Contents lists available at ScienceDirect

Applied Soft Computing Journal

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

A hierarchical Gamma Mixture Model-based method for estimating the number of clusters in complex data

Muhammad Azhar [a], Joshua Zhexue Huang [a,b], Md Abdul Masud [a], Mark Junjie Li [a,b,*], Laizhong Cui [a,b]

[a] College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China
[b] National Engineering Laboratory for Big Data System Computing Technology, Shenzhen University, Shenzhen, China

ARTICLE INFO

Article history:
Received 9 November 2018
Received in revised form 27 September 2019
Accepted 19 October 2019
Available online 20 November 2019

Keywords:
Number of clusters
Initial cluster centers
Gamma Mixture Model (GMM)
EM algorithm
Clustering algorithms

ABSTRACT

This paper proposes a new method for estimating the true number of clusters and initial cluster centers in a dataset with many clusters. The observation points are assigned to the data space to observe the clusters through the distributions of the distances between the observation points and the objects in the dataset. A Gamma Mixture Model (GMM) is built from a distance distribution to partition the dataset into subsets, and a GMM tree is obtained by recursively partitioning the dataset. From the leaves of the GMM tree, a set of initial cluster centers are identified and the true number of clusters is estimated. This method is implemented in the new GMM-Tree algorithm. Two GMM forest algorithms are further proposed to ensemble multiple GMM trees to handle high dimensional data with many clusters. The GMM-P-Forest algorithm builds GMM trees in parallel, whereas the GMM-S-Forest algorithm uses a sequential process to build a GMM forest. Experiments were conducted on 32 synthetic datasets and 15 real datasets to evaluate the performance of the new algorithms. The results have shown that the proposed algorithms outperformed the existing popular methods: Silhouette, Elbow and Gap Statistic, and the recent method I-nice in estimating the true number of clusters from high dimensional complex data.

1. Introduction

Clustering is an important unsupervised data analysis technique that is used to partition the data objects into a set of clusters in a way that the intra-cluster objects are more similar as compared to the inter-cluster objects [1–3]. A well known problem in data clustering is to estimate the number of clusters inherent in the data in advance as many state-of-the-art clustering techniques require the number of clusters as an input parameter [4–6]. This problem becomes more challenging in clustering high dimensional data which contain a large number of clusters because the number of clusters specified for a clustering algorithm can adversely affect the clustering results. Estimating the number of clusters has a great implication in clustering related applications [7,8]. For example, in marketing, the number of customer segments generated by a clustering algorithm from customer data has a big impact on the findings of good target customer groups for effective product promotions.

Several methods have been proposed to estimate the true number of clusters inherent in data, such as the popular ones of Silhouette [9], Elbow [10] and Gap Statistic [11]. These techniques determine the number of clusters in a dataset by evaluating a sequence of clustering results in a range of different cluster numbers. They work well on the data with a small number of clusters [12–17] but they are not suitable to complex data sets with many clusters due to their computational complexity and incompetence in dealing with complex data.

Clusters in real world data are usually categorized by the sets of objects forming high density regions in the data space. Density-based clustering methods are often effective in finding the number of clusters in data [18,19]. DBSCAN is the classical example of the density-based clustering methods. Some latest density peaks clustering algorithms are proposed in [20,21]. However, these density-based clustering methods are incapable of working on high dimensional large data due to computational complexity. Estimation of the number of clusters in high dimensional data with a large number of clusters is a challenging research problem and effective methods are still rare if not unavailable.

Recently, it was found that the high density regions that categorize the clusters in high dimensional data can be identified from one dimensional distributions of distances between the objects in the dataset and some observation points assigned to the data space. The distance distribution of objects with respect

[*] Corresponding author at: College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China.
E-mail addresses: azhar@szu.edu.cn (M. Azhar), jj.li@szu.edu.cn (M.J. Li).

https://doi.org/10.1016/j.asoc.2019.105891
1568-4946/© 2020 Elsevier B.V. All rights reserved.

to an observation point in the object space carries the information of high density regions in the dataset. By modeling and analyzing the one dimensional distance distribution, the number of clusters in the dataset can be estimated and the initial cluster centers can be identified. Following this observation, the I-nice method was proposed in [22] to estimate the number of clusters and identify the initial cluster centers. The experimental results have shown that I-nice can outperform the popular methods: Elbow, Silhouette and Gap Statistic on high dimensional data. However, I-nice performance deteriorates when the high dimensional dataset contains a large number of clusters as the distance distributions are unable to carry all high density regions in the input dataset and some clusters are missed.

In this paper, we propose a hierarchical method for estimating the true number of clusters and identifying the initial cluster centers in high dimensional data with many clusters. At the root, an observation point is assigned to the data space of the input dataset, and the distance distribution of the objects with respect to the observation point is computed. Then, a Gamma Mixture Model(GMM) is built with the EM algorithm to partition the dataset at the root into subsets, which are defined as the child nodes of the root. At each child node, a new observation point is assigned to the data space of the subset, and a new GMM model is built with EM to further partition the subset. This partitioning operation is recursively conducted on the child nodes until the leaf nodes that satisfy the termination conditions are obtained and a GMM tree is generated. Finally, from the leaves of the GMM tree, a set of initial cluster centers are identified and the true number of clusters is estimated.

Two GMM forest algorithms are further proposed to ensemble multiple GMM trees to handle high dimensional data with a large number of clusters. One is called the GMM-S-Forest which is sequentially built and the other one is the GMM-P-Forest which is built in parallel. The GMM-S-Forest builds the first GMM tree for identifying the initial set of cluster centers. Then, the initial set is appended with new cluster centers discovered from the following GMM trees until no new cluster center is found. The GMM-P-Forest builds multiple GMM trees in parallel and ensembles the results of multiple GMM trees to produce the final set of initial cluster centers whose number is taken as the estimation of the true number of clusters. The GMM-P-Forest can run faster in the distributed environment.

We conducted experiments on 32 synthetic datasets and 15 real world datasets, and compared the results of the GMM-Tree and GMM forest algorithms (i.e., GMM-S-Forest and GMM-P-Forest) with the results of Elbow, Silhouette, Gap Statistic and I-nice. The results have shown that GMM forests significantly outperformed all other algorithms in estimating the true number of clusters, especially on the high dimensional datasets with up to 100 clusters.

The rest of this paper is organized as follows. Related work is presented in Section 2. Section 3 reviews the I-nice method. Section 4 presents the GMM-Tree algorithm and Section 5 presents two GMM Forest algorithms. Section 6 shows the experiments on both synthetic and real datasets and discusses the comparison results. Conclusions and future work are given in Section 7.

2. Related work

Finding the number of clusters in data is a classical problem in cluster analysis [1,23–27]. Widely used methods include Elbow [10], Silhouette [9] and Gap Statistic [11]. However, these methods are not working well on datasets with many clusters [22]. Density based methods are effective in finding clusters of any shapes in data but they do not work well on high dimensional data due to computational cost. Therefore, it is necessary to develop new methods for high dimensional data with a large number of clusters and complex clustering structures. In this section, we review some popular and recent methods for finding the number of clusters in data and discuss their shortcomings in dealing with high dimensional data.

Silhouette method [9] computes an index value based on the relative difference between the intra-cluster and the inter-cluster distances of objects. The silhouette value is a measure of how similar an object is to the intra-cluster objects in comparison to the inter-cluster objects. The silhouette value ranges from -1 to $+1$ and a high value indicates that the object is well matched to its own cluster and poorly matched to inter-clusters. The silhouette index value is directly proportional to the clustering result, i.e., the larger the index, the better the clustering result.

Elbow [10] is another widely used method to find the number of clusters. This method calculates the average sum of intra-cluster distances between objects and the centroids. As the number of clusters increases, the average sum decreases. By plotting this average sum value against the number of clusters, the elbow position of the curve indicates the better estimation of the number of clusters inherent in the data. Both Silhouette and Elbow methods have shortcomings. The elbow method may not show the elbow shape when clustering a dataset, which indicates that the true number of clusters is not identifiable. The silhouette method does not always produce consistent results across various datasets.

Gap Statistic [11] is a statistical method that uses Gap variance to determine the number of clusters in a dataset. This method is based on the expectation that the true number of clusters can be found in the elbow of the normalized performance plot of a clustering algorithm. The elbow is explored by generating a number of reference partitions. From the normalized performance curve, the smallest number of clusters is identified where the gain in performance is not higher than expected.

Some clustering techniques do not need the number of clusters as input parameter in advance. These techniques directly estimate the number of clusters before finding clusters in the data. One such important technique is TURN [28] that finds the knee in the curve above a threshold specified by the user where knee of a curve is defined as the point of maximum curvature in the curve made between the number of clusters and the evaluation metric. Some other such methods are density-based methods [29] which use density to find the number of clusters in the data. These algorithms calculate pair-wise distances in ambient space to find the density peaks. Based on these density peaks, the number of clusters is estimated. One example is correlation clustering (CC) [30] which estimates the number of clusters by computing positive and negative edge weights in such a way that the sum of cut edge weights is minimized. Another density based algorithm is proposed by Wang et al. [31] to handle noisy and outlier values effectively as compared to the other algebraic and geometric methods. This algorithm is based on the Bayesian nonparametric method which uses Dirichlet process to exploits the trade-off between model complexity and the data fitness.

A recent algorithm is DD (Clustering Algorithm Based on Density and Distance) [18] which is also based on density peaks. DD constructs the decision graph by computing the local density of each data point in the dataset and the shortest distance among each data point and other data points with higher local density. After that, it selects the cluster centers based on the decision graph which is constructed in the first step. At last, it puts the remaining data points into the nearest cluster with higher local density.

Even though these density based methods perform well in detecting the outliers and finding the arbitrarily shaped clusters but they have the following shortcomings: (i) They do not perform

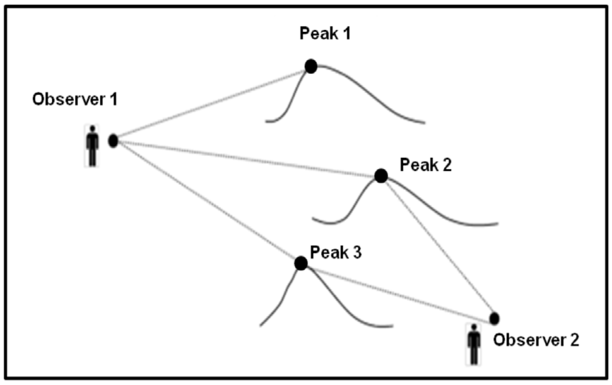


Fig. 1. Two observers observe peaks of 3 hills.

well on high dimensional data; (ii) they have very high computational cost and (iii) the clustering results and the estimated number of clusters is very sensitive to the input parameters. As most of these density based methods use pair-wise correlations to compute the similarity of data objects, they are not robust for densely distributed subspaces.

The I-nice method is a new method recently proposed by Masud et al. [22]. It uses a different approach to identify the number of clusters in data. As we use I-nice in our proposed method as a base function to estimate the true number of clusters in high dimensional data with many clusters, we give an overview of I-nice in the next section.

3. Overview of I-nice

In cluster analysis, the objects to be clustered in a dataset are the points in the high dimensional space defined by the set of attributes in the dataset. Clusters are categorized by a set of small regions with high density points. These small regions are recognized as the centers of the clusters. A terrain of hills can be used to mimic clusters in a dataset. The peaks of hills are the high density regions containing cluster centers. To find the number of hills in a terrain, a human observer is sent to the terrain to count the number of peaks. As shown in Fig. 1, two human observers count the peaks of three hills in the terrain. Whether an observer is able to see all peaks of hills depends on the location of the observer in the terrain. In a terrain with many mountains and hills, a number of observers need to be sent to different locations to observe the hills from different view angles in order to count all hills, because some mountains can block the sight of an observer from seeing the hills behind the mountains. For example in Fig. 1, Peak 2 blocks Peak 1 from the sight of Observer 2 who cannot see Peak 1.

By mimicking the clusters in a dataset as mountains and hills in a complex terrain, the observation points are allocated in the data space to observe the high density regions of clusters. The distances between an observation point and all objects in the dataset are calculated. The density curve of this distance distribution contains various peaks which are the mappings of the high density regions in the data space. Identification of these peaks from a distance distribution is similar to counting the peaks of hills by a human observer in the terrain. This metaphor can be used to Identify the number of clusters and initial cluster centers in the dataset. The method is named as I-nice [22].

Fig. 2 shows a dataset in two dimensional space containing 5 clusters. Two observation points, OP1 and OP2 are allocated at different locations of the two dimensional space. Fig. 3 shows the distance distributions with respect to OP1 and OP2. We can see

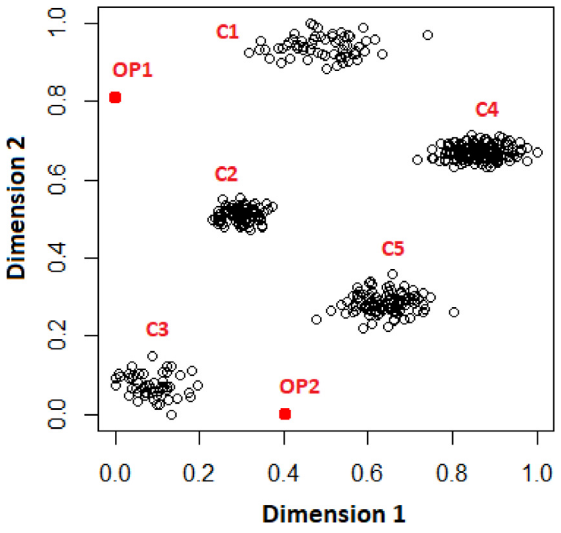


Fig. 2. Two observation points observe 5 clusters.

that the distance distribution with respect to OP1 has three peaks, whereas the distance distribution generated from OP2 has two peaks. These peaks carry the information of dense regions of clusters in the dataset. Due to the difference of locations of OP1 and OP1, the peaks of the two different distance distributions reflect different dense regions in the dataset. By analyzing the distance distributions, the dense regions in the dataset can be identified, therefore, the number of clusters and initial cluster centers being found. However, a single distance distribution may not reveal the exact number of dense regions in the dataset. In order to identify all dense regions, several distance distributions from multiple observation points at different locations are analyzed and their results are integrated to get the final result. Below, we review the steps to analyze the distance distributions and discover the number of clusters and the initial cluster centers in the dataset.

3.1. Distance distributions with respect to observation points

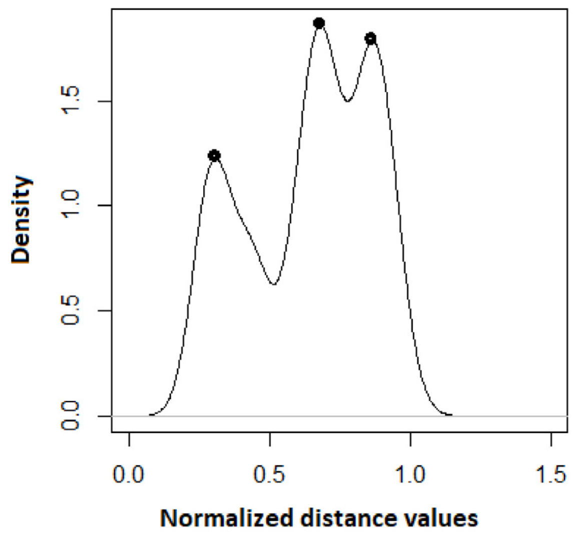
To find the number of clusters in a dataset, the distance distributions between the observation points and the objects in the dataset are first calculated. Let R^m be a data domain of m dimensions and $X \subset R^m$ a set of N objects $\{X_1, X_2, X_3, \ldots, X_N\}$ in R^m. Let $p \in R^m$ be an observation point to X randomly generated from a uniform distribution. The distance vector $X_p = (x_1, x_2, \ldots, x_N)$ is obtained by calculating the distances $x_i = d(X_i, p)$ $(1 \leq i \leq N)$ where $d(.)$ is the Euclidean distance function. This distance vector represents a distance distribution of objects in X with respect to the observation point p. In the distance vector, each distance value is indexed by its corresponding object id from which the object can be retrieved from the input dataset.

3.2. Model the distance distributions with GMMs

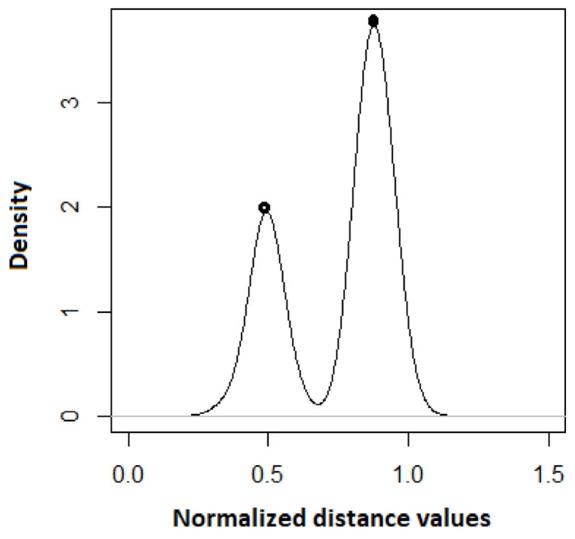
The distance distribution of X_p computed in the previous step is modeled as a Gamma Mixture Model of

$$P(x|\theta) = \sum_{j=1}^{M} \pi_j g\left(x|\theta_j\right), x \geq 0 \tag{1}$$

where π_j is the mixing proportion of the Gamma component j and θ_j are the parameters of the Gamma component j, including the shape parameter α_j and the scale parameter β_j, and M represents the number of Gamma components. The condition $\sum_{j=1}^{M} \pi_j = 1$



(a) Distance distribution with respect to OP1.



(b) Distance distribution with respect to OP2.

Fig. 3. Distance distributions of objects in the dataset of Fig. 2 with respect to the observation points OP1 and OP2.

must hold to authenticate that $P\left(x|\theta\right)$ is a well-defined probability distribution. The density function of each Gamma component is

$$g\left(x|\alpha_j,\beta_j\right)=\frac{x^{\alpha_j-1}e^{\left(\frac{-x}{\beta_j}\right)}}{\Gamma\left(\alpha_j\right)\beta_j^{\alpha_j}},\alpha_j>0,\beta_j>0 \tag{2}$$

where the Gamma function $\Gamma(x)$ is defined as $\Gamma\left(x\right)=\int_0^\infty t^x-1\,e^{-t}dt$, a definite integral for $\Re[x]$. The parameters of the GMM in Eq. (1) are solved by maximizing the log likelihood function as

$$L\left(\theta|X_p\right)=\sum_{i=1}^{N}log\left(\sum_{j=1}^{M}\pi_j g\left(x_i|\alpha_j,\beta_j\right)\right) \tag{3}$$

Eq. (3) is solved using the EM algorithm [32]. For each observation point p, ($MaxM$ -1) Gamma Mixture Models are built from the distance distribution X_p and fitted with the EM algorithm. $MaxM$ is the maximum number of GMMs to be built.

3.3. Select the best fitted GMM with AICc

Since the number of components of the GMM is not known in advance, a series of GMMs in a range of components from 2 to a given maximum number $MaxM$ are solved. The best fitted model among these GMMs is selected using the second order variant of Akaike Information Criterion (AICc) [33,34] defined as

$$AICc=-2\log\left(L\left(\theta^*\right)\right)+2q\left(\frac{N}{N-q-1}\right) \tag{4}$$

where $L\left(\theta^*\right)$ is the maximum log-likelihood value of a GMM, q is the number of parameters and N is the number of distances in X_p. The GMM with the smallest value of AICc is considered as the best fitted model for that observation point. If $N\gg q$ and $(N-q-1)\rightarrow N$, then the AICc is equivalent to AIC in the model selection.

The number of components in the best fitted model is the observed number of peaks in the distance distribution which is considered as the number of clusters in the dataset observed by the observation point. After the Gamma component distributions are found, the objects in the neighborhood of the peak of each Gamma component are extracted through the object ids. Then, the mutual distance matrix is computed and the object with the minimum distance from other objects near the peak is selected as the initial cluster center. The identified number of clusters and the initial cluster centers are used as input parameters to the K-means algorithm or other clustering algorithms to cluster the data.

Two algorithms have been developed based on the I-nice method [22]. The I-niceSO algorithm uses multiple observation points to compute multiple distance distributions. Several best fitted GMM models are selected from these distance distributions. The model with the largest number of components is chosen as the final model whose number of components is used as the estimated number of clusters in the input dataset. The I-niceSO performs better if data contains only a few clusters but is not suitable for datasets with many clusters as the distance distribution by a single observation point may not explore all clusters in the dataset as discussed in [22]. The I-niceMO was proposed as an enhancement of the I-niceSO. The I-niceMO uses all best fitted GMMs from all observation points to identify the initial cluster centers. The sets of initial cluster centers from different GMMs are merged into one set by dropping the initial cluster centers who are equal to or close to other initial cluster centers. The final set of initial cluster centers contains only the centers whose distances to other clusters' initial centers are greater than a given threshold. The number of initial cluster centers in the final set is the estimated number of clusters in the input dataset. The I-niceMO performs better on datasets with more clusters because its initial cluster centers are identified from multiple observation points who can compensate for each other to avoid missing clusters.

3.4. Limitations of the I-nice method

Even though I-niceMO performs better than I-niceSO, but still it has three major limitations which are as follows

- It generates a single level GMM from the dataset which may not be able to discover all clusters from the datasets with a large number of clusters as discussed in Section 4.1.

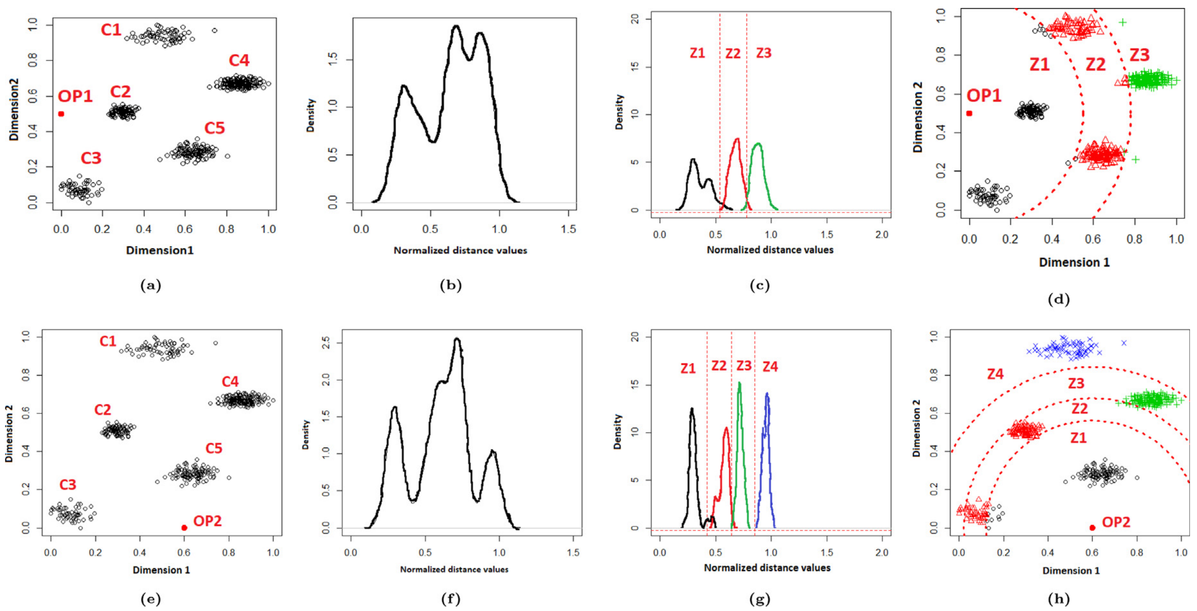


Fig. 4. Mappings of 5 clusters to two distance distributions with respect to two observation points.

- It assigns observation points randomly which may result in similar distance distributions when the observation points are close to each other. In such case, some clusters may be missed by the observation points. The details are discussed in Section 4.2.1.
- It does not provide a method to estimate the possible number of components in the GMM model, which results in solving more GMMs and is time consuming as discussed in Section 4.2.2.

These limitations are removed in our proposed hierarchical GMMs method discussed below.

4. Hierarchical GMMs

In this section, we present a divide-and-conquer method to identify the number of clusters and initial cluster centers in a high dimensional dataset with many clusters. This method uses I-nice with some modifications as a tool to partition the input dataset hierarchically into subsets and build a tree of GMMs. The leaf nodes of the GMM tree represent the candidates of possible clusters from which the number of clusters and initial cluster centers in the dataset are found. We first introduce the idea of using the multiple level GMMs to partition a dataset and then present the GMM-Tree algorithm to build a GMM tree.

4.1. Multiple level GMMs

Let X be a dataset of N objects $\{x_1, x_2, x_3, \dots, x_N\}$ with m dimensions and p an observation point which is allocated in the space of X. The distance vector $X_p = (d_1, d_2, \dots, d_N)$ is obtained using the distance function $d(x_i, p)$ for $1 \leq i \leq N$. The peaks of the distribution curve of X_p with respect to the observation point p reveal the information of clusters in X. If X contains many clusters, the distance distributions with respect to different observation points reveal different pieces of information about different clusters in X. However, the full information of all clusters in X may not be revealed by the set of the observation points due to the complexity of clusters.

Fig. 4(a) shows a dataset containing 5 clusters. The observation point OP1 is allocated at the middle left side of the data space. Fig. 4(b) is the distance distribution with respect to OP1 with 3 peaks. Fig. 4(c) shows the component distributions of the best fitted GMM obtained by I-nice. The 3 component models partition the distances into 3 zones and the 2 vertical lines are the boundaries of the 3 zones. Fig. 4(d) shows the 3 zones in the space of the original dataset. We can see that the 3 zones partition the dataset into 3 subsets. The first zone Z1, which is the first component of the GMM shown as the first left peak in Fig. 4(c), covers 2 clusters C2 and C3. Similarly, the second zone Z2 covers 2 clusters C1 and C5, and the third zone Z3 covers one cluster C4. Fig. 4(d) illustrates that the observation point OP1 can only identify cluster C4 uniquely but is failed to identify other 4 clusters due to its location in the data space. We can observe that the clusters in the same zone have similar distances to the observation point and they are reflected in the distance distribution in one peak so they cannot be distinguished by the GMM.

Fig. 4(e) shows the same dataset with a different observation point OP2 allocated at the bottom of the data space. Fig. 4(f) is the distance distribution with respect to OP2, which has 4 peaks. Fig. 4(g) shows the best fitted GMM with 4 components that form 4 distance zones bounded by the 3 vertical lines. Using OP2 as the center and the distances of three boundary lines as radii, the 4 zones are drawn in the data space as shown in Fig. 4(h). We can see that the observation result of OP2 is much better than the result of OP1. The 3 clusters C1, C4 and C5 are uniquely identified. However, the clusters C2 and C3 are still not identifiable in this GMM. This case shows that for a dataset with many clusters, it is difficult or impossible to allocate one observation point in the data space to observe all clusters separately in the distance distribution. This is the major shortcoming of the I-nice method.

However, from Fig. 4(d) and (h), we can observe that the components of the best fitted GMM generate a good partition of the data space and separates the clusters in different zones clearly. We can make use of this partition to further identify the clusters in the same zone. For example, we can take the objects in zone Z2 of Fig. 4(h) as a new dataset shown in Fig. 5(a) and allocate a new observation point OP3. Then, we build a GMM on this dataset to identify the number of clusters in it as shown in Fig. 5(d). This process can be recursively applied to all zones at different levels, which results in a GMM tree as shown in Fig. 6. We can see that each leaf of the GMM tree contains the distance distribution of only one peak which indicates that the dataset in

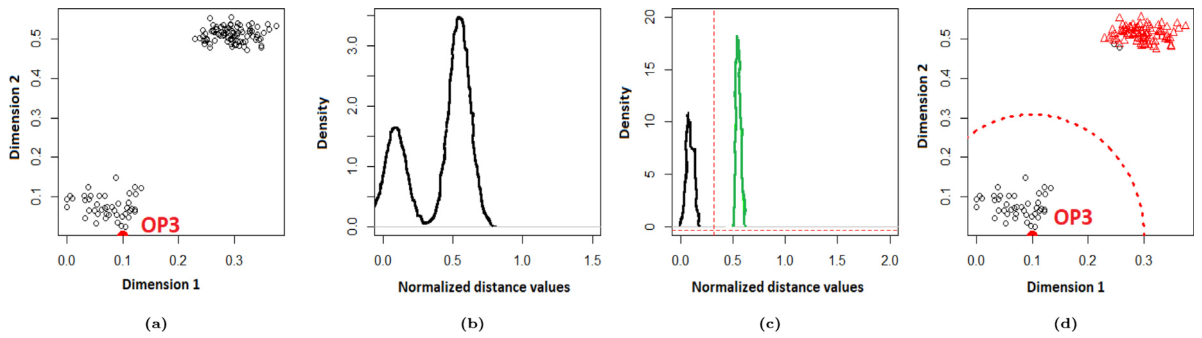


Fig. 5. Identification of individual clusters from the subset of objects in zone Z2 of Fig. 4(h).

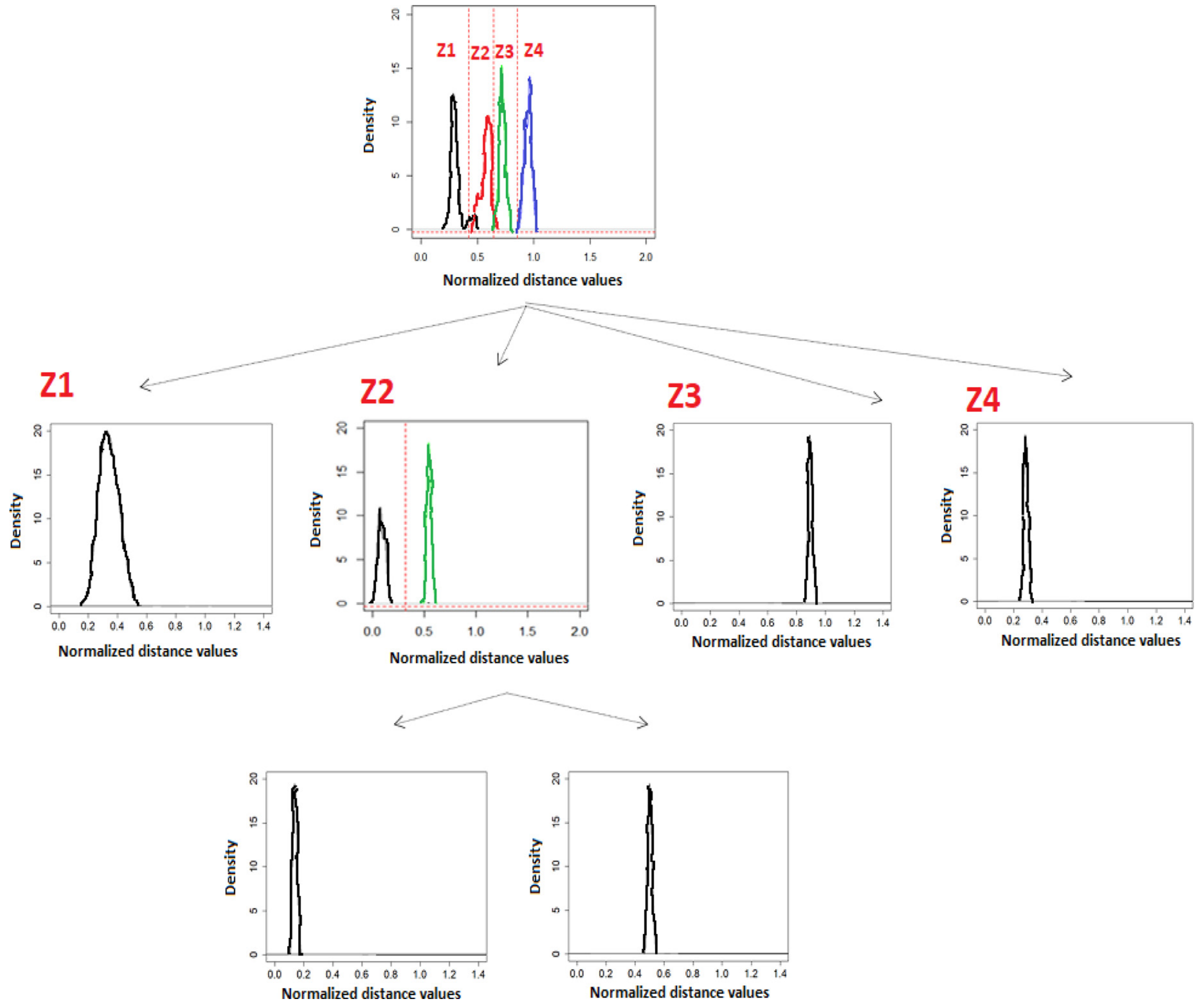


Fig. 6. A GMM tree for identifying the number of clusters in a dataset.

the leaf has only one cluster. This hierarchical way of building the multiple level GMMs can solve the problem of estimating the true number of clusters and identifying the initial cluster centers in complex high dimensional data with many clusters.

4.2. Steps to build a GMM tree

A GMM tree as shown in Fig. 6 represents a hierarchical partition of the dataset at root. Each node represents a subset of the dataset. Except for the leaf nodes, each node has a GMM built from the subset of the dataset at the node and the components of the GMM form a partition of the dataset at the node whose subsets are represented in the children nodes. To generate a GMM tree from a dataset and estimate the true number of clusters in data, the most important task is to obtain the best partition on the dataset at each node which can separate the clusters in the partition. The following five steps are crucial in carrying out this task:

1. Allocation of the observation points at the data space of a node,
2. Selection of the distance distribution based on the estimation of the maximum number of peaks,
3. Selection of the best fitted GMM for the node,

4. Stopping criteria to terminate the partition at a leaf node, and
5. Determination on whether the dataset at a leaf node is a cluster or a set of outliers.

The first three steps are performed by the revised I-nice method. To improve the performance of the first three steps, we discuss the revisions on I-nice in the following subsections. We also present the methods for the last two steps: the stopping criteria for making a leaf node and the postprocessing method to determine whether a leaf node is a cluster or a set of outliers.

4.2.1. Allocation of observation points

Given a dataset, I-nice allocates a few observation points randomly generated from a uniform distribution in the data space. This method does not consider the relative locations of the observation points among themselves and to the objects in the dataset. If the observation points are close to each other in the data space as shown in Fig. 7(a), the distance distributions will be similar to each other, which will affect their ability in observing clusters in the dataset and waste the computation time in finding the best fitted GMMs from the distance distributions with respect to these observation points. For example in Fig. 7(a), these observation points will generate the distance distributions similar to each other and carry similar information about clusters in the dataset. If some clusters are missed in one distribution, they will likely be missed in other distance distributions as well.

To solve the above problem, we use a pseudo random method to allocate observation points in the data space to replace the fully random allocation method in I-nice. In this method, we first calculate the range of values in each dimension of the dataset and find the borders of the dataset in the data space. Then, we randomly allocate observation points on the border lines as shown in Fig. 7(b). In this way, the observation points are apart to each other and also apart from the dense objects in the dataset. Therefore, diverse distance distributions can be obtained, which are able to identify more clusters in the dataset.

At each node, we use the pseudo random method to generate 5 observation points and calculate the distance distributions with respect to them. From the 5 distance distributions, we select one distribution to build GMMs as follows

1. Given a distance distribution vector $X_p = (d_1, d_2, \dots, d_N)$, we estimate the distance density function

$$p_i(d_i) = \frac{1}{N}\sum_{j=1}^{N}\frac{1}{\sqrt{2\pi}h}e^{\frac{(d_j-d_i)^2}{h}} \tag{5}$$

where h is the bandwidth of the Gaussian kernel which is computed as in [35].

2. For each distance value $d_i \in X_p$, we calculate its density value p_i using Eq. (5) and sort the density values in the descending order of d_i. Then, we use a subsequence window of 3 elements (p_{i-1}, p_i, p_{i+1}) to find the density peaks based on the condition of $p_i > p_{i-1}$ and $p_i > p_{i+1}$, where i is the index of sorted distances.
3. The distance distribution vector $X_p^{\tau^*}$ with the largest number of peaks τ^* in the density function is selected as the one which has the best observation of clusters among the 5 observation points.

We consider that each peak in the density estimate function reflects a dense region in the data space and more peaks carry denser regions in the data space, therefore, observing more clusters.

4.2.2. The number of GMMs to calculate

Another revision to I-nice is to reduce the number of GMMs to generate because it is computationally expensive to use the EM algorithm to solve a GMM. Since we know the largest number of peaks in the distance density function in the selected distance distribution, we can make a judgement that the number of components in the best fitted GMM would be close to the number of peaks in the distance density function. Instead of calculating GMMs with a range of components from 2 to the given maximum number, we calculate GMMs in the range of $[\tau^* - \Delta, \tau^* + \Delta]$ where τ^* is the number of peaks found in the distance density function and Δ is set to 2. In this setting, only 5 GMMs are computed and a lot of computation time is saved. The maximum number of components is dynamically adjusted by τ^*.

Same as I-nice, after 5 GMMs are calculated using EM, the best fitted GMM is selected according to the measure of AICc. The components of the best fitted GMM form a partition of the dataset at the node. The objects in the subsets of the partition are identified based on the probability of each object appearing in each component of the GMM.

4.2.3. Stopping criteria

The subsets of objects in the partition of a node by the best fitted GMM are made as the child nodes of the current node. After a child node is generated, a decision has to be made on whether the node is a leaf node or it requires further partitioning. Two stopping criteria are defined to make the decision.

1. If the number of objects in the dataset of a child node is smaller than a given threshold, the node is made a leaf node.
2. Otherwise, the observation points are allocated in the data space and the distance distributions with respect to the observation points are calculated. If the distance density functions contain only one peak, the node is made a leaf node, otherwise, one distance distribution with the largest number of peaks is selected for further partitioning.

The stopping criteria create a recursive process to build a GMM tree from a dataset.

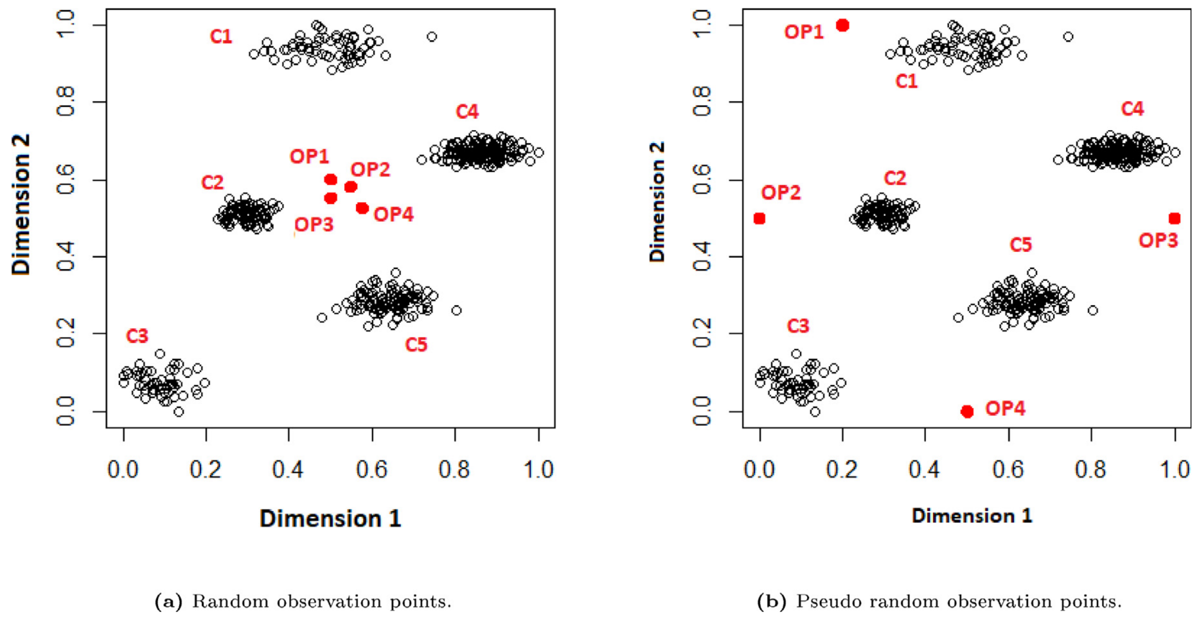
4.2.4. Postprocessing of leaf nodes

The set of leaf nodes in a GMM tree provides the initial candidates of clusters in the dataset. However, due to outliers and irregular distributions of clusters in the dataset, not all leaf nodes are justifiable as clusters. Therefore, the following postprocessing steps are developed to verify the clusters of leaf nodes.

1. At each leaf node, a neighborhood around the peak of the density distribution of the component model is determined and the objects in the neighborhood are selected from the dataset. Then, the mutual distances between objects are computed and the object with the minimum distance from other objects near the density distribution peak is selected as the initial cluster center.
2. If no object with a clear high density is found in the neighborhood, the set of objects in the leaf node is considered as outliers so no cluster is found in the leaf node.

After finding all initial cluster centers, we need to merge the clusters based on some threshold value. To dynamically find the threshold value, we use the following steps.

1. Compute the matrix of the mutual distances between these cluster centers and select the values in the lower triangle section of the matrix below the diagonal elements.



(a) Random observation points. (b) Pseudo random observation points.

Fig. 7. Two ways of generating observation points in the data space.

2. Calculate the *Max* and *Min* values from the lower triangle matrix. *Max* is the largest distance value and *Min* is the smallest distance value.
3. Divide the range [*Min*, *Max*] into intervals in a way that each interval is based on (*Min* + x*0.1) where x is 1,2,.. until the value does not exceed *Max* distance value. The value computed from (*Min* + x*0.1) is considered as the expected threshold.
4. Merge the initial cluster centers whose distances are smaller than the expected threshold to generate a new set of clusters. For each expected threshold value, we get a set of clusters. For example, if *Min* value is 0.5 and *Max* value is 0.8, then there will be 3 expected threshold values 0.6, 0.7 and 0.8. Considering each of these 3 values as expected threshold values, we get three sets of clusters.
5. For each set of clusters, we compute the *Wemmert-Gancarski* (*WG*) index value [36] and the set of clusters with the highest *WG* index is considered as the final estimation of the clusters. *WG* index value is computed as follows

$$WG = \frac{1}{N}\sum_{k=1}^{K} \max(0, n_k - \sum_{i \in I_k} \frac{(x_i - c_k)^2}{\underbrace{\min}_{1 \le k' \le K, k' \neq k} (x_i - c_{k'})^2}) \qquad (6)$$

where K is the number of clusters, N is the number of objects in the dataset, n_k is the number of objects in the kth cluster, x_i is the ith object in the kth cluster, and c_k is the center of the kth cluster. The large the *WG* index, the more compact and separated the clusters.

The initial cluster centers are computed for all clusters in the final set of clusters. The number of the initial cluster centers in the final set of clusters is considered as the true number of clusters in the dataset identified by the GMM tree.

4.3. GMM-Tree algorithm

In this section, we present the GMM-Tree algorithm that uses the revised I-nice method, the stopping criteria to determine the leaf nodes and the leaf node postprocessing method to build a GMM tree from a dataset and identify the number of clusters in the dataset, as well as the initial cluster centers, from the GMM tree. In the tree building process, the input dataset is made as the root node and the revised I-nice is used to partition the dataset at the root node into subsets which are made as the children nodes of the root node. As discussed in Section 3, the dataset of the root node is partitioned by the best fitted GMM and each component of the GMM represents a subset of the dataset in the child node. After the root is partitioned into the children nodes, the subset of data in each child node is evaluated by the stopping criteria to determine whether it is a leaf node or not. If a node is not a leaf node, it is further partitioned by the revised I-nice to generate the next level child-nodes which are also evaluated by the stopping criteria. This recursive process continues until all leaf nodes are found. A simple example of a GMM tree is illustrated in Fig. 6. Each leaf node contains a subset of the input dataset and is a candidate cluster. Finally, the postprocessing method is used to determine the initial cluster centers and the number of clusters in the dataset.

The pseudo code of the GMM-Tree algorithm is shown in Algorithm 1. The input to the algorithm is a dataset X. The outputs of this algorithm are the estimated number of clusters K and the set of initial cluster centers K_{cent}. This algorithm uses Breadth-first search (BFS) approach to build GMM tree in which all the nodes at the same level of tree are explored and then the child-nodes of the next level are explored. At each level, the non-leaf nodes are saves in a queue Q and explored as first-come-first-serve (FCFS) basis. The algorithm starts with generating the root by calling function GMM-TreeNode() in Line 4. GMM-TreeNode() is the pseudo code of the revised I-nice (Lines 18–33) that is used to find the best fitted GMM and partition the input dataset into subsets using the components of the GMM. In this first call, the root of a tree is created first and the child-nodes are saved in queue Q. The input dataset X is partitioned into subsets by the best fitted GMM from the distance distributions with respect to P observation points. The set of child-nodes, created under the root, represents a subset of data partitioned by components of the GMM.

After the nodes under the root are created, each node under the root is explored using Breadth-first search (BFS) approach. This process is shown in the *while* clause of Lines (5–15). If the node is a leaf, it is marked as a leaf node and its subset of data

Algorithm 1: GMM-Tree

```
   Input: Dataset $X = (x_1, x_2, ..., x_N)$
   Output: Number of clusters $K$, initial cluster centers $K_{\text{cent}}$
1  Initialize $P$ = 5;      ▷ The number of observation points;
2  $\Delta$ = 2; ▷ Used to set the number of GMMs to be generated (Section 4.2.2);
3  $nOB$ = 10;      ▷ The threshold for the minimum number of objects
4  $Q \leftarrow$ Call GMM-TreeNode($X$, $P$);   ▷ Generate a GMM tree root and save child-nodes in a Queue $Q$
5  while $(size(Q) \neq 0)$ do
6                          ▷ Check if $Q$ has some nodes to be explored
7      $X_{\text{sub}}$ = $Q.pop()$;▷ Pop up the front node from the $Q$ (Tree is implemented by Breadth-first search approach using queue)
8      if $(objects\ in\ X_{sub} > nOB)$ then
9          child-nodes $\leftarrow$ Call GMM-TreeNode($X_{\text{sub}}$, $P$);
10         if $(child\text{-}nodes.size > 1)$ then
11             $Q$.push(child-nodes) ;   ▷ Add child-nodes at the end of $Q$
12         else
13             Add $X_{\text{sub}}$ to leaves-list $\Omega$;
14     else
15         Add $X_{\text{sub}}$ to leaves-list $\Omega$;
16 Merge Leaves of GMM-Tree in $\Omega$ based on highest $WG$ Index value using Eq. (7) ;
17 Output the number of clusters $K$ and initial cluster centers $K_{\text{cent}}$ ;
18 Function $GMM\text{-}TreeNode(S,P)$
19     Generate $P$ pseudo random observation points;
20     for $i := 1\ to\ P$ do
21         Compute the distance distribution vector between objects of S and observation point $p_i$ ;
22         Identify the number of peaks $\tau_i$ in distance distribution vector generated by observation point $p_i$ by using the distance density function Eq. (5) ;
23     Keep the maximum number of peaks $\tau^* = \max(\tau_1, \tau_2, \ldots, \tau_P)$ ;
24     Keep the distance vector $X_p^{\tau^*} = (d_1^{\tau^*}, d_2^{\tau^*}, \ldots, d_u^{\tau^*})$ ;
25     if $(\tau^* > 1)$ then
26         for $r := (\tau^* - \Delta)\ to\ (\tau^* + \Delta)$ do
27             Model $X_p^{\tau^*}$ to GMM($r$) using Eq. (1) ;
28             Solve Eq. (3) by EM and calculate AICc using Eq. (4) ;
29         Select the best fitted GMM $\theta_{best}$ with the minimum AICc value ;
30         Separate the object IDs for each component of the best fitted GMM $\theta_{best}$;
31         Return the list of subsets of $S$ based on objects IDs ;
32     else
33         Return $S$;
```

X_{sub} is put into the leaves-list Ω. Then, the next node is evaluated. If the node is not a leaf node, GMM-TreeNode function is called again to further partition its subset of data and create a set of new child-nodes under the node. After exploring all child-nodes in the same level of the GMM tree, the process starts to evaluate the each new child node at lower lever of the tree to make it as a leaf node or further partition it again using GMM-TreeNode function. This process continues until all new child-nodes are marked as leaf nodes and a GMM tree is generated. The leaves-list Ω contains all subsets of leaves in the GMM tree. After all leaves are obtained in Ω, they are analyzed in Line 16 of the postprocessing step to find the number of clusters and initial cluster centers which are output in Line 17.

Function GMM-TreeNode() is an implementation of the revised I-nice. This function starts in Line 19 with the generation of P observation points using the pseudo random method described in Section Section 4.2.1. Then, P distance distribution vectors with respect to P observation points are calculated in Lines (20–22) and the distance distribution vector with the maximum number of peaks is retained in Line 23. If there is only one peak in the retained vector, it is made a leaf node and the function exits. If there is more than one peak, $2\Delta + 1$ GMMs are built using EM algorithm in Lines (25–28) and the best fitted GMM is selected using AICc in Line 29. Then, the set of nodes by the GMM is returned to the upper of input dataset S.

This algorithm has three parameters, the number of observation points P, the number of the minimum objects in a subset nOB and Δ. The default values of these parameters are selected based on empirical analysis of multiple results from both synthetic and real datasets. Selections of these default values reflect the tradeoff between the accuracy of the results and the execution time of the algorithm. For instance, increasing P and Δ may increases the accuracy of the results on some datasets. However, the execution time will also increase significantly. The default values for these parameters produced the overall best results in our experiments. In our implementation of this algorithm, these parameters are not hard coded. Users can change these default values, if necessary, to tune these parameters for their datasets.

5. Forest of GMM Trees

The GMM-Tree algorithm uses multiple level GMMs to find the number of clusters in a dataset. A GMM tree is generated with a set of observation points selected at each node of the tree, except for the leaf nodes. Although the best observation point is selected from several candidates at each node, its selection cannot affect the selection of the best observation points at other nodes. In this sense, the GMM-Tree algorithm is a greedy algorithm that cannot guarantee that the set of observation points for generation of a GMM tree is the best selection. As a consequence, some clusters in a complex dataset may be missed in the GMM tree if the dataset has many clusters.

To avoid missing clusters from complex high dimensional datasets, we can use the GMM-Tree algorithm to built multiple GMM trees from the dataset, each with different sets of the best observation points, and ensemble the leaves of the GMM tress into one set of candidate clusters to estimate the true number of clusters and find the initial cluster centers in the dataset. Since different sets of observation points are used to build multiple diverse GMM trees, more inherent clusters in the dataset are found.

In this section, we propose two GMM forest algorithms for estimating the true number of clusters in a complex dataset with many clusters. One is called the GMM-P-Forest algorithm that uses the GMM-Tree algorithm to generate multiple GMM trees from the same dataset independently and in parallel, to form a GMM forest. The final set of the initial cluster centers and the number of clusters are determined from all leaves of GMM trees in the forest. The other is named as the GMM-S-Forest algorithm that uses the GMM-Tree algorithm to generate a forest of GMM trees independently and sequentially, one GMM tree at a time. The initial cluster centers obtained from the first GMM tree are made as the reference list of the initial cluster centers. The newly found initial cluster centers in the following GMM trees are

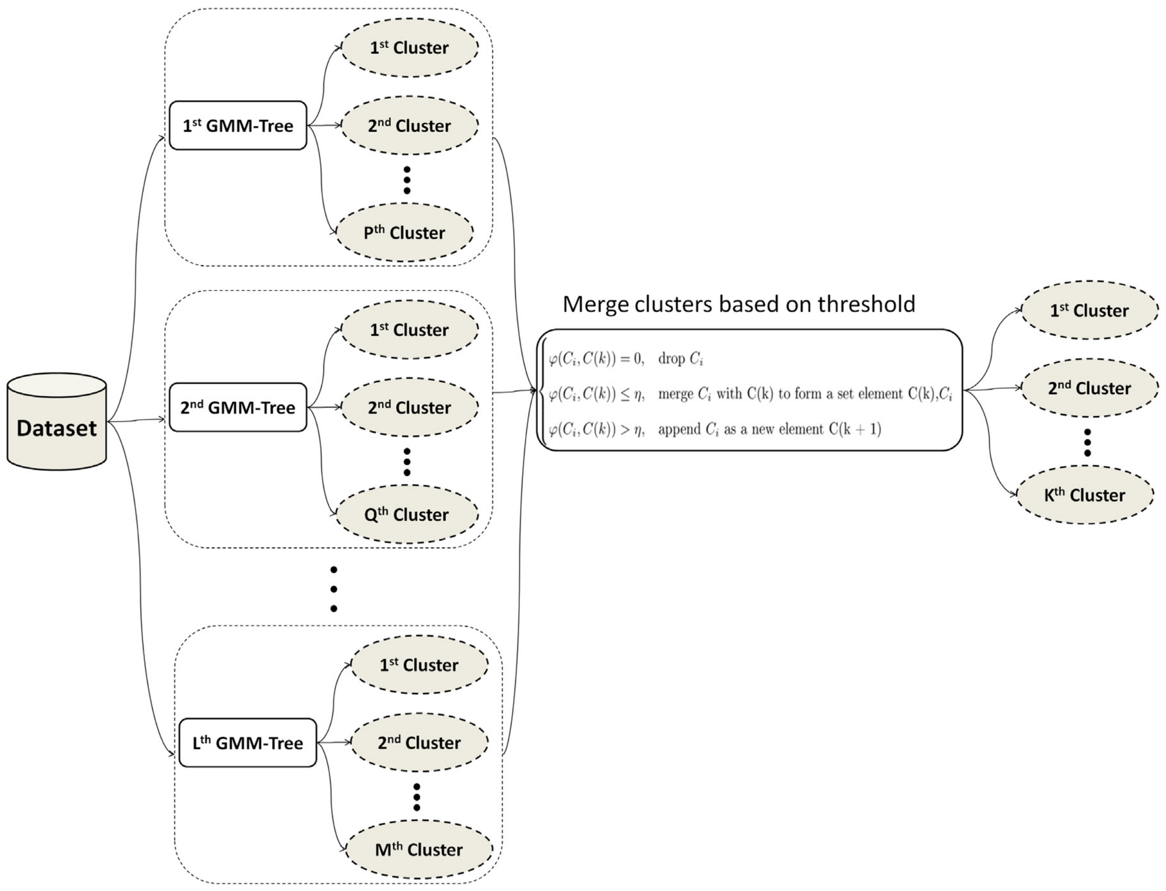

Dataset
1st GMM-Tree
1st Cluster
2nd Cluster
Pth Cluster
2nd GMM-Tree
1st Cluster
2nd Cluster
Qth Cluster
Lth GMM-Tree
1st Cluster
2nd Cluster
Mth Cluster
Merge clusters based on threshold
$\varphi(C_i, C(k)) = 0$, drop C_i
$\varphi(C_i, C(k)) \le \eta$, merge C_i with C(k) to form a set element C(k),C_i
$\varphi(C_i, C(k)) > \eta$, append C_i as a new element C(k + 1)
1st Cluster
2nd Cluster
Kth Cluster

Fig. 8. The GMM-P-Forest process to find the final set of initial cluster centers.

appended to the reference list until no new initial cluster center is found. The tree growing process will terminate and the number of the initial cluster centers in the reference list is considered as the true number of clusters in the dataset.

5.1. GMM-P-Forest algorithm

The GMM-P-Forest algorithm is a straightforward extension to the GMM-Tree algorithm for parallel execution. Fig. 8 illustrates the flow chart of building multiple GMM trees in parallel to produce the final set of the initial cluster centers and estimate the true number of clusters in a dataset. Given a dataset X, the number of GMM trees, L, is specified, and the GMM-Tree algorithm is run L times independently and in parallel on X to generate L GMM trees. The leaf nodes of all L GMM trees are integrated to produce the final set of the initial cluster centers that is used to estimate the true number of clusters in X. If it runs on multiple nodes of a computing cluster, each GMM tree is generated independently on each computing node, and the leaf nodes of the GMM trees are transmitted to the master node for integration. Each GMM tree can also be calculated on each core in the multi-core system to speed up the multiple GMM tree generation process.

The parameter L is given in advance to the GMM-P-Forest algorithm. If a small L is specified, in particular if L is smaller or equal to the number of computing nodes of the cluster, the GMM-P-Forest algorithm will be efficient in generating the L GMM trees to find the estimated true number of clusters and the set of the initial cluster centers. However, the risk is that some clusters could be missed because less sets of observation points are used in observing the clusters in a complex dataset with many clusters. A large L increases the chance of finding more clusters in data. The consequence is the high computational cost because more GMM trees have to be built and more leaf nodes are integrated. In practice, the parameter L should be specified with respect to the complexity of the dataset.

After multiple GMM trees are built using the GMM-Tree algorithm, the leaf nodes of each GMM tree are processed independently using the same postprocessing method in Section 4.2.4. In this process, the leaf nodes containing outliers are removed and the initial cluster centers are found from the remaining leaf nodes. L GMM trees result in L sets of initial cluster centers which are the candidates of the initial cluster centers in the dataset. However, they cannot be taken as the final set of the initial cluster centers because the same cluster centers can be identified by different GMM trees. The duplicate centers must be removed and the close centers must be merged. The final set of the initial cluster centers is obtained as follows.

Given L sets of initial cluster centers, we take one set as the reference list of the initial cluster centers, denoted as $CList = \{C(1), C(2), \dots, C(k)\}$. We compare the initial cluster centers in the remaining $L-1$ sets with the initial cluster centers in $CList$. For any initial cluster center C_i in the remaining $L-1$ sets, we calculate the distance $\varphi(C_i, C(k))$ where $C(k)$ is an initial cluster center in the reference list $CList$, and compare the distance φ with a given threshold η. The following three actions are taken according to the comparison.

1. If $\varphi(C_i, C(k)) = 0$, C_i is dropped from the remaining sets and $CList$ is not changed.
2. If $\varphi(C_i, C(k)) \le \eta$, C_i is merged with $C(k)$ to form a new set element $\{C(k), C_i\}$ in $CList$.

```
Algorithm 2: GMM-P-Forest
  Input: A high dimensional dataset $X = (x_1, x_2, ..., x_N)$, number of GMM trees $L$ to generate
  Output: Number of clusters $K$, list of initial cluster centers $K_{\text{cent}}$
1 Master node instructs the slave nodes to run the GMM-Tree algorithm to generate L GMM trees ;
2 Each slave node generates a GMM tree based on Algorithm (1);
3 Each slave node runs postprocessing steps to produce the set of initial cluster centers from the GMM tree;
4 Each slave node send its set of initial cluster centers to the master node;
5 Master node receives the initial cluster centers of L slave nodes;
6 Master node selects one set of the initial cluster centers $CList = \{C(1), C(2), ..., C(k)\}$ as the reference list from L sets and
  compares with other (L-1) sets;
7 Master node computes the distance $\varphi(C_i, C(k))$ as discussed in Section 5.1. The duplicate initial cluster centers are dropped and
  the close centers with the distance less than the threshold $\eta$ are merged into set elements in CList;
8 $K_{\text{cent}} \leftarrow$ Compute cluster centers of the final clusters after merging process;
9 Output the number of clusters $K$ and list of initial cluster centers $K_{\text{cent}}$;
```

3. If $\varphi(C_i, C(k)) > \eta$, C_i is appended to *CList* as a new element $C(k+1)$, which indicates that a new initial cluster center is found.

The distances between C_i and all elements in *CList* are computed. If the first condition is met, C_i is dropped. If the second condition is met with any element in *CList*, C_i is merged with the element. If the third condition is met with all elements, C_i is added to *CList* as a new initial cluster center. After all initial cluster centers in the remaining $L-1$ sets are compared with the initial cluster centers of *CList*, the means of the initial cluster centers in the set elements in *CList* are calculated to produce the final set of the initial cluster centers. The number of initial cluster centers in the final set is the estimated true number of clusters found by the GMM-P-Forest algorithm from the dataset

The pseudo code of the GMM-P-Forest algorithm is given in Algorithm 2. Lines (1–4) are the steps of generating the L GMM trees in parallel. In Line 1, the master node instructs the slave nodes to run the GMM-Tree algorithm to generate L GMM trees. In Line 2, each slave node runs GMM-Tree on X to generate a GMM tree. In Line 3, after the GMM tree is generated, each node runs the postprocessing step to produce the set of initial cluster centers from the GMM tree. In Line 4, all slave nodes send their sets of initial cluster centers to the master node.

Lines (5–7) shows the steps of integrating the L sets of initial clusters centers into the final set of initial cluster centers. In Line 5, the master node receives the L sets of initial cluster centers from the slave nodes. In Line 6, the master node selects one set as the reference list *CList*. In Line 7, the initial cluster centers in the remaining $(L-1)$ sets are compared with the initial cluster centers in *CList*, duplicate initial cluster centers are dropped, close centers are merged into set elements in *CList*, and new initial cluster centers are added to *CList*. In Line 8, the set elements in *CList* are computed to use the means as the initial cluster centers and the final set of initial cluster centers is generated. The number of the initial cluster centers in the final set is calculated as the estimated true number of clusters in the dataset. In Line 9, the number of clusters K and the set of initial cluster centers are output. The results of the GMM-P-Forest algorithm can be used as inputs to the k-means algorithm to cluster dataset X.

5.2. GMM-S-Forest algorithm

The GMM-P-Forest algorithm is efficient when running on a parallel and distributed environment. However, it requires the number of GMM trees to be given in advance. To drop this requirement, we propose the GMM-S-Forest algorithm which generates GMM trees sequentially and stops when a new GMM tree does not find any new initial cluster center.

Fig. 9 shows the flow chart of building GMM trees sequentially to find the final set of the initial cluster centers. This is done in the following steps:

1. Given a dataset X, call the GMM-Tree algorithm to build a GMM tree and produce a set of initial cluster centers. Make this set as the reference list $CList = \{C(1), C(2), \dots, C(k)\}$.
2. Call the GMM-Tree algorithm again to build a new GMM tree and produce a new set of initial cluster centers.
3. Compare these new cluster centers with the cluster centers in *CList* and append *CList* with the new cluster centers according to the same conditions used in the above GMM-P-Forest algorithm.
4. If *CList* is added with any new initial cluster center, go to Step 2. Otherwise, calculate the means of the set elements in *CList* to produce the final set of the initial cluster centers.

The pseudo code of the GMM-S-Forest algorithm is illustrated in Algorithm 3. In Line 1, the first GMM tree is generated using the GMM-Tree algorithm, and the reference list of initial clusters is created from this tree. Lines (2–6) implement sequential process of generating the GMM trees one by one. Line 3 calls the GMM-Tree algorithm to generate a new GMM tree and produce a new set of initial clusters. Line 4 compares the initial clusters in the new set and the elements of CList and Line 5 appends CList with new initial cluster centers. Line 6 checks whether any new initial cluster is found in the comparison. If no new initial cluster is found, terminate the GMM tree generation process and go to the next step. Otherwise, go Line 3 to generate another GMM tree. Line 7 computes the means of the set elements in *CList* as the initial cluster centers and generate the final set of initial cluster centers. The number of the initial cluster centers in the final set is calculated as the estimated true number of clusters in the dataset. Line 8 outputs the number of clusters K and the list of initial cluster centers K_{cent}.

```
Algorithm 3: GMM-S-Forest
  Input: A high dimensional dataset $X = (x_1, x_2, ..., x_N)$
  Output: Number of clusters $K$, list of initial cluster centers
          $K_{\text{cent}}$
1 $CList \leftarrow$ GMM-Tree($X$) ;
2 repeat
3     new-clust-centers $\leftarrow$ GMM-Tree($X$) ;
4     if (new-clust-centers $\not\subset$ CList) then
5         CList $\leftarrow$ newly found cluster centers ;
6 until (new-clust-centers $\not\subset$ CList);
7 $K_{\text{cent}} \leftarrow$ Compute cluster centers $K$ of CList ;
8 Output the number of clusters $K$ and list of initial cluster
  centers $K_{\text{cent}}$;
```

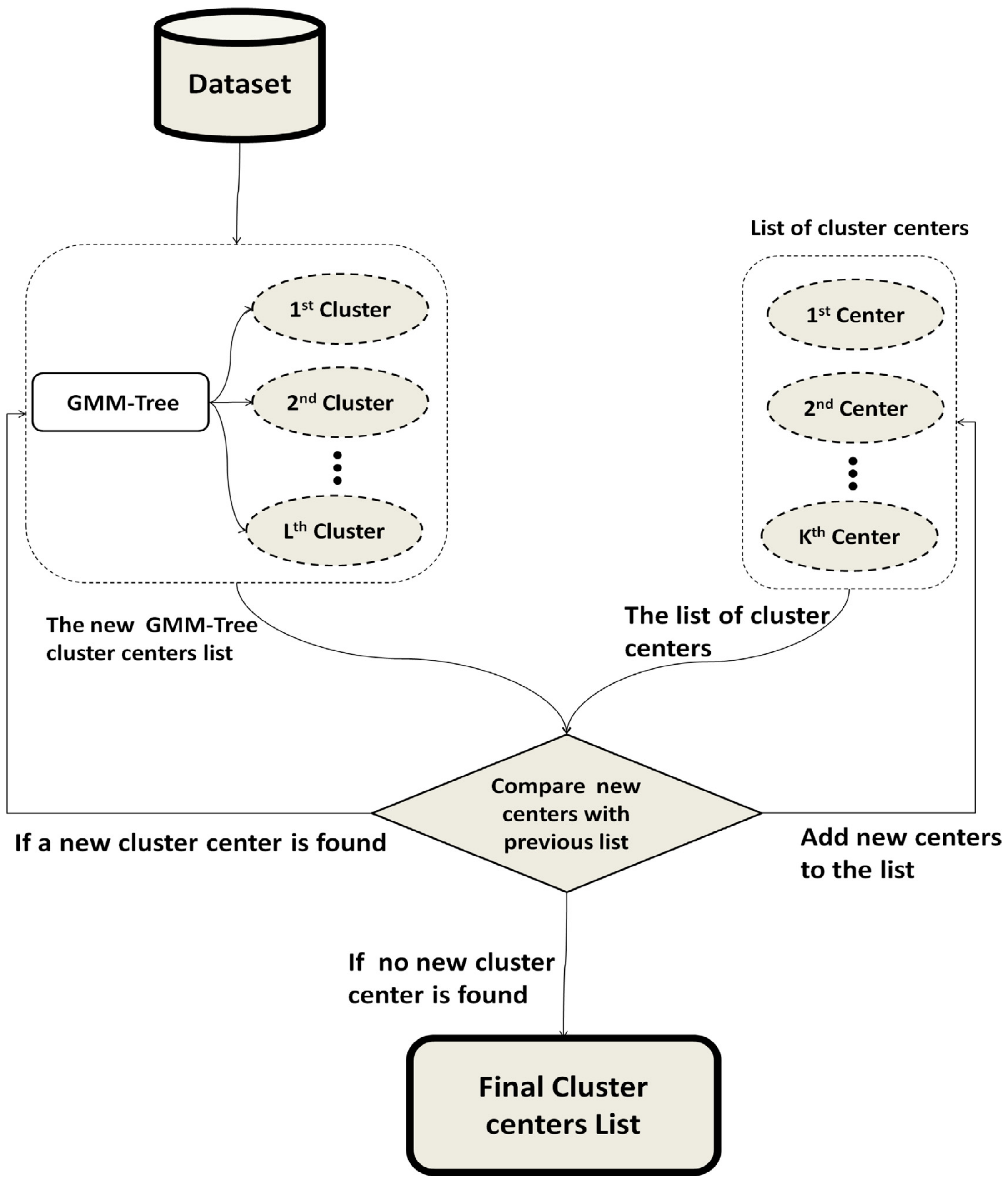

Dataset
GMM-Tree
1st Cluster
2nd Cluster
Lth Cluster
List of cluster centers
1st Center
2nd Center
Kth Center
The new GMM-Tree cluster centers list
The list of cluster centers
Compare new centers with previous list
If a new cluster center is found
Add new centers to the list
If no new cluster center is found
Final Cluster centers List

Fig. 9. The GMM-S-Forest process to find the final set of initial cluster centers.

6. Experiments

In this section, we present the results of performance analysis of the three proposed algorithms in comparison with Gap Statistic, Silhouette, Elbow, I-niceSO and I-niceMO in identifying the number of clusters inherent in the data. We did not compare our proposed methods with some other famous methods like k-means++ [37] and Modified Mountain Clustering Algorithm (MMCA) [38] as I-niceMO performed better than these methods [22]. Both synthetic and real world datasets with different numbers of clusters and classes were used in the experiments. We also present the comparison of the clustering results of the K-means algorithm using the number of clusters and the initial cluster centers obtained with the three proposed algorithms and the existing methods including Gap Statistic, Silhouette, I-niceSO and I-niceMO.

6.1. Datasets

The experiments were conducted on 32 synthetic datasets and 15 real datasets. The R package clusterGeneration [39] was used in generating the synthetic datasets. The characteristics of the 32 synthetic datasets are lised in Table 1. The second column from the left shows the number of samples in each dataset. The third and the fourth columns show the number of features and the number of clusters. The last column is the cluster separation index which indicates the degree of separation between a cluster and its nearest neighbor cluster. The larger the degree of separation, the more separated the clusters.

The real world datasets are listed in Table 2. The first 14 datasets were downloaded from the UCI repository [40], while the Aloi(short) dataset was taken from Amsterdam Library of Object Images dataset repository [41]. The Aloi(short) dataset contains 10 800 samples representing 100 objects (clusters). These

Table 1
Characteristics of the synthetic datasets.

Datasets	Samples	Features (m)	numClust	sepIndex
DS1	500	2	5	0.3
DS2	1000	2	10	0.5
DS3	1500	2	15	0.3
DS4	2000	2	20	0.5
DS5	2500	2	25	0.3
DS6	3000	2	30	0.5
DS7	3500	2	35	0.3
DS8	4000	2	40	0.5
DS9	500	10	5	0.3
DS10	1000	10	10	0.5
DS11	1500	10	15	0.3
DS12	2000	10	20	0.5
DS13	2500	10	25	0.3
DS14	3000	10	30	0.5
DS15	3500	10	35	0.3
DS16	4000	10	40	0.5
DS17	500	20	5	0.3
DS18	1000	20	10	0.5
DS19	1500	20	15	0.3
DS20	2000	20	20	0.5
DS21	2500	20	25	0.3
DS22	3000	20	30	0.5
DS23	3500	20	35	0.3
DS24	4000	20	40	0.5
DS25	500	30	5	0.3
DS26	1000	30	10	0.5
DS27	1500	30	15	0.3
DS28	2000	30	20	0.5
DS29	2500	30	25	0.3
DS30	3000	30	30	0.5
DS31	3500	30	35	0.3
DS32	4000	30	40	0.5

Table 2
Characteristics of the real world datasets.

Number	Datasets	Samples	Features	Classes
1	Appendicitis	106	7	2
2	Banana	5 300	2	2
3	Mammographic Mass	961	5	2
4	Iris	150	4	3
5	Wine	178	13	3
6	Seed	210	7	3
7	Breast Tissue	106	9	6
8	Satellite	6 435	36	6
9	Glass	214	9	6
10	Ecoli	336	7	8
11	Yeast	1 484	8	10
12	Texture	5 500	40	11
13	Libras Mov.	360	90	15
14	ISOLET	6 238	617	26
15	Aloi(short)	10 800	128	100

datasets have diverse representations in application domains, sizes, dimensions and numbers of inherent clusters.

6.2. Experimental settings

The default values of the GMM-Tree algorithm were used in the experiments. For I-niceSO and I-niceMO methods, the maximum number of components for building GMMs and other settings were set according to the I-nice paper [22]. For GMM-P-Forest, the value of the number of GMM-Trees L was set to 5.

The R packages *cluster* [42], *factoextra* [43] and *NbClust* [44] were used to obtain the results of Elbow, Silhouette and Gap Statistic. For Gap Statistic, the number of Monte Carlo bootstrap samples was set to 50. To estimate the number of clusters, the global maximum and the first local maximum method were used in Gap Statistic. All other algorithms were implemented in R (3.2.2). The package mixtools [45] was used for GMM and EM. The experiments were conducted on a machine with 3.30 GHz Intel(R) Core(TM)i5-4590 CPU and 12 GB of memory running Windows 7 of 64-bit.

6.3. Experimental results

In this section, we first discuss the results of comparisons of our proposed algorithms with other methods in finding the number of clusters. Then, we discuss the clustering results of the K-means algorithm based on the measures of purity and Adjusted Random Index (ARI).

6.3.1. Performance comparison on identifying the number of clusters

The first experiment was to compare the performance of the GMM-Tree, GMM-P-Forest and GMM-S-Forest algorithms with Elbow, Gap Statistic, Silhouette, I-niceSO and I-niceMO methods in identifying the numbers of clusters in both synthetic datasets in Table 1 and real datasets in Table 2. The results are shown in Tables 3 and 4, respectively. The column Clusters in Table 3 lists the true number of clusters in each dataset. The column Classes in Table 4 shows the number of classes in each real dataset. We take the number of classes as the true number of clusters in a real dataset. The mark "—" indicates that the number of clusters could not be found in the dataset by the corresponding method.

We can see from Table 3 that Elbow and Gap Statistic performed badly in identifying the numbers of clusters from these synthetic datasets. Only when the number of true clusters in the dataset is small, they were able to estimate a number which is closer to the true number. They failed to estimate the true number when the dataset has more clusters. Silhouette performed a little better but the results are not consistent. Silhouette performed better than Elbow and Gap Statistic.

Generally speaking, I-niceSO and I-niceMO performed better than Silhouette. They identified the numbers of clusters in the datasets which are closer to the true numbers in comparison with the identified numbers by Silhouette. However, their performance deteriorated as the number of features and the number of clusters increased. Compared to I-niceSO, the I-niceMO algorithm performed a little better. However, they both underestimated the number of clusters on the high dimensional datasets with many clusters.

We can see that GMM-Tree, GMM-S-Forest and GMM-P-Forest performed much better than all other methods. On the first 8 low dimensional datasets, GMM-S-Forest and GMM-P-Forest found almost all true numbers of clusters except DS8. GMM-Tree missed some clusters on the datasets DS5 to DS8. Therefore, they are able to identify the true number of clusters in low dimensional datasets with many clusters. GMM-Tree also performed well on the second group of datasets from DS9 to DS16 but tended to underestimate the number of clusters when the datasets have 20 clusters or more, but still the estimated number is close to the true number of clusters. In general, GMM-Tree performed better than I-niceMO.

GMM-S-Forest and GMM-P-Forest performed best in comparison with other methods. On the datasets with 20 clusters or less, both algorithms obtained the true numbers of clusters, even on those high dimensional datasets. As the number of clusters increased, GMM-S-Forest and GMM-P-Forest tended to underestimate the number of clusters in the datasets. However, both GMM-S-Forest and GMM-P-Forest obtained the numbers of clusters closer to the true numbers of clusters on the datasets with more clusters.

Table 4 shows the results of these methods on the real datasets. Elbow failed on almost all datasets. Gap Statistic and Silhouette underestimated the numbers of clusters when the classes

Table 3
Numbers of clusters identified from each synthetic dataset by 8 methods.

Datasets	Clusters	GMM-Tree	GMM-S-Forest	GMM-P-Forest	Elbow	Silhouette	Gap Stat	I-niceSO	I-niceMO
DS1	05	**05**	**05**	**05**	04	**05**	04	**05**	**05**
DS2	10	**10**	**10**	**10**	04	**10**	06	9	**10**
DS3	15	**15**	**15**	**15**	03	17	7	11	14
DS4	20	**20**	**20**	**20**	04	24	8	13	16
DS5	25	24	**25**	**25**	–	29	7	17	19
DS6	30	27	**30**	**30**	4	34	8	22	26
DS7	35	33	**35**	**35**	6	42	5	21	27
DS8	40	37	38	**39**	7	49	9	31	33
DS9	05	**05**	**05**	**05**	04	04	4	**05**	**05**
DS10	10	**10**	**10**	**10**	04	08	7	8	9
DS11	15	14	**15**	**15**	03	12	5	11	12
DS12	20	18	**20**	**20**	–	18	6	14	18
DS13	25	23	**25**	**25**	04	20	4	18	21
DS14	30	28	**30**	**30**	5	27	5	20	23
DS15	35	33	33	**34**	5	37	2	21	26
DS16	40	37	**39**	**39**	6	35	5	28	29
DS17	05	**05**	**05**	**05**	04	**05**	4	4	**5**
DS18	10	09	**10**	**10**	–	13	6	8	9
DS19	15	14	**15**	**15**	–	17	5	9	11
DS20	20	18	**20**	**20**	–	23	5	13	15
DS21	25	23	**26**	27	04	28	4	12	15
DS22	30	27	**29**	28	7	34	11	15	18
DS23	35	32	**34**	33	9	39	2	17	21
DS24	40	36	38	**39**	13	42	3	21	24
DS25	05	**05**	**05**	**05**	04	02	**05**	04	**05**
DS26	10	09	**10**	**10**	04	15	6	9	9
DS27	15	13	**15**	**15**	03	13	5	8	10
DS28	20	16	19	**20**	–	17	5	13	19
DS29	25	21	24	**26**	04	32	–	11	15
DS30	30	26	**29**	**29**	–	34	–	18	22
DS31	35	30	33	33	10	**34**	3	17	19
DS32	40	34	37	**38**	11	48	7	23	28

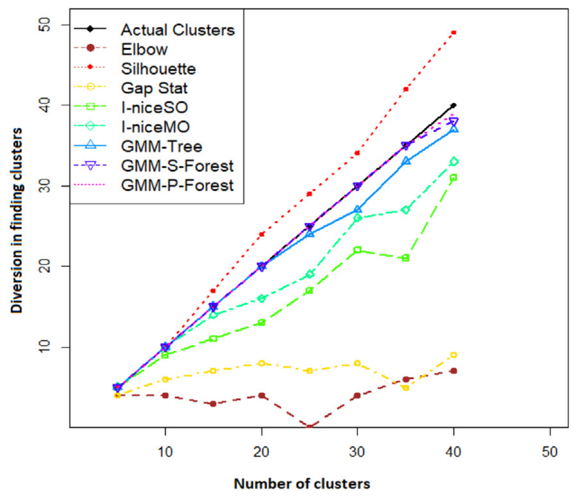
Table 4
Number of clusters identified from each real dataset by the 8 methods.

Datasets	Classes	GMM-Tree	GMM-S-Forest	GMM-P-Forest	Elbow	Silhouette	Gap Stat	I-niceSO	I-niceMO
Appendicitis	2	**2**	**2**	**2**	3	4	**2**	**2**	**2**
Banana	2	4	4	4	5	3	**2**	**2**	**2**
Mammographic	2	**2**	**2**	**2**	3	3	**2**	**2**	**2**
Iris	3	5	4	4	**3**	2	**3**	**3**	**3**
Wine	3	**3**	**3**	**3**	2	**3**	**3**	**3**	**3**
Seed	3	**3**	**3**	**3**	**3**	2	**3**	**3**	**3**
Breast Tissue	6	7	**6**	**6**	3	2	2	**6**	**6**
Satellite	6	**6**	**6**	**6**	3	3	**6**	2	**6**
Glass	6	**6**	**6**	**6**	4	7	2	**6**	**6**
Ecoli	8	**8**	**8**	**8**	3	5	2	2	**8**
Yeast	10	9	**10**	**10**	–	**10**	2	15	**10**
Texture	11	12	**11**	**11**	3	2	4	8	**11**
Libras Mov.	15	**15**	**15**	**15**	4	7	5	**15**	**15**
ISOLET	26	28	28	28	3	2	11	11	14
aloi(short)	100	94	**116**	**116**	4	4	8	–	42

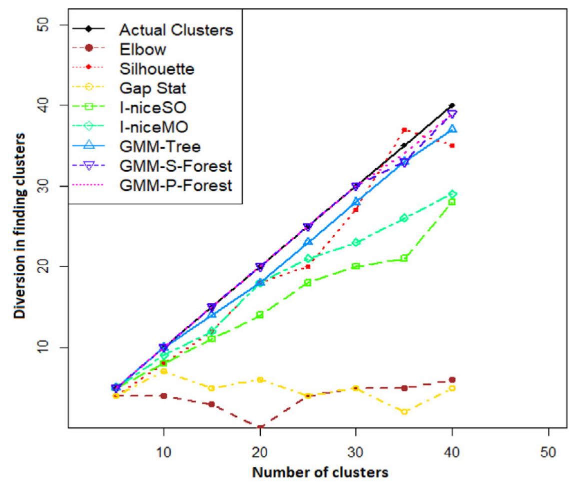
are more than 5, while they performed well when the datasets contain the number of clusters less than 5. I-niceMO performed much better than Gap Statistic, Silhouette and I-niceSO but still underestimated the numbers of clusters on the datasets with large number of clusters. These results indicate that one level GMMs are not able to find all clusters in datasets with many clusters. GMM-Tree improved the results significantly by identifying the numbers of clusters in the datasets which are closers to the true numbers. However, it still produced diverted results from the actual number of clusters. On the contrary, the GMM-S-Forest and GMM-P-Forest algorithms produced overestimated results but the numbers of clusters identified from the datasets are also close to the numbers of classes in the datasets. Unlike the synthetic datasets, some classes in the real datasets may contain more than one cluster. Therefore, the GMM forest algorithms identified the numbers of clusters which are greater than the numbers of classes in the datasets.

Fig. 10 plots the performances of the eight methods on the four groups of synthetic datasets. The horizontal axis is the number of true clusters in the eight datasets in each group. The vertical axis is the number of clusters identified from the eight datasets by the eight methods. Each curve links eight points representing eight pairs of the true numbers and the identified numbers of clusters by one method in the eight datasets. The straight line links the eight dark points representing the eight pairs of the true numbers of clusters. The closer the curve to the straight line, the better the performance of the corresponding method. From these curves, we can see that the curves of both GMM-P-Forest and GMM-S-Forest performed almost equally best and their estimated numbers of clusters are very close to the true numbers of clusters.

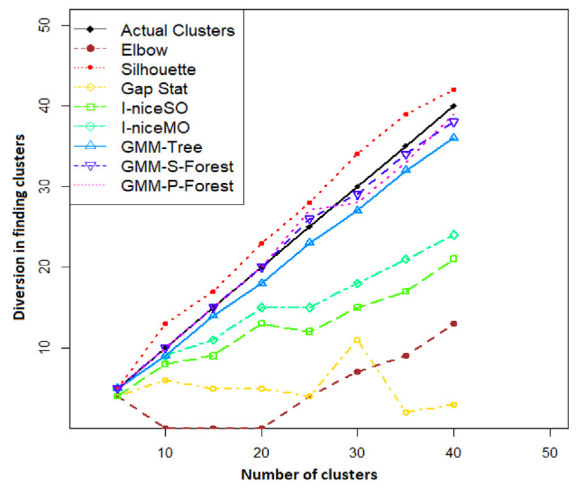
Fig. 11 shows the comparisons of execution times of the five algorithms I-niceSO, I-niceMO, GMM-Tree, GMM-P-Forest and GMM-S-Forest on the four groups of the synthetic datasets. We can see that as the number of clusters in the dataset increased, the execution time of these algorithms increased. However, the



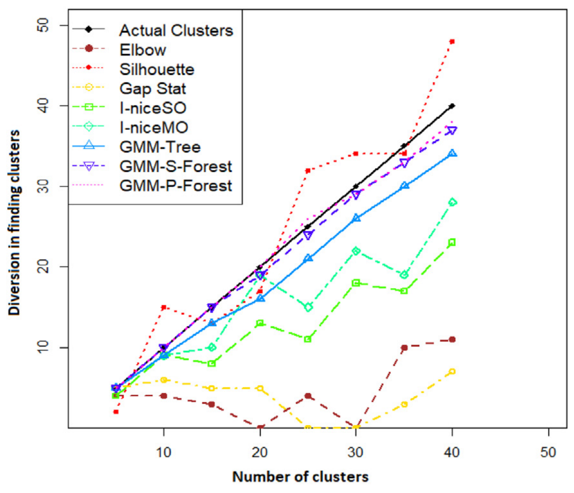
(a) Synthetic Datasets (DS1-DS8) Dimensions = 2 and Objects (500-4000).



(b) Synthetic Datasets (DS9-DS16) Dimensions = 10 and Objects (500-4000).



(c) Synthetic Datasets (DS17-DS24) Dimensions = 20 and Objects (500-4000).



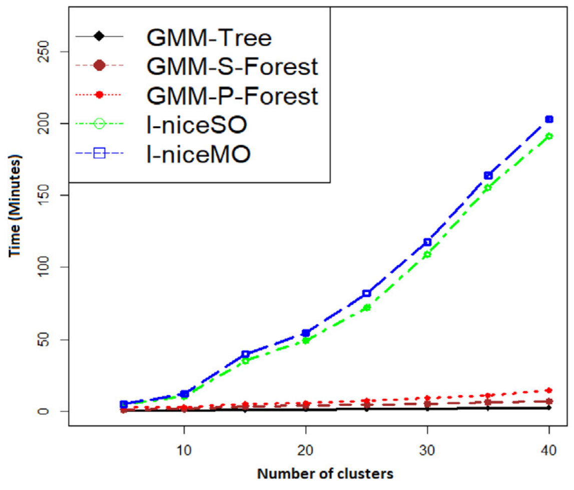
(d) Synthetic Datasets (DS25-DS32) Dimensions = 30 and Objects (500-4000).

Fig. 10. Accuracy comparison in finding the number of clusters (Elbow, Silhouette, Gap Statistic, I-niceSO, and I-niceMO with our 3 proposed algorithms).

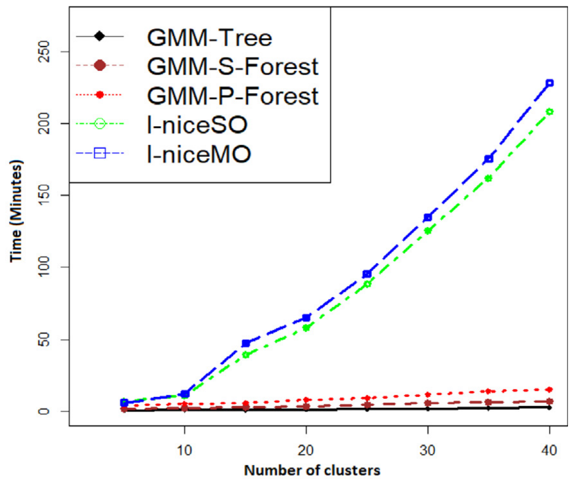
execution time of I-niceSO, I-niceMO increased more dramatically than GMM-Tree, GMM-P-Forest and GMM-S-Forest. The key reason behind this exponential increase of time complexity in I-niceSO and I-niceMO is the parameter *Mmax*, which decides the maximum number of GMM components needs to be generated. In I-niceSO and I-niceMO, Mmax is given as a good guess. If the dataset is considered containing a small number of clusters, Mmax is given a value which is assumed larger than the number of clusters in the dataset. For a dataset with a large number of clusters, Mmax is assigned to an even bigger value. Unfortunately, for any given distance distribution, I-nice starts building a GMM with two components, then three, then four until Mmax. This means that I-nice builds Mmax-1 GMMs and use AIC to select the best fitted one from the *Mmax* -1 GMMs to partition the distance data. Therefore, more clusters in the data set, more GMMs are built, and more running time is required.

However, in GMM-Tree algorithm, the number of high density peaks, τ^*, is calculated using the distance density function Eq. (5). The best fitted GMM is assumed to be in the range of number of components of $[\tau^*-2, \tau^*+2]$. Therefore, only 5 GMMs are computed from each distance distribution no matter how many clusters are contained in the dataset. As such, the running times for the proposed methods do not change much as the number of clusters increase. Since less GMMs are computed in the proposed methods, their running time is much faster than I-niceSO and I-niceMO algorithms. Since the proposed algorithms have heuristic steps, it is not tractable for formal complexity analysis. Therefore, we give experiment results in Fig. 11 to show the execution performance properties.

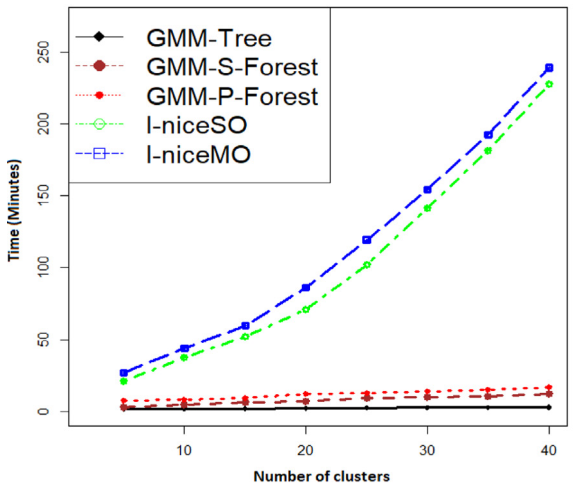
In GMM-S-Forest, there is no need to set the number of GMM trees to be generated. It automatically explores when to stop the generation of a new GMM tree. GMM-P-Forest may take more time as compared to the GMM-Tree and GMM-S-Forest algorithms on a single machine because each GMM tree has to be built sequentially. In the case of running on a cluster with many nodes, GMM-P-Forest performs better as compared to GMM-S-Forest in execution time.



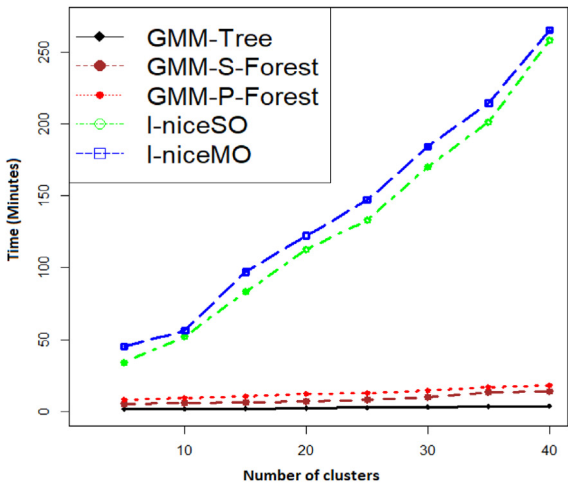
(a) Synthetic Datasets (DS1-DS8) Dimensions = 2 and Objects (500-4000).



(b) Synthetic Datasets (DS9-DS16) Dimensions = 10 and Objects (500-4000).



(c) Synthetic Datasets (DS17-DS24) Dimensions = 20 and Objects (500-4000).



(d) Synthetic Datasets (DS25-DS32) Dimensions = 30 and Objects (500-4000).

Fig. 11. Processing time comparison (GMM-Tree, GMM-S-Forest, GMM-P-Forest, I-niceSO, I-niceMO).

6.3.2. *Performance on clustering*

The second experiment was to compare the performance on clustering both synthetic and real datasets by the K-means algorithm using the number of clusters and the initial cluster centers as inputs. In this experiment, three types of the initial cluster centers were used as the inputs to the K-means algorithm. They are

1. **True cluster centers-** A true center was computed as the mean vector of the objects in each cluster in the dataset. For a real dataset, a true center was calculated as the mean vector of the objects in the same class in the dataset.
2. **Identified initial cluster centers-** The initial cluster centers were identified by Silhouette, Gap Statistic, I-niceSO, I-niceMO, GMM-Tree, GMM-S-Forest and GMM-P-Forest.
3. **Random initial cluster centers-** The initial cluster centers were randomly selected from a dataset. The number of clusters was equal to the number of true clusters in a synthetic dataset or the number of classes in a real dataset.

The clustering performance is measured by Purity [46] which is defined as

$$Purity = \frac{1}{N} \sum_{i=1}^{K} \underbrace{max}_{1 \le j \le Q} |C_i \cap Y_j| \tag{7}$$

where K is the number of true clusters, Q is the number of clusters by K-means, C_i is the set of objects in the true cluster i and Y_j is the set of objects in the cluster j by K-means. $\|$ indicates the number of objects in the intersection of C_i and Y_j, and N is the total number of objects in the dataset. The value of Purity is between 0 to 1.

The Adjusted Rand Index (ARI) [47] is also used to measure the clustering performance. It is a measure of agreement between two partitions: one given by the clustering process and the other defined by the ground truth. Given a data set S of N objects, and two partitions of the N objects, namely $C = C_1, C_2, \dots, C_K$, the partition of N objects into K clusters, and $Y = Y_1, Y_2, \dots, Y_P$, the partition of N objects into P classes (the ground truth), the ARI is

Table 5
Comparison of the clustering results of our proposed techniques with other state-of-the-art methods on the synthetic datasets based on purity.

Datasets	Clusters	True	GMM-Tree	GMM-S-Forest	GMM-P-Forest	Random	Silhouette	Gap Stat	I-niceSO	I-niceMO
DS1	05	0.999	0.985	0.988	**0.989**	0.865	0.885	0.693	0.891	0.921
DS2	10	1.000	0.989	**0.992**	0.991	0.879	0.899	0.684	0.908	0.913
DS3	15	0.998	0.981	0.983	**0.984**	0.851	0.871	0.675	0.882	0.891
DS4	20	1.000	0.982	0.985	**0.986**	0.872	0.892	0.682	0.899	0.909
DS5	25	0.997	0.979	0.981	**0.982**	0.869	0.889	0.576	0.881	0.889
DS6	30	0.999	0.981	0.983	0.984	0.871	**0.986**	0.417	0.891	0.897
DS7	35	0.993	0.967	0.969	**0.971**	0.753	0.884	0.331	0.872	0.976
DS8	40	1.000	0.975	**0.979**	0.978	0.801	0.900	0.392	0.881	0.889
DS9	05	0.998	0.982	0.985	**0.986**	0.882	0.893	0.682	0.889	0.895
DS10	10	1.000	0.987	**0.989**	**0.989**	0.889	0.898	0.689	0.899	0.903
DS11	15	0.997	0.976	**0.979**	0.978	0.866	0.875	0.666	0.878	0.883
DS12	20	1.000	0.978	0.981	**0.982**	0.882	0.893	0.682	0.894	0.898
DS13	25	0.996	0.972	**0.976**	0.975	0.872	0.884	0.472	0.881	0.887
DS14	30	1.000	0.971	0.975	**0.977**	0.797	0.867	0.167	0.882	0.892
DS15	35	0.995	0.873	**0.879**	0.878	0.753	0.865	0.057	0.869	0.872
DS16	40	1.000	0.963	0.965	**0.967**	0.750	0.800	0.125	0.873	0.879
DS17	05	0.998	0.979	0.982	**0.983**	0.879	0.899	0.679	0.881	0.887
DS18	10	1.000	0.982	**0.986**	0.985	0.872	0.892	0.672	0.892	0.899
DS19	15	0.997	0.972	0.977	**0.978**	0.872	0.892	0.672	0.871	0.878
DS20	20	1.000	0.972	0.975	**0.976**	0.812	0.832	0.612	0.891	0.894
DS21	25	0.995	0.922	0.937	**0.941**	0.822	0.842	0.422	0.869	0.873
DS22	30	1.000	0.971	0.975	**0.979**	0.733	0.867	0.367	0.871	0.882
DS23	35	0.997	0.854	0.873	**0.874**	0.724	0.840	0.057	0.861	0.893
DS24	40	1.000	0.961	**0.968**	0.968	0.710	0.825	0.075	0.871	0.891
DS25	05	0.998	0.971	0.977	**0.979**	0.871	0.881	0.671	0.879	0.883
DS26	10	1.000	0.980	0.984	**0.986**	0.883	0.893	0.683	0.890	0.893
DS27	15	0.998	0.967	**0.969**	0.967	0.867	0.877	0.667	0.865	0.871
DS28	20	1.000	0.969	**0.978**	0.977	0.869	0.879	0.599	0.883	0.889
DS29	25	0.997	0.920	0.928	**0.929**	0.820	0.830	0.320	0.860	0.869
DS30	30	1.000	0.967	**0.972**	0.971	0.733	0.859	–	0.867	0.872
DS31	35	0.996	0.822	0.836	0.838	0.760	**0.841**	0.083	0.811	0.872
DS32	40	1.000	0.954	0.958	**0.959**	0.700	0.810	0.175	0.812	0.834

Table 6
Comparison of the clustering results of our proposed techniques with other state-of-the-art methods on the synthetic datasets based on ARI.

Datasets	Clusters	True	GMM-Tree	GMM-S-Forest	GMM-P-Forest	Random	Silhouette	Gap Stat	I-niceSO	I-niceMO
DS1	05	0.999	0.995	**0.996**	0.995	0.895	0.981	0.864	0.984	0.992
DS2	10	1.000	0.996	**0.999**	0.997	0.910	0.995	0.873	0.985	0.994
DS3	15	0.994	**0.998**	0.997	0.998	0.898	0.982	0.761	0.982	0.991
DS4	20	1.000	0.995	0.997	**0.998**	0.902	0.978	0.864	0.983	0.992
DS5	25	0.989	0.973	0.974	**0.975**	0.871	0.953	0.735	0.962	0.971
DS6	30	1.000	0.983	0.985	**0.989**	0.840	0.926	0.311	0.973	0.981
DS7	35	0.987	0.951	0.962	**0.965**	0.832	0.952	0.242	0.943	0.951
DS8	40	1.000	0.985	**0.989**	0.988	0.879	0.838	0.344	0.965	0.974
DS9	05	0.993	0.989	0.991	**0.992**	0.889	0.969	0.843	0.972	0.981
DS10	10	1.000	0.992	0.995	**0.997**	0.893	0.989	0.855	0.975	0.984
DS11	15	0.990	0.905	0.909	**0.915**	0.816	0.895	0.667	0.894	0.902
DS12	20	1.000	0.991	0.995	**0.996**	0.823	0.905	0.676	0.875	0.894
DS13	25	0.989	0.868	**0.888**	0.878	0.728	0.858	0.524	0.849	0.858
DS14	30	1.000	0.987	0.989	**0.991**	0.841	0.868	0.183	0.876	0.885
DS15	35	0.988	0.856	**0.866**	0.864	0.823	0.942	0.053	0.828	0.837
DS16	40	1.000	0.983	**0.989**	0.988	0.772	0.788	0.144	0.845	0.854
DS17	05	0.989	0.986	0.987	0.989	0.892	0.975	0.632	0.972	**0.990**
DS18	10	1.000	0.991	**0.993**	0.992	0.893	0.983	0.551	0.979	0.988
DS19	15	0.986	0.896	0.897	**0.899**	0.794	0.884	0.656	0.877	0.886
DS20	20	1.000	0.988	**0.991**	0.989	0.895	0.978	0.735	0.964	0.982
DS21	25	0.952	0.836	**0.841**	0.840	0.744	0.827	0.609	0.807	0.816
DS22	30	1.000	0.872	0.901	0.901	0.782	**0.921**	0.251	0.864	0.873
DS23	35	0.984	0.789	0.793	**0.799**	0.788	0.889	0.052	0.754	0.762
DS24	40	1.000	0.968	**0.969**	0.968	0.709	0.899	0.058	0.703	0.711
DS25	05	0.991	0.941	**0.948**	0.946	0.863	0.933	0.502	0.913	0.922
DS26	10	1.000	0.959	**0.964**	0.962	0.877	0.946	0.515	0.922	0.941
DS27	15	0.989	0.883	**0.887**	0.886	0.795	0.874	0.445	0.866	0.875
DS28	20	1.000	0.943	0.949	**0.947**	0.865	0.935	0.513	0.916	0.924
DS29	25	0.985	0.881	0.887	**0.886**	0.794	0.873	0.442	0.814	0.822
DS30	30	1.000	0.937	**0.943**	0.942	0.787	0.935	0.215	0.844	0.853
DS31	35	0.982	0.846	0.852	0.852	0.863	**0.903**	0.049	0.702	0.711
DS32	40	1.000	0.926	0.931	**0.932**	0.693	0.902	0.0921	0.803	0.812

Table 7
Comparison of the clustering results of our proposed techniques with other state-of-the-art methods on the real datasets based on purity.

Datasets	Classes	True	GMM-Tree	GMM-S-Forest	GMM-P-Forest	Random	Silhouette	Gap Stat	I-niceSO	I-niceMO
Appendicitis	2	0.811	0.830	0.833	0.833	0.802	**0.887**	0.802	0.830	0.830
Banana	2	0.567	0.585	**0.588**	**0.588**	0.567	0.553	0.567	0.566	0.566
Mammographic	2	0.790	0.786	**0.788**	**0.788**	0.740	0.780	0.537	0.685	0.685
Iris	3	0.833	0.887	0.887	0.887	0.667	0.667	0.667	**0.893**	**0.893**
Wine	3	0.966	0.949	0.959	0.959	0.635	0.966	0.966	**0.702**	**0.702**
Seed	3	0.919	0.890	0.901	0.897	0.929	0.657	**0.919**	0.895	0.895
Breast	6	0.585	**0.594**	**0.594**	**0.594**	0.574	0.396	0.396	0.433	0.452
Satellite	6	0.741	0.742	0.741	**0.742**	0.741	0.517	0.741	–	0.737
Glass	6	0.551	0.533	0.535	0.535	0.533	0.547	0.369	0.570	**0.588**
Ecoli	8	0.679	0.676	0.678	0.678	0.667	**0.679**	0.586	–	0.630
Yeast	10	0.531	0.518	0.518	0.518	0.549	0.533	0.057	**0.557**	0.516
Texture	11	0.613	0.638	**0.645**	**0.645**	0.584	0.182	0.324	–	0.642
Lib. Mov.	15	0.486	0.461	**0.489**	**0.489**	0.417	0.333	0.244	0.438	0.488
ISOLET	26	0.636	0.580	**0.582**	**0.582**	0.577	0.077	0.349	0.307	0.272
Aloi(short)	100	0.769	0.653	**0.723**	**0.723**	0.700	0.040	0.080	–	0.490

Table 8
Comparison of the clustering results of our proposed techniques with other state-of-the-art methods on the real datasets based on ARI.

Datasets	Classes	True	GMM-Tree	GMM-S-Forest	GMM-P-Forest	Random	Silhouette	Gap Stat	I-niceSO	I-niceMO
Appendicitis	2	0.335	0.378	**0.379**	**0.379**	0.230	0.195	0.230	0.378	0.378
Banana	2	0.018	**0.029**	**0.029**	**0.029**	0.018	0.006	0.018	0.017	0.017
Mammographic	2	0.335	0.326	**0.328**	**0.328**	0.315	0.284	0.004	0.136	0.136
Iris	3	0.620	0.716	0.726	0.726	0.457	0.568	0.568	**0.730**	**0.730**
Wine	3	0.897	0.847	0.847	0.847	0.381	**0.897**	0.892	0.371	0.371
Seed	3	0.773	0.705	0.705	0.705	0.687	0.481	**0.773**	0.716	0.716
Breast	6	0.317	0.266	**0.269**	0.267	0.236	0.197	0.197	0.186	0.175
Satellite	6	0.529	0.526	0.526	0.526	0.529	0.294	0.528	–	**0.529**
Glass	6	0.160	0.204	0.206	0.206	0.122	0.169	0.005	**0.261**	0.255
Ecoli	8	0.479	0.256	0.256	0.256	0.379	**0.495**	0.271	–	0.426
Yeast	10	0.187	0.188	0.188	**0.189**	0.166	0.182	0.124	0.112	0.145
Texture	11	0.466	0.443	**0.466**	0.465	0.394	0.111	0.228	–	0.408
Lib. Mov.	15	0.292	0.296	0.296	0.296	0.287	0.210	0.149	0.265	**0.317**
ISOLET	26	0.543	0.454	**0.486**	0.485	0.449	0.060	0.350	0.135	0.137
Aloi(short)	100	0.639	0.527	**0.569**	**0.569**	0.547	0.048	0.087	–	0.391

defined as

$$ARI = \frac{\sum_{ij}\binom{N_{ij}}{2} - \left[\sum_i \binom{\alpha_i}{2} \sum_j \binom{\beta_j}{2}\right] / \binom{N}{2}}{1/2\left[\sum_i \binom{\alpha_i}{2} \sum_j \binom{\beta_j}{2}\right] - \left[\sum_i \binom{\alpha_i}{2} \sum_j \binom{\beta_j}{2}\right] / \binom{N}{2}} \tag{8}$$

where N_{ij} is the number of objects in both cluster C_i and class Y_j, α_i is the number of objects in cluster C_i and β_j is the number of objects in class Y_j.

Table 5 shows the clustering results of the synthetic datasets measured by Purity, while Table 6 shows the clustering results of the synthetic datasets measured by ARI. The column *TrueCenters* shows the clustering results by K-means using the true cluster centers as the initial cluster centers. Columns Random, Silhouette, Gap Stat, GMM-Tree, GMM-S-Forest, GMM-P-Forest, I-niceSO and I-niceMO are the clustering results by K-means using the initial cluster centers identified by the corresponding algorithms as inputs. We can see clearly that the results with the initial clusters from GMM-S-Forest and GMM-P-Forest are very close to the results with the true cluster centers in most of the datasets. GMM-Tree, I-niceSO and I-niceMO also performed good on the first two groups of the low dimensional datasets and the last two groups of the high dimensional datasets with less clusters. However, their performance was dropped a little on the high dimensional datasets with more clusters.

The column Random shows the results by K-means using randomly selected initial cluster centers. We can see the performance of K-means clustering deteriorated badly. The columns Silhouette and Gap Statistic show the results produced by the Silhouette method and the Gap Statistic method. These results are not as good as the results by our proposed methods, i.e., GMM-Tree, GMM-S-Forest and GMM-P-Forest.

Tables 7 and 8 show the clustering results of the real datasets measured by purity and ARI, respectively. From the results of the column *TrueCenters*, we can see that these classes are quite separated although they contain many clusters. Similar to the results of synthetic datasets, the results of GMM-S-Forest and GMM-P-Forest are very close to the results of true clusters. The results of column Random are the worst. For datasets Letters and Alot(short), Silhouette and Gap Statistic performed worse than other methods except for Random. I-niceSO and I-niceMO all performed well on these real datasets but a little worse than GMM-S-Forest and GMM-P-Forest, which are superior to all other methods in general.

We remark that the Purity performance measure is not same as the measure of classification accuracy if there is no one to one correspondence between the classes and the identified clusters. If the number of clusters is underestimated, some identified cluster can contain more than class. In this case, the accuracy is reduced because two classes are not separated by one cluster and one class is miss-classified. If the number of clusters is overestimated, some class may be clustered into more than one cluster. In this case, if the clusters are labeled with the same class, the classification accuracy can be higher than the underestimated case. In any case, a high purity measure indicates a better partition of classes in the clustering result.

7. Conclusions and future work

In this paper, we have presented a new GMM-Tree algorithm that uses a distance distribution of objects with respect to an observation point and a Gamma Mixture Model to hierarchically partition a high dimensional dataset to build a GMM tree and use the leaf nodes of the tree as candidate clusters to estimate the number of clusters in the dataset and find the initial cluster centers. The estimated number of clusters and the initial cluster centers can be used as inputs to other clustering algorithms such as K-means. The new algorithm is able to estimate the true number of clusters in datasets with many clusters. For datasets with more clusters, the GMM-S-Forest and GMM-P-Forest algorithms are proposed. Compared with the popular methods such as Elbow, Silhouette and Gap Statistic, the new algorithms can identify more clusters in high dimensional datasets. Because default values are used, these new algorithms are easy to use in practice and efficient in dealing with large datasets.

As in case of very high dimensional data, the data becomes sparse and it becomes difficult to find high density regions. GMM-Tree does not perform well if there is no high dense region in the data. Thus, the future directions of this work will be on building GMM trees in the subspaces of high dimensional data and classification GMM trees from labeled datasets.

Declaration of competing interest

No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.asoc.2019.105891.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (NSFC) Grant No. 61473194 and Shenzhen - Hong Kong Technology Cooperation Fund, China (SGLH20161209101100926).

References

[1] C. Meyer, S. Race, K. Valakuzhy, Determining the number of clusters via iterative consensus clustering, in: Proceedings of the 2013 SIAM International Conference on Data Mining, 2013, pp. 94-102.
[2] Y. Ye, J.Z. Huang, X. Chen, S. Zhou, G. Williams, X. Xu, Neighborhood density method for selecting initial cluster centers in k-means clustering, in: Pacific-Asia Conference on Knowledge Discovery and Data Mining, 2006, pp. 189–198.
[3] C. Hennig, T.F. Liao, How to find an appropriate clustering for mixed-type variables with application to socio-economic stratification, J. R. Stat. Soc. Ser. C. Appl. Stat. 62 (3) (2013) 309–369.
[4] P.S. Bradley, U.M. Fayyad, Refining Initial Points for K-Means Clustering, in: Proceedings of the Fifteenth International Conference on Machine Learning (Vol. 98), 1998, pp. 91-99.
[5] S.S. Khan, A. Ahmad, Cluster center initialization algorithm for K-means clustering, Pattern Recognit. Lett. 25 (11) (2004) 1293–1302.
[6] S. Deelers, S. Auwatanamongkol, Enhancing k-means algorithm with initial cluster centers derived from data partitioning along the data axis with the highest variance, Int. J. Comput. Sci. 2 (4) (2007) 247–252.
[7] M.P.S. Bhatia, D. Khurana, Analysis of initial centers for k-means clustering algorithm, Int. J. Comput. Appl. 71 (5) (2013).
[8] G. Vegas-Sanchez-Ferrero, M. Martin-Fernandez, J.M. Sanches, A gamma mixture model for IVUS imaging, in: Multi-modality Atherosclerosis Imaging and Diagnosis, 2014, pp. 155–171.
[9] P.J. Rousseeuw, Silhouettes: a graphical aid to the interpretation and validation of cluster analysis, J. Comput. Appl. Math. 20 (1987) 53–65.
[10] R.L. Thorndike, Who belongs in the family? Psychometrika 18 (4) (1953) 267–276.
[11] R. Tibshirani, G. Walther, T. Hastie, Estimating the number of clusters in a data set via the gap statistic, J. R. Stat. Soc. Ser. B Stat. Methodol. 63 (2) (2001) 411–423.
[12] C.W. Tsai, W.L. Chen, M.C. Chiang, A modified multiobjective EA-based clustering algorithm with automatic determination of the number of clusters, in: 2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2012, pp. 2833–2838.
[13] L. Wang, C. Leckie, K. Ramamohanarao, J. Bezdek, Automatically determining the number of clusters in unlabeled data sets, IEEE Trans. Knowl. Data Eng. 21 (3) (2009) 335–350.
[14] R.C. de Amorim, C. Hennig, Recovering the number of clusters in data sets with noise features using feature rescaling factors, Inform. Sci. 324 (2015) 126–145.
[15] R. Jain, A. Koronios, Innovation in the cluster validating techniques, Fuzzy Optim. Decis. Mak. 7 (3) (2008) 233–242.
[16] Y. Zhong, S. Zhang, L. Zhang, Automatic fuzzy clustering based on adaptive multi-objective differential evolution for remote sensing imagery, IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. 6 (5) (2013) 2290–2301.
[17] Y. Zhong, A. Ma, Y. soon Ong, Z. Zhu, L. Zhang, Computational intelligence in optical remote sensing image processing, Appl. Soft Comput. 64 (2018) 75–93.
[18] A. Rodriguez, A. Laio, Clustering by fast search and find of density peaks, Science 344 (6191) (2014) 1492–1496.
[19] B.J. Frey, D. Dueck, Clustering by passing messages between data points, Science 315 (5814) (2007) 972–976.
[20] S. Ding, M. Du, T. Sun, X. Xu, Y. Xue, An entropy-based density peaks clustering algorithm for mixed type data employing fuzzy neighborhood, Knowl.-Based Syst. 133 (2017) 294–313.
[21] X. Xu, S. Ding, M. Du, Y. Xue, DPCG: an efficient density peaks clustering algorithm based on grid, Int. J. Mach. Learn. Cybern. (2018) 1–12.
[22] M.A. Masud, J.Z. Huang, C. Wei, J. Wang, I. Khan, M. Zhong, I-nice: A new approach for identifying the number of clusters and initial cluster centres, Inform. Sci. 466 (2018) 129–151.
[23] J.D. Banfield, A.E. Raftery, Model-based Gaussian and non-Gaussian clustering, Biometrics 80 (1993) 3–821.
[24] X. Hu, L. Xu, Automatic cluster number determination via BYY harmony learning, in: International Symposium on Neural Networks, 2004, pp. 828–833.
[25] J. Shen, S.I. Chang, E.S. Lee, Y. Deng, S.J. Brown, Determination of cluster number in clustering microarray data, Appl. Math. Comput. 169 (2) (2005) 1172–1185.
[26] O. Arbelaitz, I. Gurrutxaga, J. Muguerza, J.M. Perez, I. Perona, An extensive comparative study of cluster validity indices, Pattern Recognit. 46 (1) (2013) 243–256.
[27] J.A. Aslam, R.A. Popa, R.L. Rivest, On Estimating the Size and Confidence of a Statistical Audit, in: Proceedings of the USENIX Workshop on Accurate Electronic Voting Technology, 7, 2007, p. 8.
[28] T. Chiu, D. Fang, J. Chen, Y. Wang, C. Jeris, A robust and scalable clustering algorithm for mixed type attributes in large database environment, in: Proceedings of the seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2001, pp. 263-268.
[29] M. Ester, H.P. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, in: Knowledge Discovery in Databases, vol. 96, 1996, pp. 226–231, No. 34.
[30] T. Beier, F.A. Hamprecht, J.H. Kappes, Fusion moves for correlation clustering, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 3507-3516.
[31] Y. Wang, J. Zhu, DP-space: Bayesian nonparametric subspace clustering with small-variance asymptotics, in: International Conference on Machine Learning, 2015, pp. 862–870.
[32] A.P. Dempster, N.M. Laird, D.B. Rubin, Maximum likelihood from incomplete data via the EM algorithm, J. R. Stat. Soc. Ser. B Stat. Methodol. 39 (1) (1977) 1–22.
[33] N. Sugiura, Further analysts of the data by akaike's information criterion and the finite corrections: Further analysts of the data by akaike's, Comm. Statist. Theory Methods 7 (1) (1978) 13–26.
[34] C.M. Hurvich, C.L. Tsai, Regression and time series model selection in small samples, Biometrika 76 (2) (1989) 297–307.
[35] G.H. John, P. Langley, Estimating continuous distributions in Bayesian classifiers, in: Proceedings of the Eleventh conference on Uncertainty in Artificial Intelligence, 1995, pp. 338-345.
[36] B. Desgraupes, clusterCrit: clustering indices, R package version, 1(3), 2013, pp. 4-5.
[37] D. Arthur, S. Vassilvitskii, k-means++: The advantages of careful seeding, in: Proceedings of The Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, 2007, pp. 1027-1035.
[38] M.S. Yang, K.L. Wu, A modified mountain clustering algorithm, Pattern Anal. Appl. 8 (1–2) (2005) 125–138.
[39] W. Qiu, H. Joe, clusterGeneration: random cluster generation (with specified degree of separation), R package version, 1(7), 2009, pp. 75275-0122.
[40] C.L. Blake, C.J. Merz, UCI Repository of Machine Learning Databases, Department of Information and Computer Sciences, University of California, Irvine, 1998, Also available at: http://www.ics.uci.edu/mlearn/MLRepository.html.
[41] J.M. Geusebroek, G.J. Burghouts, A.W. Smeulders, The amsterdam library of object images, Int. J. Comput. Vis. 61 (1) (2005) 103–112.
[42] L. Kaufman, P.J. Rousseeuw, Finding Groups in Data: An Introduction to Cluster Analysis (Vol. 344), John Wiley and Sons, 2009.
[43] A. Kassambara, F. Mundt, Factoextra: extract and visualize the results of multivariate data analyses, R package version, 1(3), 2016, p. 2016.

[44] M. Charrad, N. Ghazzali, V. Boiteau, A. Niknafs, NbClust Package: finding the relevant number of clusters in a dataset, J. Stat. Softw. (2012).
[45] T. Benaglia, D. Chauveau, D. Hunter, D. Young, Mixtools: An R package for analyzing finite mixture models, J. Stat. Softw. 32 (6) (2009) 1–29.
[46] C. Manning, P. Raghavan, H. Schutze, Introduction to information retrieval, Nat. Lang. Eng. 16 (1) (2010) 100–103.
[47] L. Hubert, P. Arabie, Comparing partitions, J. Classification 2 (1) (1985) 193–218.