Knowledge-Based Systems xxx (2015) xxx–xxx

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

A tree-based incremental overlapping clustering method using the three-way decision theory

Hong Yu *, Cong Zhang, Guoyin Wang

Chongqing Key Laboratory of Computational Intelligence, Chongqing University of Posts and Telecommunications, Chongqing 400065, China

ARTICLE INFO

Article history:
Received 18 January 2015
Received in revised form 24 April 2015
Accepted 31 May 2015
Available online xxxx

Keywords:
Incremental clustering
Overlapping clustering
Search tree
Three-way decision theory

ABSTRACT

Existing clustering approaches are usually restricted to crisp clustering, where objects just belong to one cluster; meanwhile there are some applications where objects could belong to more than one cluster. In addition, existing clustering approaches usually analyze static datasets in which objects are kept unchanged after being processed; however many practical datasets are dynamically modified which means some previously learned patterns have to be updated accordingly. In this paper, we propose a new tree-based incremental overlapping clustering method using the three-way decision theory. The tree is constructed from representative points introduced by this paper, which can enhance the relevance of the search result. The overlapping cluster is represented by the three-way decision with interval sets, and the three-way decision strategies are designed to updating the clustering when the data increases. Furthermore, the proposed method can determine the number of clusters during the processing. The experimental results show that it can identifies clusters of arbitrary shapes and does not sacrifice the computing time, and more results of comparison experiments show that the performance of proposed method is better than the compared algorithms in most of cases.

1. Introduction

Most of existing clustering algorithms usually analyze static datasets in which objects are kept unchanged after being processed [1,2]. However, in many practical applications, the datasets are dynamically modified which means some previously learned patterns have to be updated accordingly [3,4]. Although these approaches have been successfully applied, there are some situations in which a richer model is needed for representing a cluster [5,6]. For example, a researcher may collaborate with other researchers in different fields, therefore, if we cluster the researchers according to their interested areas, it could be expected that some researchers belong to more than one cluster. In these areas, overlapping clustering is useful and important as well as incremental clustering.

For this reason, the problem of incremental overlapping clustering is addressed in this paper. The main contribution of this work is an incremental overlapping clustering detection method, called TIOC-TWD (Tree-based Incremental Overlapping Clustering method using the Three-Way Decision theory). The proposed method introduces a new incremental clustering framework with three-way decision using interval sets and a new searching tree based on representative points, which together allows to obtain overlapping clusters when data increases. Besides, the TIOC-TWD introduces new three-way strategies to update efficiently the clustering after multiple objects increases. Furthermore, the proposed method can dynamically determine the number of clusters, and it does not need to define the number of cluster in advance. The above characteristics make the TIOC-TWD appropriate for handling overlapping clustering in applications where the data is increasing.

The experimental results show that the proposed method not only can identify clusters of arbitrary shapes, but also can merge small clusters into the big one when the data changes; the proposed method can detect new clusters which might be the result of splitting or new patterns. Besides, more experimental results show that the performance of proposed method is better than the compared algorithms in most of cases. We note that a short version of this work had been appeared in the RSCTC-2014 Workshop on the Three-Way Decisions and Probabilistic Rough Sets [7].

2. Related work

Nowadays, there are some achievements on the incremental clustering approaches. Ester et al. [8] put forward the IncDBSCAN

* Corresponding author. Tel.: +86 13617676007.
E-mail addresses: yuhong@cqupt.edu.cn (H. Yu), zhangcong0214@163.com (C. Zhang), wanggy@cqupt.edu.cn (G. Wang).

http://dx.doi.org/10.1016/j.knosys.2015.05.028
0950-7051/

clustering algorithm based on the DBSCAN. After that, Goyal et al. [9] proposed the derivation work which is more efficient than the IncDBSCAN because it is capable of adding points in bulk to existing set of clusters. Patra et al. [10] proposed an incremental clustering algorithm based on distance and leaders, but the algorithm needs to search the whole data space to find the surrounding leaders. Ibrahim et al. [11] proposed an incremental clustering algorithm which can maximize the relatedness of distances between patterns of the same cluster. Ning et al. [12] proposed an incremental spectral clustering approach by efficiently updating the eigen-system, but it could not find the overlapping clusters. Pensa et al. [13] proposed an incremental hierarchical co-clustering approach, which computes a partition of objects and a partition of features simultaneously but it cannot find out the overlapping clusters.

Meanwhile, some approaches, addressing the problem of incremental overlapping clustering, have been reported. Hammouda and Kamel [14] proposed similarity histogram-based clustering method based on the concept of Histogram Ratio of a cluster. Gil-García and Pons-Porrata [15] proposed dynamic hierarchical compact method and dynamic hierarchical star method, these methods is time consuming due to the framework of hierarchical clustering. Pérez-Suárea et al. [16] proposed an algorithm based on density and compactness for dynamic overlapping clustering, but it builds a large number of small clusters. Lughofer [17] proposed dynamic split-and merge operations for evolving cluster models, which are learned incrementally but can only deal with crisp clustering. Labroche [18] proposed online fuzzy medoid based clustering algorithms, which are adapted to overlapping clusters but the number of clusters need to define in advance.

Therefore, the main objective of this paper is to propose an approach that combine both processing of incremental data and obtaining of overlapping clusters. For this kind of problem, some scholars had pointed out that the clustering approaches to combine with rough sets are impactful [19]. Thus, Parmar et al. [20] proposed an algorithm for clustering categorical data using rough set theory; Chen et al. [21,22] researched the incremental data mining with rough set theory; Peters et al. [23] proposed the dynamic rough clustering; and Lingras et al. [24] reviewed fuzzy and rough approaches for soft clustering.

Further, the three-way decision with interval sets provides an ideal mechanism to represent overlapping clustering. The concept of three-way decisions was developed with researching the rough set theory [25]. A theory of three-way decision is constructed based on the notions of acceptance, rejection and noncommitment, and it is an extension of the commonly used binary-decision model with an added third option [26]. Three-way decision approaches have been successfully applied in decision systems [27–29], email spam filtering [30], three-way investment decisions [31,32], clustering analysis [33], and a number of other applications [25,34]. In our previous work [33], we had proposed a three-way decision strategy for overlapping clustering, where a cluster is described by an interval set. In fact, Lingras and Yan [35] had introduced the concept of interval sets to represent clusters, and Lingras and West [36] proposed an interval set clustering method with rough k-means for mining clusters of web visitors. Yao et al. [37] represented each cluster by an interval set instead of a single set as the representation of a cluster. Chen and Miao [38] described a clustering method by incorporating interval sets in the rough k-means.

In this paper, we propose the three-way decision clustering, which is applicable to crisp clustering as well as overlapping clustering. There are three relationships between an object and a cluster: (1) the object certainly belongs to the cluster, (2) the object certainly does not belong to the cluster, and (3) the object might or might not belong to the cluster. It is a typical three-way decision processing to decide the relationship between an object and a cluster. Objects in the lower bound are definitely part of the cluster, and only belong to that cluster; while objects between the two bounds are possibly part of that cluster and potentially belong to some other clusters.

Furthermore, in the field of incremental learning, it is common to learn from new incremental samples based on the existing results. The tree structures are particularly well suited for this task because they enable a simple and effective way to search and update. At the same time, trees are easy to store the learned patterns (results), which can save lots of duplicate learning time. Tree structures have been successfully used in some typical incremental learning approaches [39,40]. Therefore, this paper will use a tree to store the searching space, where a node of tree indicates the information corresponding to some representative points.

3. Description of the problem

3.1. Three-way decision clustering

To define our framework, let a universe be $U = \{\mathbf{x}_1, \ldots, \mathbf{x}_n, \ldots, \mathbf{x}_N\}$, and the resulting clustering scheme $\mathbf{C} = \{C_1, \ldots, C_k, \ldots, C_K\}$ is a family of clusters of the universe. The $\mathbf{x}_n$ is an object which has D attributes, namely, $\mathbf{x}_n = (x_n^1, \ldots, x_n^d, \ldots, x_n^D)$. The x_n^d denotes the value of the d-th attribute of the object $\mathbf{x}_n$, where $n \in \{1, \ldots, N\}$, and $d \in \{1, \ldots, D\}$.

We can look at the cluster analysis problem from a decision making perspective. For crisp clustering, it is a typical two-way decision; meanwhile for overlapping clustering or soft clustering, it is a type of three-way decision. Let's review some basic concepts of clustering using interval sets from our previous work [33]. In contrast to the general crisp representation of a cluster, where a cluster is a set of objects, we represent a cluster as an interval set. That is,

$$C_k = [\underline{C_k}, \overline{C_k}], \tag{1}$$

where $\underline{C_k}$ is the lower bound of the cluster C_k, $\overline{C_k}$ is the upper bound of the cluster C_k, and $\underline{C_k} \subseteq \overline{C_k}$.

Therefore, we can define a cluster by the following properties:

$$(i)\ \underline{C_k} \neq \emptyset, 0 < k \leqslant K; \quad (ii)\ \bigcup \overline{C_k} = U. \tag{2}$$

Property (i) implies that a cluster cannot be empty. This makes sure that a cluster is physically meaningful. Property (ii) states that any object of U must belong to the upper bound of a cluster, which ensures that every object is properly clustered.

With respect to the family of clusters, $\mathbf{C}$, we have the following family of clusters formulated by interval sets as:

$$\mathbf{C} = \{[\underline{C_1}, \overline{C_1}], \ldots, [\underline{C_k}, \overline{C_k}], \ldots, [\underline{C_K}, \overline{C_K}]\}. \tag{3}$$

Therefore, the sets $\underline{C_k}, \overline{C_k} - \underline{C_k}$ and $U - \overline{C_k}$ formed by certain decision rules constitute the three regions of the cluster C_k as the positive region, boundary region and negative region, respectively. The three-way decisions are given as:

$$\begin{aligned} POS(C_k) &= \underline{C_k}, \\ BND(C_k) &= \overline{C_k} - \underline{C_k}, \\ NEG(C_k) &= U - \overline{C_k}. \end{aligned} \tag{4}$$

Objects in $POS(C_k)$ definitely belong to the cluster C_k, objects in $NEG(C_k)$ definitely do not belong to the cluster C_k, and objects in the region $BND(C_k)$ might or might not belong to the cluster.

Any data mining technique needs to have a clear and precise evaluation measure. In clustering, evaluations such as the similarity between objects and compactness of clusters are appropriate indicators of quality of clusters. We will attempt to obtain the three regions by comparing these evaluation values with a pair of thresholds in the following work.

Under the representation, we can formulate the overlapping clustering and crisp clustering as follows. For a clustering, if there exists $k \neq t$, such that

$$\begin{aligned} &(1)\ POS(C_k) \cap POS(C_t) \neq \emptyset, or \\ &(2)\ BND(C_k) \cap BND(C_t) \neq \emptyset, or \\ &(3)\ POS(C_k) \cap BND(C_t) \neq \emptyset, or \\ &(4)\ BND(C_k) \cap POS(C_t) \neq \emptyset, \end{aligned} \tag{5}$$

we call it is a overlapping (soft) clustering; otherwise, it is a crisp (hard) clustering.

As long as one condition from Eq. (5) is satisfied, there must exist at least one object belonging to more than one cluster.

3.2. Incremental overlapping clustering

In this paper, we suppose such a scenario: the data we observed and collected are incremental, we have collected some data initially and then the data we collected are increasing as time passed. In order to save time and computational resources, we hope that we can adjust the clustering results obtained in the previous step according to new incremental data, rather than re-implement clustering algorithm on the whole dataset. Such a scenario often happens in real world, so it's an important problem in data mining.

Assume there is a given information system, $IS^t = (U^t, A^t)$, where U^t means the universe and A^t means the set of attributes. The clustering result is known as $\mathbf{C}^t = \{C_1^t, \ldots, C_i^t, \ldots, C_{|\mathbf{C}^t|}^t\}$, and the structure information of each cluster is known. The problem of incremental clustering is: for a given dataset U^t and the previous clustering result $\mathbf{C}^t$, how to compute $\mathbf{C}^{t+1} = \{C_1^{t+1}, \ldots, C_i^{t+1}, \ldots, C_{|\mathbf{C}^{t+1}|}^{t+1}\}$ efficiently and effectively according to the new arriving objects $\triangle U$. Sometimes, we also use U^{t+1} to denote $U^t \cup \triangle U$.

In order to represent overlapping clustering as well as incremental clustering, each cluster will be represented as three regions introduced in the last subsection.

4. The TIOC-TWD clustering method

The processing of the proposed TIOC-TWD method is illustrated in Fig. 1. In fact, we also devise an overlapping clustering algorithm using three-way decision strategy for the initial static data, which is based on a graph of representative points by calculating the similarity between representative regions. It is called Algorithm 1 and described in Section 4.2.

4.1. Related definitions

In a D-dimensions space, when considering a small enough region, the objects are usually well-distributed, thus we can use a fictional point called representative point to represent these objects.

Let $Distance(\mathbf{x}, \mathbf{y})$ be the distance between two objects; shorter the distance between $\mathbf{y}$ and $\mathbf{x}$, more similar they are. For a point $\mathbf{x}$, a distance threshold δ, we use $Neighbor(\mathbf{x})$ denotes the objects which are near enough to $\mathbf{x}$, that is, $Neighbor(\mathbf{x}) = \{\mathbf{y} | Distance(\mathbf{x}, \mathbf{y}) \leqslant \delta\}$. $Neighbor(\mathbf{x})$ means the area whose center is $\mathbf{x}$ and δ is the radius, we say that objects in the area are the distance of δ relative to x. The cardinality of $Neighbor(\mathbf{x})$, $|Neighbor(\mathbf{x})|$, reflects the density of the area. Bigger $|Neighbor(\mathbf{x})|$ is, more compact the area is. If the density is big enough, it is reasonable that we view the area in its entirety.

Definition 1 (*Representative points*). If $|Neighbor(r)| \geqslant \zeta$, we call that r is a representative point and represents the objects in the area whose center is r and δ is the radius.

Representative points can be fictional points, not points/objects in the system. ζ is the density threshold. In this paper, we propose a sort method, described in Example 1, to obtain representative points instead of directly using the threshold due to reduce the number of thresholds.

Definition 2 (*Representing region*). Every representative point r is the representative of a circular area where the point is the fictitious center of the area and the radius is δ, we call the area is the representing region of the corresponding representative point r.

All objects in the representing region of a representative point are seen as an entirety; an object which does not be represented by any representative points is deemed to be a noise.

Assuming r_k is the kth representative point, we use $Cover(r_k) = \{x_1, \ldots, x_k\}$ to denote the objects in its representative region. Since the representative point r_k can represent the region, it is reasonable to suppose that the fictional point has D-dimensions attributes. That is $r_k = (r_k^1, \ldots, r_k^d, \ldots, r_k^D)$, and a range is used to represent the r_k^d, namely, $r_k^d = [r_k^d.left, r_k^d.right]$. The following formulas are used to compute them: $r_k^d.left = min\{x_1^d, \ldots, x_k^d\}$, and $r_k^d.right = max\{x_1^d, \ldots, x_k^d\}$. Generally speaking, it is possible that there exists overlapping region between two representative regions.

Definition 3 (*Similarity between representative regions*). Let r_i and r_j be two arbitrary representative points, the similarity between their representative regions is defined as follows.

$$SimilarityRR(r_i, r_j) = \frac{|Cover(r_i) \cap Cover(r_j)|}{min(|Cover(r_i)|, |Cover(r_j)|)}. \tag{6}$$

Here, $|\cdot|$ means the cardinality of a set.

In order to speed up searching the similar space with the incremental data, we build the searching tree based on representative points. The root represents the original space composing of all representative points, and we sort the attributes by significance. According to the most significance attribute, we construct the nodes in the 1st layer; that is, all representative points are split according to these representative points' values in the corresponding attribute. Then the second significance attribute and so on.

Definition 4. Let $Node_j^i$ be the jth node of the i layer in the searching tree, let $R = \{r_1, \ldots, r_{|Node_j^i|}\}$ be the set of representative points belonging to the node $Node_j^i$.

A node is represented by a value range: $Node_j^i = [Node_j^i.left, Node_j^i.right]$, where $Node_j^i.left = min\{r_1^i.left, \ldots, r_{|Node_j^i|}^i.left\}$, and $Node_j^i.right = max\{r_1^i.right, \ldots, r_{|Node_j^i|}^i.right\}$. A node represents not a representative point, but a set of representative points whose values among the value range.

In addition, we need to measure the similarity between the representative points of the incremental data and nodes of the tree.

That is, we need to measure the similarity between two mathematical value ranges.

Definition 5 (*Similarity of value range*). For arbitrary value range $Range1$ and $Range2$, wherein $Range1 = [Range1.left, Range1.right]$, so as to $Range2$. If $Range1.left \leqslant Range2.left$, and $Range1.right \geqslant Range2.left$, then we call that $Range1$ is similar to $Range2$.

In other words, we say two ranges are similar to each other if and only if $Range1 \cap Range2 \neq \emptyset$.

4.2. Clustering the initial data

The basic idea of the initial clustering algorithm is a static overlapping clustering algorithm using three-way decisions based on representative points, shorted by SOC-TWD, and Algorithm 1 outlines the top level overview.

Algorithm 1. SOC-TWD: Static overlapping clustering in the initial data set

Input: $U = \{\mathbf{x}_1, \cdots, \mathbf{x}_N\}$, α, β, and δ.
Output: $\mathbf{C} = \{[\underline{C_1}, \overline{C_1}], \cdots, [\underline{C_i}, \overline{C_i}], \cdots, [\underline{C_{|\mathbf{C}|}}, \overline{C_{|\mathbf{C}|}}]\}$;
R; $Neighbor(r_i)$, $r_i \in R$; G.
1 $R = \emptyset$; $\mathbf{C} = \emptyset$; $Neighbor = \emptyset$
2 Compute the $N * N$ distance matrix $[Distance(\mathbf{x}, \mathbf{y})]$ using Eq.7.
3 **for** *each row* $\boldsymbol{x}_i$ *in* $[Distance(\boldsymbol{x}, \boldsymbol{y})]$ **do**
4   **for** *each column* $\boldsymbol{x}_j$ *in* $[Distance(\boldsymbol{x}, \boldsymbol{y})]$ **do**
5   **if** $Distance(\boldsymbol{x}_i, \boldsymbol{x}_j) \leq \delta$ **then**
6             $Neighbor(\mathbf{x}_i) = Neighbor(\mathbf{x}_i) \cup \{\mathbf{x}_j\}$;
7 Sort $\mathbf{x}_i$ according to the value of $|Neighbor(\mathbf{x}_i)|$ in descending order.
//look for representative points in U
8 new=0;
9 **while** $[Distance(\boldsymbol{x}, \boldsymbol{y})] \neq []$ **do**
10   new+=1;
11   Set the centroid of r_{new} be the object $\mathbf{x}_1$ corresponding to the first row of $[Distance(\mathbf{x}, \mathbf{y})]$;
12     $Cover(r_{new}) = Neighbor(\mathbf{x}_1)$;
13   **for** *each* $\boldsymbol{x}_i$ *in* $Cover(r_{new})$ **do**
14   **for** *each dimension value* x_i^d *of* $\boldsymbol{x}_i$ **do**
15   if $r_{new}^d.left > x_i^d, r_{new}.left = x_i^d$;
16   if $r_{new}^d.right < x_i^d, r_{new}.right = x_i^d$;
17   Delete the corresponding $\mathbf{x}_i$ row in $[Distance(\mathbf{x}, \mathbf{y})]$.
18   add r_{new} to R;
19 //construct the relation graph G of representative points
20 **for** *each representative point* r_i *in* R **do**
21   **for** *each representative point* $r_j \neq r_i$ *in* R **do**
22   if $Distance(r_i, r_j) \leq 2\delta$, $Neighbor(r_i) = Neighbor(r_i) \cup r_j$;
23   Compute $SimilarityRR(r_i, r_j)$ using Eq.6;
24   if $SimilarityRR(r_i, r_j) \geq \alpha$, to add a strong linked edge between them;
25   if $\beta \leq SimilarityRR(r_i, r_j) < \alpha$, to add a weak linked edge;
26 //obtain the initial clustering
27 Find out strong connected subgraphs whose edges are connected by strong linked edges in G using breadth-first search.
28 new=0;
29 **for** *each sub-graph* G' *in* G **do**
30   new+=1;
31   **for** *each representative point* r_i *in* G' **do**
32         $POS(C_{new}) = POS(C_{new}) \cup Cover(r_i)$;
33   **for** *each* r_j *which is linked to* G' *with weak edge* **do**
34         $BND(C_{new}) = BND(C_{new}) \cup Cover(r_j)$;
35     $\mathbf{C} = \mathbf{C} \cup C_{new}$;

The algorithm starts with computing the distance between objects, $Distance(\mathbf{x}, \mathbf{y})$. This paper uses the Euclidean distance to measure the distance between two objects as follows:

$$Distance(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{d=1}^{D} (x^d - y^d)^2}. \tag{7}$$

Obviously, shorter the distance between two objects is, more similar they are. Of course, we can define the similarity between two objects as follows:

$$Similarity(\mathbf{x}, \mathbf{y}) = 1 - Distance(\mathbf{x}, \mathbf{y}). \tag{8}$$

Larger the similarity between two objects is, more similar they are.

From Line 9 to Line 18, the algorithm determines the representation points, where a removing strategy is used in order to reduce the number of thresholds. That is, the algorithm set the object which has the maximum neighbors to be the first representation point. Then, it removes the corresponding rows from the distance matrix. After that, the algorithm find the second representation point in the rest matrix, and so on.

From Line 19 to Line 25, Algorithm 1 constructs a undirected graph G based on the set of representation points, R, using the idea of three-way decisions. Here, α and β are thresholds. For all $r_i, r_j \in R$, to compute the $SimilartyRR(r_i, r_j)$ according to Eq. (6). If $SimilartyRR(r_i, r_j) \geqslant \alpha$, there is a strong linked edge between them, if $\beta \leqslant SimilartyRR(r_i, r_j) < \alpha$, there is a weak linked edge between them. Line 21 computes the neighbor representative points of every representative points, which will be used in Algorithm 4.

From Line 27 to Line 35, the algorithm searches the subgraph which is strong connected by strong linked edges in the graph G. For every such subgraph, the objects corresponding to it form the positive region of a cluster, $POS(C)$; the objects, in the union of representative regions which have weak edges connected to this subgraph, form the boundary of the cluster $BND(C)$.

Example 1. An Example to obtain the representative points.

Table 1 describes a dataset, which has ten objects. Table 2 gives the distance matrix between these objects. Set the threshold $\delta = 1.5$. Then, we select the object which has the maximum similar objects from Table 2. Thus, we choose $\mathbf{x}_9$ to be the geometrical center of the first representation point r_1, the corresponding representation region is $Cover(r_1) = \{\mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_5, \mathbf{x}_6, \mathbf{x}_7, \mathbf{x}_8, \mathbf{x}_9\}$. After that, the rows which the objects in the corresponding representation region are removed from the matrix, then we obtain Table 3.

From Table 3, we choose the object, $\mathbf{x}_1$, which has the maximum similar objects, to be the geometrical center of the second representation point r_2. Likewise, the other two representation regions are $Cover(r_2) = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_5, \mathbf{x}_{10}\}$, and $Cover(r_3) = \{\mathbf{x}_4\}$.

4.3. Creating the searching tree

The basic idea of the proposed method is to represent the incremental data as representation points firstly, which brings benefit of saving computing time compared with methods based on objects. When we know the relationships between new representation points and existing representation points, we can make decisions accordingly. Therefore, we store the existing representation points on a tree, and we can take the advantage of searching and updating operations on a tree structure.

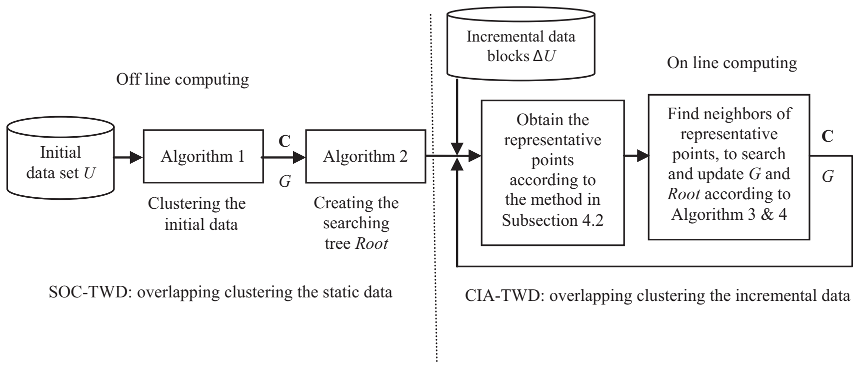


Fig. 1. Illustration of the processing of TIOC-TWD method.

Algorithm 2. Creating the searching tree

Input: R, A, and λ.
Output: $Root$, the root of the searching tree.

1 $i = 0$, $Root = node_0^0 = R$; and $Node(i) = Node(i) \bigcup node_0^0$;
2 **while** $\frac{|Node(i-1)|}{|Node(i)|} < \lambda$ *or* $i \le |A|$ **do**
3   **for** *each* $node_j^i$ *in* $Node(i)$ **do**
4   Sort the representative points in $node_j^i$ according to the $i+1$th attribute value in ascending order;
5   Initial a new $i+1$th layer node $node_j^{i+1} = r_k$;
6   **for** *each* r_p *in* $node_j^i$ *where* $p = k+1$ **do**
7   To determine whether r_p is similar to $node_j^{i+1}$ according to the $i+1$th attribute using Definition 5;
8   **if** r_p *is similar to* $node_j^{i+1}$ **then**
9         $node_j^{i+1} = node_j^{i+1} \cup \{r_p\}$;
10   **else**
11   For all representation points in $node_j^{i+1}$, to compute the value ranges on the $i+1$th attribute according to Definition 4;
12         $Node(i+1) = Node(i+1) \cup node_j^{i+1}$;
13         $node_{j+1}^{i+1} = r_p$;//generate a new child node of $node_j^i$;
14       $k = k+1$;
15   $i = i+1$;

The method of creating searching tree is similar to that of creating the decision tree, which is built top-down. It constructs the tree according to the attribute importance. This paper utilizes a measure of node impurity to scale the attribute importance. The common indices include the entropy index, Gini index, misclassification error, and so on [41]. The entropy index is used to measure the attribute importance in this paper. Algorithm 2 outlines the top level overview of creating searching tree.

The sorted attributes are denoted as A. Algorithm 2 builds every layer according to the attribute importance, the more important attribute is prior to be constructed. Line 2 to Line 14 consider a situation that there exists two adjacent layer whose numbers of nodes are roughly same, the algorithm stops building the searching tree in order to reduce the depth of tree. Here, $Node(i)$ denotes all of nodes in the ith layer, and $|Node(i)|$ is the number of nodes of the ith layer. $\lambda \in (0,1)$ is a threshold, if $|Node(i-1)|/|Node(i)| \geqslant \lambda$, the algorithm stops.

Example 2. An Example to create the searching tree.

Fig. 2 is an example of creating the searching tree. There are 5 representation points in Fig. 2(a), where the solid line denotes the strong linked edge and the dotted line denotes the weak linked edge. Table 4 shows the value ranges of representation points on every attribute; we assume the importance of a_1 is greater than a_2. According to Algorithm 2, the root of searching tree concludes the five representation points firstly as denoted in Fig. 2(b). According to Definition 5 and the most important attribute a_1, the root is split into two nodes to build the first layer, as shown in Fig. 2(b). Then, the algorithm splits the tree according to the second important attribute a_2. In this example, the second layer is the same as the first layer, the algorithm stops.

We need to note that, the node of the searching tree should map one representation point or multiple representation points.

Table 1
A dataset U.

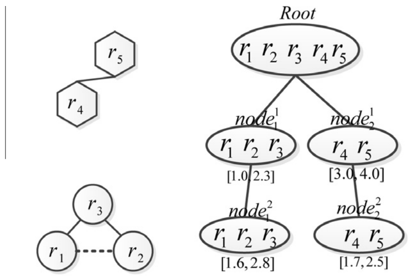
U	a_1	a_2	a_3	a_4	a_5
$\mathbf{x}_1$	0.5	0	1	2	0
$\mathbf{x}_2$	1	0	0	1	0
$\mathbf{x}_3$	1	0	1	1	0
$\mathbf{x}_4$	1.5	1	2	0	1
$\mathbf{x}_5$	1	0	0	2	0
$\mathbf{x}_6$	0	1	0	1	0
$\mathbf{x}_7$	1	2	0	1	0
$\mathbf{x}_8$	0	1.5	0	1	0
$\mathbf{x}_9$	1	1	0	1	0
$\mathbf{x}_{10}$	0	0	1	2	0

Table 2
The distance matrix of U.

$Distance(\mathbf{x}_i, \mathbf{x}_j)$	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$	$\mathbf{x}_5$	$\mathbf{x}_6$	$\mathbf{x}_7$	$\mathbf{x}_8$	$\mathbf{x}_9$	$\mathbf{x}_{10}$
$\mathbf{x}_1$	0.0	1.5	1.1	2.8	1.1	1.8	2.5	2.1	1.8	0.5
$\mathbf{x}_2$	1.5	0.0	1.0	2.6	1.0	1.4	2.0	1.8	1.0	1.7
$\mathbf{x}_3$	1.1	1.0	0.0	2.0	1.4	1.7	2.2	2.0	1.4	1.4
$\mathbf{x}_4$	2.8	2.6	2.0	0.0	3.2	2.8	2.6	2.9	2.5	3.0
$\mathbf{x}_5$	1.1	1.0	1.4	3.2	0.0	1.7	2.2	2.0	1.4	1.4
$\mathbf{x}_6$	1.8	1.4	1.7	2.8	1.7	0.0	1.4	0.5	1.0	1.7
$\mathbf{x}_7$	2.5	2.0	2.2	2.6	2.2	1.4	0.0	1.1	1.0	2.6
$\mathbf{x}_8$	2.1	1.8	2.0	2.9	2.0	0.5	1.1	0.0	1.1	2.0
$\mathbf{x}_9$	1.8	1.0	1.4	2.5	1.4	1.0	1.0	1.1	0.0	2.0
$\mathbf{x}_{10}$	0.5	1.7	1.4	3.0	1.4	1.7	2.6	2.0	2.0	0.0

Table 3
The distance matrix after removing objects in $Cover(r_1)$.

$Distance(\mathbf{x}_i, \mathbf{x}_j)$	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$	$\mathbf{x}_5$	$\mathbf{x}_6$	$\mathbf{x}_7$	$\mathbf{x}_8$	$\mathbf{x}_9$	$\mathbf{x}_{10}$
$\mathbf{x}_1$	0.0	1.5	1.1	2.8	1.1	1.8	2.5	2.1	1.8	0.5
$\mathbf{x}_4$	2.8	2.6	2.0	0.0	3.2	2.8	2.6	2.9	2.5	3.0
$\mathbf{x}_{10}$	0.5	1.7	1.4	3.0	1.4	1.7	2.6	2.0	2.0	0.0



(a) Relation graph G (b) The searching tree

Fig. 2. Example to create the searching tree.

Table 4
Value ranges of representation points in Fig. 2.

R	r_1	r_2	r_3	r_4	r_5
a_1	[1.2,1.6]	[1.0,1.5]	[1.4,2.3]	[3.0,3.6]	[3.3,4.0]
a_2	[2.0,2.8]	[1.6,1.9]	[1.8,2.2]	[1.9,2.5]	[1.7,2.0]

4.4. Clustering the incremental data

The incremental data is denoted by ΔU, the TIOC-TWD clustering method does not need to cluster again on all data ($U \cup \Delta U$), it just need to execute the clustering incremental data algorithm, abbreviated as CIA-TWD.

The objects in the ΔU are not completely irrelative with each other, there exists some links among them. Therefore, it is reasonable that we obtain representation points in the incremental data firstly using Algorithm 1. That is to say the CIA-TWD algorithm is based on representation points, and it includes two steps:

Step 1: it obtains the representative points in the ΔU according to the method in Algorithm 1, and a new representative point in ΔU is denoted by r_{wait};

Step 2: it searches and updates the relation graph G obtained in Algorithm 1 according to the method described in Algorithm 4.

Obviously, how to carry out Step 1 of the CIA-TWD is clear, and how to carry out Step 2 is introduced in Algorithm 4. The basic idea of the CIA-TWD is to search neighbors of every representative point r_{wait}, and update the related area on the tree and the graph. Therefore, we need to introduce how to find the neighbors of every representative point firstly, which is recorded in Algorithm 3.

4.4.1. Finding neighbors of r_{wait}

Algorithm 3 not only searches the tree but also updates the tree at the same time. Let $SimilarNode(i) = \{Node^i_{k1}, \ldots, Node^i_{kn}\}$ be the set of similar nodes with r_{wait} in the ith layer, and $Node^i_{new}$ be the new node in ith layer. Algorithm 3 first finds out the similar nodes with r_{wait} according to Definition 5. Line 28 to Line 32 describe how to find the neighbors of r_{wait} from the representative points which in the set of similar nodes with r_{wait}.

Considering the relationships between r_{wait} and nodes in the searching tree: Case 1, there only one node is similar with r_{wait} in every layer; Case 2, there are more than one node similar with r_{wait} at least in one layer; Case 3, there are no nodes similar with r_{wait} at least in one layer. Under the similar cases, they will lead to merge nodes. For Case 1, the merging just arises to r_{wait} and the similar node; for Case 2, the merging might arise the similar nodes and their children. To the contrary, under the nosimilar case, there will arise splitting operations.

Algorithm 3. FindingNeighbors(r_{wait})

Input: $Root$, r_{wait}, δ.
Output: $R_{neighbor}$, the neighbor representative points of r_{wait}; $Path$, to record the path of searching.

1 $i = 1$; $SimilarNode[] = \emptyset$;
2 $Root = Root \bigcup r_{wait}$, pointer P points to $Root$ and add P to $Path$;
3 add all kids of $Root$ to $SimilarNode(i)$;
4 **while** $SimilarNode(i) \neq NULL$ **do**
5 initial a new ith layer node $node^i_{new} = r_{wait}$;
6 **for** *each* $node^i_j$ *in* $SimilarNode(i)$ **do**
7 //To determine whether $node^i_{new}$ is similar to $node^i_j$ according to the ith important attribute value using Definition 5;
8 **if** $node^i_{new}$ *is similar to* $node^i_j$ **then**
9 merge $node^i_{new}, node^i_j$ together;
10 **for** *each* $node^{i+1}_k$ *in child nodes of* $node^i_j$ **do**
11 $ChildNode = ChildNode \cup node^{i+1}_k$;
12 **if** *none* $node^i_j$ *in* $SimilarNode(i)$ *is similar to* $node^i_{new}$ **then**
13 add $node^i_{new}$ to kids set of P;
14 **while** *i is no more than the depth of tree* **do**
15 pointer P points to $node^i_{new}$; $i = i + 1$;
16 initial a new ith layer node $node^i_{new} = r_{wait}$;
17 add $node^i_{new}$ to kids set of P;
18 break;
19 pointer P points to $node^i_{new}$, add P to $Path$;
20 Sort the nodes in $ChildNode = \{node^{i+1}_1, \cdots, node^{i+1}_j, \cdots\}$ according to the $i+1$th important attribute value in ascending order;
21 initial a new the $i+1$th node $node^{i+1}_{new} = node^{i+1}_j$;
22 **for** *each* $node^{i+1}_p$ *in* $ChildNode$ *where* $p = j + 1$ **do**
23 **if** $node^{i+1}_p$ *is similar to* $node^{i+1}_{new}$ **then**
24 merge $node^{i+1}_{new}, node^{i+1}_p$ together;
25 **else**
26 add $node^{i+1}_{new}$ to $SimilarNode(i+1)$,$node^{i+1}_{new+1} = node^{i+1}_p$;
27 $j = j + 1$;
28 **if** *the ith layer is the last layer of the tree* **then**
29 **for** *each* $r \in node^i_{new}$ *and* $r \neq r_{wait}$ **do**
30 **if** $Distance(centroid\ of\ r_{wait}, centroid\ of\ r) \leq 2 * \delta$ **then**
31 $R_{neighbor} = R_{neighbor} \cup r$;
32 break;
33 **else**
34 $i = i + 1$;

In view of the different value region of r_{wait}, we explain how to find the similar nodes through the following example respectively. We take the example searching tree in Fig. 2(b) as the initial searching tree.

Example 3. Examples to find similar nodes with r_{wait}.

Fig. 3 shows an example how to deal with Case 1. Here, to assume the value region of r_{wait} in attribute a_1 is $[1.2, 1.8]$, the value region in a_2 is $[2.0, 2.5]$. The incremental representative point r_{wait} is added to the root firstly. The indicator $Path$, which records the path of searching, indicates the root.

Fig. 3(a) shows processing with the 1st layer, $Node^1_1 = [1.0, 2.3]$, the value region of r_{wait} in attribute a_1 is $[1.2, 1.8]$. Because $[1.0, 2.3] \cap [1.2, 1.8] \neq \emptyset$, we have r_{wait} is similar with $Node^1_1$ according to Definition 5. Then, r_{wait} is added to the node $Node^1_1$; and the indicator $Path$ moves to $Node^1_1$. The algorithm moves to the 2nd layer, the child of $Node^1_1$, namely $Node^2_1 = [1.6, 2.8]$. We have

Please cite this article in press as: H. Yu et al., A tree-based incremental overlapping clustering method using the three-way decision theory, Knowl. Based Syst. (2015), http://dx.doi.org/10.1016/j.knosys.2015.05.028

$[1.6, 2.8] \cap [2.0, 2.5] \neq \emptyset$, thus r_{wait} is similar with $Node_1^2$. The new representative point is added to the node $Node_1^2$ and the *Path* moves to $Node_1^2$. The result is shown in Fig. 3(a).

Finally, the algorithm finds the neighbors of r_{wait} by computing the distance between centrals of representative points in $Node_1^2$ and r_{wait}; which is described in Line 28 to Line 32.

Fig. 4 gives an example of Case 2. Here, assume the value region of r_{wait} in attribute a_1 is $[2.0, 3.1]$, the value region in a_2 is $[2.2, 2.3]$. In this situation, there are more than one node similar to r_{wait}. We have r_{wait} is similar with $Node_1^1$ and $Node_2^1$, because $[1.0, 2.3] \cap [2.0, 3.1] \neq \emptyset$ and $[3.0, 4.0] \cap [2.0, 3.1] \neq \emptyset$. Then, $Node_1^1, Node_2^1$ and r_{wait} are merged into one node as $Node_1^1$, which is shown in Fig. 4(a); and *Path* indicates $Node_1^1$. After that, the algorithm need to determine whether the kids of $Node_1^1$ can be merged. That is, because $[1.6, 2.8] \cap [1.7, 2.5] \neq \emptyset$, the $Node_1^2$ and $Node_2^2$ are merged into one node as $Node_1^2$; which is shown in Fig. 4(b). Likewise, the algorithm deals with the 2nd layer. Because $[1.6, 2.8] \cap [2.2, 2.3] \neq \emptyset, r_{wait}$ is added to $Node_1^2$; which is shown in Fig. 4(c). $Node_1^2$ is added to *Path*. Finally, the algorithm moves to Line 28 to find the neighbors of r_{wait}.

Fig. 5 gives an example of Case 3. In this situation, there are no nodes similar to r_{wait}. Assume the value region of r_{wait} in attribute a_1 is $[2.8, 3.2]$, the value region in a_2 is $[2.6, 3.0]$. Fig. 5(a) shows the processing on the 1st layer. Because $[3.0, 4.0] \cap [2.8, 3.2] \neq \emptyset, r_{wait}$ is added to $Node_2^1$; and $Node_2^1$ is added to *Path*. Then, we observe the 2nd layer. Because $[1.7, 2.5] \cap [2.6, 3.0] = \emptyset$, there is no similar node with r_{wait} in 2nd layer. Thus, the node $Node_2^1$ will split into two child nodes as $Node_2^2$ and $Node_3^2$; which is shown in Fig. 5(b).

4.4.2. Updating operations

After processing Algorithm 3, we know that a new representative point r_{wait} might or not have some neighbors. Algorithm 4 presents the high level of the updating processing.

Let $R_{neighbor}$ be the set of neighboring representative points with r_{wait}. Obviously, there exists three relationships between r_{wait} and its neighbors. Relationship 1: $R_{neighbor} \neq \emptyset$, and the representative region of r_{wait} is covered completely by representative regions of $R_{neighbor}$. Relationship 2: $R_{neighbor} \neq \emptyset$, and the part of representative region of r_{wait} is covered by representative regions of $R_{neighbor}$. Relationship 3: $R_{neighbor} = \emptyset$, namely r_{wait} has no neighbors.

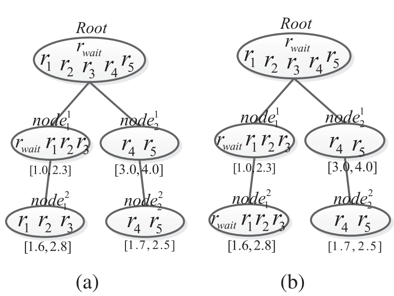


Fig. 3. Case 1 of finding similar nodes of r_{wait}.

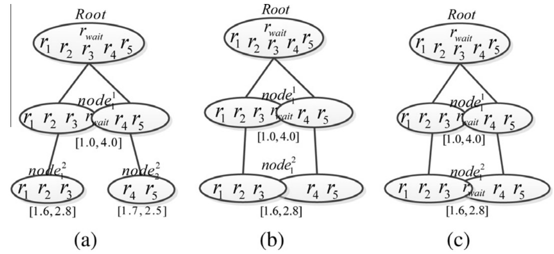


Fig. 4. Case 2 of finding similar nodes of r_{wait}.

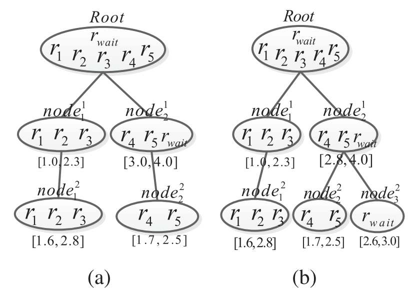
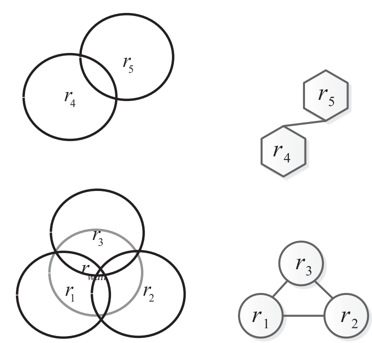


Fig. 5. Case 3 of finding similar nodes of r_{wait}.



(a) Diagram of Relationship 1 (b) After updating

Fig. 6. Example for updating operations on Relationship 1.

In view of the different relationships, we explain how to update the graph through the following examples respectively. We take the example in Fig. 2 as the initial clustering pattern.

Example 4. Examples to update the relation graph *G*.

Fig. 6 shows an example for updating operations on Relationship 1. Here, $R_{neighbor} = \{r_1, r_2, r_3\}$. r_{wait} is covered completely by representative regions of $R_{neighbor}$. Under this situation, no new representative point is produced. Objects in representative region of r_{wait} are mapped to the corresponding areas represented by $R_{neighbor}$; which is shown in Fig. 6(a).

Algorithm 4. UpdatingClustering

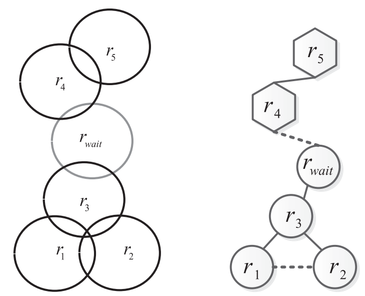
Input: R, the set of incremental representative points; G, α, β and δ.
Output: $\mathbf{C} = \{[\underline{C_1}, \overline{C_1}], \cdots, [\underline{C_i}, \overline{C_i}], \cdots, [\underline{C_{|\mathbf{C}|}}, \overline{C_{|\mathbf{C}|}}]\}$.

1 **for** *each* r_{wait} *in* R **do**
2 FindingNeighbors(r_{wait});//Algorithm 3
3 //mapping each $\mathbf{x}_i$ in r_{wait} to neighbor representative points in $R_{neighbor}$;
4 **for** *each* $\mathbf{x}_i$ *in* r_{wait} **do**
5 **for** *each* r_i *in* $R_{neighbor}$ **do**
6 if $Distance(\mathbf{x}_i$,centroid of $r_i) \leq \delta$,
7 $Cover(r_i) = Cover(r_i) \cup x_i$;
8 //no mapping;
9 **if** *there exists* $\mathbf{x}_j$ *which can't mapping to any* r *in* $R_{neighbor}$ **then**
10 **for** *each* r_i *in* $R_{neighbor}$ **do**
11 **for** *each* $\mathbf{x}_j$ *in* r_i **do**
12 if $Distance(\mathbf{x}_j$,centroid of $r_{wait}) \leq \delta$,
13 $Cover(r_{wait}) = Cover(r_{wait}) \cup x_j$;
14 add r_{wait} to $R_{neighbor}$;
15 //mapping;
16 **else**
17 **for** *each tree node* $node^i_j$ *in Path* **do**
18 find the representative point r_{wait} from $node^i_j$, delete it;
19 //update the changed subgraph in G;
20 **for** *each* r_i *in* $R_{neighbor}$ **do**
21 **for** *each* r_j *in* $neighbor(r_i)$ **do**
22 compute the similarity value between r_i, r_j using **Eq.6**;
23 if $SimilarityRR(r_i, r_j) \geq \alpha$, add a strong linked edge between them;
24 if $\beta \leq SimilarityRR(r_i, r_j) < \alpha$, add a weak linked edge;
25 if $SimilarityRR(r_i, r_j) < \beta$, no edge between them;
26 //obtain the final clustering result
27 new=0;
28 **for** *each changed sub-graph* G' *in* G **do**
29 new+=1;
30 **for** *each representative point* r_i *in* G' **do**
31 $POS(C_{new}) = POS(C_{new}) \cup Cover(r_i)$;
32 **for** *each* r_j *which is linked to* G' *with weak edge* **do**
33 $BND(C_{new}) = BND(C_{new}) \cup Cover(r_j)$;
34 $\mathbf{C} = \mathbf{C} \cup C_{new}$;

Because there is no representative point produced in the updated tree, we need to remove r_{wait} from the *Path*. On the other hand, because new data added, the representative regions of $R_{neighbor}$ will change. The similarity among $R_{neighbor}$ will be recalculated. In this example, to assume $SimilarityRR(r_1, r_2) \geqslant \alpha$, which means the relation between r_1 and r_2 is changed from weak link to strong link. The updated graph is shown in Fig. 6(b).

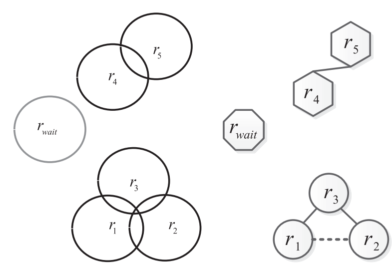
Fig. 7 shows an example on Relationship 2. Here, $R_{neighbor} = \{r_3, r_4\}$. The representative region of r_{wait} is covered partly by representative regions of $R_{neighbor}$. Under this situation, a new representative point is produced in the tree. Objects in representative region of r_{wait} covered by representative region of $R_{neighbor}$ are mapped to the corresponding areas represented by the neighbors, and objects in representative region of $R_{neighbor}$ covered by representative region of r_{wait} are mapped to the corresponding area represented by the r_{wait}; which is shown in Fig. 7(a).

Because the representative region of $R_{neighbor}$ changes, the similarity between $R_{neighbor}$ and r_{wait} will be recalculated. In this example, to assume $\beta \leqslant SimilarityRR(r_4, r_{wait}) < \alpha$, there is a weak link between r_4 and r_{wait}; if we have $SimilarityRR(r_3, r_{wait}) \geqslant \alpha$, there is a strong link between r_3 and r_{wait}. The updated graph is shown in Fig. 7(b).



(a) Diagram of Relationship 2 (b) After updating

Fig. 7. Example for updating operations on Relationship 2.



(a) Diagram of Relationship 3 (b) After updating

Fig. 8. Example for updating operations on Relationship 3.

Fig. 8 shows an example for updating operations on Relationship 3. Under this situation, $R_{neighbor} = \emptyset, r_{wait}$ is a new representative point; which is shown in Fig. 8(a). The updated graph is shown in Fig. 8(b). There is a new cluster produced.

Three-way decision strategy is used to update the changed subgraphs from Line 19 to Line 25. If similarity between representative regions is no less than α, namely $SimilarityRR(r_i, r_j) \geqslant \alpha$, we add a strong linked edge between them; if $\beta \leqslant SimilarityRR(r_i, r_j) < \alpha$, we add a weak linked edge between them. From Line 26 to Line 34, the algorithm outputs the finial incremental clustering result. That is, a strong linked edge means the adjacent node is in the corresponding positive region, and a weak linked edge means the adjacent node is in the corresponding boundary region.

4.5. Time complexity analysis

We can also execute the static overlapping clustering Algorithm 1 on the whole dataset $U \cup \Delta U$ to obtain clustering results, while the incremental clustering method CIA-TWD is executed mainly on the incremental data ΔU. On the other hand, for the CIA-TWD algorithm in Section 4.4, it mainly concludes Algorithms 3 and 4. Therefore, this subsection will analyze time complexities of these algorithms.

Let's see the static overlapping clustering SOC-TWD algorithm, namely, Algorithm 1. Assume the number of objects is $|U| = n$, the average number of objects in a representative region is p, the number of representative points is R. Then, to find all the representative points has a complexity of $O(n^2 + R \times p + nlog(n))$. To construct the relation graph G of representative points has a complexity of $O(R^2)$. To search the graph has a complexity of $O(R^2)$. Assume the number of clusters is C, and the average number of representative points in a cluster is k. Then, clustering on the relation graph of representative points has a complexity of $O(C \times k)$. Thus, the algorithm has a complexity of

Table 5
Information about the datasets.

Datasets	N	D	M	δ	α	β	λ
AD	2000	3	4	0.17	0.15	0.01	0.9
Banknote	1372	5	2	2	0.06	0.03	0.9
LetterABC	2291	17	3	4.6	0.06	0.03	0.9
LetterAGI	2317	17	3	4.6	0.06	0.03	0.9
Page blocks	5473	11	5	400	0.10	0.05	0.9
Pendigits389	3165	17	3	45	0.26	0.13	0.9
Pendigits1234	4486	17	4	30	0.30	0.15	0.9
Pendigits1469	4398	17	4	38	0.20	0.10	0.9
Waveform	5000	22	3	8	0.9	0.7	0.9
Landsat	6435	37	6	38	0.6	0.3	0.9

$O(n^2 + R \times p + nlog(n) + 2 \times R^2 + C \times k)$. In fact, p, C and k are very small, the algorithm has thus a complexity of $T(\text{SOC-TWD}) = O(n^2 + nlog(n) + 2 \times R^2)$.

Let's see Algorithm 3, which finds neighbors of the $r(wait)$ by searching and updating the tree. Assume the depth of tree is h, and the average number of similar nodes with the $r(wait)$ on every layer is k_1. There are k_2 child nodes of these k_1 nodes. To merge these child nodes requires $k_2 \times log(k_2) + k_2 - 1$ operations, and to merge these subtree requires $k_1 - 1$ operations. Thus, to search the tree requires a complexity of $O(h \times ((k_1 - 1) + (k_2 \times log(k_2) + k_2 - 1)))$. Assume the average number of representative points in leave nodes is k_3, the algorithm needs k_3 operations to find the neighbors. Thus, the algorithm has a complexity of $O(h \times ((k_1 - 1) + (k_2 \times log(k_2) + k_2 - 1)) + k_3)$. In fact, there is little nodes which are similar with the incremental representative point, that is k_1 is very small. Therefore, the algorithm has thus a complexity of $O(h \times (k_2 \times log(k_2)) + k_3)$. In the worst case, the depth of the tree is the number of attributes, that is $h = m$.

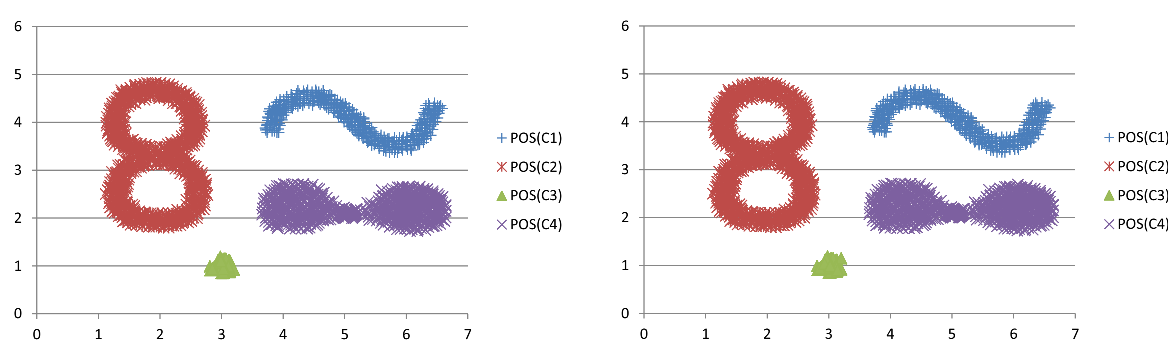
Let's see Algorithm 4, which updating clustering results. First, to search the neighbors of new incremental representative points has a complexity of $O(h \times (k_2 \times log(k_2)) + k_3)$ according to Algorithm 3. Assume the number of new representative points is R', the average number of objects in a representative region is p_1, and the average number of representative points in a neighbor is p_2. Mapping each object in $r(wait)$ to neighbor representative points costs $2 \times p_1 \times p_2$ operations. Assume there is p_3 representative points in G related to the new representative point. To update the graph needs p_3^2 operations. Assume there is C' subgraph related to the new representative point, and the average number of representative points of a subgraph is k_4. Then, to update the clustering results needs $O(C' \times k_4)$. Therefore, the algorithm has a complexity of $T_4 = O(R' \times (h \times (k_2 \times log(k_2)) + k_3) + R' \times (2 \times p_1 \times p_2 + p_3^2)) + O(C' \times k_4))$.

Considering the new arriving data, the number of objects is n', the number of new representative points is R', and the average number of objects in a representative region is p_1. According to Algorithm 1. We know that to find all the representative points has a complexity of $O(n'^2 + R' \times p_1 + n'log(n'))$. Therefore, the complete CIA-TWD algorithm has a complexity of $T(\text{CIA-TWD}) = O(n'^2 + R' \times p_1 + n'log(n')) + T_4$. Generally speaking, the parameters in T_4 are far less than n', even if we set R' or p_1 near to n', the complexity of CIA-TWD algorithm is $O(n'^2 + n'log(n'))$ at the worst case. However, because of $n' \ll (n + n'), T(\text{CIA-TWD}) \ll T(\text{SOC-TWD})$.

5. Experimental results

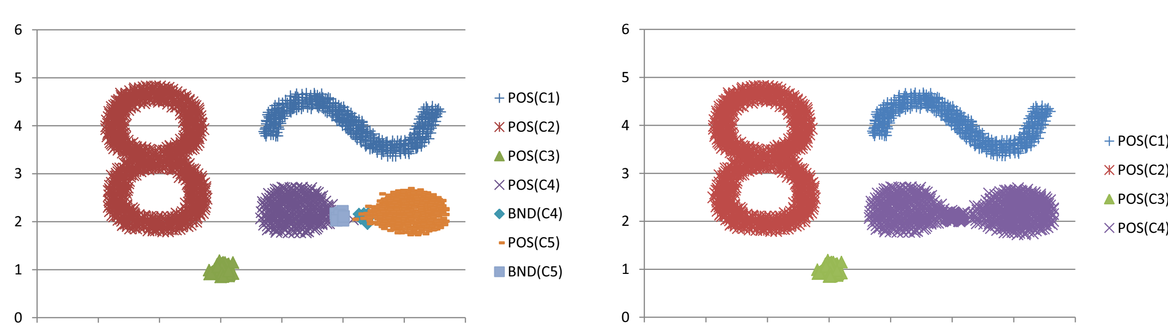
5.1. Evaluation indices and datasets

We evaluate the proposed TIOC-TWD clustering approach through the following experiments. All the experiments are



(a) Before the new data arriving (b) After the new data arriving

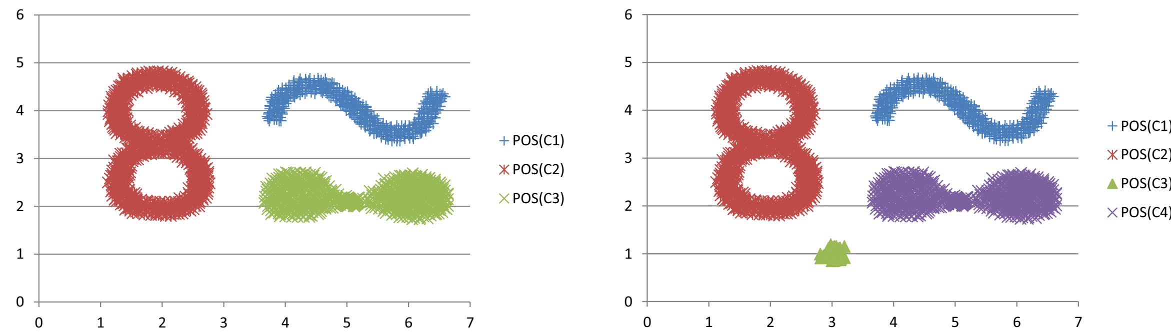
Fig. 9. Clustering results of Test 1.



(a) Before the new data arriving (b) After the new data arriving

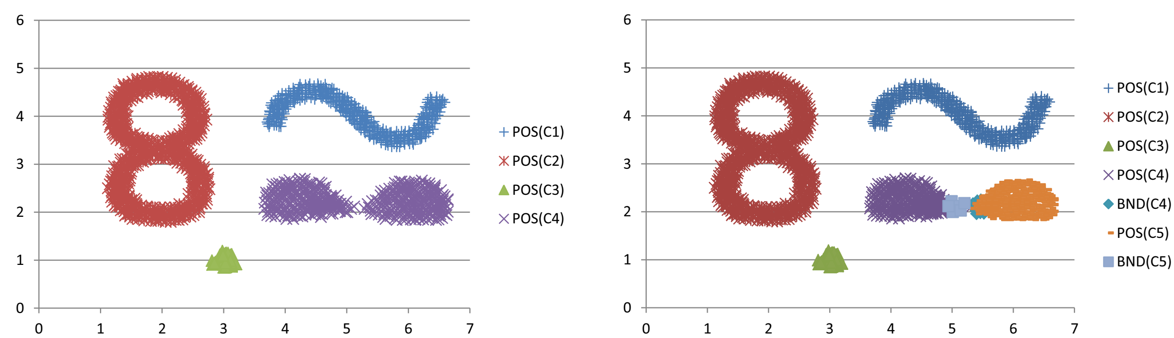
Fig. 10. Clustering results of Test 2.

Please cite this article in press as: H. Yu et al., A tree-based incremental overlapping clustering method using the three-way decision theory, Knowl. Based Syst. (2015), http://dx.doi.org/10.1016/j.knosys.2015.05.028



(a) Before the new data arriving (b) After the new data arriving

Fig. 11. Clustering results of Test 3.



(a) Before the new data arriving (b) After the new data arriving

Fig. 12. Clustering results of Test 4.

performed on a 2.67 GHz computer with 4 GB memory, and all algorithms are programmed in C++. The quality of the final clustering is evaluated by the traditional indices such as the accuracy, F-measure [42] and NMI [43], where the objects in boundary regions are deemed to be positive regions to fit these common formula.

Table 5 gives the summary information about the datasets and the parameters used in our experiments. AD is an artificial dataset, the other datasets come from the real datasets [44]. N, D and M means the number of objects, the number of attributes, and the number of ground-truth clusters, respectively. δ, α, β and λ are the input parameters. LetterABC is the subset of the original dataset with the letter "A" or "B" or "C"; LetterAGI is the subset of the letter "A" or "G" or "I". Pendigits389 is the subset with the digit 3, 8 and 9; PenDigits1234 and Pendigits1469 also are the corresponding subsets.

Table 6
Comparison of experimental results on the size of incremental data block is 10% under Situation 1.

Index	Method	Banknote	LetterABC	LetterAGI	Pageblocks	Pendigits389	Pendigits1234	Pendigits1469	Waveform	Landsat
Accuracy	TIOC-TWD	**0.85 ± 0.00**	0.89 ± 0.01	0.73 ± 0.00	0.88 ± 0.00	**0.83 ± 0.02**	0.83 ± 0.01	**0.88 ± 0.01**	**0.90 ± 0.01**	**0.73 ± 0.02**
	OFCMD	0.74 ± 0.11	0.91 ± 0.01	0.81 ± 0.02	0.90 ± 0.00	0.81 ± 0.11	0.89 ± 0.01	0.85 ± 0.10	0.52 ± 0.09	0.53 ± 0.01
	HOFCMD	0.80 ± 0.13	0.74 ± 0.16	0.69 ± 0.12	0.90 ± 0.00	0.62 ± 0.10	0.76 ± 0.11	0.76 ± 0.19	0.67 ± 0.05	0.61 ± 0.11
	IncDBSCAN	0.84 ± 0.00	0.80 ± 0.03	0.71 ± 0.02	0.87 ± 0.00	0.78 ± 0.00	0.71 ± 0.00	0.73 ± 0.00	0.34 ± 0.00	0.45 ± 0.00
	SOC-TWD	0.86 ± 0.00	0.85 ± 0.00	0.73 ± 0.01	0.88 ± 0.00	0.91 ± 0.00	0.84 ± 0.00	0.86 ± 0.04	0.92 ± 0.00	0.79 ± 0.00
Fmeasure	TIOC-TWD	**0.87 ± 0.00**	**0.91 ± 0.01**	0.72 ± 0.00	**0.86 ± 0.00**	**0.89 ± 0.02**	0.86 ± 0.00	**0.93 ± 0.01**	**0.62 ± 0.01**	**0.70 ± 0.02**
	OFCMD	0.61 ± 0.09	0.89 ± 0.01	0.79 ± 0.02	0.85 ± 0.00	0.68 ± 0.08	0.87 ± 0.01	0.54 ± 0.08	0.43 ± 0.07	0.47 ± 0.01
	HOFCMD	0.71 ± 0.09	0.64 ± 0.15	0.67 ± 0.14	0.85 ± 0.00	0.40 ± 0.08	0.49 ± 0.13	0.36 ± 0.11	0.38 ± 0.06	0.38 ± 0.06
	IncDBSCAN	0.86 ± 0.00	0.86 ± 0.02	0.80 ± 0.02	0.84 ± 0.00	0.87 ± 0.00	0.79 ± 0.00	0.67 ± 0.00	0.17 ± 0.00	0.40 ± 0.00
	SOC-TWD	0.92 ± 0.00	0.88 ± 0.00	0.82 ± 0.01	0.86 ± 0.00	0.91 ± 0.00	0.87 ± 0.00	0.91 ± 0.02	0.63 ± 0.00	0.70 ± 0.00
NMI	TIOC-TWD	**0.57 ± 0.00**	**0.77 ± 0.05**	0.57 ± 0.00	**0.04 ± 0.00**	0.80 ± 0.03	**0.77 ± 0.00**	**0.89 ± 0.01**	0.22 ± 0.03	**0.59 ± 0.01**
	OFCMD	0.04 ± 0.05	0.65 ± 0.03	0.51 ± 0.03	0.00 ± 0.00	0.43 ± 0.04	0.72 ± 0.01	0.45 ± 0.03	0.30 ± 0.04	0.43 ± 0.01
	HOFCMD	0.11 ± 0.09	0.44 ± 0.06	0.41 ± 0.09	0.00 ± 0.00	0.40 ± 0.13	0.46 ± 0.12	0.14 ± 0.03	0.15 ± 0.07	0.40 ± 0.03
	IncDBSCAN	0.56 ± 0.00	0.69 ± 0.03	0.66 ± 0.01	0.02 ± 0.00	0.80 ± 0.00	0.70 ± 0.00	0.81 ± 0.00	0.00 ± 0.00	0.52 ± 0.00
	SOC-TWD	0.80 ± 0.00	0.71 ± 0.00	0.66 ± 0.03	0.11 ± 0.00	0.76 ± 0.00	0.78 ± 0.0	0.86 ± 0.03	0.24 ± 0.00	0.64 ± 0.00
CPU(s)	TIOC-TWD	**0.29 ± 0.01**	1.45 ± 0.07	1.32 ± 0.03	**1.48 ± 0.25**	3.27 ± 0.08	6.83 ± 0.29	4.85 ± 0.20	6.69 ± 1.75	15.41 ± 0.26
	OFCMD	0.35 ± 0.03	1.16 ± 0.03	1.26 ± 0.03	20.02 ± 8.33	1.06 ± 0.02	3.97 ± 0.15	3.66 ± 0.31	3.51 ± 0.24	12.60 ± 0.42
	HOFCMD	0.46 ± 0.05	1.46 ± 0.14	2.05 ± 2.33	5.13 ± 0.57	2.32 ± 0.41	6.14 ± 0.46	4.63 ± 0.69	3.92 ± 0.46	30.14 ± 4.62
	IncDBSCAN	1.41 ± 0.11	3.51 ± 0.38	3.64 ± 0.51	242.78 ± 56.80	4.97 ± 0.14	13.23 ± 0.53	21.12 ± 1.41	11.76 ± 0.24	36.46 ± 6.77
	SOC-TWD	0.86 ± 0.05	3.41 ± 0.43	3.40 ± 0.13	7.59 ± 0.25	9.83 ± 0.30	35.10 ± 2.07	3.40 ± 0.13	34.00 ± 5.05	48.99 ± 1.81

Please cite this article in press as: H. Yu et al., A tree-based incremental overlapping clustering method using the three-way decision theory, Knowl. Based Syst. (2015), http://dx.doi.org/10.1016/j.knosys.2015.05.028

Table 7
Comparison of experimental results on the size of incremental data block is 20% under Situation 1.

Index	Method	Banknote	LetterABC	LetterAGI	Pageblocks	Pendigits389	Pendigits1234	Pendigits1469	Waveform	Landsat
Accuracy	TIOC-TWD	**0.85 ± 0.00**	**0.89 ± 0.01**	0.73 ± 0.00	0.88 ± 0.00	0.83 ± 0.01	0.83 ± 0.00	**0.88 ± 0.01**	**0.90 ± 0.00**	**0.73 ± 0.02**
	OFCMD	0.83 ± 0.13	0.59 ± 0.15	0.81 ± 0.01	0.90 ± 0.00	0.85 ± 0.02	0.88 ± 0.00	0.74 ± 0.23	0.50 ± 0.03	0.52 ± 0.05
	HOFCMD	0.62 ± 0.20	0.69 ± 0.19	0.38 ± 0.13	0.90 ± 0.00	0.50 ± 0.16	0.62 ± 0.15	0.71 ± 0.68	0.60 ± 0.15	0.62 ± 0.14
	IncDBSCAN	0.84 ± 0.00	0.80 ± 0.03	0.71 ± 0.02	0.87 ± 0.00	0.78 ± 0.00	0.71 ± 0.00	0.73 ± 0.00	0.34 ± 0.00	0.45 ± 0.00
	SOC-TWD	0.86 ± 0.00	0.85 ± 0.00	0.73 ± 0.01	0.88 ± 0.00	0.91 ± 0.00	0.84 ± 0.00	0.86 ± 0.04	0.92 ± 0.00	0.79 ± 0.00
Fmeasure	TIOC-TWD	**0.87 ± 0.00**	**0.91 ± 0.00**	0.72 ± 0.00	**0.86 ± 0.00**	**0.89 ± 0.01**	**0.86 ± 0.00**	**0.93 ± 0.01**	**0.62 ± 0.01**	**0.71 ± 0.02**
	OFCMD	0.67 ± 0.10	0.59 ± 0.13	0.79 ± 0.01	0.85 ± 0.00	0.71 ± 0.02	0.86 ± 0.00	0.47 ± 0.16	0.41 ± 0.01	0.46 ± 0.05
	HOFCMD	0.57 ± 0.19	0.59 ± 0.16	0.25 ± 0.18	0.85 ± 0.00	0.31 ± 0.14	0.38 ± 0.14	0.35 ± 0.10	0.36 ± 0.12	0.41 ± 0.09
	IncDBSCAN	0.86 ± 0.00	0.86 ± 0.02	0.80 ± 0.02	0.84 ± 0.00	0.87 ± 0.00	0.79 ± 0.00	0.67 ± 0.00	0.17 ± 0.00	0.40 ± 0.00
	SOC-TWD	0.92 ± 0.00	0.88 ± 0.00	0.82 ± 0.01	0.86 ± 0.00	0.91 ± 0.00	0.87 ± 0.00	0.91 ± 0.02	0.63 ± 0.00	0.70 ± 0.00
NMI	TIOC-TWD	**0.56 ± 0.00**	**0.75 ± 0.01**	0.57 ± 0.00	**0.04 ± 0.00**	0.80 ± 0.01	**0.77 ± 0.00**	**0.89 ± 0.01**	0.22 ± 0.01	**0.60 ± 0.01**
	OFCMD	0.06 ± 0.04	0.47 ± 0.07	0.52 ± 0.03	0.00 ± 0.00	0.42 ± 0.02	0.72 ± 0.01	0.35 ± 0.09	0.31 ± 0.03	0.44 ± 0.01
	HOFCMD	0.09 ± 0.11	0.39 ± 0.11	0.06 ± 0.13	0.00 ± 0.00	0.22 ± 0.19	0.43 ± 0.06	0.14 ± 0.03	0.15 ± 0.12	0.36 ± 0.04
	IncDBSCAN	0.56 ± 0.00	0.69 ± 0.03	0.66 ± 0.01	0.02 ± 0.00	0.80 ± 0.00	0.70 ± 0.00	0.81 ± 0.00	0.00 ± 0.00	0.52 ± 0.00
	SOC-TWD	0.80 ± 0.00	0.71 ± 0.00	0.66 ± 0.03	0.11 ± 0.00	0.76 ± 0.02	0.78 ± 0.00	0.86 ± 0.03	0.24 ± 0.00	0.64 ± 0.00
CPU(s)	TIOC-TWD	**0.37 ± 0.03**	1.71 ± 0.11	1.43 ± 0.07	**1.57 ± 0.23**	3.27 ± 0.19	6.95 ± 0.33	5.04 ± 0.21	7.71 ± 3.40	16.24 ± 0.42
	OFCMD	0.38 ± 0.03	1.40 ± 0.03	1.42 ± 0.04	3.08 ± 0.35	1.46 ± 0.01	5.90 ± 0.12	4.61 ± 0.28	4.93 ± 0.53	16.16 ± 0.55
	HOFCMD	0.77 ± 0.18	2.07 ± 0.42	2.53 ± 4.25	6.25 ± 0.53	4.65 ± 1.76	10.91 ± 2.29	6.69 ± 0.88	5.96 ± 0.73	34.63 ± 4.61
	IncDBSCAN	1.41 ± 0.11	3.51 ± 0.38	3.64 ± 0.51	242.78 ± 56.80	4.97 ± 0.14	13.23 ± 0.53	21.12 ± 1.41	11.76 ± 0.24	36.46 ± 6.77
	SOC-TWD	0.86 ± 0.05	3.41 ± 0.43	3.40 ± 0.13	7.59 ± 0.25	9.83 ± 0.30	35.10 ± 2.07	3.40 ± 0.13	34.00 ± 5.05	48.99 ± 1.81

Table 8
Comparison of experimental results on the size of incremental data block is 10% under Situation 2.

Index	Method	Banknote	LetterABC	LetterAGI	Pageblocks	Pendigits389	Pendigits1234	Pendigits1469	Waveform	Landsat
Accuracy	TIOC-TWD	0.82 ± 0.05	**0.89 ± 0.00**	0.73 ± 0.03	0.88 ± 0.00	**0.85 ± 0.03**	**0.79 ± 0.05**	**0.76 ± 0.07**	**0.95 ± 0.01**	**0.64 ± 0.03**
	OFCMD	0.66 ± 0.09	0.89 ± 0.02	0.77 ± 0.15	0.90 ± 0.00	0.47 ± 0.13	0.70 ± 0.01	0.70 ± 0.09	0.65 ± 0.12	0.54 ± 0.02
	HOFCMD	0.68 ± 0.06	0.58 ± 0.18	0.40 ± 0.09	0.90 ± 0.00	0.59 ± 0.09	0.36 ± 0.12	0.52 ± 0.00	0.53 ± 0.22	0.35 ± 0.06
	IncDBSCAN	0.84 ± 0.00	0.67 ± 0.00	0.63 ± 0.01	0.86 ± 0.02	0.69 ± 0.04	0.72 ± 0.00	0.74 ± 0.00	0.34 ± 0.00	0.46 ± 0.00
	SOC-TWD	0.85 ± 0.00	0.85 ± 0.00	0.74 ± 0.00	0.88 ± 0.00	0.91 ± 0.00	0.84 ± 0.00	0.87 ± 0.01	0.92 ± 0.00	0.78 ± 0.00
Fmeasure	TIOC-TWD	0.86 ± 0.03	**0.90 ± 0.00**	**0.74 ± 0.04**	**0.86 ± 0.00**	**0.91 ± 0.02**	0.80 ± 0.09	**0.80 ± 0.10**	**0.59 ± 0.01**	**0.64 ± 0.02**
	OFCMD	0.62 ± 0.08	0.88 ± 0.02	0.73 ± 0.19	0.85 ± 0.00	0.36 ± 0.11	0.63 ± 0.01	0.45 ± 0.06	0.57 ± 0.12	0.43 ± 0.03
	HOFCMD	0.63 ± 0.05	0.48 ± 0.18	0.36 ± 0.12	0.85 ± 0.00	0.37 ± 0.07	0.18 ± 0.07	0.24 ± 0.01	0.28 ± 0.12	0.20 ± 0.04
	IncDBSCAN	0.86 ± 0.00	0.75 ± 0.00	0.71 ± 0.01	0.84 ± 0.01	0.78 ± 0.04	0.80 ± 0.00	0.68 ± 0.01	0.17 ± 0.00	0.41 ± 0.00
	SOC-TWD	0.91 ± 0.00	0.89 ± 0.00	0.82 ± 0.00	0.86 ± 0.00	0.88 ± 0.03	0.87 ± 0.00	0.92 ± 0.01	0.63 ± 0.00	0.70 ± 0.00
NMI	TIOC-TWD	0.54 ± 0.04	**0.74 ± 0.01**	0.58 ± 0.01	**0.11 ± 0.00**	**0.84 ± 0.03**	**0.74 ± 0.04**	0.79 ± 0.06	0.10 ± 0.01	**0.59 ± 0.01**
	OFCMD	0.05 ± 0.01	0.64 ± 0.05	0.45 ± 0.16	0.00 ± 0.00	0.38 ± 0.01	0.56 ± 0.00	0.48 ± 0.06	0.35 ± 0.04	0.46 ± 0.01
	HOFCMD	0.04 ± 0.02	0.17 ± 0.13	0.07 ± 0.08	0.00 ± 0.00	0.12 ± 0.04	0.49 ± 0.07	0.27 ± 0.08	0.01 ± 0.03	0.35 ± 0.04
	IncDBSCAN	0.55 ± 0.00	0.58 ± 0.00	0.60 ± 0.00	0.02 ± 0.00	0.74 ± 0.03	0.71 ± 0.00	0.82 ± 0.00	0.00 ± 0.00	0.53 ± 0.00
	SOC-TWD	0.79 ± 0.00	0.72 ± 0.00	0.68 ± 0.00	0.11 ± 0.00	0.77 ± 0.03	0.79 ± 0.01	0.87 ± 0.01	0.24 ± 0.00	0.63 ± 0.00
CPU(s)	TIOC-TWD	**0.22 ± 0.04**	1.52 ± 0.51	**1.14 ± 0.05**	**1.62 ± 0.22**	2.95 ± 0.06	5.26 ± 0.24	**3.68 ± 0.18**	9.45 ± 3.58	14.60 ± 0.81
	OFCMD	0.31 ± 0.03	1.17 ± 0.04	2.47 ± 3.46	5.00 ± 1.82	1.17 ± 0.04	4.76 ± 0.22	3.73 ± 0.28	3.94 ± 0.16	13.38 ± 0.64
	HOFCMD	0.44 ± 0.09	1.20 ± 0.16	6.78 ± 5.65	4.81 ± 0.51	2.12 ± 0.21	6.32 ± 0.25	5.45 ± 0.64	4.86 ± 0.80	23.53 ± 3.40
	IncDBSCAN	1.75 ± 0.09	3.34 ± 0.10	2.97 ± 0.14	285.94 ± 38.42	4.85 ± 0.25	8.15 ± 0.11	25.04 ± 2.79	14.10 ± 1.18	46.83 ± 6.25
	SOC-TWD	0.83 ± 0.08	3.29 ± 0.41	3.59 ± 0.30	7.57 ± 0.32	10.06 ± 0.33	34.61 ± 2.13	19.44 ± 1.32	34.07 ± 5.08	47.97 ± 1.98

5.2. Performance illustration with artificial data

This subsection conducts a number of experiments on artificial datasets to validate the proposed method has the ability of processing differen incremental situations as well as to deal with the dataset with arbitrary shape. 1900 objects of AD are used as the initial dataset, and 100 objects of the dataset are used as the incremental data.

Test 1: the incremental data is distributed randomly in every cluster. The clustering results of before and after the new incremental data arriving are shown in Fig. 9(a) and (b) respectively.

To compare the results in Fig. 9(a) and Fig. 9(b), both of the results show that the proposed method can determine the ground cluster correctly. Furthermore, the shape of clusters in the visual example is arbitrary. The results also show that the proposed method has the ability of clustering datasets with arbitrary shape.

Test 2: the incremental data is increased mainly in boundary region between some clusters. The clustering results of before and after the new incremental data arriving are shown in Fig. 10(a) and (b) respectively.

From Fig. 10(a), we see that there are 5 clusters, and there exists the overlapping boundary region between C_4 and C_5. It is reasonable because the density of this region is not high enough. However, when we increase objects in this region as shown in Fig. 10(b), C_4 and C_5 might be merged into one cluster. The results just reflect the inherent data structure in datasets.

Test 3: the incremental data is just a new cluster to the original dataset. The clustering results of before and after the new incremental data arriving are shown in Fig. 11(a) and (b) respectively.

To observe Fig. 11, we can see that the proposed TIOC-TWD clustering method has the ability of detecting the new structure in datasets.

Test 4: the incremental data is just increasing on the core of some clusters. The clustering results of before and after the new incremental data arriving are shown in Fig. 12(a) and (b) respectively.

Table 9
Comparison of experimental results on the size of incremental data block is 20% under Situation 2.

Index	Method	Banknote	LetterABC	LetterAGI	Pageblocks	Pendigits389	Pendigits1234	Pendigits1469	Waveform	Landsat
Accuracy	TIOC-TWD	0.81 ± 0.05	**0.89 ± 0.00**	**0.75 ± 0.03**	0.88 ± 0.00	**0.86 ± 0.03**	**0.78 ± 0.05**	**0.76 ± 0.07**	**0.95 ± 0.00**	**0.66 ± 0.04**
	OFCMD	0.82 ± 0.08	0.88 ± 0.02	0.66 ± 0.18	0.90 ± 0.00	0.57 ± 0.12	0.68 ± 0.02	0.71 ± 0.17	0.63 ± 0.16	0.51 ± 0.01
	HOFCMD	0.65 ± 0.02	0.55 ± 0.15	0.44 ± 0.08	0.90 ± 0.00	0.48 ± 0.15	0.36 ± 0.12	0.47 ± 0.10	0.44 ± 0.15	0.35 ± 0.07
	IncDBSCAN	0.84 ± 0.00	0.67 ± 0.00	0.63 ± 0.01	0.86 ± 0.02	0.69 ± 0.04	0.72 ± 0.00	0.74 ± 0.00	0.34 ± 0.00	0.46 ± 0.00
	SOC-TWD	0.85 ± 0.00	0.85 ± 0.00	0.74 ± 0.00	0.88 ± 0.00	0.91 ± 0.00	0.84 ± 0.00	0.87 ± 0.01	0.92 ± 0.00	0.78 ± 0.00
Fmeasure	TIOC-TWD	0.85 ± 0.03	**0.91 ± 0.00**	**0.76 ± 0.04**	**0.86 ± 0.00**	**0.92 ± 0.02**	0.80 ± 0.09	**0.79 ± 0.11**	**0.59 ± 0.00**	**0.65 ± 0.03**
	OFCMD	0.76 ± 0.07	0.87 ± 0.02	0.61 ± 0.22	0.85 ± 0.00	0.44 ± 0.11	0.61 ± 0.02	0.44 ± 0.11	0.56 ± 0.15	0.40 ± 0.01
	HOFCMD	0.62 ± 0.03	0.47 ± 0.14	0.39 ± 0.09	0.85 ± 0.00	0.29 ± 0.12	0.17 ± 0.06	0.22 ± 0.05	0.23 ± 0.09	0.20 ± 0.05
	IncDBSCAN	0.86 ± 0.00	0.75 ± 0.00	0.71 ± 0.01	0.84 ± 0.01	0.78 ± 0.04	0.80 ± 0.00	0.68 ± 0.01	0.17 ± 0.00	0.41 ± 0.00
	SOC-TWD	0.91 ± 0.00	0.89 ± 0.00	0.82 ± 0.00	0.86 ± 0.00	0.88 ± 0.03	0.87 ± 0.00	0.92 ± 0.01	0.63 ± 0.00	0.70 ± 0.00
NMI	TIOC-TWD	0.53 ± 0.04	**0.75 ± 0.01**	0.59 ± 0.02	**0.11 ± 0.00**	**0.86 ± 0.02**	**0.76 ± 0.04**	0.79 ± 0.06	0.10 ± 0.01	**0.60 ± 0.02**
	OFCMD	0.21 ± 0.09	0.63 ± 0.05	0.38 ± 0.15	0.00 ± 0.00	0.38 ± 0.02	0.55 ± 0.02	0.37 ± 0.13	0.34 ± 0.03	0.46 ± 0.00
	HOFCMD	0.06 ± 0.04	0.18 ± 0.10	0.10 ± 0.09	0.00 ± 0.00	0.11 ± 0.12	0.46 ± 0.07	0.21 ± 0.07	0.01 ± 0.03	0.35 ± 0.03
	IncDBSCAN	0.55 ± 0.00	0.58 ± 0.00	0.60 ± 0.00	0.02 ± 0.00	0.74 ± 0.03	0.71 ± 0.00	0.82 ± 0.00	0.00 ± 0.00	0.53 ± 0.00
	SOC-TWD	0.79 ± 0.00	0.72 ± 0.00	0.68 ± 0.00	0.11 ± 0.00	0.77 ± 0.03	0.79 ± 0.01	0.87 ± 0.01	0.24 ± 0.00	0.63 ± 0.00
CPU(s)	TIOC-TWD	**0.23 ± 0.06**	1.72 ± 0.18	**1.19 ± 0.06**	**1.61 ± 0.22**	3.09 ± 0.20	**5.43 ± 0.26**	**3.87 ± 0.22**	6.49 ± 1.41	**15.39 ± 0.74**
	OFCMD	16.52 ± 0.40	1.64 ± 0.17	6.67 ± 9.69	3.11 ± 0.52	1.62 ± 0.01	5.78 ± 0.18	4.70 ± 0.24	5.82 ± 0.36	16.52 ± 0.40
	HOFCMD	0.57 ± 0.15	2.92 ± 4.94	7.49 ± 6.98	6.57 ± 0.53	5.50 ± 1.40	10.47 ± 2.03	9.48 ± 1.68	9.51 ± 1.33	36.96 ± 10.84
	IncDBSCAN	1.75 ± 0.09	3.34 ± 0.10	2.97 ± 0.14	285.94 ± 38.42	4.85 ± 0.25	8.15 ± 0.11	25.04 ± 2.79	14.10 ± 1.18	46.83 ± 6.25
	SOC-TWD	0.83 ± 0.08	3.29 ± 0.41	3.59 ± 0.30	7.57 ± 0.32	10.06 ± 0.33	34.61 ± 2.13	19.44 ± 1.32	34.07 ± 5.08	47.97 ± 1.98

Table 10
Comparison of experimental results on the size of incremental data block is 10% under Situation 3.

Index	Method	Banknote	LetterABC	LetterAGI	Pageblocks	Pendigits389	Pendigits1234	Pendigits1469	Waveform	Landsat
Accuracy	TIOC-TWD	0.82 ± 0.05	0.87 ± 0.07	**0.73 ± 0.02**	0.88 ± 0.00	**0.76 ± 0.10**	0.82 ± 0.03	0.82 ± 0.03	**0.89 ± 0.08**	**0.69 ± 0.03**
	OFCMD	0.66 ± 0.09	0.92 ± 0.01	0.52 ± 0.01	0.90 ± 0.00	0.52 ± 0.02	0.87 ± 0.00	0.99 ± 0.01	0.71 ± 0.03	0.56 ± 0.05
	HOFCMD	0.68 ± 0.06	0.66 ± 0.14	0.50 ± 0.08	0.90 ± 0.00	0.63 ± 0.10	0.76 ± 0.07	0.44 ± 0.16	0.52 ± 0.14	0.50 ± 0.14
	IncDBSCAN	0.84 ± 0.00	0.83 ± 0.00	0.70 ± 0.01	0.87 ± 0.00	0.73 ± 0.05	0.71 ± 0.01	0.73 ± 0.01	0.34 ± 0.00	0.40 ± 0.04
	SOC-TWD	0.85 ± 0.00	0.85 ± 0.00	0.73 ± 0.01	0.88 ± 0.00	0.91 ± 0.00	0.84 ± 0.00	0.82 ± 0.02	0.92 ± 0.00	0.79 ± 0.00
Fmeasure	TIOC-TWD	0.85 ± 0.03	0.86 ± 0.10	0.73 ± 0.04	**0.86 ± 0.01**	0.75 ± 0.13	0.85 ± 0.05	**0.88 ± 0.02**	0.53 ± 0.05	**0.67 ± 0.02**
	OFCMD	0.62 ± 0.08	0.90 ± 0.00	0.45 ± 0.00	0.85 ± 0.00	0.43 ± 0.02	0.85 ± 0.00	0.64 ± 0.01	0.63 ± 0.02	0.48 ± 0.06
	HOFCMD	0.63 ± 0.05	0.52 ± 0.14	0.48 ± 0.09	0.85 ± 0.00	0.40 ± 0.07	0.50 ± 0.07	0.20 ± 0.07	0.33 ± 0.10	0.28 ± 0.10
	IncDBSCAN	0.86 ± 0.00	0.87 ± 0.00	0.80 ± 0.00	0.84 ± 0.00	0.82 ± 0.05	0.80 ± 0.01	0.67 ± 0.01	0.17 ± 0.03	0.35 ± 0.03
	SOC-TWD	0.91 ± 0.00	0.86 ± 0.00	0.82 ± 0.00	0.86 ± 0.00	0.89 ± 0.03	0.87 ± 0.00	0.88 ± 0.01	0.63 ± 0.00	0.70 ± 0.00
NMI	TIOC-TWD	0.54 ± 0.04	0.69 ± 0.04	0.57 ± 0.04	**0.10 ± 0.03**	0.71 ± 0.07	**0.77 ± 0.01**	**0.83 ± 0.02**	0.08 ± 0.05	**0.62 ± 0.02**
	OFCMD	0.05 ± 0.01	0.68 ± 0.03	0.27 ± 0.02	0.00 ± 0.00	0.42 ± 0.03	0.73 ± 0.00	0.32 ± 0.01	0.39 ± 0.01	0.44 ± 0.02
	HOFCMD	0.04 ± 0.02	0.26 ± 0.12	0.13 ± 0.10	0.00 ± 0.00	0.13 ± 0.04	0.36 ± 0.08	0.16 ± 0.02	0.08 ± 0.05	0.38 ± 0.08
	IncDBSCAN	0.55 ± 0.00	0.70 ± 0.00	0.65 ± 0.00	0.02 ± 0.00	0.76 ± 0.04	0.71 ± 0.01	0.80 ± 0.01	0.00 ± 0.00	0.47 ± 0.04
	SOC-TWD	0.79 ± 0.00	0.62 ± 0.00	0.69 ± 0.02	0.11 ± 0.00	0.76 ± 0.03	0.78 ± 0.00	0.82 ± 0.01	0.24 ± 0.00	0.64 ± 0.00
CPU(s)	TIOC-TWD	**0.22 ± 0.04**	**1.20 ± 0.08**	1.48 ± 0.07	**2.42 ± 0.64**	2.51 ± 0.14	6.03 ± 0.30	4.57 ± 0.17	13.16 ± 2.89	**14.25 ± 0.59**
	OFCMD	0.28 ± 0.02	1.22 ± 0.06	1.23 ± 0.08	3.79 ± 2.11	1.23 ± 0.01	3.91 ± 0.12	3.65 ± 0.36	4.01 ± 0.18	20.52 ± 0.47
	HOFCMD	0.44 ± 0.09	1.29 ± 0.20	3.25 ± 3.52	5.39 ± 0.46	2.05 ± 0.22	6.30 ± 1.09	4.84 ± 0.66	4.42 ± 0.55	25.58 ± 4.50
	IncDBSCAN	1.73 ± 0.09	3.22 ± 0.22	2.88 ± 0.12	250.76 ± 25.33	4.09 ± 0.17	10.97 ± 0.93	15.68 ± 2.26	12.05 ± 1.93	20.01 ± 2.15
	SOC-TWD	0.84 ± 0.08	3.25 ± 0.80	3.31 ± 0.11	7.49 ± 0.29	9.98 ± 0.29	34.27 ± 1.88	21.08 ± 0.93	34.23 ± 5.13	59.25 ± 2.19

To observe Fig. 12, we see that the proposed TIOC-TWD clustering method has the ability of splitting a big cluster into small clusters. The new clusters might has the overlapping regions, which just reveal the underlying structure in the dataset.

5.3. Results of comparison experiments

This subsection describes experiments on some of the UCI datasets [44] with the proposed approach TIOC-TWD, the SOC-TWD algorithm, the IncDBSCAN algorithm [8], the OFCMD algorithm and HOFCMD algorithm [18]; the accuracy, F-measure and NMI indices are evaluated there. The SOC-TWD algorithm is the only one which is not an incremental clustering approach, and it is used to as a comparison to incremental approaches. There are four parameters $\delta, \alpha, \beta, \lambda$ used in our method; the compared algorithms are also depended on some parameters. For example, the OFCMD algorithm has to set the number of clusters, the number of candidates to determine the medoids, the size of the data chunks, the decay factor and so on. The parameters used in the compared algorithms are set as in the original references.

To simulate the incremental environment, 60% of each static UCI dataset is deemed as the initial dataset, and the rest of dataset is the incremental dataset. Algorithms 1 and 2 are carried out on the initial dataset, and the CIA-TWD is implemented on the incremental dataset. For the experiments described in the following, the results are always averaged over all the 10 runs, and the standard deviation variances are also reported in results.

On the one hand, considering the data might be increasing continuously in the real incremental application environment, we need to simulate the continued incremental data. Therefore, each incremental dataset is divided into a plurality of incremental data blocks; then the CIA-TWD algorithm is executed on each block until there is no new block.

On the other hand, considering the underlying structures in the new incremental data are unknown, we need to simulate the different situations to valuate the performance of the proposed method. Because the new incremental objects might belong to all known clusters, or belong to parts of known clusters, or form new clusters, we design three experiments corresponding to these three different situations.

Therefore, for each run and for each dataset, the 40% of incremental dataset is divided into different size of incremental data blocks. When the size of incremental data block is 10%, the incremental dataset is divided into 4 blocks; and when the size of incremental data block is 20%, the incremental dataset is divided into 2 blocks. The accuracy, F-measure, NMI index, and the CPU time are recorded in each running.

Situation 1: the incremental data are distributed randomly on most of clusters. That is, the new incremental objects might belong to all known clusters. Tables 6 and 7 show the comparison results on the size of incremental data block is 10% and 20% of the original dataset respectively.

From Table 6, we see that: the accuracies of proposed method are higher than that of the compared incremental algorithms in Banknote, Pendigits389, Pendigits1469, Waveform and Landsat datasets, and the accuracies of proposed method are very near to the best of compared algorithms in other datasets. Thus, the performance of the proposed approach is roughly equal to the compared algorithms in terms of the accuracy index. In terms of the F-measure and NMI indices, the performance of proposed method is much better than that of the compared algorithms in most of datasets. Though the CPU time of the proposed algorithm in some datasets are slightly more than that of the compared algorithms; for Page blocks dataset, the CPU time of the proposed algorithm is 1.48 ± 0.25, but the CPU time of the compared algorithms are $20.02 \pm 8.33, 5.13 \pm 0.57, 242.78 \pm 56.80, 7.59 \pm 0.25$, respectively. The advantage is obvious on computing time or on the standard deviation. Of course, the CPU time of TIOC-TWD is much less than the static algorithm SOC-TWD, and the difference is more obvious on the big datasets. It is interesting that the indices of static algorithm are not always better than the incremental method, which shows the necessity of developing incremental methods from another perspective. Observe the results in Table 7, we can find almost the same conclusions as the above for these methods.

When we compare results in Tables 6 and 7, we find that the performance of proposed approach is better on the larger size of incremental data block with higher computing time though there is no algorithm is absolutely best in every index. Generally speaking, when the incremental data objects are distributed randomly, the performance of the proposed approach is slightly better than that of the compared methods.

Situation 2: the incremental data are distributed randomly on part of clusters. That is, the new incremental objects might belong to some specific known clusters. Table 8 and Table 9 show the comparison results on the size of incremental data block is 10% and 20% of the original dataset respectively.

In experiments, the incremental data objects come from the class “2” of the Banknote, the class “B” and “C” of LetterABC, the class “G” and “I” of the LetterAGI, the class “2” and “5” of Page blocks, the class “8” and “9” of Pendigits389, the class “3” and “4” of Pendigits1234, the class “6” and “9” of Pendigits1469, the class “1” and “2” of Waveform, the class “4”, “5” and “7” of Landsat respectively.

Observing the results in Tables 8 and 9, we can see that the performance of proposed approach is better when the size of incremental data block is bigger. Under this situation, the proposed approach has higher performance than the compared algorithms at the most of cases, especially on the F-measure and NMI indices.

Situation 3: the incremental data to produce new clusters. That is, the new incremental objects might not belong to any known clusters. Table 10 and Table 11 show the comparison results on the size of incremental data block is 10% and 20% of the original dataset respectively.

In experiments, the 40% incremental data objects are composed of several entire classes in a dataset to simulate Situation 3. The incremental data objects come from the whole class “2” of Banknote, the class “C” of LetterABC, the class “G” of LetterAGI, the class “1” and “2” of Page blocks, the class “9” of Pendigits389, the class “1” of Pendigits1234, the class “9” of Pendigits1469, the class “1” and “2” of Waveform, and the class of “1”, “2” and “3” of Landsat, respectively.

Comparing results in Tables 10 and 11, we see that it is not very distinct the performance of proposed approach between the different sizes of incremental data block under Situation 3; which shows the stability of the proposed approach to detect new patterns in

Table 11
Comparison of experimental results on the size of incremental data block is 20% under Situation 3.

Index	Method	Banknote	LetterABC	LetterAGI	Pageblocks	Pendigits389	Pendigits1234	Pendigits1469	Waveform	Landsat
Accuracy	TIOC-TWD	0.81 ± 0.05	0.85 ± 0.10	**0.73 ± 0.00**	0.88 ± 0.00	**0.77 ± 0.10**	**0.82 ± 0.04**	0.77 ± 0.06	**0.88 ± 0.09**	**0.68 ± 0.03**
	OFCMD	0.82 ± 0.08	0.94 ± 0.01	0.46 ± 0.06	0.90 ± 0.00	0.51 ± 0.01	0.82 ± 0.08	0.99 ± 0.01	0.71 ± 0.13	0.60 ± 0.03
	HOFCMD	0.65 ± 0.02	0.73 ± 0.15	0.44 ± 0.11	0.90 ± 0.00	0.53 ± 0.10	0.71 ± 0.07	0.36 ± 0.13	0.62 ± 0.10	0.51 ± 0.17
	IncDBSCAN	0.84 ± 0.00	0.83 ± 0.00	0.70 ± 0.01	0.87 ± 0.00	0.73 ± 0.05	0.71 ± 0.01	0.73 ± 0.01	0.34 ± 0.00	0.40 ± 0.04
	SOC-TWD	0.85 ± 0.00	0.85 ± 0.00	0.73 ± 0.01	0.88 ± 0.00	0.91 ± 0.00	0.84 ± 0.00	0.82 ± 0.02	0.92 ± 0.00	0.79 ± 0.00
Fmeasure	TIOC-TWD	0.85 ± 0.03	0.83 ± 0.13	0.72 ± 0.01	0.85 ± 0.01	0.77 ± 0.13	**0.85 ± 0.06**	**0.82 ± 0.10**	0.53 ± 0.04	**0.67 ± 0.02**
	OFCMD	0.76 ± 0.07	0.91 ± 0.00	0.39 ± 0.05	0.85 ± 0.00	0.42 ± 0.02	0.78 ± 0.09	0.64 ± 0.03	0.60 ± 0.10	0.52 ± 0.03
	HOFCMD	0.62 ± 0.03	0.59 ± 0.15	0.42 ± 0.11	0.85 ± 0.00	0.29 ± 0.10	0.44 ± 0.06	0.16 ± 0.06	0.37 ± 0.07	0.28 ± 0.10
	IncDBSCAN	0.86 ± 0.00	0.87 ± 0.00	0.80 ± 0.00	0.84 ± 0.00	0.82 ± 0.05	0.80 ± 0.01	0.67 ± 0.01	0.17 ± 0.03	0.35 ± 0.03
	SOC-TWD	0.91 ± 0.00	0.86 ± 0.00	0.82 ± 0.00	0.86 ± 0.00	0.89 ± 0.03	0.87 ± 0.00	0.88 ± 0.01	0.63 ± 0.00	0.70 ± 0.00
NMI	TIOC-TWD	0.53 ± 0.04	0.67 ± 0.05	0.56 ± 0.02	**0.08 ± 0.03**	0.71 ± 0.08	**0.77 ± 0.01**	0.80 ± 0.05	0.10 ± 0.07	**0.61 ± 0.03**
	OFCMD	0.21 ± 0.09	0.67 ± 0.01	0.28 ± 0.01	0.00 ± 0.00	0.42 ± 0.03	0.66 ± 0.06	0.40 ± 0.04	0.32 ± 0.06	0.45 ± 0.01
	HOFCMD	0.06 ± 0.04	0.29 ± 0.14	0.09 ± 0.06	0.00 ± 0.00	0.04 ± 0.04	0.28 ± 0.05	0.15 ± 0.03	0.10 ± 0.06	0.32 ± 0.05
	IncDBSCAN	0.55 ± 0.00	0.70 ± 0.00	0.65 ± 0.00	0.02 ± 0.00	0.76 ± 0.04	0.71 ± 0.01	0.80 ± 0.01	0.00 ± 0.00	0.47 ± 0.04
	SOC-TWD	0.79 ± 0.00	0.62 ± 0.00	0.69 ± 0.02	0.11 ± 0.00	0.76 ± 0.03	0.78 ± 0.00	0.82 ± 0.01	0.24 ± 0.00	0.64 ± 0.00
CPU(s)	TIOC-TWD	**0.23 ± 0.06**	**1.53 ± 0.12**	1.47 ± 0.06	**3.21 ± 1.28**	2.53 ± 0.10	6.25 ± 0.39	**4.49 ± 0.27**	15.38 ± 4.26	**14.41 ± 0.46**
	OFCMD	0.44 ± 0.04	1.93 ± 0.14	1.92 ± 0.16	3.66 ± 0.35	1.63 ± 0.02	4.98 ± 0.13	4.72 ± 0.30	6.32 ± 0.48	16.39 ± 0.40
	HOFCMD	0.56 ± 0.14	1.82 ± 0.33	1.18 ± 0.15	6.10 ± 0.41	3.47 ± 0.46	10.69 ± 1.70	11.32 ± 2.36	8.51 ± 1.12	45.49 ± 10.74
	IncDBSCAN	1.73 ± 0.09	3.22 ± 0.22	2.88 ± 0.12	250.76 ± 25.33	4.09 ± 0.17	10.97 ± 0.93	15.68 ± 2.26	12.05 ± 1.93	20.01 ± 2.15
	SOC-TWD	0.84 ± 0.08	3.25 ± 0.80	3.31 ± 0.11	7.49 ± 0.29	9.98 ± 0.29	34.27 ± 1.88	21.08 ± 0.93	34.23 ± 5.13	59.25 ± 2.19

Please cite this article in press as: H. Yu et al., A tree-based incremental overlapping clustering method using the three-way decision theory, Knowl. Based Syst. (2015), http://dx.doi.org/10.1016/j.knosys.2015.05.028

some sense. Under this situation, the proposed approach has higher performance on NMI index and CPU time than the compared algorithms in most of cases; the performance of proposed approach in the accuracy or F-measure is very close to the best even if it is not the best.

To sum up, the performance of proposed approach is better than the compared algorithms in most of cases. Moreover, the proposed approach has the following advantages in contrast with other methods including the compared algorithms: the result of clustering is represented by three-way decisions, that is a cluster is composed of a positive region and a boundary region, which is helpful to make further investigation; the time cost of the proposed method is not always best, it is still very valuable especially in applications, because the proposed method does not define the number of clusters in advance as other methods.

6. Conclusions

Existing clustering approaches are either restricted to crisp clustering or static datasets. In order to develop an approach to deal with overlapping clustering as well as incremental clustering, this paper proposed a new tree-based incremental overlapping clustering method using the three-way decision theory, called TIOC-TWD.

This paper first introduced three-way decision clustering to represent the overlapping clustering as well as crisp clustering, and described the problem of incremental overlapping clustering; and proposed notions of representative points and the similarity between representative regions. Then, the paper introduced a new searching tree based on representative points, which can not only enhance the relevance of the search result but it can also save the computation time. Besides, the paper devised three-way strategies to update efficiently the clustering after multiple objects increased. Moreover, the proposed method does not need to define the number of cluster in advance, it can dynamically determine the number of clusters. The above characteristics make the TIOC-TWD appropriate for handling overlapping clustering in applications where the data is increasing.

This paper conducted experiments to illustrate the salient features of the proposed algorithm and evaluate its performance. The experimental results show that the proposed method not only can identify clusters of arbitrary shapes, but also can merge small clusters into the big one when the data changes; the proposed method can detect new clusters which might be the result of splitting or new patterns. More results of comparison experiments show that the proposed method has better performance especially on F-measure and NMI indices than the compared methods. The further analysis of parameters will be our planned future work.

Acknowledgment

This work was supported in part by the National Natural Science Foundation of China under Grant Nos. 61379114 & 61272060.

References

[1] A.K. Jain, Data clustering: 50 years beyond k-means, Pattern Recogn. Lett. 31 (8) (2010) 651–666.
[2] G. Peters, F. Crespo, P. Lingras, R. Weber, Soft clustering-fuzzy and rough approaches and their extensions and derivatives, Int. J. Approx. Reason. 54 (2) (2013) 307–322.
[3] H.L. Sun, J.B. Huang, X. Zhang, J. Liu, D. Wang, H.L. Liu, J.H. Zou, Q.B. Song, Incorder: incremental density-based community detection in dynamic networks, Knowl.-Based Syst. 72 (2014) 1–12.
[4] S. Lee, G. Kim, S. Kim, Self-adaptive and dynamic clustering for online anomaly detection, Expert Syst. Appl. 38 (12) (2011) 14891–14898.
[5] A. Pérez-Suárez, J.F. Martńez-Trinidad, J.A. Carrasco-Ochoa, J.E. Medina-Pagola, A new overlapping clustering algorithm based on graph theory, Advances in Artificial Intelligence, vol. 7629, Springer, 2013, pp. 61–72.
[6] A.A. Abbasi, M. Younis, A survey on clustering algorithms for wireless sensor networks, Comput. Commun. 30 (14) (2007) 2826–2841.
[7] H. Yu, C. Zhang, F. Hu, An incremental clustering approach based on three-way decisions, in: Rough Sets and Current Trends in Computing, Springer, 2014, pp. 152–159.
[8] M. Ester, H.-P. Kriegel, J. Sander, M. Wimmer, X.W. Xu, Incremental clustering for mining in a data warehousing environment, in: VLDB, vol. 98, 1998, pp. 323–333.
[9] N. Goyal, P. Goyal, K. Venkatramaiah, P.C. Deepak, P.S. SANNOP, An efficient density based incremental clustering algorithm in data warehousing environment, in: 2009 International Conference on Computer Engineering and Applications, IPCSIT, vol. 2, 2009, pp. 482–486.
[10] B.K. Patra, O. Ville, R. Launonen, S. Nandi, K.S. Babu, Distance based incremental clustering for mining clusters of arbitrary shapes, in: Pattern Recognition and Machine Intelligence, Springer, 2013, pp. 229–236.
[11] R. Ibrahim, N. Ahmed, N.A. Yousri, M.A. Ismail, Incremental mitosis: discovering clusters of arbitrary shapes and densities in dynamic data, 11th International Conference on Machine Learning and Applications, vol. 1, IEEE Computer Society, 2012, pp. 102–107.
[12] H.Z. Ning, W. Xu, Y. Chi, Y.H. Gong, T.S. Huang, Incremental spectral clustering by efficiently updating the eigen-system, Pattern Recogn. 43 (1) (2010) 113–127.
[13] R.G. Pensa, D. Ienco, R. Meo, Hierarchical co-clustering: off-line and incremental approaches, Data Mining Knowl. Discovery 28 (1) (2014) 31–64.
[14] K.M. Hammouda, M.S. Kamel, Efficient phrase-based document indexing for web document clustering, IEEE Trans. Knowl. Data Eng. 16 (10) (2004) 1279–1296.
[15] R. Gil-García, A. Pons-Porrata, Dynamic hierarchical algorithms for document clustering, Pattern Recogn. Lett. 31 (6) (2010) 469–477.
[16] A. Pérez-Suárez, J.F. Martínez-Trinidad, J.A. Carrasco-Ochoa, J.E. Medina-Pagola, An algorithm based on density and compactness for dynamic overlapping clustering, Pattern Recogn. 46 (11) (2013) 3040–3055.
[17] E. Lughofer, A dynamic split-and-merge approach for evolving cluster models, Evolv. Syst. 3 (3) (2012) 135–151.
[18] N. Labroche, Online fuzzy medoid based clustering algorithms, Neurocomputing 126 (2014) 141–150.
[19] J.F. Peters, A. Skowron, Z. Suraj, W. Rzsa, M. Borkowski, Clustering: a rough set approach to constructing information granules, in: Soft Computing and Distributed Processing, 2002, pp. 57–61.
[20] D. Parmar, T. Wu, J. Blackhurst, Mmr: an algorithm for clustering categorical data using rough set theory, Data Knowl. Eng. 63 (3) (2007) 879–893.
[21] H.M. Chen, T.R. Li, C. Luo, S.J. Horng, G.Y. Wang, A rough set-based method for updating decision rules on attribute values' coarsening and refining, IEEE Trans. Knowl. Data Eng. 26 (12) (2014) 2886–2899.
[22] H.M. Chen, T.R. Li, C. Luo, S.J. Horng, G.Y. Wang, A decision-theoretic rough set approach for dynamic data mining, IEEE Trans. Fuzzy Syst. (2015). http://dx.doi.org/10.1109/TFUZZ.2014.238787.
[23] G. Peters, R. Weber, R. Nowatzke, Dynamic rough clustering and its applications, Appl. Soft Comput. 12 (10) (2012) 3193–3207.
[24] P. Lingras, G. Peters, F. Crespo, R. Weber, Soft clustering – fuzzy and rough approaches and their extensions and derivatives, Int. J. Approx. Reason. 54 (2) (2013) 307–322.
[25] Y.Y. Yao, The superiority of three-way decisions in probabilistic rough set models, Inform. Sci. 181 (6) (2011) 1080–1096.
[26] Y.Y. Yao, An outline of a theory of three-way decisions, in: Rough Sets and Current Trends in Computing, Springer, 2012, pp. 1–17.
[27] C. Luo, T.R. Li, H.M. Chen, L.X. Lu, Fast algorithms for computing rough approximations in set-valued decision system while updating criteria values, Inform. Sci. 299 (2015) 221–242.
[28] C. Luo, T.R. Li, H.M. Chen, Dynamic maintenance of approximations in set-valued ordered decision systems under the attribute generalization, Inform. Sci. 257 (2014) 210–228.
[29] N. Azam, J.T. Yao, Analyzing uncertainties of probabilistic rough set regions with game-theoretic rough sets, Int. J. Approx. Reason. 55 (1) (2014) 142–155.
[30] B. Zhou, Y.Y. Yao, J.G. Luo, Cost-sensitive three-way email spam filtering, J. Intell. Inform. Syst. 42 (1) (2014) 19–45.
[31] D.C. Liang, D. Liu, A novel risk decision-making based on decision-theoretic rough sets under hesitant fuzzy information, IEEE Trans. Fuzzy Syst. PP (99) (2014) 1–11.
[32] D.C. Liang, D. Liu, Systematic studies on three-way decisions with interval-valued decision-theoretic rough sets, Inform. Sci. 276 (2014) 186–203.
[33] H. Yu, Y. Wang, P. Jiao, A three-way decisions approach to density-based overlapping clustering, in: Transactions on Rough Sets XVIII, Springer, 2014, pp. 92–109.
[34] H. Yu, Z.G. Liu, G.Y. Wang, An automatic method to determine the number of clusters using decision-theoretic rough set, Int. J. Approx. Reason. 55 (1) (2014) 101–115.
[35] P. Lingras, R. Yan, Interval clustering using fuzzy and rough set theory, in: Processing NAFIPS'04: IEEE Annual Meeting of the Fuzzy Information, vol. 2, 2004, pp. 780–784.
[36] P. Lingras, C. West, Interval set clustering of web users with rough k-means, J. Intell. Inform. Syst. 23 (1) (2004) 5–16.

[37] Y.Y. Yao, P. Lingras, R.Z. Wang, D.Q. Miao, Interval set cluster analysis: a re-formulation, in: Rough Sets, Fuzzy Sets, Data Mining and Granular Computing, Springer, 2009, pp. 398–405.
[38] M. Chen, D.Q. Miao, Interval set clustering, Expert Syst. Appl. 38 (4) (2011) 2923–2932.
[39] E. Ikonomovska, J. Gama, S. Džeroski, Online tree-based ensembles and option trees for regression on evolving data streams, Neurocomputing 150 (2014) 458–470.
[40] C.W. Tsai, K.W. Huang, M.C. Chiang, C.S. Yang, A fast tree-based search algorithm for cluster search engine, in: IEEE International Conference on Systems, Man and Cybernetics, 2009, pp. 1603–1608.
[41] L. Breiman, J. Friedman, C.J. Stone, R. Olshen, Classification and regression trees, AMC 10 (1984) 12.
[42] B. Larsen, C. Aone, Fast and effective text mining using linear-time document clustering, in: Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 1999, pp. 16–22.
[43] A. Strehl, J. Ghosh, Cluster ensembles—a knowledge reuse framework for combining multiple partitions, J. Machine Learn. Res. 3 (2003) 583–617.
[44] Uci, 2014. <http://archive.ics.uci.edu/ml>.