede, “Banknote authentication with mobile devices,” in *Media Watermarking, Security, and Forensics*, 2013, p. 866507.

[38] U. Von Luxburg, “A tutorial on spectral clustering,” *Stat. Comput.*, vol. 17, no. 4, pp. 395–416, 2007.

[39] J. B. MacQueen, “Some methods for classification and analysis of multivariate observations,” in *Proceedings of the Fifth Berkeley Symp. Math. Stat. Probab*, pp. 281–297, 1967.

[40] C. A. McGrory and D. M. Titterington, “Variational approximations in Bayesian model selection for finite mixture distributions,” *Comput. Stat. Data Anal.*, vol. 51, no. 11, pp. 5352–5367, 2007.

[41] A. Okabe, B. Boots, K. Sugihara, and S. N. Chiu, Spatial tessellations: concepts and applications of Voronoi diagrams, 2nd ed. Chichester, England: John Wiley & Sons., 1999.

[42] P. Płoński and K. Zaremba, “Self-organising maps for classification with metropolis-hastings algorithm for supervision,” in *Proceedings of International Conference on Neural Information Processing*, 2012, pp. 149–156.

[43] A. Rodriguez and A. Laio, “Clustering by fast search and find of density peaks,” *Science,* vol. 344, no. 6191, pp. 1493–1496, 2014.

[44] P. J. Rousseeuw, “Silhouettes: A graphical aid to the interpretation and validation of cluster analysis,” *J. Comput. Appl. Math.,* vol. 20, pp. 53–65, 1987.

[45] S. Z. Selim and M. A. Ismail, “K-means-type algorithms: a generalized convergence theorem and characterization of local optimality,” *IEEE Trans. Pattern Anal. Mach. Intell.,* vol. PAMI-6, no. 1, pp. 81–87, 1984.

[46] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in *Proceedings of International Conference on Learning Representations*, 2015, pp. 1–14.

[47] J. W. Smith, J. E. Everhart, W. C. Dickson, W. C. Knowler, and R. S. Johannes, “Using the ADAP learning algorithm to forecast the onset of diabetes mellitus,” in *Proceedings of Annual Symposium on Computer Application in Medical Care*, 1988, pp. 261–265.

[48] P. Tellaroli, M. Bazzi, M. Donato, A. R. Brazzale, and S. Drăghici, “Cross-clustering: a partial clustering algorithm with automatic estimation of the number of clusters,” *PLoS One,* vol. 11, no. 3, p. e0152333, 2016.

[49] I. Timón, J. Soto, H. Pérez-Sánchez, and J. M. Cecilia, “Parallel implementation of fuzzy minimals clustering algorithm,” *Expert Syst. Appl.,* vol. 48, pp. 35–41, 2016.

[50] V. Vapnik and R. Izmailov, Statistical Learning and Data Sciences, vol. 9047. 2015.

[51] C. Wang, S. Member, J. Lai, and D. Huang, “SVStream : A Support Vector Based Algorithm for Clustering Data Streams,” *IEEE Trans. Knowl. Data Eng.,* vol. 25, no. 1, pp. 1410–1424, 2011.

[52] G. S. Xia, J. Hu, F. Hu, B. Shi, X. Bai, Y. Zhong, and L. Zhang, “AID: A benchmark dataset for performance evaluation of aerial scene classification,” *IEEE Trans. Geosci. Remote Sens.,* vol. 55, no. 7, pp. 3965–3981, 2017.

[53] R. R. Yager and D. P. Filev, “Approximate clustering Via the mountain method,” *IEEE Trans. Syst. Man. Cybern.,* vol. 24, no. 8, pp. 1279–1284, 1994.

[54] M. S. Yang and Y. Nataliani, “Robust-learning fuzzy c-means clustering algorithm with unknown number of clusters,” *Pattern Recognit.,* vol. 71, pp. 45–59, 2017.