Information Sciences 181 (2011) 4658–4672

Contents lists available at ScienceDirect

Information Sciences

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

A novel ant-based clustering algorithm using the kernel method

Lei Zhang *, Qixin Cao

Research Institute of Robotics, State Key Lab of Mechanical System and Vibration, Shanghai Jiao Tong University, No. 800 Dongchuan Road, Shanghai 200240, China

ARTICLE INFO

Article history:
Available online 12 November 2010

Keywords:
Ant-based clustering
Kernel
Swarm intelligence
Kernel principal component analysis

ABSTRACT

A novel ant-based clustering algorithm integrated with the kernel (ACK) method is proposed. There are two aspects to the integration. First, kernel principal component analysis (KPCA) is applied to modify the random projection of objects when the algorithm is run initially. This projection can create rough clusters and improve the algorithm's efficiency. Second, ant-based clustering is performed in the feature space rather than in the input space. The distance between the objects in the feature space, which is calculated by the kernel function of the object vectors in the input space, is applied as a similarity measure. The algorithm uses an ant movement model in which each object is viewed as an ant. The ant determines its movement according to the fitness of its local neighbourhood. The proposed algorithm incorporates the merits of kernel-based clustering into ant-based clustering. Comparisons with other classic algorithms using several synthetic and real datasets demonstrate that ACK method exhibits high performance in terms of efficiency and clustering quality.

1. Introduction

Clustering is a method that divides a dataset into groups of similar objects, thereby minimizing the similarities between different clusters and maximizing the similarities between objects in the same cluster. Clustering is widely applied in data mining, such as in document clustering and Web analysis. Classic clustering approaches include partition-based methods, such as *K*-means, *K*-medoids, and *K*-prototypes [20,22]; hierarchy-based methods, such as BIRCH [44]; density-based methods [1,10]; grid-based methods [43]; and model-based methods, such as neural networks and self-organizing map (SOM) [4,30].

Recently, ant-based clustering, which is a type of clustering algorithm that imitates the behaviour of ants, has earned researchers' attention. Ant-based clustering can be divided into two classes. The first class imitates the ant's foraging behaviour, which involves finding the shortest route between a food source and the nest. This intelligent behaviour is achieved by means of pheromone trails and information exchange between ants. Shelokar et al. [34] and Chen et al.'s [5] proposed algorithms belong to this class. These algorithms treat clustering as an optimization task and utilize ant colony optimization (ACO) methods to obtain optimal clusters. An extension of ACO, called constrained ACO (C-ACO) [6], was suggested to cluster data involving arbitrary shapes or outliers. A variant of ACO, called the aggregation pheromone density-based clustering algorithm (APC), was also suggested [12,13]. Similar to ACO, APC is based on the aggregation pheromones found in ants. The advantage of these methods is that the objective function is explicit. The key elements of these algorithms are the pheromone matrix updating rule and the heuristic function.

The second class imitates ants' behaviour of clustering their corpses and forming cemeteries. Some ants can pick up dead bodies randomly distributed in the nest and group them into different sizes. The large group of bodies attracts the ants to

* Corresponding author.
E-mail addresses: zhanglei75@sina.com, zhanglei@sjtu.edu.cn (L. Zhang).

0020-0255/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.ins.2010.11.005

deposit more dead bodies and becomes larger and larger. The essence of this phenomenon is positive feedback [25]. One of the first studies related to this domain is the work of Deneubourg [9], who came up with the basic model (BM) to explain the ants' movement. In the BM, the ants move randomly and pick up or drop objects according to the number of similar surrounding objects to cluster them. Lumer and Faieta [24] extended the model and applied it to data analysis (they called this the LF algorithm). In their analysis, an object with n attributes can be viewed as a point in the R^n space. The point is projected into a low-dimensional space (often a two-dimensional plane). The similarity of the object with those in the local neighbourhood is calculated to determine whether the object should be picked up or dropped by ants. As a basic algorithm, LF was followed and improved by a number of modified algorithms in different applications. Wu et al. [37,38] further explained the idea of the similarity coefficient and suggested a more simple probability conversion function. Ramos and Merelo [29] studied ant-based clustering with different ant speeds to cluster text documents. Yan et al. [40,41] suggested multiple ant colonies consisting of independent colonies and a queen ant agent. Each ant colony had a different moving speed and probability conversion function. The hypergraph model was used to combine the results of all parallel ant colonies.

In addition to the above-mentioned studies, a series of research by Handl deserves special attention. She came up with a set of strategies for increasing the robustness of the LF algorithm and applying it to document retrieval [15]. She performed a comparative study of ant-based clustering with *K*-means, average links, and 1d-SOM [16,17]. An improved version, ATTA, which incorporates adaptive and heterogeneous ants and time-dependent transporting activity, was proposed in her latest paper [18]. The main feature of this kind of algorithm is that the algorithm directly imitates the ant's behaviour to cluster data and the clustering objective is implicitly defined [19].

Beyond the two classes of ant-based clustering, Tsang and Kwong [36] proposed ant colony clustering for anomaly intrusion detection. This method integrates the characteristics of the two above-mentioned classes. Specifically, cluster formation and searching for an object are regarded as nest building and food foraging, respectively. The ants exhibit picking up and dropping behaviours while simultaneously depositing cluster-pheromones on the grid. Xu et al. [39] suggested a novel ant movement model wherein each object was viewed as an ant. The ant determines its behaviour according to the fitness of its local neighbourhood. Essentially, this model is similar to that in the second class of ant-based clustering.

Combinations of ant-based clustering with other clustering methods can also be found. For example, ant-based clustering has been combined with *K*-means [26] and with *K*-harmonic means [21]; ant colonies have been hybridized with fuzzy *C*-means [28,42]; fuzzy ants have been endowed with intelligence in the form of IF–THEN rules [23]; and the hybrid approach has been generated based on particle swarm optimization (PSO), ACO, and *K*-means [27]. In these methods, the role of ant-based clustering is mainly to create initial clusters for other clustering algorithms.

A general simulation of swarm and collective intelligence has been described [33] as well as a comprehensive overview of ant-based and swarm-based clustering [19].

Our particular interest is in the second kind of ant-based clustering discussed above. Although ant-based clustering has been modified gradually, it still needs improvement in terms of its applications. The focus of our work is on the following two important problems:

- Improving the algorithm's efficiency
  Ant-based clustering can be implemented through the parallel computing of each ant [7], which may lead to the development of an efficient algorithm. However, it is not highly efficient because of the randomness in the algorithm. Initially, the objects are randomly projected onto the toroidal grid; thus, the similarities of the objects in a local neighbourhood are very low. Therefore, the objects are easily picked up but not easily dropped by the ants. It takes long time to go from the inception of the algorithm to the moment when the rough clusters are created. Commonly, 100,000 iterations are needed for ant-based clustering algorithms [2].
- Improving the algorithm's clustering quality
  In essence, ant-based clustering algorithms are distance-based because the similarity of the objects is computed by Euclidean distance or Cosine distance. Just like other distance-based clustering algorithms, they are effective for datasets with an ellipsoidal or Gaussian distribution. If the separation boundaries between clusters are nonlinear, however, the algorithms will fail. An alternative approach to solving this problem is kernel mapping, which transforms the data into a high-dimensional feature space and then performs the clustering in the feature space.

Kernel-based clustering was proposed by Mark [14]. Its integration with *K*-means, fuzzy *K*-means, SOM, and support vector machines has been shown to be effective in improving clustering quality [11].

In this paper, we incorporated the kernel method into ant-based clustering and created the novel ant-based clustering with the kernel method (ACK). The applications of kernels in ACK are shown in two respects. First, kernel principal component analysis (KPCA) is used to modify the initial projection of all objects. Second, the Euclidean distance in the feature space is applied as a measure of the similarity between the objects. These two applications are geared toward solving the problems mentioned above. Compared with general ant-based clustering, ACK can create better clustering results in some datasets with non-Gaussian distribution. Moreover, the clustering quality and efficiency are greatly improved.

The paper is organized as follows: Section 2 describes the basics of the ant-based clustering algorithm. Section 3 introduces kernel-based clustering. Section 4 proposes the novel ant-based clustering algorithm using the kernel method. Section 5 compares the proposed algorithm with other clustering algorithms. Finally, Section 6 gives the conclusions and discusses future work.

2. The ant-based clustering algorithm

The algorithm introduced by Lumer and Faieta [24] represents the basic ant-based clustering method. The LF algorithm projects the objects into a two-dimensional plane. Assume that an ant is located at site r at cycle t and finds an object o_i at that site. The local similarity measure of o_i is given by

$$f(o_i) = \begin{cases} \frac{1}{s^2} \sum_{o_i \in L(o_i,r)} \left[1 - \frac{d(o_i,o_j)}{\alpha}\right], & \text{when } f > 0 \\ 0 \quad \text{otherwise} \end{cases}, \tag{1}$$

where $L(o_i, r)$ denotes the local neighbourhood surrounding the site r, which is often a square of size $s \times s (s = 2r + 1)$. $d(o_i, o_j)$ is the distance between two objects; typically, Euclidean distance is used. α is a factor that defines the scale for dissimilarity. Choosing α to be too small prevents the formation of clusters; on the other hand, choosing α to be too large results in the fusion of individual clusters [15–17,24,37,38].

The probability that an ant will pick up or drop the object is

$$P_p(o_i) = \left(\frac{k_1}{k_1 + f(o_i)}\right)^2, \tag{2}$$

$$P_d(o_i) = \begin{cases} 2f(o_i) & \text{when } f(o_i) < k_2 \\ 1 & \text{when } f(o_i) \geqslant k_2 \end{cases}, \tag{3}$$

where k_1, k_2 are two constants. A high-level description of the LF algorithm is presented in Table 1.

A number of modifications have been introduced to the basic LF algorithm to improve clustering quality and convergence speed, which can be summarized as

- Using inhomogeneous ants that move at different speeds [15,37,40,41]. Then, $f(o_i)$ becomes:

$$f(o_i) = \begin{cases} \frac{1}{s^2} \sum_{o_i \in L(o_i,r)} \left[1 - \frac{d(o_i,o_j)}{\alpha(1+(v-1)/v_{\max})}\right], & \text{when } f > 0 \\ 0 \quad \text{otherwise} \end{cases}, \tag{4}$$

where v denotes the speed of the ant and $v_{\max}$ is the maximum speed. Fast-moving ants form rough clusters on a large scale, while slow ants group objects on a smaller scale by dropping objects with more accuracy.

Table 1
The pseudo-code of the LF algorithm.

Algorithm: LF Algorithm

```
/* initialization */
Randomly scatter the objects on the toroidal grid
Randomly place the ants on the toroidal grid
Initialize all parameters: r1, tmax, α, k1, k2
/* main loop*/
for t = 1 to tmax do
  for all ants do
    if(ant unladen) and (grid occupied by object oi) then
      compute f(oi) and Pp(oi)
      draw a random real number p ∈ (0,1)
        if(p ⩽ Pp(oi) then
          pick up object oi
        end if
      else
        if(ant carrying object oi) and (grid empty)then
            compute f(oi) and Pd(oi)
            draw a random real number p ∈ (0,1)
            if (p ⩽ Pd(oi) then
            drop object oi
            end if
        end if
      end if
      Move to randomly selected neighbouring grid not occupied by other ants
  end for
adjust r, α, k1,k2
end for
Output locations of all objects;
```

- Using ants with short-term memory [2,16,17,36].
  An ant stores two objects in its short-term memory: visited places and visited objects. Short-term memory can help ants to drop the object quickly to improve the algorithm speed.
- Making some important parameters adaptive, such as the neighbourhood radius r and the scale factor α [2,14,15,36,39].
  As mentioned above, ant-based clustering should avoid complex parameter settings to simplify its use.
- Designing the ant as an agent [39,41,42].
  All of these modifications are intended to improve the performance of the basic ant-based clustering and make it more applicable. In the following sections, we will show how ant-based clustering is integrated with the kernel method to improve the algorithm's performance. Before the introduction of ACK, kernel-based clustering is introduced.

3. Kernel-based clustering

3.1. Mercer kernels

Consider a smooth, continuous nonlinear mapping Φ from the data space to the feature space F:

$$\Phi : R^N \to F.$$

Then, the data samples in the input space $x_k \in R^N (k = 1,2,\ldots,l)$ are mapped into $\Phi(x_1)$, $\Phi(x_2),\ldots,\Phi(x_l)$. Note that the dot product in the feature space can be computed using Mercer kernels in the input space:

$$K(x_i, x_j) = \Phi(x_i) \cdot \Phi(x_j). \tag{5}$$

In other words, by employing a specific kernel function, the dot product that it returns implicitly defines the nonlinear mapping Φ to the feature space [14].

Commonly used kernel functions include the following:

Polynomial kernel: $K(x,y) = (x \cdot y + 1)^d$.

Gaussian kernel: $K(x,y) = \exp\left(-\frac{\|x-y\|^2}{2\sigma^2}\right)$.

Neural network type kernel: $K(x,y) = \tanh((x \cdot y) + b)$.

3.2. Kernel-based clustering

Supposing data samples x and y, the Euclidean distance in the input space is

$$d(x,y) = \sqrt{\|x-y\|^2}. \tag{6}$$

After the samples are mapped into the feature space, the Euclidean distance between $\Phi(x)$ and $\Phi(y)$ in the feature space becomes:

$$d_F(x,y) = \sqrt{\|\Phi(x) - \Phi(y)\|^2} = \sqrt{\Phi(x)\cdot\Phi(x) - 2\Phi(x)\cdot\Phi(y) + \Phi(y)\cdot\Phi(y)}. \tag{7}$$

Based on (5), (7) can be computed as

$$d_F(x,y) = \sqrt{K(x,x) - 2K(x,y) + K(y,y)}. \tag{8}$$

Eq. (8) can replace (6) as the similarity measure in the clustering algorithms. Based on this principle, kernel-based clustering transforms the data into a high-dimensional feature space and then performs clustering in the feature space. The application of kernels in K-means, fuzzy K-means, and evolution algorithms has been proven to be effective in terms of improving clustering performance [8,11].

4. The novel ant-based clustering with the kernel method (ACK)

In this paper, we applied the kernel method in ant-based clustering. With respect to the problems mentioned in Section 1, kernels are applied in two ways: first, all objects are preprocessed by KPCA then projected onto a plane based on the first two kernel principals; second, the Euclidean distance of objects in the feature space is used as a similarity measure, which means that ant-based clustering is performed in the feature space after kernel mapping.

4.1. Projection of the objects based on KPCA

In general, in ant-based clustering algorithms, the objects are randomly projected onto the plane. As a result, that one pattern corresponds randomly with a pair of coordinates. This random projection leads to few similarities between the

objects in the local neighbourhood at the beginning of the algorithm. Therefore, the objects are easily picked up but not easily dropped by the ants. It takes a long time for an object to be similar to nearby objects from the inception of the algorithm.

To reduce the influence of randomness in this stage, we have suggested a modified projection based on principal component analysis (PCA) [45]. The objects are preprocessed by PCA; then, the first two principal components (PCs) remain, and these are processed. According to the principles of PCA, the first two PCs can retain most of the information about the original objects. If the objects are projected with the two processed PCs, the objects that are close to each other in the R^n space will be close to each other in the projection plane. As a result, the object is very similar to others in its local neighbourhood at the beginning of the algorithm. This result is similar to that of ant-based clustering algorithm after many cycles; thus, the running time is reduced significantly.

In this paper, we applied KPCA, which was suggested by Bernhard [32], to replace PCA. Compared with linear PCA, KPCA can extract features that are more useful for classification. After the first two kernel principal components (KPCs) are obtained, they also need to be processed as the projection coordinates. The processing methods are as follows:

- *Enlarging:* If the KPCs are very small, they are multiplied by a large number to be distinguished easily.
- *Rounding:* Obtaining the integer.
- *Shifting:* Finding the minimums of the first KPC and the second KPC. The last processed values are obtained by subtracting the minimums from the pair of KPCs. The aim of this processing is to distribute the coordinates of the objects in the first quadrant.

The two processed KPCs are taken as the projection coordinates of the objects, where the first KPC is the x-coordinate and the second KPC is the y-coordinate:

$$\begin{cases} x_{o_i}(t=1) = \mathrm{Pre}(KPC1_{o_i}) \\ y_{o_i}(t=1) = \mathrm{Pre}(KPC2_{o_i}) \end{cases}, \tag{9}$$

where t is the number of cycles; $\Pr e(x)$ is the function describing the whole preprocess; and $KPC1_{o_i}$ and $KPC2_{o_i}$ are the first and second KPC of the object o_i, respectively.

4.2. The ant movement model operating in the feature space

In the above-mentioned ant-based clustering algorithms, the objects are picked up or dropped by the virtual ants. The ants and the objects are separate. In this study, we applied the ant movement model suggested by Xu et al. [39], where each object is looked at as an ant. Each ant has two states: movement and sleep. If the ant finds that a location is suitable for it to rest, it will stop moving and enter a sleeping state; otherwise, it will continue to move to another place. The fitness of the local neighbourhood is computed as (10), which is similar to (1) in Section 2.

$$f(o_i) = \begin{cases} \frac{1}{s^2} \sum\limits_{o_j \in L(o_i,r)} \left[1 - \frac{d(o_i,o_j)}{\alpha_i}\right], & \text{when } f > 0 \\ 0 \quad \text{otherwise} \end{cases}, \tag{10}$$

where $\alpha_i = \frac{1}{N-1}\sum_{j=1}^{N} d(o_i, o_j)$ is the average distance between o_i and other objects.

If we apply a kernel function to map the objects into the feature space, then the clustering can be performed according to the similarities of the objects in the feature space. The fitness becomes:

$$f(o_i) = \begin{cases} \frac{1}{s^2} \sum\limits_{o_j \in L(o_i,r)} \left[1 - \frac{d_F(o_i,o_j)}{\alpha_i}\right], & \text{when } f > 0 \\ 0 \quad \text{otherwise} \end{cases}, \tag{11}$$

where $\alpha_i = \frac{1}{N-1}\sum_{j=1}^{N} d_F(o_i, o_j)$ and $d_F(o_i,o_j)$ is the distance between o_i and o_j in the feature space. Based on the introduction in Section 3.2, $d_F(o_i,o_j)$ can be obtained by the kernel function according to (12). A Gaussian kernel is applied in this study:

$$d_F(o_i, o_j) = \sqrt{K(o_i, o_i) - 2K(o_i, o_j) + K(o_j, o_j)}. \tag{12}$$

For an ant, the probability of being activated by the local neighbourhood is

$$P_a(o_i) = \left(\frac{\beta}{\beta + f(o_i)}\right)^2, \tag{13}$$

where $\beta \in R^+$ is the threshold of the ant's active fitness. Eq. (13) is also similar to (2) in Section 2. It can be seen that when $f \ll \beta$, $P_a(o_i)$ is close to 1. Thus, if the fitness of o_i is much smaller than the threshold, o_i has a high probability of being activated. The active o_i moves in the plane and searches for a more comfortable place to sleep. When $f \gg \beta$, $P_a(o_i)$ is close to 0. Therefore, o_i does not wake up and continues to sleep.

The principle of the ant movement model is the same as that in the LF algorithm [24]. The big difference is that the passive movement of objects is transformed into active movement. The adaptive adjustment of the parameters α and β is also similar in the two models; this concept will be described in Section 4.4.

4.3. The process of the ACK algorithm

Here, we give a simple description of the ACK algorithm. Initially, KPCA is first performed for the dataset. For each object, the first two KPCs are processed as the projection coordinates. After the objects are projected onto the plane, each object is viewed as an ant. All the ants are in an active state, and they move randomly around on the plane following the movement strategy. The simplest movement strategy is choosing a new location in the neighbourhood as the next destination. In each cycle, the ant will recalculate its current fitness f and probability P_a to decide whether it should continue to move or sleep. The pseudo-code of the main body of the ACK algorithm is given in Table 2.

It should be noted that KPCA and the distance between the objects in the feature space can be calculated beforehand once the kernel function is given. The distance values can be stored in the matrix; then, the remaining calculations in the main algorithm are simple matrix manipulations. This preprocessing can reduce the time of the algorithm.

4.4. Parameter setting

(1) Size of the projection plane
The size of the projection plane is set as $S \times S$, S is computed by

$$S = \mathrm{MAX}(\mathrm{Pre}(KPC1_{o_i}), \quad \mathrm{Pre}(KPC2_{o_i})i \in [1, N], \tag{14}$$

where N is the size of the data.

(2) Radius r
The radius r determines the region that the ant perceives. A larger radius means that it takes in more information but there is a higher time cost. Furthermore, a larger radius inhibits the quick formation of clusters in the initial phase. We applied a changing radius that gradually increases over cycles [16,17,39]. The initial value r_1 is 1, and the maximum $r_{\max}$ is 5; the adjustment of r is

$$r_t = \mathrm{Round}(r_1 + (r_{\max} - r_1) \cdot t/t_{\max}). \tag{15}$$

Round(x) is the function used to get the integer of x.

(3) Kernel size σ
σ is an important parameter of the Gaussian kernel. The selection of σ is one of the major questions under consideration in support vector and kernel methods. There are no good ways to solve this problem. Cross-validation and the leave-one-out technique are usually used. Sometimes, σ can be selected by experience. In this paper, we applied the try and trial method, and the value that generated the best results in KPCA was selected.

(4) Scaling parameter α
In (11), the value of α_i is a constant. Several approaches to determine the value of α have been described [39]. Here, we applied an adaptive method. Suppose that in cycle t, $\gamma_i(t)$ is the average similarity of the object o_i with other objects ino_i 's local neighbourhood. Then:

Table 2
The pseudo-code of the ACK algorithm.

```
Algorithm: ACK Algorithm
0   /* initialization */
27  All objects are placed on the toroidal grid based on KPCA
28  Initialize all parameters: r1, tmax,α, β
29  /* main loop */
30  for t = 1 to tmax do
31    for i = 1 to N do (N is the number of the objects)
32          compute f(oi) and Pa(oi)
33          draw a random real number p ∈ (0,1)
34            if (p ⩽ Pa) then
35              activate ant and move to next place
36            else
37              stay at current site and sleep
38            end if
39    end for
40  adjust r,α,β
41  end for
42  Output locations of all objects;
```

$$\gamma_i(t) = \frac{1}{n-1} \sum_{o_j \in L(o_i,r)}^{n} d_F(o_i, o_j), \tag{16}$$

where n is the number of the objects in o_i 's local neighbourhood. α_i can be computed as

$$\alpha_i(t) = \frac{1}{K} \sum_{l=t-k}^{t} \gamma_i(l), \tag{17}$$

where K ($K = 10$ in this paper) is a constant. Eq. (17) shows the α_i is the average value of γ_i K times before t. This adaptive adjusting method means that the ant has a memory of its fitness. The short memory method had been used in general ant-based clustering to remember the places or objects that the ant has visited recently.

(5) Threshold β

The value of β tends to decrease gradually in the process of clustering. We can adjust β every 100 cycles by the following equation:

$$\beta(t+1) = \begin{cases} 0.95\beta(t) \text{ if } \mathrm{Mod}(t, 100) = 0, \\ \beta(t) \text{ otherwise.} \end{cases} \tag{18}$$

5. Experimental results and analysis

5.1. Evaluation functions

The following functions are used to evaluate the performance of the ACK algorithm and other clustering algorithms.

(1) The *F*-measure (*F*)

The *F*-measure combines the ideas of precision and recall from information retrieval. The precision and recall of a cluster j (generated by the algorithm) with respect to a class i(given by the class labels of the dataset) are defined as

$$p(i,j) = n_{ij}/n_j, \tag{19}$$

$$r(i,j) = n_{ij}/n_i, \tag{20}$$

where n_{ij} is the number of elements of class i in cluster j, n_j is the number of members in cluster j, and n_i is the number of members in cluster i.

The corresponding value under the *F*-measure is

$$F(i,j) = \frac{(b^2+1) \cdot p(i,j) \cdot r(i,j)}{b^2 \cdot p(i,j) + r(i,j)}, \tag{21}$$

where we choose $b = 1$ to obtain equal weighting for $p(i,j)$ and $r(i,j)$. The overall *F*-value for the partitioning is

$$F = \sum_{i}^{k} \frac{n_i}{k} \max_j \{F(i,j)\}, \tag{22}$$

where k is the total number the clusters. F is limited to the interval $[0,1]$ and should be maximized.

(2) The Dunn Index (*DI*)

The Dunn Index determines the minimum ratio between inter-cluster distance and cluster diameter for a given partitioning. It captures the notion that, in a good clustering solution, data elements within one cluster should be much closer to each other than to elements within different clusters. It is defined as

$$DI = \min_{c,d \in C} \left[\frac{\delta(\mu_c, \mu_d)}{\max_{e \in C}[\mathrm{diam}(e)]} \right], \tag{23}$$

where C is the set of all clusters, the diameter $\mathrm{diam}(c)$ is computed as the maximum intra-cluster distance, and $\delta(\mu_c, \mu_d)$ is the distance between the centroids of clusters c and d. Cosine distance is applied in this study, and *DI* is maximized.

(3) The inner cluster variance (*ICV*)

The inner cluster variance (*ICV*) computes the sum of squared deviations between all data items and their associated cluster centre, which reflects that data elements within the individual cluster must be similar. It is given by

$$ICV = \sum_{c \in C} \sum_{i \in c} \delta(i, \mu_c)^2, \tag{24}$$

where C is the set of all clusters, μ_c is the centroid of cluster c, and $\delta(i, \mu_c)$ is the distance function employed to compute the deviation between each data item i and its associated cluster centre. Cosine distance is applied in this study, and *ICV* is minimized.

(4) The error rate (*ER*)
The error rate is computed as

$$ER = n_{\text{error}}/N, \tag{25}$$

where n_{error} is the number of objects that are clustered falsely, and N is the total size of the dataset. This function requires the class label to be known previously.

(5) Time cost (*T*)
All algorithms are performed on an Intel core E7200 2.53 GHz personal computer. ATTA is programmed in C++ and executed in the Linux operating system, but all other algorithms are programmed in MATLAB and executed in the Windows operating system.
All results presented through the evaluation functions were averaged over 10 runs.

5.2. Experimental data

Ten datasets, five synthetic and five real, were used to assess the algorithms. Some of these datasets are benchmarks that have been widely applied in ant-based and kernel-based clustering. The datasets are briefly introduced in Table 3.

Square: The Square dataset has been used in many ant-based clustering algorithms. The dataset is two-dimensional and consists of four clusters arranged as a square. The data are generated according to a normal distribution $N(u,\sigma^2)$. The normal distributions of the four clusters in our study are as follows: ($N(-5,2)$, $N(-5,2)$), ($N(5,2)$, $N(5,2)$), ($N(-5,2)$, $N(5,2)$), and ($N(5,2)$, $N(-5,2)$).
2D3C: This dataset is a variation of the Square. It consists of three clusters: two are normal distributions with different standard deviations, and one is a uniform distribution. The densities of the three clusters are different.
Ring: This dataset is generated by two distributions: an isotropic Gaussian and a uniform "Ring" distribution. A total of 100 data points were drawn for each distribution.
Line: This dataset is composed of two clusters: one is Gaussian and one is linear with two parts.
Moon: This dataset includes two clusters of data with a valley structure. It is often used to test clustering algorithms such as spectral clustering and manifold clustering.

The real datasets Iris, Wine, Wisconsin, Zoo, and Yeast all come from the database of University of California Irvine (UCI) for machine learning [46]; these datasets are often used to test the performance of all kinds of algorithms.

All the datasets should be preprocessed using the following steps: the data vectors are normalized in each dimension; then, the degree of similarity is computed using Euclidean distance and normalized to lie within the interval [0,1].

5.3. Comparison results and discussion

The ACK algorithm was compared with the following algorithms:

- The classic clustering algorithm
  - *K*-means
- The classic kernel-based algorithm
  - Kernel-based *K*-means (K*K*-means)
- Ant-based clustering algorithms
  - LF algorithm [5,24]
  - ATTA [18], which represents the latest modified algorithm of ant-based clustering
  - Ant-based clustering with PCA (ACP) [45], which is an early version of our research. The initial projection of the objects is modified by PCA, and ant-based clustering is performed in the input space. The other parts are the same as ACK

Table 3
The datasets used for comparing ACK with other clustering algorithms (*C* is the number of the clusters, *D* is the dimensionality, *N* is the total number of the data items, N_i is the number of items of cluster *i*).

Name	*C*	*D*	*N*	N_i	References
Square	4	2	400	4×100	[2,16,17]
2D3C	3	2	300	3×100	[16]
Ring	3	2	200	2×100	[11,14,31]
Line	2	2	200	2×100	[35]
Moon	2	2	210	2×105	[3]
Iris	3	4	150	3×50	[2,5,14,16,31,36,39,41]
Wine	3	13	178	59, 71, 48	[2,5,31,36,39]
Wisconsin	2	9	699	458, 241	[16,18]
Zoo	7	16	101	41, 20, 5, 13, 4, 8, 10	[2,18]
Yeast	10	8	1484	463, 429, 244, 163, 51, 44, 37, 30, 20, 5	[5,16,39]

- ACP-F: The initial projection of the objects is modified by PCA, and ant-based clustering is performed in the feature space. The other parts are the same as ACK
- ACK-I: The initial projection of the objects is modified by KPCA, and ant-based clustering is performed in the input space. The other parts are the same as ACK

The design of ACP-F and ACK-I is used to compare PCA with KPCA and the clustering in the input space with that in the feature space. The comparison results are shown in Tables 4 and 5.

5.3.1. Summary of the algorithm performance using synthetic and real datasets

For most datasets, ACK outperforms other algorithms in the clustering quality shown by the *F*-value, *ICV*, and *ER*, but it is not the most efficient. K*K*-means shows a relative good performance in quality and efficiency; its improved performance in comparison with *K*-means proves that the application of kernels is effective. Beyond ACK and K*K*-means, ATTA also exhibits good performance, especially in terms of *DI* and time cost. As for some of the complex datasets, such as Zoo and Yeast, none of the clustering algorithms can get satisfactory results.

5.3.2. Time cost

It can be seen that *K*-means and K*K*-means are highly efficient, but these two algorithms require a priori knowledge of the number of clusters. In our experiment, they were run by being given the correct number of clusters.

For ant-based clustering, the time cost of LF is quite large because the ants move randomly and spend a lot of time finding proper places to drop or pick up objects. Compared with LF, ACP, ACP-F, ACK-I and ACK are more efficient, because the random projections of the patterns are modified. ATTA is the most efficient of all the algorithms, but we should note that ATTA's code is written in C++ and the program runs in the Linux operation system (because there are several modifications in ATTA, it is difficult for us to implement ATTA in our code). The program was downloaded from Handl's web site [47]), which saves time compared to MATLAB code in the Windows operating system. Although ATTA is highly efficient, ACK outperforms it in clustering quality, as shown by the *F*-value, *ICV*, and *ER* in most datasets, because ACK performs clustering in the feature space after kernel mapping.

Table 4
Comparison of several algorithms using synthetic datasets (the best result is indicated by bold entity).

	K-means	KK-means	LF	ATTA	ACP	ACP-F	ACK-I	ACK
Square								
F	0.983	0.984	0.894	0.980	0.964	0.978	0.968	**0.989**
ICV	0.357	0.352	0.393	0.376	0.382	0.372	0.380	**0.296**
DI	3.702	3.717	2.896	3.654	3.601	3.650	3.648	**3.926**
ER	0.99	0.99	2.05	1.02	1.85	1.06	1.23	**0.97**
T (s)	25.16	28.78	450.72	**5.14**	332.86	320.25	220.06	204.78
2D3C								
F	0.982	**0.985**	0.978	0.981	0.980	0.980	0.978	0.983
ICV	0.447	**0.439**	0.496	0.453	0.489	0.467	0.451	0.448
DI	2.477	**2.478**	2.052	2.228	2.149	2.152	2.127	2.426
ER	0.94	**0.93**	3.47	1.12	1.12	0.94	1.13	0.95
T (s)	20.32	20.88	386.67	**4.98**	189.33	191.36	170.23	164.48
Ring								
F	0.664	0.893	0.842	0.843	0.846	0.902	0.889	**0.982**
ICV	1.162	1.047	1.055	1.067	1.084	1.023	1.053	**0.966**
DI	1.349	1.714	1.478	1.895	1.712	**1.903**	1.715	1.900
ER	19.65	7.02	12.67	8.48	8.05	4.33	6.89	**3.06**
T (s)	25.26	27.69	238.45	**8.46**	201.89	210.46	101.38	85.66
Line								
F	0.637	0.818	0.593	0.602	0.615	0.794	0.719	**0.826**
ICV	1.089	0.994	1.179	1.168	1.044	1.105	1.038	**0.965**
DI	1.252	1.297	0.973	1.058	1.024	1.396	1.377	**1.489**
ER	15.45	5.54	17.23	15.78	16.10	8.45	11.28	**5.42**
T (s)	23.35	23.46	296.46	**7.97**	194.58	220.72	177.80	163.26
Moon								
F	0.794	**0.890**	0.748	0.838	0.829	0.864	0.833	0.882
ICV	0.829	**0.792**	0.910	0.804	0.813	0.793	0.806	0.795
DI	2.165	2.454	2.044	**2.466**	2.389	2.430	2.391	2.453
ER	9.05	**6.72**	11.30	9.47	10.58	7.83	8.49	7.43
T (s)	30.10	30.12	199.51	**7.62**	187.24	196.84	120.28	101.22

Table 5
Comparison of several algorithms using real datasets* (the best result is indicated by bold entity).

	K-means	KK-means	LF	ATTA	ACP	ACP-F	ACK-I	ACK
Iris								
F	0.822	0.830	0.772	0.818	0.797	0.831	0.804	**0.835**
ICV	0.965	0.711	0.929	0.824	0.715	0.706	0.867	**0.683**
DI	2.669	2.686	2.118	**2.923**	2.446	2.673	2.502	2.898
ER	10.72	8.02	14.49	12.64	12.88	8.74	10.44	**7.45**
T (s)	30.75	31.66	186.15	**3.30**	150.84	130.29	96.78	80.42
Wine								
F	0.813	0.859	0.856	0.855	0.842	0.861	0.845	**0.868**
ICV	2.672	2.408	2.872	2.493	2.445	2.402	2.436	**2.332**
DI	1.932	3.927	2.034	**4.242**	3.986	3.994	3.986	4.197
ER	5.24	3.14	3.90	3.85	4.12	3.08	4.09	**3.02**
T (s)	30.10	31.25	199.51	**4.25**	187.24	184.38	134.06	101.22
WI								
F	0.956	0.960	0.874	0.968	0.966	0.968	**0.972**	**0.972**
ICV	2.009	1.894	2.052	1.613	1.748	1.510	1.332	**1.032**
DI	4.155	4.872	4.963	**5.488**	5.029	5.443	5.435	5.428
ER	4.24	4.20	6.03	4.08	4.15	4.12	3.89	**3.42**
T (s)	33.64	46.55	356.65	**10.27**	268.21	256.46	289.35	252.95
Zoo								
F	0.774	0.804	0.785	**0.825**	0.789	0.816	0.793	0.818
ICV	2.531	2.525	2.507	**2.484**	2.550	2.492	2.502	2.492
DI	1.2836	1.386	1.147	1.396	1.384	1.559	1.147	**1.562**
ER	23.40	14.39	21.98	**11.27**	20.46	12.90	18.38	12.83
T (s)	10.58	10.69	166.65	**6.62**	78.44	86.22	79.40	80.95
Yeast								
F	0.448	**0.451**	0.435	0.439	0.442	0.446	0.445	0.446
ICV	1.650	**1.633**	1.733	1.882	1.894	1.742	1.744	1.744
DI	0.8705	1.920	1.543	**1.956**	1.585	1.852	1.845	1.847
ER	52.47	**50.69**	61.08	57.14	54.20	53.38	51.84	52.63
T (s)	40.58	64.70	600.54	**10.45**	312.43	412.97	290.62	302.75

* Wisconsin is abbreviated as "WI" to save table space.

5.3.3. KPCA vs. PCA

Fig. 1 shows the initial data, and the projections of the objects based on PCA and KPCA of the synthetic datasets. Fig. 2 shows the initial projections of the objects and the final clustering results of the real datasets. Some conclusions can be drawn from Figs. 1 and 2.

- First, for most datasets, the projections based on PCA and KPCA can create rough clusters. Especially for the synthetic datasets Square, 2D3C, and Ring, the initial projections clearly create clusters, and no further clustering is required.
- Projection based on KPCA is superior to that based on PCA. The non-linearly separable objects in PCA projection can be linearly separable in KPCA projection, which is shown clearly in the Ring, Line, Moon, Iris, Wine, and Wisconsin datasets.
- The rough clusters after the projections can provide the basis for the following fine clustering and reduce the time cost greatly. The time is reduced even if the rough clusters cannot be created clearly, such as in the dataset Yeast, which is the dataset with the worst results. The modified projections can also create some small clusters, which then act as seeds that collect other objects to create larger clusters.

Comparing ACP with ACK-I and ACP-F with ACK in Tables 4 and 5, we can see:

- In the same clustering space (whether the input space or the feature space), KPCA takes less time than PCA for most data sets, and in particular for the Ring, Line, Moon, Iris, and Wine datasets.
- For some datasets, such as Square, 2D3C, Iris, and Wine, KPCA does not exhibit a significant improvement in clustering quality compared with PCA. Thus, KPCA plays a more important role in terms of time cost than clustering quality.

5.3.4. Feature space vs. input space

Through comparing *K*-means with K*K*-means, ACP with ACP-F, ACK-I with ACK in Tables 4 and 5, some conclusions can be drawn.

- The performance improvement of K*K*-means compared with *K*-means indicates that the application of kernels is effective.
- For ant-based clustering, the clustering quality in the feature space is better than in the input space (whether PCA or KPCA). The improvement is obvious in the Ring, Line, Moon, Iris, and Wine datasets.

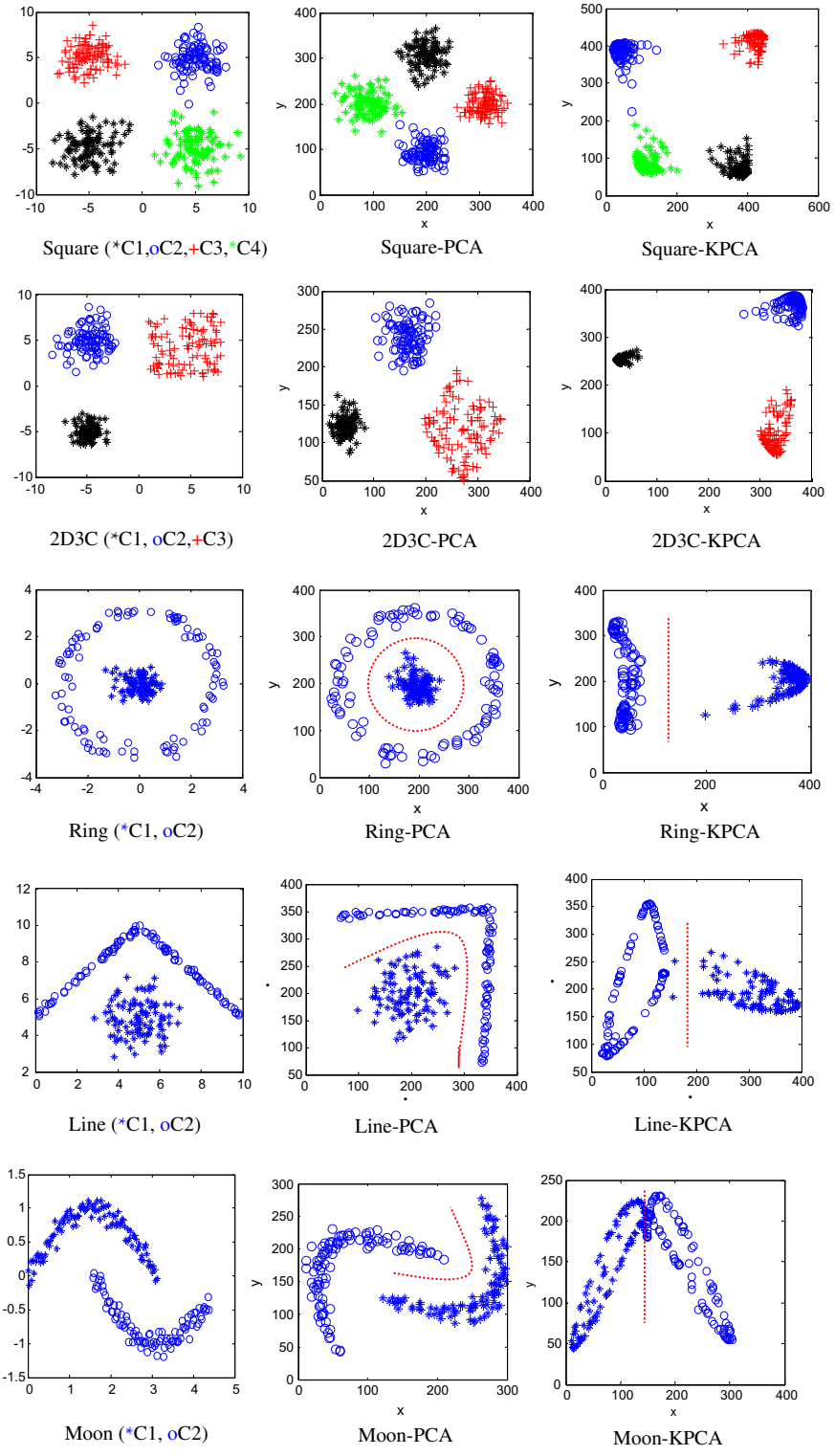

Fig. 1. The initial data and projections of the synthetic datasets.

- As for the 2D3C, Ring, Line, Moon, Zoo, and Yeast datasets, the time cost of ACP-F is higher than that of ACP, which indicates that the accurate clustering results generate time expense. Thus, clustering in the feature space plays a more important role in the clustering quality than in the time cost.

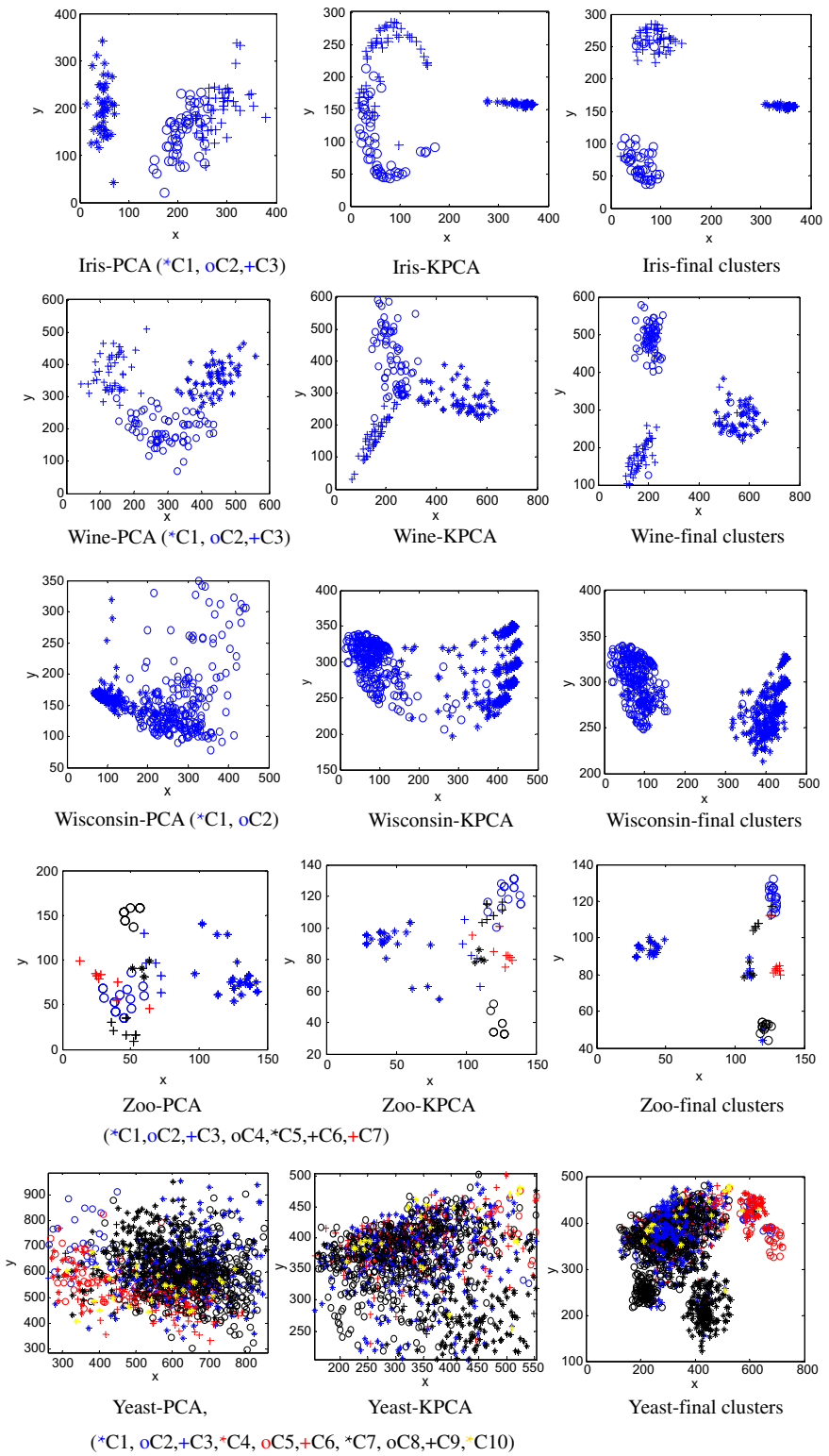

Fig. 2. The projections and final clustering results of the real datasets.

For each dataset, Fig. 3 shows the ratio values of the distances between the centres in the feature space and those in the input space. Because the distance matrix is symmetrical and the elements in the diagonal are zeros, only one half of the

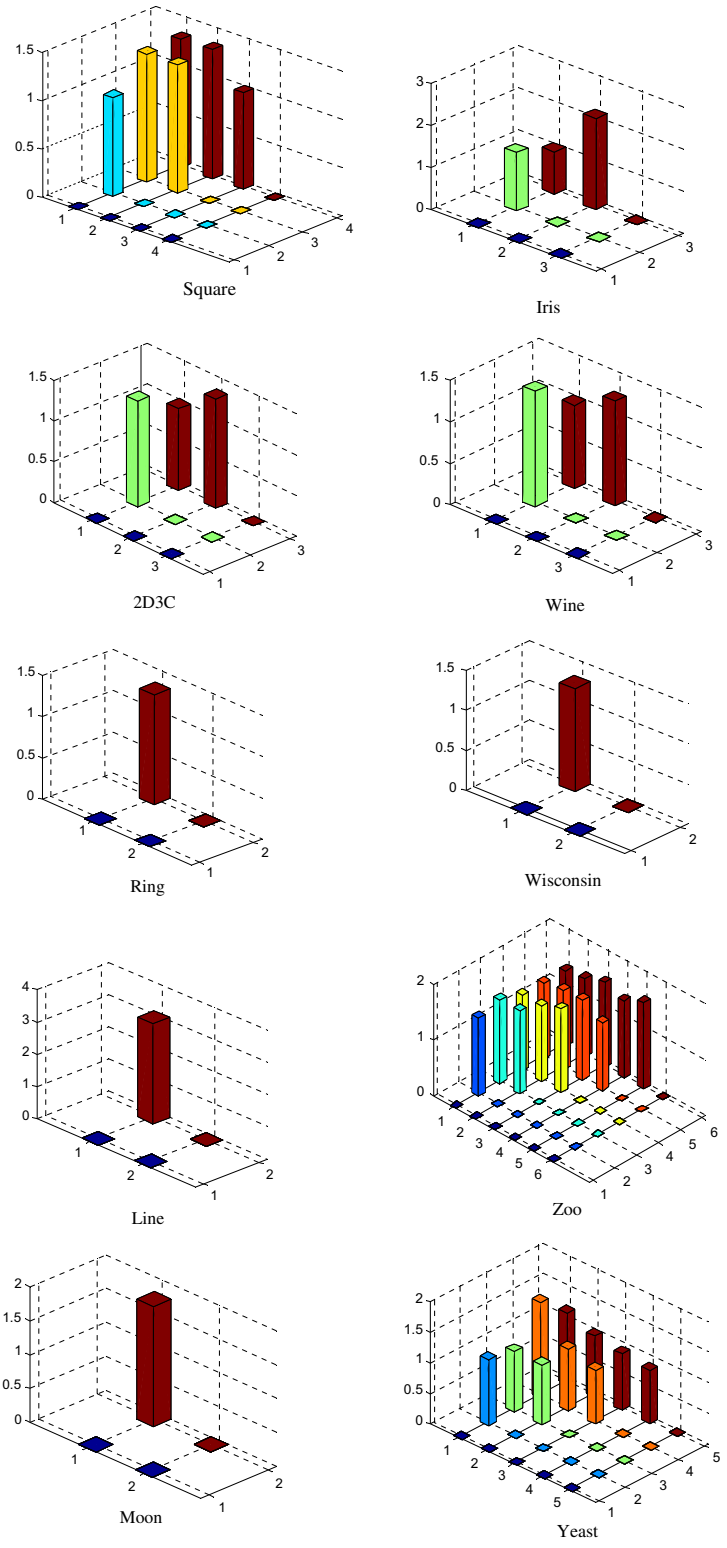

Fig. 3. Ratio of the distances between the centers in the feature space and those in the input space.

matrix is shown by the histograms. It can be seen from Fig. 3 that, for most datasets, the ratio values are larger than one, which means that the clusters are more separable in the feature space than in the input space. Thus, the clustering in the feature space more easily returns better results.

In general, the performance of ACK is greatly improved by applying KPCA and clustering in the feature space. KPCA plays an important role in time saving; at the same time, clustering in the feature space increases clustering quality. For some non-linearly separable datasets, such as Ring, Line, and Moon, conventional ant-based clustering such as LF, ATTA, and ACP cannot generate satisfactory results, while ACK performs very well. As for the real datasets Iris, Wine, and Wisconsin, the clustering quality of ACK represented by the *F*-value, *ICV*, and *ER*, is also better than in conventional algorithms. For the Zoo and Yeast datasets, no ant-based clustering can generate the correct number of clusters. In most cases, the numbers created are 6 for Zoo and 5 for Yeast (the actual numbers are 7 and 10, respectively). The performance of ant-based clustering still needs to be improved for complex datasets.

6. Conclusion

A novel ant-based clustering algorithm integrated with the kernel method was proposed in this paper. The algorithm inherits some advantages of traditional ant-based algorithms. For example, it does not need any prior knowledge about clustering. Its clustering results are visible, and it can be performed by parallel computing. Moreover, the algorithm has some new characteristics. First, it can deal with some datasets with non-Gaussian distribution because of the incorporation of the kernel function. Second, the projection based on KPCA creates rough clusters, which reduces the running time and increases the algorithm's efficiency. Finally, performing clustering in the feature space after kernel mapping can improve clustering quality.

This paper mainly focuses on the basic process of ACK. Its application to additional datasets and improved methods to adjust the parameters requires further study.

Acknowledgements

This paper is supported by the National Natural Science Foundation of China (No. 50705054) and the Research Fund of State Key Lab of MSV, China (No. MSV-2010-02). The authors gratefully acknowledge the providers of the datasets in the UCI machine learning database for research. Moreover, the authors sincerely appreciate the reviewers for their advice and suggestions to improve the quality of the paper.

References

[1] R.M. Aliguliyev, Performance evaluation of density-based clustering methods, Information Sciences 179 (2009) 3583–3602.
[2] U. Boryczka, Finding groups in data: cluster analysis with ants, Applied Soft Computing 9 (2009) 61–70.
[3] M. Breitenbach, G.Z. Grudic, Clustering through ranking on manifolds, in: Proceedings of the 22nd International Conference on Machine Learning, vol. 119, Bonn, Germany, 2005, pp. 73–80.
[4] D. Brugger, M. Bogdan, W. Rosenstiel, Automatic clustering detection in Kohonen's SOM, IEEE Transactions on Neural Networks 19 (3) (2008) 442–459.
[5] L. Chen, L. Tu, H.J. Chen, A novel ant clustering algorithm with digraph, Lecture Notes in Computer Science, vol. 3611, Springer, Berlin/Heidelberg, 2005. pp. 1218–1228.
[6] S.C. Chu, J.F. Roddick, C.J. Su, J.S. Pan, Constrained ant colony optimization for data clustering, in: Eighth Pacific Rim International Conference on Artificial Intelligence, Lecture Notes in Artificial Intelligence, vol. 3157, Springer, Berlin/Heidelberg, 2004, pp. 534–543.
[7] S.C. Chu, J.F. Roddick, J.S. Pan, Ant colony system with communication strategies, Information Sciences 167 (1–4) (2004) 63–76.
[8] S. Das, S. Sil, Kernel-induced fuzzy clustering of image pixels with an improved different evolution algorithm, Information Sciences 180 (2010) 1237–1256.
[9] J.L. Deneubourg, S. Goss, N. Frank, The dynamics of collective sorting: robot-like ants and ant-like robots, in: Proceedings of the 1st International Conference on Simulation of Adaptive Behavior: From Animals to Animats, MIT Press/Bradford Books, Cambridge, MA, 1991, pp. 356–363.
[10] L. Duan, L. Xu, F. Guo, A local-density based spatial clustering algorithm with noise, Information Systems 32 (2007) 978–986.
[11] M. Filippone, F. Camastra, F. Masulli, S. Rovetta, A survey of kernel and spectral methods for clustering, Pattern Recognition 41 (1) (2008) 176–190.
[12] A. Ghosh, A. Halder, M. Kothari, S. Ghosh, Aggregation pheromone density based data clustering, Information Sciences 178 (2008) 2816–2831.
[13] S. Ghosh, M. Kothari, A. Halder, A. Ghosh, Use of aggregation pheromone density for image segmentation, Pattern Recognition Letters 30 (2009) 939–949.
[14] M. Girolami, Mercer kernel-based clustering in feature space, IEEE Transactions on Neural Networks 13 (2) (2002) 780–784.
[15] J. Handl, B. Meyer, Improved ant-based clustering and sorting in document retrieval interface, Lecture Notes in Computer Science, vol. 2439, Springer, Berlin/Heidelberg, 2002. pp. 913–923.
[16] J. Handl, J. Knowles, M. Dorigo, Ant-based clustering: a comparative study of its relative performance with respect to *k*-means, average link and 1D-SOM, Technical Report TR/IRIDIA/2003-24, IRIDIA, University Libre de Bruxelles, Belgium, 2003.
[17] J. Handl, J. Knowles, M. Dorigo, Strategies for the increased robustness of ant-based clustering, Lecture Notes in Artificial Intelligence, vol. 2977, Springer-Verlag, Berlin/Heidelberg, 2004. pp. 90–104.
[18] J. Handl, J. Knowles, M. Dorigo, Ant-based clustering and topographic mapping, Artificial Life 12 (1) (2004) 1–36.
[19] J. Handl, B. Meyer, Ant-based and swarm-based clustering, Swarm Intelligence 1 (2) (2007) 95–113.
[20] Z.X. Huang, Extensions to the *k*-means algorithms for clustering large data sets with categorical values, Data mining and Knowledge Discovery 2 (1998) 283–304.
[21] H. Jiang, S. Yi, J. Li, F. Yang, X. Hu, Ant clustering algorithm with *K*-harmonic means clustering, Expert Systems and Applications (2010), doi:10.1016/j.eswa.2010.06.061.
[22] T. Kanungo, D.M. Mount, N.S. Netanyahu, An efficient *k*-means clustering algorithm: analysis and implementation, IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (7) (2002) 881–892.
[23] E. Lefever, T. Fayruzov, V. Hoste, M. De Cock, Clustering web people search results using fuzzy ants, Information Sciences 180 (2010) 3192–3209.
[24] E. Lumer, B. Faieta, Diversity and adaptation in populations of clustering ants, in: Proceedings of the 3rd International Conference on Simulation of Adaptive Behavior: From Animals to Animats, MIT Press/Bradford Books, Cambridge, MA, 1994, pp. 501–508.
[25] D. Marco, B. Eric, T. Guy, Ant algorithms and stigmergy, Future Generation Computer System 16 (2000) 851–871.
[26] N. Monmarche, M. Slimane, G. Venturini G, AntClass: discovery of clusters in numeric data by a hybridization of an ant colony with the *k*-means algorithm, Internal Report No. 213, Available from: <http://www.antsearch.univ-tours.fr/public/MonSliVen99b.pdf>, 1999.

[27] T. Niknam, B. Amiri, An efficient hybrid approach based on PSO, ACO and *k*-means for cluster analysis, Applied Soft Computing 10 (2010) 183–197.
[28] M.K. Parag, O.H. Lawrence, Fuzzy ants and clustering, IEEE Transactions on Systems, Man, and Cybernetics – Part A 37 (5) (2007) 758–769.
[29] V. Ramos, J.J. Merelo, Self-organized stigmergic document maps: environment as a mechanism for context learning, in: Proceedings of the 1st Spanish Conference on Evolutionary and Bio-Inspired Algorithms, Centro Uni. De Merida, Spain, 2002, pp. 284–293.
[30] S.K. Rangarajan, V.V. Phoha, K.S. Balagani, Adaptive neural network clustering of Web users, Computer 37 (4) (2004) 34–40.
[31] S.J. Roberts, R. Everson, I. Rezek, Maximum certainty data partitioning, Pattern Recognition 33 (2000) 833–839.
[32] B. Scholkopf, A. Smola, K.R. Muller, Nonlinear component analysis as a kernel eigenvalue problem, Neural Computing 10 (5) (1998) 1299–1319.
[33] M.C. Schut, On model design for simulation of collective intelligence, Information Sciences 180 (2010) 132–155.
[34] P.S. Shelokar, V.K. Jayaraman, B.D. Kulkarni, An ant colony approach for clustering, Analytica Chimica Acta 509 (2004) 187–195.
[35] J. Tang, Z. X Chen, A.W. Fu, D.W. Cheung, Enhancing effectiveness of outlier detections for low density patterns, in: Proceedings of the 6th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining, Lecture Notes in Computer Science, vol. 2336, Springer, Berlin/Heidelberg, 2002, pp. 535–548.
[36] C.H. Tsang, S. Kwong, Ant colony clustering and feature extraction for anomaly intrusion detection, Studies in Computing Intelligence, vol. 3, Springer, Berlin/Heidelberg, 2006. pp. 101–123.
[37] B. Wu, Z.Z. Shi, A clustering algorithm based on swarm intelligence, in: International Conference on Info-Tech and Info-Net, IEEE Xplore, vol. 3, Beijing, China, 2001, pp. 58–66.
[38] B. Wu, Y. Zheng, S. Liu, Z.Z. Shi, CSIW: a document clustering algorithm based on swarm intelligence, in: Proceedings of the 2002 Congress on Evolutionary Computation, vol. 1, Honolulu, HI, USA, 2002, pp. 477–482.
[39] X.H. Xu, L. Chen, Y.X. Chen, A4C: an adaptive artificial ants clustering algorithm, in: Proceedings of the IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology, 2004, pp. 268–274.
[40] Y. Yan, M. Kamel, F. Jin, Topic discovery from document using ant-based clustering combination, Lecture Notes in Computer Science, vol. 3399, Springer, Berlin/Heidelberg, 2005. pp. 100–108.
[41] Y. Yan, M. Kamel, An aggregated clustering approach using multi-ant colonies algorithms, Pattern Recognition 39 (2006) 1278–1289.
[42] Z. Yu, O.C. Au, R. Zou, W. Yu, J. Tian, An adaptive unsupervised approach toward pixel clustering and color image segmentation, Pattern Recognition 43 (2010) 1889–1906.
[43] S.H. Yue, M.M. Wei, J.S. Wang, A general grid-clustering approach, Pattern Recognition Letters 29 (9) (2008) 1372–1384.
[44] T. Zhang, L.M. Ramakrishna, BRICH: an efficient data clustering method for very large databases, in: Proceedings of the ACM SIGMOD International Conference on Management of Data, ACM Press, New York, 1996, pp. 103–114.
[45] L. Zhang, Q.X. Cao, J. Lee, A modified clustering algorithm based on swarm intelligence, in: The First International Conference on Natural Computation, ICNC 2005, vol. 3, Changsha, China, 2005, pp. 535–542.
[46] <http://www.ics.uci.edu/~mlearn/MLRepository.html>.
[47] <http://dbkgroup.org/handl/ants/>.