Kernel Based Automatic Clustering Using Modified Particle Swarm Optimization Algorithm

Ajith Abraham
Q2S, Norwegian University of Science and Technology, Norway
(47) 73592743
ajith.abraham@ieee.org

Swagatam Das
Jadavpur University
Kolkata 700032, India
(91)9831219774
swagatamdas19@yahoo.co.in

Amit Konar
Jadavpur University
Kolkata 700032, India
(91)03324162697
konaramit@yahoo.com

ABSTRACT
This paper introduces a method for clustering complex and linearly non-separable datasets, without any prior knowledge of the number of naturally occurring clusters. The proposed method is based on an improved variant of the Particle Swarm Optimization (PSO) algorithm. In addition, it 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. Computer simulations have been undertaken with a test bench of five synthetic and three real life datasets, in order to compare the performance of the proposed method with a few state-of-the-art clustering algorithms. The results reflect the superiority of the proposed algorithm in terms of accuracy, convergence speed and robustness.

Categories and Subject Descriptors
I.2.2 [Automatic Programming], I.2.6 [Learning], I.5.3 [Clustering], H.3.3 [Information Search and Retrieval]

General Terms
Algorithms, Experimentation.

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 [1-3].

Data clustering algorithms can be *hierarchical* or *partitional* [4]. In hierarchical clustering, the output is a tree showing a sequence of clustering with each cluster being a partition of the data set [4]. 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 [4]. A comprehensive survey of the various clustering algorithms can be found in [4].

The problem of partitional clustering has been approached from diverse fields of knowledge like statistics (multivariate analysis) [5], graph theory [6], expectation maximization algorithms [7], artificial neural networks [8], evolutionary computing [9-12], swarm intelligence [13-15] and so on.

Most of the existing evolutionary or swarm based clustering algorithms employ an Euclidian distance metric to construct their fitness functions. They work well with datasets in which the natural clusters are nearly hyper-spherical and linearly separable (like the artificial dataset 1 used in this paper). But the same algorithms provide severe misclassifications when the dataset is complex, with linearly non-separable patterns (like the synthetic datasets 2, 3 and 4 described in Section 5). Moreover, very few works [16-18] have been undertaken to determine the 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 [19].

In this paper, we propose an alternative approach towards the problem of automatic clustering (without having any prior knowledge of k initially) using a modified version of the PSO algorithm [20]. Our procedure employs a kernel induced similarity measure instead of the conventional Euclidean distance metric used by majority of the existing clustering techniques. 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 preserves the inherent structure of groups in the input space, but also simplifies the associated structure of the data patterns [21, 22]. Several kernel-based learning methods, including the Support Vector Machine (SVM), have recently shown to perform remarkably well in supervised learning [23-25]. The kernelized versions of the k-means [26] and the fuzzy c-means (FCM) [27] algorithms reported in [22] and [25] have reportedly outperformed their original counterparts over several test cases.

GECCO'07, July 7–11, 2007, London, England, United Kingdom.

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

i) First, we develop a framework for detecting the number of partitions in a dataset along with the simultaneous refining of the cluster prototypes 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 solutions than the classical PSO in context to the present problem.

iii) We reformulate the CS cluster validity index [28] using the kernelized distance measure. This reformulation eliminates the need to compute the cluster-centroids repeatedly for evaluating CS value, due to the implicit mapping via the kernel function. The new CS measure forms the objective function in our algorithm.

We have undertaken extensive performance comparisons in order to establish the effectiveness of the proposed method. The rest of the paper is organized 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 THE VALIDITY INDICES

2.1 The Crisp Clustering Problem

Let $X = \{\vec{x}_1, \vec{x}_2, ..., \vec{x}_n\}$ 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, ...,d) of the i-th pattern (i =1,2,...., n). Given such a set, the partitional clustering algorithm tries to find a partition C = {C_1, C_2,....., 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) $C_i \neq \Phi \ \forall i \in \{1,2,...,k\}$.

2) $C_i \cap C_j = \Phi$, $\forall i \neq j$ and $i, j \in \{1,2,...,k\}$.

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,

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

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*:

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

Where $\vec{x}_i = [x_{i,1}, x_{i,2}, ..., x_{i,d}]^T$ and

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

By applying the mapping, a dot product $\vec{x}_i^{\,T}.\vec{x}_j$ is transformed into $\varphi^T(\vec{x}_i).\varphi(\vec{x}_j)$. Now, the central idea in kernel-based learning is that the mapping function φ need not be explicitly specified. The dot product $\varphi^T(\vec{x}_i).\varphi(\vec{x}_j)$ 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:

$\varphi : \Re^2 \to H = \Re^3$, and $[x_{i,1}, x_{i,2}]^T \to [x_{i,1}^2, \sqrt{2}.x_{i,1}x_{i,2}, x_{i,2}^2]^T$

Now the dot product in feature space H:

$\varphi^T(\vec{x}_i).\varphi(\vec{x}_j)$

$= [x_{i,1}^2, \sqrt{2}.x_{i,1}x_{i,2}, x_{i,2}^2].[x_{j,1}^2, \sqrt{2}.x_{j,1}.x_{j,2}, x_{j,2}^2]^T$.

$= [x_{i,1}.x_{j,1} + x_{i,2}.x_{j,2}]^2$

$= [\vec{x}_i^T.\vec{x}_j]^2 = K(\vec{x}_i, \vec{x}_j)$

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:

$$\left\| \varphi(\vec{x}_i) - \varphi(\vec{x}_j) \right\|^2 = (\varphi(\vec{x}_i) - \varphi(\vec{x}_j))^T (\varphi(\vec{x}_i) - \varphi(\vec{x}_j))$$

$$= \varphi^T(\vec{x}_i).\varphi(\vec{x}_i) - 2.\varphi^T(\vec{x}_i).\varphi(\vec{x}_j) + \varphi^T(\vec{x}_j).\varphi(\vec{x}_j)$$

$$= K(\vec{x}_i, \vec{x}_i) - 2.K(\vec{x}_i, \vec{x}_j) + K(\vec{x}_j, \vec{x}_j) \qquad (3)$$

Among the various kernel functions used in literature, in the present context, we have chosen the most widely used Gaussian kernel (also known as the Radial Basis Function) [32], which can be represented as:

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

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

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

2.3 The Kernelized Validity Index

Cluster validity indices correspond to the statistical-mathematical functions used to judge the results of any clustering algorithm on a quantitative basis. For crisp clustering, a few well-known indices available in the literature are the Dunn's index (DI) [29], Calinski-Harabasz index [30] and Davis-Bouldin (DB) index [31]. In this work, we have based our fitness function on a recently proposed validity index known as CS measure [28]. This is due to the fact that, CS measure has been shown to improve over several

other popular validity indices in tackling clusters of different densities and/or sizes in [28]. According to Chou *et al*. the price to be paid for this greater accuracy comes in the form of high computational load with increasing *k* and *n*.

Before applying the CS measure, centroid of a cluster is computed by averaging the data vectors belonging to that cluster using the formula,

$$\vec{m}_i = \frac{1}{N_i} \sum_{x_j \in C_i} \vec{x}_j \tag{6}$$

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,

$$CS(k) = \frac{\frac{1}{k}\sum_{i=1}^{k}[\frac{1}{N_i}\sum_{\vec{X}_i \in C_i} \max_{\vec{X}_q \in C_i}\{d(\vec{x}_i, \vec{x}_q)\}]}{\frac{1}{k}\sum_{i=1}^{k}[\min_{j \in K, j \neq i}\{d(\vec{m}_i, \vec{m}_j)\}]}$$

$$= \frac{\sum_{i=1}^{k}[\frac{1}{N_i}\sum_{\vec{X}_i \in C_i} \max_{\vec{X}_q \in C_i}\{d(\vec{x}_i, \vec{x}_q)\}]}{\sum_{i=1}^{k}[\min_{j \in K, j \neq i}\{d(\vec{m}_i, \vec{m}_j)\}]} \tag{7}$$

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

$$CS_{kernel}(k) = \frac{\sum_{i=1}^{k}[\frac{1}{N_i}\sum_{\vec{X}_i \in C_i} \max_{\vec{X}_q \in C_i}\{\|\varphi(\vec{x}_i) - \varphi(\vec{x}_q)\|^2\}]}{\sum_{i=1}^{k}[\min_{j \in K, j \neq i}\{\|\varphi(\vec{m}_i) - \varphi(\vec{m}_j)\|\}]}$$

$$= \frac{\sum_{i=1}^{k}[\frac{1}{N_i}\sum_{\vec{X}_i \in C_i} \max_{\vec{X}_q \in C_i}\{2(1 - K(\vec{x}_i, \vec{x}_q))\}]}{\sum_{i=1}^{k}[\min_{j \in K, j \neq i}\{2(1 - K(\vec{m}_i, \vec{m}_j))\}]} \tag{8}$$

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

3. THE MULTI-ELITIST PSO (MEPSO)

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, $\vec{Z}_i = (Z_{i1}, Z_{i2}, Z_{i3},...,Z_{in})$ and $\vec{V}_i = (V_{i1}, V_{i2}, V_{i3},...,V_{in})$. 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

$$V_{id}(t+1) = \omega.V_{id}(t) + C_1.\varphi_1.(P_{lid} - X_{id}(t)) + C_2.\varphi_2.(P_{gd} - X_{id}(t))$$
$$Z_{id}(t+1) = Z_{id}(t) + V_{id}(t+1) \tag{9}$$

The variables φ_1 and φ_2 are random positive numbers, drawn from a uniform distribution and defined by an upper limit $\varphi_{max,}$ which is a parameter of the system. C_1 and C_2 are called acceleration coefficients whereas ω is called inertia weight. $\vec{P}_l$ 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. 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 [32, 33]. In many occasions, the convergence is premature, especially if the swarm uses a small inertia weight ω or constriction coefficient [33]. 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 [34]. 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. Therefore, the fitness value of the new global best is always higher than the old global best. The pseudo code about MEPSO is as follows:

Procedure MEPSO

For *t* =1 to t_{max}
  For *j* =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)
      $\beta_j = \beta_j + 1;$
    End
    Update Local best*j* .
    If (the fitness of Local bestj > that of Global best now*)*
      Choose Local bestj 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 The 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 c_{max} entries are positive floating-point numbers in (0, 1), each of which controls 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:

$\vec{Z}_i(t) =$

$T_{i,1}$	$T_{i,2}$	.....	$T_{i,Kmax}$	$\vec{m}_{i,1}$	$\vec{m}_{i,2}$	.....	$\vec{m}_{i,k\max}$

Activation Threshhold **Cluster Centroids**

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 parameters (we call them *activation thresholds*) in the particles governing the selection of the active cluster centers. The rule for selecting the actual number of clusters specified by one particle 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** (10)

The quality of the partition yielded by such a particle can be judged by an appropriate cluster validity index.

If due to mutation 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 the *i*-th particle can be described as:

$$f_i = \frac{1}{CS_{\ker nel_i}(k) + eps} \quad (11)$$

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 (a positive infinity like 5.0/0.0 or not a number situation like 0.0/0.0) may be encountered. This can occur when one of the selected cluster centers is outside the boundary of distributions of the data set. 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. Suppose there are *k* active cluster centers in this particle and we have total *n* data points. Then we collect *n/k* data points for every individual cluster center that has the minimum distance between data points and itself. Finally, we update the new cluster centers by computing the average value of *n/k* data points that were selected into the corresponding cluster centers.

4.4 Complete Procedure

The complete algorithm can now be presented in the following form:

Step 1: Initialize each chromosome 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 =1to $t_{\max}$ **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 $\vec{Z}_i$.

ii) Assign $\vec{X}_p$ to that particular cluster center $\vec{m}_{i,j}$ where

$$d(\vec{X}_p, \vec{m}_{i,j}) = \min_{\forall b \in \{1,2,....,k\}} \left\{ d(\vec{X}_p, \vec{m}_{i,b}) \right\}$$

iii) Check if the number of data points belonging to any cluster center m_{ij} 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. EXPERIMENTS

To test the effectiveness of the proposed method, we compare its performance with five other clustering algorithms using a test-suit of five artificial and three real world datasets. Among the considered clustering algorithms, there are two recently developed automatic clustering algorithms well known as the GCUK (Genetic Clustering with an Unknown number of clusters K) [12] and the DCPSO (Dynamic Clustering PSO) [18]. In order to investigate the effects of the changes made in the classical PSO algorithm, we have compared MEPSO with an ordinary PSO based kernel-clustering method that uses the same particle representation and fitness function as the MEPSO. Both of these algorithms were let run on the same initial populations. The rest of the clustering algorithms are the kernel k-means algorithm [22] and a kernelized version of the subtractive clustering [35]. Both the algorithms were provided with the correct number of clusters as they are non-automatic.

Table 2. Parameter settings for different algorithms

GCUK		DCPSO		PSO		MEPSO	
Pop_size	70	Pop_size	40	Pop_size	40	Pop_size	40
Cross-over Probability μ_c	0.85	Inertia weight	0.75	Inertia weight	0.75	Inertia weight	0.794
Mutation probability μ_m	0.005	C_1, C_2	1.494	C_1, C_2	2.00	C_1 C_2	0.35 → 2.4 2.4 → 0.35
		P_{ini}	0.80				
K_{max} K_{min}	15 2	K_{max} K_{min}	15 2	K_{max} K_{min}	15 2	K_{max} K_{min}	15 2

Table 1. Description of the datasets

Dateset	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
Vowel	871	6	3
Breast cancer	683	2	9

We used datasets with a wide variety in the number and shape of clusters, number of data points and the count of features of each datum. The real life datasets used in the experiments are well-known as the Vowel [12], Glass and the Wisconsin breast cancer data set [36]. The synthetic datasets included, 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. Scatter plot of the synthetic datasets have also been shown in Figure 1.The clustering results were judged using Huang's accuracy measure [37]:

$$r = \frac{\sum_{i=1}^{k} n_i}{n} \tag{12}$$

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 vowel and glass dataset for the RBF kernel. In these experiments, the kernel k-means was run 100 times with the initial centroids randomly 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 $\beta = 1.5$ [35]. Rest of the parameter settings are shown 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 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).Table 5 presents the mean and standard deviation of the number of classes found by the three automatic clustering algorithms. In Figure 2 we present the clustering results on the synthetic datasets by the new MEPSO algorithm (to save space we do not provide all the results obtained through simulation). 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 evolutionary clustering algorithm on each dataset and stop as soon as the algorithm achieves the proper number of clusters as well as the threshold accuracy. 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 6. 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.

As evident from Tables 3, 5 and 6, 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 including the simple kernel k-means and the kernel-subtractive clustering algorithm, 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.

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 function evaluations for GCUK, DCPSO, Kernel_ PSO and Kernel_MEPSO)

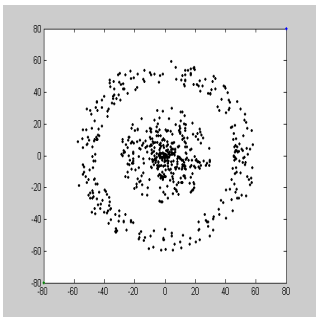
Datasets	Algorithms					
	Kernel k-means	Kernel Sub_clust	GCUK	DC-PSO	Kernel_PSO	Kernel_MEPSO
$Synthetic_1$	83.45 (0.032)	87.28	54.98 (0.88)	57.84 (0.065)	90.56 (0.581)	**99.45 (0.005)**
$Synthetic_2$	71.32 (0.096)	75.73	65.82 (0.146)	59.91 (0.042)	61.41 (0.042)	**80.92 (0.0051)**
$Synthetic_3$	89.93 (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)**
Vowel	70,83 (0.202)	54.36	79.04 (0.381)	86.31 (0.362)	84.63 (0.903)	**94.15 (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)**
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

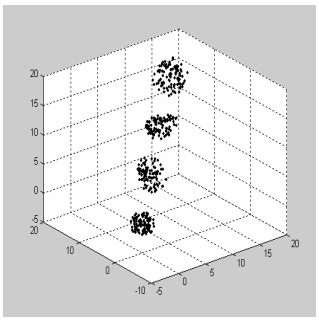
Datasets	Std. Err	*t*	95% Conf. Interval	Two-tailed *P*	Significance
$Synthetic_1$	0.005	976.36	(-5.01, -4.98)	< 0.0001	**Extremely significant**
$Synthetic_2$	0.001	9094.7	(-7.19, -7.18)	< 0.0001	**Extremely significant**
$Synthetic_3$	0.015	129.88	(-1.94, -1.88)	< 0.0001	**Extremely significant**
$Synthetic_4$	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**
Vowel	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**

Table 5: Mean and standard deviation (in parenthesis) of the number of clusters found for four automatic clustering algorithms over 40 independent runs.

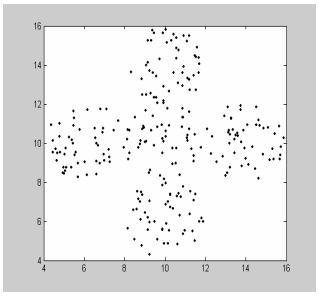
Algorithms	Synthetic_1	Synthetic_2	Synthetic_3	Synthetic_4	Synthetic_5	Glass	Breast cancer	Vowel
GCUK	2.50 (0.021)	3.05 (0.118)	4.15 (0.039)	9.85 (0.241)	4.25 (0.921)	5.85 (0.035)	2.25 (0.063)	5.05 (0.008)
DCPSO	2.45 (0.121)	2.80 (0.036)	4.25 (0.051)	9.05 (0.726)	6.05 (0.223)	5.60 (0.009)	2.25 (0.026)	7.50 (0.057)
Ordinary PSO	2.50 (0.026)	2.65 (0.126)	4.10 (0.062)	9.35 (0.335)	2.25 (0.361)	5.75 (0.075)	**2.00 (0.00)**	5.70 (0.025)
Kernel_MEPSO	**2.10 (0.033)**	**2.15 (0.102)**	**4.00 (0.00)**	**10.05 (0.021)**	**2.05 (0.001)**	**6.05 (0.015)**	**2.00 (0.00)**	**5.90 (0.001)**



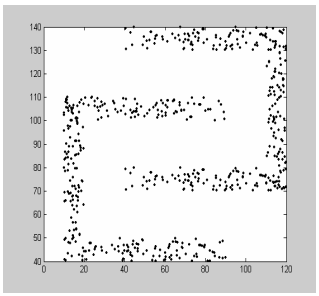
(a) Synthetic$_1$



(b) Synthetic$_3$

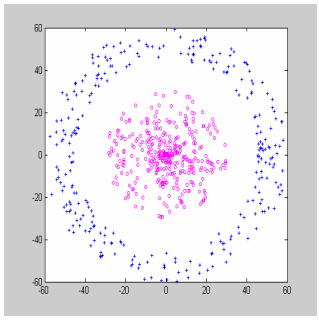


(c) Synthetic$_4$

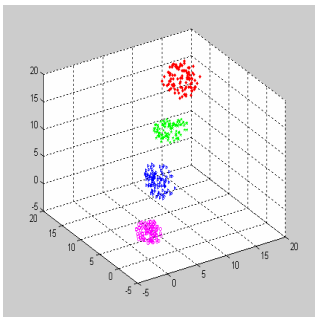


(d) Synthetic$_5$

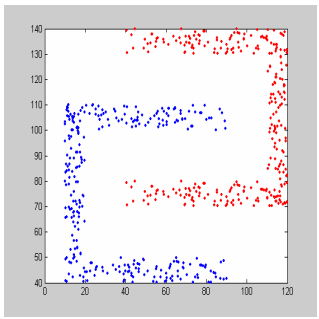
Fig. 1. Two- and three-dimensional synthetic datasets used in this study.



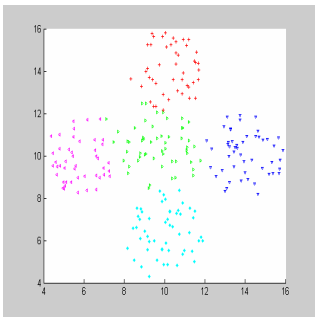
(a) Synthetic$_1$



(b) Synthetic$_3$



(c) Synthetic$_5$



(d) Synthetic$_4$

Fig. 2. Two- and three-dimensional synthetic datasets clustered with MEPSO.

Table 6: Mean and standard deviations of the number of FEs (over 40 successful runs) required by each algorithm to reach a predefined cut-off value of the classification accuracy.

Dateset	Thresh-hold 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)**
Vowel	55.00	43392.25 (31.83)	41271.05 (8.03)	34070.65 (5.26)	**32847.95 (23.62)**
Breast Cancer	65.00	40390.00 (11.45)	37262.65 (13.64)	35872.05 (8.32)	**32837.65 (4.26)**

6. CONCLUSIONS

This paper has presented a new, PSO-based strategy for clustering of linearly non-separable patterns. 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 any previously unhandled dataset. The proposed kernel_MEPSO algorithm has been shown to outperform the other state-of-the-art clustering algorithms in a statistically meaningful way over all the benchmark datasets discussed in this paper. This certainly does not lead us to claim that it may outperform DCPSO or GCUK over every 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. However, the only conclusion we can draw at this point is that MEPSO with a kernel based CS measure can serve as an attractive alternative for dynamic clustering of completely unknown datasets with complex cluster structures.

7. REFERENCES

[1] I. E Evangelou, D. G Hadjimitsis, A. A Lazakidou, C. Clayton, Data Mining and Knowledge Discovery in Complex Image Data using Artificial Neural Networks, Workshop on Complex Reasoning an Geographical Data, Cyprus, 2001.

[2] T Lillesand, R Keifer, Remote Sensing and Image Interpretation, John Wiley & Sons, 1994.

[3] H. C Andrews, Introduction to Mathematical Techniques in Pattern Recognition, John Wiley & Sons, New York, 1972.

[4] A. K. Jain, M. N. Murty, P. J. Flynn, Data clustering: a review, ACM Computing Surveys, vol. 31, no.3, (1999) 264—323.

[5] E.W. Forgy, Cluster Analysis of Multivariate Data: Efficiency versus Interpretability of classification, Biometrics, 21, (1965) 768-9.

[6] C. T. Zahn, Graph-theoretical methods for detecting and describing gestalt clusters. IEEE Transactions on Computers C-20 (1971), 68–86.

[7] T. Mitchell, Machine Learning. McGraw-Hill, Inc., New York, NY (1997).

[8] J. Mao, A. K. Jain, Artificial neural networks for feature extraction and multivariate data projection. IEEE Trans. Neural Networks.vol. 6, (1995) 296–317.

[9] E. Falkenauer, Genetic Algorithms and Grouping Problems, John Wiley and Son, Chichester (1998).

[10] S. Paterlini, T. Minerva, Evolutionary Approaches for Cluster Analysis. In A. Bonarini, F. Masulli, G. Pasi (eds.) Soft Computing Applications. Springer-Verlag, Berlin. (2003)167-178.

[11] C.A. Murthy, N. Chowdhury, In search of optimal clusters using genetic algorithm, Pattern Recognition Letters, 17, (1996) 825-832.

[12] S. Bandyopadhyay, U. Maulik, Genetic clustering for automatic evolution of clusters and application to image classification, Pattern Recognition, 35, (2002) 1197-1208.

[13] S. Paterlinia, T. Krink, Differential evolution and particle swarm optimisation in partitional clustering, Computational Statistics & Data Analysis, Volume 50, Issue 5, (2006) 1220-1247.

[14] M. Omran, A. Engelbrecht and A. Salman, Particle Swarm Optimization Method for Image Clustering, International Journal of Pattern Recognition and Artificial Intelligence, vol. 19, no. 3, (2005) 297-322,

[15] P. M. Kanade, L. O. Hall 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. (2003).

[16] G. Hamerly, C. Elkan, Learning the k in k-means. Neural Information Processing Systems, NIPS 2003, December 8-13, Vancouver and Whistler, British Columbia, Canada (2003).

[17] M. Sarkar,, B. Yegnanarayana and D. Khemani, A clustering algorithm using an evolutionary programming-based approach, Pattern Recognition Letters, 18, (1997) 975–986.

[18] M. Omran, A. Salman and A. P. Engelbrecht. Dynamic Clustering using Particle Swarm Optimization with Application in Unsupervised Image Classification. Fifth World Enformatika Conference (ICCI 2005), Prague, Czech Republic, 2005.

[19] C. Rosenberger, K. Chehdi, Unsupervised clustering method with optimal estimation of the number of clusters: Application to image segmentation, in Proc. IEEE International Conference on Pattern Recognition (ICPR), vol. 1, Barcelona, (2000) 1656-1659.

[20] 20. J. Kennedy, R. Eberhart, Particle swarm optimization, In Proceedings of IEEE International Conference on Neural Networks, (1995) 1942-1948.

[21] K. R. Muller, S. Mika, G. Ratsch, K. Tsuda and B. Scholkopf, An introduction to kernel-based learning algorithms. IEEE Trans. Neural Networks 12 (2), 181–202 (2001).

[22] M. Girolami, Mercer kernel-based clustering in feature space. IEEE Trans. Neural Networks 13(3), 780–784 (2002).

[23] B. Scholkopf and A.J. Smola, Learning with Kernels. The MIT Press, Cambridge (2002).

[24] D. Q. Zhang, S. C. Chen, Clustering incomplete data using kernel-based fuzzy c-means algorithm. Neural Process Letters 18, 155–162, (2003).

[25] R. Zhang, A.I. Rudnicky, A large scale clustering scheme for kernel k-means. In: The Sixteenth International Conference on Pattern Recognition, p. 289–292, (2002).

[26] E.W. Forgy, Cluster Analysis of Multivariate Data: Efficiency versus Interpretability of classification, Biometrics, 21, (1965).

[27] J. C. Bezdek , Pattern recognition with fuzzy objective function algorithms. New York: Plenum. (1981).

[28] C.H. Chou, M.C. Su, E. Lai, A new cluster validity measure and its application to image compression. Pattern Analysis and Applications 7(2), 205-220, (2004).

[29] J.C. Dunn, Well separated clusters and optimal fuzzy partitions. J. Cybern. 4, (1974) 95-104.

[30] R.B. Calinski, J. Harabasz, Adendrite method for cluster analysis, Commun. Statistics, 1– 27, (1974)

[31] D.L. Davies, D.W. Bouldin, A cluster separation measure, IEEE Transactions on Pattern Analysis and Machine Intelligence 1 (1979) 224–227.

[32] R. C. Eberhart, Y. Shi, Particle swarm optimization: Developments, applications and resources, In Proceedings of IEEE International Conference on Evolutionary Computation, vol. 1, 81-86, (2001).

[33] M. Clerc, J. Kennedy, The particle swarm - explosion, stability, and convergence in a multidimensional complex space, In IEEE Transactions on Evolutionary Computation 6(1): 58-73, (2002).

[34] K. Deb, A. Pratap, S. Agarwal and T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Trans. on Evolutionary Computation, Vol.6, No.2, (2002).

[35] D. –W. Kim, K. Y. Lee, D. Lee, K. H. Lee, A kernel-based subtractive clustering method. Pattern Recognition Letters 26(7): 879-891 (2005).

[36] C. Blake, E.Keough and C. J. Merz, UCI repository of machine learning database (1998). http://www.ics.uci.edu/~mlearn/MLrepository.html

[37] Z. Huang and M.K. Ng, A fuzzy k-modes algorithm for clustering categorical data. IEEE Trans. Fuzzy Systems 7 (4), 446–452, (1999).