



Available online at www.sciencedirect.com



ELSEVIER

Pattern Recognition Letters 29 (2008) 688–699

Pattern Recognition Letters

www.elsevier.com/locate/patrec

Automatic kernel clustering with a Multi-Elitist Particle Swarm Optimization Algorithm

Swagatam Das^{a,*}, Ajith Abraham^b, Amit Konar^a

^a Department of Electronics and Telecommunication Engineering, Jadavpur University, Kolkata 700 032, India

^b Center of Excellence for Quantifiable Quality of Service (Q2S), Norwegian University of Science and Technology, Trondheim, Norway

Received 3 February 2007; received in revised form 18 August 2007

Available online 15 December 2007

Communicated by W. Pedrycz

Abstract

This article introduces a scheme for clustering complex and linearly non-separable datasets, without any prior knowledge of the number of naturally occurring groups in the data. The proposed method is based on a modified version of classical Particle Swarm Optimization (PSO) algorithm, known as the Multi-Elitist PSO (MEPSO) model. It also employs a kernel-induced similarity measure instead of the conventional sum-of-squares distance. Use of the kernel function makes it possible to cluster data that is linearly non-separable in the original input space into homogeneous groups in a transformed high-dimensional feature space. A new particle representation scheme has been adopted for selecting the optimal number of clusters from several possible choices. The performance of the proposed method has been extensively compared with a few state of the art clustering techniques over a test suit of several artificial and real life datasets. Based on the computer simulations, some empirical guidelines have been provided for selecting the suitable parameters of the PSO algorithm.

© 2007 Elsevier B.V. All rights reserved.

Keywords: Particle Swarm Optimization; Kernel; Clustering; Validity index; Genetic algorithm

1. Introduction

Clustering means the act of partitioning an unlabeled dataset into groups of similar objects. Each group, called a ‘cluster’, consists of objects that are similar between themselves and dissimilar to objects of other groups. In the past few decades, cluster analysis has played a central role in diverse domains of science and engineering (Evangelou et al., 2001; Lillesand and Keifer, 1994; Andrews, 1972; Rao, 1971; Duda and Hart, 1973; Fukunaga, 1990; Everitt, 1993; Hartigan, 1975).

Data clustering algorithms can be *hierarchical* or *partitional* (Frigui and Krishnapuram, 1999; Leung et al., 2000). In hierarchical clustering, the output is a tree show-

ing a sequence of clustering with each cluster being a partition of the data set (Leung et al., 2000). Partitional clustering algorithms, on the other hand, attempts to decompose the data set directly into a set of disjoint clusters. They try to optimize certain criteria (e.g. a squared-error function). The criterion function may emphasize the local structure of the data, as by assigning clusters to peaks in the probability density function, or the global structure. Clustering can also be performed in two different modes: crisp and fuzzy. In crisp clustering, the clusters are disjoint and non-overlapping in nature. Any pattern may belong to one and only one class in this case. In case of fuzzy clustering, a pattern may belong to all the classes with a certain fuzzy membership grade (Jain et al., 1999). A comprehensive survey of the various clustering algorithms can be found in Jain et al. (1999).

The problem of partitional clustering has been approached from diverse fields of knowledge like statistics

* Corresponding author.

E-mail addresses: swagatamdas19@yahoo.co.in (S. Das), ajith.abraham@ieee.org (A. Abraham), konaramit@yahoo.co.in (A. Konar).

(multivariate analysis) (Forgy, 1965), graph theory (Zahn, 1971), expectation maximization algorithms (Mitchell, 1997), artificial neural networks (Mao and Jain, 1995; Pal et al., 1993; Kohonen, 1995), evolutionary computing (Falkenauer, 1998; Paterlini and Minerva, 2003; Murthy and Chowdhury, 1996; Bandyopadhyay and Maulik, 2002), swarm intelligence (Paterlini and Krink, 2006; Omran et al., 2005; Kanade and Hall, 2003) and so on.

The Euclidean distance metric, employed by most of the existing partitional clustering algorithms, work well with datasets in which the natural clusters are nearly hyperspherical and linearly separable (like the artificial dataset 1 used in this paper). But it causes severe misclassifications when the dataset is complex, with linearly non-separable patterns (like the synthetic datasets 2, 3 and 4 described in Section 4 of the present paper). We would like to mention here that, most evolutionary algorithms could potentially work with an arbitrary distance function and are not limited to the Euclidean distance.

Moreover, very few works (Bandyopadhyay and Maulik, 2002; Hamerly and Elkan, 2003; Sarkar et al., 1997; Omran et al., 2005) have been undertaken to make an algorithm learn the correct number of clusters ‘ k ’ in a dataset, instead of accepting the same as a user input. Although, the problem of finding an optimal k is quite important from a practical point of view, the research outcome is still unsatisfactory even for some of the benchmark datasets (Rosenberger and Chehdi, 2000).

In the present work, we propose a new approach towards the problem of automatic clustering (without having any prior knowledge of k initially) using a modified version of the PSO algorithm (Kennedy and Eberhart, 1995). Our procedure employs a kernel induced similarity measure instead of the conventional Euclidean distance metric. A kernel function measures the distance between two data points by implicitly mapping them into a high dimensional feature space where the data is linearly separable. Not only does it preserve the inherent structure of groups in the input space, but also simplifies the associated structure of the data patterns (Muller et al., 2001; Girolami, 2002). Several kernel-based learning methods, including the Support Vector Machine (SVM), have recently been shown to perform remarkably in supervised learning (Scholkopf and Smola, 2002; Vapnik, 1998; Zhang and Chen, 2003; Zhang and Rudnicky, 2002). The kernelized versions of the k -means (Forgy, 1965) and the fuzzy c -means (FCM) (Bezdek, 1981) algorithms reported in Zhang and Rudnicky (2002) and Zhang and Chen (2003) respectively, have reportedly outperformed their original counterparts over several test cases.

Now, we may summarize the new contributions made in the paper as follows:

- (i) Firstly, we develop an alternative framework for learning the number of partitions in a dataset besides the simultaneous refining of the clusters, through one shot of optimization.

- (ii) We propose a new version of the PSO algorithm based on the multi-elitist strategy, well-known in the field of evolutionary algorithms. Our experiments indicate that the proposed MEPSO algorithm yields more accurate results at a faster pace than the classical PSO in context to the present problem.
- (iii) We reformulate a recently proposed cluster validity index (known as the *CS measure*) (Chou et al., 2004) using the kernelized distance metric. The new CS measure forms the objective function to be minimized for optimal clustering.

We have undertaken extensive performance comparisons in order to establish the effectiveness of the proposed method in detecting clusters from several synthetic as well as real world datasets. Some empirical guidelines for choosing the parameters of the MEPSO based clustering algorithm has been provided. Effect of the growth of feature-space dimensionality on the performance of the algorithm was also studied based on the real life datasets. The rest of the paper is organised as follows: Section 2 briefly describes the clustering problem, the kernel distance metric and the reformulation of the CS measure. In Section 3, we briefly outline the classical PSO and then introduce the MEPSO algorithm. Section 4 describes the novel procedure for automatic clustering with MEPSO. Experimental results are presented and discussed in Section 5. Finally the paper is concluded in Section 6.

2. Kernel based clustering and corresponding validity index

2.1. The crisp clustering problem

Let $X = \{\vec{x}_1, \vec{x}_2, \dots, \vec{x}_n\}$ be a set of n unlabeled patterns in the d -dimensional input space. Here, each element $x_{i,j}$ in the i th vector \vec{x}_i corresponds to the j th real valued feature ($j = 1, 2, \dots, d$) of the i th pattern ($i = 1, 2, \dots, n$). Given such a set, the partitional clustering algorithm tries to find a partition $C = \{C_1, C_2, \dots, C_k\}$ of k classes, such that the similarity of the patterns in the same cluster is maximum and patterns from different clusters differ as far as possible. The partitions should maintain the following properties:

- (1) $C_i \neq \emptyset \forall i \in \{1, 2, \dots, k\}$.
- (2) $C_i \cap C_j = \emptyset \forall i \neq j$ and $i, j \in \{1, 2, \dots, k\}$.
- (3) $\bigcup_{i=1}^k C_i = P$.

The most popular way to evaluate similarity between two patterns amounts to the use of the Euclidean distance, which between any two d -dimensional patterns \vec{x}_i and \vec{x}_j is given by

$$d(\vec{x}_i, \vec{x}_j) = \sqrt{\sum_{p=1}^d (x_{i,p} - x_{j,p})^2} = \|\vec{x}_i - \vec{x}_j\|. \quad (1)$$

2.2. The kernel based similarity measure

Given a dataset X in the d -dimensional real space \Re^d , let us consider a non-linear mapping function from the input space to a high dimensional feature space H :

$$\varphi : \Re^d \rightarrow H, \quad \vec{x}_i \rightarrow \varphi(\vec{x}_i), \quad (2)$$

where $\vec{x}_i = [x_{i,1}, x_{i,2}, \dots, x_{i,d}]^\top$ and

$$\varphi(\vec{x}_i) = [\varphi_1(\vec{x}_i), \varphi_2(\vec{x}_i), \dots, \varphi_H(\vec{x}_i)]^\top.$$

By applying the mapping, a dot product $\vec{x}_i^\top \cdot \vec{x}_j$ is transformed into $\varphi^\top(\vec{x}_i) \cdot \varphi(\vec{x}_j)$. Now, the central idea in kernel-based learning is that the mapping function φ need not be explicitly specified. The dot product $\varphi^\top(\vec{x}_i) \cdot \varphi(\vec{x}_j)$ in the transformed space can be calculated through the kernel function $K(\vec{x}_i, \vec{x}_j)$ in the input space \Re^d . Consider the following simple example:

Example 1. Let $d = 2$ and $H = 3$ and consider the following mapping: $\varphi : \Re^2 \rightarrow H = \Re^3$, and $[x_{i,1}, x_{i,2}]^\top \rightarrow [x_{i,1}^2, \sqrt{2} \cdot x_{i,1}x_{i,2}, x_{i,2}^2]^\top$. Now the dot product in feature space H :

$$\begin{aligned} \varphi^\top(\vec{x}_i) \cdot \varphi(\vec{x}_j) &= [x_{i,1}^2, \sqrt{2} \cdot x_{i,1}x_{i,2}, x_{i,2}^2] \cdot [x_{j,1}^2, \sqrt{2} \cdot x_{j,1}x_{j,2}, x_{j,2}^2]^\top \\ &= [x_{i,1} \cdot x_{j,1} + x_{i,2} \cdot x_{j,2}]^2 = [\vec{x}_i^\top \cdot \vec{x}_j]^2 = K(\vec{x}_i, \vec{x}_j). \end{aligned}$$

Clearly the simple kernel function K is the square of the dot product of vectors \vec{x}_i and \vec{x}_j in \Re^d .

Hence, the kernelized distance measure between two patterns \vec{x}_i and \vec{x}_j is given by

$$\begin{aligned} \|\varphi(\vec{x}_i) - \varphi(\vec{x}_j)\|^2 &= (\varphi(\vec{x}_i) - \varphi(\vec{x}_j))^\top (\varphi(\vec{x}_i) - \varphi(\vec{x}_j)) \\ &= \varphi^\top(\vec{x}_i) \cdot \varphi(\vec{x}_i) - 2 \cdot \varphi^\top(\vec{x}_i) \cdot \varphi(\vec{x}_j) \\ &\quad + \varphi^\top(\vec{x}_j) \cdot \varphi(\vec{x}_j) = K(\vec{x}_i, \vec{x}_i) - 2 \cdot K(\vec{x}_i, \vec{x}_j) \\ &\quad + K(\vec{x}_j, \vec{x}_j). \end{aligned} \quad (3)$$

Among the various kernel functions used in literature, in the present context, we have chosen the well-known Gaussian kernel (also referred to as the Radial Basis Function) owing to its better classification accuracy over the linear and polynomial kernels on many test problems (Girolami, 2002; Pirooznia et al.; Hertz et al., 2006; Dunn, 1974). The Gaussian Kernel may be represented as

$$K(\vec{x}_i, \vec{x}_j) = \exp\left(-\frac{\|\vec{x}_i - \vec{x}_j\|^2}{2\sigma^2}\right), \quad (4)$$

where $\sigma > 0$. Clearly, for Gaussian kernel, $K(\vec{x}_i, \vec{x}_i) = 1$ and thus relation (3) reduces to

$$\|\varphi(\vec{x}_i) - \varphi(\vec{x}_j)\|^2 = 2 \cdot (1 - K(\vec{x}_i, \vec{x}_j)). \quad (5)$$

2.3. Reformulation of CS measure

Cluster validity indices correspond to the statistical-mathematical functions used to evaluate the results of a clustering algorithm on a quantitative basis. For crisp clustering, some of the well-known indices available in the

literature are the Dunn's index (DI) (Hertz et al., 2006; Dunn, 1974), Calinski-Harabasz index (Calinski and Harabasz, 1974), Davis-Bouldin (DB) index (Davies and Bouldin, 1979), PBM index (Pakhira et al., 2004), and the CS measure (Chou et al., 2004). In this work, we have based our fitness function on the CS measure as according to the authors, CS measure is more efficient in tackling clusters of different densities and/or sizes than the other popular validity measures, the price being paid in terms of high computational load with increasing k and n (Chou et al., 2004). Before applying the CS measure, centroid of a cluster is computed by averaging the data vectors belonging to that cluster using the formula,

$$\vec{m}_i = \frac{1}{N_i} \sum_{x_j \in C_i} \vec{x}_j. \quad (6)$$

A distance metric between any two data points \vec{x}_i and \vec{x}_j is denoted by $d(\vec{x}_i, \vec{x}_j)$. Then the CS measure can be defined as

$$\begin{aligned} \text{CS}(k) &= \frac{\frac{1}{k} \sum_{i=1}^k \left[\frac{1}{N_i} \sum_{\vec{x}_i \in C_i} \max_{\vec{x}_q \in C_i} \{d(\vec{x}_i, \vec{x}_q)\} \right]}{\frac{1}{k} \sum_{i=1}^k [\min_{j \in K, j \neq i} \{d(\vec{m}_i, \vec{m}_j)\}]} \\ &= \frac{\sum_{i=1}^k \left[\frac{1}{N_i} \sum_{\vec{x}_i \in C_i} \max_{\vec{x}_q \in C_i} \{d(\vec{x}_i, \vec{x}_q)\} \right]}{\sum_{i=1}^k [\min_{j \in K, j \neq i} \{d(\vec{m}_i, \vec{m}_j)\}]. \end{aligned} \quad (7)$$

Now, using a Gaussian kernelized distance measure and transforming to the high dimensional feature space, the CS measure reduces to (using relation (5)):

$$\begin{aligned} \text{CS}_{\text{kernel}}(k) &= \frac{\sum_{i=1}^k \left[\frac{1}{N_i} \sum_{\vec{x}_i \in C_i} \max_{\vec{x}_q \in C_i} \{\|\varphi(\vec{x}_i) - \varphi(\vec{x}_q)\|^2\} \right]}{\sum_{i=1}^k [\min_{j \in k, j \neq i} \{\|\varphi(\vec{m}_i) - \varphi(\vec{m}_j)\|^2\}]} \\ &= \frac{\sum_{i=1}^k \left[\frac{1}{N_i} \sum_{\vec{x}_i \in C_i} \max_{\vec{x}_q \in C_i} \{2(1 - K(\vec{x}_i, \vec{x}_q))\} \right]}{\sum_{i=1}^k [\min_{j \in k, j \neq i} \{2(1 - K(\vec{m}_i, \vec{m}_j))\}]} \end{aligned} \quad (8)$$

The minimum value of this CS measure indicates an optimal partition of the dataset. The value of ' k ' which minimizes $\text{CS}_{\text{kernel}}(k)$ therefore gives the appropriate number of clusters in the dataset.

3. The Multi-Elitist PSO (MEPSO) algorithm

The concept of Particle Swarms, although initially introduced for simulating human social behaviors, has become very popular these days as an efficient search and optimization technique. In PSO, a population of conceptual 'particles' is initialized with random positions \vec{Z}_i and velocities \vec{V}_i , and a function, f , is evaluated, using the particle's positional coordinates as input values. In an n -dimensional search space, $\vec{Z}_i = (Z_{i1}, Z_{i2}, Z_{i3}, \dots, Z_{in})$ and $\vec{V}_i = (V_{i1}, V_{i2}, V_{i3}, \dots, V_{in})$. Positions and velocities are adjusted, and the function is evaluated with the new coordinates at each time-step. The basic update equations for the d th dimension of the i th particle in PSO may be given as

$$\left. \begin{aligned} V_{id}(t+1) &= \omega \cdot V_{id}(t) + C_1 \cdot \phi_1 \cdot (P_{lid} - X_{id}(t)) \\ &\quad + C_2 \cdot \phi_2 \cdot (P_{gd} - X_{id}(t)), \\ X_{id}(t+1) &= X_{id}(t) + V_{id}(t+1). \end{aligned} \right\} \quad (9)$$

The variables ϕ_1 and ϕ_2 are random positive numbers, drawn from a uniform distribution and defined by an upper limit ϕ_{\max} , which is a parameter of the system. C_1 and C_2 are called acceleration coefficients whereas ω is called inertia weight. \vec{P}_i is the local best solution found so far by the i th particle, while \vec{P}_g represents the positional coordinates of the fittest particle found so far in the entire community or in some neighborhood of the current particle. Once the iterations are terminated, most of the particles are expected to converge to a small radius surrounding the global optima of the search space.

The canonical PSO has been subjected to empirical and theoretical investigations by several researchers (Eberhart and Shi, 2001; Clerc and Kennedy, 2002). In many occasions, the convergence is premature, especially if the swarm uses a small inertia weight ω or constriction coefficient (Eberhart and Shi, 2001). As the global best found early in the searching process may be a poor local minima, we propose a multi-elitist strategy for searching the global best of the PSO. We call the new variant of PSO the MEPSO. The idea draws inspiration from the works reported in Deb et al. (2002). We define a growth rate β for each particle. When the fitness value of a particle at the t th iteration is higher than that of a particle at the $(t - 1)$ th iteration, the β will be increased. After the local best of all particles are decided in each generation, we move the local best, which has higher fitness value than the global best into the candidate area. Then the global best will be replaced by the local best with the highest growth rate β . The elitist concept can prevent the swarm from tending to the global best too early in the searching process. The MEPSO follows the *g_best* PSO topology in which the entire swarm is treated as a single neighborhood. The pseudo code about MEPSO is as follows:

Procedure MEPSO

For $t \equiv 1$ to t_{\max}

For $i \equiv 1$ to N // swarm size is N

If (the fitness value of particle_j in *t*th time-step > that of particle *j* in (*t* - 1)th time-step)
 $\beta_j(t) = \beta_i(t-1) + 1;$

End

Update Local best j .

If (the fitness of Local best j > that of Global best now)

Choose Local best i put into candidate area.

End

End
Calculate β of every candidate, and record the candidate of β_{\max} .

Update the Global best to become the candidate of β_{\max} .

Else

Update the Global best to become the particle of highest fitness value.

End

End

4. The automatic clustering algorithm

4.1. Particle representation

In the proposed method, for n data points, each p -dimensional, and for a user-specified maximum number of clusters k_{\max} , a particle is a vector of real numbers of dimension $k_{\max} + k_{\max} \times p$. The first k_{\max} entries are positive floating-point numbers in $(0, 1)$, each of which determines whether the corresponding cluster is to be activated (i.e. to be really used for classifying the data) or not. The remaining entries are reserved for k_{\max} cluster centers, each p -dimensional. A single particle can be shown as

The diagram shows the expression $Z_i(t) =$ followed by two horizontal brackets under a row of boxes. The first bracket, labeled "Activation Threshold", covers the first four boxes containing $T_{i,1}$, $T_{i,2}$, \dots , and $T_{i,k_{\max}}$. The second bracket, labeled "Cluster Centroids", covers the last three boxes containing $\vec{m}_{i,1}$, $\vec{m}_{i,2}$, and \dots , $\vec{m}_{i,k_{\max}}$.

The j th cluster center in the i th particle is active or selected for partitioning the associated dataset if $T_{i,j} > 0.5$. On the other hand, if $T_{i,j} < 0.5$, the particular j th cluster is inactive in the i th particle. Thus the $T_{i,j}$ s behave like control genes (we call them *activation thresholds*) in the particle governing the selection of the active cluster centers. The rule for selecting the actual number of clusters specified by one chromosome is

IF $T_{i,j} > 0.5$ **THEN** the j th cluster center $\vec{m}_{i,j}$ is **ACTIVE**
ELSE $\vec{m}_{i,j}$ is **INACTIVE**

Consider the following example:

Example 2. Positional coordinates of one particular particle is illustrated below. There are at most five 3-dimensional cluster centers among which, according to the rule presented in (10) the second (6, 4.4, 7), third (5.3, 4.2, 5) and fifth one (8, 4, 4) have been activated for partitioning the dataset and marked in bold. The quality of the partition yielded by such a particle can be judged by an appropriate cluster validity index.

During the PSO iterations, if some threshold T in a particle exceeds 1 or becomes negative, it is fixed to 1 or zero, respectively. However, if it is found that no flag could be set to one in a particle (all activation thresholds are smaller than 0.5), we randomly select 2 thresholds and re-initialize them to a random value between 0.5 and 1.0. Thus the minimum number of possible clusters is 2.

4.2. The fitness function

One advantage of the proposed algorithm is that it can use any suitable validity index as its fitness function. We have used the kernelized CS measure as the basis of our fitness function, which for i th particle can be described as

$$f_i = \frac{1}{CS_{\text{kernel}_i}(k) + eps}, \quad (11)$$

where eps is a very small constant (we used 0.0002). Maximization of f_i implies a minimization of the kernelized CS measure leading to the optimal partitioning of the dataset.

4.3. Avoiding erroneous particles with empty clusters or unreasonable fitness evaluation

There is a possibility that in our scheme, during computation of the kernelized CS index, a division by zero may be encountered. For example, the positive infinity (such as 4.0/0.0) or the not-a-number (such as 0.0/0.0) condition always occurs when one of the selected cluster centers is outside the boundary of distributions of data set as far as possible. To avoid this problem we first check to see if in any particle, any cluster has fewer than 2 data points in it. If so, the cluster center positions of this special particle are re-initialized by an average computation. If k clusters ($2 < k < k_{\max}$) are selected for this particle, we put n/k data points for every individual activated cluster center, such that a data point goes with a center that is nearest to it.

4.4. Putting it altogether

Step 1: Initialize each particle to contain k number of randomly selected cluster centers and k (randomly chosen) activation thresholds in $[0, 1]$.

Step 2: Find out the active cluster centers in each particle with the help of the rule described in (10).

Step 3: For $t = 1$ to t_{\max} do

- (i) For each data vector \vec{X}_p , calculate its distance metric $d(\vec{X}_p, \vec{m}_{i,j})$ from all active cluster centers of the i th particle \vec{V}_i .
- (ii) Assign \vec{X}_p to that particular cluster center $\vec{m}_{i,j}$ where $d(\vec{X}_p, \vec{m}_{i,j}) = \min_{\forall b \in \{1, 2, \dots, k\}} \{d(\vec{X}_p, \vec{m}_{i,b})\}$.
- (iii) Check if the number of data points belonging to any cluster center $m_{i,j}$ is less than 2. If so, update the cluster centers of the particle using the concept of average described earlier.

- (iv) Change the population members according to the MEPSO algorithm. Use the fitness of the particles to guide the dynamics of the swarm.

Step 4: Report as the final solution the cluster centers and the partition obtained by the globally best particle (one yielding the highest value of the fitness function) at time $t = t_{\max}$.

5. Experimental results

5.1. General comparison with other clustering algorithms

To test the effectiveness of the proposed method, we compare its performance with six other clustering algorithms using a test-bed of five artificial and three real world datasets. Among the other algorithms considered, there are two recently developed automatic clustering algorithms well-known as the GCUK (Genetic Clustering with an Unknown number of clusters K) (Bandyopadhyay and Maulik, 2002) and the DCPSO (Dynamic Clustering PSO) (Omran et al., 2005). Moreover, in order to investigate the effects of the changes made in the classical g_best PSO algorithm, we have compared MEPSO with an ordinary PSO based kernel-clustering method that uses the same particle representation scheme and fitness function as the MEPSO. Both the algorithms were let run on the same initial populations. The other algorithms are the kernel k -means algorithm (Girolami, 2002; Zhang and Chen, 2003) and a kernelized version of the subtractive clustering (Kim et al., 2005). Both the algorithms were provided with the correct number of clusters as they are non-automatic.

We used datasets with a wide variety in the number and shape of clusters, number of datapoints and the count of features of each datapoint. The real life datasets used here are the Glass, the Wisconsin breast cancer, the image segmentation, the Japanese Vowel and the automobile (Blake et al., 1998). The synthetic datasets included here, comes with linearly non-separable clusters of different shapes (like elliptical, concentric circular dish and shell, rectangular, etc.). Brief details of the datasets have been provided in Table 1. Scatterplot of the synthetic datasets have also been shown in Fig. 1. The clustering results were judged using Huang's accuracy measure (Huang and Ng, 1999):

$$r = \frac{\sum_{i=1}^k n_i}{n}, \quad (12)$$

where n_i is the number of data occurring in both the i th cluster and its corresponding true cluster, and n is the total number of data points in the data set. According to this measure, a higher value of r indicates a better clustering result, with perfect clustering yielding a value of $r = 1$.

We used $\sigma = 1.1$ for all the artificial datasets, $\sigma = 0.9$ for breast cancer dataset and $\sigma = 2.0$ for the rest of the real life datasets for the RBF kernel following (Camastra and Verri, 2005). In these experiments, the kernel k -means was run 100 times with the initial centroids randomly

Table 1
Description of the datasets

Dataset	Number of datapoints (n)	Number of clusters (k)	Data-dimension (d)
Synthetic_1	500	2	2
Synthetic_2	52	2	2
Synthetic_3	400	4	3
Synthetic_4	250	5	2
Synthetic_5	600	2	2
Glass	214	6	9
Wine	178	3	13
Breast cancer	683	2	9
Image segmentation	2310	7	19
Japanese vowel	640	9	12

selected from the data set. A termination criterion of $\varepsilon = 0.001$. The parameters of the kernel-based subtractive methods were set to $\alpha = 5.4$ and $\beta = 1.5$ as suggested by Pal and Chakraborty (2000). For all the competitive algorithms, we have selected their best parameter settings as reported in the corresponding literatures. The control parameters for MEPSO where chosen after experimenting with several possible values. Some of the experiments focussing on the effects of parameter-tuning in MEPSO has been reported in the next subsection. The same set of parameters were used for all the test problems for all the algorithms. These parameter settings have been reported in Table 2.

Table 3 compares the algorithms on the quality of the optimum solution as judged by the Huang's measure. The mean and the standard deviation (within parentheses) for 40 independent runs (with different seeds for the random number generator) of each of the six algorithms are presented in Table 3. Missing values of standard deviation in this table indicate a zero standard deviation. The best solution in each case has been shown in bold. Table 4 shows results of unpaired t -tests between the better of the new algorithm (MEPSO) and the best of the other five in each case (standard error of difference of the two means, 95% confidence interval of this difference, the t value, and the two-tailed P value). Tables 5 and 6 present the mean and standard deviation of the number of classes found by the three automatic clustering algorithms. In Fig. 1 we present the clustering results on the synthetic datasets by the new MEPSO algorithm (to save space we do not provide results for all the six algorithms).

For comparing the speed of the stochastic algorithms like GA, PSO or DE, we choose *number of fitness function evaluations (FEs)* as a measure of computation time instead of generations or iterations. From the data provided in Table 3, we choose a threshold value of the classification accuracy for each dataset. This threshold value is somewhat larger than the minimum accuracy attained by each automatic clustering algorithm. Now we run an algorithm on each dataset and stop as soon as it achieves the proper number of clusters as well as the threshold accu-

racy. We then note down the number of fitness function evaluations the algorithm takes. A lower number of FEs corresponds to a faster algorithm. The speed comparison results are provided in Table 7. The kernel k -means and the subtractive clustering method are not included in this table, as they are non-automatic and do not employ evolutionary operators as in GCUK and PSO based methods.

A close scrutiny of Tables 3, 5 and 6 reveals that the kernel based MEPSO algorithm performed markedly better as compared to the other competitive clustering algorithms, in terms of both accuracy and convergence speed. We note that in general, the kernel based clustering methods outperform the GCUK or DCPSO algorithms (which do not use the kernelized fitness function) especially on linearly non-separable artificial datasets like synthetic_1, synthetic_2 and synthetic_5. Although the proposed method provided a better clustering result than the other methods for Synthetic_5 dataset, its accuracy for this data was lower than the seven other data sets considered. This indicates that the proposed approach is limited in its ability to classify non-spherical clusters.

The PSO based methods (especially MEPSO) on average took lesser computational time than the GCUK algorithm over most of the datasets. One possible reason of this may be the use of less complicated variation operators (like mutation) in PSO as compared to the operators used for GA.

We also note that the MEPSO performs much better than the classical PSO based kernel-clustering scheme. Since both the algorithms use same particle representation and starts with the same initial population, difference in their performance must be due to the difference in their internal operators and parameter values. This demonstrates the effectiveness of the multi-elitist strategy incorporated in the MEPSO algorithm.

5.2. Choice of parameters for MEPSO algorithm

The MEPSO has a number of control parameters that affect its performance on different clustering problems. In this section we discuss the influence of parameters like swarm size, the inertia factor ω and the acceleration factors C_1 and C_2 on the Kernel_MEPSO algorithm.

(1) *Swarm size*: To investigate the effect of the swarm size, the MEPSO was executed separately with 10–80 particles (keeping all other parameter settings same as reported in Table 2) on all the datasets. In Fig. 2 we plot the convergence behavior of the algorithm (average of 40 runs) on the image segmentation dataset (with 2310 datapoints and 19 features, it is the most complicated synthetic dataset considered in this section) for different population sizes. We omit the other results here to save space. The results reveal that the number of particles more than 40 gives more or less identical accuracy of the final clustering results for MEPSO. This observation is in accordance with Van den Bergh and Engelbrecht,

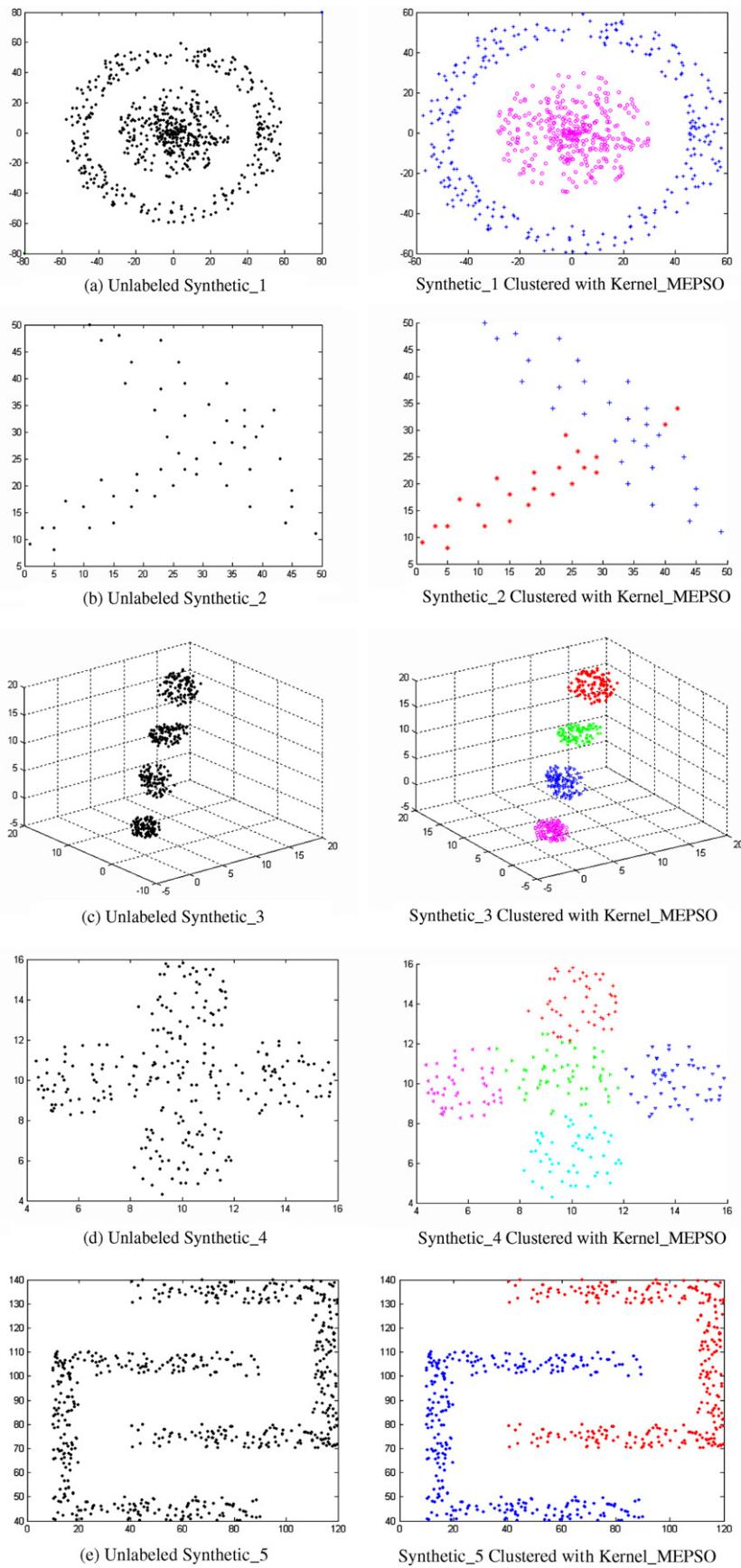


Fig. 1. Two- and three-dimensional synthetic datasets clustered with MEPSO.

Table 2
Parameter settings for different competitor algorithms

GCUK	DCPSO	PSO	MEPSO
Pop_size	70	Pop_size	100
Cross-over probability μ_c	0.85	Inertia weight	0.72
Mutation probability μ_m	0.005	C_1, C_2	1.494
		P_{ini}	0.75
K_{max}	20	K_{max}	20
K_{min}	2	K_{min}	2
		K_{min}	2

Table 3
Mean and standard deviation of the clustering accuracy (%) achieved by each clustering algorithm over 40 independent runs (each run continued up to 50,000 FEs for GCUK, DCPSO, Kernel_PSO and Kernel_MEPSO)

Datasets	Algorithms					
	Kernel k-means	Kernel Sub_clust	GCUK	DC-PSO	Kernel_PSO	Kernel_MEPSO
Synthetic_1	83.45 (0.032)	87.28	54.98 (0.88)	57.84 (0.065)	90.56 (0.581)	99.45 (0.005)
Synthetic_2	71.32 (0.096)	75.73	65.82 (0.146)	59.91 (0.042)	61.41 (0.042)	80.92 (0.0051)
Synthetic_3	89.93 (0.88)	94.03	97.75 (0.632)	97.94 (0.093)	92.94 (0.193)	99.31 (0.001)
Synthetic_4	67.65 (0.104)	80.25	74.30 (0.239)	75.83 (0.033)	78.85 (0.638)	87.84 (0.362)
Synthetic_5	81.23 (0.127)	84.33	54.45 (0.348)	52.55 (0.209)	89.46 (0.472)	99.75 (0.001)
Glass	68.92 (0.032)	73.92	76.27 (0.327)	79.45 (0.221)	70.71 (0.832)	92.01 (0.623)
Wine	73.43 (0.234)	59.36	80.64 (0.621)	85.81 (0.362)	87.65 (0.903)	93.17 (0.002)
Breast cancer	66.84 (0.321)	70.54	73.22 (0.437)	78.19 (0.336)	80.49 (0.342)	86.35 (0.211)
Image segmentation	56.83 (0.641)	70.93	78.84 (0.336)	81.93 (1.933)	84.32 (0.483)	87.72 (0.982)
Japanese vowel	44.89 (0.772)	61.83	70.23 (1.882)	82.57 (0.993)	79.32 (2.303)	84.93 (2.292)
Average	72.28	75.16	74.48	76.49	77.58	91.65

Table 4
Results of unpaired *t*-tests on the data of Table 3

Datasets	Std. Err	<i>t</i>	95% Conf. Intvl	Two-tailed <i>P</i>	Significance
Synthetic_1	0.005	976.36	(−5.01, −4.98)	<0.0001	Extremely significant
Synthetic_2	0.001	9094.7	(−7.19, −7.18)	<0.0001	Extremely significant
Synthetic_3	0.015	129.88	(−1.94, −1.88)	<0.0001	Extremely significant
Synthetic_4	0.057	132.61	(−7.70, −7.48)	<0.0001	Extremely significant
Synthetic_5	0.075	137.88	(10.14, 10.44)	<0.0001	Extremely significant
Glass	0.105	120.17	(−12.77, −12.35)	<0.0001	Extremely significant
Wine	0.057	134.52	(−7.81, −7.58)	<0.0001	Extremely significant
Breast cancer	0.063	130.07	(8.04, 8.28)	<0.0001	Extremely significant
Image segmentation	0.173	19.6495	(3.055, 3.744)	<0.0001	Extremely significant
Japanese vowel	0.395	5.9780	(−3.147, −1.574)	<0.0001	Extremely significant

Table 5
Mean and standard deviation (in parentheses) of the number of clusters found over the synthetic datasets for four automatic clustering algorithms over 40 independent runs

Algorithms	Synthetic_1	Synthetic_2	Synthetic_3	Synthetic_4	Synthetic_5
GCUK	2.50 (0.021)	3.05 (0.118)	4.15 (0.039)	9.85 (0.241)	4.25 (0.921)
DCPSO	2.45 (0.121)	2.80 (0.036)	4.25 (0.051)	9.05 (0.726)	6.05 (0.223)
Ordinary PSO	2.50 (0.026)	2.65 (0.126)	4.10 (0.062)	9.35 (0.335)	2.25 (0.361)
Kernel_MEPSO	2.10 (0.033)	2.15 (0.102)	4.00 (0.00)	10.05 (0.021)	2.05 (0.001)

who in den Bergh and Engelbrecht (2001) showed that though there is a slight improvement in the solution quality with increasing swarm sizes, a larger swarm increases the number of function evaluations to converge to an error limit. For most of the problems, it was found that keeping the swarm size 40 provides a reasonable trade-off between the quality of the final results and the convergence speed.

(2) *The inertia factor ω :* Provided all other parameters are fixed at the values shown in Table 2, the MEPSO was run with several possible choices of the inertia factor ω . Specifically we used a time-varying ω (linearly decreasing from 0.9 to 0.4 following Shi and Eberhart, 1999), random ω (Eberhart and Shi, 2001), $\omega = 0.1$, $\omega = 0.5$ and finally $\omega = 0.794$ (Kennedy and Eberhart, 1995). In Fig. 3, we show how the fitness of the globally

Table 6

Mean and standard deviation (in parentheses) of the number of clusters found over the synthetic datasets for four automatic clustering algorithms over 40 independent runs

Algorithms	Glass	Wine	Breast cancer	Image segmentation	Japanese vowel
GCUK	5.85 (0.035)	4.05 (0.021)	2.25 (0.063)	7.05 (0.008)	9.50 (0.218)
DCPSO	5.60 (0.009)	3.75 (0.827)	2.25 (0.026)	7.50 (0.057)	10.25 (1.002)
Ordinary PSO	5.75 (0.075)	3.00 (0.00)	2.00 (0.00)	7.20 (0.025)	9.25 (0.822)
Kernel_MEPSO	6.05 (0.015)	3.00 (0.00)	2.00 (0.00)	7.00 (0.00)	9.05 (0.021)

Table 7

Mean and standard deviations of the number of fitness function evaluations (over 40 successful runs) required by each algorithm to reach a predefined cut-off value of the classification accuracy

Dataset	Threshold accuracy (in %)	GCUK	DCPSO	Ordinary PSO	Kernel_MEPSO
Synthetic_1	50.00	48000.05 (21.43)	42451.15 (11.57)	43812.30 (2.60)	37029.65 (17.48)
Synthetic_2	55.00	41932.10 (12.66)	45460.25 (20.97)	40438.05 (18.52)	36271.05 (10.41)
Synthetic_3	85.00	40000.35 (4.52)	35621.05 (12.82)	37281.05 (7.91)	32035.55 (4.87)
Synthetic_4	65.00	46473.25 (7.38)	43827.65 (2.75)	42222.50 (2.33)	36029.05 (6.38)
Synthetic_5	50.00	43083.35 (5.30)	39392.05 (7.20)	42322.05 (2.33)	35267.45 (9.11)
Glass	65.00	47625.35 (6.75)	40382.15 (7.27)	38292.25 (10.32)	37627.05 (12.36)
Wine	55.00	44543.70 (44.89)	43425.00 (18.93)	3999.65 (45.90)	35221.15 (67.92)
Breast cancer	65.00	40390.00 (11.45)	37262.65 (13.64)	35872.05 (8.32)	32837.65 (4.26)
Image segmentation	55.00	39029.05 (62.09)	40023.25 (43.92)	35024.35 (101.26)	34923.22 (24.28)
Japanese vowel	40.00	40293.75 (23.33)	28291.60 (121.72)	29014.85 (21.67)	24023.95 (20.62)

best particle (averaged over 40 runs) varies with no. of FEs for the image segmentation dataset over different values of ω . It was observed that for all the problems belonging to the current test suit, best convergence behavior of MEPSO is observed for $\omega = 0.794$.

(3) *The acceleration coefficients C_1 and C_2 :* Provided all other parameters are fixed at the values given in Table 2, we let MEPSO run for different settings of the acceleration coefficients C_1 and C_2 as reported in various literatures on PSO. We used $C_1 = 0.7$ and $C_2 = 1.4$, $C_1 = 1.4$ and $C_2 = 0.7$ (Shi and Eberhart, 1999), $C_1 = C_2 = 1.494$ (Kennedy et al., 2001), $C_1 = C_2 = 2.00$ (Kennedy et al., 2001) and finally a time varying

acceleration coefficients where C_1 linearly increases from 0.35 to 2.4 and C_2 linearly decreases from 2.4 to 0.35 (Ratnaweera and Halgamuge, 2004). We noted that the linearly varying acceleration coefficients gave the best clustering results over all the problems considered. This is perhaps due to the fact that an increasing C_1 and gradually decreasing C_2 boost the global search over the entire search space during the early part of the optimization and encourage the particles to converge to global optima at the end of the search. Fig. 4 illustrates the convergence characteristics of MEPSO over the image segmentation dataset for different settings of C_1 and C_2 .

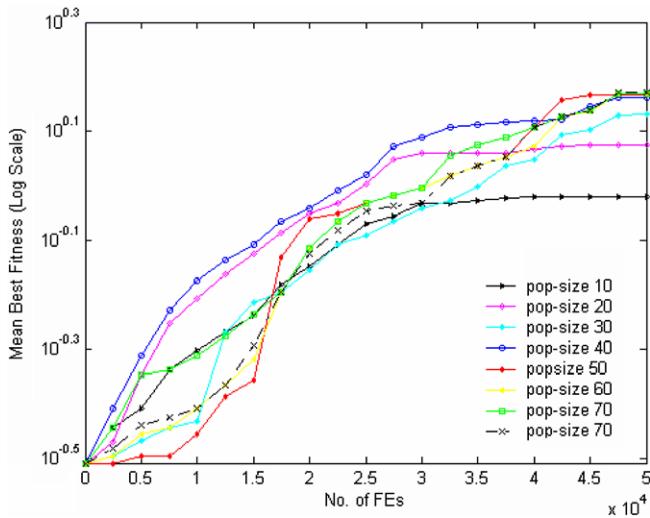


Fig. 2. The convergence characteristics of the MEPSO over the Image segmentation dataset for different population sizes.

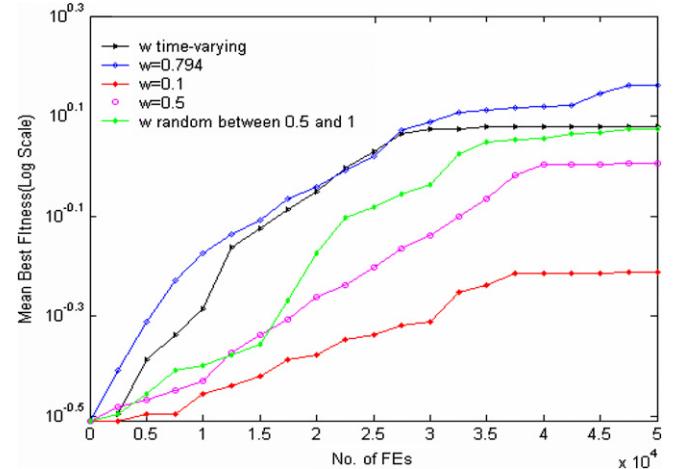


Fig. 3. The convergence characteristics of the MEPSO over the Image segmentation dataset for different inertia factors.

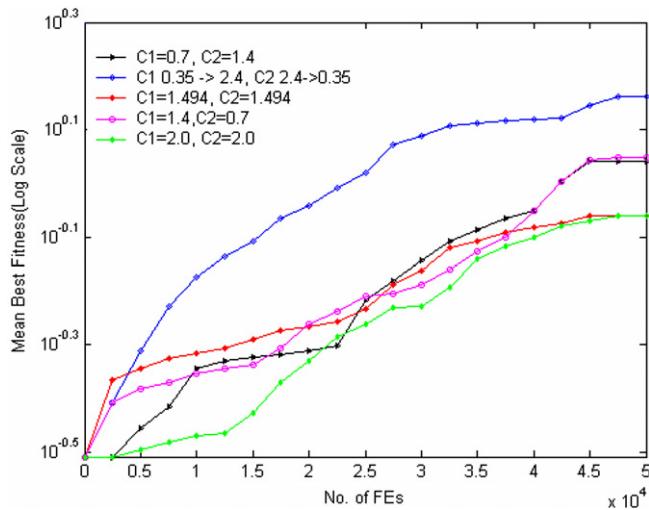


Fig. 4. The convergence characteristics of the MEPSO over the Image segmentation dataset for different acceleration coefficients.

5.3. Scalability of the kernel_MEPSO algorithm

In Section 5.1, we saw that the Kernel_MEPSO algorithm outperformed its competitors over the 2 and 3 dimensional synthetic datasets as well as over the real life datasets with dimensions ranging up to 19. In what follows, we further investigate the effect of growth of the dimensionality of the feature space on the clustering performance of Kernel_MEPSO. For this purpose, we used a few highly complex synthetic datasets of dimensions 20, 40, 60, 80 and 100 obtained from the Gaussian and ellipsoidal cluster generators available in www.dbk.ch.umist.ac.uk/handl/generators/. The detailed statistical description of the generators can be found in Handl et al. (2005). Fig. 5 shows a 100 dimensional dataset with 10 elongated and arbitrarily oriented clusters used in our experiment, when projected in the first two dimensions.

In these experiments, we compare the Kernel_MEPSO with GCUK, DCPSO and two scalable clustering algorithms including BIRCH (Zhang et al., 1996) and MOCK (MultiObjective Clustering with automatic K determination) (Handl and Knowles, 2005). We exclude the subspace clustering algorithms like CLIQUE (Agrawal et al., 1998) from this comparison, as they, unlike the algorithms being dealt here, seek for meaningful clusters in a lower dimensional subspace (a subset of the feature space dimensions).

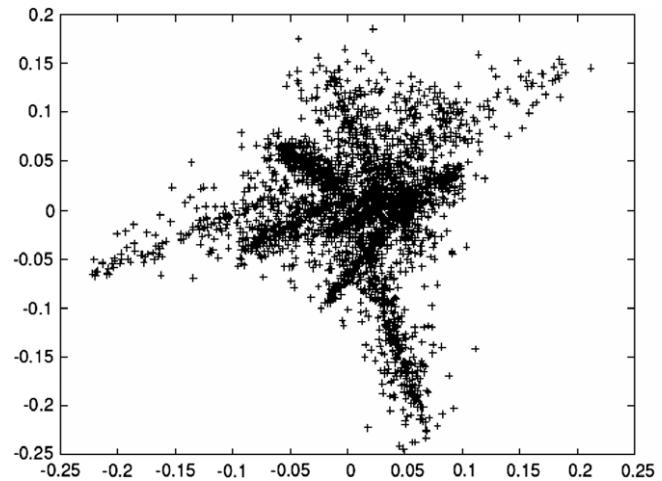


Fig. 5. 100-Dimensional data set containing 10 clusters (projected to two dimensions).

Since our primary objective is to study the scalability issues of the Kernel_MEPSO, we kept the number of clusters as well as the number of datapoints same for all the synthetic datasets generated for this purpose. Among the algorithms compared, only BIRCH was provided with the true number of classes in each case. Rest of the competitors could determine the number of clusters on the run. In Table 8, we report the mean number of classes found and the mean classification accuracy over 25 independent runs of all the competitor algorithms except BIRCH which was run once for each problem as it is deterministic and non-automatic. Fig. 6 shows how the CPU time consumed by different stochastic algorithms varies with the data dimensionality. For BIRCH we used the default parameter values provided in Zhang et al. (1996). The CPU time was measured on 1 Pentium IV, 2.2 GHz PC, with 512 KB cache and 2 GB of main memory in Windows Server 2003 environment.

The results suggest that for data dimensionality lesser than 40, the ability of Kernel_MEPSO to separate elliptical and overlapped clusters is best among all the methods compared. But as the dimension exceeds 60, the accuracy of Kernel_MEPSO becomes comparable to BIRCH and worse than the MOCK which employs a multi-objective evolutionary algorithm. The version of MOCK used here employs novel schemes for initialization and mutation that enable a more efficient exploration of the search space. It also modifies the null data model that is used as a basis

Table 8

Mean and standard deviation of the number of classes found and clustering accuracy (%) achieved by each Clustering algorithm over 25 independent runs (Each run continued up to 500,000 FEs for GCUK, DCPSO, MOCK and Kernel_MEPSO)

Dataset	GCUK		DCPSO		BIRCH	MOCK		Kernel_MEPSO	
	k Found	% Accuracy	k Found	% Accuracy		% Accuracy	k Found	% Accuracy	k Found
20d_10c	10.20	67.83 (2.8371)	10.50	75.88 (2.4918)	79.86	10.12	92.84 (3.2913)	10.00	93.29 (1.8382)
40d_10c	8.40	58.34 (5.9382)	8.12	56.88 (6.7382)	75.73	10.20	90.88 (4.9291)	10.08	91.84 (3.9932)
60d_10c	6.24	54.28 (4.0933)	8.36	53.25 (7.3855)	82.65	10.44	87.12 (3.289)	9.20	82.02 (1.9928)
80d_10c	5.88	43.23 (4.8256)	6.50	46.35 (5.30)	76.05	10.04	86.93 (5.9961)	8.40	67.82 (4.6617)
100d_10c	6.04	40.08 (1.4528)	5.60	42.35 (6.7587)	81.15	10.28	84.56 (2.0928)	7.44	58.93 (2.0931)

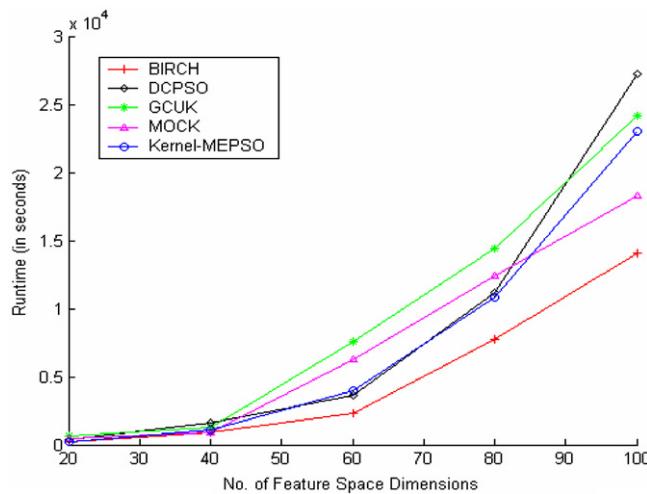


Fig. 6. The variation of CPU time with growth of data dimensionality.

for selecting the most significant solution from the Pareto front.

We also note that although the final accuracy of Kernel_MEPSO remains better than GCUK or DCPSO, above 60 dimensions, the former over fits the data, finding a larger number of clusters (than the true no. of classes) by projecting data points to an even higher dimensional space. Fig. 6 indicates that the runtime of kernel_MEPSO to converge to a threshold accuracy value, above 40 dimensions becomes worse than BIRCH and above 60 dimensions, the speed of the algorithm is hardly acceptable in comparison to that of both MOCK and BIRCH.

The deterioration of the performance of Kernel_MEPSO is in accordance with (Evangelista et al., 2006), which indicates that higher dimensionality confounds the process of kernel based learning especially in presence of unbalanced classes. The curse of dimensionality may be overcome in Kernel based clustering by utilizing the subspace models. An efficient feature selection technique may also be incorporated in the framework of the Kernel_MEPSO clustering for this purpose.

6. Conclusions

This paper has presented a novel, modified PSO-based strategy for hard clustering of complex data. An important feature of the proposed technique is that it is able to find the optimal number of clusters automatically (that is, the number of clusters does not have to be known in advance) for complex and linearly non-separable datasets. The proposed kernel_MEPSO algorithm has been shown to meet or beat the other state-of-the-art clustering algorithms in a statistically meaningful way over several benchmark datasets discussed here. This certainly does not lead us to claim that it may outperform DCPSO or GCUK over any dataset, since it is impossible to model all the possible complexities of a real life data with the limited test-suit that we used for testing the algorithms. In addition, the performance of DCPSO and GCUK may also be enhanced by

modifying there fitness functions with a kernel induced distance metric. This renders itself to further research with these algorithms.

We have provided empirical guidelines for choosing the best suited parameters for the Kernel_MEPSO after a thorough experimentation with many possible sets of values.

To investigate the effect of the growth of data dimensionality, we compared the performance of the Kernel_MEPSO with one scalable clustering algorithm BIRCH and a multi objective optimization based clustering algorithm MOCK which uses special operators to handle the curse of dimensionality issue. The test data used for these experiments included five synthetic datasets of dimensions ranging from 20 to 100. The number of data points and that of the true classes was kept same in all the datasets. It was observed that, the final classification accuracy and the mean number of classes found by the MEPSO deteriorate considerably when the feature space dimensionality exceeds 40. Future research should focus on improving the performance of the algorithm over high dimensional datasets by incorporating some feature selection mechanism in it. Automatic clustering in lower dimensional subspaces with MEPSO may also be a worthy topic of further research.

References

- Agrawal, R., Gehrke, J., Gunopulos, D., Raghavan, P., 1998. Automatic subspace clustering of high dimensional data for data mining applications. In: Proc. ACM-SIGMOD Int. Conf. Management of Data, Seattle, Washington.
- Andrews, H.C., 1972. Introduction to Mathematical Techniques in Pattern Recognition. John Wiley & Sons, New York.
- Bandyopadhyay, S., Maulik, U., 2002. Genetic clustering for automatic evolution of clusters and application to image classification. Pattern Recognition 35, 1197–1208.
- Bezdek, J.C., 1981. Pattern Recognition with Fuzzy Objective Function Algorithms. Plenum, New York.
- Blake, C., Keough, E., Merz, C.J., 1998. UCI repository of machine learning database. <http://www.ics.uci.edu/~mlearn/MLrepository.html>.
- Calinski, R.B., Harabasz, J., 1974. Adendrite method for cluster analysis. Commun. Stat., 1–27.
- Camastra, F., Verri, A., 2005. A novel kernel method for clustering. IEEE Trans. Pattern Anal. Machine Intell. 25 (7), 801–805.
- Chou, C.H., Su, M.C., Lai, E., 2004. A new cluster validity measure and its application to image compression. Pattern Anal. Appl. 7 (2), 205–220.
- Clerc, M., Kennedy, J., 2002. The particle swarm – explosion, stability, and convergence in a multidimensional complex space. In: IEEE Trans. on Evolutionary Computation, vol. 6 (1), pp. 58–73.
- Davies, D.L., Bouldin, D.W., 1979. A cluster separation measure. IEEE Trans. Pattern Anal. Machine Intell. 1, 224–227.
- Deb, K., Pratap, A., Agarwal, S., Meyarivan, T., 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. on Evolutionary Computation 6 (2).
- van den Bergh, F., Engelbrecht, A.P., 2001. Effects of swarm size on cooperative particle swarm optimizers. In: Proc. of GECCO-2001, San Francisco CA, pp. 892–899.
- Duda, R.O., Hart, P.E., 1973. Pattern Classification and Scene Analysis. John Wiley and Sons.
- Dunn, J.C., 1974. Well separated clusters and optimal fuzzy partitions. J. Cybern. 4, 95–104.

- Eberhart, R.C., Shi, Y., 2001. Particle swarm optimization: Developments, applications and resources. In: Proc. IEEE Int. Conf. Evolutionary Computation, vol. 1. pp. 81–86.
- Eberhart, R.C., Shi, Y., 2001. Particle swarm optimization: Developments, applications and resources. In: Proc. IEEE International Conference on Evolutionary Computation, vol. 1. pp. 81–86.
- Evangelista, P.F., Embleton, M.J., Szymanski, B.K., 2006. Taming the curse of dimensionality in kernels and novelty detection. Advances in Soft Computing: Applied Soft Computing Technologies: The Challenge of Complexity. Springer.
- Evangelou, I.E., Hadjimitsis, D.G., Lazakidou, A.A., Clayton, C., 2001. Data mining and knowledge discovery in complex image data using artificial neural networks. In: Workshop on Complex Reasoning an Geographical Data, Cyprus.
- Everitt, B.S., 1993. Cluster Analysis, third ed. Halsted Press.
- Falkenauer, E., 1998. Genetic Algorithms and Grouping Problems. John Wiley and Son, Chichester.
- Forgy, E.W., 1965. Cluster analysis of multivariate data: Efficiency versus interpretability of classification. *Biometrics* 21, 768–769.
- Forgy, E.W., 1965. Cluster analysis of multivariate data: Efficiency versus interpretability of classification. *Biometrics* 21.
- Frigui, H., Krishnapuram, R., 1999. A robust competitive clustering algorithm with applications in computer vision. *IEEE Trans. Pattern Anal. Machine Intell.* 21 (5), 450–465.
- Fukunaga, K., 1990. Introduction to Statistical Pattern Recognition. Academic Press.
- Girolami, M., 2002. Mercer kernel-based clustering in feature space. *IEEE Trans. Neural Networks* 13 (3), 780–784.
- Hamerly, G., Elkan, C., 2003. Learning the k in k -means. *Neural Information Processing Systems*, NIPS 2003, December 8–13, Vancouver and Whistler, British Columbia, Canada.
- Handl, J., Knowles, J., 2005. Improving the scalability of multi-objective clustering. *Proc. Congress Evolutionary Comput. (CEC 2005)* 3, 2372–2379.
- Handl, J., Knowles, J., Multiobjective clustering around medoids. In: Proceedings of the Congress on Evolutionary Computation (CEC 2005), vol. 1. pp. 632–639.
- Hartigan, J.A., 1975. Clustering Algorithms. Wiley, New York.
- Hertz, T., Bar, A., Daphna Weinshall, H., 2006. Learning a kernel function for classification with small training samples. In: Appearing in Proc. 23rd Int. Conf. on Machine Learning, Pittsburgh, PA.
- Huang, Z., Ng, M.K., 1999. A fuzzy k -modes algorithm for clustering categorical data. *IEEE Trans. Fuzzy Systems* 7 (4), 446–452.
- Jain, A.K., Murty, M.N., Flynn, P.J., 1999. Data clustering: a review. *ACM Comput. Surv.* 31 (3), 264–323.
- Kanade, P.M., Hall, L.O., 2003. Fuzzy ants as a clustering concept. In: Proceedings of the 22nd International Conference of the North American Fuzzy Information Processing Society (NAFIPS03), pp. 227–232.
- Kennedy, J., Eberhart, R., 1995. Particle swarm optimization. In: Proc. of IEEE Int. Conf. Neural Networks, pp. 1942–1948.
- Kennedy, J., Eberhart, R.C., Shi, Y., 2001. Swarm Intelligence. Morgan Kaufman Publishers, San Francisco.
- Kim, D.-W., Lee, K.Y., Lee, D., Lee, K.H., 2005. A kernel-based subtractive clustering method. *Pattern Recognition Lett.* 26 (7), 879–891.
- Kohonen, T., 1995. In: Self-Organizing Maps, Springer Series in Information Sciences, vol. 30. Springer-Verlag.
- Leung, Y., Zhang, J., Xu, Z., 2000. Clustering by space-space filtering. *IEEE Trans. Pattern Anal. Machine Intell.* 22 (12), 1396–1410.
- Lillesand, T., Keifer, R., 1994. Remote Sensing and Image Interpretation. John Wiley & Sons.
- Mao, J., Jain, A.K., 1995. Artificial neural networks for feature extraction and multivariate data projection. *IEEE Trans. Neural Networks* 6, 296–317.
- Mitchell, T., 1997. Machine Learning. McGraw-Hill, Inc., New York, NY.
- Muller, K.R., Mika, S., Ratsch, G., Tsuda, K., Scholkopf, B., 2001. An introduction to kernel-based learning algorithms. *IEEE Trans. Neural Networks* 12 (2), 181–202.
- Murthy, C.A., Chowdhury, N., 1996. In search of optimal clusters using genetic algorithm. *Pattern Recognition Lett.* 17, 825–832.
- Omran, M., Engelbrecht, A., Salman, A., 2005. Particle swarm optimization method for image clustering. *Int. J. Pattern Recognition Artificial Intell.* 19 (3), 297–322.
- Omran, M., Salman, A., Engelbrecht, A.P., 2005. Dynamic clustering using particle swarm optimization with application in unsupervised image classification. In: Fifth World Enformatika Conference (ICCI 2005), Prague, Czech Republic.
- Pakhira, M.K., Bandyopadhyay, S., Maulik, U., 2004. Validity index for crisp and fuzzy clusters. *Pattern Recognition Lett.* 37, 487–501.
- Pal, N.R., Chakraborty, D., 2000. Mountain and subtractive clustering method: Improvements and generalization. *Int. J. Intelligent Syst.* 15, 329–341.
- Pal, N.R., Bezdek, J.C., Tsao, E.C.-K., 1993. Generalized clustering networks and Kohonen's self-organizing scheme. *IEEE Trans. Neural Networks* 4, 549–557.
- Paterlini, S., Krink, T., 2006. Differential evolution and particle swarm optimisation in partitional clustering. *Comput. Stat. Data Anal.* 50 (5), 1220–1247.
- Paterlini, S., Minerva, T., 2003. Evolutionary Approaches for Cluster Analysis. In: Bonarini, A., Masulli, F., Pasi, G. (Eds.), Soft Computing Applications. Springer-Verlag, Berlin, pp. 167–178.
- Pirooznia, M., Deng, Y., SVM Classifier – a comprehensive java interface for support vector machine classification of microarray data. In: Proc. of Symp. of Computations in Bioinformatics and Bioscience (SCBB06), Hangzhou, China. <http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=1780131>.
- Rao, M.R., 1971. Cluster analysis and mathematical programming. *J. Amer. Stat. Assoc.* 22, 622–626.
- Ratnaweera, A., Halgamuge, K.S., 2004. Self organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. In: *IEEE Trans. on Evolutionary Computation*, vol. 8 (3), pp. 240–254.
- Rosenberger, C., Chehdi, K., 2000. Unsupervised clustering method with optimal estimation of the number of clusters: Application to image segmentation. In: Proc. IEEE International Conf. on Pattern Recognition (ICPR), vol. 1. Barcelona, pp. 1656–1659.
- Sarkar, M., Yegnanarayana, B., Khemani, D., 1997. A clustering algorithm using an evolutionary programming-based approach. *Pattern Recognition Lett.* 18, 975–986.
- Scholkopf, B., Smola, A.J., 2002. Learning with Kernels. The MIT Press, Cambridge.
- Shi, Y., Eberhart, R.C., 1999. Empirical Study of particle swarm optimization. In: Proc. IEEE Int. Conf. Evolutionary Computation, vol. 3. pp. 101–106.
- Vapnik, V.N., 1998. Statistical Learning Theory. Wiley, New York. www.dkb.ch.umist.ac.uk/handl/generators/.
- Zahn, C.T., 1971. Graph-theoretical methods for detecting and describing gestalt clusters. *IEEE Trans. Comput.* C-20, 68–86.
- Zhang, D.Q., Chen, S.C., 2003. Clustering incomplete data using kernel-based fuzzy c -means algorithm. *Neural Process. Lett.* 18, 155–162.
- Zhang, R., Rudnicki, A.I., 2002. A large scale clustering scheme for kernel k -means. In: The Sixteenth Int. Conf. on Pattern Recognition, pp. 289–292.
- Zhang, T., Ramakrishnan, R., Livny, M., 1996. BIRCH: An efficient data clustering method for very large databases. In: Proc. of the ACM SIGMOD Conference on Management of Data, Montreal, Canada.