Soft Comput (2016) 20:3163–3176
DOI 10.1007/s00500-015-1686-5

METHODOLOGIES AND APPLICATION

Artificial bee colony algorithm for clustering: an extreme learning approach

Abobakr Khalil Alshamiri[1] · **Alok Singh**[1] · **Bapi Raju Surampudi**[1,2]

Published online: 28 April 2015

Abstract Extreme learning machine (ELM) as a new learning approach has shown its good generalization performance in regression and classification applications. Clustering analysis is an important tool to explore the structure of data and has been employed in many disciplines and applications. In this paper, we present a method that builds on ELM projection of input data into a high-dimensional feature space and followed by unsupervised clustering using artificial bee colony (ABC) algorithm. While ELM projection facilitates separability of clusters, a metaheuristic technique such as ABC algorithm overcomes problems of dependence on initialization of cluster centers and convergence to local minima suffered by conventional algorithms such as K-means. The proposed ELM-ABC algorithm is tested on 12 benchmark data sets. The experimental results show that the ELM-ABC algorithm can effectively improve the quality of clustering.

Keywords Clustering · Extreme learning machine · K-means algorithm · Artificial bee colony algorithm

Communicated by V. Loia.

✉ Alok Singh
alokcs@uohyd.ernet.in

Abobakr Khalil Alshamiri
abobakr2030@yahoo.com

Bapi Raju Surampudi
raju.bapi@iiit.ac.in; bapics@uohyd.ernet.in

[1] School of Computer and Information Sciences, University of Hyderabad, Hyderabad 500 046, India

[2] Cognitive Science Lab, International Institute of Information Technology, Hyderabad 500 032, India

1 Introduction

Clustering is a very important problem that has been addressed in a large variety of applications and domains, for example, image segmentation, financial fraud detection, object and character recognition, document retrieval, medical diagnosis, remote sensing, data compression, etc. Clustering analysis identifies intrinsic grouping(s) of a set of patterns, points, or objects. The goal of data clustering is to group a set of objects, on the basis of a *similarity* (or *dissimilarity*) measure, into clusters (also called groups) in such a way that the similarities between objects belonging to different clusters are minimized and the similarities between objects belonging to the same cluster are maximized, i.e., the objects within a cluster are more similar to each other (high intra-cluster similarity) than objects belonging to different clusters (low inter-cluster similarity) (Han and Kamber 2001; Jain et al. 1999; Filippone et al. 2008). There are several *similarity* (or *dissimilarity*) measures reported in the literature (Xu and Wunsch II 2005) and the choice of an appropriate measure depends on the data under analysis and the purpose of the analysis.

Clustering methods are generally classified into two categories based on the structure of abstraction, viz. hierarchical clustering and partitional clustering (Jain et al. 1999; Filippone et al. 2008). Hierarchical clustering techniques group data objects by constructing a hierarchy of partitions. In contrast to hierarchical clustering, partitional clustering algorithms decompose a set of objects into some prespecified number of non-overlapping clusters without the hierarchical structure (Xu and Wunsch II 2005; Jain and Dubes 1989). The classical K-means algorithm is probably the most popular technique of partitional clustering (Ng 2000). Due to its simplicity and efficiency, K-means algorithm has been

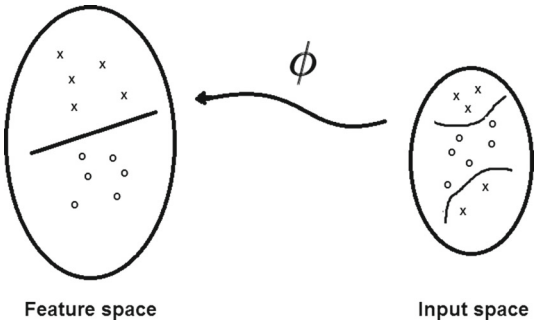

Fig. 1 Mapping from input space to feature space. The mapping function ϕ maps the data from input space to a high-dimensional feature space where the data are expected to be more separable

widely used in the literature. However, K-means has the shortcomings of dependence on the initial cluster centers and convergence to a local minima.

Distance-based clustering algorithms such as K-means or any heuristic clustering algorithm which uses distance (such as Euclidean distance or Cosine distance) as measure of the similarity between objects, are effective for data with an ellipsoidal or hyper-spherical distribution. If the separation boundaries between clusters are nonlinear, then the algorithms will fail. One of the approaches to solve this problem is to nonlinearly transform the data into a high-dimensional feature space using kernel functions or extreme learning machine (ELM), and then, perform the clustering within this feature space (Zhang and Cao 2011). Performing a nonlinear transformation of a set of nonlinearly separable patterns into a high-dimensional feature space increases the probability of the linear separability of these patterns within the transformed space, thereby simplifying the associated structure of data and enabling easier clustering (see Fig. 1) (Girolami 2002). Kernel methods have become very popular in the past few years and several clustering methods have been modified incorporating kernels (see Sect. 2). Unlike the feature mapping in the kernel-based clustering methods which is implicitly defined by the use of the kernel functions and which is not computationally efficient, the ELM feature mapping is explicit, very intuitive and straightforward. Thus compared to kernel-based clustering algorithms, clustering in ELM feature space is more convenient.

In this paper, to take advantage of the linear separability of data in the high-dimensional ELM feature space and to overcome the shortcomings of K-means algorithm, we have incorporated the ELM method into an artificial bee colony (ABC) algorithm-based clustering approach and proposed a novel ABC clustering algorithm with the ELM feature space (ELM-ABC). The ELM method is used to project the data into a high-dimensional feature space and the ABC algorithm, using the Euclidean distance in the feature space as a measure of similarity between objects, performs clustering within this feature space. Computational results show the effectiveness of our approach. Clearly, clustering is a grouping problem, i.e., a problem that seeks a partitioning of a given set of objects (instances in case of clustering) into various groups (clusters) subject to some constraints so that a given cost function is optimized (Falkenauer 1998). Recently, ABC algorithm has been applied to solve several grouping problems where it obtained better results in comparison to existing state-of-the-art metaheuristic approaches (Sundar and Singh 2010, 2014; Venkatesh and Singh 2015; Chaurasia and Singh 2015). The success of ABC algorithm in solving these grouping problems has motivated us to develop the proposed ABC approach for clustering.

The remaining part of this paper is organized as follows: Section 2 presents a brief summary of existing algorithms in the literature. Section 3 gives the necessary background for this study. Section 4 presents the ELM K-means algorithm. Section 5 presents ABC-based clustering algorithm and ELM-ABC algorithm. Experiments and results are presented and discussed in Sect. 6. Finally, Sect. 7 concludes this paper.

2 Related works

Kernel-based clustering algorithms exploit the notion that performing a nonlinear transformation of a set of nonlinearly separable patterns into a high-dimensional feature space increases the probability of the linear separability of these patterns within the transformed space, thereby improving the quality of clustering. The linear partitioning in this feature space corresponds to a nonlinear partitioning in the input space. Consequently, kernel-based clustering methods may achieve better generalization performance by working in this feature space. Various kernel-based clustering methods have been proposed in the literature. Girolami (2002) proposed a kernel method for clustering in feature space. The method also provides estimation of the possible number of clusters using the kernel matrix. Scholkopf et al. (1998) proposed kernel K-means method where the standard K-means algorithm was presented in the feature space by employing the kernel trick. The main drawbacks of the kernel K-means clustering method are the local minima and scalability as it requires computing the full kernel matrix whose size is quadratic in the number of data points. To overcome the local minima problem, Tzortzis and Likas (2009) proposed the global kernel k-means algorithm, which optimizes the clustering error in the feature space by locating near-optimal solutions. A solution to large-scale kernel clustering is presented in Zhang and Rudnicky (2002) and Chitta et al. (2011). Camastra and Verri (2005) proposed a kernel method for clustering inspired by the classical K-means algorithm and it is based on one-class support vector machine (SVM) description of a data set.

Several heuristic clustering algorithms have been introduced in the literature to overcome the shortcomings of K-means algorithm such as dependence on the initial states and convergence to local minima. These algorithms perform clustering either in the input space or in the kernel feature space. Krishna and Murty (1999) proposed a hybrid approach called genetic K-means algorithm for clustering. They have designed new operators, such as distance-based mutation operator, to achieve global search and fast convergence. Selim and Al-sultan (1991) proposed a simulated annealing approach for solving the clustering problem. A particle swarm optimization approach for data clustering is proposed by van der Merwe and Engelhrecht (2003). Shelokar et al. (2004) proposed an ant colony optimization algorithm for data clustering. Zhang and Cao (2011) proposed a novel ant-based clustering algorithm integrated with the kernel method. Karaboga and Ozturk (2010) and Zhang et al. (2010) used the ABC algorithm to solve the clustering problem. Yan et al. (2012) proposed a hybrid ABC algorithm for data clustering. In their algorithm, the crossover operator of Genetic Algorithm (GA) is introduced to enhance the information exchange between bees. The integration of kernels with K-means, fuzzy K-means, SOM, Neural gas, and one-class SVM has been shown to be effective in improving the quality of clustering (Filippone et al. 2008).

Performance of kernel-based methods depends on the choice of kernel function which is highly data specific. Most of these methods are also compute intensive, because of the need to compute full kernel matrix. In addition, the most frequently used kernel, viz. RBF kernel (Girolami 2002), requires tuning of its width. Recently, K-means algorithm has been combined with extreme learning approach (He et al. 2014; Alshamiri et al. 2014) to perform clustering which does not suffer from the drawbacks associated with the kernel-based methods. This combination (ELM K-means) has been shown to obtain better clustering performance compared to classical K-means algorithm. This is due to the possibility of linear separability of data patterns in the ELM high-dimensional feature space. Similar to classical K-means, ELM K-means has shortcomings of dependence on the initial cluster centers and convergence to local minima.

3 Basic concepts

3.1 Kernel functions

Kernel functions implicitly define some mapping function that often increases the separability of the data by nonlinearly transforming them into a high-dimensional space called *feature space*.

Suppose we are given a data set $\aleph = \{(\mathbf{x}_i) \mid \mathbf{x}_i \in \mathbf{R}^d, i = 1, \ldots, N\}$ and a mapping function ϕ that maps the data $\mathbf{x}_i$ from the input space R^d to a new feature space F. The kernel function is defined as the dot product in the feature space:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j) \tag{1}$$

Different kernel functions have a different impact on the classification results. Some commonly used kernel functions are given below:

– Polynomial of degree p:

$$K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j + 1)^p, \quad p \in N. \tag{2}$$

– Gaussian:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\| \mathbf{x}_i - \mathbf{x}_j \|^2}{2\sigma^2}\right), \quad \sigma \in R. \tag{3}$$

– Sigmoid:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(a\mathbf{x}_i \cdot \mathbf{x}_j + b), \quad a, b \in R. \tag{4}$$

3.2 Extreme learning machine

ELM is a new learning algorithm, proposed by Huang et al. (2004, 2006a, b), for single-hidden layer feedforward neural networks (SLFNs), which randomly generates hidden neurons and analytically determines the output weights of SLFNs. Unlike the traditional slow gradient-based learning algorithms [such as backpropagation algorithms (BP)] for SLFNs, which require all parameters (weights and biases) of all the layers of the feedforward networks to be tuned, ELM randomly chooses the input weights and the hidden layer biases and analytically determines the output weights of SLFNs. Input weights are the weights of the connections between input neurons and hidden neurons, and output weights are the weights of the connections between hidden neurons and output neurons. In theory, ELM algorithm tends to produce good generalization performance at extremely low computational cost. ELM has been extensively used for solving classification and regression problems. For formally defining the ELM, we will follow the same notational convention as used in Huang et al. (2006b). For instance, we are given a set of training examples $\aleph = \{(\mathbf{x}_i, \mathbf{t}_i) \mid \mathbf{x}_i \in \mathbf{R}^d, \mathbf{t}_i \in \mathbf{R}^m, i = 1, \ldots, N\}$, standard SLFNs with L hidden neurons and activation function $g(x)$, then the output of SLFNs can be represented as in Huang et al. (2006b):

$$\sum_{i=1}^{L} \beta_i g(\mathbf{w}_i \cdot \mathbf{x}_j + b_i) = \mathbf{y}_j, \quad j = 1, \ldots, N. \tag{5}$$

where $\beta_i = [\beta_{i1}, \ldots, \beta_{im}]^T$ is the weight vector connecting the ith hidden neuron and the output neurons, $\mathbf{w}_i =$

$[w_{i1}, \ldots, w_{id}]^T$ is the weight vector connecting the ith hidden neuron and the input neurons, and b_i is the bias of the ith hidden neuron. $\mathbf{w}_i \cdot \mathbf{x}_j$ is the inner product of $\mathbf{w}_i$ and $\mathbf{x}_j$. With that standard SLFNs the parameters $\beta_i, i = 1, \ldots, L$ can be estimated such that

$$\sum_{i=1}^{L} \beta_i g(\mathbf{w}_i \cdot \mathbf{x}_j + b_i) = \mathbf{t}_j, \quad j = 1, \ldots, N. \tag{6}$$

Equation (6) can be written as in Huang et al. (2006b):

$$\mathbf{H}\boldsymbol{\beta} = \mathbf{T} \tag{7}$$

where

$$\mathbf{H} = \begin{bmatrix} g(\mathbf{w}_1 \cdot \mathbf{x}_1 + b_1) & \ldots & g(\mathbf{w}_L \cdot \mathbf{x}_1 + b_L) \\ \vdots & \ldots & \vdots \\ g(\mathbf{w}_1 \cdot \mathbf{x}_N + b_1) & \ldots & g(\mathbf{w}_L \cdot \mathbf{x}_N + b_L) \end{bmatrix}_{N \times L} \tag{8}$$

$$\boldsymbol{\beta} = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_L^T \end{bmatrix}_{L \times m} \quad \text{and} \quad \mathbf{T} = \begin{bmatrix} \mathbf{t}_1^T \\ \vdots \\ \mathbf{t}_N^T \end{bmatrix}_{N \times m} \tag{9}$$

where $\mathbf{H}$ is called the hidden layer output matrix of the neural network (Huang 2003).

The main steps of the ELM algorithm are given below (Lan et al. 2010):

ELM Algorithm: Given a data set $\aleph = \{(\mathbf{x}_i, \mathbf{t}_i) \mid \mathbf{x}_i \in \mathbf{R}^d, \mathbf{t}_i \in \mathbf{R}^m, i = 1, \ldots, N\}$, activation function $g(x)$, and hidden neurons number L.

1. Randomly generate input weight $\mathbf{w}_i$ and bias b_i, $i = 1, \ldots, L$.
2. Compute the hidden layer output matrix $\mathbf{H}$.
3. Calculate the output weight $\boldsymbol{\beta} : \boldsymbol{\beta} = \mathbf{H}^{\dagger}\mathbf{T}$, where $\mathbf{H}^{\dagger}$ is the Moore–Penrose generalized inverse (Serre 2002) of the hidden layer output matrix $\mathbf{H}$ and $\mathbf{T} = [\mathbf{t}_i, \ldots, \mathbf{t}_N]^T$.

In ELM theory, the number of neurons in the hidden layer of the ELM should be large enough to achieve good generalization performance (Huang et al. 2012). A detailed discussion on hidden neurons selection in particular and ELM in general can be found in Huang et al. (2006b) and Lan et al. (2010).

3.3 K-means algorithm

K-means is an unsupervised learning algorithm that, based on some optimization measures, partitions the data set into a given number of clusters. The clustering problem, which K-means algorithm is designed to solve, can be stated as follows. Given a representation of N patterns, find K clusters based on a measure of similarity such that the patterns within a cluster are more similar to each other (high intra-cluster similarity) than patterns belonging to different clusters (low inter-cluster similarity).

Let $X = \{\mathbf{x}_i, i = 1, \ldots, N\}$ be the set of N patterns to be clustered into a set of K clusters, $C = \{c_k, k = 1, \ldots, K\}$. Typically $K \ll N$ and each pattern is a vector of dimension d ($\mathbf{x}_i \in R^d$). K-means algorithm finds a partition such that the squared Euclidean distance between the center of a cluster and the patterns in the cluster is minimized. Let μ_k be the mean of cluster c_k and it is defined as:

$$\mu_k = \frac{1}{N_k} \sum_{\mathbf{x}_i \in c_k} \mathbf{x}_i \tag{10}$$

N_k is the number of patterns in cluster c_k.

The squared error between μ_k and the patterns in cluster c_k is defined as in Jain (2010):

$$J(c_k) = \sum_{\mathbf{x}_i \in c_k} \| \mathbf{x}_i - \mu_k \|^2 \tag{11}$$

The main objective of K-means algorithm is to minimize the sum of the squared error overall K clusters

$$J(C) = \sum_{k=1}^{K} \sum_{\mathbf{x}_i \in c_k} \| \mathbf{x}_i - \mu_k \|^2. \tag{12}$$

K-means is an iterative algorithm. It starts by initializing the centers randomly. In every iteration, each pattern is assigned to its closest cluster, based on the distance between the pattern and the cluster center. The cluster centers in the next iteration are determined by computing the mean value of the patterns for each cluster. The algorithm terminates when there is no reassignment of any pattern from one cluster to another. The main steps of K-means algorithm are as follows:

1. Initialize K cluster centers.
2. Assign each pattern to its closest cluster.
3. Compute new cluster centers using (10).
4. Repeat steps 2 and 3 until there is no change for each cluster.

3.4 Artificial bee colony algorithm

The ABC algorithm, proposed by Karaboga (2005) for optimizing numerical problems, is a relatively new population-based metaheuristic technique which simulates the intelligent behavior performed by the honey bees while seeking food around their hives. In ABC algorithm, the colony of artificial bees consists of three types of bees, namely employed bees, onlookers and scouts. The employed bees are responsible for exploiting the food sources and bringing the loads

of nectar to the hive. They also gather the relevant information about these food sources such as location, quality and quantity of nectar, and share this information with the onlooker bees. Onlooker bees wait in the hive for the information to be shared by the employed bees. After getting the required information, they choose a food source to exploit with a probability directly proportional to its quality and becomes employed. Scout bees are those bees which look for new food sources in the vicinity of the hive. Once a scout bee finds a food source, it becomes employed. When a food source is exhausted, each of its associated employed bee becomes either a scout or an onlooker. Inspired by this intelligent foraging behavior Karaboga (2005) proposed the ABC algorithm. ABC algorithm also divides the (artificial) bees into same three types, viz. employed, onlooker and scout with functions similar to those explained above. In ABC algorithm, the position of a food source represents a possible solution to the problem to be optimized and the nectar amount of a food source corresponds to the quality (fitness) of the solution represented by that food source. Usually, half of the ABC colony consists of employed artificial bees and the other half contains the onlooker bees. Only one employed bee is assigned to each food source. Therefore, the number of the employed bees or the onlooker bees is equal to the number of food sources (Karaboga 2005; Karaboga and Basturk 2007). In ABC algorithm, the employed bee of an exhausted food source always becomes a scout. This scout is immediately turned into employed by generating a new food source and associating the scout with this newly generated food source.

The ABC algorithm is an iterative algorithm and starts by generating randomly distributed initial solutions (food source positions), evaluating their fitness and assigning the employed bees to the food sources. Only one employed bee is assigned to each food source. After initialization, the algorithm tries to improve the population of solutions and find the optimal by subjecting the population to repeated iterations of three search phases, viz. the employed bee phase, onlooker bee phase, and scout bee phase. A simple scheme of the original ABC algorithm is shown in Algorithm 1.

Algorithm 1: The artificial bee colony algorithm

```
Initialize the population of food sources;
Evaluate the population;
repeat
    Employed Bees Phase;
    Onlookers Bees Phase;
    Scouts Bees Phase;
    Memorize the best food source achieved so far;
until termination condition is satisfied;
```

Employed bee phase Each employed bee determines a new food source in the vicinity of its originally assigned (or old) food source. Once the new food source is determined, its nectar amount (fitness) is evaluated. If the fitness of the new food source is better than that of the old one, the employed bee replaces the old food source with the new one; otherwise, the old one is retained. The exact method of determining a new food source varies from one problem to the other and even for the same problem from one implementation of ABC algorithm to the other.

Onlooker bee phase Once all employed bees have completed the process of determining a new food source and have taken a decision to move to newly determined food source or not, they share the information of their food sources with the onlooker bees. An onlooker bee evaluates the fitness information taken from all employed bees and chooses a food source to exploit with a probability related to its fitness. The higher the fitness of the food source is, the more the probability of it being selected by the onlooker bees. Once the onlooker bee has selected her food source, she finds a new food source in the neighborhood of the selected food source. As in the case of the employed bee, if the new food source has a better quality than the old one, it will replace the old one; otherwise, the old one is retained.

Scout bee phase If the quality of the food source cannot be improved further over a predetermined number of attempts *limit*, then the food source is assumed to be exhausted and the employed bee associated with that food source abandons it to become a scout. This scout is immediately turned into employed by generating a new food source and associating the scout with this newly generated food source. Usually, the new food source for scout bee is generated randomly from scratch. However, some ABC algorithm versions generate this food source by suitably perturbing either the current or the best solution. These three phases are repeated until the termination condition is met. The main steps of the algorithm are given below:

1. Randomly generate the population of initial food sources.
2. Evaluate the population.
3. Produce new food sources for the employed bees and evaluate their quality.
4. Perform a comparison between the old and new food sources and select the better ones.
5. Calculate the probabilities of the current food sources and assign onlooker bees to them.
6. Produce new food sources for the onlooker bees and evaluate their quality.
7. Perform a comparison between the old and new food sources and select the better ones.
8. Replace the abandoned food source with the new one discovered by scout bee.
9. Memorize the best food source achieved so far.
10. If the termination condition is not met, go to step 3; otherwise, stop the algorithm.

4 ELM K-means algorithm

This section presents the basic idea behind the approaches of He et al. (2014) and Alshamiri et al. (2014) which combine the ELM and K-means algorithms. ELM performs a nonlinear transformation of the input data into a high-dimensional feature space. This transformation increases the separability of the input data in the high-dimensional feature space. The incorporation of ELM enables the K-means algorithm to explore the inherent data structure in the new space. In ELM, the hidden layer maps the data from the input space R^d to the high-dimensional feature space R^L ($L \gg d$) where the data clustering is performed. This idea of ELM mapping is similar to the idea behind the use of kernels, i.e., linearly non-separable features in the input space often become linearly separable after they are mapped to a high-dimensional feature space (see Fig. 1).

Given two data points $\mathbf{x}$ and $\mathbf{z}$ and an ELM with L neurons defining a mapping ϕ from the input space R^d to the feature space F

$$\phi\colon R^d \rightarrow F.$$

The Euclidean distance between $\mathbf{x}$ and $\mathbf{z}$ in the input space is

$$d(\mathbf{x}, \mathbf{z}) = \sqrt{\| \mathbf{x} - \mathbf{z} \|^2}. \tag{13}$$

After the points $\mathbf{x}$ and $\mathbf{z}$ are mapped into the feature space, the Euclidean distance between $\phi(\mathbf{x})$ and $\phi(\mathbf{z})$ in the feature space becomes (Zhang and Cao 2011)

$$\begin{aligned} d_F(\mathbf{x}, \mathbf{z}) &= \sqrt{\| \phi(\mathbf{x}) - \phi(\mathbf{z}) \|^2} \\ &= \sqrt{\phi(\mathbf{x}) \cdot \phi(\mathbf{x}) - 2\phi(\mathbf{x}) \cdot \phi(\mathbf{z}) + \phi(\mathbf{z}) \cdot \phi(\mathbf{z})}. \end{aligned} \tag{14}$$

Equation (13) can be replaced by Eq. (14) as the similarity measure in the clustering algorithms working in a high-dimensional feature space. Based on this principle, the ELM K-means algorithm can be visualized as mapping the data into a high-dimensional feature space and then performing clustering in the feature space.

The main steps of the ELM K-means algorithm are as follows:

ELM K-means Algorithm: Given a data set $\aleph = \{(\mathbf{x}_i) \mid \mathbf{x}_i \in \mathbf{R}^d, i = 1, \ldots, N\}$, activation function $g(x)$, and hidden neurons number L.

1. Assign arbitrary input weight $\mathbf{w}_i$ and bias b_i, $i = 1, \ldots, L$.
2. Compute the hidden layer output matrix $\mathbf{H}$.

1	2	1	2	2	1	2	2	1

Fig. 2 Solution representation

3. Apply K-means algorithm on the hidden layer output matrix $\mathbf{H}$.

where $\mathbf{H} = [h(\mathbf{x}_i), \ldots, h(\mathbf{x}_N)]^T$ and $h(\mathbf{x}) = [g(\mathbf{w}_1, b_1, \mathbf{x}), \ldots, g(\mathbf{w}_L, b_L, \mathbf{x})]$ is the output vector of the hidden layer with respect to the input $\mathbf{x}$. $h(\mathbf{x})$ actually projects the data from the d-dimensional input space into the L-dimensional hidden layer feature space $\mathbf{H}$ in which the clustering is performed.

5 Artificial bee colony algorithm-based clustering

This section presents our ELM-ABC Algorithm for clustering after describing the salient features of ABC algorithm for clustering.

5.1 Initial population

In clustering problem, the ABC algorithm generates a randomly distributed initial population of SN solutions (food source positions), where SN denotes the number of employed bees or onlooker bees. Each solution (food source) $\mathbf{s}_i (i = 1, 2, \ldots, SN)$ is a N-dimensional vector. Here, N is the number of instances in the problem. An initial solution is constructed by randomly assigning each instance to one of the K clusters. Let $\mathbf{s}_i = \{s_{i,1}, s_{i,2}, \ldots, s_{i,N}\}$ represent the ith solution in the population, then each solution is generated as follows:

$$s_{i,j} = \text{rand}(K), \tag{15}$$

where $i = 1, 2, \ldots,$ SN and $j = 1, 2, \ldots, N$. rand is a function that returns an integer uniformly at random in the interval [1, K]. Figure 2 shows the solution representation for problem with nine instances and two clusters where the instances 1, 3, 6 and 9 are assigned to cluster 1 and the instances 2, 4, 5, 7 and 8 belong to cluster 2.

5.2 Neighboring solution

The neighborhood procedure in Algorithm 2 is used to obtain a new solution $\mathbf{v}$ from the current solution $\mathbf{s}$. In this procedure, the tunable parameter cp is used to control how much percentage of the current solution can be copied to the new solution.

Algorithm 2: The neighborhood procedure

Let $\mathbf{s}_i = \{s_{i,1}, s_{i,2}, \ldots, s_{i,N}\}$ be current solution and $\mathbf{v}_i$ is its neighbor;
for $j = 1$ *to* N **do**
  if $random \leq cp$ **then**
    Copy $s_{i,j}$ to $v_{i,j}$;
  else
    Set $v_{i,j} = -1(unassigned)$;
Calculate cluster centers from assigned instances ($v_{i,j} \neq -1$);
for *each unassigned instance* $v_{i,j} == -1$ **do**
  Reassign to the closest cluster center using Euclidean distance;
if $\boldsymbol{v}_i$ *is unfeasible* **then**
  Set $\mathbf{v}_i = \mathbf{s}_i$;

5.3 ABC algorithm for clustering

The algorithm starts by generating randomly distributed initial solutions using Eq. (15) and then evaluating their fitness. In partitional clustering problem, the goal is to find a partition of the given data that minimizes (or maximizes) some criterion function. The sum of squared error function is one of the most widely used criteria. The objective function used in ABC-based clustering algorithm as defined in Eq. (12) computes the sum of squared distances between all data patterns and their associated cluster center, which reflects that data patterns within the individual cluster must be similar. The fitness of the ith solution in the population is calculated as follows:

$$\text{fit}_i = \frac{1}{1 + J_i} \tag{16}$$

where J_i is the objective function value of the ith solution in (12). Instead of using the objective function directly as a measure of fitness where the lower objective function value corresponds to a more fit solution, we have used this fitness function to follow the convention that the higher value of fitness function corresponds to a more fit solution. The term $1 + J_i$ in the denominator of the fitness function facilitates the use of this function even when J_i is zero. This fitness function is same as the one used in Karaboga and Ozturk (2010).

After associating each employed bee with an initial solution, every employed bee produces a new solution using the steps in Algorithm 2 and then evaluates its fitness. If the new solution has better fitness value than the old one, the old one is replaced by the new one. The employed bees tend to share the fitness information of their solutions with the onlooker bees. Based on this information, each onlooker bee selects a solution with a probability related to its fitness value. The probability of choosing a solution i is defined as follows:

$$p_i = \frac{\text{fit}_i}{\sum_{i=1}^{\text{SN}} \text{fit}_i} \tag{17}$$

As in the case of the employed bee, each onlooker bee produces a new solution, evaluates its fitness and applies a greedy selection on the new and old solutions. The solution, the fitness value of which is not improved after performing a predetermined number of trials *limit*, is abandoned, and the associated employed bee becomes a scout. The scout produces a new solution randomly using Eq. (15). The process of abandoning and replacing an exhausted food source (solution) enables the algorithm to avoid suboptimal solutions. The search processes of the employed, onlooker and scout bees are repeated until the termination condition is met. The pseudo-code of the ABC algorithm for clustering is given in Algorithm 3.

Algorithm 3: The pseudo-code of the ABC algorithm for clustering

Initialize the population of solutions $s_{i,j}, i = 1, 2, \ldots, \text{SN}, j = 1, 2, \ldots, N$. Assign employed bees to the solutions;
Evaluate the fitness $\text{fit}(\mathbf{s}_i)$ of the population $\mathbf{s}_i, i = 1, 2, \ldots, \text{SN}$;
Set $trial_i = 0, i = 1, 2, \ldots, \text{SN}$;
repeat
  /* Employed bees phase */
  for $i = 1$ *to* SN **do**
    Produce a new solution $\mathbf{v}_i$ from $\mathbf{s}_i$ using Algorithm 2;
    Evaluate the fitness of the new solution using Eq. (16);
    if $\text{fit}(\boldsymbol{v}_i) > \text{fit}(\boldsymbol{s}_i)$ **then**
      Replace s_i with v_i and $trial_i = 0$;
    else
      $trial_i = trial_i + 1$;
  Calculate the probability values p_i for the solutions using Eq. (17);
  /* Onlooker bees phase */
  for *each onlooker bee* **do**
    Select a solution $\mathbf{s}_i$ depending on p_i;
    Produce a new solution $\mathbf{v}_i$ from $\mathbf{s}_i$ using Algorithm 2;
    Evaluate the fitness of the new solution using Eq. (16);
    if $\text{fit}(\boldsymbol{v}_i) > \text{fit}(\boldsymbol{s}_i)$ **then**
      Replace s_i with v_i and $trial_i = 0$;
    else
      $trial_i = trial_i + 1$;
  /* Scout bees phase */
  for *each solution* $\boldsymbol{s}_i$ **do**
    if $trial_i > limit$ **then**
      Replace $\mathbf{s}_i$ with a randomly produced solution using Eq. (15);
  Memorize the best solution achieved so far;
until *termination condition is satisfied*;

5.4 Proposed ELM-ABC algorithm for clustering

The ABC algorithm presented in Algorithm 3 is a distance-based clustering algorithm. As already mentioned in Section

1, the distance-based clustering algorithm may get stuck in suboptimal solutions if the separation boundaries between clusters are nonlinear (Girolami 2002). In ABC algorithm, the process of abandoning and replacing non-improving solution is one way to avoid suboptimal solutions. The other way is to project the input data into a high-dimensional space and then perform clustering in that space. The ELM hidden layer is used to nonlinearly transform the input data into a high-dimensional space called ELM feature space. This transformation often increases the separability of the input data in the ELM feature space. The combined algorithm will be referred to as ELM-ABC algorithm subsequently. The pseudo-code of the ELM-ABC algorithm is given in Algorithm 4.

Algorithm 4: The pseudo-code of the ELM-ABC algorithm

Given a data set $\aleph = \{(\mathbf{x}_i) \mid \mathbf{x}_i \in \mathbf{R}^d, i = 1, \ldots, N\}$, activation function $g(x)$, and hidden neurons number L;
Randomly generate input weight $\mathbf{w}_i$ and bias $b_i, i = 1, \ldots, L$;
Compute the hidden layer output matrix $\mathbf{H}$;
Apply Algorithm 3 on the hidden layer output matrix $\mathbf{H}$;

6 Experimental study

In this paper, 12 benchmark data sets are used to evaluate the performance of the proposed ELM-ABC algorithm. These data sets, except USPST data set, can be downloaded from the UCI Machine Learning Repository[1]. The data sets and their characteristics, viz. the number of patterns, the number of features and the number of classes are given in alphabetical order in Table 1.

6.1 Data sets

The data sets considered in this work can be described briefly as follows. Balance data set was generated to model psychological experimental results. Each pattern is classified as having the balance scale tip to the right, tip to the left, or remains exactly in the middle. The data set includes 4 features and 3 classes, and there are 625 patterns. Cancer-Diagnostic and Cancer-Original data sets are based on the "breast cancer Wisconsin-Diagnostic" and "breast cancer Wisconsin-Original" data sets, respectively. Both data sets classify a tumor as either benign or malignant. Cancer-Diagnostic data set contains 569 patterns and 30 features. Cancer-Original data set contains 699 patterns. After removing the 16 database patterns with missing values, the database consists of 683 patterns and 9 features.

[1] http://www.ics.uci.edu/~mlearn/MLRepository.html.

Table 1 Data sets characteristics

Data sets	Patterns	Features	Classes
Balance	625	4	3
Cancer-Diagnostic	569	30	2
Cancer-Original	683	9	2
Cardiotocography-3	2126	21	3
Cardiotocography-10	2126	21	10
CNAE	1080	856	9
Dermatology	358	34	6
Glass	214	9	6
Iris	150	4	3
LIBRAS	360	90	15
Spam	1534	57	2
USPST	2007	256	10

Cardiotocography data set consists of measurements of fetal heart rate (FHR) and uterine contraction (UC) features on cardiotocograms classified by expert obstetricians. Classification was both with respect to a morphologic pattern (10 classes: 1–10) and to a fetal state (three classes: normal, suspect and pathologic). Therefore, the data set can be used either for 10-class (Cardiotocography-10) or 3-class (Cardiotocography-3) experiments. The data set consists of 2126 21-dimensional patterns. CNAE data set contains 1080 documents of free-text business descriptions of Brazilian companies categorized into a subset of 9 categories. The original texts were pre-processed so that each document can be represented as a vector, where the weight of each word is its frequency in the document. This data set is highly sparse (99.22 % of the matrix is filled with zeros). Dermatology data set aims to determine the type of Erythemato-Squamous Disease. The data set contains 366 patterns. After the removal of the 8 data set patterns with missing values, the data set consists of 358 34-dimensional patterns belonging to six different classes. Glass data set contains 214 patterns, 9 features and 6 glass types. The glass types are float processed building windows, non-float processed building windows, vehicle windows, containers, tableware and head lamps. The Fisher Iris data (Fisher 1936) are one of the most popular data sets to test the performance of novel methods in pattern recognition and machine learning. There are three classes in this data set (Setosa, Versicolor and Virginica), each having 50 patterns with four features (sepal length, sepal width, petal length and petal width). One of the classes (viz. Setosa) is linearly separable from the other two, while the remaining two are not linearly separable (see Fig. 3, in which only two features are used). LIBRAS data set is based on the Libras Movement data set. The data set contains 15 classes of 24 patterns each. Each class references to a hand movement type in LIBRAS (Portuguese name 'LÍngua BRAsileira de

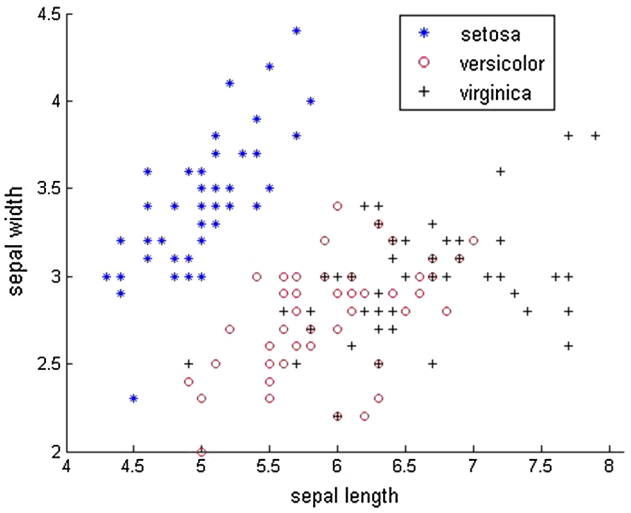

Fig. 3 Iris data set. There are three classes, Setosa class is linearly separable from the other two classes. Versicolor and Virginica Classes are not linearly separable

Sinais', oficial brazilian sign language). The Spam data set consists of 1534 patterns from two different classes, spam and not-spam. Each pattern is represented by a 57-dimensional feature vector. The USPST data set is a subset (the testing set) of the well-known handwritten digit recognition data set USPS. The data set includes 256 features and 10 classes, and there are 2007 patterns.

6.2 Results and discussion

The proposed ELM-ABC algorithm is tested on 12 benchmark data sets and its performance is compared with the K-means, ELM K-means and ABC algorithms. While comparison with K-means provides a baseline, comparing with ELM K-means enables assessment of advantages of metaheuristic-based clustering approach. We used the sigmoid function for nonlinear mapping and the number of ELM hidden neurons was set to 1000 for all the data sets. The input weights and biases were randomly generated from a uniform distribution over $[-1, 1]$. For ABC and ELM-ABC, the number of employed bees and the number of onlookers are set to be equal to the number of food sources (SN), which is 10, the *limit* = 10 and the value of the *cp* is 0.7. Both ABC and ELM-ABC algorithms terminate if they have been executed for 1000 iterations or if the best solution found so far is not improved after performing a predetermined number of iterations *glimit*. We set *glimit* = 200. We have executed all algorithms 20 times independently. All the simulations are carried out in MATLAB environment running in Core i5-2400, 3.10 GHZ CPU with 4 GB RAM.

For each data set, we report the average of correctly clustered patterns (ACC) which is defined as

$$\text{ACC} = \frac{\sum_{i=1}^{\#\text{ of runs}} \#\text{ of correctly clustered patterns}}{\#\text{ of runs}}. \quad (18)$$

We also report the percentage of average correct clustering (PACC) which is the average of correctly clustered patterns (ACC) percentaged to the size of the data set.

$$\text{PACC} = 100 \times \frac{\text{ACC}}{\text{size of data set}} \quad (19)$$

The PACC values are shown in parenthesis in Table 2. The average performance, in terms of correctly clustered patterns, of these algorithms is reported in Table 2. From the results, we can observe that ELM-ABC algorithm has obtained satisfying results on all the data sets. It yielded the best clustering performance among the four algorithms on 10 out of 12 data sets. ELM-ABC algorithm, on some data sets, has obtained similar results as ELM K-means algorithm which may be due to the use of the same ELM feature space. However, on other data sets, such as Dermatology, Iris, Spam and USPST, ELM-ABC obtains significantly better results than ELM K-means. This may be due to the fact that ELM-ABC algorithm provides a way of avoiding suboptimal solutions through the use of *scout mechanism*. For CNAE data set, the standard deviation is high for all algorithms. This can be attributed to the presence of outliers as the data set is highly sparse. The results obtained by ELM K-means and ELM-ABC algorithms demonstrate the advantages of performing clustering in the ELM high-dimensional feature space. These results are also in line with the concept that the data which are not linearly separable in the input space often become separable in high-dimensional space which enables the clustering algorithms to achieve significantly better performance than that in the input space. Figures 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, and 15 show the clustering performance (PACC) of ELM K-means and ELM-ABC with different number of hidden neurons [50, 100, . . . , 1000]. From these results, we can observe the following:

- In most cases, the highest clustering performance (PACC) obtained by ELM-ABC is generally higher than that of ELM K-means.
- The ELM-ABC performance dominated ELM K-means for different choices of number of ELM hidden neurons.
- The performance of ELM-ABC and ELM K-means reaches steady-state after the number of ELM neurons is 500 and above for most data sets.

To see how the time increases as more patterns are considered, we have conducted an experiment on USPST data set with different number of patterns. Figure 16 shows the time

Table 2 Average performance, in terms of correctly clustered patterns, of the K-means, ELM K-means, ABC and ELM-ABC algorithms

Data sets	K-means	ELM K-means	ABC	ELM-ABC
Balance	419.3 ± 24 (67.09 %)	434.2 ± 16.84 (69.47 %)	407.9 ± 7.23 (65.26%)	429.5 ± 1.82 (68.72 %)
Cancer-Diagnostic	528 ± 0 (92.79 %)	536 ± 0 (94.2 %)	528 ± 0 (92.79%)	536 ± 0 (94.2 %)
Cancer-Original	656 ± 0 (96.05 %)	659 ± 0 (96.49 %)	656 ± 0 (96.05 %)	659 ± 0 (96.49 %)
Cardiotocography-3	1655 ± 0 (77.85 %)	1666.9 ± 16.64 (78.41 %)	1655 ± 0 (77.85 %)	1655 ± 0 (77.85 %)
Cardiotocography-10	984.35 ± 35.73 (46.3 %)	1022.35 ± 45.88 (48.09 %)	1024 ± 5 (48.17 %)	1048.5 ± 2.3 (49.3 %)
CNAE	527.6 ± 55.13 (48.85 %)	602.25 ± 56.95 (55.76 %)	496.35 ± 31.60 (45.96 %)	602.45 ± 40.71 (55.78 %)
Dermatology	296.2 ± 20.54 (82.74 %)	304.15 ± 22.78 (84.96 %)	310 ± 0 (86.59 %)	315.2 ± 10.67 (88.04 %)
Glass	122.5 ± 3.8 (57.24 %)	126.15 ± 8.7 (58.95 %)	126 ± 0 (58.89 %)	127 ± 0 (59.35 %)
Iris	128.9 ± 12.46 (85.93 %)	129.9 ± 22.51 (86.6 %)	134 ± 0 (89.33 %)	146 ± 0 (97.33 %)
LIBRAS	168.1 ± 7.5 (46.69 %)	176.3 ± 6.89 (48.97 %)	164.35 ± 1.14 (45.65 %)	179.55 ± 2.26 (49.88 %)
Spam	1097.85 ± 155.77 (71.57 %)	1101.65 ± 158.08 (71.82 %)	930 ± 0 (60.62 %)	1241 ± 0 (80.9 %)
USPST	1365.45 ± 46.09 (68.03 %)	1425.4 ± 76.51 (71.02 %)	1389.4 ± 8 (69.23 %)	1486.3 ± 5.8 (74.06 %)

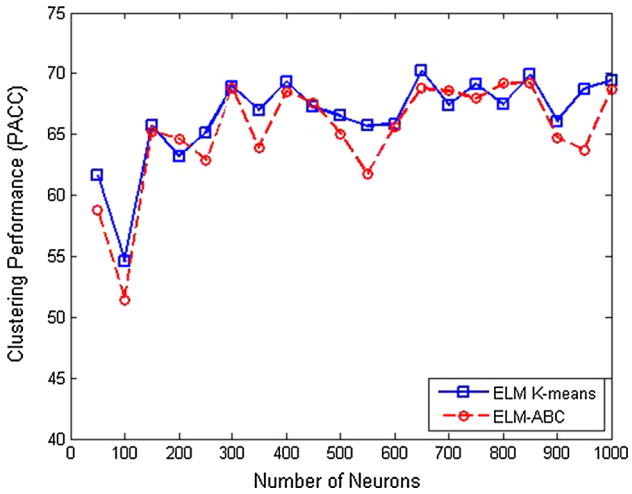

Fig. 4 Clustering performance on Balance with respect to different number of neurons

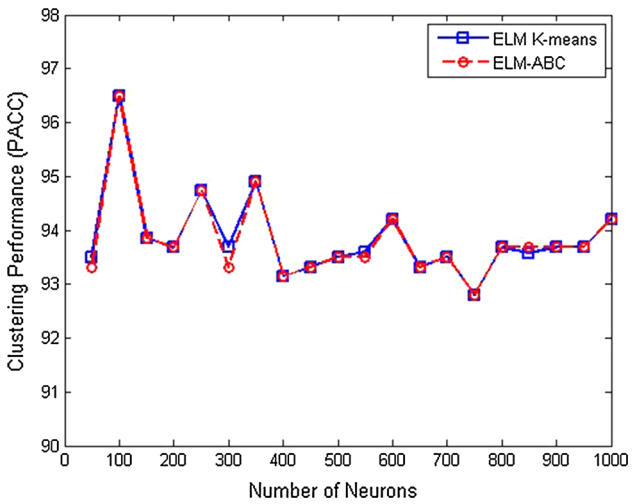

Fig. 5 Clustering performance on Cancer-Diagnostic with respect to different number of neurons

spent by ELM-ABC algorithm on USPST data set with different number of patterns [100, 200, . . . , 1000, 1200, . . . , 1800, 2007]. The number of hidden neurons in ELM-ABC is fixed to 1000. The graph indicates approximately linear increase in time with respect to number of patterns.

6.3 Computational efficiency

In this paper, the idea is to perform clustering in ELM feature space. In projecting the data into a high-dimensional ELM feature space, there is a time cost, and in this section, we evaluate the time–accuracy tradeoff. ELM K-means and ELM-ABC are the variations of K-means and ABC algorithms, respectively. Direct comparison of absolute running times between these two algorithms would be rather unfair because metaheuristic techniques are known to be compute

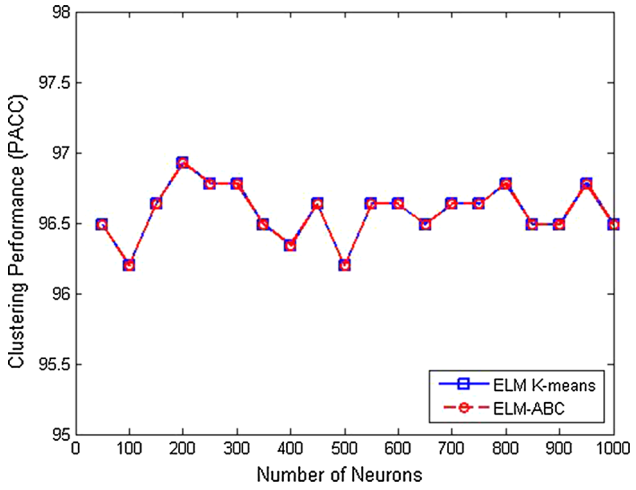

Fig. 6 Clustering performance on Cancer-Original with respect to different number of neurons

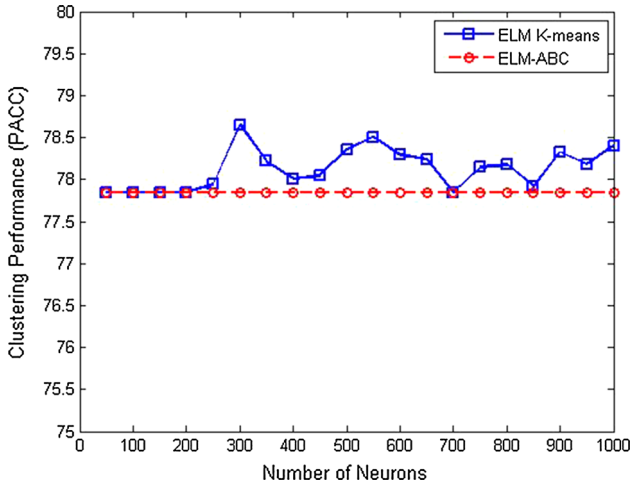

Fig. 7 Clustering performance on Cardiotocography-3 with respect to different number of neurons

intensive. In addition, the proposed ELM-ABC approach is not for realtime clustering tasks, but for offline clustering. Therefore, instead of making a direct absolute time comparison, we will compare the time–accuracy tradeoff of both algorithms. Table 3 shows time–accuracy tradeoff of ELM-ABC and ELM K-means algorithms, where NPIA is normalized percentage improvement in accuracy which is defined as:

$$\text{NPIA} = 100 \times \frac{\text{PACC in ELM space} - \text{PACC in input space}}{\text{PACC in input space}} \tag{20}$$

and NCT is normalized change in time which is defined as:

$$\text{NCT} = \frac{\text{Time spent in ELM space} - \text{Time spent in input space}}{\text{Time spent in input space}} \tag{21}$$

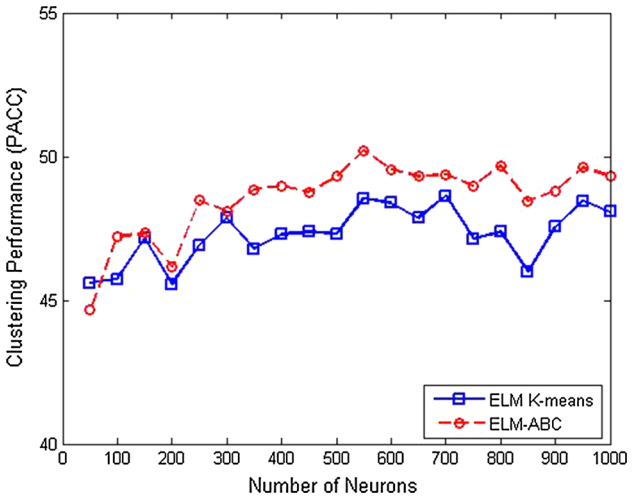

Fig. 8 Clustering performance on Cardiotocography-10 with respect to different number of neurons

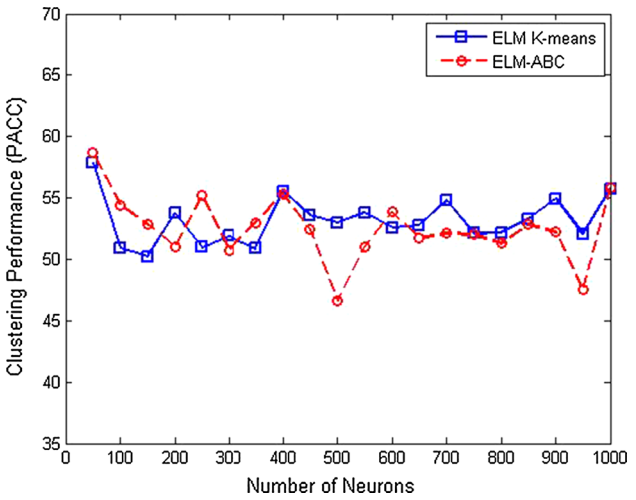

Fig. 9 Clustering performance on CNAE with respect to different number of neurons

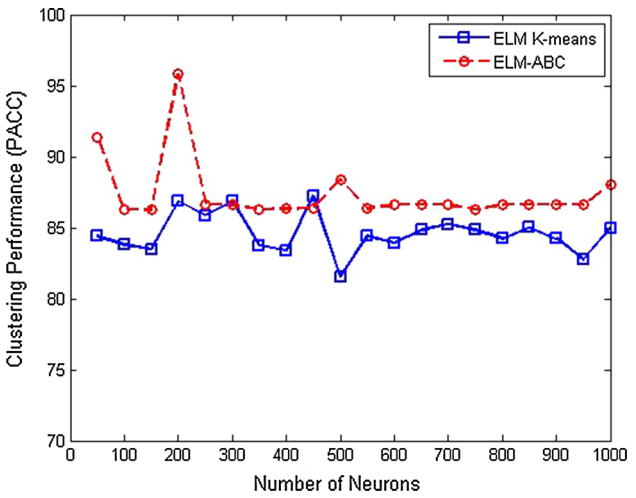

Fig. 10 Clustering performance on Dermatology with respect to different number of neurons

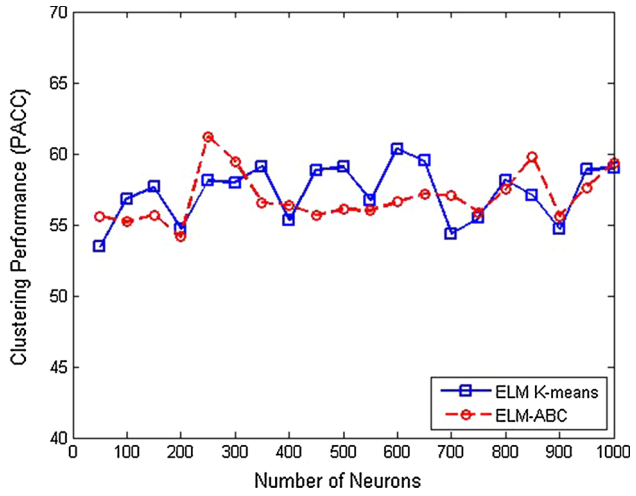

Fig. 11 Clustering performance on Glass with respect to different number of neurons

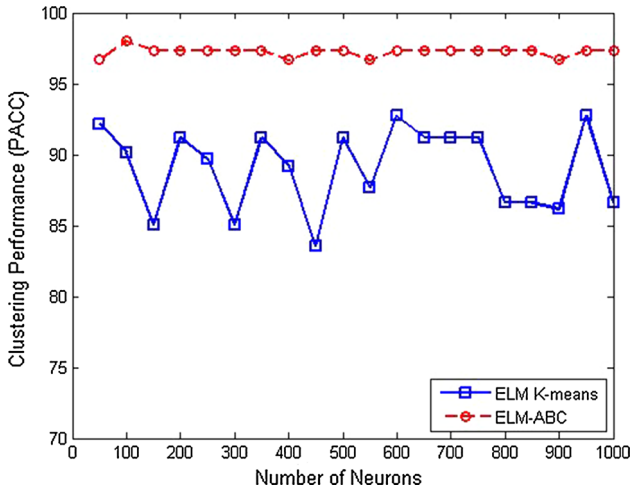

Fig. 12 Clustering performance on Iris with respect to different number of neurons

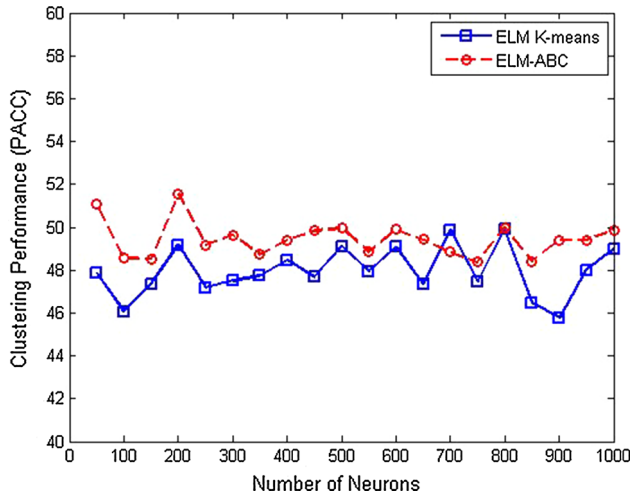

Fig. 13 Clustering performance on LIBRAS with respect to different number of neurons

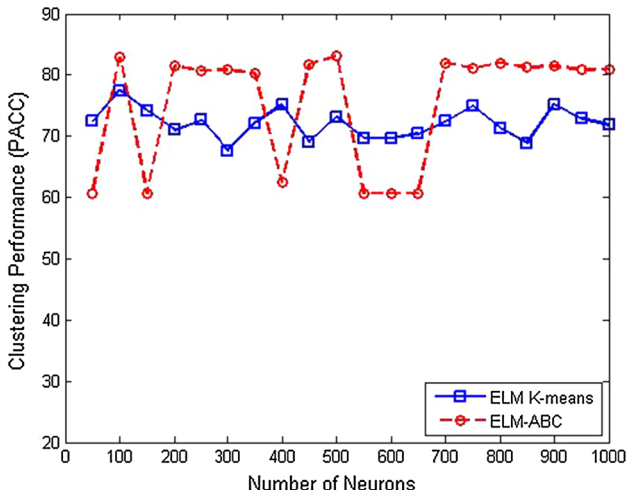

Fig. 14 Clustering performance on Spam with respect to different number of neurons

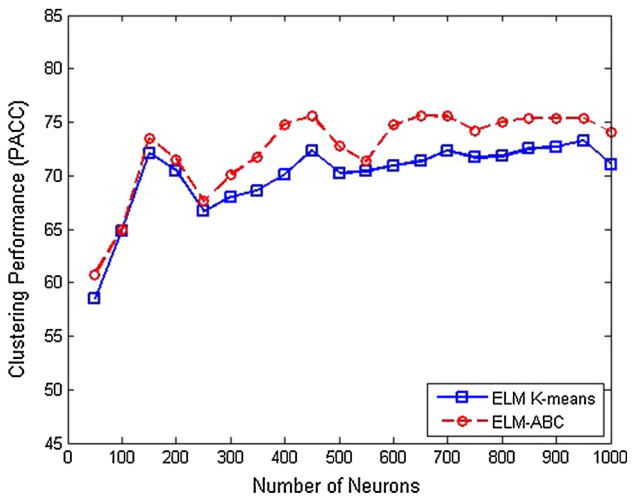

Fig. 15 Clustering performance on USPST with respect to different number of neurons

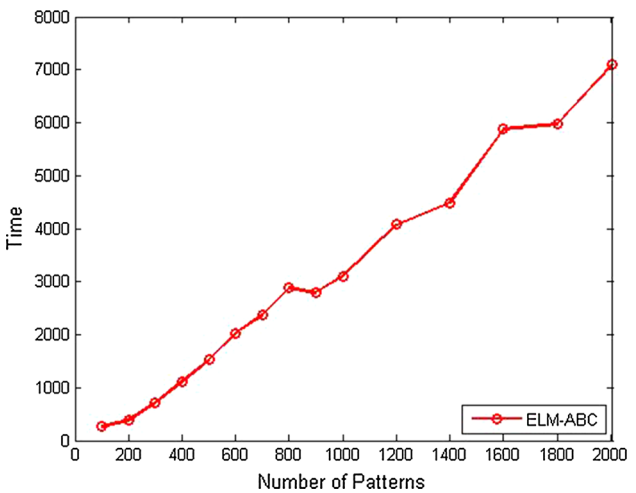

Fig. 16 Execution time of ELM-ABC on USPST with respect to different number of patterns

Table 3 Time–accuracy tradeoff of ELM-ABC and ELM K-means

Data sets	ELM-ABC		ELM K-means	
	NPIA	NCT	NPIA	NCT
Balance	5.3	24.61	3.55	62.81
Cancer-Diagnostic	1.52	17.98	1.52	14.54
Cancer-Original	0.46	20.35	0.46	11.29
Cardiotocography-3	0	19.62	0.72	40.26
Cardiotocography-10	2.39	30.1	3.87	114.58
CNAE	21.37	0.11	14.15	2.2
Dermatology	1.67	19.17	2.68	17.45
Glass	0.78	14.3	2.99	11.48
Iris	8.96	9.57	0.78	3.33
LIBRAS	9.27	11.25	4.88	15.31
Spam	33.45	11.79	0.35	4.6
USPST	6.98	5.28	4.4	3.41
Average	7.68	15.34	3.36	25.11

The NPIA and NCT of ELM-ABC are with respect to ABC algorithm and those for ELM K-means are with respect to K-means algorithm. From the Table 3, we can see that ELM combined with ABC improves the clustering accuracy on average by 7.68 %, but with average NCT of 15 units, whereas in ELM K-means, the improvement in accuracy on average is 3.36 % at the average NCT of 25 units. The NCT of ELM-ABC on average is about two-thirds of that of ELM K-means, while the clustering accuracy improvement is double. So ELM-ABC has a better time–accuracy tradeoff in comparison to ELM K-means.

To further verify the effectiveness of the proposed ELM-ABC, the two-tailed t-test has been conducted at the 5 % significance level to compare proposed ELM-ABC with three other algorithms. The results of the t-tests are shown in Table 4. The N/A in Table 4 stands for not applicable, covering those cases for which both compared algorithms have zero standard deviation. The t-test results show that the difference between the ELM-ABC and the other algorithms is statistically significant in most cases. If we look in Table 2 for results corresponding to 11 N/A cases in Table 4, we can observe that there are seven cases where ELM-ABC approach obtained better results. For example, on Cancer-Diagnostic data set, we can not perform t-test between ABC and ELM-ABC as both the approaches have standard deviation of zero. However, on this data set, ELM-ABC correctly classifies 536 instances in each of the 20 runs, whereas ABC correctly classifies only 528 instances in each of the 20 runs. As in this case and in other six cases, results are obtained with standard deviation of zero; therefore, it is extremely improbable that randomness has any role in the better performance of ELM-ABC over the other method in consideration, and hence, the result of ELM-ABC in these seven cases can also be considered significant.

Table 4 P-values from two-sample t-tests of ELM-ABC against the other techniques

Data sets	K-means	ELM K-means	ABC
Balance	0.0657	0.2222	<0.0001
Cancer-Diagnostic	N/A	N/A	N/A
Cancer-Original	N/A	N/A	N/A
Cardiotocography-3	N/A	0.0028	N/A
Cardiotocography-10	<0.0001	0.0151	<0.0001
CNAE	<0.0001	0.9899	<0.0001
Dermatology	0.0007	0.0568	0.0356
Glass	<0.0001	0.6646	N/A
Iris	<0.0001	0.0028	N/A
LIBRAS	<0.0001	0.0522	<0.0001
Spam	0.0002	0.0003	N/A
USPST	<0.0001	0.001	<0.0001

7 Conclusion

Recently, encouraging results of clustering performance of K-means algorithm have been obtained using the ELM high-dimensional feature space. ELM K-means algorithm is also limited by its dependence on the choice of initial cluster center locations and convergence to (suboptimal) local minima, the problems that usually plague conventional algorithms such as K-means. To overcome these limitations, in this paper we combine the ELM approach with ABC algorithm for unsupervised clustering. We have evaluated ELM K-means and ELM-ABC on wide range of real-world benchmark data sets. Experimental results show that ELM-ABC gives significantly better time–accuracy tradeoff compared to ELM K-means algorithm. These results also demonstrate that the integration of ELM method with ABC algorithm improves the quality of clustering performed by the ABC algorithm itself.

As a future work, we intend to incorporate the ELM method into some other metaheuristic techniques such as genetic algorithm and particle swarm optimization, and compare the performance of the resulting methods with the ELM-ABC algorithm. There is also scope for integrating ABC algorithm with some kernel methods such as RBF kernel and comparing the clustering performance of ABC algorithm in ELM and kernel feature space.

References

Alshamiri AK, Singh A, Surampudi BR (2014) A novel elm k-means algorithm for clustering. In: Proceedings of 5th joint international conference on swarm, evolutionary and memetic computing (SEMCCO 2014) and fuzzy and neural computing (FANCCO 2014), Odisha, India (To appear)

Camastra F, Verri A (2005) A novel kernel method for clustering. IEEE Trans Pattern Anal Mach Intell 27(5):801–805
Chaurasia SN, Singh A (2015) A hybrid swarm intelligence approach to the registration area planning problem. Inform Sci 302:50–69
Chitta R, Jin R, Havens TC, Jain AK (2011) Approximate kernel k-means: solution to large scale kernel clustering. In: Proceedings of 17th ACM SIGKDD international conference on knowledge discovery and data mining (KDD), New York, USA, pp 895–903
Falkenauer E (1998) Genetic algorithm and grouping problems. Wiley, New York
Filippone M, Camastra F, Masulli F, Rovetta S (2008) A survey of kernel and spectral methods for clustering. Pattern Recognit 41:176–190
Fisher RA (1936) The use of multiple measurements in taxonomic problems. Ann Eugen 7:179–188
Girolami M (2002) Mercer kernel based clustering in feature space. IEEE Trans Neural Netw 13(3):780–784
Han J, Kamber M (2001) Data mining: concepts and techniques. Academic Press, San Diego
He Q, Jin X, Du C, Zhuang F, Shi Z (2014) Clustering in extreme learning machine feature space. Neurocomputing 128:88–95
Huang G-B (2003) Learning capability and storage capacity of two-hidden-layer feedforward networks. IEEE Trans Neural Netw 14(2):274–281
Huang G-B, Zhu Q-Y, Siew C-K (2004) Extreme learning machine: a new learning scheme of feedforward neural networks. In: Proceedings of international joint conference on neural networks (IJCNN), vol 2. Budapest, Hungary, pp 985–990
Huang G-B, Chen L, Siew C-K (2006a) Universal approximation using incremental constructive feedforward networks with random hidden nodes. IEEE Trans Neural Netw 17(4):879–892
Huang G-B, Zhu Q-Y, Siew C-K (2006b) Extreme learning machine: theory and applications. Neurocomputing 70:489–501
Huang G-B, Zhou H, Ding X, Zhang R (2012) Extreme learning machine for regression and multiclass classification. IEEE Trans Syst Man Cybern B Cybern 42(2):513–529
Jain AK (2010) Data clustering: 50 years beyond k-means. Pattern Recognit 31:651–666
Jain AK, Dubes RC (1989) Algorithms for clustering data. Prentice-Hall, Englewood Cliffs
Jain AK, Murty MN, Flynn PJ (1999) Data clustering: a review. ACM Comput Surv 31(3):264–323
Karaboga D (2005) An idea based on honey bee swarm for numerical optimization. In: Technical report-TR06. Erciyes University, Engineering Faculty, Computer Engineering Department, Turkey
Karaboga D, Basturk B (2007) A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm. J Glob Optim 39(3):459–471
Karaboga D, Ozturk C (2010) A novel clustering approach: artificial bee colony (abc) algorithm. Appl Soft Comput 11:652–657
Krishna K, Murty MN (1999) Genetic k-means algorithm. IEEE Trans Syst Man Cybern B Cybern 29(3):433–439
Lan Y, Soh YC, Huang G-B (2010) Constructive hidden nodes selection of extreme learning machine for regression. Neurocomputing 73:3191–3199
Ng MK (2000) A note on constrained k-means algorithms. Pattern Recogn 33:515–519
Scholkopf B, Smola A, Muller K (1998) Nonlinear component analysis as a kernel eigenvalue problem. Neural Comput 10(5):1299–1319
Selim SZ, Al-sultan K (1991) A simulated annealing algorithm for the clustering problems. Pattern Recogn 24(10):1003–1008
Serre D (2002) Matrices: theory and applications. Springer, NewYork
Shelokar P, Jayaraman V, Kulkarni B (2004) An ant colony approach for clustering. Analytica Chimica Acta 509:187–195
Sundar S, Singh A (2010) A swarm intelligence approach to the quadratic multiple knapsack problem. In: Proceedings of the 17th international conference on neural information processing (ICONIP 2010). Lecture notes in computer science, vol 6443, pp 626–633
Sundar S, Singh A (2014) Metaheuristic approaches for the blockmodel problem. IEEE Syst J. doi:10.1109/JSYST.2014.2342931
Tzortzis GF, Likas AC (2009) The global kernel k-means algorithm for clustering in feature space. IEEE Trans Neural Netw 20(7):1181–1194
van der Merwe D, Engelhrecht A (2003) Data clustering using particle swarm optimization. In: Proceedings of IEEE congress on evolutionary computation (CEC 03). Canbella, Australia, pp 215–220
Venkatesh P, Singh A (2015) Two metaheuristic approaches for the multiple traveling salesperson problem. Appl Soft Comput 26:74–89
Xu R, Wunsch II D (2005) Survey of clustering algorithms. IEEE Trans Neural Netw 16(3):645–678
Yan X, Zhu Y, Zou W, Wang L (2012) A new approach for data clustering using hybrid artificial bee colony algorithm. Neurocomputing 97:241–250
Zhang L, Cao Q (2011) A novel ant-based clustering algorithm using the kernel method. Inform Sci 181:4658–4672
Zhang R, Rudnicky AI (2002) A large scale clustering scheme for kernel k-means. In: Proceedings of 16th international conference on pattern recognition (ICPR), vol 4, Quebec, Canada, pp 289–292
Zhang C, Ouyang D, Ning J (2010) An artificial bee colony approach for clustering. Expert Syst Appl 37:4761–4767