Pattern Recognition 43 (2010) 3130–3143

Contents lists available at ScienceDirect

Pattern Recognition

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

On cluster tree for nested and multi-density data clustering ☆

Xutao Li [a], Yunming Ye [a], Mark Junjie Li [b], Michael K. Ng [c,*]

[a] Department of Computer Science, Shenzhen Graduate School, Harbin Institute of Technology, China
[b] Institute of Advanced Computing and Digital Engineering, Centre for High Performance Computing Technology, Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen, PR China
[c] Department of Mathematics, Hong Kong Baptist University, Kowloon Tong, Hong Kong

ARTICLE INFO

Article history:
Received 10 August 2009
Received in revised form
30 January 2010
Accepted 25 March 2010

Keywords:
Hierarchical clustering
Multi-densities
Cluster tree
k-Means-type algorithm

ABSTRACT

Clustering is one of the important data mining tasks. Nested clusters or clusters of multi-density are very prevalent in data sets. In this paper, we develop a hierarchical clustering approach—a cluster tree to determine such cluster structure and understand hidden information present in data sets of nested clusters or clusters of multi-density. We embed the agglomerative *k*-means algorithm in the generation of cluster tree to detect such clusters. Experimental results on both synthetic data sets and real data sets are presented to illustrate the effectiveness of the proposed method. Compared with some existing clustering algorithms (DBSCAN, X-means, BIRCH, CURE, NBC, OPTICS, Neural Gas, Tree-SOM, EnDBSAN and LDBSCAN), our proposed cluster tree approach performs better than these methods.

1. Introduction

Clustering data in databases is an important task in real applications of data mining and knowledge discovery. Clustering is the process of partition of a data set into clusters so that similar objects are put into the same cluster while dissimilar objects are put into different clusters. Many clustering techniques have been developed and studied. Based on different theories and methodologies, their types include partitional, hierarchical, density-based, grid-based and model-based clustering algorithms, for instance, see the survey paper [1].

One of the challenges in data clustering is to detect nested clusters or clusters of multi-density in a data set. Multi-density clusters refer to the clusters that are formed in different densities. Nested clusters means a cluster is composed of several sub-clusters, for example, cluster A contains two clusters B and C, i.e., $A = B \cup C$. Nested clusters or clusters of multi-density are very prevalent in real data sets. For instance, in a geographical data set, a district may contain several cities and each city is further composed of different blocks. Each block have different areas. In a text data set, a collection of documents belongs to several topics, where each topic is also composed of several different sub-topics. There are several number of documents for each sub-topics.

In the literature, there are some clustering algorithms that have been proposed to detect nested clusters or clusters of multi-density in a data set. Erotz et al. [2] developed a density based clustering algorithm based on the similarity between points by looking at the number of nearest neighbours that two points share. However, its clustering performance is not good enough for multi-density data sets. Karypis et al. [3] proposed and developed a hierarchical clustering algorithm (CHAMELEON) to measure the similarity of two clusters based on a dynamic model. In the clustering process, two clusters are merged only if the inter-connectivity and proximity between two clusters are high enough relative to the internal inter-connectivity of the clusters and proximity of objects within the clusters. However, its computational complexity is quite high for a large data set. Zhao et al. [4] studied a density-grid based clustering algorithm for clustering data sets containing clusters of various densities. Its clustering performance is not good for low density clusters. Recently, there are several density-based clustering algorithms: OPTICS [5], NBC (neighborhood based clustering) [6], EnDBSCAN [7], LDBSCAN [8], GDCIC [9] and GMDBSCAN [10] for clustering multi-density data sets. The OPTICS cannot produce a clustering result explicitly but create an augmented ordering of data set representing the density-based structures, from which embedded and multi-density clusters can be identified. However, its computational complexity can be very high for large data and some nested clusters cannot be identified well. The NBC is an extension to DBSCAN based on a neighborhood-based density factor, which makes it capable of dealing with multi-density clusters. But its performance is highly sensitive to the density factor. The

☆ Research supported in part by RGC 201508 and HKBU FRGs.
* Corresponding author. Tel.: +852 3411 7317; fax: +852 3411 5811.
E-mail addresses: xutaolee08@gmail.com (X. Li), yeyunming@hit.edu.cn (Y. Ye), jj.li@sub.siat.ac.cn (M.J. Li), mng@math.hkbu.edu.hk (M.K. Ng).

0031-3203/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2010.03.020

LDBSCAN extends the *local density factor* [11] to encode the local-density information, which makes it capable of detecting multi-density clusters. However, its parameters are not easy to set for high dimensional data sets. Both EnDBSCAN and GMDBSCAN are variants of DBCSAN for identifying multi-density clusters, but their clustering results are highly sensitive to the parameter settings. GDCIC is a grid-based density clustering algorithm, which is not able to handle high dimensional data sets because the size of grid is not easy to determine.

In this paper, we demonstrate the effectiveness of the multi-level approach to discovering hierarchical structured clusters in data sets of nested clusters or clusters of multi-density. We present a hierarchical clustering approach—a cluster tree to determine such cluster structure and understand hidden information present in data sets. We embed the agglomerative k-means algorithm in the generation of cluster tree to detect nested clusters and clusters of different densities at different levels. One of the interesting property of the agglomerative k-means algorithm is that it can discover cluster structure with different densities of clusters.

The process of clustering is conducted as follows. Starting from a given data set, we first use the agglomerative k-means algorithm to cluster objects in the data set. The agglomerative k-means algorithm can determine the number of clusters to be produced at this level. Then we use cluster validation techniques to evaluate the clusters generated at this level. If we find any separate clusters, we identify them as isolated clusters and we do not need to partition it furthermore. For other clusters that have nested structure or different densities, we use the agglomerative k-means algorithm to further partition it. We repeat this process to create a tree of clusters from which interesting clusters can be gradually identified. Experimental results on both synthetic data sets and real data sets are presented to illustrate the effectiveness of the proposed method. Compared with some existing clustering algorithms (DBSCAN, X-means, BIRCH, CURE, NBC, OPTICS, Neural Gas, Tree-SOM, EnDBSAN and LDBSCAN), our proposed cluster tree approach performs better than these clustering methods.

The outline of this paper is as follows. In Section 2, we present the framework of a cluster tree, and the proposed algorithm. In Section 3, experimental results are demonstrated to show the effectiveness of the proposed method. Finally, some concluding remarks are given in Section 4.

2. A cluster tree

Our approach to clustering a data set is to use the agglomerative k-means algorithm and various cluster validation techniques to build a tree of clusters from which interesting nested clusters and clusters of different densities can be discovered. In this section, we define a cluster tree and discuss how it is built from a data set.

2.1. Definitions

Let $\boldsymbol{X}$ be a data set containing N objects, that is, $\boldsymbol{X}=\{x_1,x_2,\ldots,x_N\}$.

Definition 1. An p-clustering of $\boldsymbol{X}$ is a partition of $\boldsymbol{X}$ into p subsets (clusters), which satisfies $C_i\neq\phi$, $i=1, 2,\ldots, p$, $\bigcup_{i=1}^{p} C_i=\boldsymbol{X}$, and $C_i\cap C_j=\phi$, $i\neq j$ for $i, j=1, 2, \ldots, p$.

In addition, the objects in a cluster C_i are "more similar" to each other and "less similar" to objects of other clusters.

Definition 2. A clustering S with k clusters is said to be nested in another clustering T, which contains $r(<k)$ clusters, if for any cluster C_i in S, there is a cluster C_j in T such that $C_i\subseteq C_j$. And there exists at least one cluster in S, which holds $C_i\subset C_j$ and $C_i\neq C_j$.

Definition 3. A cluster tree is a sequence of q nested clusterings, $\{S_1,S_2,\ldots,S_q\}$, so that for any i,j with $i<j$ and for any $C_j\in S_j$, there is $C_i\in S_i$ such that $C_j\subseteq C_i$.

As an example, Fig. 1 shows a data set containing six clusters in 2-dimensional plane. At the high level, the whole data set is partitioned into two nested clusterings. We see that the first clustering S_1 contains three groups of objects. At this level, the first group A (left) contains only one cluster, the second group B (middle) contains three clusters, and the third group C (right) contains two clusters. $S_1=\{A,B,C\}$. At the bottom level, for B, we can further partition it into three clusters $\{B_1, B_2, B_3\}$. For C, there are two clusters $\{C_1, C_2\}$. The second clustering S_2 is given by $\{A, B_1, B_2, B_3, C_1, C_2\}$, and S_2 is nested in S_1. We call B and C composite clusters and others atomic clusters.

Our aim is to deal with data sets with nested structure and multi-density clusters. In contrast to the bottom-up hierarchical clustering methods, we use a top-down approach to building a cluster tree from data. We consider the whole data set, and generate a tree of clusters. In the tree, a node containing children is a composite cluster while all leaves are atomic clusters. In this approach, we have to make two decisions at each node to proceed the clustering process. That is, to decide whether a node being considered is a composite or an atomic cluster and to determine the number of clusters to be created from it if the node is a composite cluster. In the following two subsections, the cluster validation techniques and the agglomerative k-means algorithm are employed to obtain such knowledge and make decisions.

2.2. The number of clusters

Let us first review the agglomerative k-means algorithm [12]. By introducing a penalty term to the objective function, the algorithm can make the clustering process not sensitive to the initial cluster centers. The algorithm can produce more consistent clustering results from different sets of initial cluster centers.

Given a set $\boldsymbol{S}$ of n objects denoted as $\{x_1, x_2, \ldots, x_n\}$, where each object x_i is represented as m numerical attributes indicated as $[x_{i,1}, x_{i,2}, \ldots, x_{i,m}]^T$. The following objective function in the agglomerative k-means algorithm is used to partition n objects into k

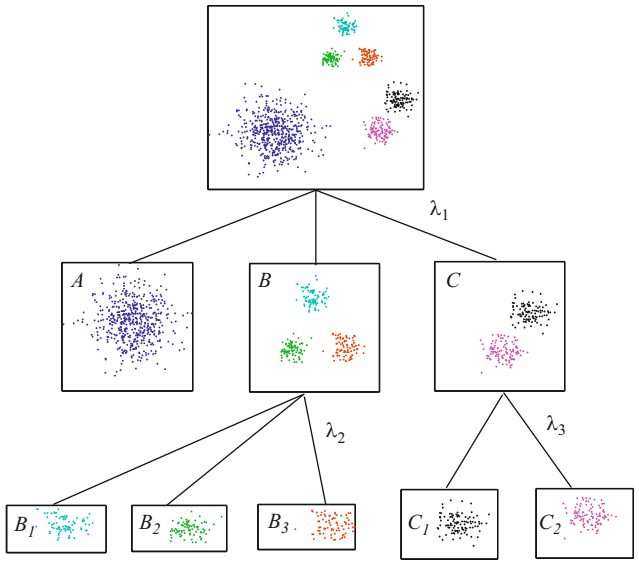

Fig. 1. An example for illustrating the idea of building a cluster tree.

clusters:

$$\min P(U,Z) = \sum_{j=1}^{k}\sum_{i=1}^{n} u_{i,j}D_{i,j} + \lambda \sum_{j=1}^{k}\sum_{i=1}^{n} u_{i,j}\log u_{i,j} \quad (1)$$

subject to

$$\sum_{j=1}^{k} u_{i,j} = 1, \quad u_{i,j} \in (0,1], \quad 1 \le i \le n, \quad (2)$$

where $U = [u_{i,j}]$ represents an n-by-k membership matrix, $u_{i,j}$ is the association degree of membership of the i th object x_i to the j th cluster z_j, $Z = [z_1, z_2, \ldots, z_k]^T$ is a k-by-m matrix of the cluster centers, and $D_{i,j} = \sum_{l=1}^{m}(z_{j,l} - x_{i,l})^2$, is the square of the Euclidean norm measure between the j th cluster center and the i th object.

By solving the optimizing problem above, the updating formulas for the center matrix Z and the membership matrix U are given as follows:

$$z_{j,l} = \frac{\sum_{i=1}^{n} u_{i,j}\, x_{i,l}}{\sum_{i=1}^{n} u_{i,j}}, \quad (3)$$

$$u_{i,j} = \frac{\exp\left(\frac{-D_{i,j}}{\lambda}\right)}{\sum_{l=1}^{k}\exp\left(\frac{-D_{i,l}}{\lambda}\right)}. \quad (4)$$

The penalty term in Eq. (1) is a weighted negative entropy of objects-to-clusters membership. When the penalty factor λ is small, minimization of the objective function mainly depends on the first term and the centers adjust their locations to minimize the dispersions of clusters; however, when the penalty factor λ is large, minimization of the objective function mainly depends on the penalty term and the centers are more likely to move to the same location, resulting in the possible decrease of the number of clusters. By exploiting the property of the penalty term, the agglomerative k-means can obtain a serial results of different cluster numbers by increasing the penalty factor λ gradually, each of which is obtained by updating center matrix Z and membership matrix U using (3) and (4) iteratively until convergence given a fixed λ. The analysis of the agglomerative k-means algorithm can be found in [12].

One of the interesting property of the agglomerative k-means algorithm is that we can make use of λ to control the clusters to be formed at a specific density level. To show this property, we consider that each object x_i is generated by a *mixtures of Gaussian* model,

$$p(x_i) = \sum_{j=1}^{k} p(x_i|y_i = j;\ z_j,\ \Sigma_j)p(y_i = j), \quad (5)$$

where y_i represents the latent variable whose value takes from {1,2,…,k} (the values of the latent variable refer to the cluster label of the object), $p(\cdot|y_i = j; z_j, \Sigma_j)$ represents the j th Gaussian component with mean z_j and covariance matrix Σ_j, and $p(y_i = j)$ represents the prior probability that the j th Gaussian component is chosen.

In the agglomerative k-means algorithm, the penalty factor λ is used. Here we would like to link this penalty factor to the mixtures of Gaussian model. Let us further assume that the k Gaussian components have the same covariance matrix $\Sigma_1 = \Sigma_2 = \ldots = \Sigma_k = \lambda/2I_m$, where I_m is the $m \times m$ identity matrix. Then we have

$$p(x_i|y_i = j; z_j,\ \Sigma_j) = \frac{1}{(2\pi)^{m/2}\left(\frac{\lambda}{2}\right)^{m/2}} \exp\left(-\frac{1}{2}\frac{(x_i - z_j)^t(x_i - z_j)}{\left(\frac{\lambda}{2}\right)}\right). \quad (6)$$

If these k components have the same prior probability to be chosen, that is $p(y_i = 1) = p(y_i = 2) = \cdots = p(y_i = k) = 1/k$, by using Bayes rule and Eqs. (4) and (6), we can compute the posterior probability that x_i is generated by the j th Gaussian component as follows:

$$p(y_i = j|x_i) = \frac{p(x_i|y_i = j; z_j, \Sigma_j)p(y_i = j)}{\sum_{l=1}^{k} p(x_i|y_i = l; z_l, \Sigma_l)p(y_i = l)} = \frac{\exp\left(\frac{-D_{i,j}}{\lambda}\right)}{\sum_{l=1}^{k}\exp\left(\frac{-D_{i,l}}{\lambda}\right)} = u_{i,j}. \quad (7)$$

Eq. (7) builds a connection between the mixtures of Gaussian model and the agglomerative k-means algorithm. The connection means updating the objects-to-clusters membership $u_{i,j}$ in the agglomerative k-means algorithm is equivalent to updating the posterior probability that x_i is generated by the j th Gaussian component in the mixtures of Gaussian model. Also, it means the penalty factor λ in the agglomerative k-means controls the density level of clusters to be formed because their covariance matrices are all equal to $\lambda/2I_m$ according to the assumption in the mixtures of Gaussian model.

For instance, we show in Fig. 2 the relationship between λ and the density of clusters to be formed by the agglomerative k-means algorithm. We can see that the covariance matrices that generate the data satisfy $\Sigma_1 = \Sigma_2 = \Sigma_3 = \lambda/2I_2$. Meanwhile, we can see that λ needs to be different for identifying clusters with different densities. This motivates us to embed the agglomerative k-means in our cluster tree to identify multi-density clusters. For instance, for the data set in Fig. 1, we can use different values of λ to partition the objects in different nodes, and obtain the results presented.

Moreover, the agglomerative fuzzy k-means algorithm is not sensitive to the initial cluster centers [12]. This algorithm can improve the traditional k-means algorithm by generating more consistent clustering results in different clustering runs. In the algorithm, we employ a validation index proposed by Sun et al. [13]. This validation index is constructed based on average scattering (compactness) of clusters and distances (separations) between clusters. The validation index [13] is given as

$$\mathcal{V}(U,Z,k) = \mathcal{SCATTER}(k) + \frac{\mathcal{DISTANCE}(k)}{\mathcal{DISTANCE}(k_{\max})},$$

where

$$\mathcal{SCATTER}(k) = \frac{\frac{1}{k}\sum_{i=1}^{k}\|\sigma(z_i)\|}{\|\sigma(\boldsymbol{S})\|}$$

shows the average scattering of the clusters. Here

$$\sigma(\boldsymbol{S}) = [\sigma_1(\boldsymbol{S}),\sigma_2(\boldsymbol{S}),\ldots,\sigma_m(\boldsymbol{S})]^T, \sigma_j(\boldsymbol{S}) = \frac{1}{n}\sum_{i=1}^{n}(x_{i,j} - \overline{x}_j)^2,$$

$$\overline{x}_j = \frac{1}{n}\sum_{i=1}^{n} x_{i,j}, \sigma(z_l) = [\sigma_1(z_l),\sigma_2(z_l),\ldots,\sigma_m(z_l)]^T$$

and

$$\sigma_m(z_l) = \frac{1}{n}\sum_{i=1}^{n} u_{i,l}(x_{i,j} - z_{l,j})^2,$$

when the number of clusters k is large, the value of $\mathcal{SCATTER}(k)$ is small. The second term $\mathcal{DISTANCE}(k)$ measures the separation between clusters:

$$\mathcal{DISTANCE}(k) = \frac{d^2_{\max}}{d^2_{\min}}\sum_{i=1}^{k}\sum_{j=1}^{k}\frac{1}{\|z_i - z_j\|^2}, d_{\min} = \min_{i \ne j}\|z_i - z_j\|$$

and $\quad d_{\max} = \max_{i \ne j}\|z_i - z_j\|.$

We note that the smaller the $\mathcal{V}$, the better the partition is.

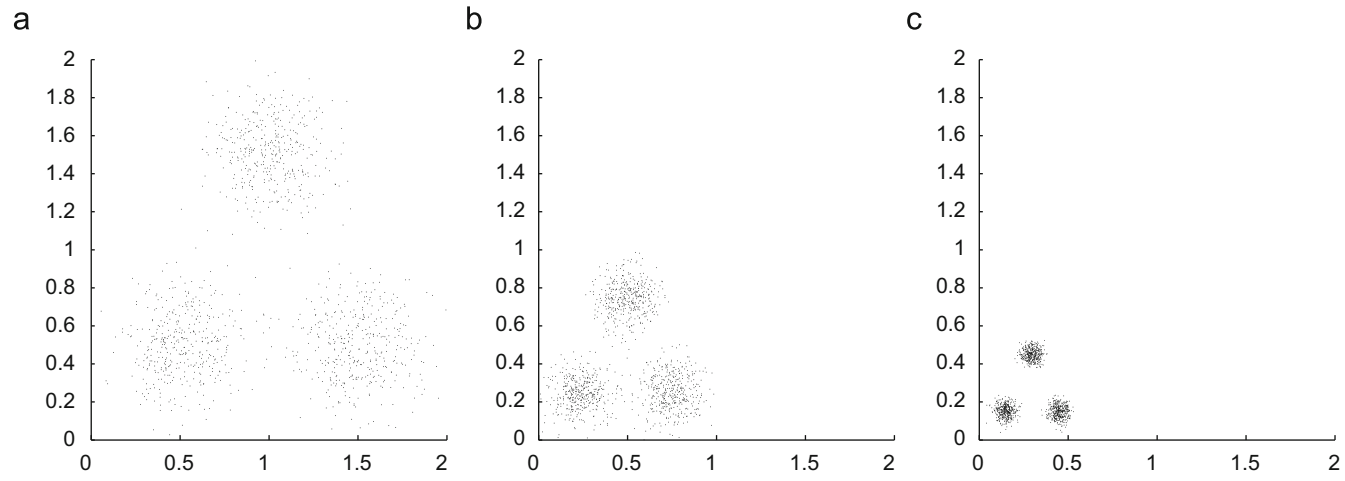

Fig. 2. The relationship between λ and the density of clusters to be formed by the agglomerative k-means algorithm. (a) $\lambda=0.06$, $\Sigma_1=\Sigma_2=\Sigma_3=0.03\times I_2$; (b) $\lambda=0.016,\Sigma_1=\Sigma_2=\Sigma_3=0.008\times I_2$ and (c) $\lambda=0.002,\Sigma_1=\Sigma_2=\Sigma_3=0.001\times I_2$.

2.3. Composite or atomic clusters

In a cluster tree, we need to decide whether a node being considered is a composite or an atomic cluster. Assessment of node validity is based on compactness criteria. The scattering of a cluster defined in the previous subsection is a good indicator for the compactness of clusters. If the scattering of a cluster is large (small), we can label the node as a composite (an atomic) cluster.

One simple way for testing the scattering value of a cluster is based on the goodness of fit for Gaussian multi-modal clusters. Let $\boldsymbol{S}^{(l)}=\{x_1^{(l)},x_2^{(l)},\ldots,x_q^{(l)}\}$ denote q objects in a node l. To check the goodness of these objects fitting a Gaussian, the sufficient statistics is based the mean and the covariance matrix. Both can be efficiently estimated by using these objects as follows:

$$\mathbf{mean}^{(l)}=\frac{1}{q}\sum_{j=1}^{q}x_j^{(l)}, \tag{8}$$

$$\mathbf{cov}^{(l)}=\frac{1}{q-1}\sum_{j=1}^{q}(x_j^{(l)}-\mathbf{mean}^{(l)})(x_j^{(l)}-\mathbf{mean}^{(l)})^t, \tag{9}$$

where y^t denotes the transpose of y. Given the mean and the covariance matrix, we can synthetically generate a Gaussian distribution with q points. Then we sort these q objects and these q generated points in terms of their distances to the mean, respectively. After dividing them into g equal length intervals, respectively, we get g observed events $\{O_1, O_2, \ldots, O_g\}$ and g expected events $\{E_1, E_2, \ldots,E_g\}$, where O_j is the number of objects falling into j th interval and E_j is the number of generated points falling into j th interval. Then we can employ the Pearson's chi-square statistics to measure the goodness of fit:

$$\chi^2=\sum_{j=1}^{g}\frac{(O_j-E_j)^2}{E_j}. \tag{10}$$

A statistical significance α can be specified to test the goodness of fit hypothesis.

2.4. The overall algorithm

By using the agglomerative k-means algorithm, the goodness of fit test and cluster validation method, we can decide a node refers to a composite or an atomic cluster, and there are how many clusters in a composite cluster. There are three parameters in the proposed cluster tree algorithm: the maximum number $k_{\max}$ of possible clusters set in the agglomerative k-means algorithm, the statistical significance α for the goodness of fit test, and the maximum number g of intervals in the goodness of fit test. The overall algorithm is implemented in the framework as follows:

Step 1: Set the data set $\boldsymbol{X}$ to the root of cluster tree, put this node in a queue **Q**, and maintain a list **L** for storing possible outliers.
Step 2: If $\mathbf{Q}.size\neq 0$, goto step 3, otherwise goto step 9.
Step 3: Get the first node in the queue **Q** and select it as current node N_c. Delete it from the queue.
Step 4: Perform the agglomerative k-means with $k_{\max}$ randomly selected centers on the data set in N_c and record the serial of clustering results with different cluster numbers.
Step 5: Select the "optimal" clustering results by using the cluster validation index $\mathcal{V}$, and generate k_{best} "candidate" children nodes $child_1,child_2,\ldots,child_{k_{best}}$ using each cluster, where k_{best} denotes the number of clusters in the "optimal" clustering result.
Step 6: Compare the size of each "candidate" child node with other $k_{best}-1$ "candidate" children nodes. If its size is sufficient smaller than another (say 10 times smaller), put its data into the list **L**; otherwise set it as a child node of current node N_c.
Step 7: For each child node of current node N_c, test whether it should be atomic cluster or not by the goodness of fit test with the parameters α and g. If it is an atomic cluster, mark it as a leaf; otherwise mark it as a non-leaf.
Step 8: Put all children nodes of current node N_c marked as non-leaf in the queue **Q**. Goto step 2.
Step 9: For each data in the list **L**, compute the distances between it and each leaf cluster center. Try to assign it to the leaf cluster with the minimum distance. If the minimum distance is smaller than 2 times of the average distance between the corresponding leaf cluster center and their own belonged objects, assign it to the leaf cluster; otherwise, identify it a as true outlier.
Step 10: End

The recursive procedure of cluster tree is implemented using a queue. For each node in need of further splitting, the algorithm puts them into the queue, denoted by step 1 and step 8. Then it

fetches the first node from the queue, performs the agglomerative k-means algorithm on it and selects the best clustering result as its "candidate" children, described from step 3 to step 5. Since there may be spurious child formed by "possible outliers" among the "candidate" children, step 6 discriminate it from others and put its data into a list **L**. After the tree is built, the algorithm handles the set of "possible outliers" in list **L**, see step 9. Step 7 is the stopping criterion (goodness of fit test) for newly generated children. Sometimes the data set may be too complicated to be decomposed into Gaussian distributions. We can use some heuristic methods as stopping criteria, such as setting the maximum depth of the cluster tree or the minimum number of objects in a node. When the proposed cluster tree algorithm finishes, it outputs a tree structure and all leaf nodes represent a partition or a clustering result of data set $\boldsymbol{X}$.

3. Experimental results

In this section, we present five experiments and show the effectiveness of the proposed cluster tree approach for handling nested clusters and clusters of multi-density.

The first four experiments are performed on synthetic data sets. Here we construct data sets with multi-density clusters by recursively griding the space and generating normal distributions within some randomly chosen grids at the same time. The clusters generated in the same grid at the same time have the same variance. Since data sets are generated by a divisive recursion, we can define the depth of the recursion as levels. We also define two cases of overlapping for the generated data sets. We define the minimum distance between the means of two distributions at the same level l as $d_{\min}^{(l)}$, and the variance as $\delta^{(l)}$. We call a data set is well-separated if $d_{\min}^{(l)} > 5\delta^{(l)}$ holds true for each level l, while a data set is overlapped if $d_{\min}^{(l)} < 3\delta^{(l)}$ holds true for some levels l. Among the first three experiments, the first experiment is for 2-dimensional well-separated data set; the second experiment is for 2-dimensional overlapped data set; the third experiment is for 2-dimensional overlapped data set with outliers; the fourth experiment is high-dimensional data sets. For these four experiments, we set the maximum number of children for each node $k_{\max}$ to be 10, the statistical significance α to be 0.001 and the maximum number of intervals g to be 6. In the last experiment, we conduct the proposed algorithm on a real data set from UCI to identify some interesting clusters.

To evaluate the clustering results, we use the adjusted rand index [14]. It measures the number of pairwise agreements between a clustering result (K) and a set of cluster labels (C), which is conceptually formulated as follows:

$$J(C,K) = \frac{\text{observed index} - \text{expected index}}{\text{maximum index} - \text{expected index}}. \tag{11}$$

More precisely, it can be computed as the following formula [15]:

$$J(C,K) = \frac{N(a+d) - [(a+b)(a+c)+(c+d)(b+d)]}{N^2 - [(a+b)(a+c)+(c+d)(b+d)]}, \tag{12}$$

where a denotes the number of pairs with the same label in C and assigned to the same cluster in K, b denotes the number of pairs with the same label in C but in different clusters in K, c denotes the number of pairs with different labels in C but in the same cluster in K, and d denotes the number of pairs with different labels in C and assigned to different clusters in K as well. Here the quantity $N=(a+b+c+d)$. The value of the adjusted rand index is in the range of [0,1] and it is 1.0 when C and K are identical. The higher the adjusted rand index value is, the better the agreement between C and K is.

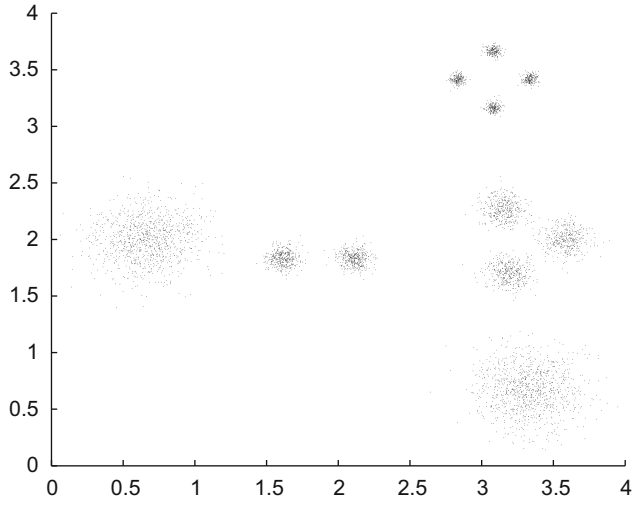

Fig. 3. A 2-dimensional well-separated data set.

3.1. Experiment 1

In the first experiment, our aim is to demonstrate how the proposed cluster tree can handle nested and multi-density clusters. A 2-dimensional well-separated data set with 4800 objects is generated, and is also composed of 11 clusters of different densities as shown in Fig. 3. There are three multi-modal clusters composed of different numbers (2,3,4) of sub-clusters in the data set.

The algorithm begins with all objects in the root node and the agglomerative k-means algorithm is applied to the data set with ten initial cluster centers as shown in Fig. 4(a). Even through the initial cluster centers are not separated enough, i.e., they are not good initial cluster centers, the agglomerative k-means algorithm still works quite well. In Figs. 4(a)–(g), we show the different clusters formed by using different values of the penalty factor in the agglomerative k-means algorithm. According to the validation indices, we find in Fig. 4(h) that the validation index value of the five clusters formed shown in Fig. 4(c) is smaller than those by the other clusters formed shown in the other figures. These results suggest that the root has five children nodes at the first level, see Fig. 5.

The stopping criterion (goodness fit of test) is tested for each child node. If the child node satisfies the stopping criterion, it is marked as a leaf node and stop for further evaluation; otherwise, it continues a further cluster partition using the agglomerative k-means algorithm. After evaluating all non-leaf nodes, the proposed method can find all eleven generated "true" clusters, and returns us a tree structure with each node representing a "cluster" of different densities. For the clustering accuracy, the adjusted rank index of the resulting clusters is 1.0000.

As we can see from our procedure, the resulting cluster tree not only finds all the interesting clusters but also exploits the structural information among the clusters. Some clusters are similar to each other since they share a node in a large density level, while they are different from each other in a small density level. The cluster tree capturing both structural information and clusters can be more informative for data exploration, especially for some multi-modal data sets with different densities.

3.2. Experiment 2

In this experiment, we generate a more complicated and 2-dimensional overlapped data set to show the robustness and

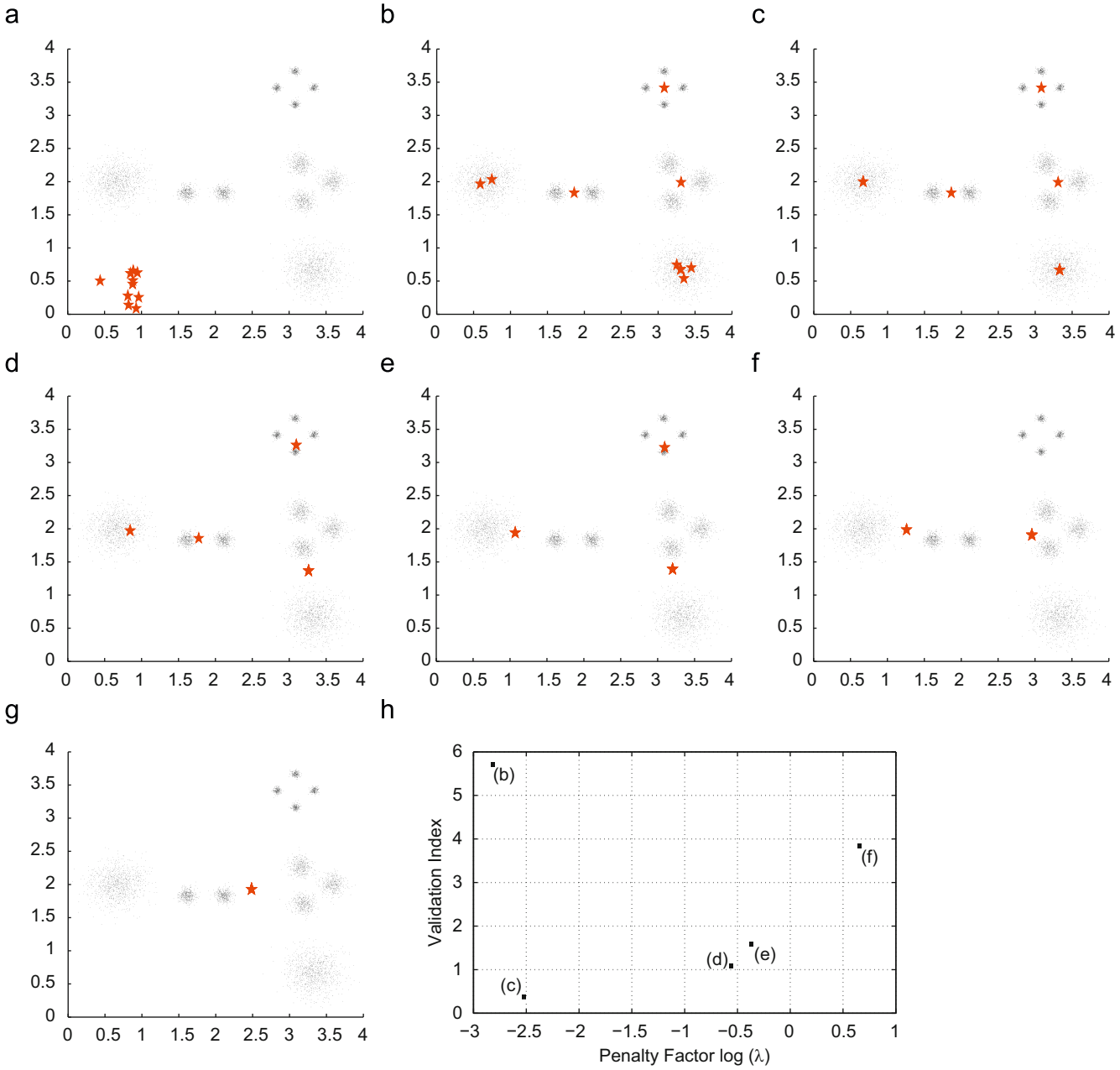

Fig. 4. (a) Ten initial centers; (b) nine non-overlapping centers; (c) five non-overlapping centers; (d) four non-overlapping centers; (e) three non-overlapping centers; (f) two non-overlapping centers; (g) one center and (h) the validation index values from (b) to (f).

effectiveness of our algorithm. It has 5600 objects and 19 clusters with different densities, as is shown in Fig. 6. This data set has more overlapped clusters than that in Experiment 1, and has a more complicated structure.

We apply the agglomerative k-means algorithm on the whole data set, the clustering results for different values of λ are shown in Figs. 7(a)–(i). According to the validation index values computed in Fig. 8, the best clustering result is given in Fig. 7(c), and therefore we have seven children nodes. In Fig. 9, we show all these seven clusters. For simplicity, we mention these seven nodes as 1,2,…,7, respectively. Since the data are seriously overlapped, we can see that there are some outliers in each node. We employ the goodness of fit test on these seven nodes, and find that the nodes 1 and 6 can be regarded as atomic clusters. However, for the nodes 2, 3, 4, 5, 7, they are classified as composite clusters.

Visually, we see from Fig. 9 that the nodes 2 and 3 have four and two genuine clusters, respectively. Indeed, when we apply the agglomerative k-means algorithm on these two nodes, we successfully partition the objects in these two nodes, and find correctly four and two genuine clusters, respectively, for the nodes 2 and 3. For the node 4, there are five sub-clusters and some outliers. The clustering results for the node 4 are shown in Fig. 10. Their corresponding validation index values are given in Fig. 11. In this case, we decide the node 4 contains seven sub-clusters. When we apply the goodness of fit test to these seven sub-clusters at the node 4, we check that five sub-clusters are good enough, but the other two sub-clusters containing the outliers fail. The five sub-clusters are atomic clusters and the two sub-clusters are not atomic clusters. Since the number of objects in the two sub-clusters are not sufficient large compared with the other sub-clusters at this level, we classify them some possibly outliers instead of forming composite clusters. Similarly, the number of sub-clusters are five and one for the nodes 5 and 7, respectively. Each node has two sub-clusters containing possibly outliers.

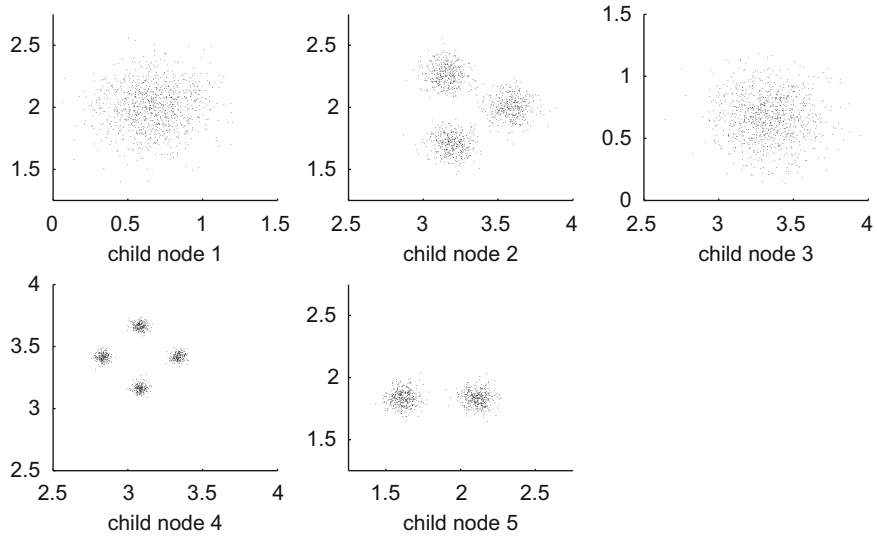

Fig. 5. Five children nodes are generated at the first level.

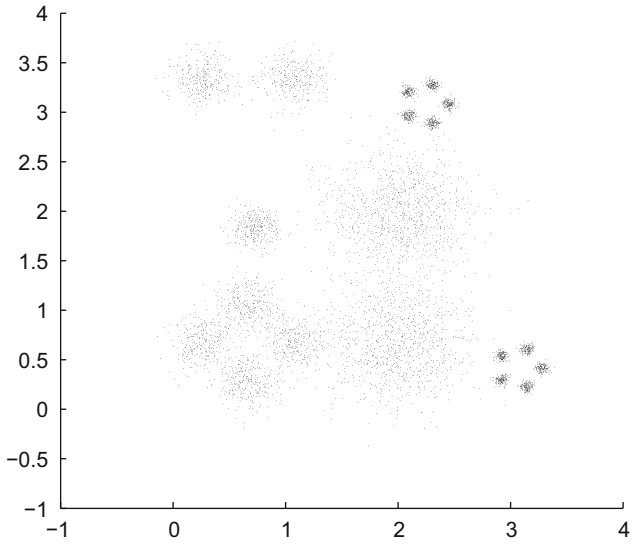

Fig. 6. A 2-dimensional overlapped data set with 19 clusters.

Finally, the cluster tree is built and there are 19 genuine clusters which match with the actual number of clusters generated in the data set. For the clustering accuracy, the adjusted rank index of the resulting clusters is 0.8739.

In our cluster tree algorithm, we put all the possibly outliers in a "possibly outliers" set, and assign these possibly outliers to the nineteen genuine clusters in the cluster tree. The assignment can be based on the distance between the possibly outlier and cluster centers, and the average distance between cluster centers and their own belonged objects. In case the assignment cannot be done, i.e., we cannot assign the membership of the possibly outlier to any cluster, we classify the possibly outlier to be a real outlier for the whole data set. In next subsection, we test a data set containing outliers.

3.3. Experiment 3

In this experiment, we generate a 2D overlapped multi-density data set with six inherent clusters and some artificially added outliers, as is shown in Fig. 12(a). We would like to demonstrate that our algorithm can deal with overlapped data with outliers.

We obtain the clustering result shown as in Fig. 12(b). It is obvious that our algorithm identified all those inherent cluster structures and almost all outliers except for the ones at (1,1), $(5,-1)$ and (2,0). It is easy to understand the reason for the first two outliers is that they lie in the vicinity of inherent clusters. To understand the reason for the last outlier, we need to understand the space partition process of our approach.

Our cluster tree algorithm is actually a process of partitioning the space by Voronoi polygon recursively, see Figs. 12(c) and (d). We can see that the cluster tree partitioned the space into five regions at the first level and the outlier at (2,0) lies in the Voronoi region of the "magenta" cluster. This is the reason why it is misclassified. Subsequently, two regions among the five were partitioned into sub-regions further at the second level and other outliers were identified correctly. For the clustering accuracy, the adjusted rank index of the resulting clusters is 0.9503.

The Voronoi partition process also suggests that our algorithm cannot directly handle clusters of arbitrary shape because only convex regions can be obtained during the process. However, one can obtain the clusters of arbitrary shape by some post-handling, such as merging some overlapped clusters based on their inter-connectivity together. For instance, in Fig. 12(b), if we merge the "cyan" cluster, "blue" cluster and the "magenta" cluster together, we can obtain a cluster shaped as a tilted "T".

3.4. Experiment 4

In this experiment, we compare the proposed cluster tree algorithm with other existing algorithms DBSCAN [16], X-means [17], BIRCH [18], CURE [19], NBC [6], OPTICS [5], Neural Gas [20], Tree-SOM [21], EnDBSCAN [7] and LDBSCAN [8]. We generate data sets with multi-density, and their dimensions vary from low (3, 5, 7) to high (15, 20, 25). Their clusters can be well-separated or overlapped. The descriptions of the data sets are shown as Table 1. We test the following algorithms, and tune suitable parameters for each algorithm in the comparison.

DBSCAN: We used the DBSCAN code implemented by Weka 3.5. In order to set the radius Epsilon and the minimum number of points *Minpts* well, we searched from ranges for them. For our generated data in Table 1, we set the ε in the range of [0.01, 1] and

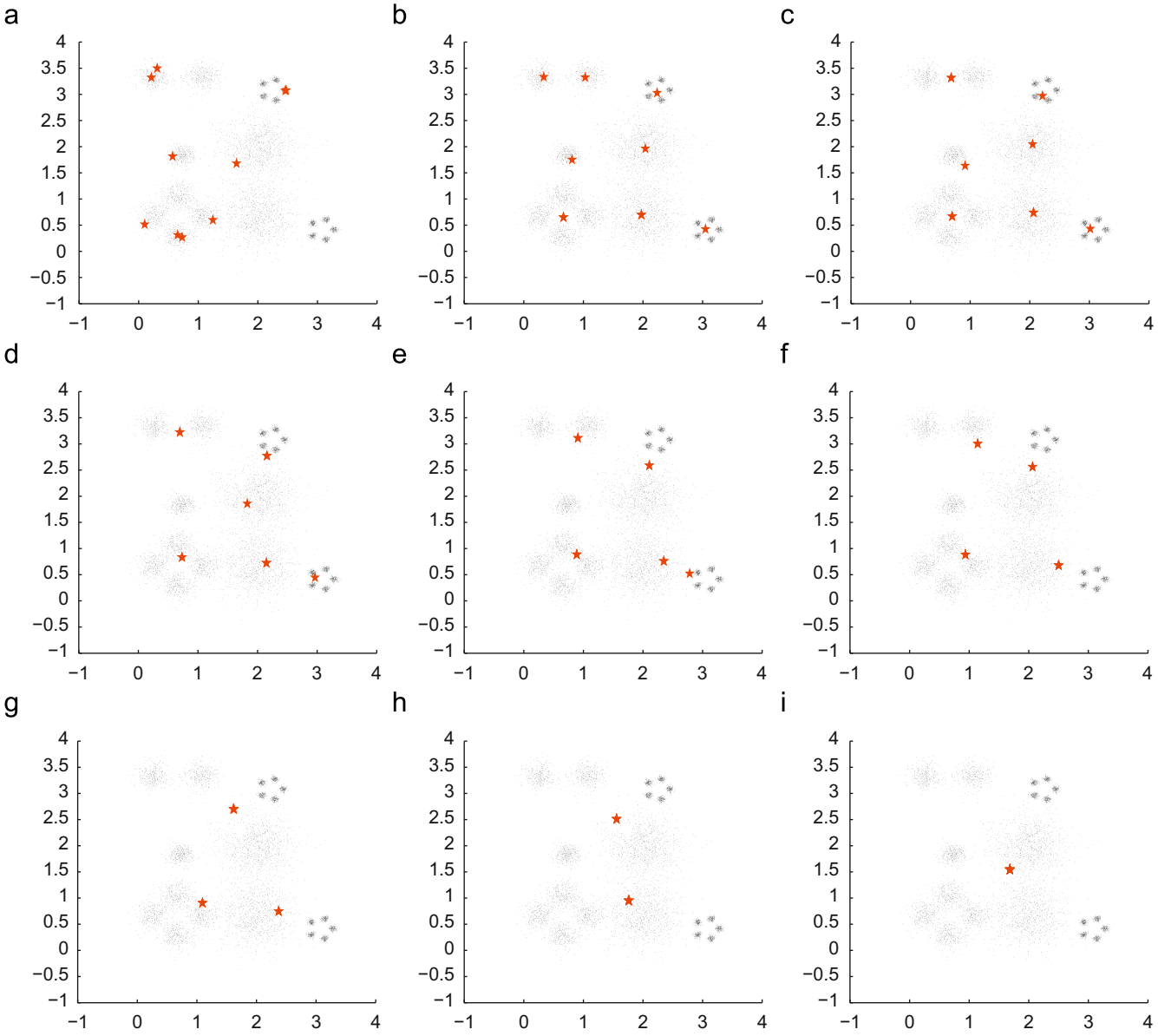

Fig. 7. (a) Ten initial centers; (b) eight non-overlapping centers; (c) seven non-overlapping centers; (d) six non-overlapping centers; (e) five non-overlapping centers; (f) four non-overlapping centers; (g) three non-overlapping centers; (h) two non-overlapping centers and (i) one center.

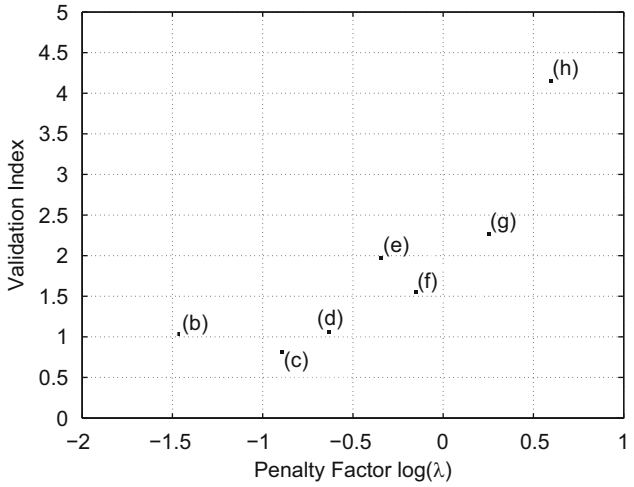

Fig. 8. The validation index values for Figs. 7(b)–(h).

increasing step as 0.01. We set the *MinPts* in the range of [1, 30] and the increasing step as 1. Then we tried different settings for both values in the intervals and found the one with the highest adjusted rand index among them.

X-means: We used the X-means code implemented by Weka 3.5. As we can see from the Table 1, the number of clusters in data set is between 14 and 29. Therefore, we set the miniNumCluster as 14 and maxNumCluster as 29. To make the results less sensitive to the initial centers, we set the maxIterations to 50. All the other values were set as default values by Weka 3.5.

BIRCH: The BIRCH code was downloaded from the internet which was implemented by its authors. For BIRCH, the most important parameter is the number of clusters. We set it as the genuine number of clusters for each data set. All the other parameters were set as default values given by authors.

CURE: The CURE code was downloaded from the internet. We set the number of clusters as the genuine number of it. We set the shrink factor α to be 0.5 which was in the range of 0.2–0.7 suggested by the authors [19]. And for each cluster, we set the number of representative to be 10.

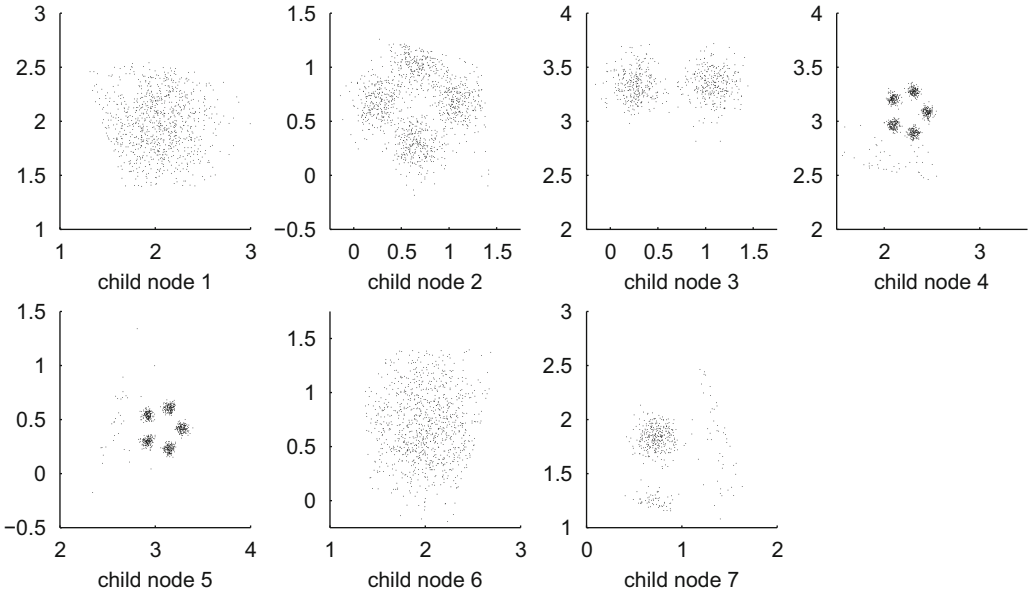

Fig. 9. Seven children nodes are generated at the first level.

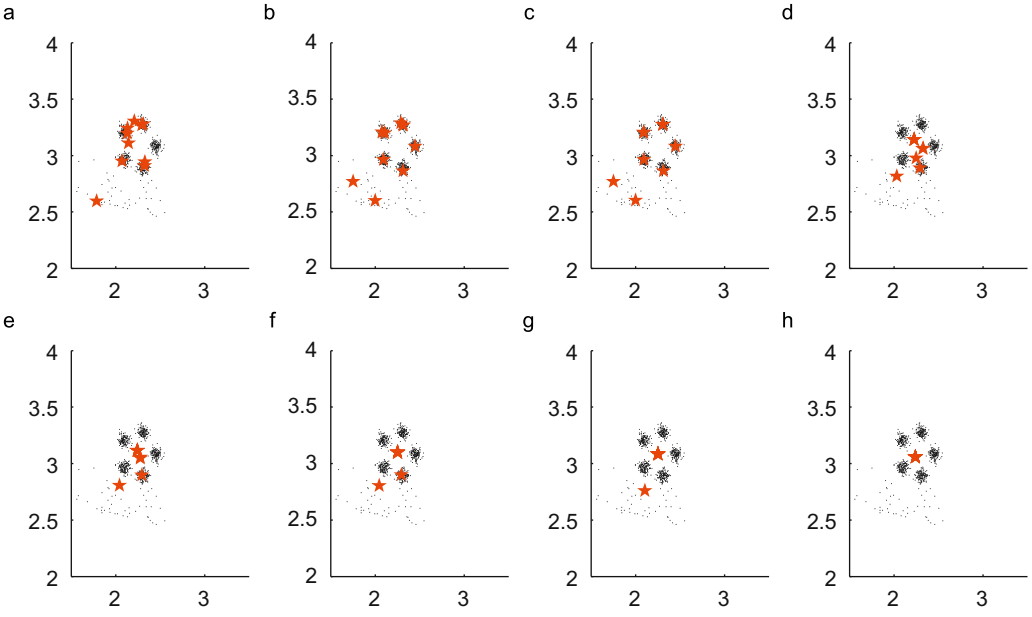

Fig. 10. (a) Ten initial centers; (b) nine non-overlapping centers; (c) seven non-overlapping centers; (d) five non-overlapping centers; (e) four non-overlapping centers; (f) three non-overlapping centers; (g) two non-overlapping centers and (h) one center.

NBC: The code of NBC was implemented as the descriptions in [6]. NBC only needs one parameter d to define the neighborhood-based density factor. We increased d from 1 to 100 with step being 1, and obtained 100 different the clustering results. We chose the one with highest adjusted rand index among them as the final result.

OPTICS: We used the OPTICS code implemented by Weka 3.5. For all the tested data sets, we set ε to be 1 and *Minpts* to be 8. It is reasonable because only one cluster contains almost all points in each tested data set with respected to these parameters, which is suggested in [5]. Then OPTICS obtained a order list of the data to identify clusters structures. To obtain the clustering result from this order list, we increased ε' from 0.01 to 1 with step being 0.01, and generated different clustering results. Finally, we chose the one with highest adjusted rand index as the final result.

Neural Gas: We set the number of neurons to be the number of genuine clusters and the number of epoch to be 100 multiplied by the number of objects. For both initial step size and initial decay constant, we used the default the values as 0.5 and half of the number of neurons.

Tree-SOM: We tried the division threshold as 200, 300, 400, 500 and the division factor as 2, 3, 4, 5, respectively. We set the other parameters eta0, sigma0, tau1, tau2, decay constant and k-means round as 0.3, 0.8, 4, 2, 0.8 and 2, respectively. Due to the clustering results are sensitive to the order of data, we tried 10 times for each parameter setting and considered the averaged accuracy. Then we chose the one with highest averaged accuracy as the final result from the results of different parameters settings. The codes of both Neural Gas and tree-SOM we used were downloaded from http://www.cis.hut.fi/research/software.

EnDBSCAN: We implemented the code of EnDBSCAN as its description in [7]. We set the tolerance factor to be 1. And we searched in the intervals of [1, 30] and [0.5, 3] for *Minpts* and ε, with increasing steps 1 and 0.1, respectively. Among the clustering results with different parameter settings, we chose the best one as final result.

LDBSCAN: The code was obtained from the author of LDBSCAN. We set the parameter *Minpts* and number of neighbors to be 15 and 10, respectively. And we searched in the intervals of [1, 2] and [1, 2] for UB-lof and PCT, with increasing steps 0.1 and 0.1, respectively. Among the clustering results with different parameter settings, we chose the best one as final result.

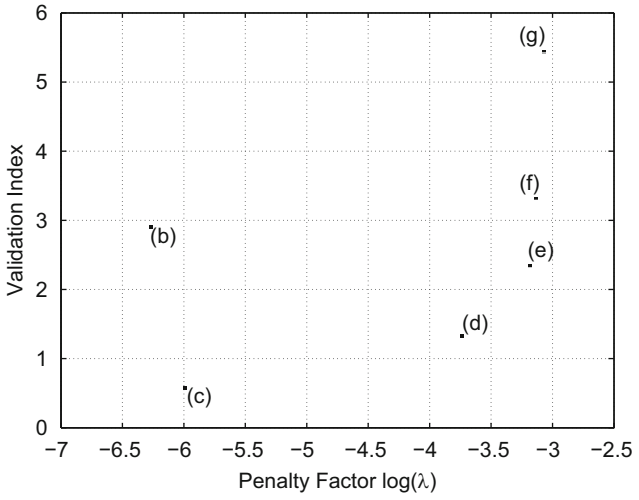

Fig. 11. The validation index values for Figs. 10 (b)–(g).

Table 1
The descriptions of generated data sets.

Data ID	Data sets	No. of objects	No. of genuine clusters
1	3D well-separated	4200	22
2	3D overlapped	3600	20
3	5D well-separated	2800	15
4	5D overlapped	3400	18
5	7D well-separated	7000	25
6	7D overlapped	7600	28
7	15D well-separated	2500	16
8	15D overlapped	2300	22
9	20D well-separated	2600	23
10	20D overlapped	2700	22
11	25D well-separated	3700	28
12	25D overlapped	3100	25

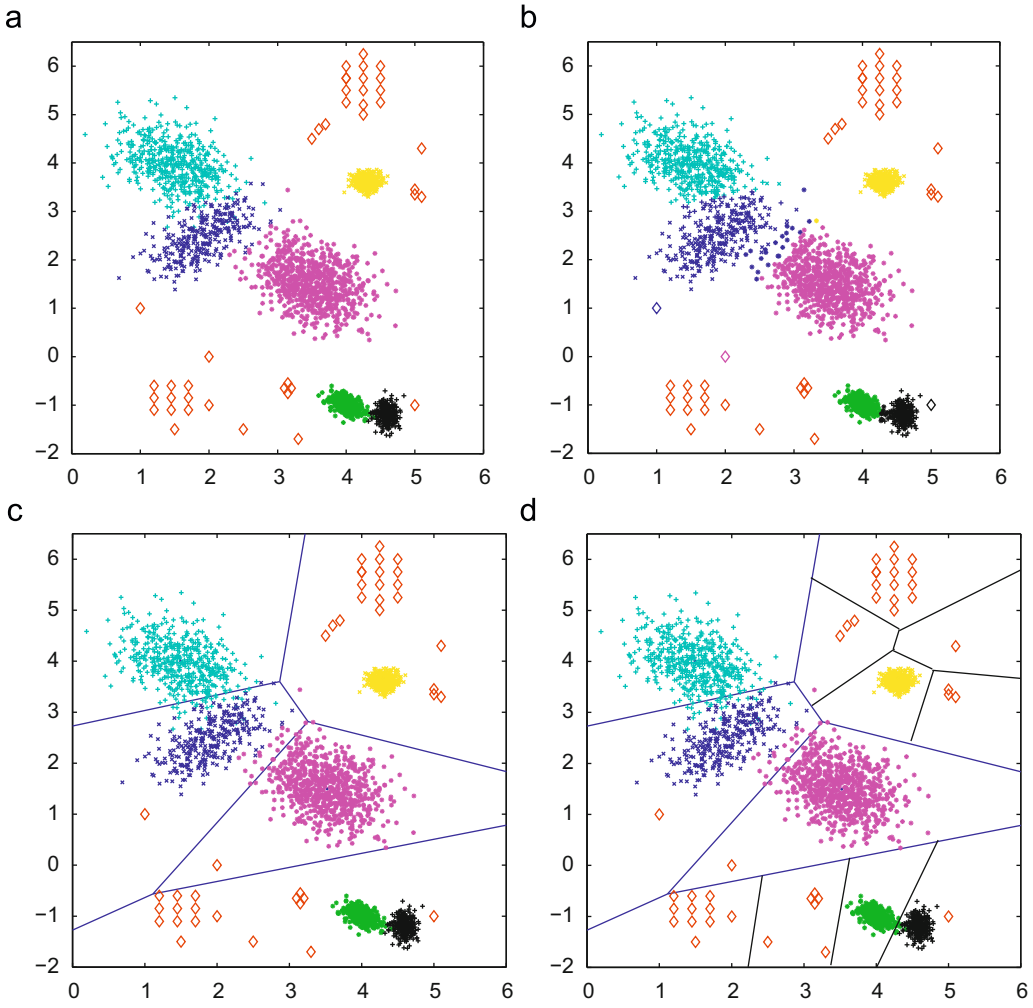

Fig. 12. (a) The overlapped data with six inherent clusters and some outliers depicted with different colors, respectively; (b) the clustering result by cluster tree depicted with different colors; (c) the first-level Voronoi partition of cluster tree and (d) the second-level Voronoi partition of cluster tree depicted with black color. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 2
The comparisons of clustering accuracy by adjusted rand index.

Data ID	DB-SCAN	X-means	BIRCH	CURE	NBC	OPTICS	Neural Gas	Tree-SOM	Cluster tree	EnDB-SCAN	LDB-SCAN
1	0.9310	0.7767	0.8578	0.3028	0.9811	0.9010	0.6134	0.8009	**0.9995**	0.8939	0.9666
2	0.7012	0.4897	0.5611	0.2865	0.7848	0.6874	0.5844	0.6710	**0.9853**	0.6833	0.7066
3	0.7481	0.7759	0.6297	0.5894	0.9875	0.7473	0.6634	0.7577	**0.9990**	0.9023	0.9751
4	0.6491	0.8028	0.3789	0.4213	0.8166	0.5968	0.7063	0.6940	**0.9858**	0.5420	0.6720
5	0.8849	0.8251	0.3318	0.4380	0.8140	0.8833	0.6452	0.7578	**0.9989**	0.5314	0.4606
6	0.5021	0.5666	0.2870	0.2619	0.4657	0.4655	0.6359	0.7185	**0.9591**	0.4644	0.6324
7	0.7326	0.6897	0.4725	0.5433	0.7854	0.6661	0.4901	0.6654	**0.9980**	0.7208	0.4544
8	0.4331	0.5800	0.4070	0.1735	0.4385	0.4266	0.7298	0.7254	**0.8931**	0.4353	0.4206
9	0.5269	0.8385	0.5173	0.3664	0.5513	0.5269	0.6620	0.8615	**0.9967**	0.5236	0.4957
10	0.5986	0.7012	0.3932	0.4358	0.6370	0.5933	0.4666	0.6307	**0.9051**	0.5911	0.5817
11	0.5854	0.6424	0.5220	0.3507	0.6612	0.5854	0.6397	0.6740	**0.9981**	0.5833	0.6031
12	0.5562	0.6662	0.5962	0.3164	0.5998	0.5411	0.5474	0.6880	**0.9443**	0.5970	0.5472

Table 3
The comparisons of identification of the correct number of clusters.

Data ID	No. of genuine clusters	DB-SCAN	X-means	NBC	OPTICS	Tree-SOM		Cluster tree	EnDB-SCAN	LDB-SCAN
						Aver. no.	Std.			
1	22	20	16	**22**	19	11.6	1.8	**22**	19	20
2	20	9	14	11	8	10.5	3.2	**20**	13	28
3	15	8	17	**15**	8	8.5	1.6	**15**	12	17
4	18	10	**17**	11	11	16.2	4.3	**19**	12	15
5	25	24	29	23	24	26.6	7.5	**25**	13	15
6	28	10	16	8	7	31.8	0.5	**29**	9	15
7	16	8	21	10	7	8.7	0.5	**16**	13	21
8	22	6	16	7	7	23.9	2.8	**22**	8	12
9	23	10	22	11	9	25.9	0.3	**23**	11	27
10	22	7	15	8	7	8	0.5	**22**	8	31
11	28	9	22	12	9	37	0.8	**28**	10	12
12	25	10	16	12	10	30.4	1.1	**25**	23	11

Aver. no. means the averaged number of clusters and std. means the standard deviation of the number of clusters.

Table 4
The comparisons of critical parameters to be set in each algorithm.

Data ID	DBSCAN		BIRCH (no. of clusters)	CURE (no. of clusters)	NBC (d)	OPTICS (ε')	Neural gas (no. of neurons)	Tree-SOM		EnDB-SCAN		LDB-SCAN	
	ε	Pts						D.T.	D.F.	Pts	ε	UB-lof	PCT
1	0.03	4	22	22	62	0.03	22	500	5	4	0.5	1	1.3
2	0.07	9	20	20	26	0.09	20	500	5	3	0.6	1.9	1.4
3	0.17	2	15	15	75	0.17	15	500	5	30	1.4	1.2	1.5
4	0.10	18	18	18	46	0.08	18	400	4	3	0.8	1.1	1.1
5	0.06	8	25	25	41	0.06	25	500	5	3	1.2	1.2	1.7
6	0.06	14	28	28	95	0.08	28	500	5	29	0.5	1.1	1.9
7	0.12	27	16	16	36	0.11	16	500	5	5	0.7	1.2	1.2
8	0.26	23	22	22	82	0.29	22	200	2	3	2.5	1.1	1.9
9	0.30	1	23	23	24	0.30	23	300	3	3	1.3	1.1	1.2
10	0.27	5	22	22	87	0.27	22	400	4	30	2.5	1.1	1.7
11	0.24	6	28	28	63	0.24	28	200	2	8	1.5	1	1.9
12	0.38	27	25	25	22	0.35	25	200	2	2	2.3	1	1.3

D.T. represents the division threshold and D.F. represents the division factor.

Table 2 gives the comparison results in clustering accuracy for all tested algorithms. We can see that the proposed algorithm achieves a very high clustering accuracy when dealing with data sets with nested structure and multi-density. NBC yields acceptable accuracies on the low dimensional well separated data sets while low accuracies on overlapped or high dimensional data sets. This is because it cannot evaluate the neighborhood reasonably to identify the correct number of clusters for overlapped data sets. LDBSCAN only achieves acceptable accuracies for 3D and 5D well-separated data sets. OPTICS, DBSCAN, X-means, Tree-SOM and EnDBSCAN cannot achieve a high accuracy because they cannot detect the correct number of clusters. The clustering accuracies of BIRCH, CURE and Neural Gas are also not high.

Table 3 gives the comparison results in identification of the correct number of clusters. We can see that the proposed algorithm can detect almost correctly the number of clusters in each generated data set. NBC can almost detect the correct number of clusters on low dimensional well-separated data sets while rarely correct on overlapped or high dimensional data sets. OPTICS, DBSCAN, Tree-SOM, EnDBSCAN and LDBSCAN cannot accurately detect the number of clusters because the parameters

of them are not easy to set for these data sets with nested structure and multi-density. Although we set the range of clusters for X-means, it is still difficult for it to identify the correct number of clusters. Table 4 gives the comparison of the settings of critical parameters for each algorithm. We remark in all these data sets that the parameters of the proposed cluster tree algorithm are the maximum number of children for each node k_{max} set to be 10, the statistical significance α set to be 0.001 and the maximum number of intervals g set to be 6. The parameters of the NBC, OPTICS, DBSCAN, Tree-SOM, EnDBSCAN and LDBSCAN must change for different data sets in order to get a reasonable high accuracies. For the BIRCH, CURE and Neural Gas, we need to know the accurate number of clusters, which is not easy. X-means only needs to know the range of clusters, however, it is not able to detect the correct number of clusters.

In summary, these results show that the performance of the proposed algorithm is better than that of the other tested algorithms.

Table 5
The description of class labels and the corresponding numbers of objects.

Class ID	Class label	No. of objects
1	Red soil	1072
2	Cotton crop	479
3	Grey soil	961
4	Damp grey soil	415
5	Soil with vegetation stubble	470
6	Very damp grey soil	1038

3.5. Experiment 5

In this experiment, we conduct the proposed cluster tree algorithm on Landsat Satellite Data sets from UCI Machine Learning Repository (see http://archive.ics.uci.edu/ml/). We would like to demonstrate how to use cluster tree to discover some interesting clusters for knowledge discovery.

The data set, composed of 4435 objects, is obtained from remote-sensing satellite images. Each object represents a 3×3 region and each sub-region is recorded by the four measurements of intensity taken at different wavelength. Therefore, each object has 36 attributes. A class label indicating the type of the central sub-region is also given for each object. The class labels and the number of associated objects are described as Table 5.

Previous studies on this data set have shown that there are some interesting hierarchical structures [22,23]. We have found something similar as shown in Fig. 13 when conducting our algorithm on it with $k_{max}=5$ and the maximum depth of the cluster tree being 2. The sub-figures, i.e., Figs. 13(a)–(c) and so on, are the histograms of class ID corresponding to its node. Table 6 shows the distribution of classes for each node in Fig. 13.

At the first level of the cluster tree, three clusters are generated. One of the clusters is "cotton crop" (the 2nd Class), which is separated clearly from other types of land as shown in Fig. 13(c); while the other two are mixed as shown in Figs. 13(b) and (d), respectively. The fact means that the "cotton crop" can be easily discriminated from other types of land. When the cluster tree grows deeper, the clusters of mixed types begin to be separated into different classes. One cluster mixed with "red soil"

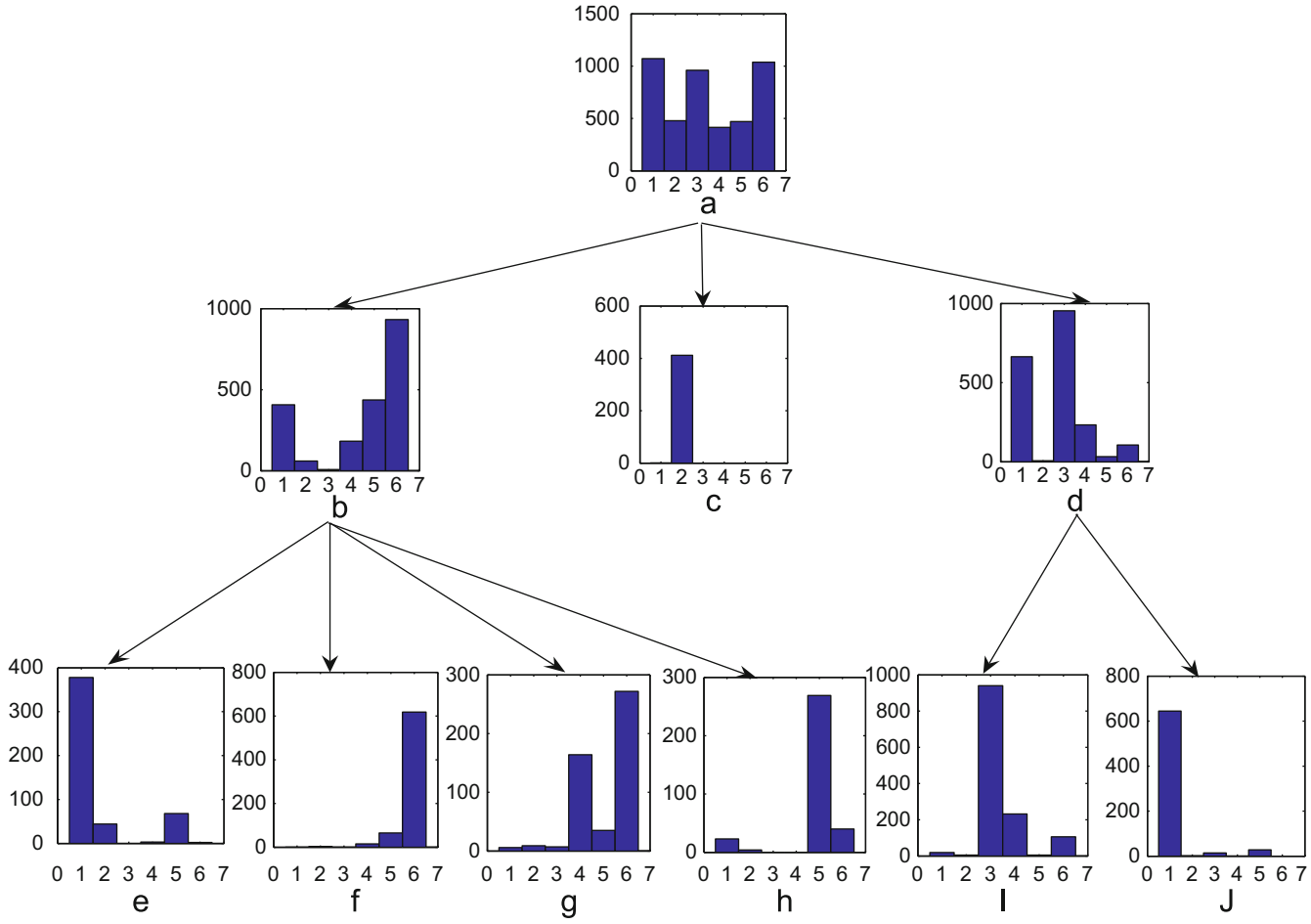

Fig. 13. The first two-level hierarchical structures of the Cluster Tree on Landsat Satellite Data (omitting the further decomposition of node (c) for space reason).

Table 6
The distribution of classes for each node in Fig. 13.

Node	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6
(a)	1072	479	961	415	470	1038
(b)	407	60	7	182	437	932
(c)	1	412	0	0	0	0
(d)	664	7	954	223	33	106
(e)	378	44	0	3	68	2
(f)	0	3	0	15	65	618
(g)	6	9	7	164	35	272
(h)	23	4	0	0	269	40
(i)	19	5	940	232	5	106
(j)	645	2	14	1	28	0

(the 1st Class), "damp grey soil" (the 4th Class), "soil with vegetation stubble" (the 5th Class) and "very damp grey soil" (the 6th Class), indicated by Fig. 13(b), is nearly clearly separated. Another cluster mixed with "red soil" (the 1st Class), "grey soil" (the 3rd Class) and "damp grey soil" (the 4th Class), denoted by Fig. 13(d), is also grouped into classes gradually. However, the lands of "grey soil" (the 3rd Class), "damp grey soil" (the 4th Class) and "very damp grey soil" (the 6th Class) cannot be well separated, especially "damp grey soil" and "very damp grey soil", as is shown in Figs. 13(i) and (g). This is because these types of land are too similar to each other.

4. Concluding remarks

In this paper, we have presented a cluster tree approach to identify nested and multi-density clusters. Our approach embeds the agglomerative *k*-means in the generation of cluster tree to identify multi-density clusters level by level. Combined with cluster validation techniques, we can determine how many composite clusters are there and whether it is a composite cluster or atomic cluster at each level. Experimental results on both synthetic data and real data have shown the effectiveness of cluster tree in identifying the nested and multi-density clusters and discovering the hidden knowledge. Compared with some existing clustering algorithms DBSCAN, X-means, BIRCH, CURE, NBC, OPTICS, Neural Gas, Tree-SOM, EnDBSCAN and LDBSCAN, our approach performs better than these methods for nested and multi-density data sets.

In this paper, we do not study very high-dimensional data sets like text data and gene expression data. In the future work, we would like to extend the proposed cluster tree algorithm to handle very high-dimensional data sets. A hyperbolic space can be used to display hierarchically structured data such that the size of a neighborhood around any point can be increased exponentially [24]. This exponentially scaling approach has been successfully used for high-dimensional text data [25] and genome data [26].

Acknowledgments

This research is supported in part by NSFC under Grant no. 60603066, China National High-tech Program under Grant no. 2007AA01Z436, Shenzhen Science and Technology Program under Grant nos. NSKJ-200707, 08CXY-44, PCT200805190162A, and RGC 201508 and HKBU FRGs.

References

[1] R. Xu, D. Wunsch, Survey of clustering algorithms, IEEE Transaction on Neural Networks 6 (3) (2005) 645–672.
[2] L. Ertoz, M. Steinbach, V. Kumar, Finding clusters of different sizes, shapes and densities in noisy, high dimensional data, in: SIAM International Conference on Data Mining, 2003.
[3] G. Karypis, J. Han, V. Kumar, CHAMELEON: a hierarchical clustering algorithm using dynamic modeling, IEEE Computer 32 (8) (1999) 68–75.
[4] Y. Zhao, S. Mei, X. Fan, S. Junde, Clustering datasets containing clusters of various densities, Journal of Beijing University of Posts and Telecommunications 26 (2) (2003) 42–47.
[5] M. Ankerst, M.M. Breuing, H.P. Kriegel, J. Sander, OPTICS: ordering points to identify the clustering structure, in: ACMSIGMOD, 1999, pp. 49–60.
[6] S. Zhou, Y. Zhao, J. Guan, Z. Huang, A neighborhood-based clustering algorithm, in: Ninth Pacific-Asia Conference on Knowledge Discovery and Data Mining, Springer Press, Hanoi, 2005, pp. 361–371.
[7] S. Roy, D.K. Bhattacharyya, An approach to find embedded clusters using density based techniques, in: Proceedings of the ICDCIT, Lecture Notes in Computer Science, vol. 3816, 2005, pp. 523–535.
[8] L. Duan, L. Xu, F. Guo, J. Lee, B. Yan, A local-density based spatial clustering algorithm with noise, Information Systems 32 (2007) 978–986.
[9] S. Gao, Y. Xia, A grid-based density-confidence-interval clustering algorithm for multi-density dataset in large spatial database, in: International Conference on Intelligent Systems Design and Applications, vol. 1, 2006, pp. 713–717.
[10] X. Chen, Y. Min, Y. Zhao, P. Wang, GMDBSCAN: multi-density DBSCAN cluster based on grid, in: IEEE International Conference on E-Business Engineering, 2008, pp. 780–783.
[11] M.M. Breunig, H.-P. Kriegel, R.T. Ng, J. Sander, LOF: identifying density-based local outliers, in: Proceedings of the ACM SIGMOD 2000 International Conference on Management of Data, 2000, pp. 93–104.
[12] M. Li, M.K. Ng, Y.M. Cheung, Z. Huang, Agglomerative fuzzy K-means clustering algorithm with selection of number of clusters, IEEE Transaction on Knowledge and Engineering 20 (11) (2008) 1519–1534.
[13] H. Sun, S. Wang, Q. Jiang, FCM-based model selection algorithms for determining the number of clusters, Pattern Recognition 37 (2004) 2027–2037.
[14] L. Hubert, P. Arabie, Comparing partitions, Journal of Classification 2 (1985) 193–218.
[15] L.M. Collins, C.W. Dent, Omega: a general formulation of the rand index of cluster recovery suitable for non-disjoint solutions, Multivariate Behavioral Research 23 (1988) 231–242.
[16] E. Martin, K. Hans-Peter, S. Jorg, X. Xu, A density-based algorithm for discovering clusters in large spatial database with noise, in: International Conference on Knowledge Discovery in Databases and Data Mining, Montreal, Canada, August 1995.
[17] D. Pelleg, A. Moore, X-means: extending *K*-means with efficient estimation of the number of clusters, in: Proceedings of the 17th International Conference on Machine Learning, July 2000.
[18] T. Zhang, R. Ramakrishnan, M. Livny, BIRCH: an efficient data clustering method for very large database, in: Proceedings of the ACM SIGMOD Conference on Management of Data, 1996, pp. 103–114.
[19] S. Guha, R. Rastogi, K. Shim, CURE: an efficient clustering algorithm for large database, in: Proceedings of the ACM SIGMOD International Conference on Management of Data, 1998, pp. 73–84.
[20] T. Martinetz, K. Schulten, A neural-gas network learns topologies, Artificial Neural Network, vol. 1, 1991, pp. 397–402.
[21] J. Pakkanen, J. Iivarinen, E. Oja, The evolving tree—a novel self-organizing network for data analysis, Neural Processing Letters 20 (2004) 199–211.
[22] C. Bishop, M. Tipping, A hierarchical latent variable model for data visualization, IEEE Transactions on Pattern Analysis and Machine Intelligence 20 (1998) 281–293.
[23] T. Su, J. Dy, Automated hierarchical mixtures of probabilistic principal component analyzers, in: Proceedings of the 21st International Conference on Machine Learning, no. 98 Banff, Canada, July 2004.
[24] J. Ontrup, H. Ritter, Large scale data exploration with the hierarchical growing hyperbolic SOM, Neural Networks 19 (2006) 751–761.
[25] J. Ontrup, H. Ritter, Hyperbolic self-organizing maps for semantic navigation, Advances in Neural Information Processing Systems 14 (2002) 1417–1424.
[26] C. Martin, N. Diaz, J. Ontrup, T. Nattkemper, Hyperbolic SOM-based clustering of DNA fragment features for taxonomic visualization and classification, Bioinformatics 24 (2008) 1568–1574.

About the Author—XUTAO LI received the BS and MS degrees in Computer Science from Lanzhou University and Harbin Institute of Technology in China in 2007 and 2009, respectively. He is currently working towards the PhD degree in the Department of Computer Science, Harbin Institute of Technology. His research interests include data mining, clustering algorithms, and machine learning.

About the Author—YUNMING YE received the PhD degree in Computer Science from Shanghai Jiao Tong University. He is now an Associate Professor in the Department of Computer Science, Harbin Institute of Technology. His research interests include data mining, text mining, and clustering algorithms.

About the Author—MARK LI received the BE degree in Electronics Engineering from JINAN University, Guangzhou, China, in 2002, the MSc degree in Electronic Commerce and Internet Computing at the University of Hong Kong in 2005, and the PhD degree in the Department of Mathematics, Hong Kong Baptist University. His research interests include data mining, subspace clustering algorithm, bioinformatics, and business intelligence.

About the Author—MICHAEL K. NG is a Professor in the Department of Mathematics, Hong Kong Baptist University. As an Applied Mathematician, his main research interests include bioinformatics, data mining, operations research, and scientific computing. He has published and edited five books and published more 140 journals. He is the Principle Editor of the Journal of Computational and Applied Mathematics and the Associate Editor of the SIAM Journal on Scientific Computing, Numerical Linear Algebra with Applications, the International Journal of Data Mining and Bioinformatics, and Multidimensional Systems and Signal Processing.