Information Sciences 178 (2008) 4205–4227

Contents lists available at ScienceDirect

Information Sciences

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

# GAKREM: A novel hybrid clustering algorithm

Cao D. Nguyen [a], Krzysztof J. Cios [a,b,*]

[a] Virginia Commonwealth University, Computer Science, 401 W. Main Street, Richmond, VA 23238, USA
[b] IITiS Polish Academy of Sciences, Poland

ARTICLE INFO

*Article history:*
Received 18 June 2007
Received in revised form 21 May 2008
Accepted 6 July 2008

*Keywords:*
Clustering
EM
*K*-means
Genetic algorithms
Logarithmic regression
GAKREM
Likelihood cross-validation

ABSTRACT

We introduce a novel clustering algorithm named GAKREM (Genetic Algorithm *K*-means Logarithmic Regression Expectation Maximization) that combines the best characteristics of the *K*-means and EM algorithms but avoids their weaknesses such as the need to specify a priori the number of clusters, termination in local optima, and lengthy computations. To achieve these goals, genetic algorithms for estimating parameters and initializing starting points for the EM are used first. Second, the log-likelihood of each configuration of parameters and the number of clusters resulting from the EM is used as the fitness value for each chromosome in the population. The novelty of GAKREM is that in each evolving generation it efficiently approximates the log-likelihood for each chromosome using logarithmic regression instead of running the conventional EM algorithm until its convergence. Another novelty is the use of *K*-means to initially assign data points to clusters. The algorithm is evaluated by comparing its performance with the conventional EM algorithm, the *K*-means algorithm, and the likelihood cross-validation technique on several datasets.

Published by Elsevier Inc.

## 1. Introduction

Clustering means identifying groups of objects that are similar to each other (based on some similarity measure) but dissimilar to objects belonging to other groups. Clustering is the most important and, at the same time, the most challenging of any data mining methods. The reasons are that most of huge amounts of data being collected in any given domain are unlabelled/unsupervised, meaning that there are no known inputs corresponding to the known outputs. In such situations, the only approach is to use either clustering or association rule algorithms. If the result of clustering is successful, namely, an algorithm is able to reliably find true natural groupings in the data then the problem of analyzing such data becomes much easier because domain experts could possibly label the found clusters. Then a host of supervised learning algorithms can be used to further analyze the data. Unfortunately, assessing reliability/correctness of clustering is an open research question. First, the user, for majority of clustering algorithms, must guess the number of clusters to be generated as an input parameter. If no sufficient knowledge about the domain/data exist making such guess is difficult. To address this problem, several measures for evaluating the correctness of the found clustering structure have been developed. They are called cluster validity measures and are used as follows: generate several clusters in the data and then calculate cluster validity measure and assume that its optimum value indicates the (possibly) correct grouping. However, majority of cluster validity measures suffer from the monotonicity property, namely, the more clusters are generated the better the performance index becomes. Second, clustering algorithms, such as *K*-means or EM, perform a local search where the solution obtained at the next iteration is in the vicinity of the current solution. In other words, they are very sensitive to initialization parameters and may suffer from over-fitting (too many clusters generated) while not reflecting the true underlying structure [9,16,17,22].

* Corresponding author. Address: Virginia Commonwealth University, Computer Science, 401 W. Main Street, Richmond, VA 23238, USA.
*E-mail address:* kcios@vcu.edu (K.J. Cios).

0020-0255/$ - see front matter Published by Elsevier Inc.
doi:10.1016/j.ins.2008.07.016

There are several well-known algorithms such as statistical clustering [9,33,34], hierarchical clustering [10,16,31], partitional clustering [22,35], nearest neighbor clustering [19], fuzzy C-means clustering [3,26], and neural network clustering [18]. Among them, the *K*-means is the most popular because of its simplicity and computational efficiency although it is very sensitive to the initial choice of medoids [17,22]. The much more complex EM algorithm is a statistical technique for finding maximum likelihood estimate of the parameters of an underlying distribution from an incomplete data [9,33,34]. EM has been used for parameter finding of normal mixture models [21] because of a number of its attractive properties. Namely, it monotonically converges without the need to specify a learning rate, automatically satisfies probabilistic constraints [34], has low cost per iteration [24], and is easy to implement [4]. Unfortunately, these advantages are mitigated by two serious problems: (a) it is an iterative algorithm that starts from a guessed set of parameters that govern the underlying distribution and is thus very sensitive to initialization (in other words it may converge to a local maxima of the log-likelihood function), and (b) it cannot determine the number of clusters because of overfitting.

To overcome the above described drawbacks we introduce here a novel algorithm which combines *K*-means and the EM algorithms with genetic algorithms (GAs) and logarithmic regression, into a powerful hybrid method called GAKREM (genetic algorithm *K*-means logarithmic regression expectation maximization). In particular, we use GAs and *K*-means to estimate the initial guess of parameters for the conventional EM. First, each chromosome in a population of GAs encodes a number of randomly set number of clusters and their medoids. Second, given each chromosome as input, the *K*-means algorithm computes the first guess of parameters for the EM algorithm. Third, the log-likelihood functions are calculated in few steps of the EM algorithm and are used to predict the expected log-likelihood value using logarithmic regression. Fourth, fitness values of the chromosomes are evaluated by a combination of the expected log-likelihood function and the number of clusters encoded in the chromosomes. We use GAs to thoroughly explore the entire search space so that GAKREM might find global optimum.

Several GA representations have been explored for clustering. For example, CLUSTERING [30] makes use of the nearest neighbor clustering method to group those data that are close to one another in the first stage, and the GA to group small clusters into larger clusters in its second stage. To operate, however, CLUSTERING needs the weight *w*, a trade-off parameter of inter-distance between clusters and intra-distance within a cluster, which is not easy to decide. COWCLUS algorithm [7] uses GA to maximize a variance-ratio based fitness criterion defined in terms of external cluster isolation and internal cluster homogeneity. The limitation of COWCLUS is that it requires setting the *k* parameter a priori. Tan and Taniar [29] proposed an adaptive maximum-entropy estimated distribution, the MEED clustering, to reduce the constraint dimension in model learning. The probability function provides mapping between the attributes of *k* medoids and the fitness classes. However, the MEED also suffers from the same drawback as COWCLUS because it requires the number of clusters to be specified as its input. Other GA-based clustering approaches [2,13,15] use the same scenarios: the length of each chromosome is equal to the number of data points and the fitness functions are specified as objective functions (e.g., Davies–Bouldin index [2], Silhouette index [15], and nearest neighbor [13]). Despite the fact that these approaches do not require the number of clusters specified as input, their disadvantages are very high computational cost and that the results depend heavily on initial population.

The remainder of the article is organized as follows. In Section 2, we briefly review the EM and *K*-means theory in normal mixtures and introduce our approach. The next section provides results and comparison of performance between GAKREM and the *K*-means, EM, and the likelihood cross-validation technique on several datasets.

## 2. Methods

### 2.1. Definitions

The following notation is used in the paper.

- $X = \{x_1, x_2, \ldots, x_N\}$ is an observed (or incomplete) data consisting of $N$ data points. Each data point $x_i$ is a $d$-dimensional vector. Assume that $X$ is independently and identically distributed with an underlying distribution $p$ governed by a parameter $\Theta^{(t)}$.
- $Y = \{y_1, y_2, \ldots, y_N\}$ is an unobserved (or missing) dataset. Each unobserved data point $y_i$ ($i = 1, \ldots, N$) is associated with an observed data point $x_i$ to indicate which cluster generated the data point $x_i$. In a finite mixture model, technically, $y_i = \{y_i^1, \ldots, y_i^k\}$ is a $k$-dimensional binary vector corresponding $k$ clusters, i.e. if cluster $h$ ($h = 1, \ldots, k$) produced the data point $x_i$ then $y_i^h = 1$ and $y_i^j = 0$ with $j \neq h$.
- $\Theta$ denotes parameters of the density to be estimated. In normal mixtures, $\Theta$ contains two parts: $\Theta \equiv \{\alpha, \theta\}$, where $\alpha$ denotes the mixing prior probability of unobserved data $Y$ and $\theta$ is the set of parameters $\{\mu, \Sigma\}$ defining the Gaussian distribution $p$. More specifically, if normal mixtures have $k$ clusters then $\alpha = \{\alpha_1, \ldots, \alpha_k\}$ is a $k$-dimensional vector, where $\alpha_h$ is a priori probability of cluster $h$ and $\mu = \{\mu_1, \ldots, \mu_k\}$, $\Sigma = \{\Sigma_1, \ldots, \Sigma_k\}$ are the *mean* and *covariance* vectors of Gaussian distribution $p$, respectively.
- $\Theta^{(t)} \equiv \{\alpha^{(t)}, \theta^{(t)}\}$ are parameters of the density at time $t$.
- $Z = \{X, Y\}$ is a complete dataset with $X$ is observed (or incomplete) and $Y$ is unobserved (or missing) data, respectively.

### 2.2. GAKREM algorithm

To better understand the GAKREM algorithm we start with brief explanation of the EM and *K*-means. In Section 2.2.2, the three-phase GAKREM algorithm is described in detail. There, we first describe encoding of chromosomes in a population, use of *K*-means to compute initial values of the parameters for the EM (phase I); second, we introduce a new technique to efficiently evaluate fitness values of the chromosomes by using logarithmic regression (phase II); and third, we outline the main body of GAKREM (phase III).

#### 2.2.1. EM and K-means overview

Suppose that data points in the incomplete dataset $X$ are independently and identically distributed with an underlying distribution $p$, the probability density function of $X$ can be expressed as

$$p(X|\Theta) = \prod_{i=1}^{N} p(x_i|\Theta). \tag{1}$$

From Eq. (1) the following incomplete data log-likelihood of the parameter $\Theta^{(t)}$ given dataset $X$ can be obtained:

$$l(\Theta|X) = \log p(X|\Theta) = \log \prod_{i=1}^{N} p(x_i|\Theta) = \sum_{i=1}^{N} \log p(x_i|\Theta). \tag{2}$$

In the maximum likelihood (ML) problem, we are interested in finding $\Theta$ that maximizes the incomplete data log-likelihood function in Eq. (2), namely

$$\Theta^* = \arg\max_{\Theta} l(\Theta|X). \tag{3}$$

Typically, Eq. (3) can be solved by using a Lagrange multiplier:

$$\frac{\partial_l}{\partial_\Theta} = 0. \tag{4}$$

In many problems, however, Eq. (4) becomes nonlinear. Thus, it is hard to find a closed-form solution for Eq. (4) although there exist iterative methods such as Newton or quasi-Newton for solving it. The general EM algorithm is an elaborate technique for obtaining $\Theta^*$ [9,33,34]. Recall that $Z = \{X, Y\}$ is a complete dataset, thus, the complete data log-likelihood function is defined by

$$l(\Theta|Z) = \log p(Z|\Theta) = \log p(X, Y|\Theta). \tag{5}$$

Given the current parameter estimates and the observed data $X$, the EM algorithm maximizes the incomplete data log-likelihood function by iteratively maximizing the expectation of the complete data log-likelihood function of Eq. (5). The EM algorithm consists of these two basic steps:

– *E-step:* at time $t$, the EM computes the expected value of the complete data log-likelihood function $l(\Theta^{(t)}|Z)$, given the parameters at time $t-1$ $\Theta^{(t-1)}$ and the observed data $X$. The result is a function:

$$Q(\Theta^{(t)}|\Theta^{(t-1)}) = E[\log p(Z|\Theta^{(t)})|X, \Theta^{(t-1)}] = \int \log p(X, Y|\Theta^{(t)}) \times p(Y|X, \Theta^{(t-1)}) \mathrm{d}_Y, \tag{6}$$

where $p(Y|X, \Theta^{(t-1)})$ is the marginal distribution of the unobserved data $Y$ computed by using Bayes' rule:

$$p(Y|X, \Theta^{(t-1)}) = \frac{P(X, Y|\Theta^{(t-1)})}{\int P(X, Y|\Theta^{(t-1)}) \mathrm{d}_Y}. \tag{7}$$

– *M-step*: the EM finds the parameter $\Theta$ to maximize the expectation computed in the *E*-step, that is

$$\Theta^{(t)} = \arg\max_{\Theta} l(\Theta|\Theta^{(t-1)}). \tag{8}$$

Starting from the initial guess of parameter $\Theta^{(0)}$, the EM's two steps are iteratively repeated until convergence. At each iteration the EM increases the incomplete data log-likelihood function $l(\Theta|X)$ and is guaranteed to converge to a local maximum of the log-likelihood function [9,33,34].

In the mixture density parameter estimation problem, assuming that the incomplete data $X$ is governed by $k$-cluster finite mixture distribution, its density function now can be written as the following probabilistic model:

$$p(x_i|\Theta) = \sum_{h=1}^{k} \alpha_h p_h(x_i|\theta_h), \tag{9}$$

where

- $x_i$ is a particulate $d$-dimension data point of the dataset $X$.
- $\Theta \equiv \{\alpha, \theta\}$ is the complete set parameter of the mixture.
- $\alpha = \{\alpha_1, \ldots, \alpha_k\}$ are the mixing probabilities satisfied $\alpha_h \geqslant 0\ \forall h$ and $\sum_{h=1}^{k} \alpha_h = 1$.
- $\theta = \{\theta_1, \ldots, \theta_k\}$ are the parameters defining clusters.
- $p_h$ is a Gaussian density function parameterized by $\theta_h = \{\mu_h, \Sigma_h\}$:

$$p_h(x_i|\theta_h) = p_h(x_i|\mu_h, \Sigma_h) = \frac{1}{2\pi^{d/2}|\Sigma|^{1/2}} e^{-\frac{1}{2}(x_i-\mu_h)^T \Sigma^{-1}(x_i-\mu_h)}. \tag{10}$$

Consequently, the incomplete data log-likelihood function corresponding to the $k$-cluster mixture is defined by

$$l(\Theta|X) = \log \prod_{i=1}^{N} p(x_i|\Theta) = \sum_{i=1}^{N} \log \sum_{h=1}^{k} \alpha_h \cdot p_h(x_i|\theta_h). \tag{11}$$

In the case of *k-cluster* finite mixtures, Eq. (5) for complete data log-likelihood function can be rewritten as

$$l(\Theta|Z) = \log p(Z|\Theta) = \log p(X, Y|\Theta) = \sum_{i=1}^{N} \sum_{h=1}^{k} y_i^h \log[\alpha_h \cdot p_h(x_i|\phi_h)], \tag{12}$$

where $y_i^h = 1$ and $y_i^j = 0$ with $j \neq h$ if cluster $h$ produced data point $x_i$ (see *Definitions*).

Given the initial guess of $\Theta^{(0)} \equiv \{\alpha^{(0)}, \theta^{(0)}\}$ at time 0, the two-steps EM algorithm iteratively operates until convergence:

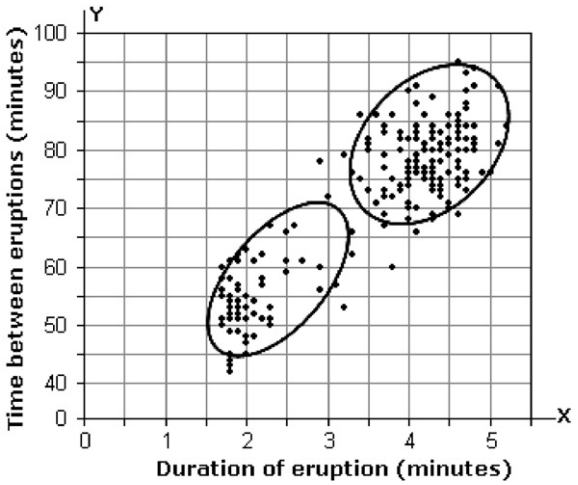



**Fig. 1.** Old Faithful data and the two-cluster result obtained by GAKREM; the optimal log-likelihood value is −8.9078.

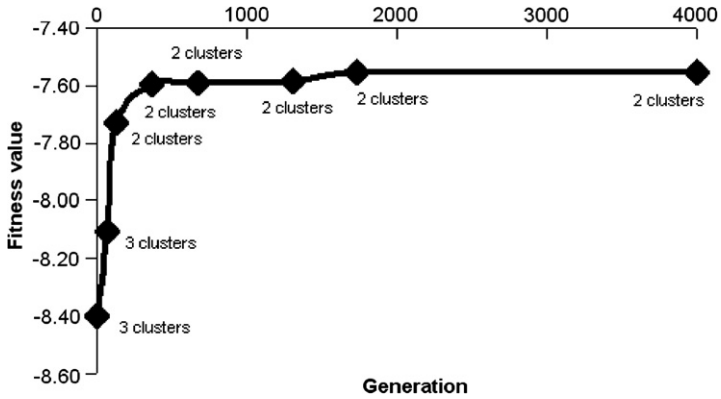
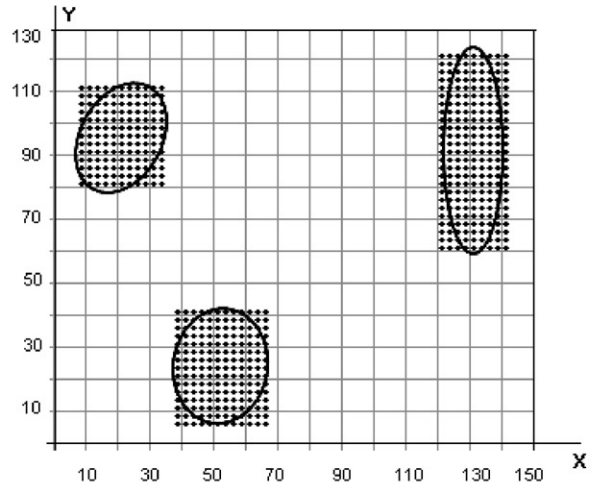



**Fig. 2.** GAKREM's behavior on the Old Faithful data; at the 1728th generation the optimal fitness value is −7.5568.

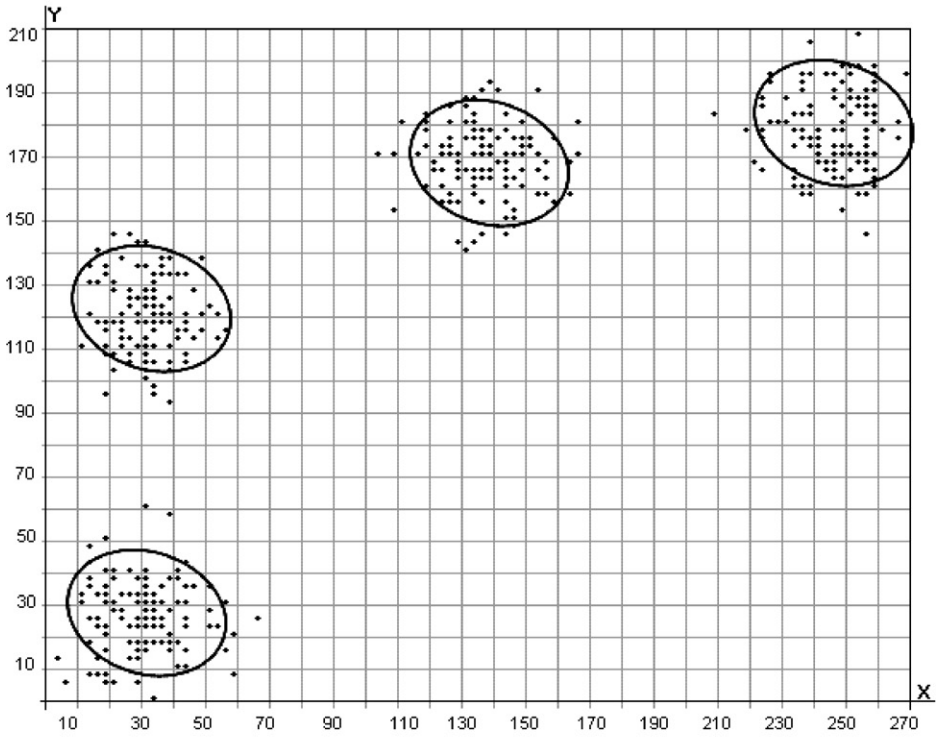
– *E-step:* at time $t$, computes the expected value of the complete data log-likelihood function in Eq. (12). Note that the conditional expected values of for every data point $y_i$ of the unobserved data $Y$ are derived by the Bayes' rule:

$$E(y_i^h|x_i,\Theta^{(t)}) = p(y_i^h = 1|x_i,\Theta^{(t)}) = \frac{\alpha_h^{(t)} p(x_i|\theta_h^{(t)})}{\sum_{j=1}^{k} \alpha_j^{(t)} p(x_i|\theta_h^{(t)})}. \tag{13}$$

– *M-step:* maximizes the log-likelihood function of the complete data $Z$ by updating the parameters $\Theta$ at time $t+1$ [4]:




(a) 3-cluster mixture data; the optima log-likelihood values is -6.85.



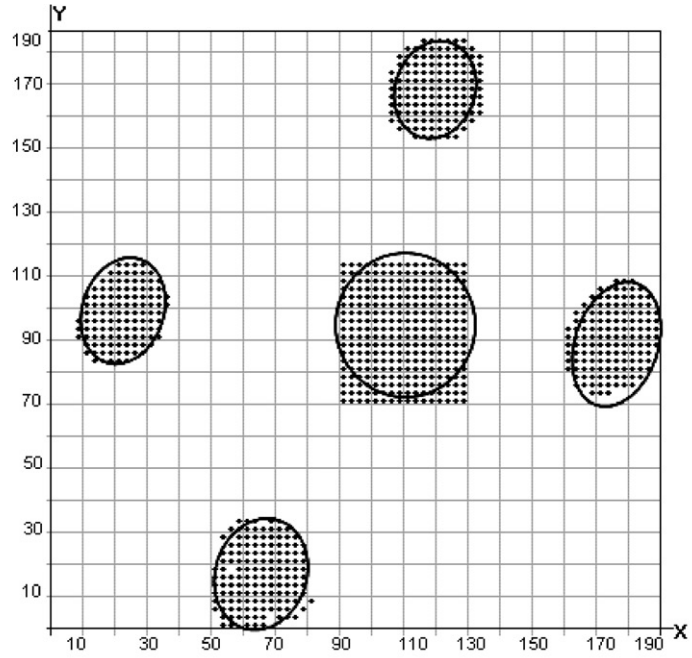

(b) 4-cluster mixture data; the optima log-likelihood values is -7.34.

**Fig. 3.** GAKREM results on several datasets (the results were the same in all 100 trials) without prior knowledge of the number of clusters to be found in the data.

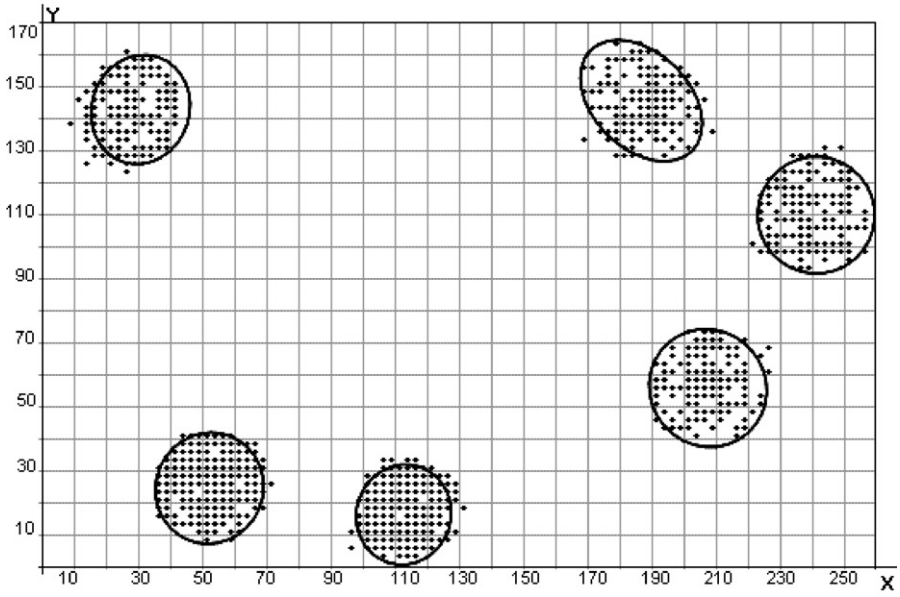
$$\alpha_h^{(t+1)} = \frac{1}{N}\sum_{i=1}^{N} E(y_i^h|x_i, \Theta^{(t)}), \tag{14}$$

$$\mu_h^{(t+1)} = \frac{\sum_{i=1}^{N} x_i E(y_i^h|x_i, \Theta^{(t)})}{\sum_{i=1}^{N} E(y_i^h|x_i, \Theta^{(t)})}, \tag{15}$$

$$\Sigma_h^{(t+1)} = \frac{\sum_{i=1}^{N} E(y_i^h|x_i, \Theta^{(t)})(x_i - \mu_h^{(t+1)})^{\mathrm{T}}(x_i - \mu_h^{(t+1)})}{\sum_{i=1}^{N} E(y_i^h|x_i, \Theta^{(t)})}. \tag{16}$$

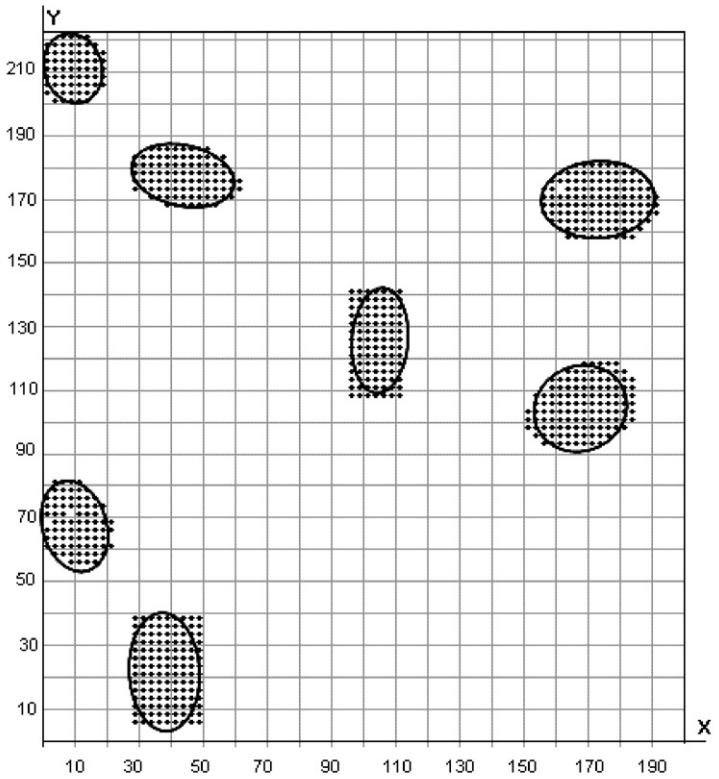



(c) 5-cluster mixture data; the optimal log-likelihood value is -7.05.

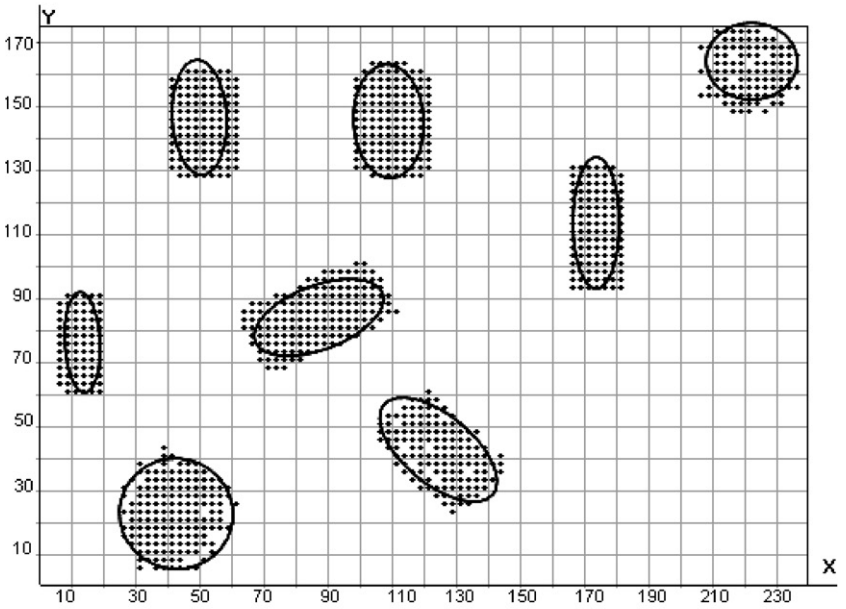



(d) 6-cluster mixture data; the optimal log-likelihood value is -7.16.

**Fig. 3** (*continued*)

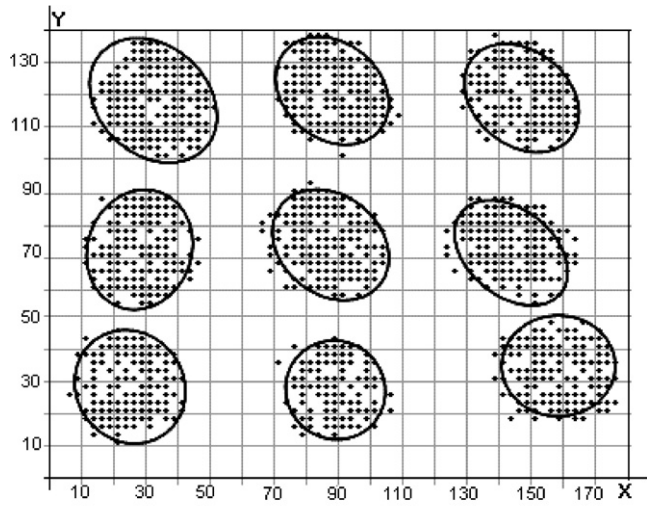



(e) 7-cluster mixture data; the optimal log-likelihood value is -6.97.




(f) 8-cluster mixture data; the optimal log-likelihood value is -7.38.

**Fig. 3** (*continued*)




(g) 9-cluster mixture data; the optimal log-likelihood value is -7.59.

**Fig. 3** (*continued*)

In addition to the initial guess of parameter $\Theta$, the EM algorithm also requires the number of clusters to be specified a priori to operate.

The objective of the *K*-means algorithm is to minimize total intra-cluster variance, namely, $V = \sum_{h=1}^{k} \sum_{x_j \in c_h} |x_j - \mu_h| \rightarrow \min$ where $c_h$ is a cluster $h$, $h = 1, \ldots, k$, and $\mu_h$ is the mean point of all the points $x_j \in c_h$. Briefly, given a $k$ number, the *K*-means algorithm performs these steps:

(1) Randomly select $k$ points as initial medoids.
(2) Repeat
(3) Form $k$ clusters by assigning each point to its closest medoid.
(4) Re-compute the medoid of each cluster as $\mu_h = \frac{1}{|C_h|} \sum_{i=1}^{|C_h|} x_i$ where $|c_h|$ is the cardinality of cluster $c_h$.
(5) Until medoids do not change.

Note that both EM and *K*-means algorithms are hill-climbing approaches to search for the optimal solutions, i.e. $\Theta^* = \arg\max_{\Theta} l(\Theta|X)$ in EM and $V = \sum_{h=1}^{k} \sum_{x_j \in c_h} |x_j - \mu_h| \rightarrow \min$ in *K*-means. Thus, EM algorithm is very similar to *K*-means algorithm in each iteration in that the alternation between expectation and maximization steps in EM corresponds to the alternation between reassigning points to closest medoids and recomputing of medoids in *K*-means.

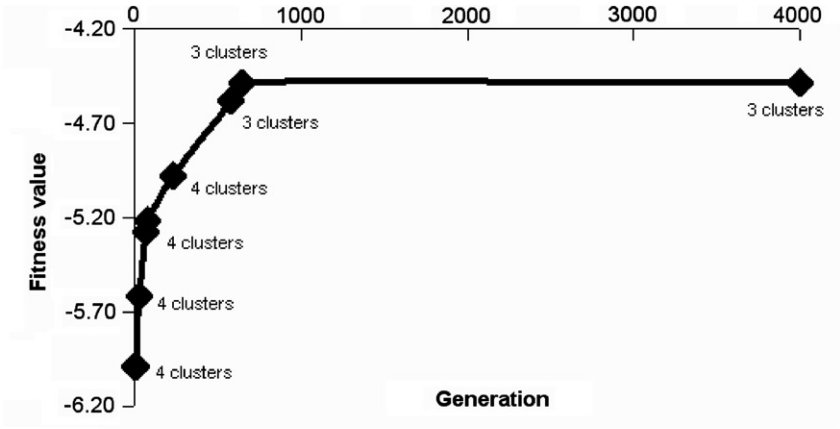
*2.2.2. GAKREM – pseudocode*

GAKREM has three phases. In phase I, it computes the initial guess of parameters. Phase II is a novel method to evaluate the fitness of the generated chromosomes, while in phase III GAs are used to simultaneously search the optimal fitness value and the best number of clusters of the mixture.

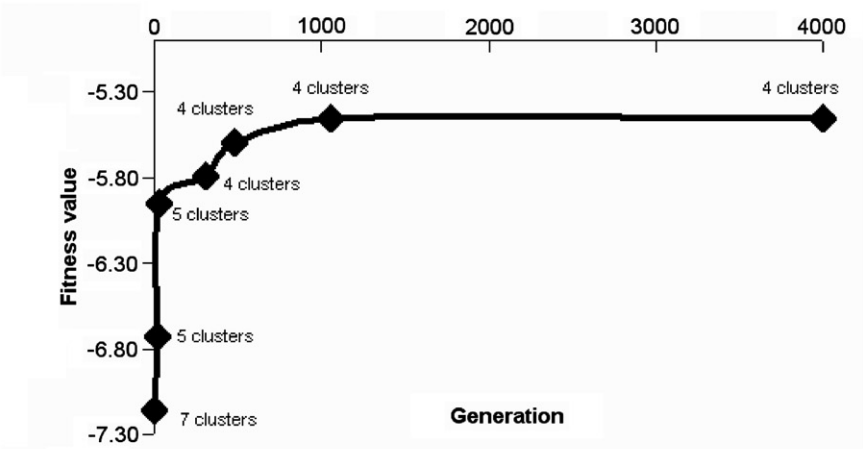
*2.2.2.1. Phase I: Computing the initial value of parameters $\Theta$.* We use GAs to perform a global search for the best solution to address the drawbacks of EM and *K*-means, namely that they often end up in a local optimum. In contrast to the *K*-means and EM's use of a hill-climbing search, we use GAs to move the population away from local optima areas [8,14]. The GAKREM starts with population $P$ which is a set of the (a) initial guess of parameter $\Theta^{(0)}$ and, (b) the number of clusters $k$ used in the EM, represented by chromosomes. A chromosome in $P$ is encoded as a binary string of length $N$ (corresponding to $N$ data points in the incomplete data $X$). The $i$th locus of a chromosome is set to 1 if the corresponding data point $i$th in dataset $X$ is a medoid of the cluster, otherwise 0. As a result, the number of 1-loci in each chromosome is also the $k$ number of clusters input to the EM. We illustrate the idea by means of the following example:

*Example:* Suppose that $N$ is 10; the number of chromosomes in $P$ is 3, then each chromosome can be randomly configured by a uniform distribution as:

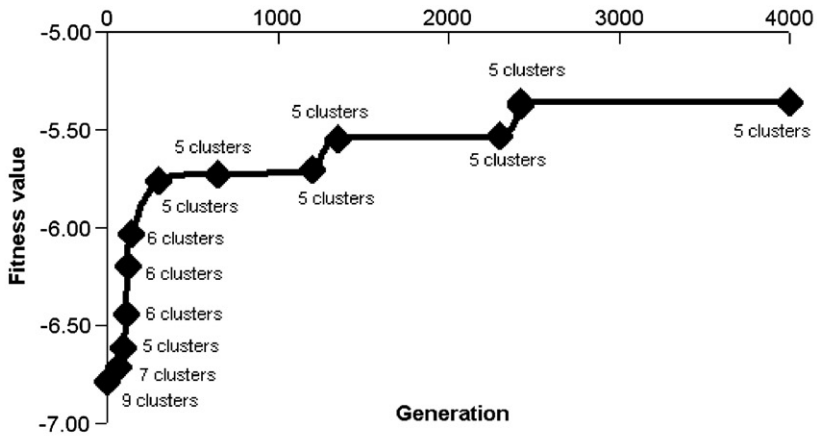
| Position | 1234567890 |
|---|---|
| Chromosome 1: | 0001000010 |
| Chromosome 2: | 0100011001 |
| Chromosome 3: | 1010100100 |




(a) behavior on the 3-cluster data; it stabilizes at the fitness value of -4.4869 that corresponds (at generation 646) to 3 clusters.

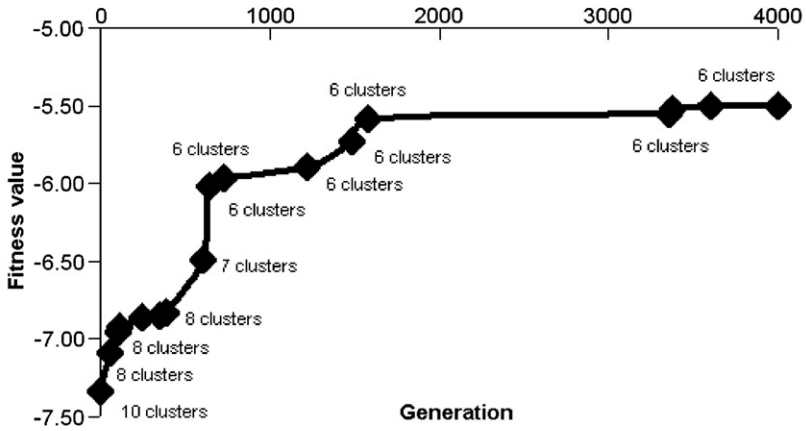



(b) behavior on the 4-cluster data; it stabilizes at the fitness value of -5.4547 that corresponds (at generation 1059) to 4 clusters.

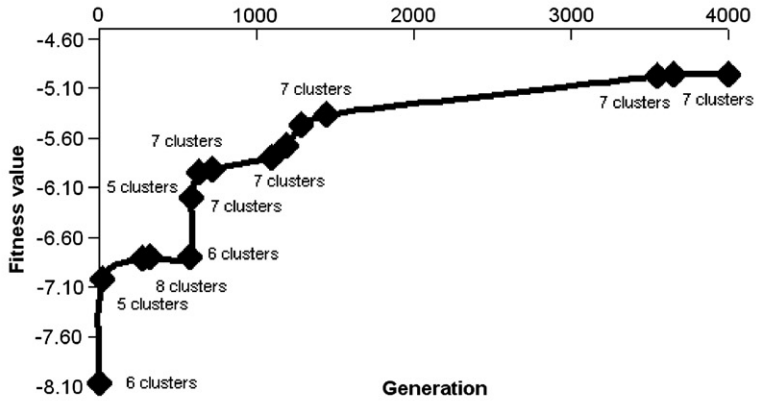



(c) behavior on the 5-cluster data; it stabilizes at the fitness value of -5.3616 that corresponds (at generation 2426) to 5 clusters.

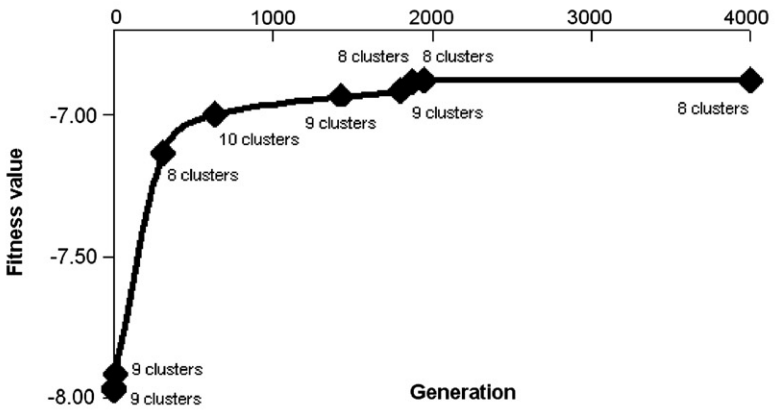
**Fig. 4.** Behavior of GAKREM on several datasets.




(d) behavior on the 6-cluster data; it stabilizes at the fitness value of -5.5028 that corresponds (at generation 3606) to 6 clusters.



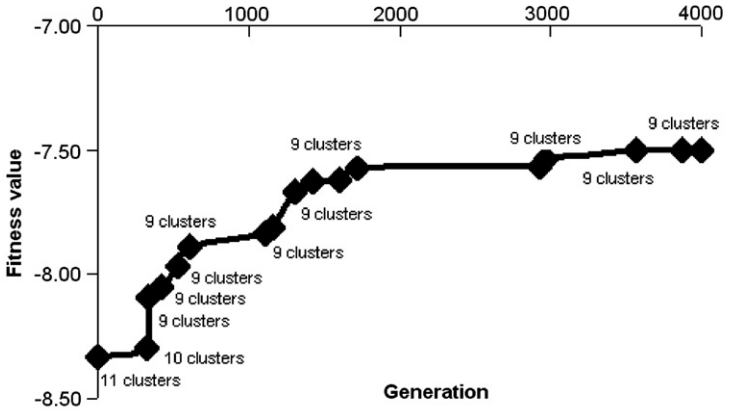

(e) behavior on the 7-cluster data; it stabilizes at the fitness value of -4.9659 that corresponds (at generation 3635) to 7 clusters.




(f) behavior on the 8-cluster data; it stabilizes at the fitness value of -6.8795 that corresponds (at generation 1946) to 8 clusters.

**Fig. 4** (*continued*)

This means that chromosome 1 specifies two clusters: the medoid of the cluster 1 is the 4th data point and the medoid of cluster 2 is the 9th data point. Chromosome 2 has four clusters in which the medoids are the 2nd, 6th, 7th, 10th data points, respectively. Similarly, the 1st, 3rd, 5th and 8th data points are the medoids of the four clusters represented by the chromosome 3.




g) behavior on the 9-cluster data; it stabilizes at the fitness value of -7.5016 that corresponds (at generation 3869) to 9 clusters.

**Fig. 4** (*continued*)

To optimize the use of computer memory, instead of storing the full length of chromosomes, we encode only the positions of 1-loci in chromosomes, thus the chromosomes in the above example can be represented as a set of ordered numbers (4,9), (2,6,7,10), and (1,3,5,8), respectively. Each chromosome now specifies the $k$ number of clusters *plus* the medoids of clusters in the data. Our encoding scheme is similar to the one used in [11,20] except that the number of 1-loci in our chromosomes ranges from 2 to $N^{1/2}$. In contrast to the label-based encoding, which usually has disastrous consequences during standard crossover and mutation operations, as thoroughly discussed in [12], our encoding is not context-sensitive. This is because any data point can be initially selected as a medoid. The only drawback of the proposed encoding is that the number of 1-loci (i.e. the number of clusters) of the generated offsprings can be out of range. In such cases, the generated offsprings are eliminated. On the other hand, the proposed encoding has one more advantage over the label-based encoding in terms of computer memory (for $k \ll N$). To use the EM algorithm we need information about the value of parameter $\Theta^{(0)}$ at time 0. To do so, we compute parameters $\{\alpha^{(0)}, \theta^{(0)}\}$ by performing $K$-means clustering as follows:

**Phase I: Computing initial parameters** $\{\alpha^{(0)}, \theta^{(0)}\}$
Input: -a dataset $X$ with $N$ data points
-a chromosome $C$ encoding $k$ clusters: $loci[1,\ldots,k]$
Output: $\Theta^{(0)} = \{\alpha^{(0)}, \theta^{(0)}\}$
1 $k \leftarrow$ length of chromosome $C$
2 **For** $i \leftarrow 1$ to $N$
3 **For** $h \leftarrow 1$ to $k$
4 Calculate the distance $\|x[i] - x[loci[h]]\|$
where $x\,[loci[h]]$ is the medoid of cluster $h$
5 **End For**
6 Assign the data point $x_i$ to the cluster with the *closest medoid.*
7 **End For**
8 **For** $h \leftarrow 1$ to $k$
9 Calculate $\alpha_h^{(0)} = \frac{|c_h|}{N}, \mu_h^0 = \frac{1}{|c_h|}\sum_{i=1}^{|c_h|} x_i, \sum_h^0 = \frac{1}{|c_h|}\sum_{i=1}^{|c_h|}(x_i - \mu_h^0)(x_i - \mu_h^0)^T$
*where* $|c_h|$ is the cardinality of cluster $c_h$
10 **End For**
11 Return $\{\alpha^{(0)}, \theta^{(0)}\}$

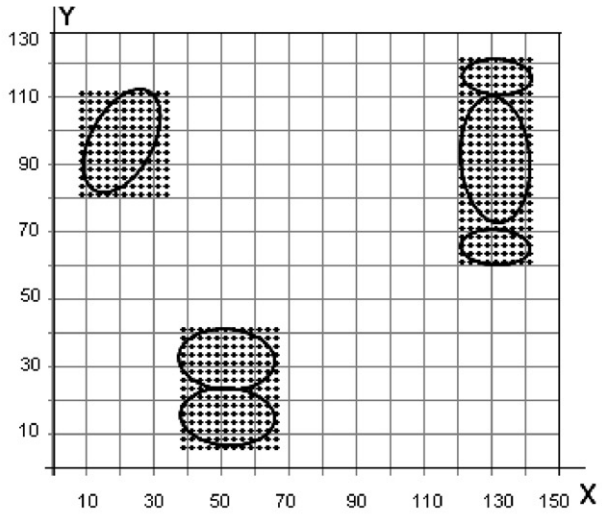
*2.2.2.2. Phase II: Calculating the chromosomes' fitness functions.* In addition to the chromosome encoding scheme we need to specify a function to evaluate their fitness. In the case of finite mixtures we cannot simply perform EM on each chromosome and use the resulting log-likelihood function as a fitness value. The reason is that the EM suffers from monotonicity property, namely, when the number of clusters increases the maximum log-likelihood $l(\theta|X)$ also increases. Therefore, each chromosome is evaluated by its maximum log-likelihood function *minus* a penalty corresponding to the number of clusters encoded by the chromosome. Using Occam's razor [6], we apply the following simple fitness function to evaluate chromosome $C$:

$$fit(C) = l(\Theta|C) - \log(k), \tag{15}$$

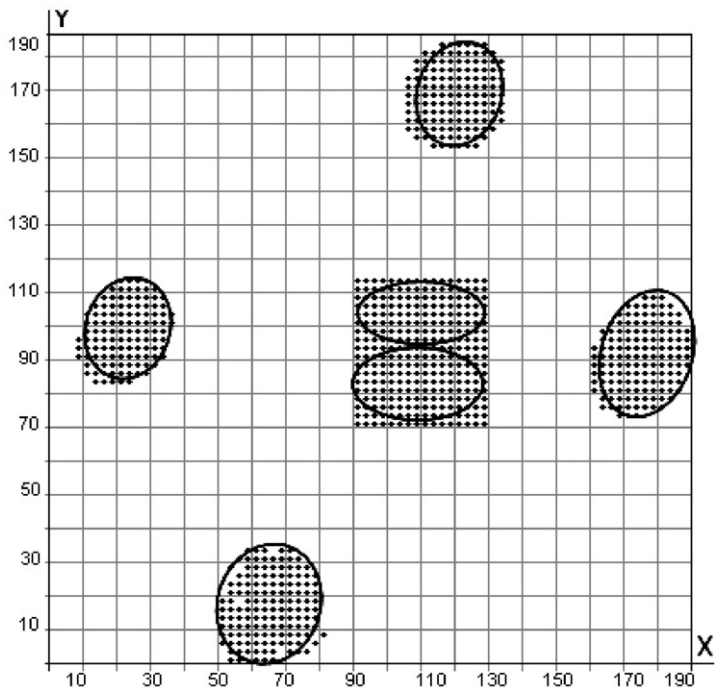
where $k$ is the number of clusters encoded by chromosome $C$ and $l(\Theta|C)$ is log-likelihood function resulting from performing the EM on chromosome $C$.

Let us note that the GAs are still exhaustive search algorithms so Eq. (15) would be an expensive formula to use. To decrease the computational cost, instead of running a full EM on each chromosome until convergence we perform only a partial EM in (a few) $r$ initial steps. Then we use logarithmic regression to predict the log-likelihood function at a convergence point. There are several papers on the convergence rate of the log-likelihood function $l(\Theta|C)$ of the general EM algorithm [9,24,33,34]. In most of cases, the convergence curves of $l(\Theta|C)$ are fitted by logarithmic curves. The expected maximum log-likelihood function is computed as

$$E(C) = a\log(t) + b, \tag{16}$$

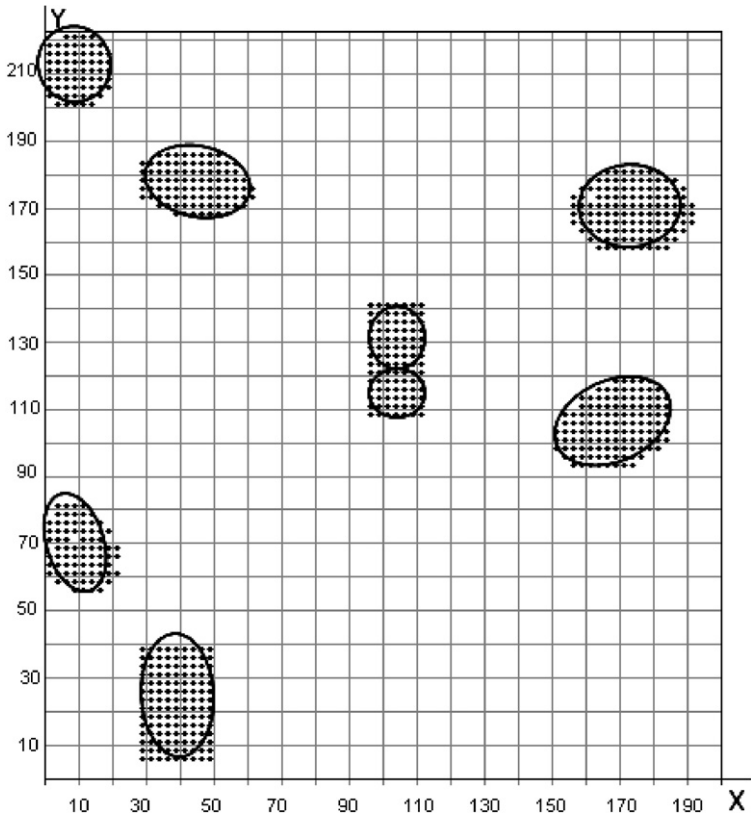



(a) 3-cluster mixture data and the (wrong) result obtained by LCV.




(b) 5-cluster mixture data and the (wrong) result obtained by LCV.

**Fig. 5.** LCV (all incorrect) results on three-, five-, seven-, eight- and nine-cluster mixture datasets.




(c) 7-cluster mixture data and the (wrong) result obtained by LCV.

**Fig. 5** (*continued*)

where

$$a = \frac{r\sum_{t=1}^{r} t \times l_t - \sum_{t=1}^{r} t \times \sum_{t=1}^{r} l_t}{r\sum_{t=1}^{r} (t)^2 - (\sum_{t=1}^{r} t)^2} \quad \text{and} \quad b = \frac{\sum_{t=1}^{r} l_t - a\sum_{t=1}^{r} t}{r}$$

with $l_t$ is the log-likelihood value at each iteration $t$ in the EM and $r$ is set to 5 in all experiments. The fitness value in Eq. (15) now becomes

$$fit(C) = E(C) - \log(k). \tag{17}$$

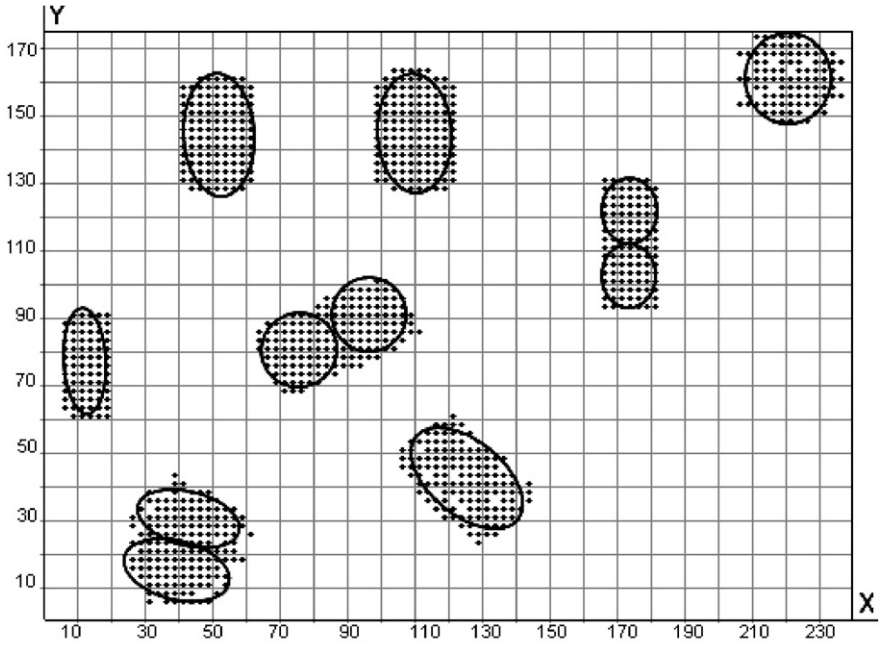
Several cluster validity techniques have been suggested to find an optimal number of clusters such as partition coefficient [3], minimum description length [25,27] and Bayesian information criterion (BIC) [27]. We compare our approach to the likelihood cross-validation (LCV) technique presented in [28] to determine the most correct number of clusters. The LCV technique is briefly reported as follows. The dataset, $D$, is repeatedly partitioned into $v$-folds (say, 10). At the time $i$th, the fold $i$th, $S_i$, is used to build the parameter $\Theta_i$ of the model and the remainder of the data, $D \setminus S_i$, is used to evaluate the model. Let $l_k$ be the cross-validated estimate of the test log-likelihood for the $k$-cluster dataset, then $l_k$ can be defined as: $l_k = \frac{1}{v}\sum_{i=1}^{v} l_k(D \setminus S_i|\Theta_i)$. Let $k_{\text{true}}$ be the "true" number of clusters of the dataset $D$. Starting with $k$ equal to 1, the number $k$ is gradually increased until $k$ locks on to the $k_{\text{true}}$, i.e. $l_{\text{true}} \leqslant l_k$.

*2.2.2.3. Phase III: Finding the optimal fitness value.* This phase works iteratively in three steps, as outlined below.

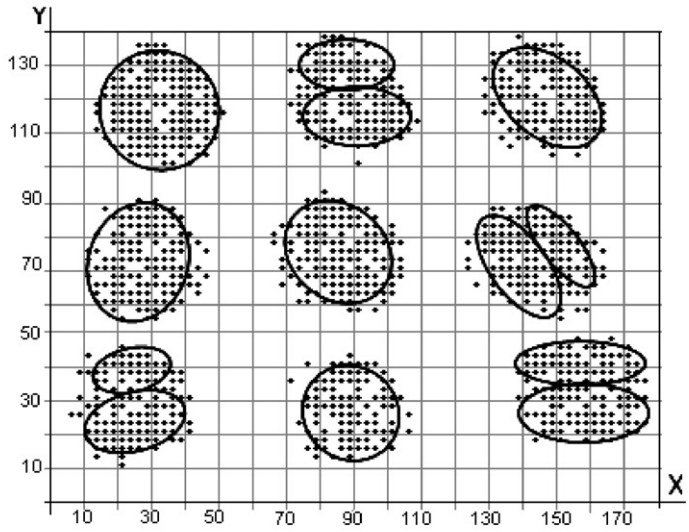
*Step 1 Initialize population and evaluate fitness value of chromosomes.*

Generate distinct $M$ chromosomes for the population. Each chromosome has $k$ 1-loci, where $k$ is a random number ranging from 2 to $N^{1/2}$. The positions of 1-loci in each chromosome are randomly distributed by uniform distribution (see example in Section 2.2.2). We use $M = 64$ in our experiments.

For each chromosome $C$ – a finite mixture model – we first compute the initial guess of parameter $\Theta^{(0)} = \{\alpha^{(0)}, \theta^{(0)}\}$ by using the method presented in phase I. Together with the number of clusters $k$ encoded in the chromosome, the parameters $\Theta^{(0)}$ and $k$ constitute input to the EM algorithm to calculate the fitness value, $fit(C)$, by Eq. (17).




(d) 8-cluster mixture data and the (wrong) result obtained by LCV.




(e) 9-cluster mixture data and the (wrong) result obtained by LCV.

**Fig. 5** (*continued*)

1 $M \leftarrow 64$
2 **For** $i \leftarrow 1$ to $M$
3 $k \leftarrow$ random number $2 \ldots \sqrt{N}$
5 **For** $h \leftarrow 1$ to $k$
6 chromosome[$i$].loci[$h$] $\leftarrow$ random number $1,\ldots,N$
7 **End For**
8 Use the method presented in phase I to compute $\Theta^{(0)}$ of chromosome[$i$].
9 Use Eq. (17) in phase II to calculate *fit* (chromosome[$i$]).
10 **End For**

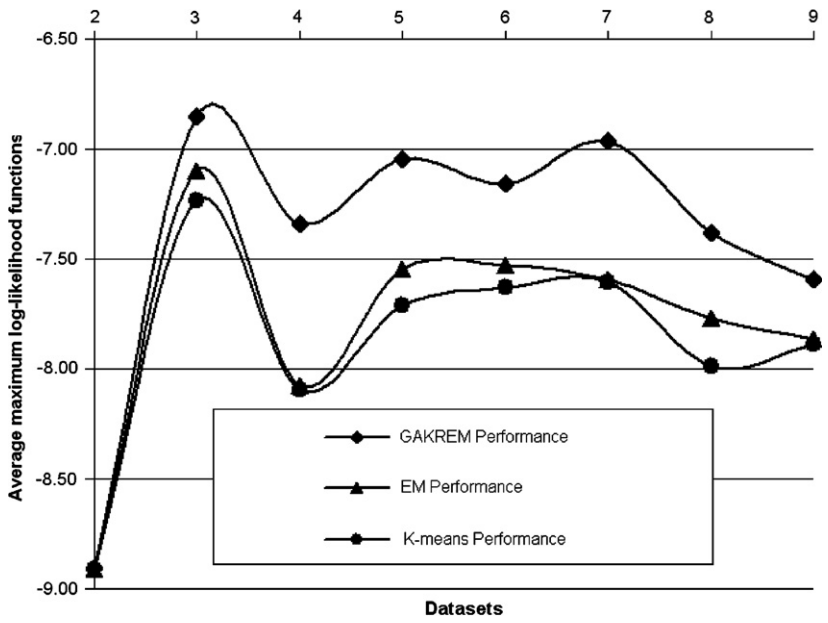



**Fig. 6.** Average maximum log-likelihood functions for GAKREM, EM and *K*-means algorithms, on two to nine-cluster datasets.

*Step 2 Generate new population.*

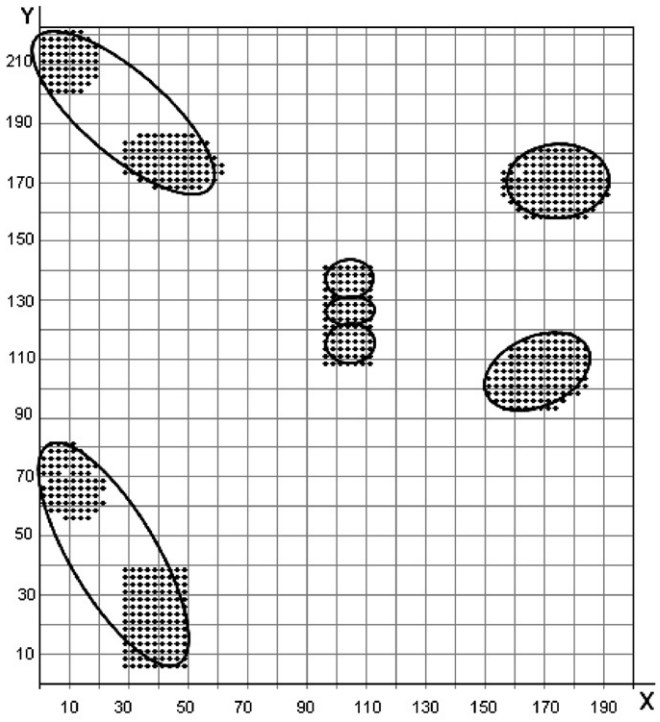
Using roulette wheel method a pair of chromosomes (parents) in the population is selected for mating. This includes two operators:

- *Crossover*, which selects loci from the parent chromosomes and creates a new offspring. GAKREM uses uniform crossover prototype where loci are randomly copied from the first or the second parent. The new offspring may have a different number of clusters and the 1-loci, thus changing the first guess of parameters.
- *Mutation*, which occurs with a probability rate. A randomly selected locus of the offspring is inversed, which may increase or decrease the number of clusters in the offspring. The mutation operator might move the population away from local optima areas.

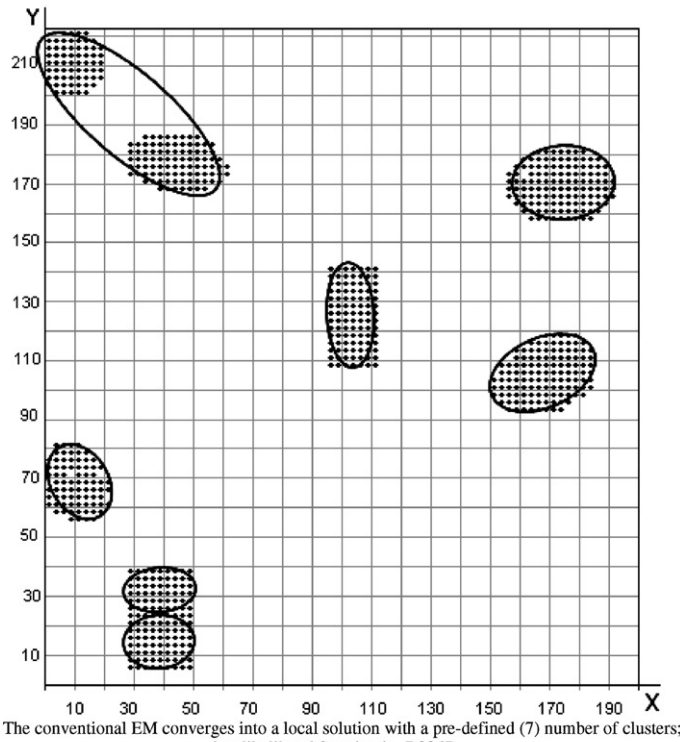
By using genetic operators, equal parents may not create equal offspring. Thus, the number of clusters encoded by offspring needs to be restrained to the interval from 2 to $N^{1/2}$. The advantage is that the population may move to a global optimal solution.

Note that if the fitness value of the new offspring is higher than the lowest fitness value in the population then the corresponding chromosome is replaced by the new offspring. There are several methods to determine when to stop the algorithm in stage 2 after a number of generations of evolutionary search [8,12,14]. One possibility is to use fitness convergence. The algorithm stops if the fitness value of the population has not changed over $n$ generations. In that case it is not guaranteed that the solution has reached global optimum because the new offsprings (of the next generation $n + 1$) may have better fitness values. Another option is to use fitness threshold: if the fitness value of the population is less than a specified threshold then stop. However, it is difficult to come up with a proper threshold for a given dataset. In GAKREM stage 2 is repeated over $g$ generations to improve the fitness value of the chromosomes in the population; we use $g = 4000$ in all experiments.

```
11  g← 4000
12  For i ← 1 to g
13    dad ← select a chromosome in population by roulette wheel method
14    mom ← select a chromosome in population by roulette wheel method
15    offspring ← crossover(dad, mom)
16    If (mutation rate is satisfied)
17      offspring ← mutation(offspring)
18    End If
19    Use the method presented in phase I to compute Θ(0) of the offspring.
20    Use Eq. (17) in phase II to compute fit(offspring).
21    If (fit (offspring) > the lowest fitness value in the population)
22      Replace the corresponding chromosome by the offspring
23    End If
22  End For
```

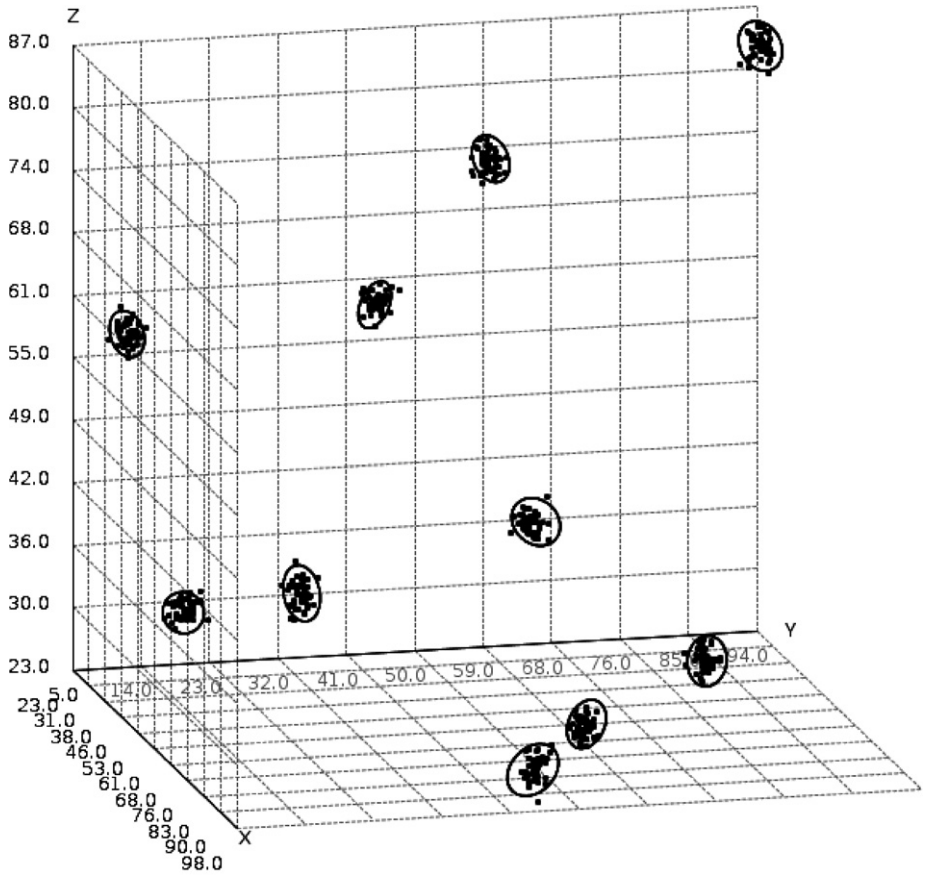



(a) K-means converges into a local solution with a pre-defined (7) number of clusters; maximum log-likelihood function is -7.6184.




(b) The conventional EM converges into a local solution with a pre-defined (7) number of clusters; maximum log-likelihood function is -7.2367.

**Fig. 7.** Sensitivity to initialization for *K*-means and conventional EM. Note that for this dataset GAKREM's average maximum likelihood function stabilized at −6.9666.




(a) 10-cluster data and the result obtained by GAKREM with maximum log-likelihood at -6.463.

**Fig. 8.** Examples illustrating sensitivity to initialization of EM and *K*-means algorithms contrasted with GAKREM performance. The above data were generated by our visualization tool available at http://www.egr.vcu.edu/cs/gakrem.

*Step 3 Use the general EM on the chromosome which has the highest fitness value, until convergence.*

*2.2.3. Analysis of computational time*

In step 1 of initializing population, GAKREM requires the computation of initial parameter $\Theta^{(0)}$: O(*Nkd*); the fitness value for each chromosome: O(*rNkd*) (with *r* is set to 5 in our experiment), thus, the overall computational time is O(*MrNkd*) (with $M = 64$). In step 2, the computational time takes O(*M*) for roulette wheel method; O(*k*) for genetic operators; O(*Nkd*) for computing the first parameter $\Theta^{(0)}$; O(*rNkd*) for the fitness value for each chromosome, thus, GAKREM takes approximately O(*grNkd*) (with $g = 4000$). Finally, the computational time for the general EM in the last step is O(*tNkd*) where *t* is experimentally set to 100. Let us note that *k* is length of chromosomes and thus, by using the encoding scheme described in Section 2.2.2, *k* is always less (or equal) than $N^{1/2}$. Therefore, the computational time of GAKREM is O(*grN2d*) + O(*tN2d*) for the lower bound and O($grN^{3/2}d$) + O($tN^{3/2}d$) for the upper bound. As a result, in terms of computational time, GAKREM is much more efficient than multiple runs of the EM/*K*-means as discussed in [1].

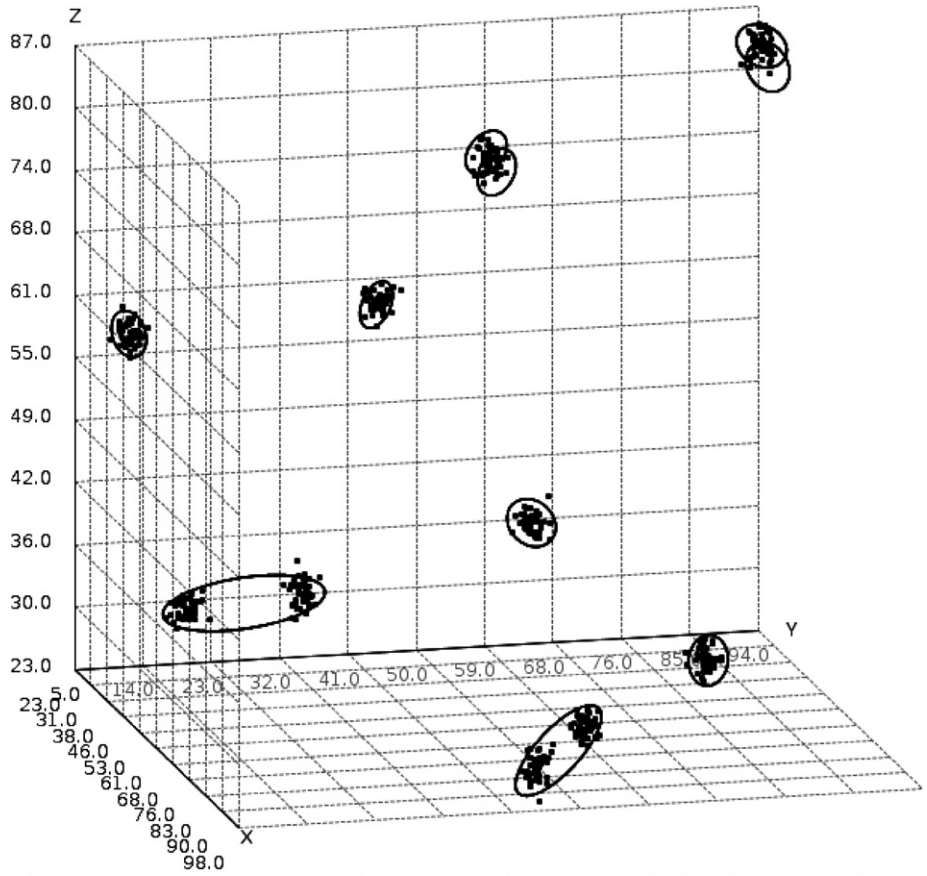
## 3. Results and discussion

We have conducted extensive experiments using GAKREM on two kinds of data. The first are manually-generated datasets and the second are randomly-generated datasets as used in [16,34]. The probability of mutation was set experimentally to 0.15 for all experiments.

### 3.1. Experiments on manually-generated data

*3.1.1. Performance*

The experiments are performed on cluster mixtures from two to nine clusters. To test the robustness of GAKREM we repeated the experiments 100 times for each dataset, and have found that it found correct number of clusters for each mixture, in all trials.




(b) 10-cluster data and the (wrong) result obtained by EM with a maximum log-likelihood function at -7.276; EM requires the number of clusters (10) to be specified a priori.

Fig. 8 (*continued*)

Fig. 1 illustrates the result of GAKREM on a two-cluster mixture derived from the Old Faithful dataset in [5]. This dataset has two dimensions: duration of an eruption of the geyser in minutes and interruption time. GAKREM precisely recognizes a two-cluster mixture, of course, without the pre-defined number of clusters. Fig. 2 shows behavior of GAKREM in a random trial, picked from 100 trials, in searching for global optimum fitness values of chromosomes in the Old Faithful data.

The results of GAKREM on three to nine-cluster mixtures are shown in Fig. 3. These datasets are manually generated by our tool available at http://www.egr.vcu.edu/cs/gakrem, which have 374, 402, 818, 724, 695, 988, 1122 data points with two dimensions, respectively. GAKREM robustly generates the correct number of clusters for each mixture in all trials.

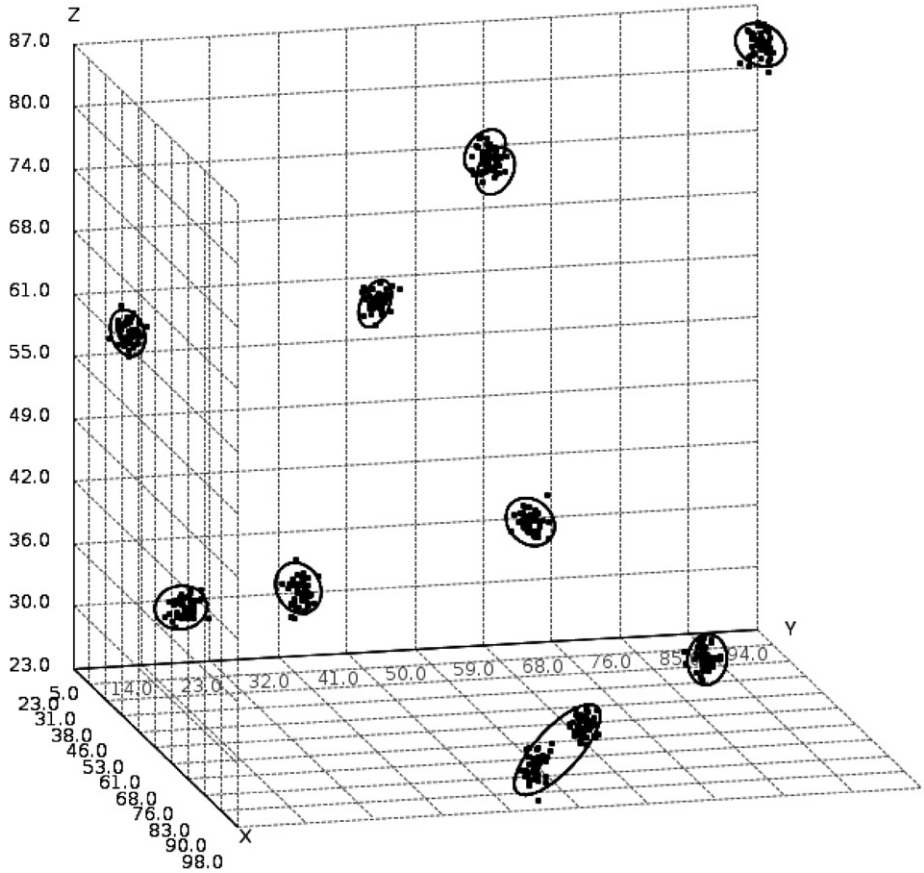
The behavior of GAKREM in a random trial (out of 100 trials) is shown in Fig. 4. It performs a thorough search simultaneously for the number of clusters and the initial medoids of clusters.

### 3.1.2. GAKREM versus LCV

For comparison in determining the best number of clusters in each generated dataset, we use the LCV technique [28] from Weka [32]. The very expensive LCV technique determined correctly the number of clusters only in two-, four- and six-cluster mixture datasets while failed to recognize the underlying clusters in three-, five-, seven-, eight- and nine-cluster mixture datasets, as shown in Fig. 5.

### 3.1.3. GAKREM versus EM and K-means algorithms

For comparison of the maximum log-likelihood functions of the mixtures performed by GAKREM, EM and *K*-means algorithms, we implemented and used the last two algorithms 100 times on the same datasets as described above. It is important to note that to operate the EM and *K*-means algorithms we must specify a priori the number of clusters. Thus, we use from two to nine clusters as input parameters. Fig. 6 shows comparison for the average maximum log-likelihood functions resulting from the EM, *K*-means and GAKREM while testing on two to nine clusters. As can be seen GAKREM




(c) 10-cluster data and the (wrong) result obtained by K-means with a local maximum log-likelihood function at -7.0783; K-means requires the number of clusters (10) to be specified a priori.

**Fig. 8** (*continued*)

significantly outperforms both the conventional EM and the *K*-means algorithms in terms of maximum log-likelihood functions.

An example of the drawback of the very sensitive initialization of the EM and *K*-means is shown in Fig. 7 where the maximum log-likelihood functions converge to local maximum areas in the seven-cluster mixture.

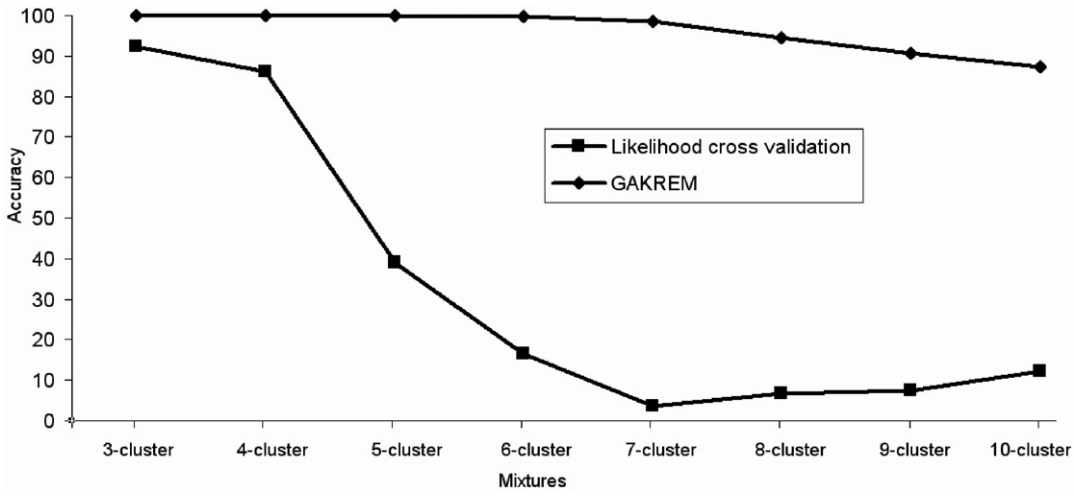



**Fig. 9.** Accuracies of GAKREM and LCV on the 3 to 10-cluster mixtures over 1000 datasets (each consisting of 500-d dimensional (2–10) data points).

### 3.2. Experiments on randomly-generated data

To randomly generate datasets we use the following algorithm based on the work reported in [16,34]:

*Pseudocode to* generate data:

Input: $N$ = the data points to be generated
$d$ = the dimension of data points
$k$ = the number of cluster to be generated
Output: a clustered dataset $N \times d$ dimensions

1 **For** $h \leftarrow 1$ to $k$
2 Set the size of cluster $h$: $c_h \leftarrow N/k$
3 **For** $i \leftarrow 1$ to $d$
4 Generate cluster center $\mu_h$ at uniform distribution over $[0 \cdots 100]$ for each dimension.
5 Scatter the appropriate number of points $c_h$ around $\mu_h$ according to a Gaussian distribution centered on $\mu_h$ and having covariance matrix $\Sigma_h$ constantly set equal to 1.
6 **End For**
7 **For** $j \leftarrow h - 1$ to 1
8 Go to line 2 if cluster $h$ overlaps cluster $j$
9 **End For**
10 **End For**

In this algorithm, the two clusters $h$ and $j$ overlap if $\frac{m_h+m_j}{c_h+c_j} > 0$, where $m_h$ and $m_j$ denote the number of misclassified points from clusters $h$ and $j$. Note that a data point $x_i$ belonging to cluster $h$ has been misclassified if $p(h|x_i) < p(j|x_i)$.

#### 3.2.1. Performance

We used the above algorithm to generate 1000 datasets, with the number of data points is fixed at 500, while the dimension $d$ is randomly set from 2 to 5 and the number of cluster $k$ is randomly set from 3 to 10. For each generated dataset, we ran the GAKREM, EM and *K*-means 100 times to gauge the robustness of the algorithms. GAKREM obtained the mixtures with the correct number of clusters at the average accuracy of 97%, without the need to specify the number of clusters. Fig. 8 demonstrates experiments for the generated 3-d dataset with 10 clusters in which GAKREM recognized the correct number of clusters while EM and *K*-means failed to do so.

#### 3.2.2. GAKREM versus LCV

Again, we used the LCV on the 1000 generated datasets to test the capability of obtaining the optimal number of clusters underlying mixture. To gauge the robustness, we repeated the experiments 100 times for each generated dataset. The accuracy of the LCV is only 31%. More importantly, as shown in Fig. 9, the performance of the LCV is good only if the mixtures have a small number of underlying clusters (less than 5), otherwise it fails to determine the best number of clusters in the mixture datasets.

In further to compare clustering results against external criteria, we use the Rand index [23] as a measure of agreement. While randomly generating datasets, if data points belong to the same cluster they will be labelled the same class. The Rand index to evaluate the agreement between classes and output clusters is computed as $\frac{a+b}{a+b+c+d}$, where

$a$ = the number of pair of data points having a different class and a different cluster.
$b$ = the number of pair of data points having the same class and the same cluster.
$c$ = the number of pair of data points having a different class and the same cluster.
$d$ = the number of pair of data points having the same class and a different cluster.

**Table 1**
Comparison of average Rand index values for GAKREM and LCV on 3 to 10-cluster mixtures of 1000 datasets (standard deviation value)

| # Clusters in dataset | Rand index | |
|---|---|---|
| | GAKREM | LCV technique |
| 3 | 1 (±0) | 0.91 (±0.03) |
| 4 | 1 (±0) | 0.85 (±0.03) |
| 5 | 1 (±0) | 0.40 (±0.04) |
| 6 | 0.99 (±0.01) | 0.20 (±0.06) |
| 7 | 0.99 (±0.01) | 0.10 (±0.07) |
| 8 | 0.96 (±0.02) | 0.10 (±0.07) |
| 9 | 0.90 (±0.02) | 0.10 (±0.09) |
| 10 | 0.86 (±0.02) | 0.15 (±0.09) |

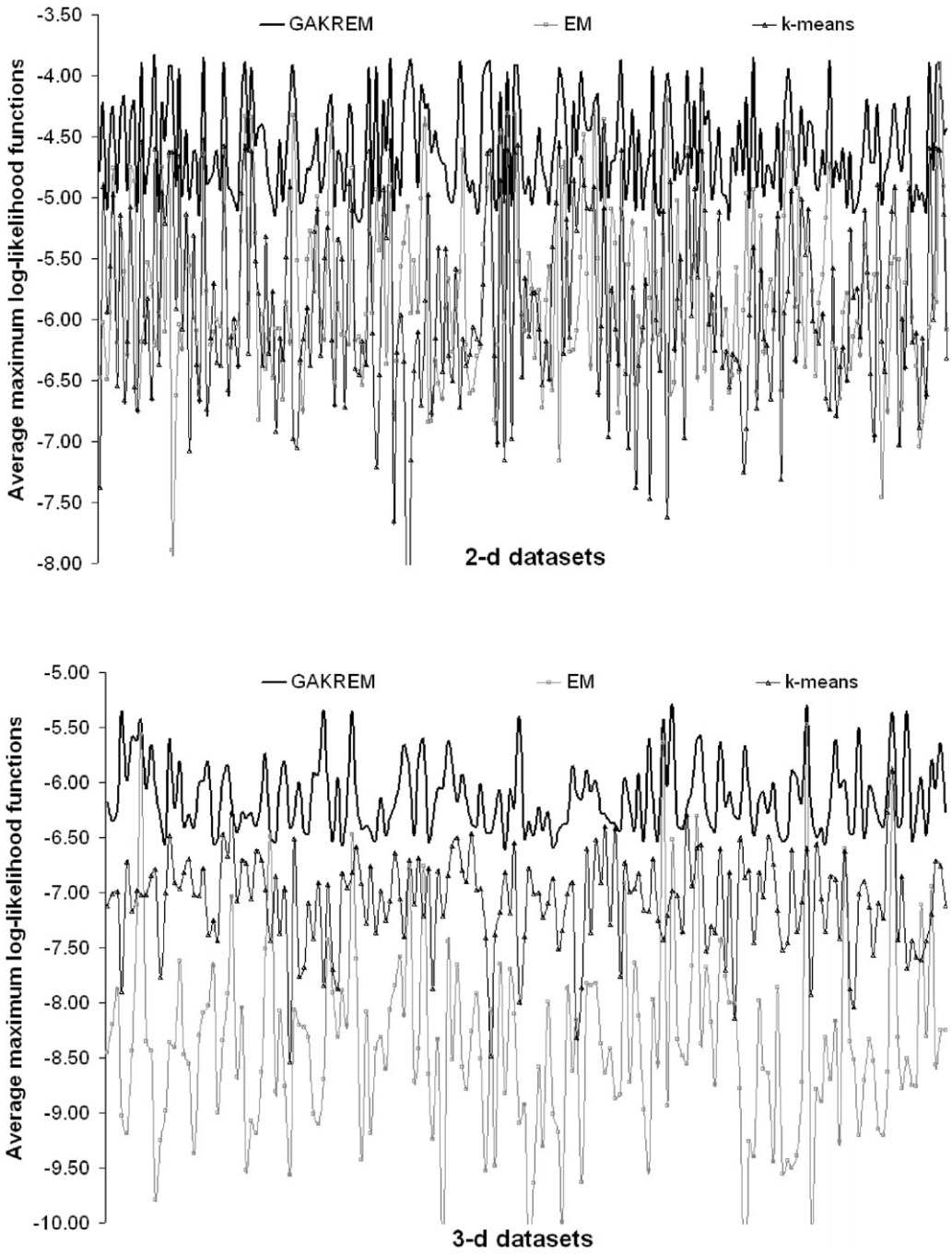



**Fig. 10.** Comparison of average maximum log-likelihood functions of GAKREM, EM, and *K*-means over 1000 datasets.

Note that the Rand index lies between 0 and 1 and the higher value of the Rand index, the better agreement between assigned classes and output clusters. The results in Table 1 show that GAKREM method outperforms the LCV technique in terms of Rand indices in all experiments.

*3.2.3. GAKREM versus EM and K-means algorithms*

As noted, both the conventional EM and *K*-means require the number of clusters specified a priori, thus we compare the performance of GAKREM, EM and *K*-means based on the average maximum log-likelihood functions and Rand indices in all trials. Fig. 10 illustrates the performance of GAKREM, EM and *K*-means in 100 trials over 2, 3, 4 and 5 dimensional datasets,

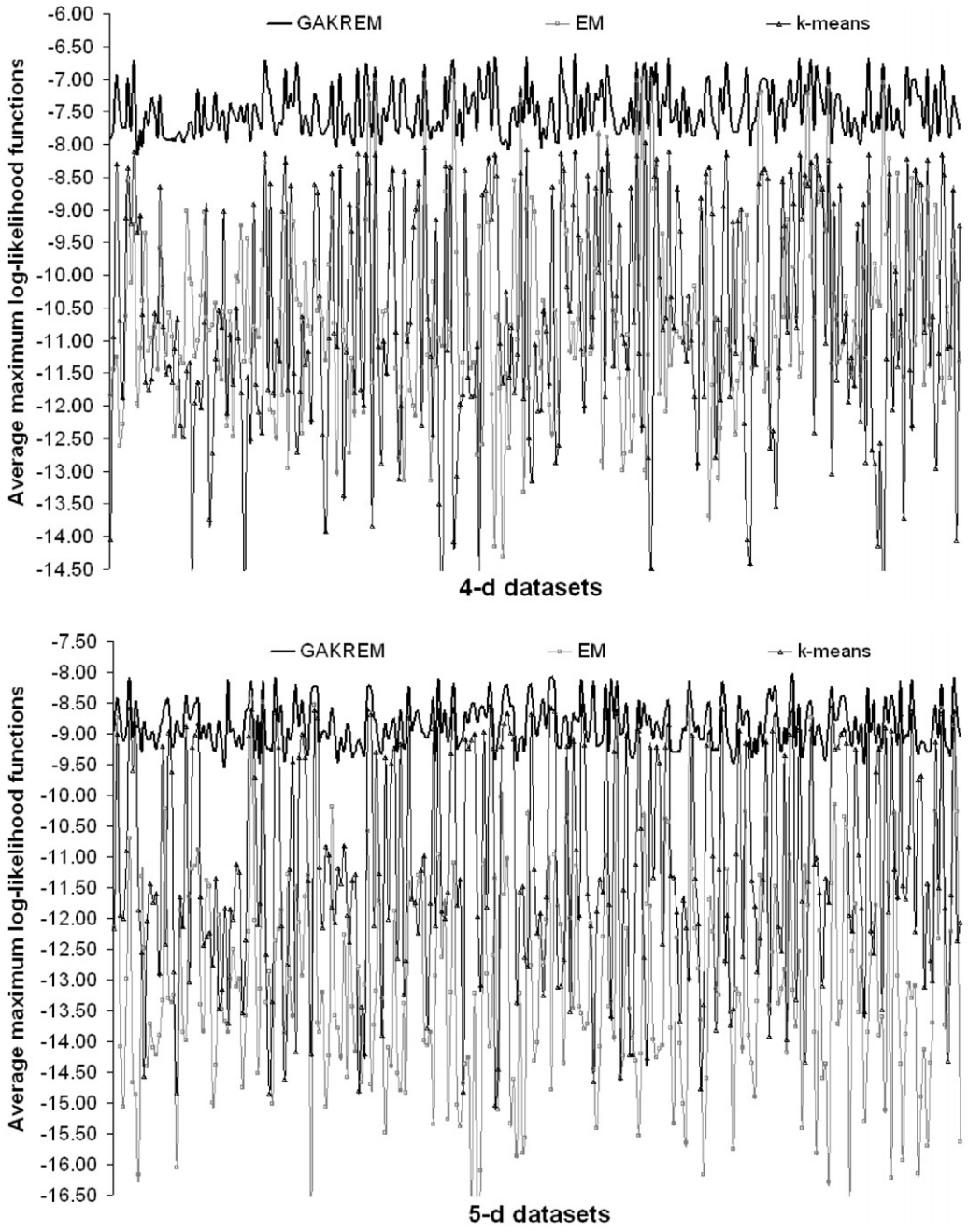



**Fig. 10** (*continued*)

**Table 2**
Comparison of average Rand indices of GAKREM, EM and *K*-means over 1000 datasets (standard deviation value)

| # Dimensions in dataset | Rand index | | |
|---|---|---|---|
| | GAKREM | *K*-means | EM |
| 2 | 0.95 (±0.01) | 0.82 (±0.12) | 0.81 (±0.12) |
| 3 | 0.94 (±0.02) | 0.81 (±0.13) | 0.82 (±0.11) |
| 4 | 0.91 (±0.01) | 0.80 (±0.11) | 0.81 (±0.10) |
| 5 | 0.90 (±0.01) | 0.75 (±0.15) | 0.77 (±0.13) |

respectively. The average maximum log-likelihood values of GAKREM are significantly better than those of EM and *K*-means in all tests.

In terms of Rand indices, Table 2 shows that there is a better agreement between the assigned class and the predicted clusters for GAKREM than for the *K*-means or EM methods.

## 4. Conclusions

The paper introduced a powerful new clustering algorithm – GAKREM – able to perform clustering without the need to a priori specify the number of clusters in the data. GAKREM is a hybrid of several methods such as genetic algorithms, *K*-means, logarithmic regression, and expectation maximization. We have shown that GAKREM is hardly sensitive to initialization of parameters and thus instead of finding a local optimum might find a global one. The hybrid approach presented in the paper can be extended to any type of mixture models that use EM and *K*-means for parameter estimation. We tested GAKREM extensively and showed that it performed better than EM, *K*-means and LCV algorithms on the same datasets.

## Acknowledgements

The authors acknowledge support from the Vietnamese Ministry of Education and Training (Cao Nguyen) and anonymous reviewers for their comments that helped to improve this manuscript.

## References

[1] V. Alves, R. Campello, E. Hruschka, A fuzzy variant of an evolutionary algorithm for clustering, IEEE International Conference on Fuzzy Systems 1 (2007) 375–380.
[2] S. Bandyopadhyay, U. Maulik, Genetic clustering for automatic evolution of clusters and application to image classification, Pattern Recognition 35 (2002) 1197–1208.
[3] J. Bezdek, Pattern Recognition with Fuzzy Objective Function Algorithms, Plenum Press, New York, NY, 1981.
[4] J. Bilmes, A gentle tutorial on the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models, Technical Report ICSI-TR-97-021, International Computer Science Institute (ICSI), Berkeley, CA, 1997.
[5] C. Bishop, Neural Networks for Pattern Recognition, Clarendon Press, Oxford, 1995.
[6] K.J. Cios, W. Pedrycz, R. Swiniarski, L. Kurgan, Data Mining: A Knowledge Discovery Approach, Springer, 2007.
[7] M. Cowgill, R. Harvey, L. Watson, A genetic algorithm approach to cluster analysis, Computational Mathematics with Applications 37 (1999) 99–108.
[8] L. Davis, Handbook of Genetic Algorithms, Van Nostrand Reinhold, 1991.
[9] A. Dempster, N. Laird, D. Rubin, Maximum likelihood from incomplete data via the EM algorithm, Journal of the Royal Statistical Society, Series B 39 (1977) 1–38.
[10] D. Dvorkin, V. Fadok, K.J. Cios, SiMCAL1 algorithm for analysis of gene expression data related to the phosphatidylserine receptor, Artificial Intelligence in Medicine 35 (2005) 49–60.
[11] V. Estivill-Castro, A. Murray, Spatial clustering for data mining with genetic algorithms, in: Proceedings of the International ICSC Symposium on Engineering of Intelligent Systems, 1997, pp. 317–323.
[12] E. Falkenauer, Genetic Algorithms and Grouping Problems, John Wiley & Sons, 1998.
[13] J. Handl, J. Knowles, An evolutionary approach to multiobjective clustering, IEEE Transactions on Evolutionary Computation 11 (1) (2007) 56–76.
[14] J. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, 1975.
[15] E. Hruschka, R. Campello, L. de Castro, Evolving clusters in gene-expression data, Information Sciences 176 (13) (2006) 1898–1927.
[16] A. Jain, R. Dubes, Algorithms for Clustering Data, Prentice-Hall Advanced Reference Series, Prentice-Hall, Inc., Upper Saddle River, NJ, 1988.
[17] A. Jain, N. Murty, J. Flynn, Data clustering: a review, ACM Computing Surveys 31 (1999) 264–323.
[18] T. Kohonen, Self-Organization and Associative Memory, Springer-Verlag, New York, NY, 1989.
[19] S. Lu, K. Fu, A sentence-to-sentence clustering procedure for pattern analysis, IEEE Transactions on Systems, Man and Cybernetics 8 (1978) 381–389.
[20] C. Lucasius, A. Dane, G. Kateman, On *k*-medoid clustering of large data sets with the aid of a genetic algorithm: background, feasibility and comparison, Analytica Chimica Acta 282 (1993) 647–669.
[21] G. MacLachlan, K. Basford, Mixture Models: Inference and Applications to Clustering, Marcel Decker, New York, 1998.
[22] J. McQueen, Some methods for classification and analysis of multivariate observations, Proceedings of Fifth Berkeley Symposium on Mathematical Statistics and Probability,, vol. 1, University of California Press, Berkeley, 1967, pp. 281–297.
[23] W. Rand, Objective criteria for the evaluation of clustering methods, Journal of the American Statistical Association 66 (1971) 846–850.
[24] R. Redner, H. Walker, Mixture densities, maximum likelihood, and the EM algorithm, SIAM Review 26 (1984) 195–239.
[25] J. Rissanen, Modeling by shortest data description, Automatica 14 (1978) 465–471.
[26] E. Ruspini, A new approach to clustering, Information Control 15 (1969) 22–32.
[27] G. Schwarz, Estimating the dimension of a model, Annals of Statistics 6 (1978) 461–464.
[28] P. Smyth, Model selection for probabilistic clustering using cross-validated likelihood, ICS Tech Report 98-09, Statistics and Computing, 1998.
[29] L. Tan, D. Taniar, Adaptive estimated maximum-entropy distribution model, Information Sciences 177 (15) (2007) 3110–3128.
[30] L. Tseng, S. Yang, A genetic approach to the automatic clustering problem, Pattern Recognition 34 (2) (2001) 415–424.
[31] J. Ward, Hierarchical grouping to optimize an objective function, Journal of the American Statistical Association 58 (1963) 236–244.
[32] I. Witten, E. Frank, Data Mining: Practical Machine Learning Tools and Techniques, Morgan Kaufman, 2005.
[33] C. Wu, On the convergence properties of the EM algorithm, Annals of Statistics 11 (1983) 95–103.
[34] L. Xu, M. Jordan, On convergence properties of the EM algorithm for Gaussian mixtures, Neural Computation 8 (1996) 129–151.
[35] C. Zahn, Graph-theoretical methods for detecting and describing gestalt clusters, IEEE Transaction on Computers C-20 (1971) 68–86.